4D SYSTEMS
TURNING TECHNOLOGY INTO ART

Primary Displays for Raspberry Pi

DATASHEET

DOCUMENT DATE: **28th June 2021**
DOCUMENT REVISION: **2.17**

4DPi-32-II

Primary Displays for Raspberry Pi

Table of Contents

1. Description

The 4DPi-32-II (Revision 2.x Hardware) is a 3.2" Primary Display for the Raspberry Pi, which plugs directly on top and displays the primary output like what is normally sent to the HDMI or Composite output. It features an integrated Resistive Touch panel, enabling the 4DPi-32-II to function with the Raspberry Pi without the need for a mouse.

Communication between the 4DPi-32-II 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 25FPS when displaying a typical image/video and can achieve higher depending if the image can be compressed.

The 4DPi-32-II 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.

Note: 4DPi-32-II Hardware states 4DPi-32 as the part number but is identifiable as the HW Rev of the PCB is 2.0 or above.

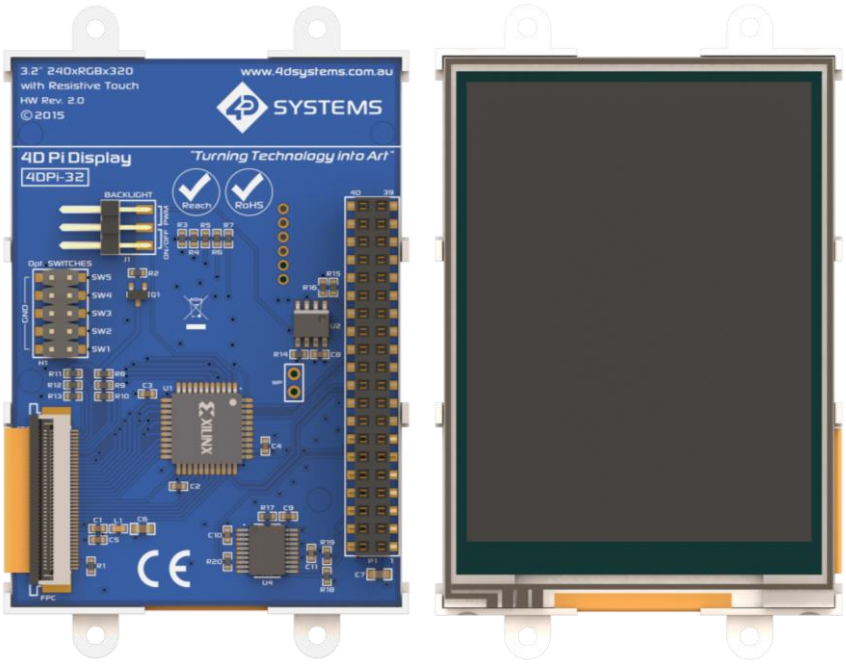
Note*: Raspberry Pi is a trademark of the Raspberry Pi Foundation, and all references to the words 'Raspberry Pi' or the use of its logo/marks are strictly in reference to the Raspberry Pi product, and how *this* product is compatible with but is not associated with the Raspberry Pi Foundation in any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

2. Features

- Universal 3.2" Primary Display for the Raspberry Pi.
- Compatible with Raspberry Pi A+, B+, Pi2, Pi3, Pi3 B+, Pi4, Pi Zero and Pi Zero W Revision 2.x hardware is not compatible with older A or B models.
- 320x240 QVGA Resolution, RGB 65K true to life colours, TFT Screen with integrated 4-wire Resistive Touch Panel.
- Display full 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 25 Frames per second (FPS), 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/Off or PWM controlled backlight, selectable by on board jumper.
- On board EEPROM for board identification, following the HAT standard.
- Module dimensions: 57.3 x 92.4 x 20.5mm (including corner plates). Weighing ~ 55g.
- Display Viewing Area: 48.60 x 64.80mm
- 4x corner plates with 2.6mm holes for mechanical mounting.
- RoHS and CE Compliant.

.

3. Pin Configuration and Summary



P1 Pinout (Raspberry Pi Connector – Female Connetor)			
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

Continued overleaf...

