GL-001

Руководство
по эксплуатации

ПРОМЫШЛЕННАЯ ЭЛЕКТРОНИКА

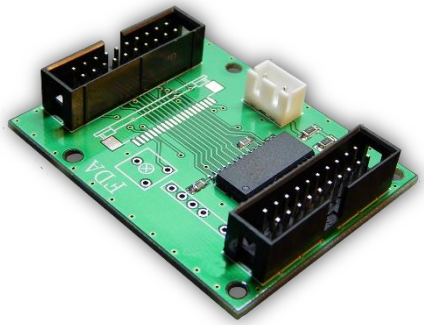
Графический контроллер

Студия разработки СпецПромДизайн
Разработка электроники и программного обеспечения ...это просто
Web: www.spd.net.ru, E-mail: info@spd.net.ru

СОДЕРЖАНИЕ

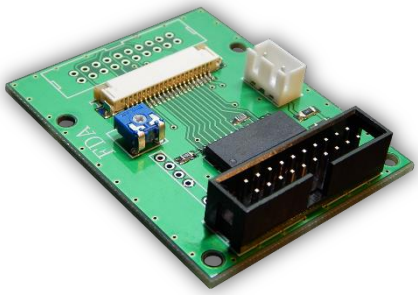
ОПИСАНИЕ

Контроллер GL-001 предназначен для разгрузки центрального микропроцессора при наличии в системе графического ЖКИ с контроллером SED1335 (S1D13305F) фирмы EPSON или RA8835A фирмы RAIO Technology Inc. Он управляется по синхронному интерфейсу SPI и позволяет рисовать линии, окружности, прямоугольники, дуги и выводить текст шрифтом двух размеров. Ядром GL-001 является микроконтроллер PIC18F26K22 фирмы Microchip®. Все алгоритмы используют целочисленную арифметику, что позволяет выполнять очень быструю отрисовку графических примитивов.



ПРИМЕНЕНИЯ

- Мобильные терминалы
- Измерительные приборы
- Пульты ввода информации



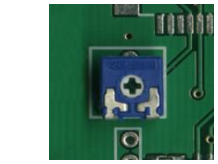
ОСОБЕННОСТИ

- Малые габариты
- Подключение посредством плоского кабеля
- 20 графических команд
- Поддержка двух страниц видеопамяти
- Управление всего по четырём проводам
- Отдельный вывод управления подсветкой ЖКИ

