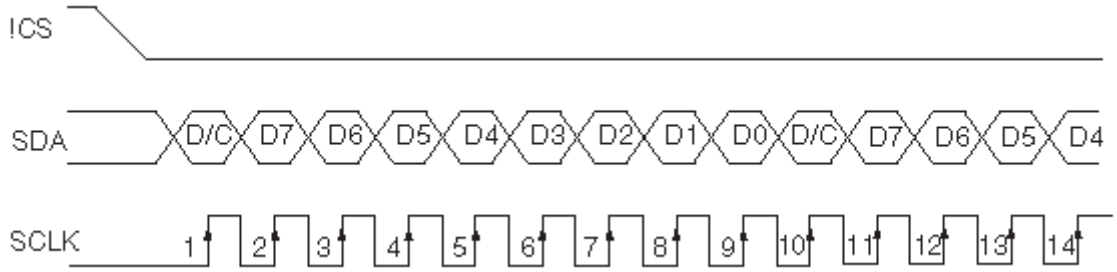
Экран от Nokia 1112 + графическая библиотека

1.Введение. Все началось с того, что в магазине запчастей для сотовых телефонов мне попался дешевый экранчик от мобильника Nokia 1112. Я подумал, что при цене чуть больше 1 бакса в моих домашних поделках это может стать хорошей альтернативой текстовым дисплеям на контроллере HD44780(аналог KS066). После ряда экспериментов я написал несложную библиотеку графического интерфейса пользователя, которая может быть запущена на многих простых 8-битных микроконтроллерах. Код был написан на Си для ATMega88.

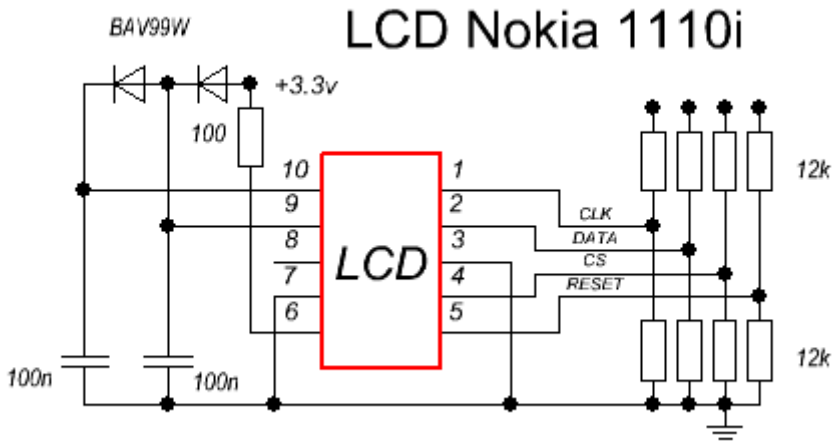
2.Немного теории. LCD экран от Nokia 1112 содержит контроллер, который по командам совместим с контроллерами PCF8814 и STE2007. Т.к. экран имеет разрешение 96x68 точек, то следует ориентироваться на описание STE2007. STE2007 может использовать несколько разных интерфейсов, которые должны заранее программироваться на заводе. В телефонах используется 3-проводной последовательный интерфейс типа SPI:



В реальности используется 4-проводное включение т.к. еще нужен сигнал сброса Reset. Данные только записываются в дисплей т.е. передача односторонняя(всегда МК->экран).

Первый бит сообщает о том, что это данные(под словом «данные» понимается изображение) или команда. За ним следует 8 бит(1 байт). Т.о. длина кадра составляет 9 бит, что не совсем удобно для использования в простых микроконтроллерах.

По питанию тоже есть свои особенности, для работы нужно 2 напряжения: +1.8 и +2.8В. К счастью их несложно получить с помощью 2-х диодов из напряжения +3.3В, которое имеет большее распространение:



Эту схему я нашел где-то на просторах интернета, на ней также показаны резистивные делители для согласования логических уровней МК с уровнями экрана.

3.Графическая библиотека.

Графическая библиотека состоит из 3-х уровней:

1. Уровень функций ввода-вывода
2. Уровень экранного буфера и графических примитивов
3. Уровень элементов графического интерфейса

3.1.Уровень функций ввода вывода. Обеспечивает взаимодействие на физическом уровне МК с контроллером графического дисплея.

