Updated 13/9/2010 Version 2.4a

omnima
Advanced lighting systems
Easy to use Seral/USB API
Programmable QVGA LCD Display
Click here to find out more ...

Programmable
NXP LPC21xx Solomon SSD192x
LCD Display / Control Panel

Technical overview

New Serial API version 2 documentation
13 September 2010

Omnima Limited, 176 Kennington Road, Oxford OX1 5PG
Company reg. 06038874, Tel. 08458692601

1. Technical details

- NXP LPC2103 based board
- Solomon SSD1926 - display controller
- 3.5” QVGA 320x240 32bit colour TFT LCD panel
- 16+5 GPIO general purpose I/O ports / SPI or i2C interface
- 1x 16bit Indirect SSD1926 bus interface
- SD card port
- 256KB RAM
- Enclosure size: 100mm x 85mm
- RoHS / Leadfree

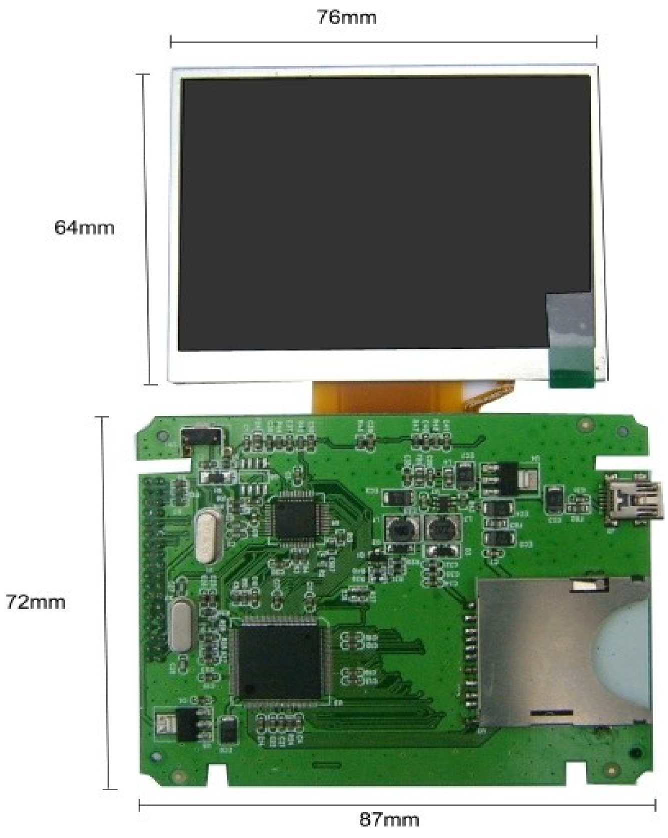
2. Main features

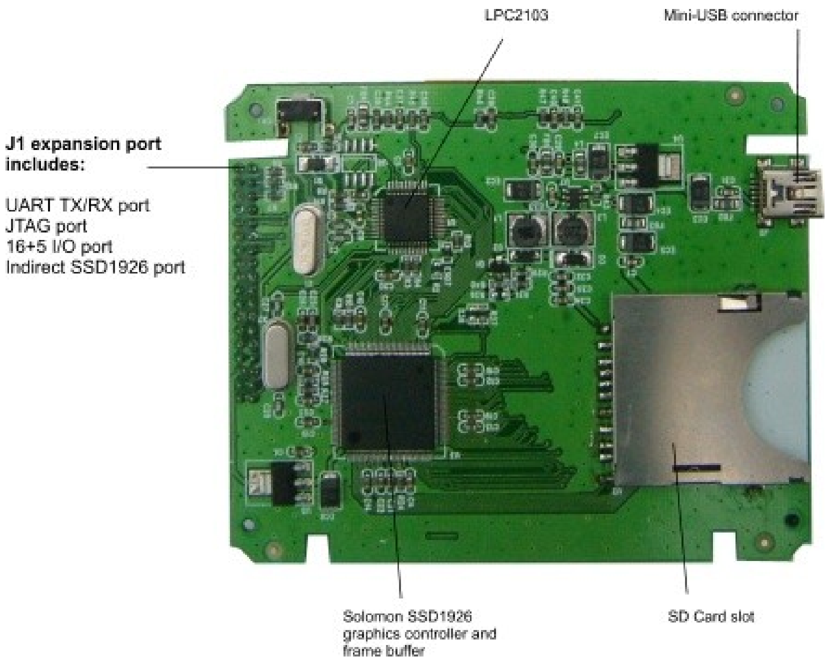
- Programmable LCD display
- USB port powered
- Solomon advanced graphics controller with JPEG decompression
- Expansion slot including 16+5 GPIOs for SPI / I2C connectivity
- Compact case with room for expansion PCBs
- Cost effective
- Simple to use serial/USB command-line interface
- Connects to all Windows and Linux computers
- SDK with complete C code available (uses GNU ARM GCC compiler)

3. Package contents

- LCD display panel, driver PCB and plastic enclosure
- USB connectivity cable
- CD-Rom including Windows driver for the USB interface adapter

4. Board layout





CPU

CPU is an LPC2103 ARM7-based MCU. The CPU is clocked at 12MHz, internally runs at 60MHz.

Graphics controller

The SSD1926 is an advanced display controller. It has a 2D graphics engine featuring panning, scrolling, rotation, and line, rectangle and ellipse drawing. There are bit block transfers, and a hardware JPEG engine. The controller supports up to 320x240 pixels in 32-bit colour.

