ИСПОЛЬЗОВАНИЕ ЦВЕТНОГО ГРАФИЧЕСКОГО LCD ОТ ТЕЛЕФОНОВ NOKIA В МИКРОКОНТРОЛЛЕРНЫХ СИСТЕМАХ

Настоящая статья посвящена проблемам использования цветных графических жидкокристаллических индикаторов от мобильных телефонов фирмы Nokia в различных микроконтроллерных системах. Описаны основные достоинства и недостатки используемых графических индикаторов, рассмотрены основные программные отличия.

Многие разработчики современных микроконтроллерных систем стремятся к использованию малогабаритных цветных графических жидкокристаллических индикаторов. Это обусловлено тем, что наличие в микроконтроллерной системе встроенного графического индикатора существенно повышает функциональные возможности системы, позволяет отображать большие объемы самой разнообразной информации, используя при этом различные формы представления данных. Наличие в устройстве цветного индикатора повышает его эргономические параметры, позволяет минимизировать количество используемых клавиш и организовать многоуровневые системы управления режимами с помощью вложенных меню. Конечно же, при этом разработчики стремятся существенно не увеличивать стоимость устройства и аппаратные затраты.

Исходя из этих соображений, можно сформулировать основные требования к подсистеме (узлу) представления (отображения) данных для микроконтроллерных устройств:

- Подсистема отображения данных должна иметь достаточное разрешение для формирования качественного изображения или текста. Это означает, что индикатор должен иметь определенное минимальное количество активных управляемых точек изображения (пикселов) по осям X и Y. Для большинства микроконтроллерных систем считается достаточным разрешение 128x128 пикселов. Это позволяет выводить примерно 16 символов по 16 строк с размером шрифта 8x8 мм, строить на экране довольно качественные графики и диаграммы, а также отображать достаточно качественные картинки. Дальнейшее увеличение разрешения экрана приводит к значительному увеличению стоимости индикатора, размеров устройства и времени построения изображения.
- Желательно, чтобы подсистема отображения была оснащена индикатором, обеспечивающим высокое цветовое разрешение. Нормальным для современных систем считается цветовое разрешение – 256 - 4096 цветов. Конечно, наличие большего цветового разрешения приветствуется, однако формирование изображений при этом значительно замедляется.
- Желательно, чтобы индикатор имел встроенный контроллер, обеспечивающий достаточно простое управление.
- Индикатор должен иметь достаточно высокую яркость и формировать высококонтрастное изображение даже при солнечном освещении.
- Индикатор должен иметь низковольтное питание, желательно такое же, как и остальная элементная база устройства.
- Индикатор должен иметь малое энергопотребление.
- Индикатор цветной подсистемы отображения должен иметь минимальную стоимость.
- Индикатор должен быть легко доступным и широко распространенным.
- Индикатор должен иметь минимальное количество линий управления и достаточно простой интерфейс.
- Желательно, чтобы индикатор был оснащен стандартизованным соединителем (разъемом).
- Индикатор должен иметь минимально возможные размеры и вес.

Очевидно, что искать подходящий индикатор можно либо среди промышленных индикаторов, либо среди индикаторов современных мобильных устройств – телефонов, КПК, плейеров, т.к. именно эти устройства являются наиболее распространенными изделиями, содержащими цветные графические индикаторы. Даже поверхностного ознакомления с продукцией фирм, выпускающих промышленные цветные графические индикаторы (например, фирм Data Vision [1], МЭЛТ [2], Intech-LCD [3], Ampire [4], Bolymin [5] и др.) достаточно, чтобы понять, что все промышленные графические индикаторы имеют достаточно большие размеры, и

соответственно, не подходят для наших целей. Таким образом, основной областью поиска являются индикаторы для мобильных устройств.

Цветные жидкокристаллические индикаторы для карманных компьютеров и различных плейеров являются довольно дорогими и дефицитными. С другой стороны, практически каждый современный мобильный телефон оснащается цветным графическим жидкокристаллическим индикатором. Поэтому, основное внимание в ходе поисков мы уделили индикаторам мобильных телефонов.

В результате ознакомления с особенностями производства мобильных телефонов выяснилось, что все разнообразие мобильных телефонов объединяется в ряд довольно больших групп по принципу платформ – некоторых однотипных схемотехнических и конструктивных решений. Это в свою очередь означает, что в таких группах используется один и тот же конструктивно совпадающий тип индикатора.

При рассмотрении различных групп жидкокристаллических индикаторов принимались во внимание следующие основные критерии:

- количество элементов изображения по осям X и Y;
- количество градаций цветовой гаммы;
- размеры области изображения;
- наличие и тип разъема;
- габаритные размеры индикатора и шлейфа с разъемом;
- наличие в телефонах, в которых используются выбранные индикаторы, конструктивных элементов крепления индикатора к плате – крепежных рамок с амортизаторами;
- количество сигнальных линий и сложность протокола;
- номенклатура питающих напряжений;
- уровни напряжений сигнальных линий;
- доступность и стоимость индикатора.

Не утруждая читателя подробностями перебора различных вариантов индикаторов, отметим, что в результате поисков наш выбор остановился на индикаторах семейства ML019/020 для телефонов Nokia[6,7].

Выбранные индикаторы имеют следующие параметры:

- Количество элементов изображения по осям X и Y – 132x132.
- Количество градаций цветности кодируется либо одним байтом (256 цветов), либо 12 битами (4096 цветов). Разрядность цветового кодирования выбирается программно.
- Размеры области изображения составляет примерно 28x28 мм.
- Собственно жидкокристаллический индикатор закреплен в пластиковом держателе, с обратной стороны которого размещены четыре светодиода подсветки. Габаритные размеры рамки пластмассового держателя составляют 35x35 мм. Соединительный шлейф с разъемом выступают за габаритные размеры рамки и также защищены пластмассовой защитной рамкой меньшего размера. Общие габаритные индикатора с защитными рамками составляют 35x48x3,5мм.
- Соединительный разъем имеет 10 контактов. Кабельная часть установлена на шлейфе индикатора, а платная часть предназначена для поверхностного (SMT - Surface Mount Technology) монтажа.
- В телефонах, в которых используются выбранные индикаторы, имеются очень удобные конструктивные элементы крепления индикатора к плате.
- Количество используемых сигнальных линий разъема, включая линии питания, составляет 9. Протокол нестандартный трехпроводный, очень похожий на SPI.
- Индикаторы имеют напряжение питания – 2,7 ÷ 3,3 В.
- Индикатор имеет достаточно низкую стоимость – примерно 9 долларов.
- Выбранный индикатор широко распространен, т.к. используется примерно в 30 типах мобильных телефонов, самые известные из которых Nokia 2650, 3100, 3200, 5100, 5140, 6100, 6220, 6610, 6610i, 6800, 6810, 6820, 7210, 7210e, 7250, 7250i и др.