Файлы: lcd1110i.h , lcd1110i.c

Макросы:

Макрос	Описание
LCD_CLK	Присвоение пину порта соответствующей интерфейсной функции
LCD_DAT	
LCD_CS	
LCD_RST	

Функции:

void lcd_init (void)

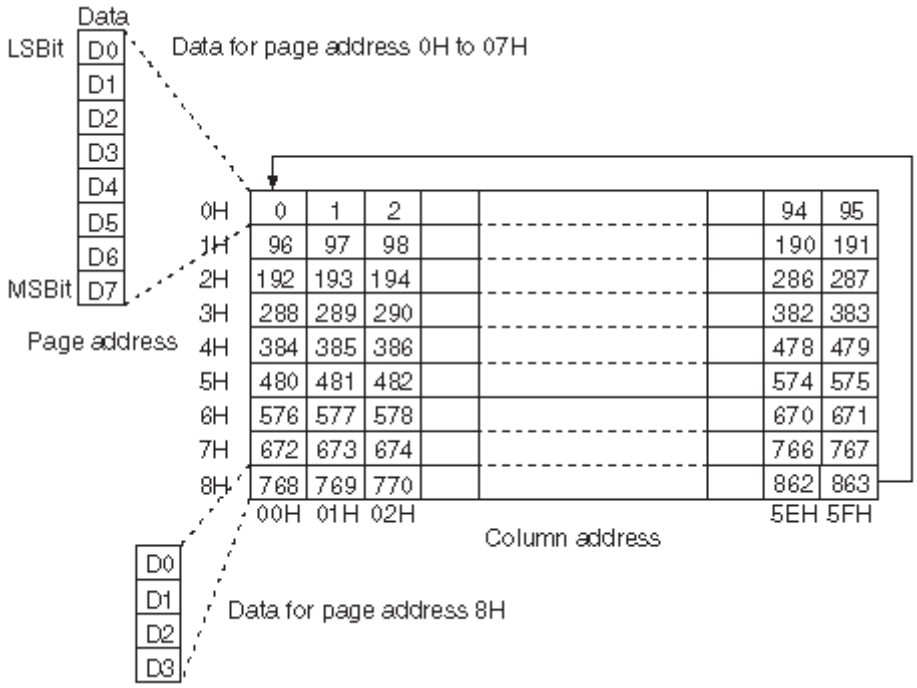
Сброс и инициализация контроллера дисплея. Это самая первая функция, которая должна вызываться после подачи питания.

void lcd_write (unsigned char cd , unsigned char c)

Записывает в экран команду/данные. Интерфейс SPI в контроллерах AVR не может работать в 9-битном режиме, поэтому данные передаются программным ногодрыганьем.

void lcd_gotoxy (unsigned char x , unsigned char y)

Переводит указатель для записи данных в определенную позицию. Т.к. изображение хранится в DDRAM контроллера с вертикальной упаковкой байт, то y=0…8. См. следующую картинку:



3.2.Уровень экранного буфера и графических примитивов.

Файлы: lcd_screen.h , lcd_screen.c

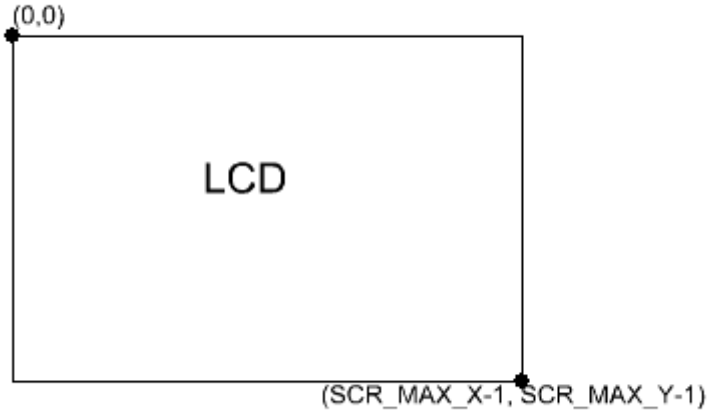
Т.к. данные не могут быть считаны из памяти экрана, то для формирования сложных изображений необходимо в ОЗУ контроллера держать экранный буфер. После прорисовки всего изображения буфер копируется из микроконтроллера в DDRAM экранчика. В отличии от самого нижнего уровня, данный уровень уже оперирует координатами пиксела на экране.