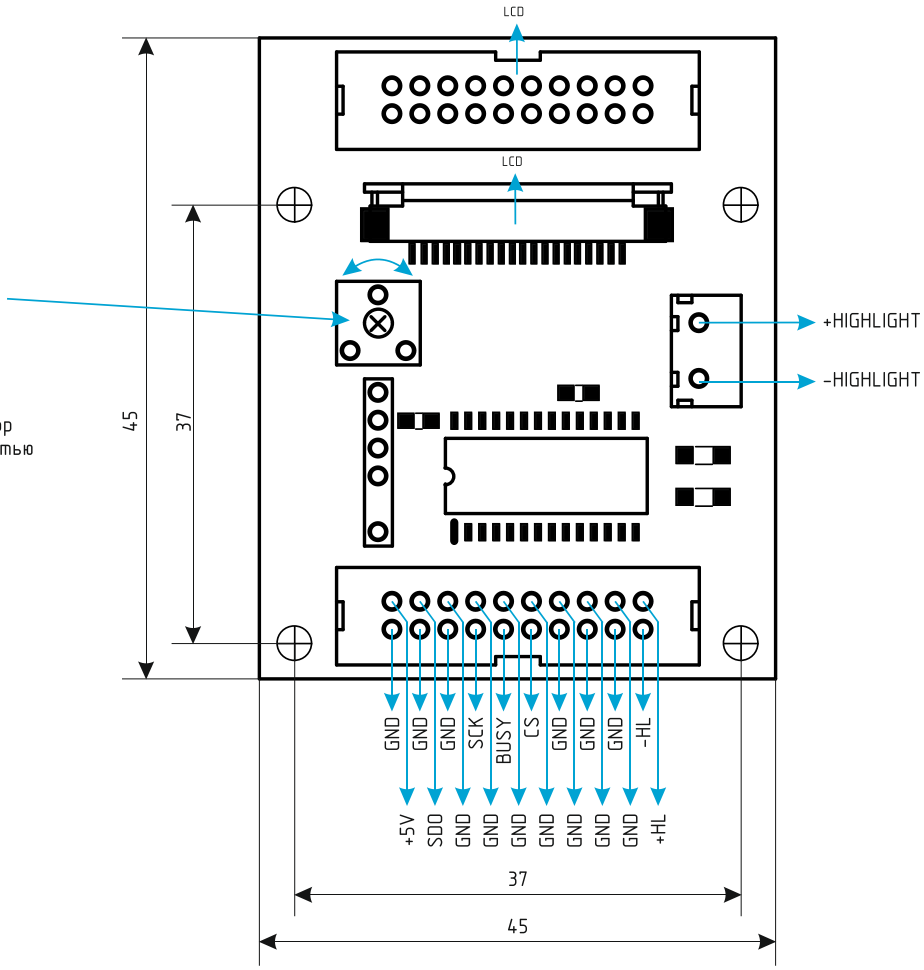
ХАРАКТЕРИСТИКИ

Напряжение питания	5 В ± 10%
Максимальный потребляемый ток	30 мА
Интерфейс	SPI, Mode 1
Скорость передачи данных	до 1 Мбит/сек
Тайм-аут приёма данных	1,5 сек
Поддерживаемое разрешение экрана ЖКИ	320 × 240
Диапазон регулировки контрастности ЖКИ	0-100%
Габаритные размеры	47 × 59 × 7 мм
Температурный диапазон работы	от –40°C до +85°C
Относительная влажность воздуха	не более 90% при +35°C

ТИПОВАЯ СХЕМА ПОДКЛЮЧЕНИЯ



Подстроечный резистор управления контрастностью



Контроллер GL-001 поддерживает два вида подключения ЖКИ: при помощи стандартного шлейфа с шагом 1.27 мм через разъём IDC-20 и при помощи гибкого плоского шлейфа с шагом 1 мм (Molex 21039-0363). Для этого на плате предусмотрено два вида разъёмов[1].

ЖКИ с подключением через разъём IDC-20 с шагом 1.27 мм уже имеют встроенный резистор для регулировки контрастности, поэтому на плате графического контроллера в данном варианте резистор не устанавливается!

Шлейф подсветки ЖКИ подключается к отдельному разъёму CWF-3 (выводы +HIGHLIGHT, -HIGHLIGHT).

Управляющие сигналы контроллера выведены на отдельный разъём типа IDC-20. Назначение его выводов следующее:

+5V – напряжение питание контроллера и ЖКИ;

-HL – отрицательный вывод питания подсветки ЖКИ;

+HL – положительный вывод питания подсветки ЖКИ. В разрыв этой линии включены резисторы, задающие ток подсветки;

[1] Конкретный тип устанавливаемого разъёма оговаривается при заказе контроллера.

GND – общий «земляной» провод платы контроллера, минус напряжения питания контроллера и ЖКИ;

SDI – вход данных;

CS – сигнал «выбор кристалла». При подачи на этот вывод лог. 0 контроллер готов к приёму команд;

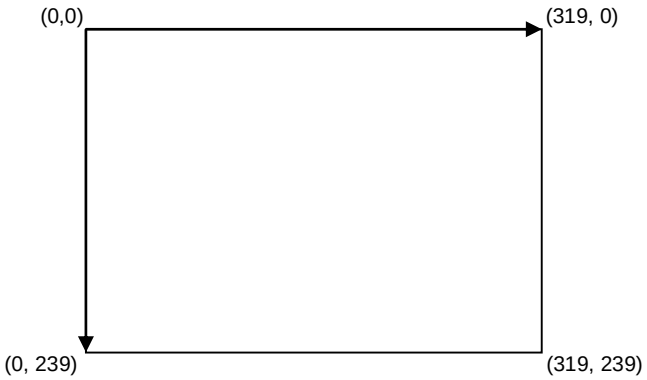
BUSY – сигнал «занято». На нём удерживается лог. 0 всё время, пока контроллер выполняет последнюю принятую команду;

SCK – сигнал тактирования.