В результате поиска информации по выбранным жидкокристаллическим индикаторам выяснилось, что эти индикаторы выпускаются рядом производителей, причем в этих индикаторах используется несколько типов контроллеров. Наиболее распространенными являются

жидкокристаллические индикаторы с контроллерами PCF8833 (Phillips) [8] и S1D15G00 (Epson) [9].

К сожалению, на многих индикаторах даже не указывается их тип, не говоря уже о типе контроллера, который в нем используется. Удалось выяснить, что индикаторы с контроллером PCF8833 фирмы Phillips имеют коричневый цвет печатной платы контроллера и шлейфа (см. рисунок 1), а индикаторы S1D15G00 фирмы Epson имеют зеленый цвет печатной платы и шлейфа (см.рисунок 2). Индикаторы фирмы Epson могут содержать и другие аналогичные контроллеры, например S1D15G10, S1D15G14, S1D15G17, имеющие незначительные отличия.

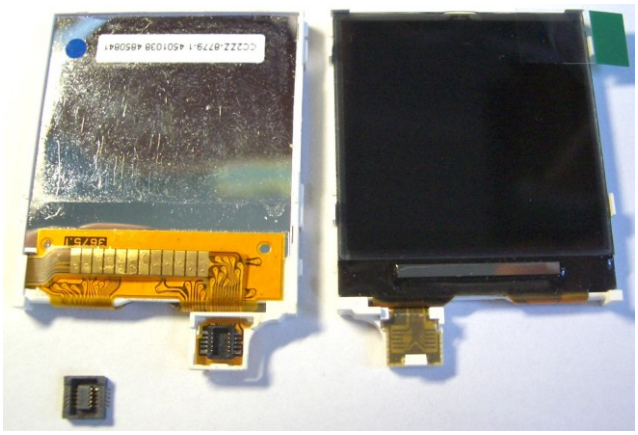

Рис.1. Жидкокристаллический индикатор с контроллером PCF8833 фирмы Phillips

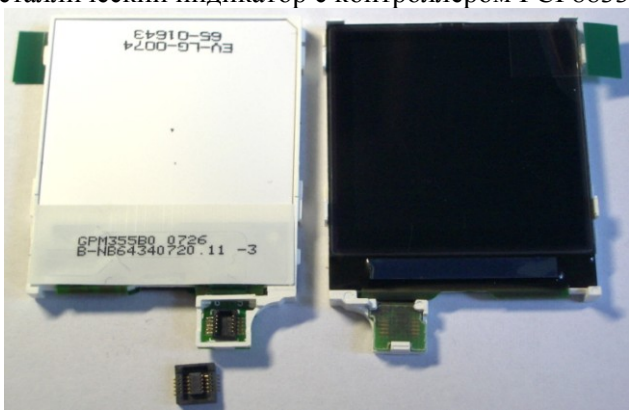

Рис.2. Жидкокристаллический индикатор с контроллером S1D15G00 фирмы Epson

На рисунках 1 и 2 индикаторы показаны с обоих сторон. В обоих индикаторах контроллеры установлены на дополнительных печатных платах со стороны разъема и заклеены бумажной наклейкой с производственными данными, как показано на рис.2. На индикаторе, показанном на рис.1, наклейка с платы контроллера снята. На каждом из рисунков показана также платная часть разъема.

Хотя ЖКИ с этими контроллерами и могут использоваться в одном и том же телефоне, они имеют контроллеры, не совместимые по системе команд. Программное обеспечение мобильных телефонов определяет тип подключенного индикатора (точнее, тип используемого в индикаторе контроллера) с помощью команд чтения ID (Identification Code) дисплеев.

На различных форумах разработчиков микроконтроллерных систем тема использования выбранных жидкокристаллических индикаторов широко обсуждается. Многие разработчики отдают предпочтение индикаторам с контроллерами фирмы Phillips. Как правило, отмечают следующие достоинства этих индикаторов: во-первых, документация на контроллер PCF8833 фирмы Phillips (коричневый шлейф) более понятная и читаемая; во-вторых, если с обратной стороны индикатора аккуратно приподнять защитную наклейку, покрывающую печатную плату контроллера, а затем приподнять шлейф – откроются 11 прекрасных больших контактов, к которым очень удобно подпаять обычные провода (см. рисунок 1); в-третьих, на описываемом индикаторе проще получить нормальную цветовую гамму, чем на индикаторе S1D15G00 фирмы

Epson. Однако, проведенные автором эксперименты показали, что если не принимать во внимание наличие у индикатора от фирмы Phillips печатных контактов на плате и пользоваться предусмотренными разъемами, остальные доводы можно считать субъективными. На самом деле достаточно хорошо программируются и ведут себя оба типа индикаторов.

Жидкокристаллические индикаторы крепятся к печатной плате телефона специальной рамкой, состоящей из двух частей (см. рис.3).

