DATASHEET

4D SYSTEMS

TURNING TECHNOLOGY INTO ART

4Display Shield with 2.2” Display

4Display-Shield-22

Document Date: 31st October 2012

Document Revision: 1.1

Contents

1. Description

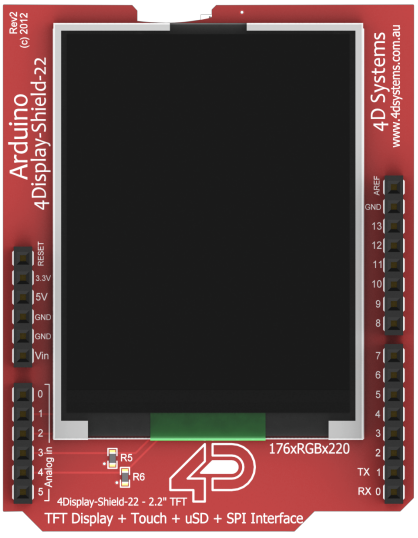
The 4Display-Shield-22 provides an easy way of connecting a display up to your Arduino project.

The shield features a 176 x 220 LCD-TFT display with a SPI interface to the Arduino, which includes a 4-wire resistive touch screen.

The shield also features a uSD Card socket to provide the Arduino with a means to store external data. The interface to the uSD Card socket is SPI.

The communication interface between the 4Display-Shield-22 and the Arduino is a combination of the SPI bus, and the analog/digital pins on the Arduino.

Both the TX and RX signals are unused, so these can be used with other devices in your Arduino project.



4Display-Shield-22 Arduino Shield

2. Features

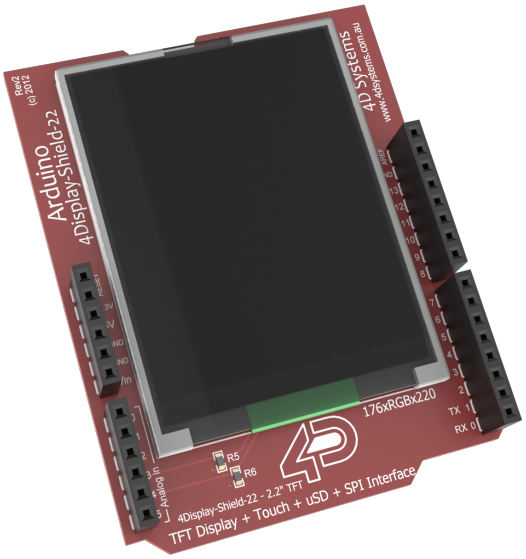
- On-board micro-SD memory card adaptor for storing of data, for the Arduino to read/write.
- Powered from the Arduino board, uses the 5V supply pin.
- 176 x 220 resolution, 262K true to life colours, LCD-TFT screen with SPI Interface, and resistive 4-wire touch screen.
- 2.2” diagonal size, Viewing Area: 49.4mm x 36.7mm.
- LED back lighting with greater than 150° viewing angle.
- RoHS Compliant.

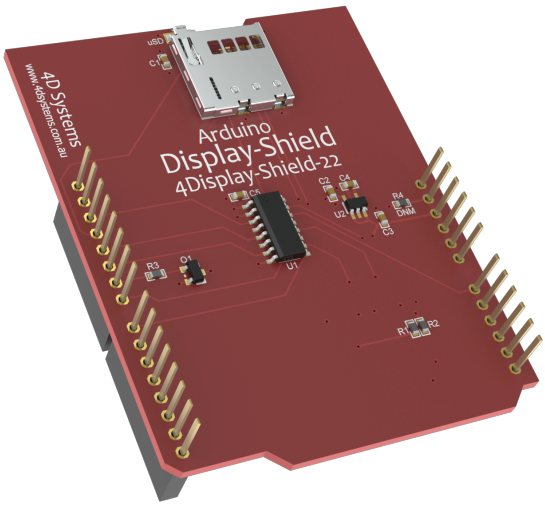
3. Hardware Description

3.1. 4Display-Shield-22 Arduino Shield

The 4Display-Shield-22 is an Arduino Shield featuring a 2.2” LCD-TFT display with resistive touch screen.
The 4Display-Shield-22 provides a SPI interface to the Arduino platform, allowing a quick and easy connection to your Arduino Project.