РАСПРЕДЕЛЕНИЕ ПАМЯТИ

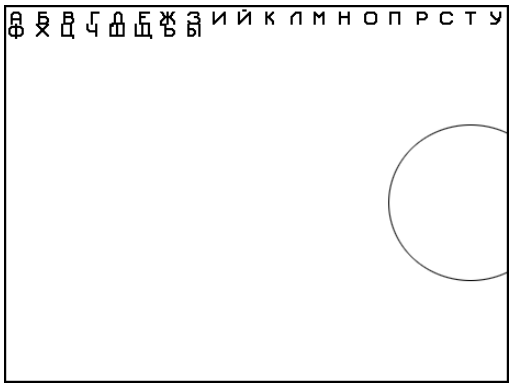
Контроллер GL-001 поддерживает две страницы видеопамяти ЖКИ размером по 9600 байт. Нулевая страница имеет смещение 3000h (12288), первая – 0000h (0). Байты видеопамяти отображаются на ЖКИ горизонтально. Подробнее о внутреннем устройстве видеопамяти можно ознакомиться в описании контроллеров SED1335 и RA8835A.

Отсчёт координат ведётся от левого верхнего края экрана:



При выводе основных графических примитивов, точки, которые не попадают в отображаемую область экрана, не выводятся. При выводе текста за пределы экрана автоматически осуществляется перенос на новую графическую строку (со сдвигом на 1 пиксель по оси Y).

В приведённом ниже примере показано отсечение окружности, не уместившей полностью на экране, и вывод строки символов «АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫ» шрифтом 16x16, начиная с координат (0, 0). Двадцать первых символов полностью уместились на экране, а остальные восемь выведены со смещением.



ИНИЦИАЛИЗАЦИЯ

После подачи питания контроллер автоматически инициализирует ЖКИ, выводит на экран номер версии своего программного обеспечения и мигающую строку «Wait…». После получения любой команды он очищает экран и выполняет эту команду.

ОБМЕН ДАННЫМИ

Контроллер является подчинённым (Slave) устройством и может только принимать данные, которые передаются в пакетном режиме. Формат пакета приведён ниже:

$$CMD,\ N_{low},\ N_{hi},\ <D_0 \ldots D_{N-1}>$$

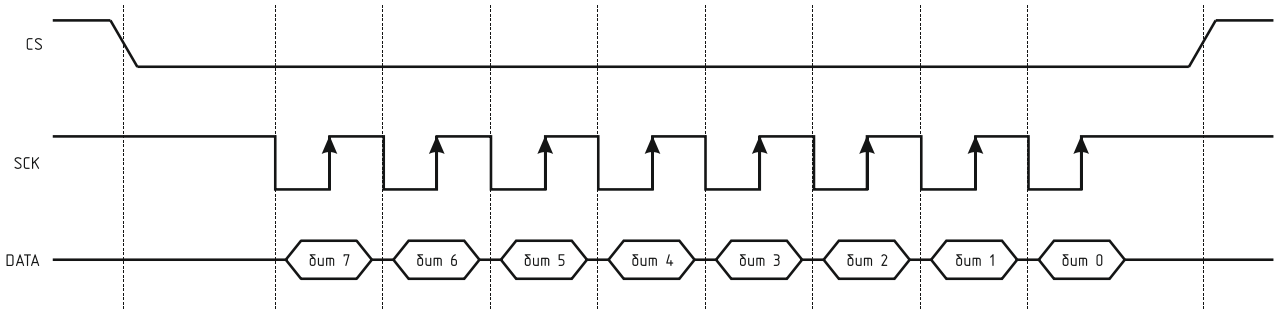
где CMD – код команды (список команд будет приведён ниже),

N_{low}, N_{hi} – младший и старший байт длины пакета,

$D_0 \ldots D_{N-1}$ – данные.