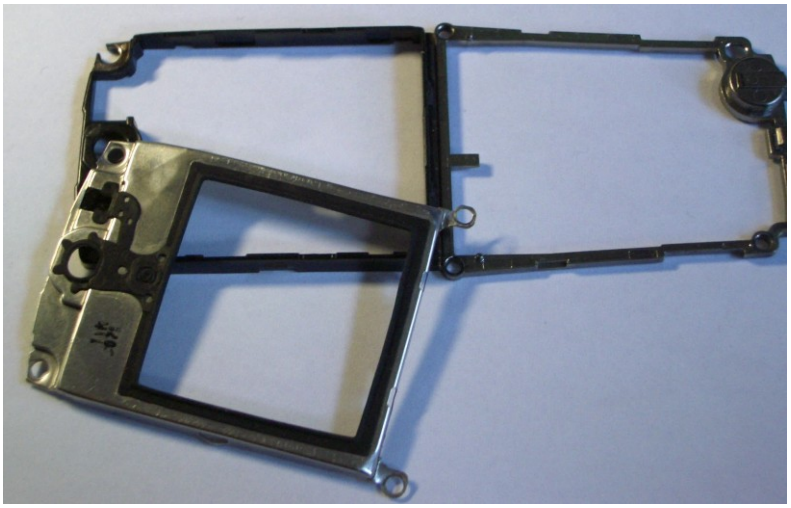

Рис.3. Рамка для крепления жидкокристаллического индикатора к печатной плате

Нижняя честь рамки – пластмассовая. Она, кроме стягивающей рамки индикатора, имеет еще и рамку для крепления клавиатуры телефона и гнездо для крепления микрофона (на рис.3. справа). Вторая верхняя часть рамки – металлическая с резиновым амортизатором. При использовании этой рамки для крепления индикатора в микроконтроллерных системах нижнюю рамку можно укоротить справа до крепежных отверстий (при этом она получится такого же размера, как и верхняя металлическая рамка). На рисунке 4 показан жидкокристаллический индикатор с крепежной рамкой, укрепленный на печатной плате контроллера.

Рис.4. Жидкокристаллический индикатор с крепежной рамкой, укрепленный на печатной плате контроллера

Многих разработчиков микроконтроллерных систем, конечно же, заинтересует вопрос о достижимом предельном (максимальном) быстродействии описываемых индикаторов. Так вот внимательное ознакомление с временными параметрами и системой команд обоих основных контроллеров позволяет посчитать максимальное быстродействие при последовательном формировании полных кадров изображения с размерами 128x128 пикселей. Время формирования полного кадра изображения на максимальной скорости составляет примерно 36 мкс. Это означает, что за одну секунду может быть полностью обновлено примерно 27,7 кадров. При этом следует подчеркнуть, что расчет производился при последовательной прорисовке кадра из массива оперативной памяти, в которой этот кадр хранился, и совершенно не учитывались никакие другие

операции. Кроме того, при расчете считалось, что используется однобайтное цветовое кодирование, т.е. каждый пиксель задается одним байтом. Отметим, что в реальных микроконтроллерных системах такое быстродействие довольно сложно достичь. Во-первых, для хранения одного кадра изображения необходимо достаточно много памяти. Очевидно, что эта величина составляет 128 x 128 байтов, или 16 Кбайт, что довольно значительная величина для микроконтроллерных систем. Во-вторых, алгоритм передачи данных многих современных микроконтроллеров предполагает сперва чтение данных из оперативной памяти (например, в аккумулятор), а затем – передача байта данных в жидкокристаллический индикатор. Для повышения быстродействия желательно использование механизма прямого доступа в память (DMA – Direct Memory Access), который является достаточно большой редкостью в микроконтроллерных системах. Если же система не использует механизм DMA, при расчете максимального быстродействия необходимо еще учитывать время, необходимое для операций модификации двухбайтного адреса данных. Таким образом, в микроконтроллерных системах, не имеющих аппаратных ускорителей (например, DMA), реально достижима частота смены полных кадров изображения примерно 15 Гц (15 кадров в секунду).

На рис.1 и 2 хорошо видны кабельная и платная части соединительного разъема. Следует отметить, что платная часть разъема не имеет ключа. Если смотреть на экран дисплея спереди, то нумерация контактов при счете сверху вниз будет 1 – 5 в левом вертикальном ряду и 10 – 6 в правом вертикальном ряду. Нумерация контактов показана также на рис.5.

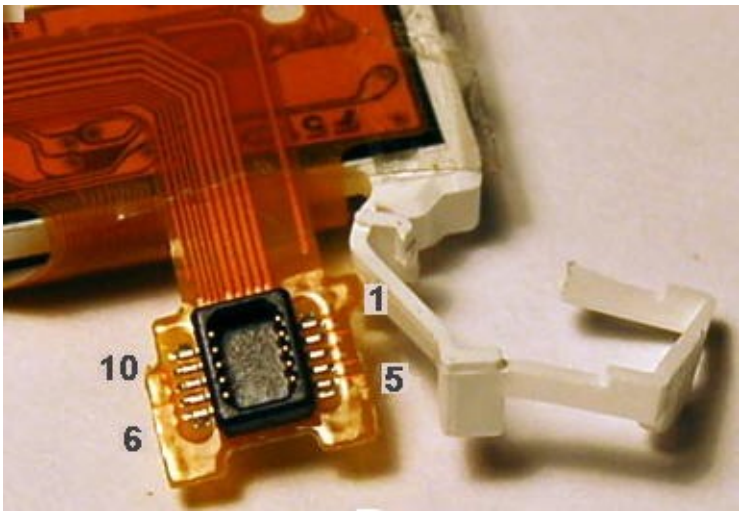

Рис.5. Нумерация контактов соединительного разъема

Назначение выводов соединительного разъема показано в таблице 1.

Таблица 1

Номер вывода	Название связи	Назначение связи
1	Digital Power	Напряжение питания контроллера, от +2,7 до +3,3 В
2	Reset/	Линия сброса (активный уровень «0»)
3	Serial Data	Линия данных
4	Serial Clock	Линия тактирования
5	Chip Select/	Линия выборки кристалла (активный уровень «0»)
6	Display Power	Напряжение питания индикатора, от +2,7 до +3,3 В
7	NC	Вывод не используется
8	GND	Общий вывод
9	LED-	Минусовой вывод светодиодов подсветки
10	LED+	Плюсовой вывод светодиодов подсветки

Отметим, что на практике, как правило, выводы 1 и 6 объединяются.

Описываемые жидкокристаллические индикаторы имеют встроенные светодиоды подсветки, расположенные с обратной стороны индикатора под пластиковым корпусом с зеркальным покрытием. Используется четыре последовательно соединенных светодиода. Рекомендуемое значение тока – 15-19 мА при напряжении около 5,8 - 6,2 В. Очевидно, что

напряжение это достаточно «неудобное» для микроконтроллерных систем, поэтому для его генерации используются специальные преобразователи, которые обычно называют White - LED Drivers. Специализированные микросхемы для этих преобразователей выпускаются многими фирмами, например фирма MAXIM выпускает 38 типов микросхем [10], предназначенных для таких преобразователей. Однако многие производители мобильных телефонов используют более дешевые и хорошо себя зарекомендовавшие микросхемы TK11850 / TK11851 фирмы TOKO[11]. Типовая схема включения для указанных микросхем и наших индикаторов приведена на рис.6.