P1 Pinout continued…			
Pin	**Symbol**	**I/O**	**Description**
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 – **unused**
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

Note: The on-board processor of the 4DPi-32-II 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 Touch Screen Controller utilises the I2C bus (SDA1, SCL1) to communicate to the Raspberry Pi. The SPI Bus can communicate with other devices also, so is not restricted only to the 4DPi's touch controller.

Primary Displays for Raspberry Pi

4. Connecting the Display to the Pi

4.1. Hardware Connection

The 4DPi-32-II is easily connected to a Raspberry Pi, by simply aligning the Female 40 way header with the Raspberry PI's Male 40 way header, and connecting them together – ensuring the aligning is correct and all pins are seated fully and correctly.

NOTE: The 4DPi-32-II is supported only by the 40 way header, and therefore pressing on the touch screen may result in the 4DPi-32-II moving towards the Raspberry Pi, and therefore the circuitry touching the Raspberry Pi. This could result in damage to either product if a short circuit were to occur. It is therefore highly encouraged to mount the display and attach the Pi to the mounted display.

If development is desired on the bench prior to the mounting of the display, please ensure some sort of support is provided between the 4DPi-32-II and the Raspberry Pi so they do not touch inadvertently.

Included in the box is a small double sided sticky rubber pad. This is optional, however can be placed on the top of the Ethernet connector of the Raspberry Pi, to provide some support to the display.

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, modifications are conflicting with our kernel release. If you are running a Raspbian version with a Kernel version later than our Kernel Pack, you are very likely to encounter problems. Please contact support if you have problems.

1) Download the Raspbian Image from the Raspberry Pi website that corresponds to the Kernel Pack from our repository. You cannot use the latest Raspbian image if we do not have a Kernel Pack to match that kernel release:

   http://downloads.raspberrypi.org/raspbian_latest

   Note: If you encounter any issues with the latest version, it is advisable to use the Raspbian OS release with matching kernel version as one of the latest 4DPi packages that you plan to use.

   This is the current latest release which matches our Kernel Pack. Please utilize this image unless you are compiling your own Kernel or know what you are doing:

   https://downloads.raspberrypi.org/raspios_armhf/images/raspios_armhf-2020-08-24/

   This image has Kernel **5.4.51-v7+,** and therefore is compatible with our Kernel Pack, **5.4.68** – which is used further in these instructions.

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.

   SSH typically has to be enabled manually from raspi-config, or it can be enabled by putting a blank file named 'ssh' (without the quotes, no extension) onto the boot drive of the microSD card before you take it out of your PC after flashing the image to the card, which will enable it without needing raspi-config.

   If logging in with Keyboard/Mouse/Monitor directly to the Pi, you can follow the Wizard the pops up on the desktop if you wish and set up your Keyboard/Location/Wifi details (Pi3/Pi4 etc). **However do NOT do the system update.** Please skip the update. Reboot the pi if prompted.

5) Typically, on modern versions of the Pi OS, this step is not required or is done automatically. However it is here for reference.

   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, do **NOT** do an apt-get upgrade, as this will almost certainly update the OS kernel version, which will cause an incompatibility with the kernel from our kernel pack. The Kernel pack must be applied to a kernel very very close if not identical to the kernel your OS is running, or there will be issues.

   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 and problems are almost certain to occur.

7) Log into your Raspberry Pi again, you will need to download and install the kernel which supports the gen4-4DPi. 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 https://4dsystems.com.au/media/downloads/4DPi/All/gen4-hats_5-4-68.tar.gz
```

   Then extract the kernel pack:

```
sudo tar --no-same-owner -xzvf gen4-hats_5-4-68.tar.gz -C /
```

   The package selects the kernel required for Pi1, Pi2, Pi3 or Pi4 automatically. If you want to check for the kernel packages released by 4D systems, proceed to ***Section: Latest Kernel Versions***

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 half
```

   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.con
f
```

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 like 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 **/etc/X11/xorg.conf.d/99-calibration.conf**

```
sudo nano /etc/X11/xorg.conf.d/99-calibratio
n.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 after the console parts in the parameter list:

```
4d_hats.rotate = 90
```

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

```
console=serial0,115200 console=tty1
4d_hats.rotate=90 root= (etc etc)
```

Save the file and restart your Raspberry Pi.

