DATASHEET

4D SYSTEMS

TURNING TECHNOLOGY INTO ART

gen4-4DPi-43T (4.3” Resistive Touch)
gen4-4DPi-50T (5.0” Resistive Touch)
gen4-4DPi-70T (7.0” Resistive Touch)
gen4-4DPi-43CT-CLB (4.3” Capacitive Touch)
gen4-4DPi-50CT-CLB (5.0” Capacitive Touch)
gen4-4DPi-70CT-CLB (7.0” Capacitive Touch)

Primary Displays for the Raspberry Pi

Document Date: 6th May 2018
Document Revision: 1.0

Contents

1. Description

The gen4-4DPi range are Primary Display's for the Raspberry Pi* A+, B+, Pi2, Pi3, Pi3 B+, Pi Zero and Pi Zero W, which display the primary output of the Raspberry Pi, like what is normally sent to the HDMI or Composite output. It features an integrated Resistive Touch panel or Capacitive Touch panel, enabling the gen4-4DPi to function with the Raspberry Pi without the need for a mouse.

Communication between the gen4-4DPi and the Raspberry Pi is interfaced with a high speed 48Mhz SPI connection, which utilises an on-board processor for direct command interpretation and SPI communication compression, and features a customised DMA enabled kernel. This combination allows this display to output high frame rate compared to other SPI display solutions, when displaying a typical image/video, and can achieve higher depending if the image can be compressed.

The gen4-4DPi is designed to work with the Raspbian Operating System running on the Raspberry Pi, as that is the official Raspberry Pi operating system. It is also compatible with Pixel and Scratch.

Please note that the display resolution of the 4.3" is 480x272 pixels, while the 5.0 and the 7.0" are 800x480 pixels, and thus may not display all menus in the Raspbian Desktop fully, without some downscaling.

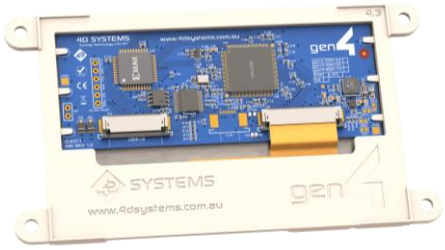
The gen4-4DPi range connect to the Raspberry Pi's 40 pin header using the gen4-4DPi Adaptor, which then connects to the gen4-4DPi display module using a 30 way FFC Cable.

Note*: Raspberry Pi is a trademark of the Raspberry Pi Foundation, and all references to the words 'Raspberry Pi' or the use of its logo/marks are strictly in reference to the Raspberry Pi product, and how *this* product is compatible with but is not associated with the Raspberry Pi Foundation in any way.

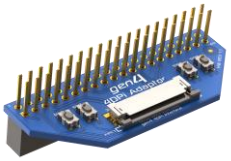
2. Features

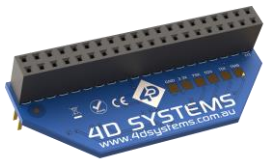
- Universal Primary Display for the Raspberry Pi.
- Compatible with Raspberry Pi A+, B+, Pi2, Pi3, P3 B+, Pi Zero and Pi Zero W.
- 480x272 Resolution (4.3")
- 800x480 Resolution (5.0" & 7.0")
- TFT Screen with integrated 4-wire Resistive Touch Panel (T), or Capacitive Touch Panel (CT) with Cover Lens Bezel (CLB).
- Display GUI output / primary output, just like a monitor connected to the Raspberry Pi
- High Speed 48MHz SPI connection to the Raspberry Pi, featuring SPI compression technology.
- Typical frame rate of 20 Frames per second (FPS) – 4.3", or 7 Frames per second (5" & 7"), higher if image can be compressed further by the kernel. Lower if no compression is possible.
- Powered directly off the Raspberry Pi, no external power supply is required.
- On board EEPROM for board identification, following the HAT standard.
- 4x 4.0mm Mounting holes on Non-Touch and Resistive Touch modules, or via adhesive for Capacitive Touch model.
- RoHS and CE Compliant.

Resistive Touch Display configuration shown.



3. Pin Configuration and Summary





H1 Pinout (Raspberry Pi Connector on gen4-4DPi-Adaptor) – FEMALE connector			
Pin	**Symbol**	**I/O**	**Description**
1	+5V	P	+5V Supply Pin, connected to the main 5V supply of the Raspberry Pi
2	+3.3V	P	+3.3V Supply Pin, connected to the main 3.3V supply of the Raspberry Pi
3	+5V	P	+5V Supply Pin, connected to the main 5V supply of the Raspberry Pi
4	SDA1	I/O	I2C SDA1
5	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
6	SCL1	O	I2C SCL1
7	GPIO14	I/O	GPIO on the Raspberry Pi - **unused**
8	GPIO4	I/O	GPIO on the Raspberry Pi - **unused**
9	GPIO15	I/O	GPIO on the Raspberry Pi - **unused**
10	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
11	GPIO18	I/O	GPIO on the Raspberry Pi – Can be used for PWM Backlight, else **unused**
12	PENIRQ	I	Interrupt for the touchscreen controller
13	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
14	KEYIRQ	I	Interrupt for the push buttons
15	GPIO23	I/O	GPIO on the Raspberry Pi - **unused**
16	GPIO22	I/O	GPIO on the Raspberry Pi - **unused**
17	GPIO24	I/O	GPIO on the Raspberry Pi - **unused**
18	+3.3V	P	+3.3V Supply Pin, connected to the main 3.3V supply of the Raspberry Pi
19	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
20	MOSI	O	SPI MOSI Pin
21	GPIO25	I/O	GPIO on the Raspberry Pi - **unused**
22	MISO	I	SPI MISO Pin
23	SPI-CS0	O	SPI Chip Select 0 – Used for Xilinx Processor for Display, to Raspberry Pi
24	SCK	O	SPI SCK Clock Pin
25	SPI-CS1	O	SPI Chip Select 1 – **usused**
26	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
27	ID-SC	O	I2C ID EEPROM
28	ID-SD	I/O	I2C ID EEPROM
29	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
30	GPIO5	I/O	GPIO on the Raspberry Pi - **unused**
31	GPIO12	I/O	GPIO on the Raspberry Pi - **unused**
32	GPIO6	I/O	GPIO on the Raspberry Pi - **unused**
33	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
34	GPIO13	I/O	GPIO on the Raspberry Pi – **unused**
35	GPIO16	I/O	GPIO on the Raspberry Pi - **unused**
36	GPIO19	I/O	GPIO on the Raspberry Pi - **unused**
37	GPIO20	I/O	GPIO on the Raspberry Pi - **unused**
38	GPIO26	I/O	GPIO on the Raspberry Pi - **unused**
39	GPIO21	I/O	GPIO on the Raspberry Pi - **unused**
40	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi

I = Input, O = Output, P = Power

Continued overleaf…

Note: The on-board Xilinx processor of the gen4-4DPi utilises one of the Chip Select (CS) pins on the Raspberry Pi's SPI Bus (SPI-CS0). There is SPI-CS1 still available for use by the User.