5. Input/Output lines

J1 pinheader is a standard 36 position 2mm pitch connector and exposes the following I/O and power supply lines:

1	Mini-USB connector D+	2	UART RX signal, LPC P0.1
3	Mini-USB connector D-	4	UART TX signal, LPC P0.0
5	+1.8V	6	+5V
7	+3.3V	8	GND
9	DB0 – LPC P0.30 (TDI), SSD DB0	10	DB1 – LPC P0.31 (TDO), SSD DB1
11	DB2 – LPC P0.2, SSD DB2	12	DB3 – LPC P0.3, SSD DB3
13	DB4 – LPC P0.4, SSD DB4	14	DB5 – LPC P0.5, SSD DB5
15	DB6 – LPC P0.6, SSD DB6	16	DB7 – LPC P0.7, SSD DB7
17	DB8 – LPC P0.8, SSD DB8	18	DB9 – LPC P0.9, SSD DB9
19	DB10 – LPC P0.10, SSD DB10	20	DB11 – LPC P0.11, SSD DB11
21	DB12 – LPC P0.12, SSD DB12	22	DB13 – LPC P0.13, SSD DB13
23	DB14 – LPC P0.14, SSD DB14	24	DB15 – LPC P0.15, SSD DB15
25	~RD – LPC P0.16, SSD ~RD	26	~WR – LPC P0.21, SSD ~ WR
27	~CS – LPC P0.23, SSD ~CS	28	~D/C – LPC P0.24, SSD ~D/C
29	RST_SSD – LPC P0.25, SSD Reset signal	30	LPC P.017 – i2c SCL signal
31	LPC P0.18 - i2c SDA signal	32	LPC P.027 i/0 pin (TRST)
33	LPC P0.28 i/o pin (TMS)	34	LPC P0.29 (TCK)
35	n/c	36	n/c

6. Power supply

The LCD is powered using a standard mini-USB connector and DC power supply of +5V. For embedded applications +5V and GND can be connected to J1 Pin 6 (+5V) and Pin 8 (GND).

7. Serial/USB command-line interface

Connection to a host PC or MCU

a) Using the USB to UART cable

- Connect the USB cable to a host PC
- On computers running Windows you'll be prompted to install a PL2303 device driver. You can download the driver from http://omnima.co.uk/docs/PL2303USBDriver.zip
- Linux kernel will usually be pre-configured to include the Prolific driver and is preinstalled.
- Serial connection configuration is 115200bps,8,N,1, no hardware handshake.
- To connect to the LCD via the serial link you can use olcdtool provided by Omnima (see section 8), HyperTerminal on Windows or minicom on Linux.

b) Connect a serial/UART/RS232 custom cable to the LCD

- Connect RX (data coming into the LCD) line to J1 Pin 2
- Connect TX (date going out of the LCD) line to J1 Pin 4
- Connect Signal ground (GND) to J1 Pin 8

The acceptable serial communication signal levels are from 3.3V to 5V including most serial/UART/RS232 protocols. Any RS232 cables operating using 12V should use a level shifter to convert to the operating range of 3.3V to 5V.

8. Serial connection terminal and the firmware upload tool

An enhanced version of the open-source lpc21isp called olcdtool program is used for programming and communicating with the LCD.

olcdtool is a cross-paltform tool with source code and binaries available and running on Linux, Mac OS X and Windows. olcdtool source code is hosted on http://code.google.com/p/omnimalcd/ and includes examples and cross-paltfom code for interfacing to the LCD from a host PC (see olcd.c, serial.c).

- Terminal mode

You can start olcdtool without any command-line parameters:

```
olcdtool <Enter>
```

The program will try to detect the communication port to which the LCD is connected to and start the terminal. The OLCD command prompt should be displayed. If it is not, press <Enter> once.

If for some reason olcdtool is not able to detect the communication port automatically you can pass additional command-line options to the program:

```
olcdtool -term -port=/dev/ttyUSB0 -baud=115200 <Enter>
olcdtool -term -port=com5 <Enter>
```

- Programming mode
olcdtool can be used to upload new firmware versions to the LPC21xx microcontroller. The command-line examples are:

```
olcdtool -upload=<pathtohexfile>     {detects com port and uploads firmware}
olcdtool -port=/dev/ttyUSB0 -upload=<pathtohexfile> {uses given com port and uploads}
olcdtool -port=com5 -term -upload=<pathtohexfile> {uploads firmware and starts terminal}
```

For more detailed description and a full list of olcdtool command-line options start olcdtool with the -help option.

9. SSD192x video mode and memory layout

The video mode is set to 16bit RGB color. This means that each pixel takes two bytes in video memory:

Gl	Gl	Gl	B	B	B	B	B	R	R	R	R	R	Gh	Gh	Gh
16							8	7							0

For example, the brightest blue color in the video memory would be recorded as 0x1E00.

Each row of 320 pixels takes 640 bytes and a whole 320x240 screen takes 153600 bytes.

The video memory can be accessed using the #@FileToSSD command, SSDREGxx commands and indirectly by making use of the hardware acceleration functions using the 2D line, rectangle and BitBlt commands. There are four drawing surfaces available.

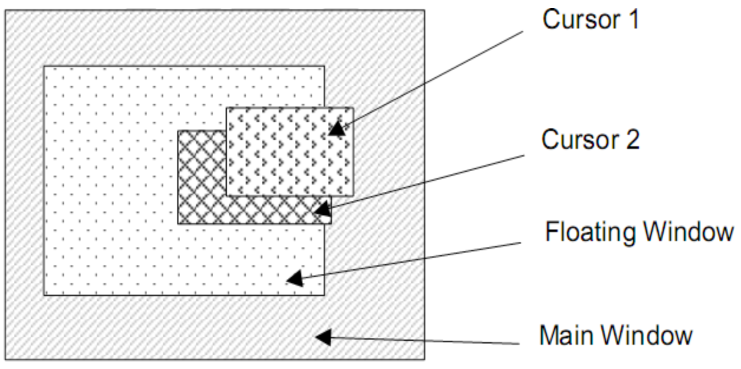


Fig. 1 Display Precedence

The main window is the main drawing surface. The main window (MW) is the lowest in the order of visual layers. The main window is overlapped by the floating window (FW) when FW is enabled. Further, the cursor 1 (CR1) overlaps the main and floating windows, while cursor 2 (CR2) is the topmost drawing surface.