Код команды и длина пакета передаются всегда, а данных в зависимости от типа команды может и не быть. Максимальное количество байтов данных в пакете составляет 642.

Байты пакета передаются по протоколу SPI в режиме 1. Эпюры сигналов приведены ниже:



Скорость передачи данных необязательно должна быть постоянной. Она может варьироваться в пределах ±50% во время передачи. Это снижает требования к стабильности частоты центрального микропроцессора системы, в которой будет использоваться контроллер GL-001. Однако следует учитывать, что максимальная пауза между передачей соседних байтов не должна превышать значения тайм-аута, иначе графический контроллер зафиксирует его нарушение и весь пакет данных будет проигнорирован.

После успешного приёма пакета графический контроллер переводит линию $\overline{BUSY}$ в состояние лог. 0 и удерживает её до тех пор, пока не выполнится соответствующая команда.

ОПИСАНИЕ КОМАНД

Графический контроллер GL-001 позволяет исполнять 20 команд. Ниже приведено подробное описание каждой из них. При описании формата пакета данных для команд 16-битные параметры будут описываться с подстрочным знаком «$_{16}$». Если параметр может иметь знак, то перед ним будет ставиться символ «±». Все 16-ричные числа будут записываться в стиле языка Си, то есть начинаться с префикса «0x».

Как уже указывалось выше, все 16-битные данные должны передаваться младшим байтом вперёд!

Контроллер GL-001 поддерживает две страницы видеопамяти. Далее будет использоваться два термина – отображаемая страница и активная. Отображаемая страница видеопамяти это страница, содержимое которой в данный момент видно на ЖКИ. Активная страница, это страница, сна которую в данный момент времени выводится информация. Наличие двух страниц позволяет сначала построить сложное изображение на активной, не неотображаемой странице, а затем одной командой переключить страницы и мгновенно увидеть всё изображение. Кроме этого, возможен вариант объединения изображений двух видеостраниц операцией ИЛИ.

---

gcInitLCD – инициализация ЖКИ.

Описание: осуществляет аппаратный сброс ЖКИ и полную очистку всех страниц видеопамяти, текущий цвет становится чёрным, а режим наложения текста отключается.

Код команды: 0x01

Формат пакета: 0x01, 0x00, 0x00

---

gcSetLCDAddr – установка внутреннего адреса видеопамяти ЖКИ.

Описание: устанавливает начальный адрес видеопамяти перед загрузкой в неё данных. Команда используется при выводе на ЖКИ произвольных растровых изображений.

Код команды: 0x02

Формат пакета: 0x02, 0x02, 0x00, <адрес$_{16}$>

gcStartLCDWrite – перевод ЖКИ в режим записи данных в видеопамять.

Описание: переключает ЖКИ в режим записи данных в видеопамять. Команда используется при выводе растровых изображений и должна выполняться перед командой установки адреса видеопамяти gcSetLCDAddr.

Код команды: 0x03

Формат пакета: 0x03, 0x00, 0x00

---

gcWriteLCDData – запись данных в видеопамять ЖКИ по текущему адресу.

Описание: загружает данные в видеопамять ЖКИ, начиная с адреса, предварительно установленного командой gcSetLCDAddr. Команда используется при выводе растровых изображений и должна выполняться после команды перевода ЖКИ в режим записи данных в видеопамять (gcStartLCDWrite). Количество байт данных не должно превышать 640.

Код команды: 0x04

Формат пакета: 0x04, <$size_{16}$ + 2>, <$size_{16}$>, <D_{D0} …D_{N-1}>
Здесь size – объём графических данных.

---

gcSetColor – установка текущего цвета.

Описание: делает текущий цвет чёрным (1) или белым (0).

Код команды: 0x32

Формат пакета: 0x32, 0x01, 0x00, 0x01 – устанавливает чёрный цвет
0x32, 0x01, 0x00, 0x00 – устанавливает белый цвет

---

gcSetTextOverlay – установка режима наложения текста.

