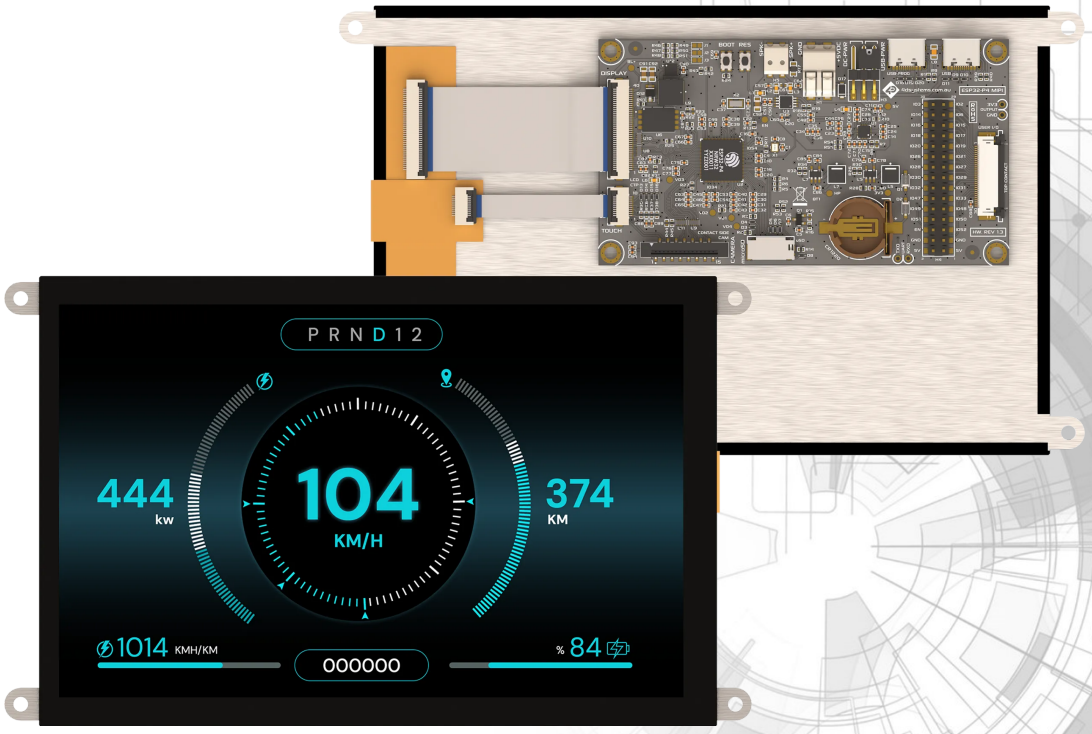
ESP32-P4 Series



ESP32-P4-70[CT][-CLB][-SB] - (7" ESP32-P4 MIPI Display Series)

ESP32-P4-80[CT][-CLB][-SB] - (8" ESP32-P4 MIPI Display Series)

ESP32-P4-90[CT][-CLB][-SB] - (9" ESP32-P4 MIPI Display Series)

ESP32-P4-101[CT][-CLB][-SB] - (10.1" ESP32-P4 MIPI Display Series)

* - CT indicates Capacitive Touch variants

* - CLB indicates Cover Lens Bezel variants

* - SB indicates Super Bright variants

Datasheet

Revision 1.1

Copyright © 2026 4D Systems

Content may change at any time. Please refer to the resource centre for latest documentation.

Contents

1. Description 4
2. Features 6
3. Hardware Overview 8
   3.1. 30-way FFC Interface Pinout 8
   3.2. 30-way Male Pin Header Pinout 10
4. Hardware Interface - Pins 11
   4.1. Serial Ports - 3.3V TTL 11
   4.2. Communications - I2C / SPI etc 11
   4.3. General Purpose I/O 11
5. Module Features 12
   5.1. ESP32-P4NRW32 Processor 12
   5.2. Chipsets used 13
   5.3. SD/SDHC Memory Cards 14
   5.4. microSD Socket Usage 15
   5.5. FAT16 vs FAT32 15
   5.6. Power Selection Jumper 16
   5.7. DC Terminal Header (H1) 16
   5.8. Audio Amplifier / Speaker Output 17
   5.9. CSI MIPI Camera Interface 17
   5.10. Real Time Clock - Battery 17
   5.11. WiFi / BT Support 17
6. Display/Module Precautions 19
7. Hardware Tools 20
   7.1. 4D-UPA 20
   7.2. USB-C Cable 22
8. Software Tools 23
   8.1. Workshop4 IDE 23
   8.2. Workshop5 IDE 24