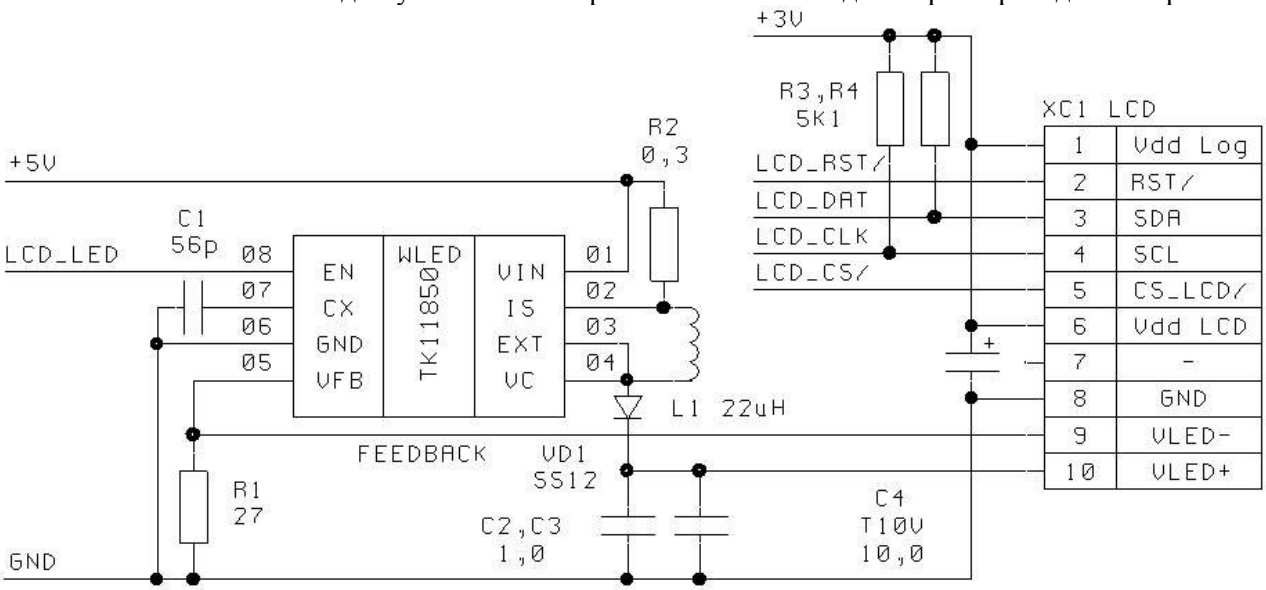

Рис.6. Подсистема LCD на микросхеме TK11850

Вход LCD_LED позволяет отключать подсветку индикатора («1» - включено). Ток подсветки задается с помощью резистора R1, а частота преобразования – емкостью конденсатора C1. Указанные на схеме номиналы рекомендованы производителем для жидкокристаллических индикаторов с четырьмя последовательными светодиодами подсветки, рассчитанными на ток примерно 20 мА. Микросхема TK11851 является дальнейшей модификацией использованной на рис.6 микросхемы TK11850. Отличие заключается в том, что резистор R2 и диод VD1, показанные на схеме рис.6, выполнены в интегральном исполнении внутри микросхемы TK11851. Иными словами, при использовании микросхемы TK11851 элементы R2 и VD1 устанавливать не нужно.

Рассмотрим основные принципы программирования жидкокристаллических индикаторов на примере совместной работы с микроконтроллерами фирмы Silicon Laboratories серии C8051F342[12], написанные на языке «C» для компилятора фирмы Keil[13].

Как уже отмечалось выше, описываемые жидкокристаллические индикаторы имеют разные системы команд. Обычно систему команд каждого из контроллеров описывают в отдельном файле описаний.

Файл описаний PCF8833.h системы команд контроллера PCF8833 от Phillips [8] выглядит следующим образом:

CODE

```
// ************************************************
// File PCF8833.h
// ************************************************

#ifdef	_PCF8833_
// ************************************************
#define  NOP                        0x00 // None
#define SOFT_RESET                  0x01 // None
#define BOOSTER_OFF                 0x02 // None
#define BOOSTER_ON                  0x03 // None
#define READ_ID                     0x04 // 4 Bytes Read
```

```
#define READ_STATUS             0x09 // 4 Bytes Read
#define SLEEP_IN                0x10 // None
#define SLEEP_OUT               0x11 // None
#define PARTIAL_ON              0x12 // None
#define DISPLAY_NORMAL          0x13 // None
#define DISPLAY_INV_OFF         0x20 // None
#define DISPLAY_INVERSE         0x21 // None
#define ALL_PIXELS_OFF          0x22 // None
#define ALL_PIXELS_ON           0x23 // None
#define SET_CONTRAST            0x25 // 1 Byte
#define DISPLAY_OFF             0x28 // None
#define DISPLAY_ON              0x29 // None
#define SET_COLUMN_ADDRESS      0x2A // 2 Bytes
#define SET_PAGE_ADDRESS        0x2B // 2 Bytes
#define WRITING_TO_MEMORY       0x2C // 1 Byte
#define RGB_SET                 0x2D // 20 Bytes
#define PARTIAL_AREA            0x30 // 2 Bytes
#define VERTICAL_SCROLL_AREA    0x33 // 3 Bytes
#define TEARING_LINE_OFF        0x34 // None
#define TEARING_LINE_ON         0x35 // 1 Byte
#define MEM_CONTROL             0x36 // 1 Byte
#define ROLLING_SCROLL_MODE     0x37 // 1 Byte
#define IDLE_MODE_OFF           0x38 // None
#define IDLE_MODE_ON            0x39 // None
#define COLOR_INTERFACE         0x3A // 1 Byte
#define SET_VOP                 0xB0 // 2 Bytes
#define BOTTOM_ROW_SWAP         0xB4 // None
#define  TOP_ROW_SWAP           0xB6 // None
#define FRAME_INVERSION         0xB9 // None
#define DATA_ORDER              0xBA // None
#define EN_DIS_TEMP_CONTROL     0xBD // None
#define EN_DIS_VQP_TEMP         0xBF // None
#define INT_EXT_OSC             0xC0 // None
#define SET_MULTIPLICATION      0xC2 // 1 Byte
#define SET_SLOPES_A_B          0xC3// 1 Byte
#define SET_SLOPES_C_D          0xC4 // 1 Byte
#define SET_DIVIDER             0xC5 // 4 Bytes
#define SET_DIVIDER_8           0xC6 // 1 Byte
#define SET_BIAS                0xC7 // 1 Byte
#define TEMP_READ_BACK          0xC8 // 1 Byte Read
#define NLINE_INVERSION         0xC9 // 1 Byte
#define READ_ID1                0xDA // 1 Byte Read
#define READ_ID2                0xDB // 1 Byte Read
#define READ_ID3                0xDC // 1 Byte Read
#define FACTORY_DEFAULT         0xEF // None
#define CALIBRATION             0xF0 // None
// ***********************************************
#endif // _PCF8833_
```