Some of the Serial API commands and the SSD API require the user to specify pointers to memory locations in the SSD frame buffer.

By default the memory is allocated as listed below.

```
-----------------------------------------------------------
                 SSD192x RAM memory allocation
-----------------------------------------------------------
1) Main window:     320x240x16bit - 150kb    or 153600 bytes
2) File r/w buffer:                   4kb    or 4096   bytes
3) Float window:    244x128x16bit -  61kb    or 62464  bytes
4) Scratch window:  320x32 x16bit -  20kb    or 20480  bytes
5) Free(cursor)mem: 320x240x2bit  -  18.75kb or 19200  bytes
6) Free memory:                      2.25kb  or  2304  bytes
-----------------------------------------------------------
   Total memory available:          256kb    or 262144 bytes
-----------------------------------------------------------
```

1) The main window memory is always in use (unless the SSD registers are updated to modify this_ To draw on the main window use the MW target option for the 2D drawing routines. For the #@FileToSSD command you can specify 0 as the start memory location. See the notes for using the #@FileToSSD command later in this document.

2) File r/w buffer of 4kb is used for fast DMA SD card reading and writing. If SD read/write operations targeting the SSD video memory are not in progress this memory can be used for other purposes.

3) The floating window overlays the main window and will obscure the area of the main window. To draw on the floating window use the FW target option for the 2D drawing routines. For the #@FileToSSD specify 157696 (or 0x26800 hex) as the start memory address.
The floating window is displayed by calling #@FloatWin Show. The floating window can be repositioned by calling #@FloatWin Move {x} {y}.

4) The scratchpad window is used for hardware font rendering. It is automatically updated when #@SetFont {fontno} {scaling} is called. If font drawing is not in use the scratchpad window can be used for any other drawing purposes.

5) The cursor memory window is used when the second cursor CR2 is displayed. Otherwise this memory can be used for any other drawing purposes.

9. Serial API reference

Destination surfaces

MW - main window (320x240 fixed size, 16bit colour)
FW - floating window (default size: 244x128, 16bit colour)
SC - scratchpad (320x32, 16bit colour – experimental)

Cursors

CR1 - cursor 1
CR2 - cursor 2

Built-in fonts

There are three built-in fonts available.

Font 1 is a 8x8 font and all standard ASCII characters are included alongside a range of symbols and windings.

Font 2 is a 16x10 font. It renders upper case characters, numbers and a few symbols

Font 3 is a 16x32 numerical font and includes numbers 0-9, decimal point and a minus sign.

Commands

- Commands are case sensitive
- Separators between parameters can be: (,) comma, (,) colon, (;) semicolon, ( ) space
- Parameters can be entered as decimal or hexadecimal numbers.
- Leading 0x is required for hexadecimal numbers i.e. use 0xff to specify 255 in decimal
- Commands should be followed by a <CR><LF> byte sequence

System commands

#@Reboot
Restarts the LCD

#@EnterISP
Restarts the LCD and enters the programming mode for 2 minutes. The LCD will restart automatically after 2 minutes. Use the olcdtool tool (the enhanced Omnima version) or FlashMagic (http://www.flashmagictool.com/) to upload new firmware to the LCD using the same serial communication cable.

#@Display On
#@Display Off
Turns the display on/off

#@UARTBaud {dll} {dlm} {fdr} {vpbdiv}
Sets the speed of the LCD serial port
Examples:

9600bps
#@UARTBaud 0xD9 0 0x54 1

19200bps
#@UARTBaud 0xB3 0 0xB1 1

115200bps
#@UARTBaud 0x1A 0 0x41 1

230400bps
#@UARTBaud 0x0C 0 0xE5 1

1152000bps
#@UARTBaud 0x3 0 0xC1

Window surfaces
There are four drawing surfaces supported by the SSD graphics controller.

Main window
The main window is fixed surface and is defined to be 320x240 pixels in 16bit colour.
All 2D drawing routines can reference this surface as their source or destination as MW.

Floating window
The floating window overlaps the main window surface. It can be optionally shown or hidden and its' size and position changed. All 2D drawing routines can reference this surface as their source or destination as FW.

#@FloatWin Open x,y,width,height
[x,y - absolute location, width, height - specify the size of the window]
Example: #@FloatWin Open 10 10 200 200
Allocates new window (hides any existing floating window)

#@FloatWin Show
Enables and displays the floating window on the screen

#@FloatWin Hide
Hides the floating window

#@FloatWin Move x,y
Example1: #@FloatWin Move -10 -10
Example2: #@FloatWin Move 10 10
Moves the floating window in either positive or negative direction

Cursors

Cursors are intended to show small areas of text or graphics.
At present, only text and the characters from the built-in font can be rendered and shown as cursors.

Cursors can be shown, hidden, moved and their blink ratio specified.
Cursors will overlap the main and floating windows when displayed on the screen.

#@Cursor [CR1 or CR2] Open x,y,width,height
Example: #@Cursor CR1 Open 50 50 320 20
Allocates new cursor (hides any previously displayed cursor)

#@Cursor [CR1 or CR2] Show
Example: #@Cursor CR1 Show
Enables and displays the cursor (CR1 or CR2) on the screen

#@Cursor [CR1 or CR2] Hide
Example: #@Cursor CR1 Hide
Hides the specified cursor (CR1 or CR2)

#@Cursor [CR1 or CR2] Move x,y,totalTime,visibleTime
Example: #@Cursor CR1 Move 10 10 100 60
Moves the specified cursor (CR1 or CR2) to the relative x,y location. The blink ratio for the cursor is specified by the last two parameters. The first blink parameter is the total display time. The second parameter specifies the proportion of total time during which the cursor is made visible.