Макросы:

Параметр	Описание
SCR_BUF_SIZE	Размер видеобуфера в байтах
SCR_MAX_X	Ширина экрана в пикселах(96)
SCR_MAX_Y	Высота экрана в пикселах(68)
_PIX_OVF_TRP	Включает обработчик выхода координаты пиксела за пределы экрана. Может быть полезен на этапе отладки программы. Будет работать только в функциях, которые для рисования используют scr_pixel().

Размер экранного буфера 864 байт, поэтому графическая библиотека может быть запущена на микроконтроллерах с объемом памяти не менее 1Кбайт ОЗУ.

Координаты экрана:



Функции:

void scr_cls(void)

Очистить буфер, заполняет массив нулями. Это первая функция, которую следует вызвать при перерисовке экрана.

void scr_pixel(unsigned char x, unsigned char y)

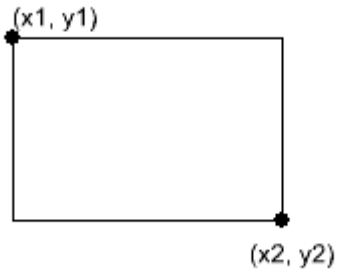
Помещает в буфер черный пиксел по координате (x,y).

void scr_line(unsigned char xn, unsigned char yn, unsigned char xk, unsigned char yk)

Рисует в буфер линию алгоритмом Бразенхема

void scr_box(unsigned char x1, unsigned char y1, unsigned char x2, unsigned char y2)

Выводит прямоугольник. Подразумевается что: x2>=x1 и y2>=y1.



void scr_circle(unsigned char x, unsigned char y, unsigned int r)

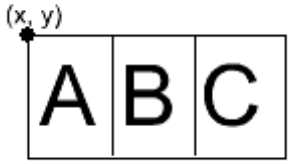
Рисует окружность алгоритмом Бразенхема с центром (x,y) и радиусом r.

void scr_invert(unsigned char x1, unsigned char y1, unsigned char x2, unsigned char y2)

Инвертирует обозначенный прямоугольник(функция побитового НЕ). Имеет некоторый косяк если (y2-y1)<8. Он выражается в виде отклонения координаты Y на несколько пикселов. Условие координат тоже, что и для прямоугольника scr_box().

void scr_xy(unsigned char x, unsigned char y)

Задает начальную координату (x,y) для вывода текста.



Фактически задает значения координат глобальным переменным scr_out_x и scr_out_y.

void scr_putchar(char c)

Печатает в буфер одиночный символ по текущим экранным координатам scr_out_x и scr_out_y. По мере печати увеличивается значение scr_out_x на ширину символа CHR_WIDTH+1(определяется в файле шрифтов lcd_font.h). Если следующий символ выходит за пределы экрана, то scr_out_y увеличивается на высоту символа CHR_HEIGHT-1, а scr_out_x обнуляется. Для печати используется шрифт 5x7 точек, он содержит изображения стандартных символов с номерами 32…127 и 192…255.

void scr_putsf(flash char *str)

Печатает строку из Flash с помощью scr_putchar().

void scr_printchar(unsigned char val, unsigned char d_num)

Печатает десятичное значение val(0…255), d_num указывает минимальное количество отображаемых на экране цифр(1…3). Пример: val=1, d_num=3, будет напечатано 001.

void scr_update(void)

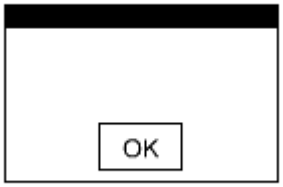
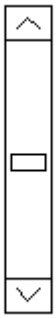
Отправляет содержимое экранного буфера из ОЗУ микроконтроллера в память экрана, т.е. обновляет экран.

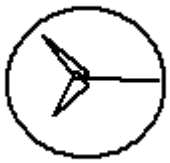
3.3. Уровень элементов графического интерфейса.

Файлы: gui.h , gui.c .