Файл описаний S1D15G00.h системы команд контроллера S1D15G00 от Epson [9] выглядит следующим образом:

CODE

```
// ***********************************************
// File S1D15G00.h
// ***********************************************

#ifdef  _S1D15G00_
// ***********************************************
#define  DISPLAY_ON                          0xAF // None
#define DISPLAY_OFF                 0xAE // None
#define DISPLAY_NORMAL              0xA6 // None
#define DISPLAY_INVERSE             0xA7 // None
#define  COMMON_SCAN_DIRECTION 0xBB // 1 Byte
#define DISPLAY_CONTROL             0xCA // 3 Bytes
#define SLEEP_IN                    0x95 // None
#define SLEEP_OUT                   0x94 // None
#define SET_PAGE_ADDRESS            0x75 // 2 Bytes
#define SET_COLUMN_ADDRESS    0x15 // 2 Bytes
#define DATA_CONTROL                0xBC // 3 Bytes
#define RGB_SET                     0xCE // 20 Bytes
#define WRITING_TO_MEMORY       0x5C // 1 Byte
#define READING_FROM_MEMORY 0x5D // 1 Byte
#define  PARTIAL_DISPLAY_IN         0xA8 // 2 Bytes
#define PARTIAL_DISPLAY_OUT     0xA9 // None
#define READ_AND_MODIFY_WRITE 0xE0 // None
#define END                         0xEE // None
#define SET_SCROLL_AREA             0xAA // 4 Bytes
#define INT_OSCILL_ON               0xD1 // None
#define INT_OSCILL_OFF              0xD2 // None
#define POWER_CONTROL               0x20 // 1 Byte
#define ELEC_VOLUME_CONTROL  0x81 // 2 Bytes
#define INC_ELEC_CONTROL            0xD6 // None
#define DEC_ELEC_CONTROL            0xD7 // None
#define SET_TEMP_GRADIENT       0x82 // 1 Byte
#define EEPROM_CONTROL              0xCD // 1 Byte
#define EEPROM_CONTROL_CONCEL 0xCC // None
#define EEPROM_WRITE                0xFC // None
#define EEPROM_READ                 0xFD // None
#define READ_REGISTER_1             0x7C // None
#define READ_REGISTER_2             0x7D // None
#define NOP                         0x25 // None
// ***********************************************
#endif //  _S1D15G00_
```

Для организации интерфейса с жидкокристаллическими индикаторами обычно используется 5 линий микроконтроллера. Для выбранного микроконтроллера C8051F342 произведем следующие определения:

CODE

```
sbit      LCD_RST_Bit = P1^0;
sbit      LCD_DAT_Bit= P1^1;
sbit      LCD_CLK_Bit = P1^2;
sbit      LCD_CS_Bit    = P1^3;
sbit      LCD_LED_Bit = P1^4;
```

Конечно же, и номера линий и номер порта ввода / вывода, могут быть выбраны произвольно. Эти определения используются для подпрограмм доступа к назначенным входам / выходам. Следует помнить, что эти подпрограммы должны быть в том же программном модуле, в котором приведены соответствующие sbit определения.

CODE

```
void    LCD_CS        (bit A)  {LCD_CS_Bit=A;}
void    LCD_RST       (bit A)  {LCD_RST_Bit=A;}
void    LCD_CLK       (bit A)  {LCD_CLK_Bit=A;}
void    LCD_DAT_Set   (bit A) {LCD_DAT_Bit=A;}
bit     LCD_DAT_Get   (void) {return LCD_DAT_Bit;}
void    LCD_LED       (bit A)  {LCD_LED_Bit=A;}
```

Подпрограммы нижнего уровня одинаковые для обоих типов контроллеров. Следует отметить, что эти подпрограммы используют две внешние подпрограммы: WDT() – подпрограмма перезапуска охранного таймера (если он используется) и Time (unsigned uS) – подпрограмма задержки примерно на uS микросекунд.