#@Cursor [CR1 or CR2] Text x,y,text
#@Cursor CR1 Text 0 0 Hello!
Draws characters specified in this command on the cursor CR1 or CR2 surface

Drawing on surfaces

#@FrColor Red,Green,Blue,Alpha
Defines foreground colour for subsequent drawing operations
Example: #@FrColor 255 255 255 0 - white colour

#@BkColor Red,Green,Blue,Alpha
Defines background colour for subsequent drawing operations
Example: #@BkColor 255 0 0 0 - red colour

Sets the foreground and background drawing colour (Red,Green,Blue,Alpha)
Alpha can be used to render transparent font in which case BkColor Alpha value can be set to 255.

#@SetFont [Font num:1,2 or 3] [scaling:1,2,3...]
Example: #@SetFont 1 1 select Font 1 and prepares the font for drawing by selecting the current foreground and background colours. Once the SetFont command is complete the colour of the font will be preserved until the next call to SetFont irrespectively of any other calls to FrColor or BkColor.

#@Text [MW or FW] [Hor,Ver,H2,V2] x,y,text
Example: #@Text MW Hor 50 140 Hello World!
Draws text on (MW,FW) located at coordinates x,y and displays the given text. The font type can be selected by calling SetFont. The orientation of the text is selected by the Hor, Ver, H2, V2 parameters. Hor selects standard horizontal text drawing. Ver draws text rotated by 90deg CCW, H2 rotation is 180deg, V2 rotation is 90deg CW.

#@Arc [MW or FW] x,y,widht,height[,startAngle,endAngle] - the arc start and end angle are optional
Example: #@Arc MW 150 20 5 5 0 360 - draws using foreground colour

#@Line MW x1,y1,x2,y2
Example: #@Line MW 10 10 100 10 - draws using foreground colour

#@Rect [MW or FW] x1,y1,width,height
Example: #@Rect MW 10 10 100 10 - draws using background colour, solid fill

#@Blt [Source: MW or FW] x1,y1,width,height [Destination: MW or FW] x,y, [destWidth,destHeight]
Copies images from source to destination, optionally stretching the image to fit the destination width and height
Example1: #@Blt MW 10 10 20 20 MW 50 50 50 50 - MW to MW copy, optional [50,50] provided to stretch the image
Example2: #@Blt FW 10 10 20 20 MW 50 50 - FW to MW copy, no stretch

#@Frame [Source: MW or FW] x1,y1,width,height,frame_width,grade
Draws a frame with a gradient rectangle. Fill colour depends on the #@BkColor if Alpha is not equal to 255, otherwise frame is drawn without the inner area being painted
Example: #@Frame MW 10 10 100 100 10 10 - draws using foreground colour and background colour

SD card reading and writing
The SD card is initialised when the LCD is powered up. If the SD card is ejected or replaced the LCD will need to be restarted. Contact info@omnima.co.uk if you require to hot swap SD cards in your application.

#@LS {part} {dir}
Lists the contents of the SD card and the specified directory path. {part} is reserved for future used when it will specify the file system partition number. {dir} is optional, if not provided the command will display the contents of the root directory.
Example: #@LS 0 \ - lists the contents of the root directory

#@Fopen {mode} {filepath}
Opens a file for reading of writing
{mode} is set to 0x11 for reading and to 0xA for writing
{filepath} is the path to the file to be written
Example: #@Fopen 0xA newtext.txt

#@Fwrite {count} {contents}
Writes {contents} to the currently opened file. {count} can be set to either 0 in which case the length of the data to be written is automatically detected or it can be set to a specific number.
Example: #@Fwrite 0 Testing

#@Fread {flag:HX or BI} {count}
Read data from the currently opened file and outputs the data to the console.
{flag} specifies how the data from the file is printed: BI – read bytes are printed as characters, HX – read bytes are printed as HEX.
{count} specifies the number of bytes to be read from file. If set to 0 the entire file will be read.

#@Fclose
Closes the file after the file has been opened for reading or writing. Call #@Fclose before #@Fopen is a file has already been opened for reading or writing. Calling #@Fclose ensures that all data has been written to the SD card.

#@FileToSSD {addr} {pathtofile}
Reads data from the given file directly to the SSD frame buffer memory using fast DMA memory transfer.
Example: #@FileToSSD 0 omnima.raw

To ensure that the video memory is updated as fast as possible, the transfer of the data is performed in block sizes of 128Kb, 64Kb, 32Kb, 16Kb, 8Kb and 4Kb depending on the size of the input file. The smallest black size is 4Kb and for this reason appropriate video memory should be available for the read operation. For example, to read in 153600 byte file, a total of

To load a image file from the SD card directly to the Main window a file containing 320x240 pixels should be written to a file with a total of 153600 bytes.

Smaller files can also be read using this command but the size of the file should reflect this. For example, if the size of the image is 320x50, the file size should be 32000 bytes.

To generate files suitable for loading using FileToSSD you can make use of the oimage tool. This tool can load all popular image file formats such as jpg, gif, bmp and more, and save the file in the SSD compatible format.

Terminal

The on-board terminal function allows the host communication to be displayed directly on the LCD.

#@Term {window:MW or FW} {flag:On or Off}
Turns terminal function on or off and specifies the target window. The current foreground and background colors as specified by FrColor and BkColor are recorded for used by the terminal.
Example: #@Term MW On

#@Cls
Clears the terminal window screen and positions the cursor in the top left corner.