Note: The on-board Resistive Touch Screen Controller or on-board Capacitive Touch Controller, utilises the I2C bus (SDA1, SCL1) to communicate to the Raspberry Pi. The I2C bus is capable of communicating with other devices also, so isn't restricted to only the 4DPi's touch controller.

gen4-4DPi 30 way FFC Interface, between gen4-4DPi-Adaptor and gen4-4DPi Display			
Pin	**Symbol**	**I/O**	**Description**
1	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
2	SDA1	I/O	I2C SDA1
3	SCL1	O	I2C SCL1
4	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
5	MOSI	O	SPI MOSI Pin
6	SCK	O	SPI SCK Clock Pin
7	MISO	I	SPI MISO Pin
8	SPI-CS0	O	SPI Chip Select 0 – Used for Xilinx Processor for Display, to Raspberry Pi
9	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
10	ID-SC	O	I2C ID EEPROM
11	ID-SD	I/O	I2C ID EEPROM
12	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
13	PENIRQ	I	Interrupt for the touchscreen controller
14	KEYIRQ	I	Interrupt for the push buttons
15	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
16	SW5	I	Button 5 (Not present on Adaptor), connected to Xilinx Processor on display
17	SW4	I	Button 4 on gen4-4DPi-Adaptor, connected to Xilinx Processor on display
18	SW3	I	Button 3 on gen4-4DPi-Adaptor, connected to Xilinx Processor on display
19	SW2	I	Button 2 on gen4-4DPi-Adaptor, connected to Xilinx Processor on display
20	SW1	I	Button 1 on gen4-4DPi-Adaptor, connected to Xilinx Processor on display
21	JTAG-TMS	-	Special Pins for Factory Programming of Xilinx Processor only – **Do Not Connect**
22	JTAG-TDI	-	Special Pins for Factory Programming of Xilinx Processor only – **Do Not Connect**
23	JTAG-TDO	-	Special Pins for Factory Programming of Xilinx Processor only – **Do Not Connect**
24	JTAG-TCK	-	Special Pins for Factory Programming of Xilinx Processor only – **Do Not Connect**
25	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
26	+5V	P	+5V Supply Pin, connected to the main 5V supply of the Raspberry Pi
27	+5V	P	+5V Supply Pin, connected to the main 5V supply of the Raspberry Pi
28	+3.3V	P	+3.3V Supply Pin, connected to the main 3.3V supply of the Raspberry Pi
29	+3.3V	P	+3.3V Supply Pin, connected to the main 3.3V supply of the Raspberry Pi
30	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi

I = Input, O = Output, P = Power

4. Connecting the Display to the Pi

4.1. Hardware Connection

The gen4-4DPi is easily connected to a Raspberry Pi. Ensure the Raspberry Pi is powered off when connecting the gen4-4DPi display or adaptor.

Simply align the Female 40 way header on the gen4-4DPi-Adaptor with the Raspberry PI's Male 40 way header, and connecting them together – ensuring the alignment is correct and all pins are seated fully and correctly. The gen4-4DPi-Adaptor should be overhanging inward of the Raspberry Pi.

Next simply connect the 30 way FFC cable between the FFC Connector of the gen4-4DPi and the gen4-4DPi-Adaptor, ensuring the copper pins are facing upward in the connector.

Please note that hardware connection to the Pi is not recommended until the Pi has been set up. Please see instructions below.

4.2. Software Download / Installation

4D Systems has prepared a custom DMA enabled kernel for use with the Raspbian Operating System, which is available for download as a single Package. This can be installed over your existing Raspbian installation, or it can be applied over a fresh image. It is recommended to apply over a fresh image.

If you are starting from a fresh image, start from Step 1, else skip to step 3 if you already have a Raspbian Image and which to apply this kernel to that. Please note, it is impossible for us to know what you have done to your Raspbian image, if you are not installing from a fresh image – so if you encounter issues, please try and use a fresh image to determine if possible modifications are conflicting with our kernel release. If you are running a Raspbian version with a Kernel version later than our Kernel Pack, you may encounter problems. Please contact support if you have problems.

1) Download the latest Raspbian Image from the Raspberry Pi website: http://downloads.raspberrypi.org/raspbian_latest
2) Load the Raspberry Pi image onto a SD card, using the instructions provided on the Raspberry Pi website for Linux, Mac or PC: http://www.raspberrypi.org/documentation/installation/installing-images/README.md
3) Insert the SD card into the Raspberry PI. Do not connect the gen4-4DPi yet. You will need an external monitor / keyboard / network connection, else simply a network connection to the Pi and the rest can be done over an SSH connection. Start up the Pi with at minimum an Ethernet connection connected.
4) Either log into the Raspberry Pi from your keyboard/monitor using the standard 'pi' and 'raspberry' credentials, else SSH into your raspberry PI and log in via your SSH session.
5) Expand the file system on downloaded image using raspi-config (submenu Expand Filesystem). After exiting raspi-config a reboot is needed.

```
#sudo raspi-config
#sudo reboot
```

6) Once rebooted, It is highly recommended to do an apt-get update and apt-get upgrade to ensure you are running the latest version of the kernel and firmware on your Pi, before you patch it for the 4DPi. After doing this, reboot once again.
Warning: An upgrade should only be done after making sure that the latest kernel is supported by the latest kernel pack from 4D. Otherwise, installing the 4D kernel pack will downgrade the kernel.
7) Once rebooted, log into your Raspberry Pi again, you will need to download and install the kernel which supports the 4DPi-24-HAT. The following step requires sudo 'root' access.
8) To download and install files, enter the following commands in terminal/shell /SSH to download the kernel from the 4D Systems Server:

```
$ wget
http://www.4dsystems.com.au/downloads/4
DPi/All/gen4-hats_4-14-34_v1.1.tar.gz

Install:

$ sudo tar -xzvf gen4-hats_4-14-
34_v1.1.tar.gz -C /
```

The package automatically selects the kernel required for Pi1, Pi2 or Pi3 automatically.

9) On newer Raspbian images, by default the system boots to Desktop GUI. Booting to

command line can be selected using the raspi-config tool, submenu Boot Options.

```
$ sudo raspi-config
```

10) Shutdown the Raspberry Pi safely, and remove the power after it has completed its shutdown. For shutting down use the following command

```
$ sudo poweroff
or
$ sudo halt
```

Shutdown can take a while, since many files have to be written from cache to SD card.

11) Connect the gen4-4DPi to the Raspberry Pi, and reapply power. The terminal should begin to show on the gen4-4DPi, and will be ready to use once the Raspberry Pi has booted.

4.3. Calibrating the Touch Screen

Each gen4-4DPi which is shipped from the 4D Systems factory is slightly different, in the sense that each of the touch screens has a slightly different calibration. In order to get the best from your gen4-4DPi, you will need to calibrate the display so it is as accurate as possible.

To calibrate the touch screen, the xinput_calibrator is required and the following steps should be carried out. Make sure the Desktop is not running before you start, quit desktop if it is and return to the terminal prompt. Please note that only resistive touch display modules could be calibrated.

1) Install xinput_calibrator (if not installed by default) by running this from terminal:

```
$ sudo apt-get install xinput-calibrator
```

2) Install the event device input driver:

```
$ sudo apt-get install xserver-xorg-input-evdev
```

3) Rename 10-evdev.conf file to 45-evdev.conf.

```
$ sudo mv /usr/share/X11/xorg.conf.d/10-evdev.conf
/usr/share/X11/xorg.conf.d/45-evdev.conf
```

4) Check if evdev.conf has a higher number than libinput.conf.

```
$ ls /usr/share/X11/xorg.conf.d/
```

The user should get something like this

```
10-quirks.conf   40-libinput.conf   45-evdev.conf 99-fbturbo.conf
```

5) Perform a reboot

```
$ sudo reboot now
```

6) Reconnect to SSH and run xinput calibrator.

```
$ DISPLAY=:0.0 xinput_calibrator
```

Perform the calibration and copy results. The result should be something similar to this:

```
Section "InputClass"
    Identifier              "calibration"
    MatchProduct    "AR1020 Touchscreen"
    Option  "Calibration"   "98 4001 175
    3840"
    Option  "SwapAxes"      "0"
EndSection
```

7) You may test the changes after xinput calibrator ends. To make the changes permanent, paste the results to

```
/usr/share/X11/xorg.conf.d/99-calibration.conf
```

```
$ sudo nano
/usr/share/X11/xorg.conf.d/99-calibration.conf
```

8) Save the file and perform a reboot

```
$ sudo reboot now
```

The Display should now be calibrated.

4.4. Change the Display Orientation

To change the display orientation, simply edit the /boot/cmdline.txt file

Add the parameter below at the second position in the parameter list:

```
4dpi.rotate = 90
```

And change this to have the value of 0, 90, 180 or 270. It should look something like:

```
dwc_otg.lpm_enable=0 4dpi.rotate=90
console=serial0,115200
```

Save the file and restart your Raspberry Pi.

The touch screen will automatically remap the alignment thanks to the custom kernel.

4.5. Push Buttons on the gen4-4DPi-Adaptor

The gen4-4DPi-Adaptor features 4 push buttons, which are connected to the Xilinx Processor. These can be used to trigger events on the Raspberry Pi. Please refer to the Appendix for code examples on how to utilize these buttons.