CODE

```
// ***********************************************
// File Low_Level_Functions.C
// ***********************************************
extern void      WDT (void);
extern void       Time (unsigned uS);

extern void      LCD_CS (bit A);
extern void      LCD_RST (bit A);
extern void      LCD_CLK (bit A);
extern void      LCD_DAT_Set (bit A);
extern bit       LCD_DAT_Get  (void);
extern void      LCD_LED (bit A);

// ************************************
// Подпрограмма выполняет сброс LCD контроллера
// ************************************
void LCD_Reset (void)
{
    LCD_CS (1);                 // Установить CS
    LCD_RST (1);                // Установить RST
    LCD_CLK (1);                // Установить CLK
    LCD_DAT_Set (1);            // Установить данные
    LCD_CLK (0);                // Сбросить CLK
    Time (200);                 // Задержка примерно 200 мкс
    LCD_RST (0);                // Сбросить RST
    Time (10);                  // Задержка примерно 10 мкс
    LCD_RST (1);                // Установить RST
}

// ************************************
// Генерация тактового импульса
```

```
// *************************************
void LCD_Pulse (void)
{
    _nop_();                    // Задержка на 1 такт MCU примерно 50 нс
    LCD_CLK(1);                 // Установить CLK
    _nop_();                    // Задержка на 1 такт MCU примерно 50 нс
    LCD_CLK(0);                 // Сбросить CLK
}


// *************************************
// Пересылка одного бита
// *************************************
void LCD_BitSend (bit B)
{
   _nop_();                     // Задержка на 1 такт MCU примерно 50 нс
   LCD_DAT_Set (B);             // Установить бит данных
   LCD_Pulse ();                // Сгенерировать тактовый импульс
}

// *************************************
// Пересылка байта в LCD
// Если CB=0, пересыка команды
// Если CB=1, пересылка данных
// *************************************
void LCD_Send (byte DAT, bit CB)
{
register byte i;
register byte mask = 0x80;

   LCD_CS  (0);                 // Сбросить CS
   _nop_();                     // Задержка на 1 такт MCU примерно 50 нс
   LCD_BitSend(CB);             // Переслать бит признака операции
                                // «0» - если далее следует байт команды
                                // «1» - если далее следует байт данных
    for (i=0;i<8;i++)
    {
        LCD_BitSend ( (DAT&mask)?1:0 ); // Переслать очередной бит
        mask=mask>>1;           // Модифицировать маску (бит)
   }
   LCD_CS  (1);                 // Установить CS
}

// Переслать байт команды
void LCD_COM (byte COM)    {LCD_Send(COM,0);}

// Переслать байт данных
void LCD_DAT (byte DAT)    {LCD_Send(DAT,1);}

// *************************************
// Чтение данных из LCD
// *************************************
byte LCD_Read (void)
{
register byte i;
register byte mask = 0x80;
```

```
register byte REZ=0;

    LCD_CS  (0);                // Сбросить CS
   _nop_();                     // Задержка на 1 такт MCU примерно 50 нс
    for (i=0;i<8;i++)
    {
        LCD_CLK(1);             // Установить  CLK
       _nop_();                 // Задержка на 1 такт MCU примерно 50 нс
       if (LCD_DAT_Get())       // Взять бит данных
          REZ|=mask;
       mask=mask>>1;            // Модифицировать маску (бит)
    }
    LCD_CS  (1);                // Установить CS
    return REZ;                 // Вернуть байт данных
}
```

К сожалению, на этом общие программы для различных жидкокристаллических индикаторов, т.е. индикаторов с различными контроллерами, заканчивается. Все остальные команды более высокого уровня располагаются в индивидуальных файлах.

Рассмотрим файл включаемый файл GraphLCD.h, содержащий определения цветов, некоторых величин, и функции инициализации, одинаковый для обоих контроллеров:

CODE

```
// ***********************************************
// File GraphLCD.h
// ***********************************************

// ВЫБОР ТИПА ИСПОЛЬЗУЕМОГО КОНТРОЛЛЕРА LCD

#define         _S1D15G00_  // Выбран LCD с контроллером EPSON
//#define       _PCF8833_

// ***********************************************
#ifdef          _S1D15G00_

#include        "S1D15G00.h"

#else

#include        "PCF8833.h"

#endif
// ***********************************************



// Определение типа byte
#ifndef  __BYTE__
#define  __BYTE__
typedef  unsigned char  byte;
#endif   /* __BYTE__ */

// ***********************************************
// Основные макроопределния
// ***********************************************
```

```
#define  LCD_Mem_Write()    LCD_COM(WRITING_TO_MEMORY)
#define  LCD_Clear()        LCD_FillRect (0,0,SCREEN_WIDTH-1,SCREEN_HEIGHT-
1,BG_COLOR)

// ***********************************************
// Определения основных цветов
// ***********************************************
#define  RGB(r,g,b)         ((r&0xE0)|((g&0xE0)>>3)|(b>>6))
#define  NONE               RGB(0x00, 0x20, 0x00)
#define  BLACK              RGB(0x00, 0x00, 0x00)
#define  WHITE              RGB(0xFF, 0xFF, 0xFF)
#define  RED                RGB(0xFF, 0x00, 0x00)
#define  GREEN              RGB(0x00, 0xFF, 0x00)
#define  BLUE               RGB(0x00, 0x00, 0xFF)
#define  YELLOW             RGB(0xFF, 0xFF, 0x00)
#define  MAGENTA            RGB(0xFF, 0x00, 0xFF)
#define  CYAN               RGB(0x00, 0xFF, 0xFF)
#define  GRAY               RGB(0x80, 0x80, 0x40)
#define  SILVER             RGB(0xA0, 0xA0, 0x80)
#define  GOLD               RGB(0xA0, 0xA0, 0x40)

// ***********************************************
// Определения основных размеров
// ***********************************************
#define  SCREEN_WIDTH     132
#define  SCREEN_HEIGHT    132
#define  SCREEN_LEFT      4
#define  SCREEN_TOP       4
#define  SCREEN_RIGHT     (SCREEN_WIDTH-4)
#define  SCREEN_BOTTOM    (SCREEN_HEIGHT-4)

// ***********************************************
// Внешние подпрограммы
// ***********************************************
extern void LCD_CS          (bit A);
extern void LCD_RST         (bit A);
extern void LCD_CLK         (bit A);
extern void LCD_DAT_Set     (bit A);
extern bit  LCD_DAT_Get     (void);
extern void LCD_LED         (bit A);
// ***********************************************
extern void Time            (unsigned mkS);
extern void Delay           (unsigned mS);
extern void WDT             (void);
// ***********************************************
extern void  LCD_Reset      (void);
extern void  LCD_Pulse      (void);
extern void  LCD_BitSend    (bit B);
extern void   LCD_Send      (byte DAT, bit CB);
extern byte   LCD_Read      (void);
extern void   LCD_COM       (byte COM);
extern void   LCD_DAT       (byte DAT);
// ***********************************************
void   LCD_Init   (void);
```

Далее приводится файл PCF8833.C, содержащий подпрограммы, необходимые для инициализации LCD с контроллером от Phillips:

CODE

```
// ************************************
// * PCF8833.C
// ************************************
#include "GraphLCD.h"

#ifdef _PCF8833_
// ************************************************
#define LCD_NOP()                   LCD_COM(NOP)
#define LCD_Soft_Reset()            LCD_COM(SOFT_RESET)
#define LCD_Booster_ON()            LCD_COM(BOOSTER_ON)
#define LCD_Booster_OFF()           LCD_COM(BOOSTER_OFF)
#define LCD_ON()                    LCD_COM(DISPLAY_ON)
#define LCD_OFF()                   LCD_COM(DISPLAY_OFF)
#define LCD_Sleep_In()              LCD_COM(SLEEP_IN)
#define LCD_Sleep_Out()             LCD_COM(SLEEP_OUT)
#define LCD_Idle_Mode_OFF()         LCD_COM(IDLE_MODE_OFF)
#define LCD_Idle_Mode_ON()          LCD_COM(IDLE_MODE_ON)
#define LCD_All_Pixels_OFF()        LCD_COM(ALL_PIXELS_OFF)
#define LCD_All_Pixels_ON()         LCD_COM(ALL_PIXELS_ON)
#define LCD_Normal_Mode()           LCD_COM(DISPLAY_NORMAL)
#define LCD_Inverse_Mode()          LCD_COM(DISPLAY_INVERSE)
#define LCD_Inverse_OFF()           LCD_COM(DISPLAY_INV_OFF)
#define LCD_Frame_Inversion()       LCD_COM(FRAME_INVERSION)
#define LCD_Tearing_Line_OFF()      LCD_COM(TEARING_LINE_OFF)
#define LCD_Partial_Mode_ON()       LCD_COM(PARTIAL_ON)
#define LCD_Oscill(BIT)             LCD_COM(INT_EXT_OSC|BIT)
#define LCD_Data_Order(BIT)         LCD_COM(DATA_ORDER|BIT)
#define LCD_Factory_Default(BIT)    LCD_COM(FACTORY_DEFAULT|BIT)
#define LCD_Top_Row_Swap(BIT)       LCD_COM(TOP_ROW_SWAP|BIT)
#define LCD_Bottom_Row_Swap(BIT)    LCD_COM(BOTTOM_ROW_SWAP|BIT)
#define LCD_Temp_VQP(BIT)           LCD_COM(EN_DIS_VQP_TEMP|BIT)
#define LCD_Temp_Control(BIT)       LCD_COM(EN_DIS_TEMP_CONTROL|BIT)

// ************************************************
// Выбор режима  8, 12 или 16 бит / пиксель
// ************************************************
static void LCD_Color_Interface  (byte COL)
{
register byte ch=0;

   switch(COL)
   {
      case 12:   ch=0x03;
                 break;
      case 16:   ch=0x05;
                 break;
      default:
      case 8:    ch=0x02;
                 break;
    }
```

```
    LCD_COM (COLOR_INTERFACE);
    LCD_DAT (ch);
}

// ***********************************************
// Установка кода контраста
// ***********************************************
static void LCD_Set_Contrast (byte CONTR)
{
    LCD_COM (SET_CONTRAST);
    LCD_DAT (CONTR);
}
// ***********************************************
// Установка направлений заполнения экрана
// ***********************************************
static void LCD_Memory_Control  (bit MirrorY,  bit MirrorX,  bit DirY,  bit LAO_BT,  bit BGR)
{
register byte ch=0;

   if (MirrorY)  ch|=0x80;
   if (MirrorX)  ch|=0x40;
   if (DirY)      ch|=0x20;
   if (LAO_BT)  ch|=0x10;
   if (BGR)      ch|=0x08;
   LCD_COM  (MEM_CONTROL);
   LCD_DAT   (ch);
}

// ***********************************************
// Подпрограмма инициализации
// ***********************************************
void LCD_Init (void)
{
   LCD_Reset ();                        // Аппаратный сброс LCD
   LCD_Soft_Reset();                    // Программный сброс LCD
   LCD_Sleep_Out();                             // Активировать LCD
   LCD_ON();                            // Включить LCD
   LCD_Booster_ON();                    // Включить «горячий» старт
   LCD_Memory_Control (0,1,0,0,0);    // Задать режим 8-битного RGB
   Delay (200);                         // Задержка примерно 200 мс
   LCD_Set_Contrast (65);               // Установить рекомендованное
                                        // значение контраста
   LCD_Color_Interface (8);             // Интерфейс цветности – 8 бит
   LCD_8Bit_Palette_Set();              // Установить 8 битную палитру
   LINE_WIDTH=1;                        // Установить текущую толщину
                                        // линии для рисования
   LCD_Set_BG_Color (WHITE);        // Установить текущий цвет фона
   LCD_Clear();                         // Очистить экран
}
// ***********************************************
#endif // _PCF8833_
```

Аналогичный файл S1D15G00.C для инициализации LCD с контроллером от Epson выглядит следующим образом:

CODE

```
// ************************************
// * S1D15G00.C
// ************************************
#include "GraphLCD.h"

#ifdef          _S1D15G00_

// ********************************************
#define         LCD_ON()              LCD_COM(DISPLAY_ON)
#define         LCD_OFF()             LCD_COM(DISPLAY_OFF)
#define         LCD_Normal_Mode()  LCD_COM(DISPLAY_NORMAL)
#define         LCD_Inverse_Mode()  LCD_COM(DISPLAY_INVERSE)
#define         LCD_Sleep_In()        LCD_COM(SLEEP_IN)
#define         LCD_Sleep_Out()       LCD_COM(SLEEP_OUT)
#define         LCD_Oscill_ON()       LCD_COM(INT_OSCILL_ON)
#define         LCD_Oscill_OFF()      LCD_COM(INT_OSCILL_OFF)
#define         LCD_END()             LCD_COM(END)
#define         LCD_NOP()             LCD_COM(NOP)
#define         LCD_Read_RG1()        LCD_COM(READ_REGISTER_1)
#define         LCD_Read_RG2()        LCD_COM(READ_REGISTER_2)
#define         LCD_Elec_Control_Inc()  LCD_COM(INC_ELEC_CONTROL)
#define         LCD_Elec_Control_Dec()  LCD_COM(DEC_ELEC_CONTROL)
#define         LCD_Partial_Display_Out()  LCD_COM(PARTIAL_DISPLAY_OUT)
#define         LCD_Read_and_Modify_Write()  LCD_COM(READ_AND_MODIFY_WRITE)
#define         LCD_EEPROM_Control_Concel() LCD_COM(EEPROM_CONTROL_CONCEL)
#define         LCD_EEPROM_Write()  LCD_COM(EEPROM_WRITE)
#define         LCD_EEPROM_Read()  LCD_COM(EEPROM_READ)

// ********************************************
// Установка частотного режима
// ********************************************
static void LCD_Display_Control (byte DIV, byte DUTY, byte FRINV)
{
   LCD_COM (DISPLAY_CONTROL);          // Установка времен задержки и
                                       // частот, необходимых для работы
   LCD_DAT (DIV);                      // Передать код делителя частоты
   LCD_DAT (DUTY);                     // Передать коэффициент заполнения
   LCD_DAT (FRINV);                    // Установить флаг инверсии
   LCD_DAT (0x00);                     // Завершить передачу частотных
                                       // параметров
}

// ********************************************
// Установить направление сканирования выводов контроллера
// ********************************************
static void LCD_Pin_Scan_Direction (byte B)
{
   LCD_COM (COMMON_SCAN_DIRECTION);  // Установка направления
   LCD_DAT (B);                        // Выбрать байт направления
}

// ********************************************
```

```
// Установить яркость и контрастность
// ***********************************************
static void LCD_Electric_Control (byte CONTRAST, byte BRIGHTNESS)
{
   LCD_COM (ELEC_VOLUME_CONTROL); // Установить значения
   LCD_DAT (CONTRAST);                    // Рекомендуемый контраст =5
   LCD_DAT (BRIGHTNESS);                  // Рекомендуемая яркость =2
}

// ***********************************************
// Установить параметры питания
// ***********************************************
static void LCD_Power_Control (byte B)
{
   LCD_COM (POWER_CONTROL);  // Включить все цепи питания LCD
   LCD_DAT (B);                           // Передать код
}

// ***********************************************
// Установить эксплуатационные параметры
// ***********************************************
static void LCD_Data_Control_Set  (byte PAGE_N_I, byte COLUMN_N_I, byte DIR_C_P,
                                   byte RGB_Type, byte GRAY_SCALE)
{
register byte ch=0;

   LCD_COM (DATA_CONTROL);    // Установка параметров
   if (PAGE_N_I)        ch|=0x01;        // Направление рядов
   if (COLUMN_N_I)   ch|=0x02;        // Направление столбцов
   if (DIR_C_P)         ch|=0x04;        // Направление заполнения

   LCD_DAT (ch);                          // Передать данные
   LCD_DAT (RGB_Type);                // Задать цветность
   LCD_DAT (GRAY_SCALE);          // Задать градации серого
   LCD_DAT (0x00);                        // Завершить передачу
}

// ***********************************************
// Подпрограмма инициализации
// ***********************************************
void LCD_Init (void)
{
   LCD_Reset ();                          // Сбросить LCD
   LCD_Display_Control(0x03,0x20,0x0c);       // Установить временные параметры
   LCD_Pin_Scan_Direction (0x01);     // Установить параметры сканирования выводов
   LCD_Oscill_ON();                       // Включить внутренний генератор
   LCD_Sleep_Out();                       // Активировать LCD
   LCD_Electric_Control (0x3f,0x01);  // Установить рекомендуемые яркость и контраст
   LCD_Power_Control (0x0F);          // Включить цепи питания
   Delay (100);                           // Задержка примерно 100 мс
   LCD_Inverse_Mode();                // Включить инверсный режим
   LCD_Data_Control_Set(0,1,0,0,0x01);  // Установить параметры данных
   LCD_8Bit_Palette_Set();            // Установить палитру
   LCD_ON();                              // Включить LCD
   Delay (200);                           // Задержка примерно 200 мс
   LINE_WIDTH=1;                          // Установить текущую толщину
```

```
                                        // линии для рисования
   LCD_Set_BG_Color (WHITE);        // Установить текущий цвет фона
   LCD_Clear();                     // Очистить экран
}
// ***********************************************
#endif //          _S1D15G00_
```

После проведения инициализации дисплей полностью готов к работе. Теперь пользователь может с помощью простейших функций строить различные изображения. В заключении приведем еще две простейшие функции, которые позволят разработчику быстро создать (нарисовать) простейшие изображения:

CODE

```
// ***********************************************
// Подпрограмма задания адреса точки или области
// ***********************************************
void LCD_SetAddr (byte x1, byte y1, byte x2, byte y2)
{
   LCD_COM(SET_PAGE_ADDRESS);       // Передать код задания
                                    // диапазона по строке
                                    // величина X1 должна быть меньше X2
   LCD_DAT(X1);                     // Адрес начала зоны
   LCD_DAT(X2);                     // Адрес конца зоны
   LCD_COM(SET_COLUMN_ADDRESS);     // Передать код задания
                                    // диапазона по столбцу
                                    // величина Y1 должна быть меньше Y2
   LCD_DAT(Y1);                     // Адрес начала зона
   LCD_DAT(Y2);                     // Адрес конца зоны
}

// ***********************************************
// Подпрограмма вывода цветовой точки
// ***********************************************
void LCD_PutPixel (byte x, byte y, byte color)
{
   LCD_SetAddr (x, y, x+1, y+1);    // Задать координаты точки
   LCD_Mem_Write ();                // Передать код записи в память
   LCD_DAT (color);                 // Записать байт цвета
}
```

Автор надеется, что приведенная статья будет полезна многим разработчикам микроконтроллерных систем, желающим выводить информацию на цветные графические жидкокристаллические индикаторы от мобильных телефонов.

Литература:

1. http://www.datavision.com.tw
2. http://www.melt.com.ru/
3. http://www.intech-lcd.com
4. http://www.ampire.com.tw
5. http://www.bolymin.com.tw
6. http://www.huton-cellphone.com/lcd/ML019-001091.htm
7. http://www.huton-cellphone.com/lcd/ML020-001092.htm
8. http://www.nxp.com/acrobat_download/datasheets/PCF8833_1.pdf
9. http://www.shellyinc.com/techsup/S1D15G00_REV1_0.pdf
10. http://para.maxim-ic.com/cache/en/results/39545.html
11. http://www.tokoam.com/semiconductors/pdf/tk11850l-e-gc3_i008b.pdf
12. http://www.silabs.com/public/documents/tpub_doc/dsheet/Microcontrollers/USB/en/C8051F34x.pdf
13. http://www.keil.com