Описание: включает (1) или выключает (0) режим наложения текста на изображение. При включенном режиме светлые точки матрицы символов текста не стирают изображения под ними, а при выключенном – стирают. На рисунке ниже показан результат вывода текстовой строки «FDA» поверх ранее нарисованной окружности с включенным режимом наложения (слева) и с выключенным (справа).

Код команды: 0x3D

Формат пакета: 0x3D, 0x01, 0x00, 0x01 – включает режим наложения
0x3D, 0x01, 0x00, 0x00 – выключает режим наложения

---

gcSetTextJustify – установка режима выравнивания текста.

Описание: задаёт режимы выравнивания текста по осям X и Y. Для каждой оси существует три режима выравнивания: по левому краю (tjLeft), по правому краю (tjRigth) и по центру (tjCenter) – для оси X, по верхнему краю (tjTop), по нижнему краю (tjBottom) и по центру (tjCenter) – для оси Y. Допустимы любые сочетания режимов для обеих осей.
Код команды: 0x41

Формат пакета: 0x41, 0x01, 0x00, <x>, <y>
Здесь x, y – номера режимов выравнивания текста соответственно по осям X и Y.

---

gcSetActivePage – установка активной страницы видеопамяти.

Описание: задаёт активную страницу в видеопамяти.

Код команды: 0x33

Формат пакета: 0x33, 0x01, 0x00, 0x00 – делает активной страницу 0
0x33, 0x01, 0x00, 0x01 – делает активной страницу 1

---

gcSetDisplayPage – установка отображаемой страницы видеопамяти.

Описание: задаёт номер страницы видеопамяти, которая в данный момент будет отображаться на ЖКИ. Имеется возможность отображения обеих страниц операцией ИЛИ. При этом изображения будет наложены друг на друга.

Код команды: 0x40

Формат пакета: 0x40, 0x01, 0x00, 0x00 – делает отображаемой страницу 0
0x40, 0x01, 0x00, 0x01 – делает отображаемой страницу 1
0x40, 0x01, 0x00, 0x02 – делает отображаемой обе страницы

gcClear – очистка активной страницы видеопамяти.

Описание: очищает активную страницу видеопамяти, заполняя её нулями.

Код команды: 0x31

Формат пакета: 0x31, 0x00, 0x00

---

gcPutPixel – точка.

Описание: рисует точку на активной странице заданным цветом.

Код команды: 0x3F

Формат пакета: 0x3F, 0x05, 0x00, <±x_{16}>, <±y_{16}>, <Color>
Здесь x, y – координаты точки, Color – цвет точки (0 – белый цвет, больше 0 – чёрный цвет).

---

gcLine – отрезок прямой линии.

Описание: рисует отрезок прямой линии на активной странице текущим цветом.

Код команды: 0x34

Формат пакета: 0x34, 0x08, 0x00, <±$x1_{16}$>, <±$y1_{16}$>, <±$x2_{16}$>, <±$y2_{16}$>
Здесь x1, y1 – координаты начала отрезка, x2, y2 – координаты конца отрезка.

---

gcCircle – окружность.

Описание: рисует окружность на активной странице текущим цветом.

Код команды: 0x35

Формат пакета: 0x35, 0x06, 0x00, <±x_{16}>, <±y_{16}>, <±R_{16}>
Здесь x, y – координаты центра окружности, R – радиус. Если R < 0, то построение окружности не произойдёт. При R = 0 будет выведена всего одна точка по координатам x, y.

---

gcFillCircle – закрашенная окружность.

Описание: рисует закрашенную окружность на активной странице текущим цветом.

Код команды: 0x3C

Формат пакета: 0x3C, 0x06, 0x00, <±x_{16}>, <±y_{16}>, <±R_{16}>
Здесь x, y – координаты центра окружности, R – радиус. Если R < 0, то построение окружности не произойдёт. При R = 0 будет выведена всего одна точка по координатам x, y.

---

gcRectangle – прямоугольник.

Описание: рисует прямоугольник на активной странице текущим цветом.

Код команды: 0x36

Формат пакета: 0x36, 0x08, 0x00, <±$x1_{16}$>, <±$y1_{16}$>, <±$x2_{16}$>, <±$y2_{16}$>
Здесь x1, y1 – координаты левого верхнего угла, x2, y2 – координаты правого нижнего угла.

