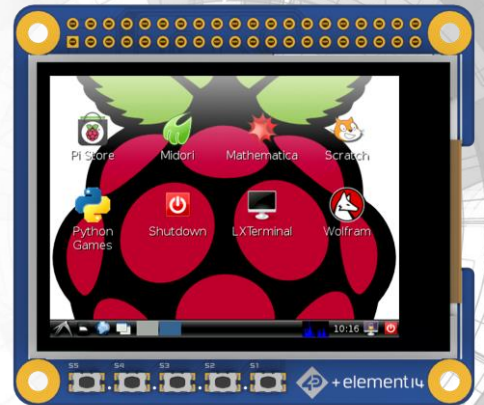


Primary Displays for Raspberry Pi

DATASHEET

DOCUMENT DATE: 14th APRIL 2020
DOCUMENT REVISION: 1.18



4DPi-24-HAT

Uncontrolled Copy when printed or downloaded.
Please refer to the 4D Systems website for the latest
Revision of this document

Table of Contents

1. Description	3
2. Features	3
3. Pin Configuration and Summary	4
4. Connecting the Display to the Pi	6
4.1. Hardware Connection	6
4.2. Software Download/Installation	6
4.3. Calibrating the Touch Screen	7
4.4. Change the Display Orientation	7
4.5. SPI Frequency and Compression	8
4.6. Backlight Control	8
4.7. Parameters Listing	8
4.8. HDMI or 4DPi Output	9
4.9. Changing DPI Output	9
4.10. Optimising appearance/sizes	9
5. Display Module Part Numbers	9
6. Mechanical Details	10
7. Schematic Diagram	11
8. Specifications	12
9. Appendix 1 – Code Examples – Push Buttons	13
9.1. Example for communicating to Push Buttons, for C:	13
9.2. Example for communicating to Push Buttons, for Python:	14
9.3. Example for Shutdown and Reset buttons, for C:	15
9.4. Example for Shutdown and Reset buttons, for Python:	16
10. Hardware Revision History	17
11. Document Revision History	18
12. Legal Notice	19
13. Contact Information	19

1. Description

The 4DPi-24-HAT is a 2.4" 320x240 Primary Display HAT for the Raspberry Pi, which plugs directly on top of a Raspberry Pi and displays the primary output which is normally sent to the HDMI or Composite output. It features an integrated Resistive Touch panel, enabling the 4DPi-24-HAT to function with the Raspberry Pi without the need for a mouse.

Communication between the 4DPi-24-HAT 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.

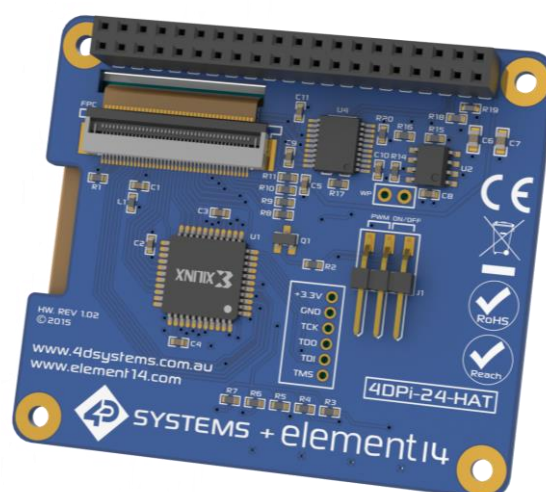
The 4DPi-24-HAT is designed to work with the Raspbian Operating System running on the Raspberry Pi, as that is the official Raspberry Pi operating system. It is also compatible with Stretch.

The 4DPi-24-HAT is powered directly off the Raspberry Pi's 40 way Header, and the 2.4" Touch Screen allows effortless interaction with the Raspbian Operating System.