4D Systems has a simple library to get you started using this Shield, which includes an example sketch that demo’s the key attributes of the library.





- The shield utilises the SPI bus connected to the Arduino, which are MOSI pin **D11**, MISO pin **D12** and SCK pin **D13**.
- The shield includes a uSD card socket, to provide the Arduino the means to access data stored on a uSD card. The uSD SPI Card Select pin is **D10**.
- The display on the 4Display-Shield-22 has an SPI interface, which is shared with the uSD card socket. The Display Card Select pin is **D7**.
- The display utilises an RS pin for selecting if data is written or if a command is written the display, utilising pin **D8**.
- There is also a display Reset pin which utilises pin **D14** (A0).
- The control of the LCD Backlight is done with pin **D9**, which can have PWM applied to it if desired to adjust the brightness of the display.
- The 4-wire Resistive Touch utilises pins **D15** (A1), **D16** (A2), **A3** and **A4**.

4. Library Overview

4.1. Library Functions

The following are functions made available to the Arduino using the 4D Systems Library. Please see the website to download this library, www.4dsystems.com.au

begin()
Initialise the display

---

clear()
Clears the display

---

invert(flag)
Inverts the display.
Parameters:
Flag = true to invert, false for normal.

---

setBacklight(flag)
Turns the backlight on or off.
Parameters:
flag = true for on, false for off.

---

setDisplay(flag)
Turns the display on or off.
Parameters:
flag = true for on, false for off.

---

setOrientation(orientation)
Sets the orientation of the display.
Parameters:
orientation = 0 Portrait, 1 Right Rotated Landscape, 2 Reverse Portrait, 3 Left Rotated Landscape.

---

uint16_t getOrientation()
Gets the current orientation of the display.
Returns:
0 = Portrait
1 = Right Rotated Landscape
2 = Reverse Portrait
3 = Left Rotated Landscape.

---

uint16_t fontX()
Gets the font size, x-axis.
Returns:
Horizontal size of current font, in pixels

---

uint16_t fontY()
Gets the font size, y-axis.
Returns:
Vertical size of current font, in pixels

---

uint16_t maxX()
Gets the size of the screen, x-axis.
Returns:
Horozontal size of the screen, in pixels.
Note: 128 means 128 pixels and thus 0..127 coordinates (decimal)

---

uint16_t maxY()
Gets the size of the screen, y-axis.
Returns:
Vertical size of the screen, in pixels.
Note: 160 means 160 pixels and thus 0..159 coordinates (decimal)

---

circle(x0, y0, radius, colour)
Draws a circle from x0, y0 center with specified radius and colour.
Parameters:
x0 = x-axis centre
y0 = y-axis centre
radius = radius of circle in pixels
colour = 16 bit colour

---

solidCircle(x0, y0, radius, colour)
Draws a solid circle from x0, y0 center with specified radius and colour.
Parameters:
x0 = x-axis centre
y0 = y-axis centre
radius = radius of circle in pixels
colour = 16 bit colour

---

setBackGroundColour(colour)
Sets the background colour of the display, default is Black.
Parameters:
colour = 16 bit colour

---

line(x1, y1, x2, y2, colour)
Draws a line from x1, y1 to x2, y2 with specified colour.
Parameters:
x1 = x-axis start pixel
y1 = y-axis start pixel
x2 = x-axis end pixel
y2 = y-axis end pixel
colour = 16 bit colour

---

rectangle(x1, y1, x2, y2, colour)
Draws a rectangle from x1, y1 to x2, y2 with specified colour.
Parameters:
x1 = x-axis start pixel
y1 = y-axis start pixel
x2 = x-axis end pixel
y2 = y-axis end pixel
colour = 16 bit colour