Элементы графического интерфейса пользователя(GUI) представляют собой стандартные кубики, с помощью которых можно быстро собрать меню или другие диалоги с пользователем. Каждый элемент имеет набор свойств(например, координаты на экране, значение, строка заголовка и т.д.), свойства декларируются собственной структурой элемента и для отображения на экране структура передается в соответствующую функцию. Я старался давать одинаковые имена однотипным свойствам у разных элементов. Может сложиться впечатление, что элементы

интерфейса являются элементами управления как в ОС Windows(или другой ОС), однако это не так. Здесь элемент интерфейса(например кнопка) – это всего лишь изображение, он не может вызывать никаких событий. Т.е. вся забота по взаимодействию с внешним миром по прежнему остается на плечах программиста.
Элементы, которые вошли в эту библиотеку:

Имя	Структура	Вид
Dialog	DLG_STRUCT	
Button	BTN_STRUCT	
Check Box	CHK_STRUCT	
Vertical Scroll Bar	VSCRL_STRUCT	
Horizontal Progress Bar	HPRGS_STRUCT	
List	LIST_STRUCT	

Analog Clock	CLK_STRUCT	
Text Clock	TXT_CLK_STRUCT	12:34:56

Имена элементов даю на английском т.к. некоторые из них не имеют короткого прямого перевода.

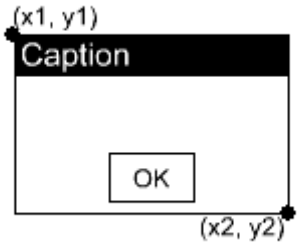
Программист сам должен позаботиться о корректности задаваемых свойств т.к. в большинстве случаев проверка на допустимость параметров не производится ради уменьшения объема кода.

Dialog – это окошко с заголовком и несколькими кнопками.

Структура:

```
struct DLG_STRUCT
    {
    unsigned char x1;      //start x position
    unsigned char y1;      //start y position
    unsigned char x2;      //end x
    unsigned char y2;      //end y
    unsigned char dlg_type; //тип диалога
    unsigned char btn_select; //выделенная кнопка
    const char *caption;    //заголовок, строка во Flash-памяти
    };
```

Координаты диалога:



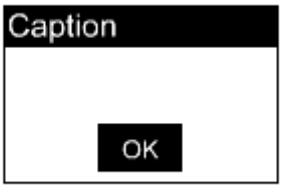
Тип диалога:

dlg_type	Описание
DLG_NO_BTN	Сообщает графической функции, что окно диалога не содержит кнопок.
DLG_OKONLY_BTN	Окошко содержит только кнопку «ОК»
DLG_YESNO_BTN	Рисует окно с кнопками «ДА» и «НЕТ»
DLG_CANCEL_BTN	Выводит только кнопку «ОТМЕНА»
DLG_NO_CAPTION	Флаг сообщает, что окно не содержит заголовка(не рисует сверху черную полосу с текстом). Может комбинироваться с другими значениями, например: dialog1.dlg_type = DLG_YESNO_BTN | DLG_NO_CAPTION;

Выделение активной кнопки:

btn_select	
DLG_SEL_NOSEL	Без выделения кнопок
DLG_SEL_OK	Выделяет кнопку «ОК»
DLG_SEL_CANCEL	Выделяет кнопку «ОТМЕНА»
DLG_SEL_YES	Выделить «ДА»
DLG_SEL_NO	Выделить «НЕТ»

В диалоге может быть выделена только одна кнопка. Выделение производится инвертированием цвета кнопок(dialog1.btn_select = DLG_SEL_OK;):



Пример использования:

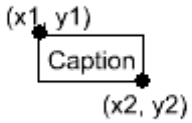
```
void test_dialog(void)
{
	struct DLG_STRUCT dialog; //декларация структуры свойств диалога
	dialog.x1 = 1;
	dialog.y1 = 1;
	dialog.x2 = 94;
	dialog.y2 = 66;
	dialog.dlg_type = DLG_YESNO_BTN;
	dialog.btn_select = DLG_SEL_YES;
	dialog.caption = "Настройка";
	scr_cls(); //очистка экрана
	gui_dialog( &dialog ); //нарисовать диалог
	scr_update(); //передать содержимое графического буфера в экран
}
```

Button – обычная кнопка с текстом.