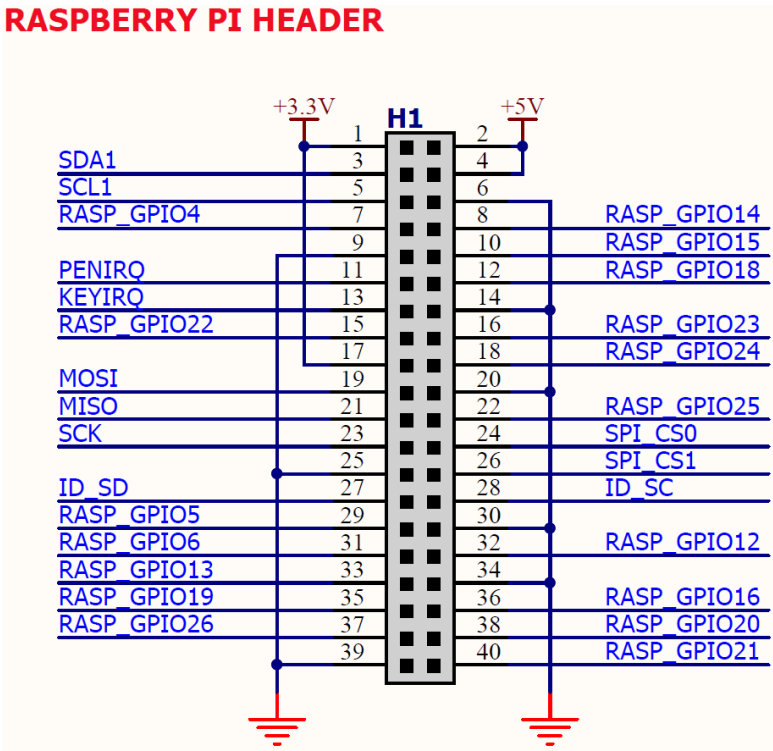
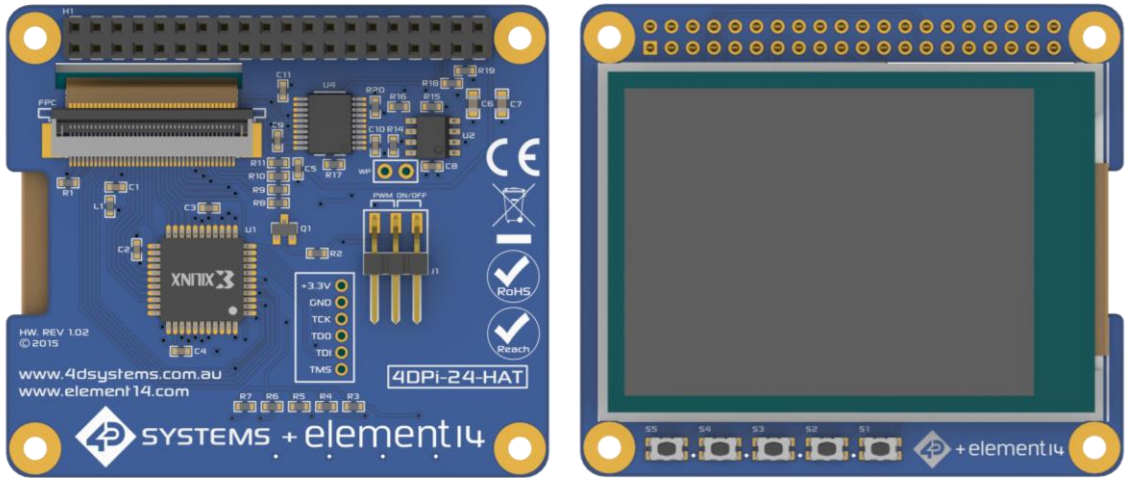
Note*: Raspberry Pi is a trademark of the Raspberry Pi Foundation, and all references to the words 'Raspberry Pi' or the use of its logo/marks are strictly in reference to the Raspberry Pi product, and how *this* product is compatible with but is not associated with the Raspberry Pi Foundation in any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

2. Features

- Universal 2.4" Primary Display for the Raspberry Pi.
- Compatible with Raspberry Pi A+, B+, Pi2, Pi3, Pi3 B+, Pi4, Zero and Zero W.
- 240 x 320 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 25FPS. Higher if image is able to be compressed by 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 identification EEPROM.
- Module dimensions: 56.5 x 65 x 14.2mm Weighing ~ 30g.
- Display Viewing Area: 49.0 x 36.7mm.
- 4x mounting holes with 2.6mm diameter for mechanical mounting.
- RoHS and CE Compliant.
- HAT Standard Compliant.