---

solidRectangle(x1, y1, x2, y2, colour)
Draws a solid rectangle from x1, y1 to x2, y2 with specified colour.
Parameters:
x1 = x-axis start pixel
y1 = y-axis start pixel
x2 = x-axis end pixel
y2 = y-axis end pixel
colour = 16 bit colour

---

point(x1, y1, colour)
Draws a pixel at from x1, y1 with specified colour.
Parameters:
x1 = x-axis pixel
y1 = y-axis pixel
colour = 16 bit colour

---

text(x0, y0, string, textColour, backColour, x-multiplier, y-multiplier)
Draw ASCII Text at the pixel coordinates with set colours and size.
Parameters:
x0 = x-axis pixel
y0 = y-axis pixel
string = text string
textColour = 16 bit colour of Text
backColour = 16 bit colour of background
x-multiplier = multiplier of x-axis text (default=1)
y-multiplier = multiplier of y-axis text (default=1)

---

boolean getTouch(x, y)
Gets touch activity and coordinates.
Parameters:
x = x-axis coordinate
y = y-axis coordinate
Note: x and y coordinates are consistent with the orientation
Returns:
true = pressed
false = otherwise

---

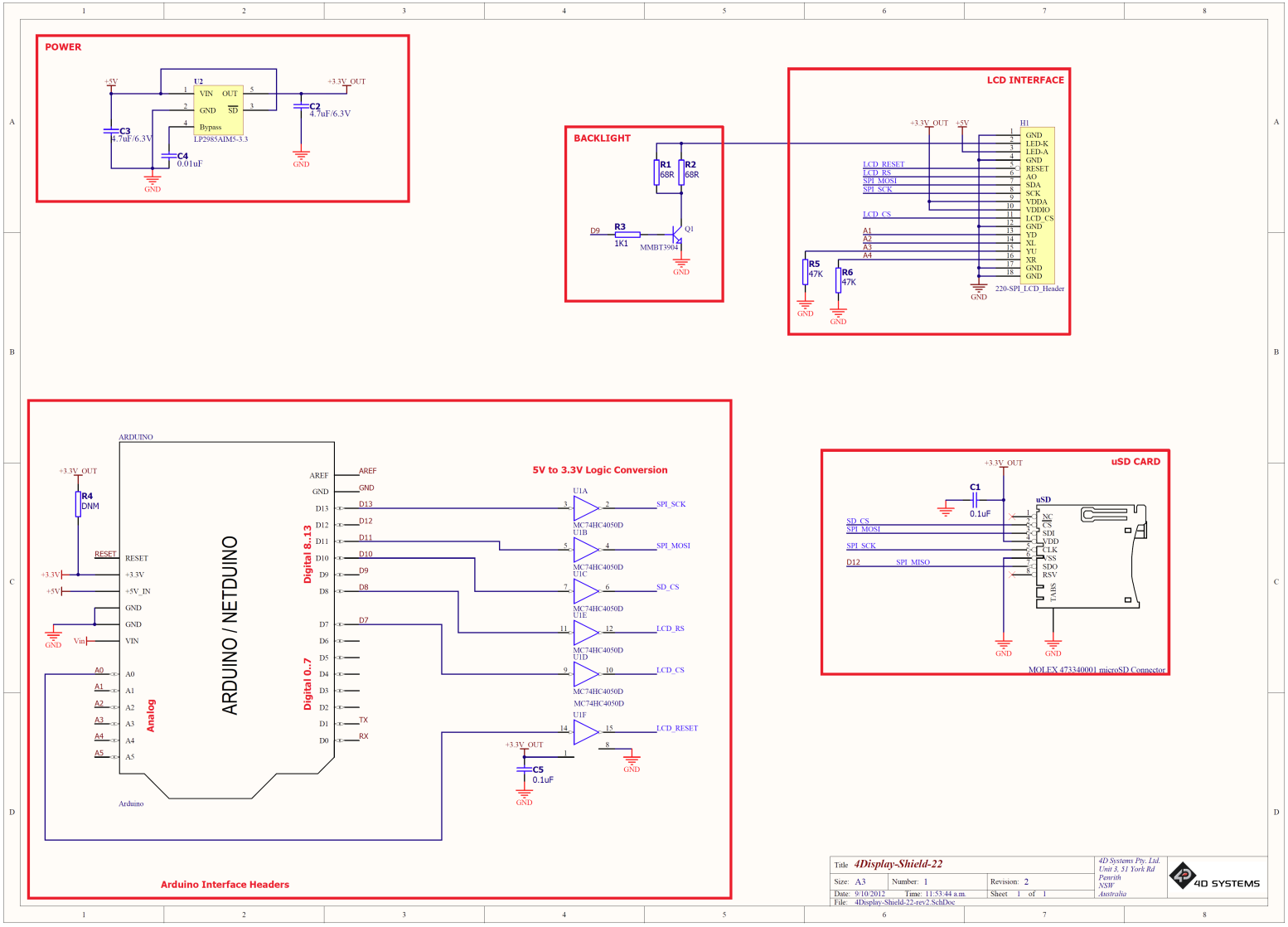
uint16_t setColour(red, green, blue)
Calculates the 16-bit colour from 8-bit Red-Green-Blue components
Parameters:
red = 0x00..0xff Red Component
green = 0x00..0xff Green Component
blue = 0x00..0xff Blue Component
Returns:
16 bit colour