4.6. Change the SPI Freq and Compression

The gen4-4DPi can be adjusted to work with a range of SPI Frequencies and levels of compression, depending on the requirements of the end product/project.

Increasing the frequency can result in a higher Frame Rate (FPS), however will use more power and processor time.
Increasing the level of the compression can also result in a higher FPS, but may cause the display to corrupt.

By default, a SPI Frequency of 48Mhz is used, with a Compression level of 7.

The following parameters are the defaults in the /boot/cmdline.txt file, and can be edited to adjust the Frequency and Compression level.

```
4dpi.sclk=48000000
4dpi.compress=7
```

Setting compress to be 1 will enable the kernel to control the level of compression based on the frequency selected. This however is not guaranteed to have a good end result, and may require manually setting the compression level if corruption on the display is experienced.

If corruption or display anomalies occur at any given compression level, try to lower it by 1 value and check if this has improved.

Note, changing the frequency and compression require a restart of the Raspberry Pi.

4.7. Backlight Control

The backlight brightness can be controlled from the terminal, or from a bash script. The following command can be used to set the backlight from 0 to 100%.

```
sudo sh -c 'echo 31 >
/sys/class/backlight/4d-hats/brightness'
```

The above will set the backlight to 100%. Simply change the ‘echo 31’ to be anything from 0 to 31.

4.8. Parameters Listing

The following is a list of all the custom parameters used by the gen4-4DPi.

rotate: Screen rotation 0/90/180/270 (int)
compress: SPI compression 0/1/2/3/4/5/6/7 (int)
sclk: SPI clock frequency (long)

Valid SPI Frequency values (4dpi.sclk):
Values can be almost anything. This has been tested up to 64Mhz. Common values would include 64000000 (64MHz), 48000000 (Default), 32000000, 24000000 etc.

Valid Compression values (4dpi.compress):
0 (compression off)
1 (compression on, auto set based on sclk value)
2 (lowest), 3, 4, 5, 6, 7 (highest compression)

These parameters can be set or read from the /boot/cmdline.txt file, and they can be read from the /sys/modules/4dpi/parameters directory.

For example:
```
cat /sys/modules/4dpi/parameters/rotate
```

Will display the current rotation saved.

4.9. HDMI or 4DPi Output

To switch the X Windows output being displayed on HDMI or 4DPi output, X can be launched using the following commands:

```
startx -- -layout TFT
```

```
startx -- -layout HDMI
```

Alternatively, these commands do the same thing:

```
FRAMEBUFFER=/dev/fb1 startx
```

```
Startx
```

4.10. Changing DPI output

It is possible to change the DPI output of the 4DPi the same way as other LXDE based systems.

login as pi and open terminal

```
nano .Xresources
```

Add this line:
```
Xft.dpi: 75
```

reboot... This will set the DPI to be 75

5. Display Module Part Numbers

The following is a breakdown on the part numbers and what they mean.

Example:
gen4-4DPi-70CT-CLB

gen4	- gen4 Display Range
4DPi	- Display Family
70	- Display size (7.0”)
T	- Resistive Touch.
CT	- Capacitive Touch
CLB	- Cover Lens Bezel

- For part numbers which do not include T or CT, these are non-touch variants.
- Cover Lens Bezels (CLB) are glass fronts for the display module with overhanging edges, which allow the display module to be mounted directly into a panel using special adhesive on the overhanging glass. These are available for Capacitive Touch only.

6. Cover Lens Bezel – Tape Spec

The perimeter of the CLB display modules features double-sided adhesive tape, designed to stick directly onto a panel, enclosure, box etc without the need for any mounting screws or hardware.

The tape used is 3M 9495LE tape, which uses the well-known and strong 3M 300LSE adhesive.

The double-sided adhesive has a thickness of 0.17mm once the backing has been removed.

More information on this adhesive can be found on the 3M website.

http://multimedia.3m.com/mws/media/771683O/3mtm-double-coated-tapes-9474le-9495le.pdf

7. Standard FFC cable specifications

Between the gen4-4DPi-Adaptor and the gen4-4DPi Display Module, the following FFC cable is supplied:

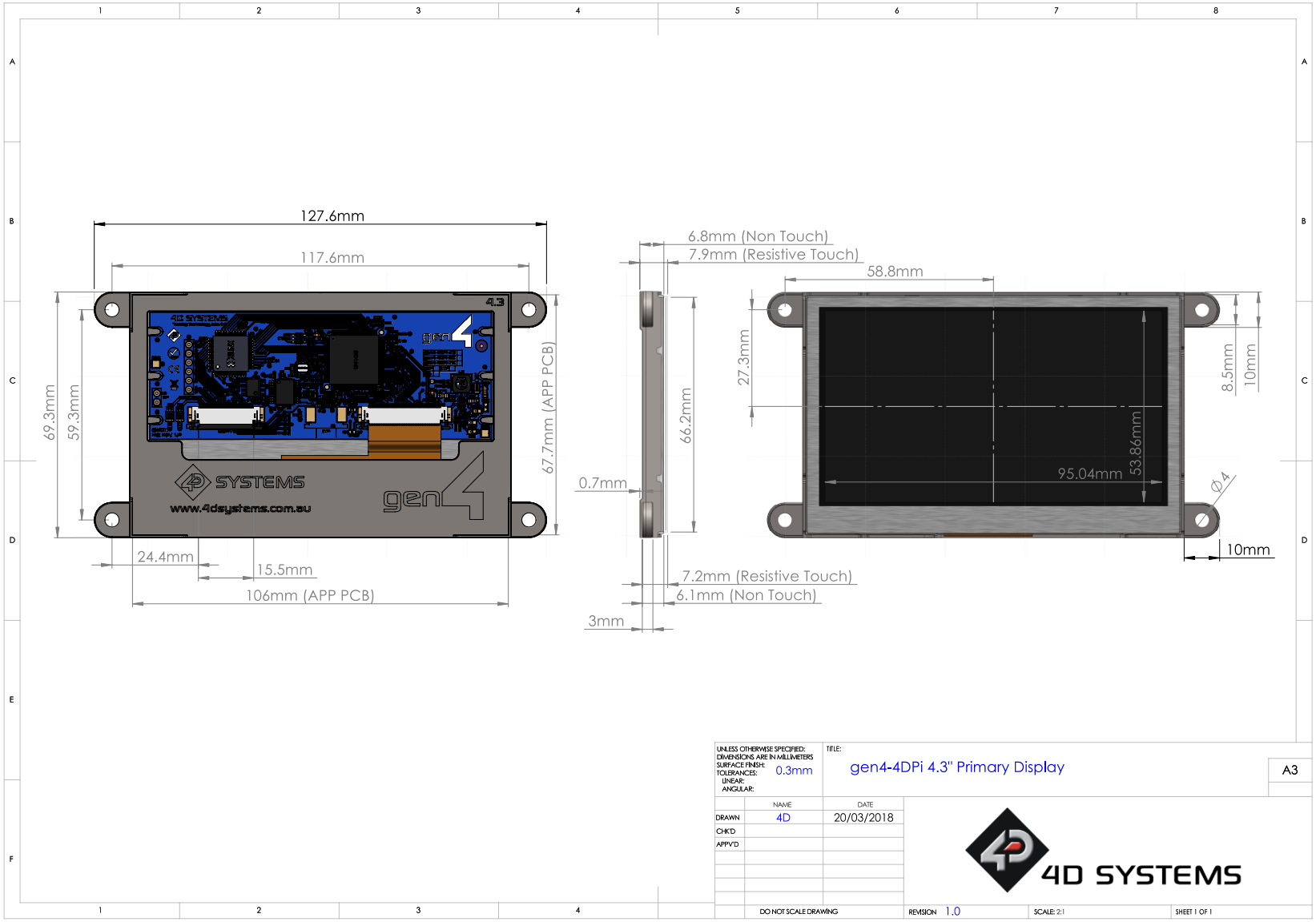
30 Pin Flexible Flat Cable, 150mm Long, 0.5mm (0.02") pitch
Cable Type: AWM 20624 80C 60V VW-1
Heat Resistance 80 Degrees Celsius
Connections on the opposite side at each end (Type B)

All FFC connectors on the gen4-4DPi are ‘Top Contact’ meaning the FFC cable has the metal ‘fingers’ pointing upward in the connector, blue stiffner on the back of the FFC cable is down on the PCB side.