3. Pin Configuration and Summary



H1 Pinout (Raspberry Pi Connector on 4DPi-24-HAT-Adaptor) – Female Connector			
Pin	Symbol	I/O	Description
1	+3.3V	P	+3.3V Supply Pin, connected to the main 3.3V supply of the Raspberry Pi
2	+5V	P	+5V Supply Pin, connected to the main 5V supply of the Raspberry Pi
3	SDA1	I/O	I2C data line - used for touchscreen controller
4	+5V	P	+5V Supply Pin, connected to the main 5V supply of the Raspberry Pi
5	SCL1	O	I2C clock line - used for touchscreen controller
6	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
7	GPIO4	I/O	GPIO on the Raspberry Pi - unused
8	GPIO14	I/O	GPIO on the Raspberry Pi - unused
9	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
10	GPIO15	I/O	GPIO on the Raspberry Pi - unused
11	PENIRQ	I	Interrupt for the touchscreen controller
12	GPIO18	I/O	GPIO on the Raspberry Pi - unused
13	KEYIRQ	I	Interrupt for the push buttons
14	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
15	GPIO22	I/O	GPIO on the Raspberry Pi – Can be used for PWM Backlight (J1), else unused
16	GPIO23	I/O	GPIO on the Raspberry Pi - unused
17	+3.3V	P	+3.3V Supply Pin, connected to the main 3.3V supply of the Raspberry Pi
18	GPIO24	I/O	GPIO on the Raspberry Pi - unused
19	MOSI	O	MOSI Pin for the SPI
20	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
21	MISO	I	MISO Pin for the SPI
22	GPIO25	I/O	GPIO on the Raspberry Pi - unused
23	SCK	O	Clock Pin for the SPI
24	SPI_CS0	O	SPI - Chip select 4DPi-24-HAT
25	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
26	SPI_CS1	O	Chip Select Pin for the SPI - unused
27	ID_SD	I/O	I2C Data - ID EEPROM Interface
28	ID_SC	O	I2C Clock - ID EEPROM Interface
29	GPIO5	I/O	GPIO on the Raspberry Pi - unused
30	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
31	GPIO6	I/O	GPIO on the Raspberry Pi - unused
32	GPIO12	I/O	GPIO on the Raspberry Pi - unused
33	GPIO13	I/O	GPIO on the Raspberry Pi - unused
34	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
35	GPIO19	I/O	GPIO on the Raspberry Pi - unused
36	GPIO16	I/O	GPIO on the Raspberry Pi - unused
37	GPIO26	I/O	GPIO on the Raspberry Pi - unused
38	GPIO20	I/O	GPIO on the Raspberry Pi - unused
39	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
40	GPIO21	I/O	GPIO on the Raspberry Pi - unused

I = Input, O = Output, P = Power

4. Connecting the Display to the Pi

4.1. Hardware Connection

The 4DPi-24-HAT 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 alignment is correct and all pins are seated fully and correctly.



NOTE: The 4DPi-24-HAT is supported only by the 40 way header, and therefore pressing on the touch screen may result in the 4DPi-24-HAT 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 on the Pi using standoffs (not supplied). Some options are available from places like Mouser.

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-24-HAT and the Raspberry Pi so they do not touch inadvertently.

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.