Responses from the LCD

#@OKEYDN:{keynum} - sent when key is pressed
#@OKEYUP:{keynum} - sent when key is released

where {keynum} is 1-5

Touch screen version only (available May 2011)

#@OTCH:{x},{y} - sent when screen touch is detected

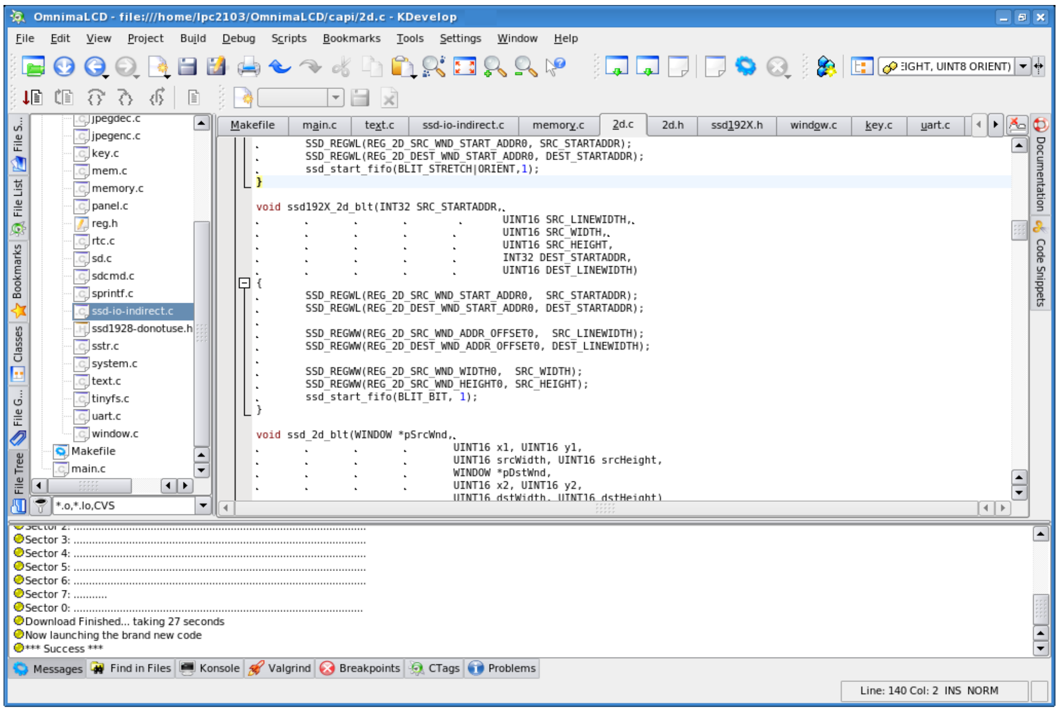
where {x} is 0-319 and {y} is 0-239

10. The C programming API and LPC21xx embedded software development

Starting from version 2 of the Omnima LCD, C source code and SDK is available from Omnima although the Professional SDK edition is compatible and can target version 1 hardware.

The C API for the LCD version 1 was based on a proprietary C compiler, preventing many enthusiasts from developing and contributing to the development of the source code due to the cost of the compiler and the difficulty in setting up the development environment.

The C API source code has now been ported to the GNU GCC ARM compiler. The GCC based SDK is now available as a Linux (Slackware) virtual machine running under Virtual Box and VMWare and as such it's simple to use on a variety of platform such as Windows, Mac and Linux.