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

After changing the Display Orientation, you need to calibrate again the screen.

4.5. SPI Frequency 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 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.

```
4d_hats.sclk=48000000
4d_hats.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 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.

4.6. Backlight Control

The backlight is controllable in two possible ways. One is using simple on/off control, which is done by sending a GPIO command to the on-board processor, which then turns the backlight on and off. The other is using a DMA-PWM output from the Raspberry PI and controlling the backlight brightness.

The control of the backlight is selected using the Jumper J1, selecting ON/OFF or DMA control. For the simple ON/OFF control GPIO18 is used.

The backlight brightness can be controlled from the terminal, or from a bash script.

Executing the following command will control the backlight.

To turn the backlight OFF:

```
sudo sh -c 'echo 0 > /sys/class/backlight/4d
-hats/brightness'
```

To turn the backlight ON:

```
sudo sh -c 'echo 1 > /sys/class/backlight/4d
-hats/brightness'
```

To control the backlight using DMA-PWM, ensure that the Jumper J1 is on PWM.

The following command can be used to set the backlight from 0 to 100%.

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

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

4.7. 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 (4d-hats.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 (4d-hats.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/module/4d_hats/parameters/ directory.

For example:

```
cat /sys/module/4d_hats/parameters/rotate
```

Will display the current rotation saved.

4.8. HDMI or 4DPi Output

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

```
startx -- -layout TFT

startx -- -layout HDMI
```

Alternatively, these commands do the same thing:

```
FRAMEBUFFER=/dev/fb1 startx

startx
```

4.9. DPI Adjustment

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

Check the current DPI settings by

```
xrdb -query -all
```

The current dpi is listed next to the `xft.dpi` listing.

To change the DPI, it can be done like this. Edit the following file, and then merge it:

```
nano ~/.Xresources
```

Add this line:
`Xft.dpi: 75`

This will set the DPI to be 75

Save and exit the file.

Merge it so the value gets used, by doing the following:

```
xrdb -merge ~/.Xresources
```

Check again by doing the query.