- 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 4DPi-24-HAT 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 'raspberrypi' credentials, else SSH into your raspberrypi 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
https://4dsystems.com.au/media/downloads/
4DPi/All/gen4-hats_4-19-97.tar.gz

Install:

$ sudo tar -xzf gen4-hats_4-19-97.tar.gz
-C /
```


The package automatically selects the kernel required for Pi1, Pi2, Pi3, or Pi4 automatically. If you want to check for the kernel version released by 4D Systems, proceed to [Section: Latest Kernel Revisions](#).

- 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 4DPi-24-HAT to the Raspberry Pi, and reapply power. The terminal should begin to show on the 4DPi-24-HAT, and will be ready to use once the Raspberry Pi has booted.

4.3. Calibrating the Touch Screen

Each 4DPi-24-HAT 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 4DPi-24-HAT, 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/
```

You 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

```
/etc/X11/xorg.conf.d/99-calibration.conf
```

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

- 8) Save the file and perform a reboot

```
$ sudo reboot now
```

You should now have a perfectly calibrated 4Dpi-24-HAT.

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.

```
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.

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

4.5. SPI Frequency and Compression

The 4DPi-24-HAT 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.

Please refer to Section 4.7 for more information

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, GPIO22 is used.

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

Controlling the backlight using ON/OFF control requires root access.

```
$ sudo su
```

Executing the following command will control the backlight:

To turn the backlight OFF:

```
# echo 0 > /sys/class/backlight/4d-hats/brightness
```

To turn the backlight ON:

```
# 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%.

```
# echo 80 > /sys/class/backlight/24-hat-pwm/brightness
```

The above will set the backlight to 80%. Simply change the 'echo 80' to be anything from 0 to 100.

4.7. Parameters Listing

The following is a list of all the custom parameters used by the 4DPi-24-HAT.

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.