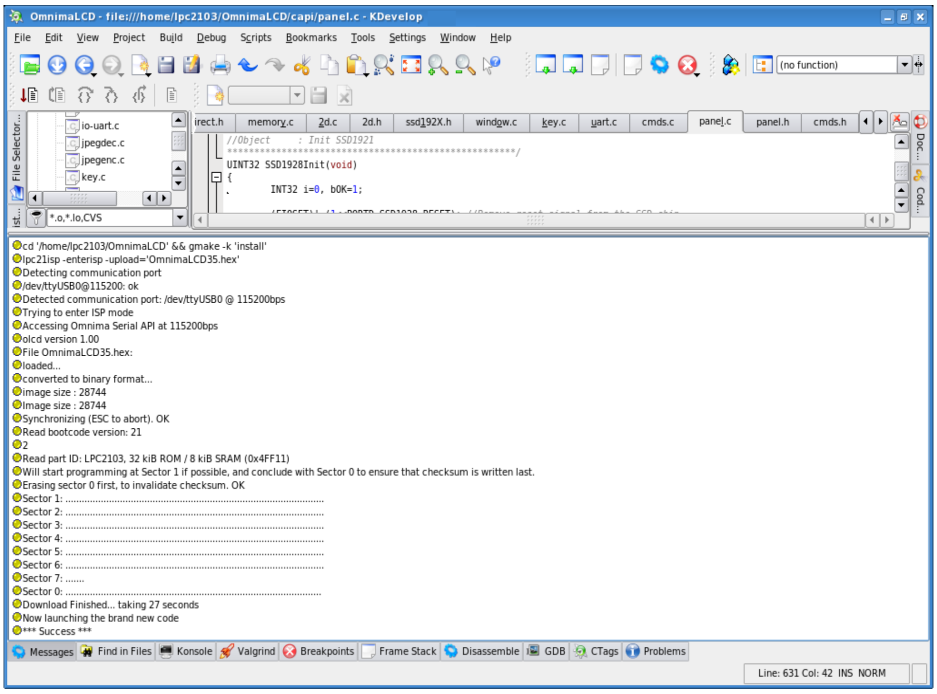

OmnimaLCD - file:///home/lpc2103/OmnimaLCD/capi/2d.c - KDevelop
File Edit View Project Build Debug Scripts Bookmarks Tools Settings Window Help
EIGHT, UINT8 ORIENT)
jpegdec.c
jpegenc.c
key.c
mem.c
memory.c
panel.c
reg.h
rtc.c
sd.c
sdcmd.c
sprintf.c
ssd-io-indirect.c
ssd1928-donotuse.h
sstr.c
system.c
text.c
tinyfs.c
uart.c
window.c
Makefile
main.c
.o,.lo,CVS
Makefile main.c text.c ssd-io-indirect.c memory.c 2d.c 2d.h ssd192X.h window.c key.c uart.c
File S... File List Bookmarks Classes File G... File Tree
Documentation Code Snippets
Messages Find in Files Konsole Valgrind Breakpoints CTags Problems
Line: 140 Col: 2 INS NORM

```
        SSD_REGWL(REG_2D_SRC_WND_START_ADDR0, SRC_STARTADDR);
        SSD_REGWL(REG_2D_DEST_WND_START_ADDR0, DEST_STARTADDR);
        ssd_start_fifo(BLIT_STRETCH|ORIENT,1);
}

void ssd192X_2d_blt(INT32 SRC_STARTADDR,
                                        UINT16 SRC_LINEWIDTH,
                                        UINT16 SRC_WIDTH,
                                        UINT16 SRC_HEIGHT,
                                        INT32 DEST_STARTADDR,
                                        UINT16 DEST_LINEWIDTH)
{
        SSD_REGWL(REG_2D_SRC_WND_START_ADDR0,  SRC_STARTADDR);
        SSD_REGWL(REG_2D_DEST_WND_START_ADDR0, DEST_STARTADDR);

        SSD_REGWW(REG_2D_SRC_WND_ADDR_OFFSET0,  SRC_LINEWIDTH);
        SSD_REGWW(REG_2D_DEST_WND_ADDR_OFFSET0, DEST_LINEWIDTH);

        SSD_REGWW(REG_2D_SRC_WND_WIDTH0,  SRC_WIDTH);
        SSD_REGWW(REG_2D_SRC_WND_HEIGHT0, SRC_HEIGHT);
        ssd_start_fifo(BLIT_BIT, 1);
}

void ssd_2d_blt(WINDOW *pSrcWnd,
                                UINT16 x1, UINT16 y1,
                                UINT16 srcWidth, UINT16 srcHeight,
                                WINDOW *pDstWnd,
                                UINT16 x2, UINT16 y2,
                                UINT16 dstWidth, UINT16 dstHeight)
```

```
Sector 2: ..................................................................
Sector 3: ..................................................................
Sector 4: ..................................................................
Sector 5: ..................................................................
Sector 6: ..................................................................
Sector 7: ..........
Sector 0: .................................................................
Download Finished... taking 27 seconds
Now launching the brand new code
*** Success ***
```

KDEvelop integrated development environment (IDE)
running under Slackware Linux

The SDK provides the following features and components:
- Slackware Linux virtual machine (provides Windows-like KDE desktop environment)
- Pre-installed GCC ARM compiler
- olcdtool - enhanced firmware upload tool and serial link terminal
- KDevelop integrated development environment
- Integrated firmware upload tool
- Comprehensive C API source code
- One button build procedure
- One button firmware upload procedure


OmnimaLCD - file:///home/lpc2103/OmnimaLCD/capi/panel.c - KDevelop
File Edit View Project Build Debug Scripts Bookmarks Tools Settings Window Help
(no function)
io-uart.c
jpegdec.c
jpegenc.c
key.c
.o,.lo,CVS
irect.h memory.c 2d.c 2d.h ssd192X.h window.c key.c uart.c cmds.c panel.c panel.h cmds.h
Messages Find in Files Konsole Valgrind Breakpoints Frame Stack Disassemble GDB CTags Problems
Line: 631 Col: 42 INS NORM

```
//Object      : Init SSD1921
***************************************************/
UINT32 SSD1928Init(void)
{
        INT32 i=0, bOK=1;
```

```
cd '/home/lpc2103/OmnimaLCD' && gmake -k 'install'
lpc21isp -enterisp -upload='OmnimaLCD35.hex'
Detecting communication port
/dev/ttyUSB0@115200: ok
Detected communication port: /dev/ttyUSB0 @ 115200bps
Trying to enter ISP mode
Accessing Omnima Serial API at 115200bps
olcd version 1.00
File OmnimaLCD35.hex:
loaded...
converted to binary format...
image size : 28744
Image size : 28744
Synchronizing (ESC to abort). OK
Read bootcode version: 21
2
Read part ID: LPC2103, 32 kiB ROM / 8 kiB SRAM (0x4FF11)
Will start programming at Sector 1 if possible, and conclude with Sector 0 to ensure that checksum is written last.
Erasing sector 0 first, to invalidate checksum. OK
Sector 1: ..........................................................................................
Sector 2: ..........................................................................................
Sector 3: ..........................................................................................
Sector 4: ..........................................................................................
Sector 5: ..........................................................................................
Sector 6: ..........................................................................................
Sector 7: .......
Sector 0: .........................................................................................
Download Finished... taking 27 seconds
Now launching the brand new code
*** Success ***
```

olcdtool programming tool is integrated into the IDE
for quick and easy firmware upload

Three SDK editions are available:

1. Starter C API SDK
Includes Slackware Linux VM, KDevelop IDE, GNU GCC compiler, C API (excludes Serial API C source code).

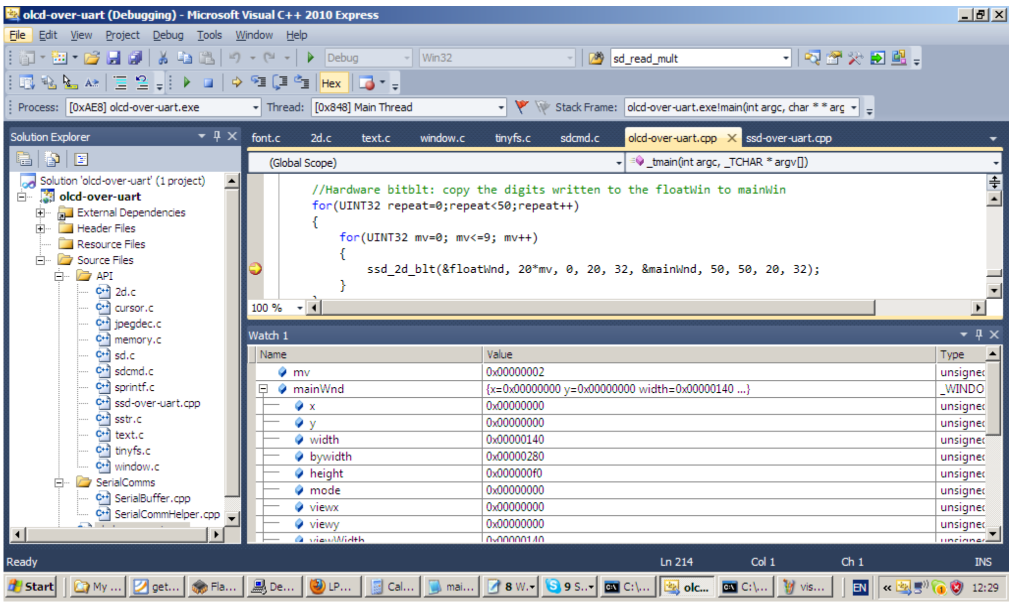
2. Standard C API SDK
Same as the Standard SDK but includes the full Serial API C source code.