---

splitcolour(rgb, red, green, blue)
Calculate the 8-bit Red-Green-Blue components from a 16-bit colour
Parameters:
rgb = 16-bit colour
red = 0x00..0xff Red Component
green = 0x00..0xff Green Component
blue = 0x00..0xff Blue Component

Please refer to the actual library for more detail regarding these functions, available for download from the 4D Systems website, www.4dsystems.com.au

5. Schematic Design



6. Specifications and Ratings

RECOMMENDED OPERATING CONDITIONS					
Parameter	**Conditions**	**Min**	**Typ**	**Max**	**Units**
Supply Voltage (VCC)		4.5	--	5.5	V
Operating Temperature		-10	--	+65	°C

ORDERING INFORMATION

Order Codes:

- 4Display-Shield-22

Package: 105mm x 65mm x 30mm

Packaging: Module sealed in antistatic foam padded 4D Systems Box

7. Legal Notice

Proprietary Information

The information contained in this document is the property of 4D Systems Pty. Ltd. and may be the subject of patents pending or granted, and must not be copied or disclosed without prior written permission.

4D Systems endeavours to ensure that the information in this document is correct and fairly stated but does not accept liability for any error or omission. The development of 4D Systems products and services is continuous and published information may not be up to date. It is important to check the current position with 4D Systems. 4D Systems reserves the right to modify, update or makes changes to Specifications or written material without prior notice at any time.

All trademarks belong to their respective owners and are recognised and acknowledged.

Disclaimer of Warranties & Limitation of Liability

4D Systems makes no warranty, either expressed or implied with respect to any product, and specifically disclaims all other warranties, including, without limitation, warranties for merchantability, non-infringement and fitness for any particular purpose.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

In no event shall 4D Systems be liable to the buyer or to any third party for any indirect, incidental, special, consequential, punitive or exemplary damages (including without limitation lost profits, lost savings, or loss of business opportunity) arising out of or relating to any product or service provided or to be provided by 4D Systems, or the use or inability to use the same, even if 4D Systems has been advised of the possibility of such damages.

4D Systems products are not fault tolerant nor designed, manufactured or intended for use or resale as on line control equipment in hazardous environments requiring fail – safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines or weapons systems in which the failure of the product could lead directly to death, personal injury or severe physical or environmental damage ('High Risk Activities'). 4D Systems and its suppliers specifically disclaim any expressed or implied warranty of fitness for High Risk Activities.

Use of 4D Systems' products and devices in 'High Risk Activities' and in any other application is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless 4D Systems from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any 4D Systems intellectual property rights.

8. Contact Information

For Technical Support: support@4dsystems.com.au

For Sales Support: sales@4dsystems.com.au

Website: www.4dsystems.com.au