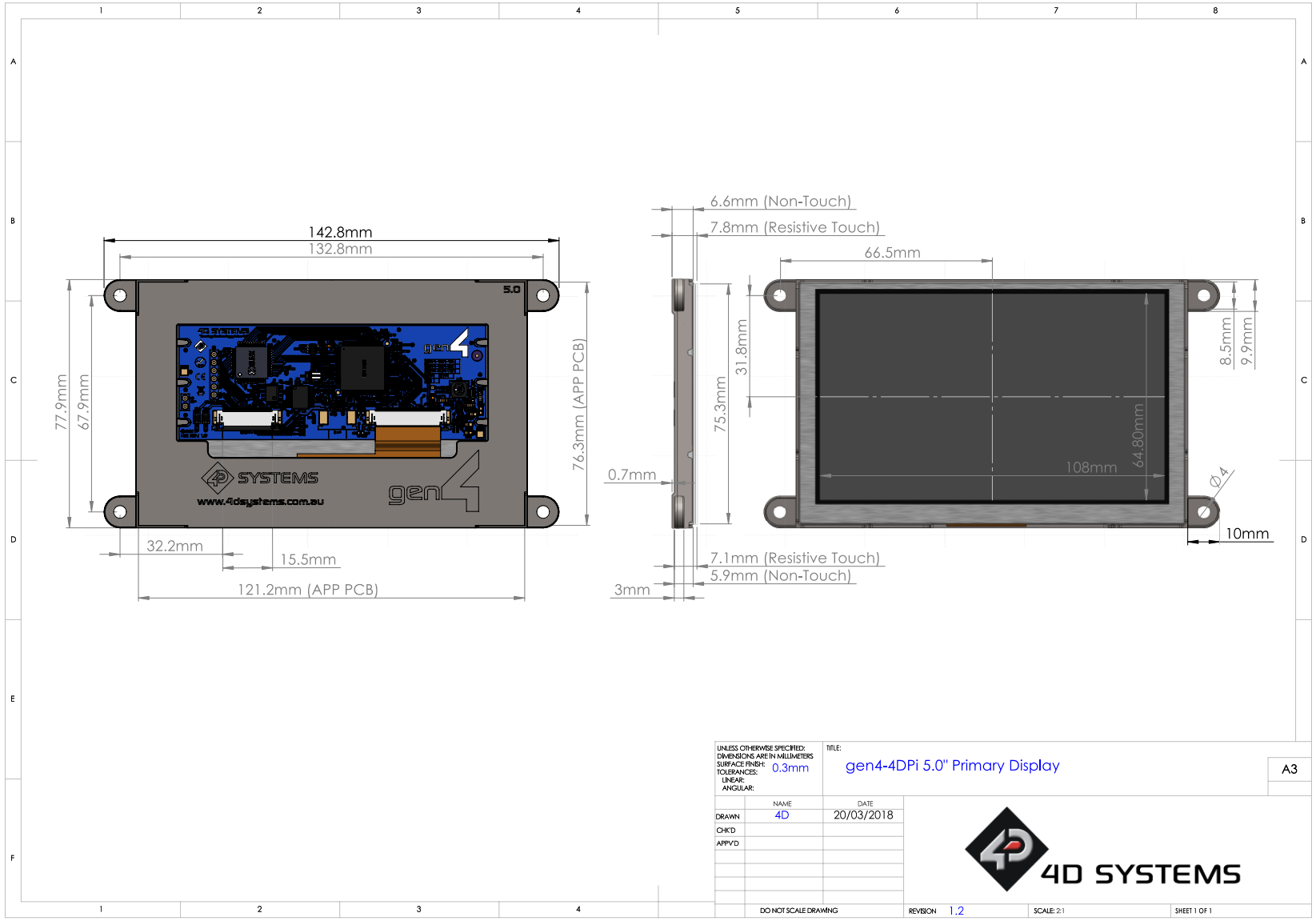
8. Notes

9. Scribble Box

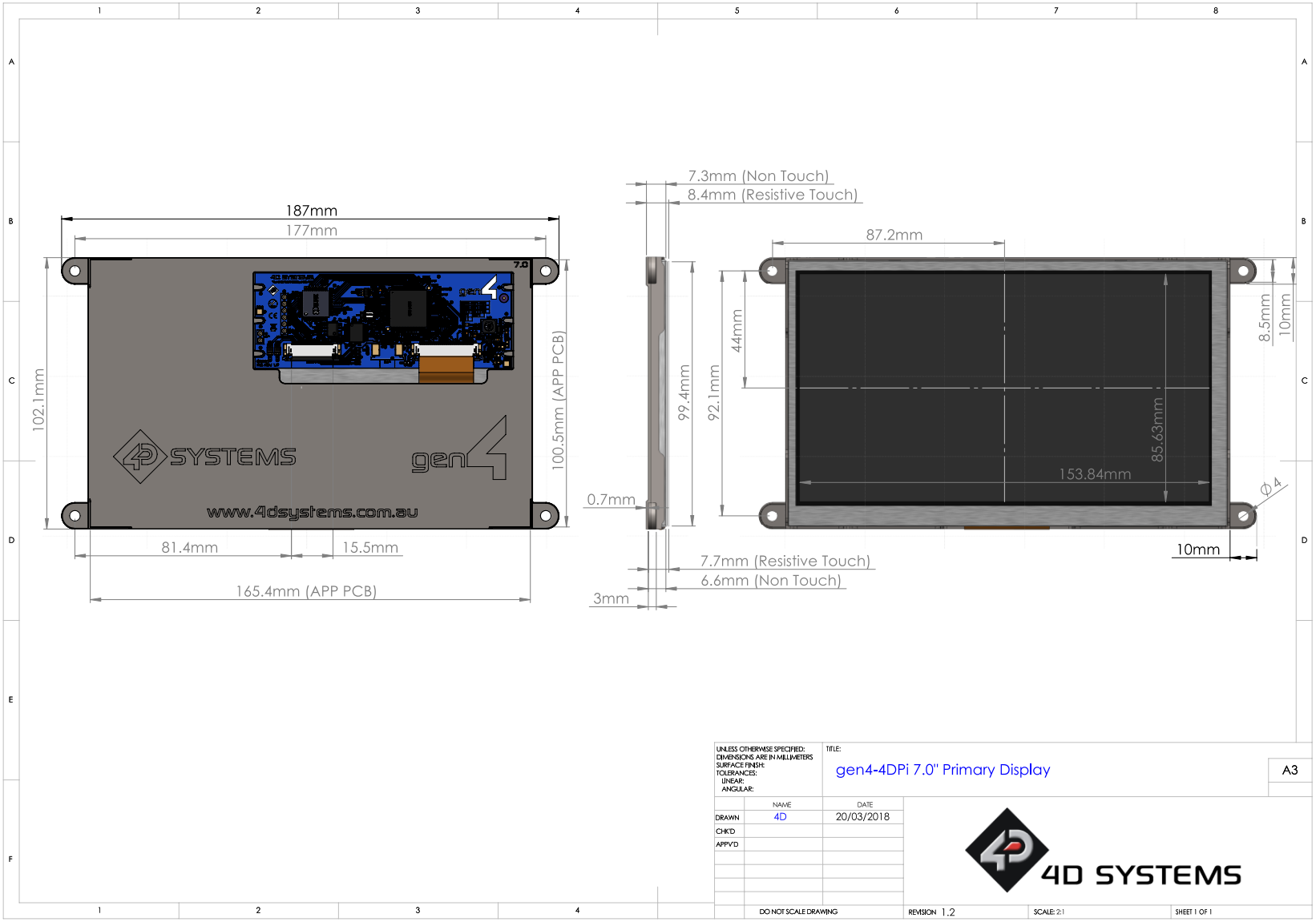
10. Mechanical Details gen4-4DPi-43T