```
xrdb -query -all
```

Reboot the Pi, and your changes should take effect.

Changing the DPI can make the screen blurry, so take care when adjusting these values. If you get to a point where it is unreadable, SSH into your Pi and change the value back to something reasonable.

Ideally DPI is set based on your resolution, however for small resolution displays, it can be desirable to make the DPI smaller so you can fit more on the screen.

4.10. Optimising appearance/ sizes

These are subject to change as the Raspbian OS gets updated, but changes such as these can assist with displaying content on the small display, if you are wanting to use it for a computer monitor, rather than an HMI or GUI for a custom application.

The 4DPi-35-II has a 480x320 resolution display, so expectations need to be realistic when dealing with the content which is viewed on this 3.5” display.

Raspbian as the OS itself, has not been optimised to run on a display with this resolution, so there are some menus and applications which will not display correctly, or fit on the screen entirely.

This however can be helped somewhat, by setting up the display appearances, and setting fonts and menus to be smaller.

It is recommended to use a USB mouse and keyboard to set this up.

Raspberry Menu -> Preferences -> Appearance Settings

go to Menu Bar tab
Select Small
go to System tab
click on font (Robo Light)
drag window, to see font size, select 8
(close window [x])

Click on File Manager in Menu Bar
Go to Edit, select Preferences,
Select Display
Choose smallest icon sizes for all icon types
Click on "Size of Large Icons", press tab 6 times (as the OK button is not visible), Press enter

Right click on menu bar (just left from pi menu, near bottom edge of menu bar), to get a pop-up menu
Choose Add/Remove Panel Items
Remove unwanted items (e.g. Bluetooth)
(close window [x])

Raspberry Menu->Shutdown->Reboot

5. Display Module Part Numbers

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

Example:
4DPi-32-II

4DPi - Display Family
32 - Display size (3.2")
II - Hardware Revision 2.0 and higher

6. Latest Kernel Versions

Here is the list of the kernel patches released by 4D systems.

Latest release:
gen4-hats_5-4-68.tar.gz

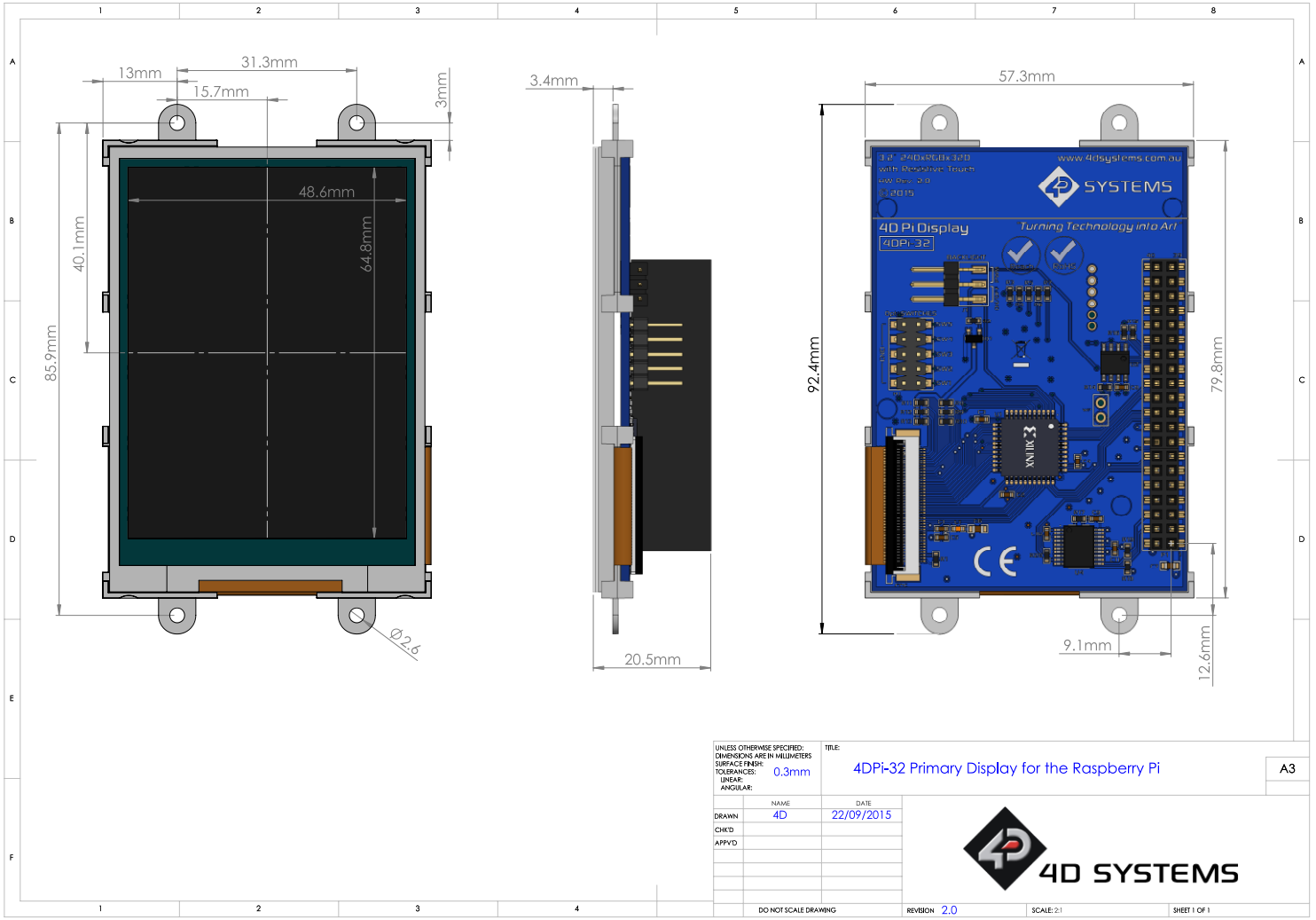
Previous releases:
gen4-hats_4-19-57-v7l+_v1.0.tar.gz