---

gcBar – закрашенный прямоугольник.

Описание: рисует закрашенный прямоугольник на активной странице текущим цветом.

Код команды: 0x37

Формат пакета: 0x37, 0x08, 0x00, <±$x1_{16}$>, <±$y1_{16}$>, <±$x2_{16}$>, <±$y2_{16}$>
Здесь x1, y1 – координаты левого верхнего угла, x2, y2 – координаты правого нижнего угла.

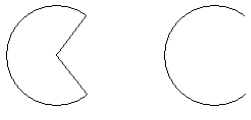
---

gcArc – дуга окружности.

Описание: рисует дугу окружности на активной странице текущим цветом.

Код команды: 0x3E

Формат пакета: 0x3E, 0x0B, 0x00, <±x_{16}>, <±y_{16}>, <±R_{16}>, <±beg_angle$_{16}$>, <±end_angle$_{16}$>, <closed>
Здесь x, y – координаты центра дуги, R – радиус, beg_angle – начальный угол, end_angle – конечный угол, closed – признак закрытой дуги (больше 0 – закрыта, 0 – открыта). Если R <0, то построение дуги не произойдёт. При R = 0 будет выведена всего одна точка по координатам x, y. На рисунке ниже показан пример закрытой (слева) и открытой (справа) дуги.



gcPrintLCD8 – вывод текстовой строки шрифтом 8x8.

Описание: рисует текстовую строку на активной странице текущим цветом и шрифтом размером 8x8. Данная команда учитывает режим наложения (см. команду gcSetTextOverlay) и режим выравнивания (см. команду gcSetTextJustify).

Код команды: 0x38

Формат пакета: 0x38, <N_{16} + 6>, <±x_{16}>, <±y_{16}>, <inv>, <s_0…s_N>
Здесь N – длина строки с учётом нулевого символа в конце (строка передаётся в формате языка Си), x, y – координаты начала строки, inv – признак инвертирования символов строки, то есть замены цвета символов на цвет фона и наоборот, s_0…s_N – символы строки, включая завершающий нулевой.

---

gcPrintLCD16 – вывод текстовой строки шрифтом 16x16.

Описание: рисует текстовую строку на активной странице текущим цветом и шрифтом размером 16x16. Данная команда учитывает режим наложения (см. команду gcSetTextOverlay) и режим выравнивания (см. команду gcSetTextJustify).

Код команды: 0x39

Формат пакета: 0x39, <N_{16} + 6>, <±x_{16}>, <±y_{16}>, <inv>, <s_0…s_N>
Здесь N – длина строки с учётом нулевого символа в конце (строка передаётся в формате языка Си), x, y – координаты начала строки, inv – признак инвертирования символов строки, то есть замены цвета символов на цвет фона и наоборот, s_0…s_N – символы строки, включая завершающий нулевой.

---

gcWindow – экранное окно.

Описание: рисует экранное окно с тенью на активной странице текущим цветом. Команда позволяет облегчить создание диалоговых окон при разработке пользовательского интерфейса для систем ввода данных.

Код команды: 0x3B

Формат пакета: 0x3B, 0x08, 0x00, <±$x1_{16}$>, <±$y1_{16}$>, <±$x2_{16}$>, <±$y2_{16}$>
Здесь x1, y1 – координаты левого верхнего угла, x2, y2 – координаты правого нижнего угла.

ПЕРЕДАЧА КОМАНД

Ниже будут приведены примеры функций передачи графических команд на языке Си. Они могут быть адаптированы к любому типу микроконтроллеров. Для этого необходимо лишь реализовать следующие функции для работы с протоколом SPI:

WaitSPI – ожидание выполнения предыдущей команды. Функция ожидает перевода линии $\overline{\text{BUSY}}$ в состояние лог. 1