Структура:

```
struct BTN_STRUCT
	{
	unsigned char x1;
	unsigned char y1;
	unsigned char x2;
	unsigned char y2;
	unsigned char select;  //выделена ли кнопка (TRUE/FALSE)
	const char *caption; //надпись на кнопке
	};
```

Образ кнопки(прямоугольник) рисуется независимо от ширины текстового поля ‘caption’, поэтому важно правильно указывать ее координату x2 (y2 считается автоматом):



Выделение производится инверсией цвета(button.select = TRUE):

Check Box – это кнопка-флажок.

Структура:

```
struct CHK_STRUCT
    {
    unsigned char x1;
    unsigned char y1;
    unsigned char select;      //выделена – не выделена
    unsigned char value;       //TRUE = отмечена, FALSE = не отмечена
    const char *caption;       //надпись
    };
```

Выделение:

select	Вид
TRUE	☒ Text
FALSE	☒ Text

Значение:

value	Вид
TRUE	☒ Text
FALSE	☐ Text

Vertical Scroll Bar – вертикальная полоса прокрутки.

Структура:

```
struct VSCRL_STRUCT
    {
    unsigned char x1;      //start
    unsigned char y1;
    unsigned char x2;      //end
    unsigned char y2;
    unsigned char max;     //максимальное значение
    unsigned char value;   //текущее значение
    unsigned char select;  //выделение
    };
```

Координаты:

max - максимальное значение шкалы.

value - текущее положение ползунка на шкале(0…max).

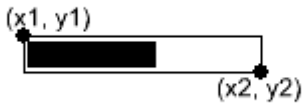
select – выделение ползунка (если select = TRUE, это свойство может подглючивать):

Horizontal Progress Bar – горизонтальный индикатор выполнения задания.

Структура:

```
struct HPRGS_STRUCT
    {
    unsigned char x1;
    unsigned char y1;
    unsigned char x2;
    unsigned char y2;
    unsigned char max;
    unsigned char value;
    };
```

Координаты:



Может немного подглючивать если (y2-y1)<10.

max – максимальное значение индикатора.

value – текущее значение индикатора(0…max).

List – список пунктов.

Структура:

```
struct LIST_STRUCT
    {
    unsigned char x1;
    unsigned char y1;
    unsigned char item_num; //число пунктов в списке
    unsigned char item_len; //длина каждого пункта в списке в символах+1(т.е. + символ 0)
    unsigned char item_select; //номер пункта в списке, который надо выделить
    };
```

Предполагается, что отображаемый список содержит небольшое количество пунктов и не выходит за пределы экрана.

Помимо этой структуры в функцию рисования списка также передается указатель на массив со строками списка(строки располагаются во Flash).

Пример:

```
#define MAIN_MNU_ITEMNUM       4
#define MAIN_MNU_ITEMLEN       10
const char main_mnu_items[MAIN_MNU_ITEMNUM][MAIN_MNU_ITEMLEN]={
"Таймер   ",
"Будильник",
"Настройка",
"Выход    "
};
    struct LIST_STRUCT list;
    list.x1 = 10;
    list.y1 = 13;
    list.item_num = MAIN_MNU_ITEMNUM;
    list.item_len = MAIN_MNU_ITEMLEN;
```

```
scr_cls();
list.item_select = i;
gui_list(main_mnu_items, &list);
scr_update();
```

Analog Clock – аналоговые часы(т.е. часы со стрелками), отображают время в 12-часовом формате. Автоматически преобразуют 24-часовое значение в 12-часовое.

Структура:

```
struct CLK_STRUCT
    {
    unsigned char cx;      //center X
    unsigned char cy;      //center Y
    unsigned char hour;
    unsigned char min;
    unsigned char sec;
    };
```

cx, cy – это координата центра часов на экране, остальное думаю и так понятно:).

Text Clock – текстовые(цифровые) часы, отображают время в 24-часовом формате.

Структура:

```
struct TXT_CLK_STRUCT
    {
    unsigned char x1;
    unsigned char y1;
    unsigned char hour;
    unsigned char min;
    unsigned char sec;
    unsigned char item_select;     //режим выделения
    };
```

x1, y1 – это начальная координата вывода на экран.

Режим выделения:

item_select	Описание	Вид
CLOCK_SELNO	Без выделения	12:34:56
CLOCK_SELHOUR	Выделить часы	12:34:56
CLOCK_SELMIN	Выделить минуты	12:34:56
CLOCK_SELSEC	Выделить секунды	12:34:56

Tashkent-2011