gen4-hats_4-14-34_v1.1.tar.gz
gen4-hats_4-9-80_v1.1.tar.gz
gen4-hats_4-9-59_v1.2.tar.gz

Note: (1) It is highly advisable to use a Raspbian OS release with **matching kernel version** as the kernel patches you decide to use. Please refer to section 4.2 point #1 regarding the current recommended OS.

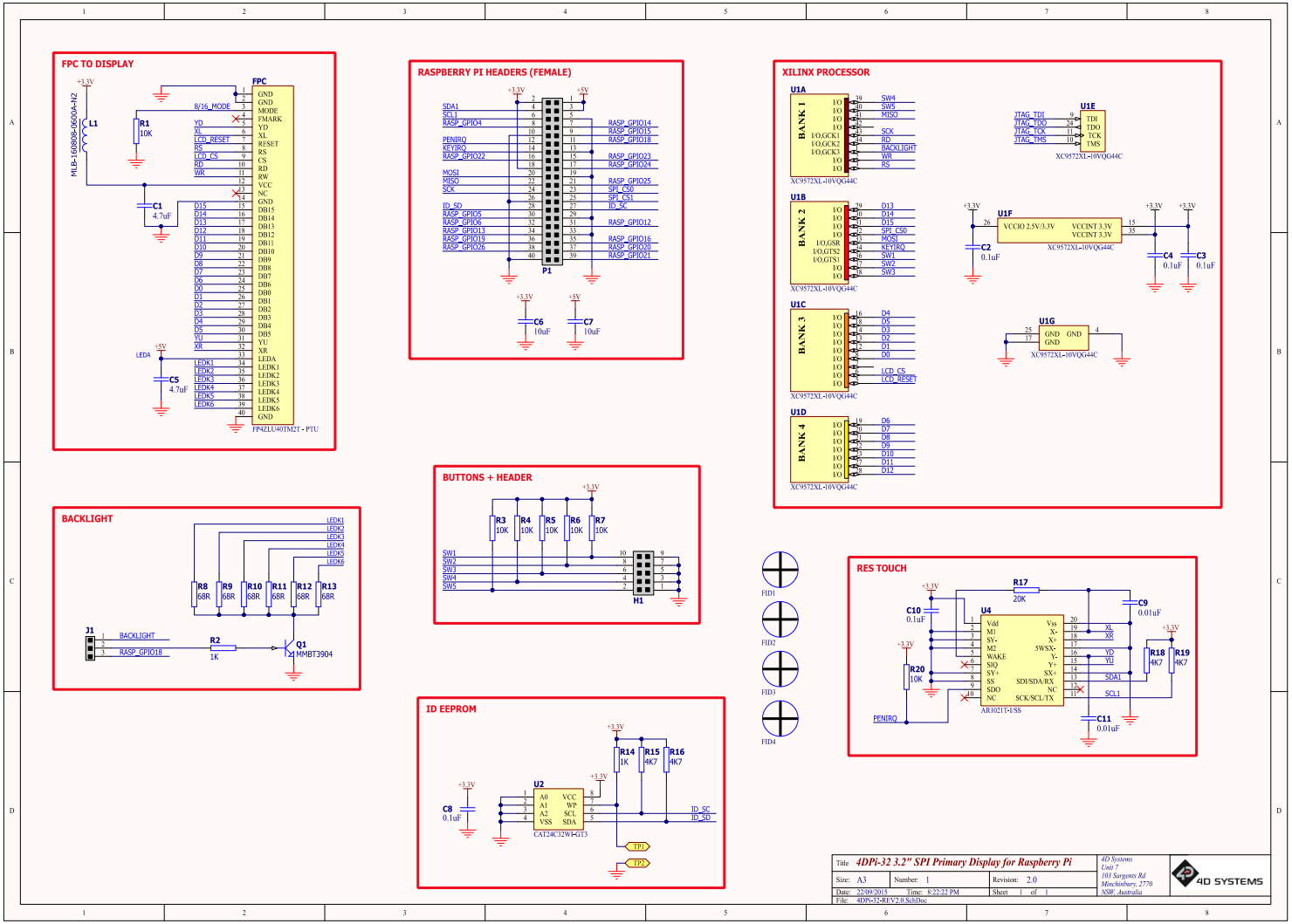
Note: (2) Some older kernel releases may be available upon request. Please contact 4D Systems Support Team for more information:

www.4dsystems.com.au/support

7. Mechanical Details



8. Schematic Diagram



9. Specifications and Ratings

ABSOLUTE MAXIMUM RATINGS	
Operating ambient temperature	-15°C to +65°C
Storage temperature	-30°C to +70°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 external supply required	3.0	3.3	4.0	V
Supply Voltage (+5V)	Stable external supply required	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 (ICC)	3.3V Supply.	--	100	--	mA
Backlight Current (ICC)	5V Supply	--	150	--	mA
Display Endurance	Hours of operation, measured to when display is 50% original brightness	--	20000	--	H

PERFORMANCE					
Parameter	Conditions	Min	Typ	Max	Units
Frame Rate (FPS)	Video Playback, Full Screen, 320x240. A higher FPS can be achieved if display outputting lots of blocks of the same colour. See Section 4.5	--	25	--	FPS

LCD DISPLAY INFORMATION		
Parameter	**Conditions**	**Specification**
Display Type		TFT Transmissive LCD
Display Sizes		3.2” Diagonal
Display Resolution		240 x 320 (Portrait View)
Display Brightness	5V Supply, gen4-uLCD-32D	200 cd/m2 (typical)
	5V Supply, gen4-uLCD-32D-CLB	194 cd/m2 (typical)
	5V Supply, gen4-uLCD-32DT	160 cd/m2 (typical)
	5V Supply, gen4-uLCD-32DCT-CLB	190 cd/m2 (typical)
Display Contrast Ratio	Typical	500:1
Display Viewing Angles	Above Centre	35 Degrees
	Below Centre	55 Degrees
	Left of Centre	55 Degrees
	Right of Centre	55 Degrees
Display Viewing Direction		6 o’clock Display (Optimal viewing is from below when in Portrait mode)
Display Backlighting	White LED Backlighting	1x6 Parallel LED’s
Pixel Pitch		0.2025 x 0.2025 (Square pixels)
Pixel Density	Number of pixels in 1 row in 25.44 mm	127 DPI/PPI

ORDERING INFORMATION

Order Code: 4DPi-32-II

Packaging: Module sealed in a 4D Systems box

10. Appendix 1 – Code Examples – Push Buttons

10.1. 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;
}
```