PutSPI – передаёт байт по протоколу SPI

PutWordSPI – передаёт 16-битное слово по протоколу SPI (сначала передаётся младший байт, потом старший)

```
//--- Коды графических команд ---

#define gcInitLCD         0x01
#define gcSetLCDAddr      0x02
#define gcStartLCDWrite   0x03
#define gcWriteLCDData    0x04

#define gcSetColor        0x32
#define gcSetTextOverlay  0x3D
#define gcSetTextJustify  0x41
#define gcClear           0x31
#define gcSetActivePage   0x33
#define gcSetDisplayPage  0x40
#define gcPutPixel        0x3F
#define gcLine            0x34
#define gcCircle          0x35
#define gcFillCircle      0x3C
#define gcRectangle       0x36
#define gcBar             0x37
#define gcArc             0x3E
#define gcPrintLCD8       0x38
#define gcPrintLCD16      0x39
#define gcWindow          0x3B

//--- Константы цветов ---

#define BLACK             1
#define WHITE             0

//--- Константы режимов выравнивания текста ---

#define tjLeft            0
#define tjRight           1
#define tjTop             0
#define tjBottom          1
#define tjCenter          2

//--- Константы страниц видеопамяти ---

#define PAGE0             0
#define PAGE1             1
#define ALLPAGES          2
```

```
//------------------------------

void InitLCD (void)
 {
  WaitSPI();

  CS = 0;

  PutSPI(gcInitLCD);
  PutWordSPI(0);

  CS = 1;
 }

//------------------------------

void SetLCDAddr (unsigned addr)
 {
  WaitSPI();

  CS = 0;

  PutSPI(gcSetLCDAddr);
  PutWordSPI(2);
  PutWordSPI(addr);

  CS = 1;
 }

//------------------------------

void StartLCDWrite (void)
 {
  WaitSPI();

  CS = 0;

  PutSPI(gcStartLCDWrite);
  PutWordSPI(0);

  CS = 1;
 }

//------------------------------

void WriteLCDData (unsigned size, const char *data)
 {
  unsigned i;

  WaitSPI();

  CS = 0;

  PutSPI(gcWriteLCDData);
  PutWordSPI(size + 2);
  PutWordSPI(size);

  for (i = 0; i < size; i++)
   PutSPI(data [i]);

  CS = 1;
 }
```

```
//------------------------------

void SetColor (char c)
 {
  WaitSPI();

  CS = 0;

  PutSPI(gcSetColor);
  PutWordSPI(1);
  PutSPI(c);

  CS = 1;
 }

//------------------------------

void SetTextOverlay (char b)
 {
  WaitSPI();

  CS = 0;

  PutSPI(gcSetTextOverlay);
  PutWordSPI(1);
  PutSPI(b);

  CS = 1;
 }

//------------------------------

void SetTextJustify (char x, char y)
 {
  WaitSPI();

  CS = 0;

  PutSPI(gcSetTextJustify);
  PutWordSPI(2);
  PutSPI(x);
  PutSPI(y);

  CS = 1;
 }

//------------------------------

void ClearLCD (void)
 {
  WaitSPI();

  CS = 0;

  PutSPI(gcClear);
  PutWordSPI(0);

  CS = 1;
 }
```

```
//------------------------------

void SetActivePage (char p)
 {
  WaitSPI();

  CS = 0;

  PutSPI(gcSetActivePage);
  PutWordSPI(1);
  PutSPI(p);

  CS = 1;
 }

//------------------------------

void SetDisplayPage (char p)
 {
  WaitSPI();

  CS = 0;

  PutSPI(gcSetDisplayPage);
  PutWordSPI(1);
  PutSPI(p);

  CS = 1;
 }

//------------------------------

void PutPixel (int x, int y, char c)
 {
  WaitSPI();

  CS = 0;

  PutSPI(gcLine);
  PutWordSPI(5);

  PutWordSPI(x);
  PutWordSPI(y);
  PutSPI(c);

  CS = 1;
 }

//------------------------------

void Line (int x1, int y1, int x2, int y2)
 {
  WaitSPI();

  CS = 0;

  PutSPI(gcLine);

  PutWordSPI(8);

  PutWordSPI(x1);
  PutWordSPI(y1);
  PutWordSPI(x2);

  PutWordSPI(y2);
```