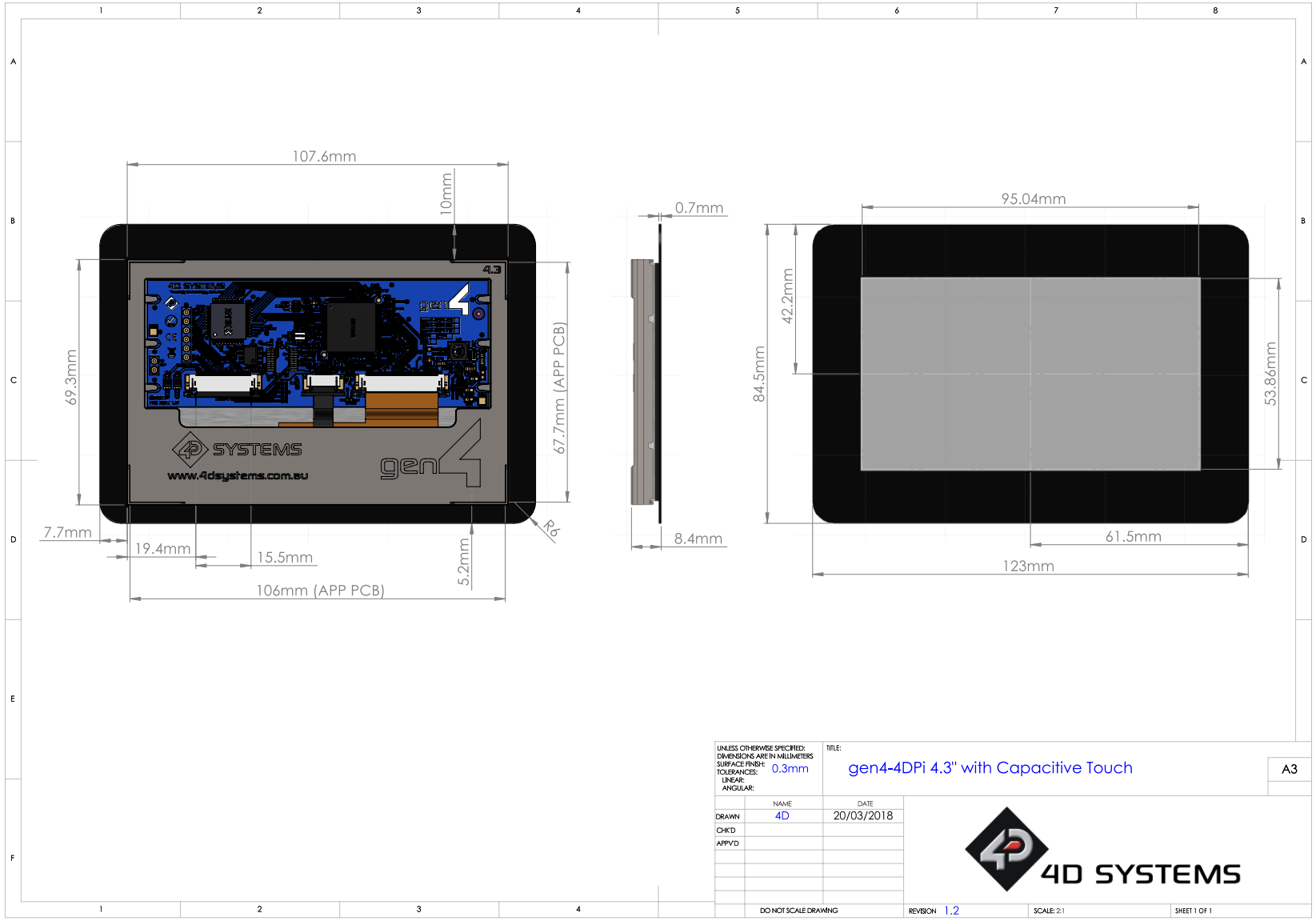
11. Mechanical Details gen4-4DPi-50T



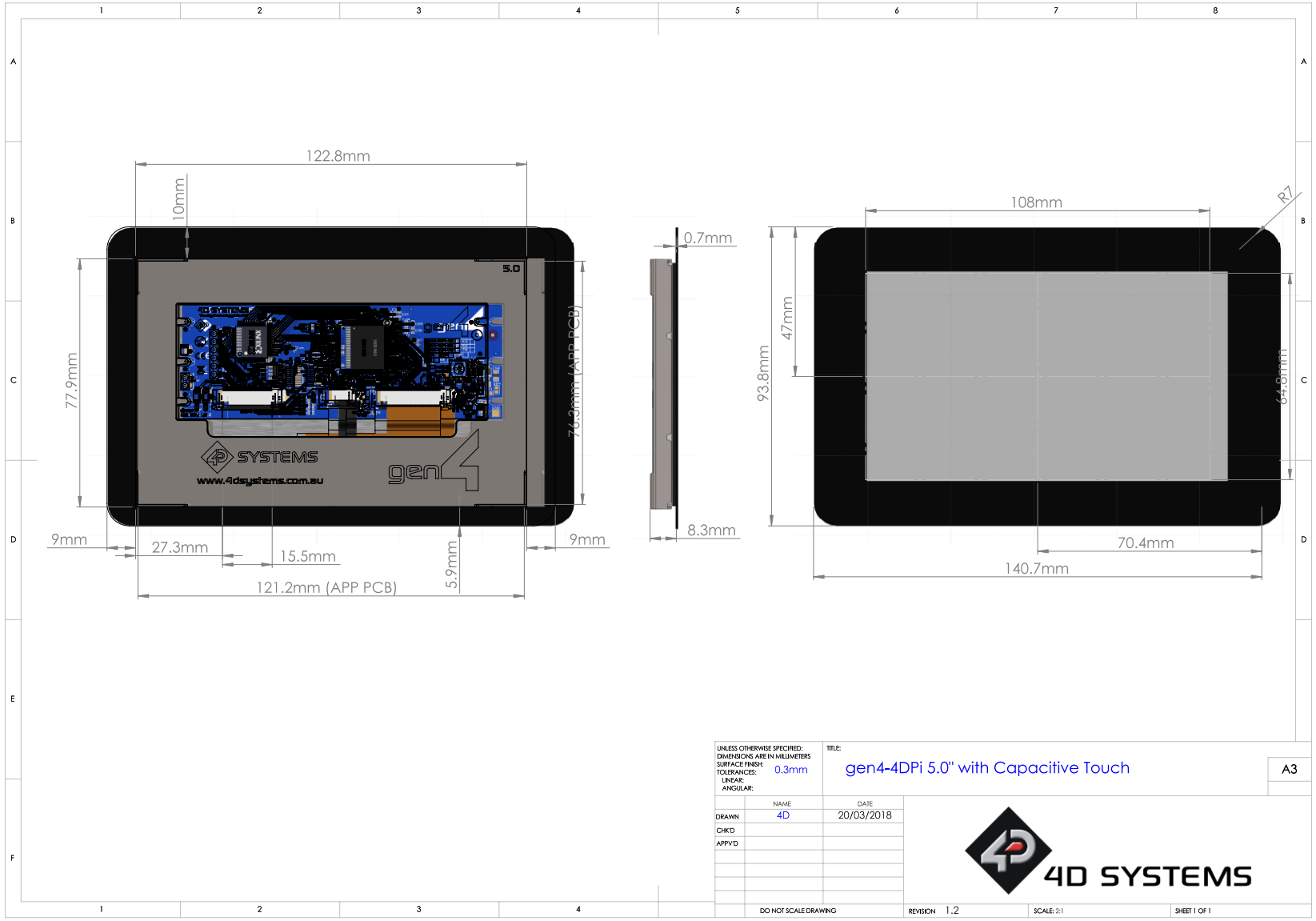
12. Mechanical Details gen4-4DPi-70T