4.8. 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
```

4.9. 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

4.10. Optimising appearance/sizes

The 4DPi-24-HAT has a 320x240 resolution display, so expectations need to be realistic when dealing with the content which is viewed on this 2.4" 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 (eg Bluetooth)

(close window [x])

Raspberry Menu->Shutdown->Reboot

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.

5. Display Module Part Numbers

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

Example:

4DPi-24-HAT

4DPi - Display Family

24 - Display size (2.4")

HAT - Hardware attached on top.

6. Latest Kernel Versions

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

Latest release:

[gen4-hats 4-19-57-v7l+ v1.0.tar.gz](#)

Previous releases:

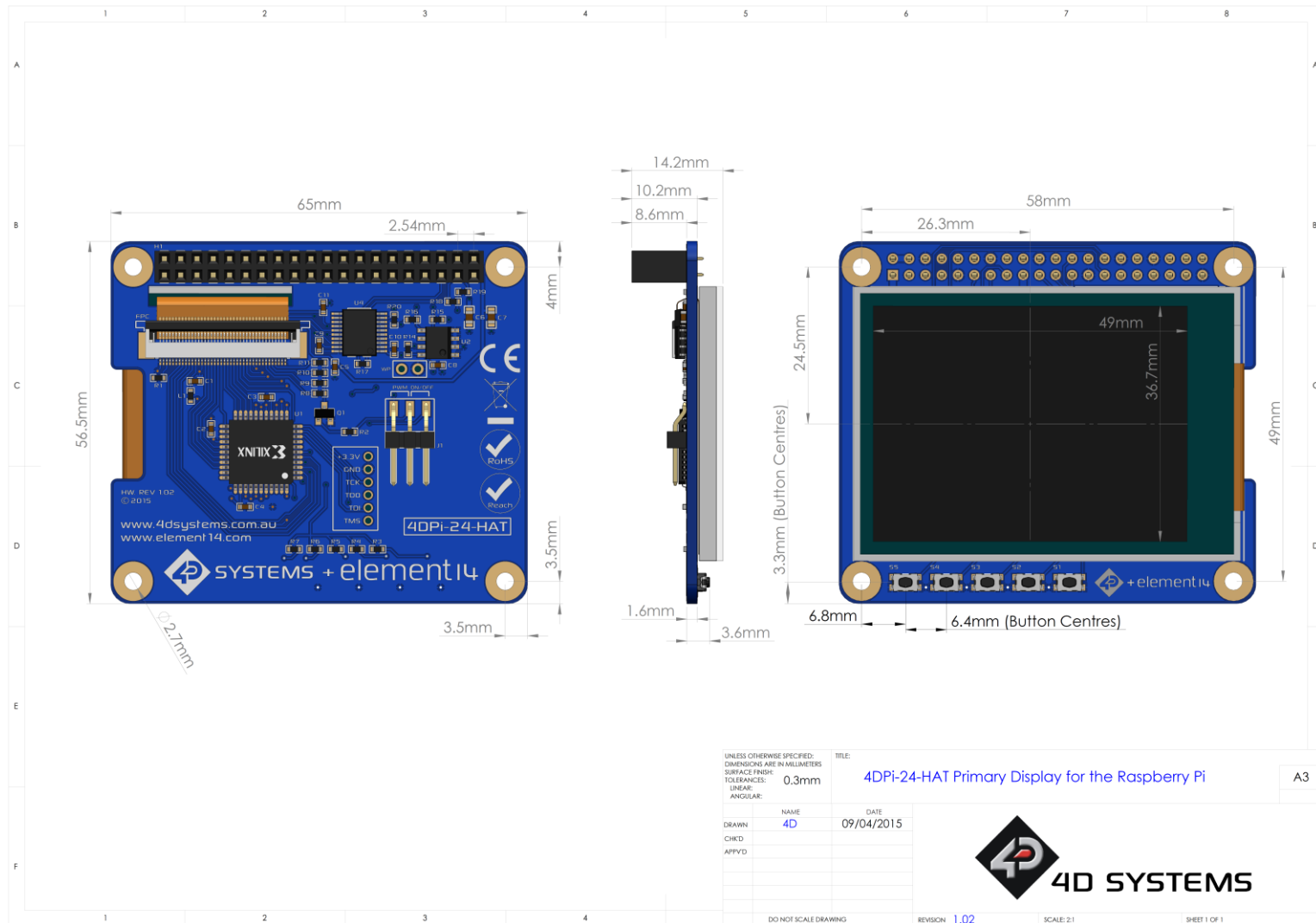
[gen4-hats 4-14-34 v1.1.tar.gz](#)

[gen4-hats 4-9-80 v1.1.tar.gz](#)

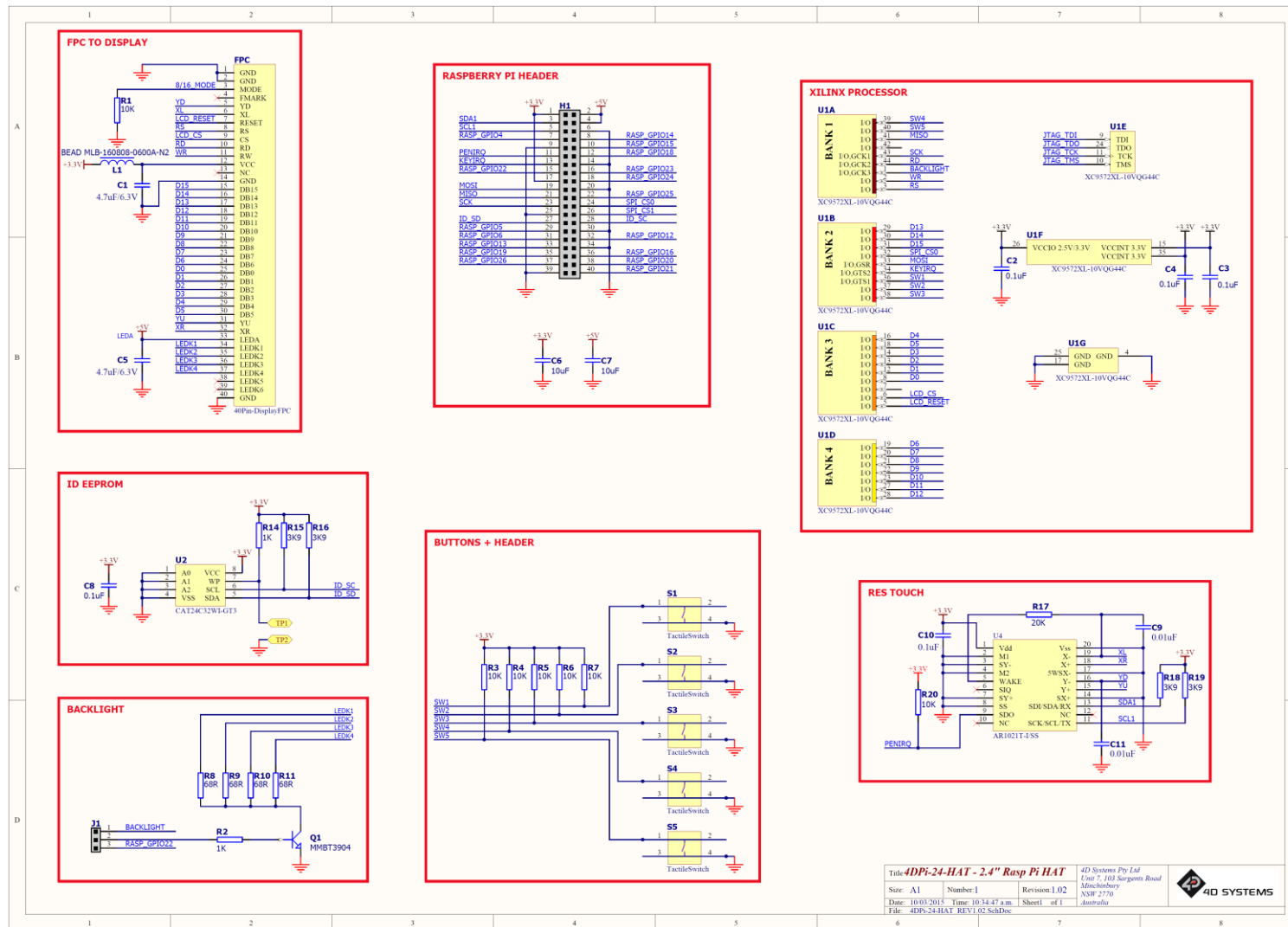
[gen4-hats 4-9-59 v1.2.tar.gz](#)

Note: Older kernel releases are 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

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, 240x320. A higher FPS can be achieved if display outputting lots of blocks of the same colour	--	25	--	FPS

ORDERING INFORMATION

Order Code: 4Dpi-24-HAT

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;
}
```