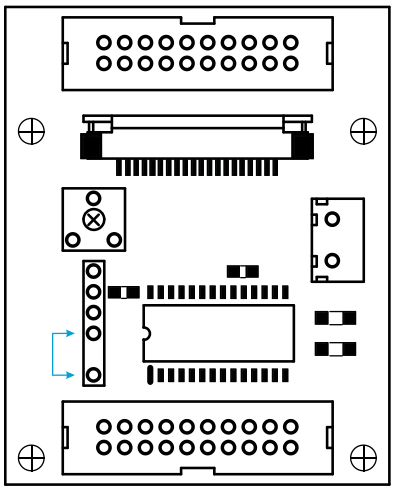
```
  CS = 1;
 }

//------------------------------

void Circle (int x, int y, int r)
 {
  WaitSPI();

  CS = 0;

  PutSPI(gcCircle);

  PutWordSPI(6);

  PutWordSPI(x);
  PutWordSPI(y);
  PutWordSPI(r);

  CS = 1;
 }

//------------------------------

void FillCircle (int x, int y, int r)
 {
  WaitSPI();

  CS = 0;

  PutSPI(gcFillCircle);

  PutWordSPI(6);

  PutWordSPI(x);
  PutWordSPI(y);
  PutWordSPI(r);

  CS = 1;
 }

//------------------------------

void Rectangle (int x1, int y1, int x2, int y2)
 {
  WaitSPI();

  CS = 0;

  PutSPI(gcRectangle);

  PutWordSPI(8);

  PutWordSPI(x1);
  PutWordSPI(y1);
  PutWordSPI(x2);
  PutWordSPI(y2);

  CS = 1;
 }
```

```
//------------------------------

void Bar (int x1, int y1, int x2, int y2)
 {
  WaitSPI();

  CS = 0;

  PutSPI(gcBar);
  PutWordSPI(8);

  PutWordSPI(x1);
  PutWordSPI(y1);
  PutWordSPI(x2);
  PutWordSPI(y2);

  CS = 1;
 }

//------------------------------

void Arc (int xc, int yc, int r, int beg_angle, int end_angle)
 {
  WaitSPI();

  CS = 0;

  PutSPI(gcArc);
  PutWordSPI(11);

  PutWordSPI(xc);
  PutWordSPI(yc);
  PutWordSPI(r);
  PutWordSPI(beg_angle);
  PutWordSPI(end_angle);
  PutSPI(closed);

  CS = 1;
 }

//------------------------------

void PrintLCD8 (int x, int y, const char *s, char inv)
{
  char i;

  WaitSPI();

  CS = 0;

  PutSPI(gcPrintLCD8);

  PutWordSPI(strlen(s) + 6);
  PutWordSPI(x);
  PutWordSPI(y);

  PutSPI(inv);

  for (i = 0; i <= strlen(s); i++)
   PutSPI(s [i]);

  CS = 1;
 }
```

```
//------------------------------

void PrintLCD16 (int x, int y, const char *s, char inv)
{
  char i;

  WaitSPI();

  CS = 0;

  PutSPI(gcPrintLCD16);

  PutWordSPI(strlen(s) + 6);
  PutWordSPI(x);
  PutWordSPI(y);

  PutSPI(inv);

  for (i = 0; i <= strlen(s); i++)
   PutSPI(s [i]);

  CS = 1;
 }

//------------------------------

void Window (int x1, int y1, int x2, int y2)
 {
  WaitSPI();

  CS = 0;

  PutSPI(gcWindow);

  PutWordSPI(8);

  PutWordSPI(x1);
  PutWordSPI(y1);
  PutWordSPI(x2);
  PutWordSPI(y2);

  CS = 1;
 }
```


ДЕМОНСТРАЦИОННЫЙ РЕЖИМ

Контроллер GL-001 позволяет самостоятельно продемонстрировать свои основные возможности. Для этого его нужно переключить в демонстрационный режим следующим образом:



- Отключить питание
- Установить перемычку на разъёме внутрисхемного программирования (см. рисунок)
- Подать питание на контроллер