Continued on next page ...

3. Professional SDK

Same as the Standard SDK but includes the tools and protocols to enable development and debugging of Solomon SSD192xx code directly on the host PC. This allows the developer to step through the code on the host PC rather than having to upload code to the microcontroller first.

The Professional SDK also includes the Visual C++ 2010 Express port of the Omnima SSD192X API allowing the SSD192X code to run on the host Windows PC but at the same time execute directly on the SSD192X hardware. This considerably shortens development time compared to the conventional compile/firmware upload/run embedded development approach.



Process: [0xAE8] olcd-over-uart.exe Thread: [0x848] Main Thread Stack Frame: olcd-over-uart.exe!main(int argc, char * * arg

```
//Hardware bitblt: copy the digits written to the floatWin to mainWin
for(UINT32 repeat=0;repeat<50;repeat++)
{
    for(UINT32 mv=0; mv<=9; mv++)
    {
        ssd_2d_blt(&floatWnd, 20*mv, 0, 20, 32, &mainWnd, 50, 50, 20, 32);
    }
```

Watch 1

Name	Value	Type
mv	0x00000002	unsignec
mainWnd	{x=0x00000000 y=0x00000000 width=0x00000140 ...}	_WINDO
x	0x00000000	unsignec
y	0x00000000	unsignec
width	0x00000140	unsignec
bywidth	0x00000280	unsignec
height	0x000000f0	unsignec
mode	0x00000000	unsignec
viewx	0x00000000	unsignec
viewy	0x00000000	unsignec

Professional edition SDK allows rapid development of SSD192X code by running on the host PC and simultaneously executing on the LCD hardware

11. Slackware 13 LCDDev virtual machine installation and usage

Running the VM in VMWare

Recommended hardware:
Windows 2000/XP, Mac OS X or Linux, 2GB RAM

You can download VMWare Player from the following link:
https://www.vmware.com/tryvmware/?p=player

Assuming you've installed VMWare Workstation, Player or Server on your
host system follow the steps below:

1. Unzip the VM archive to a directory on your computer
2. Start VMWare and select 'Open an existing virtual machine'
3. Start the VM
4. When connecting the LCD USB cable make sure that under the VM menu -> Removable devices, the Prolific USB-Serial Controller is connected i.e. select the Connect option

Running the VM in VirtualBox

You can downlaod VirtualBox from the following link:
http://www.virtualbox.org/wiki/Downloads

Assuming you've installed VirtualBox on your host system follow the steps below.

1. Unzip the VM archive to a directory on your computer
2. Select 'New' to setup a new VM, enter the name for the VM i.e. "Slackware 12 EMBDev", select "Linux 2.6" as the base OS, click Next
3. Select as least 512MB memory for the VM, click Next
4. Select an existing disk, click Existing ....
5. Click Add to add a vmdk file, select Slackware EMB.vmdk from the directory where you've extracted the zip file, close the dialog, click Next
6. Go to Settings, click Network, enable Adapter 1, select 'Intel PRO/1000 T Server' as the adapter, attached to Host Interface, select en1: AirPort on Mac OS X host or an alternative network adapter on a PC.
7. Start the VM

Running the Slackware VM for the first time

Once the Linux is started, log in using username Developer, no password
1. Type startx <Enter> to start the x-windows KDE graphical windowing system
2. Open a Konsole/Shell window - K -> Konsole - Terminal Program
3. Type ls /dev/ttyUSB* to confirm that the USB cable has been recognised. The USB cable should be listed as /dev/ttyUSB0 or /dev/ttyUSB1. If the USB cable is not detected, check that the VMWare or VirtualBox have connected the USB cable to the VM. In VMWare select the menu VM -> Removable Devices -> Prolific USB-Serial Controller and make sure that the option is

marked with a tick. If not select 'Connect', then repeat this step 8 to check that the USB cable has been recognised.
4. You can change the resolution of the desktop. The default 800x600 resolution ensures that the VM starts running on most systems but you can change the resolution to 1024x720 or above by selecting K->Computer->System Settings->Overview->General->Display-> change resolution
5. The keyboard layout is set to UK QUERTY by default. This can be changed by clicking on British flag in the taskbar or by changing the selection in K->Computer->System Settings ->Regional&Language->Keyboard Layout settings.
6. Start KDevelop from K->Search and enter kdevelop <ENTER> or click on the KDevelop 4 icon

KDevelop usage

Once KDevelop is running you can compile, build the firmware, edit the source files and upload the firmware within the IDE.

1. Open the project. The LCD C-Application should open as a default project. The title of the KDevelop will should display "C-application - KDevelop". If it doesn't, select Project->Open Recent->C-application
2. Build the project by selecting Project->Build Selection (F8)
3. Upload the firmware to the LCD by selecting Project->Install Selection
4. Explore the source files by clicking on the Project tab on the left hand side of the Kdevelop window
5. To connect to the LCD USB/serial port within KDevelop, click on the Konsole tab and enter 'olcdtool' this should detect the LCD automatically and connect to the serial line of the LCD. Press <Enter> once to get the OLCD: prompt displayed. You can then start sending Omnima LCD API commands to the LCD. Remember to press <Esc> to exit the terminal before uploading new firmware to the LCD by using Project -> Install Selected.