10.2. 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)
```

10.3. 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;
}
```


Primary Displays for Raspberry Pi

10.4. 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)
```

11. Hardware Revision History

Revision Number	Date	Description
2.0	09/22/2015	Initial Public Release Version

12. Document Revision History

Revision Number	Date	Description
2.0	09/28/2015	Initial Public Release Version
2.1	12/17/2015	Cosmetic changes on the datasheet
2.2	03/10/2016	Cosmetic changes on the datasheet
2.3	04/11/2016	Update of 4d kernel image and Cosmetic changes on the datasheet
2.4	05/24/2016	Update of 4d kernel image and Cosmetic changes on the datasheet
2.5	10/21/2016	Update of 4d kernel image and Cosmetic changes on the datasheet
2.6	01/11/2017	Update of 4d kernel image and Cosmetic changes on the datasheet
2.7	08/01/2017	Update of 4d kernel image and Cosmetic changes on the datasheet
2.8	03/07/2018	Update of 4d kernel image and Cosmetic changes on the datasheet
2.9	03/23/2018	Update of 4d kernel image and Cosmetic changes on the datasheet
2.10	03/28/2018	Cosmetic changes on the datasheet
2.11	04/11/2018	Cosmetic changes on the datasheet
2.12	07/14/2018	Update of 4d kernel image and cosmetic changes on the datasheet
2.13	03/16/2019	Cosmetic changes on the datasheet
2.14	02/08/2019	Cosmetic Changes to gen4 Primary Displays for Raspberry Pi and addition of Section – Latest Kernel Versions
2.15	14/04/2020	Cosmetic Changes to gen4 Primary Displays for Raspberry Pi and addition of Section – Latest Kernel Versions
2.16	08/27/2020	Added notes regarding using matching kernel versions
2.17	28/06/2021	Updated instructions to contain the latest known information

13. 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.

14. Contact Information

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

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

Website: www.4dsystems.com.au