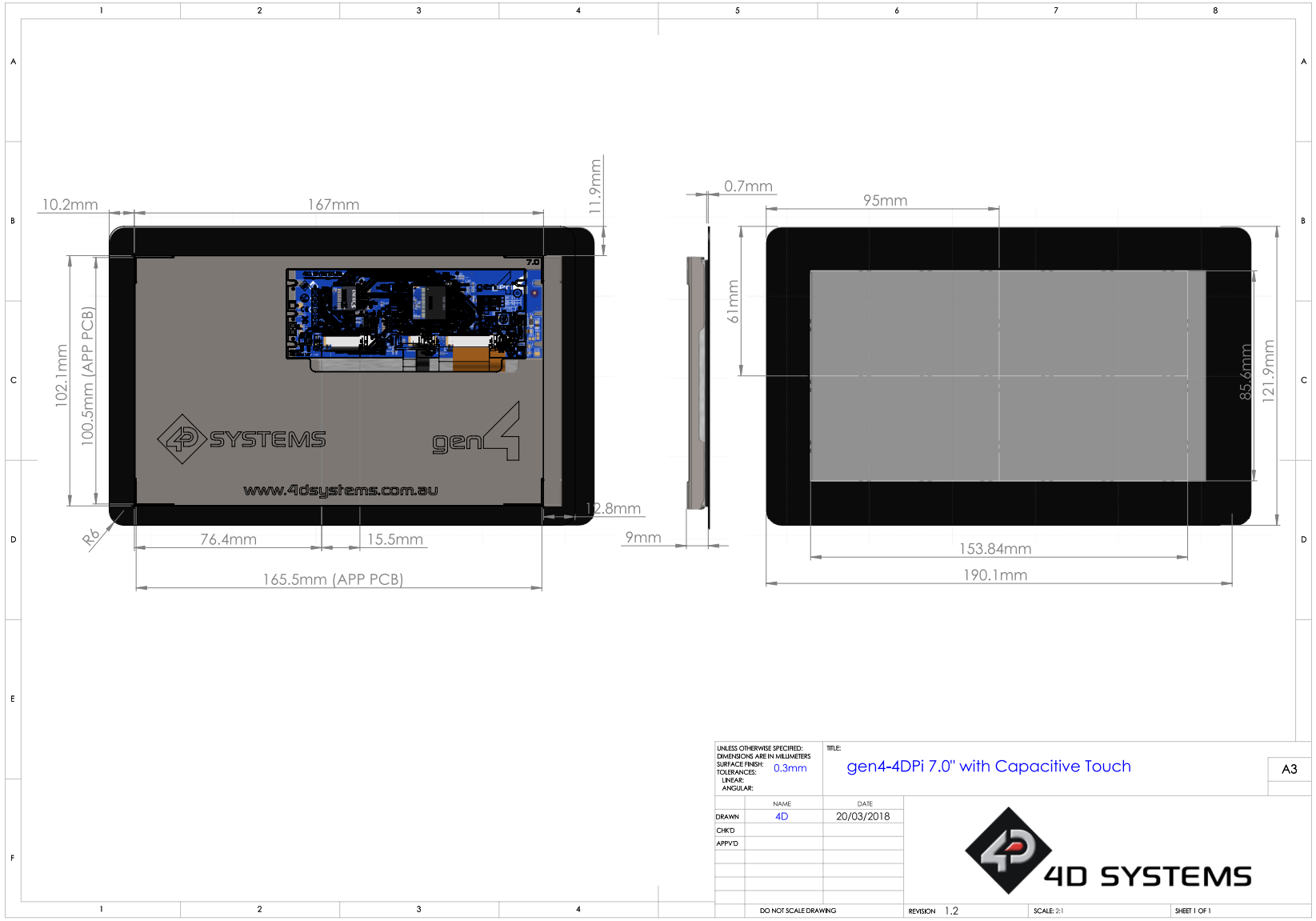
13. Mechanical Details gen4-4DPi-43CT-CLB



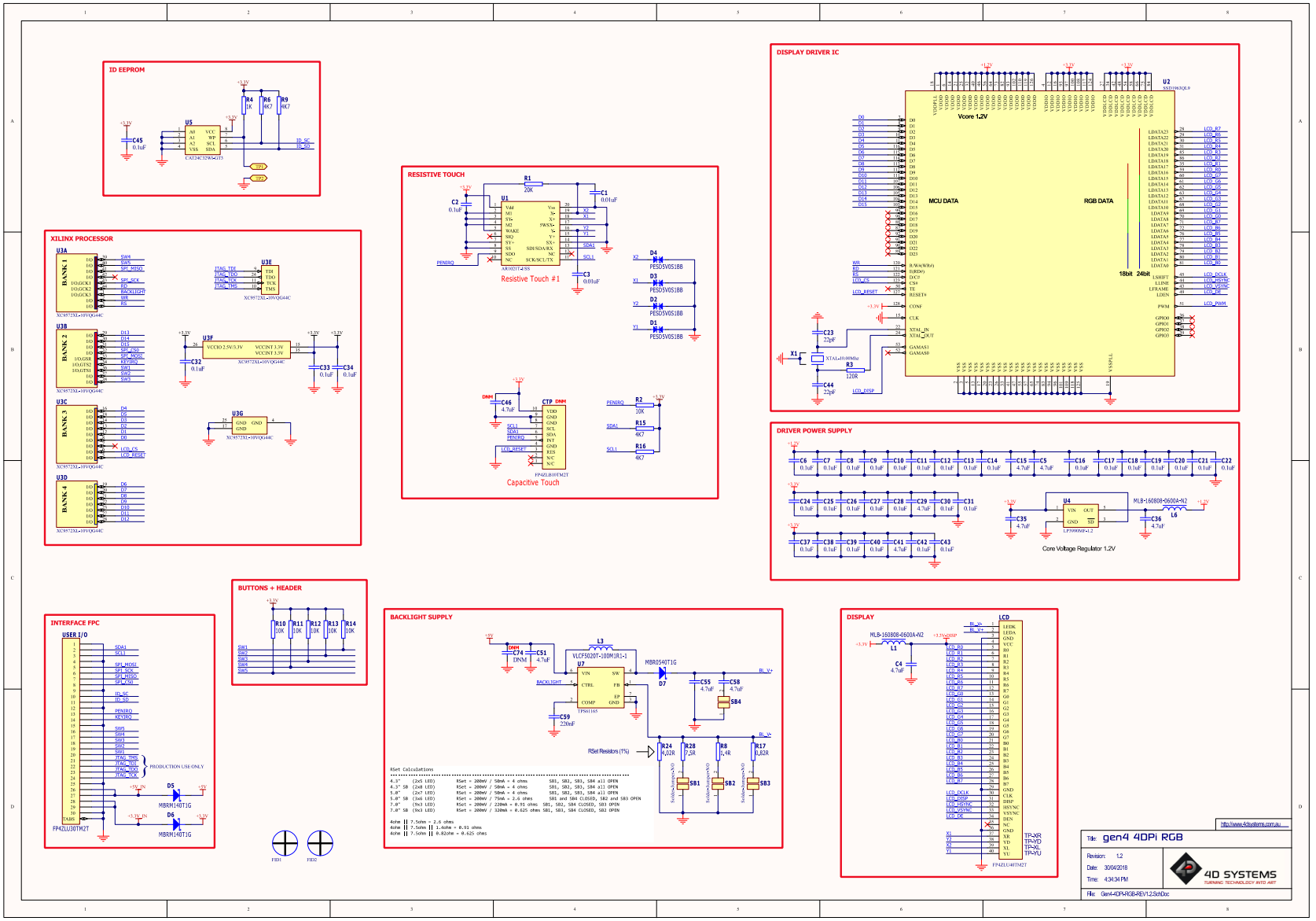
14. Mechanical Details gen4-4DPi-50CT-CLB