ISP programming of the LCD

The olcdtool tool is used to upload build firmware (OmnimaLCD35.hex) to the LCD. This is an enhanced version of the open source lpc21isp from available from sourceforge. The program was extended to provide automated detection of the USB/serial cable and also to enter the LPC2xxxx ISP programming mode automatically.

The default option is to detect the active USB/Serial port, enter the ISP mode and upload the firmware.

This will work in most situations but sometimes it will be necessary to modify the default behavior.

The default command is located in the project Makefile and this declares the following statement:

```
install:
        $(UPLOADTOOL) -upload='$(HEX)'
```

If olcdtool fails to detect the USB cable you can specify the port on the command line by adding:

-port=/dev/ttyUSB0 (or by substituting USB0 for the currently active port such as USB1)

If olcdtool fails to enter ISP mode automatically you can add the -donotenterisp option and enter the ISP mode manually by connecting the P0.14 to GND while the LCD is being powered up, than releasing the pin from the GND.

Useful Slackware Linux/KDE programs/commands

- Calculator: kcalc
- Visual diff program: kompare similar to windiff
- Text editor: kwrite
- Command prompt: konsole
- File system explorer: konqueror - press F9 to get folder tree and click on the Home or Root icon
- File system explorer: dolphin - another alternative file system explorer
- Integrated development environment: kdevelop
- Mount a Samba network share:
  mkdir /mountpoint - mountpoint is the directory where the share will be mounted, e.g. /tmp/share
  mount -t smbfs -o username=<name>,password=<passwd> //sambashare /mountpoint
- Mount a flash USB key:
  mkdir /mountpoint - mountpoint is the directory where the share will be mounted, e.g. /tmp/flash
  mount /dev/sdaX /mountpoint - where X is 1 to 9, do 'ls /dev/sda* -l' to find out available drives,
  Note: removable USB flash drives will be marked as plugdev when the -l option is passed to ls.
- Printing out the Linux boot log: dmesg

You can start these programs from K->Run menu.

Troubleshooting

1) Small or large fonts displayed in Slackware VM under VMWare
Change the font size in Control Center. You can start kcontrol from the command-line.

2) No network connection under VirtualBox
Select Intel PRO/1000 T Server (Host interface - connected to en1: AirPort under Mac OS X)
in the VM settings. Restart the VM.

3) No sound device is detected under VirtualBox
This still needs to be resolved.

4) No USB connectivity within VirtualBox
Power off the VM, go to Settings, then Ports, tick the Enable USB Controller option, then start the VM. On the host machine you may have to eject the USB flash volumes before they become available within VirtualBox -> Devices -> USB Devices.

12. Setting up development environment

12.1 Developing on Linux

- Download ARM EABI compiler from http://www.codesourcery.com/sgpp/lite/arm/portal/package7813/public/arm-none-eabi/arm-2010.09-51-arm-none-eabi-i686-pc-linux-gnu.tar.bz2
- Run: tar xvf arm-2010.09-51-arm-none-eabi-i686-pc-linux-gnu.tar.bz2
- Download http://code.google.com/p/omnimalcd/, uncompress and run make
- Uncompress OmnimaLCDSrc.zip to a directory
- Edit Makefile and update the GCCROOT variable to point to the GCC bin directory containing arm-none-eabi-gcc binary
- You can now run make to build the LCD hex firmware file
- Use olcdtool to upload the firmware to the LCD
- You can also open the project in Kdevelop

12.2 Developing on Mac OS X

- Download ARM EABI compiler from http://omnima.co.uk/docs/arm-none-eabi-osx.tar.gz
- Run: tar xvf arm-none-eabi-osx.tar.gz
- Download http://code.google.com/p/omnimalcd/, uncompress and run make
- Uncompress OmnimaLCDSrc.zip to a directory
- Edit Makefile and update the GCCROOT variable to point to the GCC bin directory containing arm-none-eabi-gcc binary
- You can now run make to build the LCD hex firmware file
- Use olcdtool to upload the firmware to the LCD
- You can also open the project in Xcode

12. 3 Developing on Windows

- Download ARM EABI compiler from http://www.codesourcery.com/sgpp/lite/arm/portal/package7814/public/arm-none-eabi/arm-2010.09-51-arm-none-eabi-i686-mingw32.tar.bz2
- Uncompress arm-2010.09-51-arm-none-eabi-i686-mingw32.tar.bz2 (you can use www.7-zip.org program to uncompress)
- Download http://code.google.com/p/omnimalcd/, uncompress and run make
- Uncompress OmnimaLCDSrc.zip to a directory
- Edit Makefile and update the GCCROOT variable to point to the GCC bin directory containing arm-none-eabi-gcc binary
- You can now run make to build the LCD hex firmware file
- Use olcdtool to upload the firmware to the LCD
- You can also open the project in Microsoft Visual C++ 2010 Express

13. Further information

For information on development guides and tools, please refer the following sources.

Latest information, updates and firmware versions	http://www.omnima.co.uk/forums/index.php?showforum=7
J1 pinheader specification and supplier info	http://www.samtec.com/standardboardtoboard/2mm_pitch.aspx
GNU 4.5.1 ARM compiler	Linux/Windows: http://www.codesourcery.com/sgpp/lite/arm/portal/release1592 Mac OS X: http://omnima.co.uk/docs/arm-none-eabi-osx.tar.gz
NXP LPC2103 datasheet	http://www.nxp.com/acrobat/datasheets/LPC2101_02_03_2.pdf
SSD1928 datasheet	http://omnima.co.uk/docs/SSD1928_1.0.pdf
SSD192x API Programming guide	http://www.omnima.co.uk/forums/index.php?act=attach&type=post&id=66
Sample LPC21xx C programme	http://linux-adm5120.svn.sourceforge.net/viewvc/linux-adm5120/lpc2103/blink/
PL2303USBDriver.zip Windows PL2303 driver	http://omnima.co.uk/docs/PL2303USBDriver.zip
olcdtool	http://code.google.com/p/omnimalcd/