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
1.02	10/03/2015	Initial Public Release Version

12. Document Revision History

Revision Number	Date	Description
1.0	08/05/2015	Initial Public Release Version
1.1	17/07/2015	Update of 4d kernel image and cosmetic changes on the datasheet
1.2	24/07/2015	Update of 4d kernel image and cosmetic changes on the datasheet
1.3	16/10/2015	Update of 4d kernel image and cosmetic changes on the datasheet
1.4	10/03/2016	Update of 4d kernel image and cosmetic changes on the datasheet
1.5	11/04/2016	Update of 4d kernel image and cosmetic changes on the datasheet
1.6	24/05/2016	Update of 4d kernel image and cosmetic changes on the datasheet
1.7	21/10/2016	Update of 4d kernel image and cosmetic changes on the datasheet
1.8	11/01/2017	Update of 4d kernel image and cosmetic changes on the datasheet
1.9	24/03/2017	Update of 4d kernel image and cosmetic changes on the datasheet
1.10	01/08/2017	Update of 4d kernel image and cosmetic changes on the datasheet
1.11	07/03/2018	Update of 4d kernel image and cosmetic changes on the datasheet
1.12	23/03/2018	Update of 4d kernel image and cosmetic changes on the datasheet
1.14	06/04/2018	Update of 4d kernel image and cosmetic changes on the datasheet
1.15	14/07/2018	Update of 4d kernel image and cosmetic changes on the datasheet
1.16	13/03/2019	cosmetic changes on the datasheet
1.17	02/08/2019	Update of 4d kernel image, cosmetic changes on the datasheet and addition of Section 6– Latest Kernel Versions
1.18	14/04/2020	Update of 4D kernel image and cosmetic change on the datasheet

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 make 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

Copyright 4D Systems Pty. Ltd. 2000-2019.