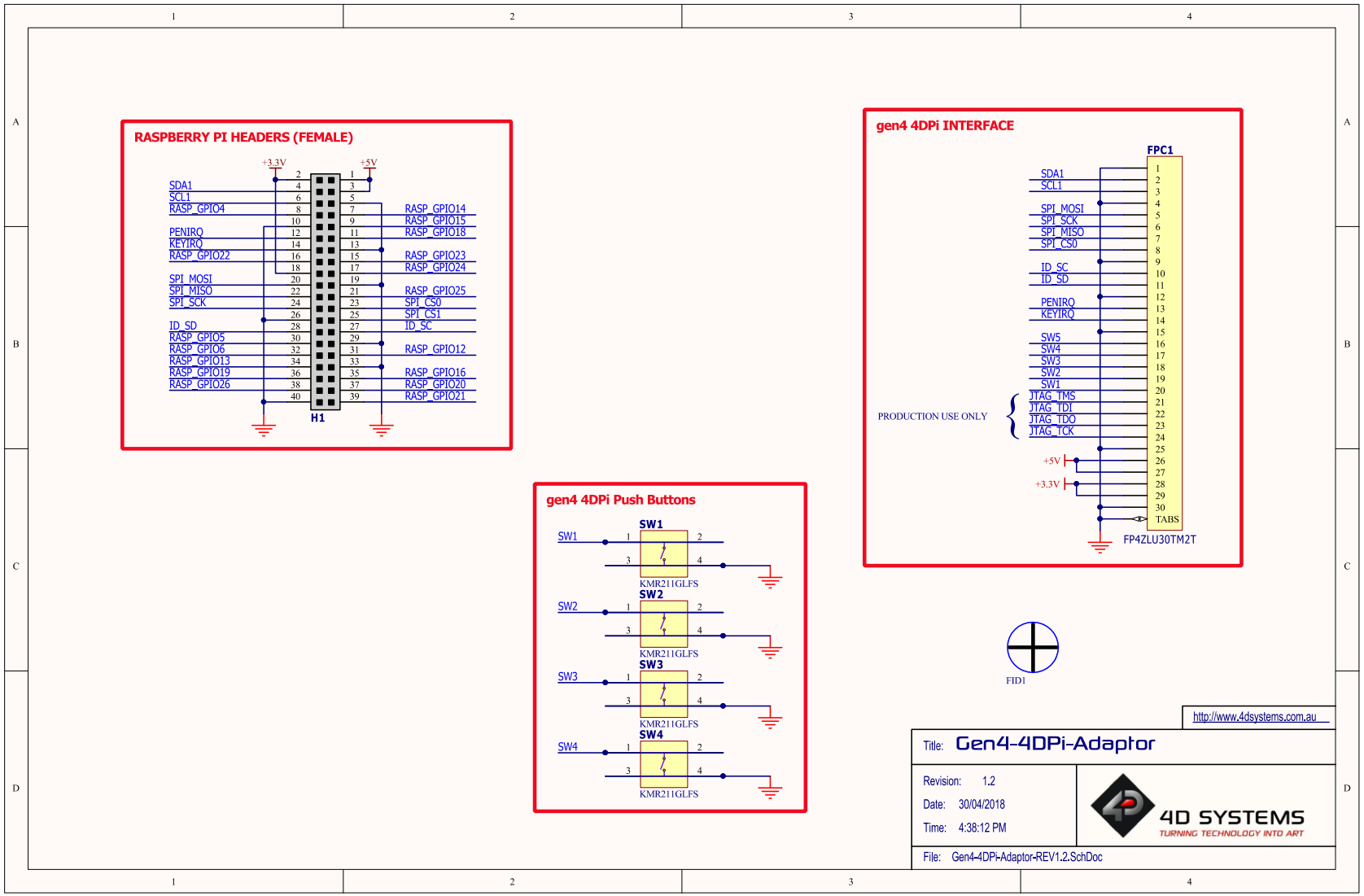
15. Mechanical Details gen4-4DPi-70CT-CLB



16. Schematic Diagram – gen4-4DPi (Display Module)



17. Schematic Diagram – gen4-4DPi-Adaptor (Display Adaptor)



18. Specifications and Ratings

ABSOLUTE MAXIMUM RATINGS	
Operating ambient temperature	-20°C to +70°C
Storage temperature	-30°C +80°C

NOTE: Stresses above those listed here may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the recommended operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED OPERATING CONDITIONS					
Parameter	Conditions	Min	Typ	Max	Units
Supply Voltage (+3.3V)	Stable supply from Raspberry Pi Bus	3.0	3.3	4.0	V
Supply Voltage (+5V)	Stable supply from Raspberry Pi Bus	4.5	5.0	5.5	V
Operating Temperature		-10	--	+60	°C

GLOBAL CHARACTERISTICS BASED ON OPERATING CONDITIONS					
Parameter	Conditions	Min	Typ	Max	Units
Supply Current Estimates (Display portion only) (RPi current not included) Idle home screen	gen4-4DPi-43T (Max Brightness)	--	250	--	mA
	gen4-4DPi-43CT-CLB (Max Brightness)	--	270	--	mA
	gen4-4DPi-50T (Max Brightness)	--	360	--	mA
	gen4-4DPi-50CT-CLB (Max Brightness)	--	380	--	mA
	gen4-4DPi-70T (Max Brightness)	--	640	--	mA
	gen4-4DPi-70CT-CLB (Max Brightness)	--	750	--	mA
Display Endurance	Hours of operation, measured to when display is 50% original brightness	30000	--	--	H
Touch Screen Endurance (Resistive Touch)	Number of touches/hits with a 12.5mm tip at a rate of 2x per second with 250gf force	--	1M	--	Touches
	Slide stylus on screen, 100gf force, 60mm/s speed with a 0.8mm polyacetal tip stylus pen	--	100K	--	Slides
Touch Screen Transparency	Resistive Touch	82	--	--	%
	Capacitive Touch	90	--	--	%
Touch Screen Operational Force (Resistive Touch)	Only use Finger or Stylus, do not use anything sharp or metal	20	--	100	Gf
CLB Hardness (Capacitive Touch)	Cover Lens Bezel Glass Hardness	--	6	--	H

LCD DISPLAY INFORMATION		
Parameter	Conditions	Specification
Display Type		TFT Transmissive LCD
Display Sizes		4.3”, 5.0” or 7.0” Diagonal

Display Resolution		480 x 272 (Landscape Viewing) – 4.3” 800 x 480 (Landscape Viewing) – 5” & 7”
Display Brightness	gen4-4DPi-43T (Max Brightness)	400 cd/m2
	gen4-4DPi-43CT-CLB (Max Brightness)	475 cd/m2
	gen4-4DPi-50T (Max Brightness)	400 cd/m2
	gen4-4DPi-50CT-CLB (Max Brightness)	475 cd/m2
	gen4-4DPi-70T (Max Brightness)	400 cd/m2
	gen4-4DPi-70CT-CLB (Max Brightness)	475 cd/m2
Display Contrast Ratio	Typical	500:1
Display Viewing Angles	Above Centre	70 Degrees
	Below Centre	60 Degrees
	Left of Centre	70 Degrees
	Right of Centre	70 Degrees
Display Viewing Direction		12 o’clock Display (Optimal viewing is from above when in Landscape/Wide mode)
Display Backlighting	gen4-4DPi-43xx Models	2x5 Parallel LED’s
	gen4-4DPi-50xx Models	2x6 Parallel LED’s
	gen4-4DPi-70xx Models	9x3 Parallel LED’s
Pixel Pitch	4.3”	0.198 x 0.198mm (Square pixels)
	5.0”	0.135 x 0.135mm (Square pixels)
	7.0”	0.1925 x 0.179mm (non-Square pixels)
Pixel Density (Number of pixels in 1 row in 25.4mm)	4.3”	128 DPI/PPI
	5.0”	183 DPI/PPI
	7.0”	132 DPI/PPI (Horizontal) 142 DPI/PPI (Vertical)

PERFORMANCE					
Parameter	**Conditions**	**Min**	**Typ**	**Max**	**Units**
Frame Rate (FPS) (4.3” only)	Video Playback, Full Screen, 480x272. A higher FPS can be achieved if display outputting lots of blocks of the same colour. See Section 4.6	--	20	--	FPS
Frame Rate (FPS) (5.0” & 7.0” only)	Video Playback, Full Screen, 800x480. A higher FPS can be achieved if display outputting lots of blocks of the same colour. See Section 4.6	--	7	--	FPS

ORDERING INFORMATION

Order Codes:

gen4-4DPi-43T

gen4-4DPi-50T

gen4-4DPi-70T

gen4-4DPi-43CT-CLB

gen4-4DPi-50CT-CLB

gen4-4DPi-70CT-CLB

Packaging: Module sealed in a 4D Systems box

19. Appendix 1 – Code Examples – Push Buttons

19.1. Example for communicating to Push Buttons, for C:

```
// test program to read state of buttons on 4D Systems 4DPi displays

#include <stdio.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>

#define LCD4DPI_GET_KEYS _IOR('K', 1, unsigned char *)

void print_keys(int fd)
{
    unsigned char keys;

    if (ioctl(fd, LCD4DPI_GET_KEYS, &keys) == -1)
    {
        perror("_apps ioctl get");
    }
    else
    {
        printf("Keys : %2x\n", keys);
    }
}

int main(int argc, char *argv[])
{
    char *file_name = "/dev/fb1";
    int fd;

    fd = open(file_name, O_RDWR);
    if (fd == -1)
    {
        perror("_apps open");
        return 2;
    }

    print_keys(fd);
    printf("Ioctl Number: (dec)%d  (hex)%x\n", LCD4DPI_GET_KEYS, LCD4DPI_GET_KEYS);

    close (fd);
    return 0;
}
```

19.2. Example for communicating to Push Buttons, for Python:

```
#!/usr/bin/python
import array, fcntl
from time import sleep
# test program to read state of buttons on 4D Systems 4DPi displays

#LCD4DPI_GET_KEYS = -2147202303

_IOC_NRBITS   =  8
_IOC_TYPEBITS =  8
_IOC_SIZEBITS = 14
_IOC_DIRBITS  =  2
_IOC_DIRMASK    = (1 << _IOC_DIRBITS) - 1
_IOC_NRMASK     = (1 << _IOC_NRBITS) - 1
_IOC_TYPEMASK   = (1 << _IOC_TYPEBITS ) - 1
_IOC_NRSHIFT = 0
_IOC_TYPESHIFT = _IOC_NRSHIFT+_IOC_NRBITS
_IOC_SIZESHIFT = _IOC_TYPESHIFT+_IOC_TYPEBITS
_IOC_DIRSHIFT  = _IOC_SIZESHIFT+_IOC_SIZEBITS
_IOC_NONE = 0
_IOC_WRITE = 1
_IOC_READ = 2

def _IOC(dir, type, nr, size):
#  print 'dirshift {}, typeshift {}, nrshift {}, sizeshift
{}'.format(_IOC_DIRSHIFT, _IOC_TYPESHIFT, _IOC_NRSHIFT, _IOC_SIZESHIFT)
  ioc = (dir << _IOC_DIRSHIFT ) | (type << _IOC_TYPESHIFT ) | (nr << _IOC_NRSHIFT )
| (size << _IOC_SIZESHIFT)
  if ioc > 2147483647: ioc -= 4294967296
  return ioc
#def _IO(type, nr):
#  return _IOC(_IOC_NONE,  type, nr, 0)
def _IOR(type,nr,size):
  return _IOC(_IOC_READ,  type, nr, size)
#def _IOW(type,nr,size):
#  return _IOC(_IOC_WRITE, type, nr, sizeof(size))

LCD4DPI_GET_KEYS = _IOR(ord('K'), 1, 4)
buf = array.array('h',[0])

print 'Press Top & Bottom buttons simultaneously to exit'

with open('/dev/fb1', 'rw') as fd:

  while True:
    fcntl.ioctl(fd, LCD4DPI_GET_KEYS, buf, 1) # execute ioctl call to read the keys
    keys = buf[0]

    if not keys & 0b00001:
      print "KEY1" ,
    if not keys & 0b00010:
      print "KEY2" ,
    if not keys & 0b00100:
      print "KEY3" ,
    if not keys & 0b01000:
      print "KEY4" ,
    if not keys & 0b10000:
      print "KEY5" ,

    if keys != 0b11111:
      print
    if keys == 0b01110: # exit if top and bottom pressed
      break

    sleep(0.1)
```

19.3. Example for Shutdown and Reset buttons, for C

```
// test program to Shutdown or Restart Pi using buttons on 4D Systems 4DPi displays

#include <stdio.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>

#define LCD4DPI_GET_KEYS _IOR('K', 1, unsigned char *)

int get_keys(int fd, unsigned char *keys)
{
    if (ioctl(fd, LCD4DPI_GET_KEYS, keys) == -1)
    {
        perror("_apps ioctl get");
       return 1;
    }
    *keys &= 0b11111;
    return 0;
}

int main(int argc, char *argv[])
{
    char *file_name = "/dev/fb1";
    int fd;
    unsigned char key_status;

    fd = open(file_name, O_RDWR);
    if (fd == -1)
    {
        perror("_apps open");
        return 2;
    }

    key_status = 0b11111;
    while(key_status & 0b00001) // press key 1 to exit
    {
        if(get_keys(fd, &key_status) != 0)
             break;

        // printf("key_status: %x\n", key_status);

        if(!(key_status & 0b10000))
        {
            system("sudo shutdown -h now");
            break;
        }

        if(!(key_status & 0b01000))
        {
            system("sudo reboot");
            break;
        }

        sleep(0.1);
    }

    close(fd);
    return 0;
}
```

19.4. Example for Shutdown and Reset buttons, for Python

```
#!/usr/bin/python
import array, fcntl, os
from time import sleep
# test program to Shutdown or Restart Pi using buttons on 4D Systems 4DPi displays

#LCD4DPI_GET_KEYS = -2147202303

_IOC_NRBITS   =  8
_IOC_TYPEBITS =  8
_IOC_SIZEBITS = 14
_IOC_DIRBITS  =  2

_IOC_DIRMASK    = (1 << _IOC_DIRBITS) - 1
_IOC_NRMASK     = (1 << _IOC_NRBITS) - 1
_IOC_TYPEMASK   = (1 << _IOC_TYPEBITS ) - 1

_IOC_NRSHIFT = 0
_IOC_TYPESHIFT = _IOC_NRSHIFT+_IOC_NRBITS
_IOC_SIZESHIFT = _IOC_TYPESHIFT+_IOC_TYPEBITS
_IOC_DIRSHIFT  = _IOC_SIZESHIFT+_IOC_SIZEBITS

_IOC_NONE = 0
_IOC_WRITE = 1
_IOC_READ = 2

def _IOC(dir, type, nr, size):
#  print 'dirshift {}, typeshift {}, nrshift {}, sizeshift
{}'.format(_IOC_DIRSHIFT, _IOC_TYPESHIFT, _IOC_NRSHIFT, _IOC_SIZESHIFT)
  ioc = (dir << _IOC_DIRSHIFT ) | (type << _IOC_TYPESHIFT ) | (nr << _IOC_NRSHIFT )
| (size << _IOC_SIZESHIFT)
  if ioc > 2147483647: ioc -= 4294967296
  return ioc
#def _IO(type, nr):
#  return _IOC(_IOC_NONE,  type, nr, 0)
def _IOR(type,nr,size):
  return _IOC(_IOC_READ,  type, nr, size)
#def _IOW(type,nr,size):
#  return _IOC(_IOC_WRITE, type, nr, sizeof(size))

LCD4DPI_GET_KEYS = _IOR(ord('K'), 1, 4)
#print 'ssd {} {:12} {:0>8x} {:0>32b}'.format(ssd1289, hex(ssd1289), ssd1289,
ssd1289)
buf = array.array('h',[0])

with open('/dev/fb1', 'rw') as fd:

  while True:
    fcntl.ioctl(fd, LCD4DPI_GET_KEYS, buf, 1) # execute ioctl call to read the keys
    keys = buf[0]

    if not keys & 0b00001:
      break

    if not keys & 0b10000:
      os.system("sudo shutdown -h now")
      break

    if not keys & 0b01000:
      os.system("sudo reboot")
      break;

    sleep(0.1)
```

20. 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.

Images and graphics used throughout this document are for illustrative purposes only. All images and graphics used are possible to be displayed on the 4D Systems range of products, however the quality may vary.

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.

21. Contact Information

For Technical Support: www.4dsystems.com.au/support

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

Website: www.4dsystems.com.au