8.3. Workshop Built-in Tools 25
9. Programming Language 26
10. ESP-IDF Development 27
10.1. LCD Initialization Codes 28
11. Display Module Part Numbers 44
12. Cover Lens Bezel - Tape Spec 45
13. FFC Cable 45
14. Starter Kit 46
15. Mechanical Details 47
15.1. 7" Non-Touch, Capacitive Touch 47
15.2. 7" Capacitive Touch w/ CLB 48
15.3. 8" Non-Touch, Capacitive Touch 49
15.4. 8" Capacitive Touch w/ CLB 50
15.5. 8" Non-Touch, Capacitive Touch 51
15.6. 9" Capacitive Touch w/ CLB 52
15.7. 10.1" Non-Touch, Capacitive Touch 53
15.8. 10.1" Capacitive Touch w/ CLB 54
16. Schematic Circuit Details REV 1.4 55
17. Specifications 56
18. Revision History 60
19. Legal Notice 61
19.1. Proprietary Information 61
19.2. Disclaimer of Warranties & Limitations of Liabilities 61

1. Description

The ESP32-P4 Series of Intelligent Display Modules is designed and manufactured by 4D Systems.

These ESP32-P4 modules are available in 7.0", 8.0", 9.0" and 10.1", and use a MIPI Interface between the ESP32-P4 Processor and the 1280x800 resolution TFT LCD Displays. The displays are IPS TFT LCD's, with wide viewing angles, suitable for viewing from all direction.

Available in Non-Touch, Capacitive Touch, and Capacitive Touch with Cover Lens Bezel (CLB). Resistive Touch is not available in this product line at this time.

The ESP32-P4 Processor provides multiple GPIO which include UART, SPI, I2C, PWM and Analog functionality, while also serving a DSI MIPI interface for the LCD Touch screen, a CSI MIPI interface for Camera input, Real Time Clock, Quad SPI Flash, microSD Card, 2 different USB-C ports, and I2S Audio Output with Amplifier.

The user interface to the ESP32-P4 series is a 30-pin FPC/ZIF socket designed for a 30-way 0.5mm pitch FFC cable and compatible with the 4D-UPA Univeral Programmer, along with a 30-pin 2.54mm pitch male pin header for easy and simple connection to an application or motherboard, or for connecting to accessory boards for a range of functionality advancements.

This series of ESP32-P4 modules are compatible with the 4D Systems Workshop4 IDE, and the Worskhop5 IDE (Coming Soon), utilising the Espressif compiler and 4D Systems purpose built libraries, allowing a feature rich design and programming experience. It is also possible to use other IDE's and libraries such as LVGL.

Any code designed and written to run on other 4D Systems display modules, such as modules featuring GOLDELOX, PICASO, PIXXI or DIABLO-16 Graphics Processors, are unfortunately not compatible with the ESP32-P4 range due to being a totally different processor family. However, please contact 4D Systems Support Team for assistance if you are planning on migrating from a different 4D Systems display model, as there are some similarities between them - such as the graphics, however a majority of the coding will have to be adapted.

Any code designed and written to run on the 4D Systems ESP32-S3 range of modules, should be easily ported to this ESP32-P4 range, without too many problems. The main difference will be the increase in resolution, so projects will have to be rescaled accordingly.

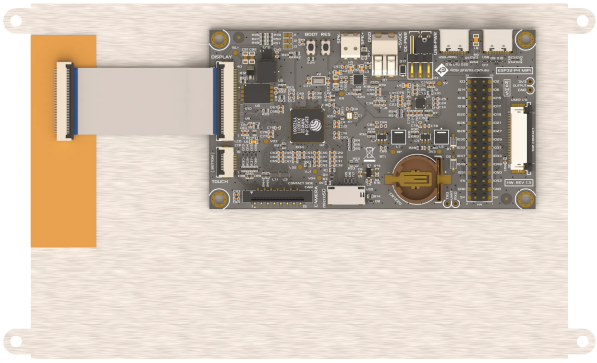
From a mechanical perspective, the ESP32-P4 Non-Touch and Capacitive Touch modules feature 4 metal mounting tabs on the side of the module which are used to mechanically fasten the display module into your choice of enlcosure or housing using bolts/screws - as required. The Capacitive Touch with Cover Lens Bezel does not feature any mounting tabs, and instead features an overhanging glass bezel over the edge of the display module, and on this edge is 3M Adhesive tape which is used to attach to a suitable enclosure or housing instead.

These ESP32-P4 modules can be programmed with the USB-C Programming port, directly to your PC, or you can also program them using the 4D Systems 4D-UPA Programmer over the Serial UART, which doubles as a UART sniffer for debugging code during development. GPIO is also available to be utilised via the 4D-UPA if required, or

directly to the 30-pin male pin headers on the ESP32-P4 module itself. Take note that the exact GPIO available on the 30-way FFC connector and the 30-way pin header, are not identical.



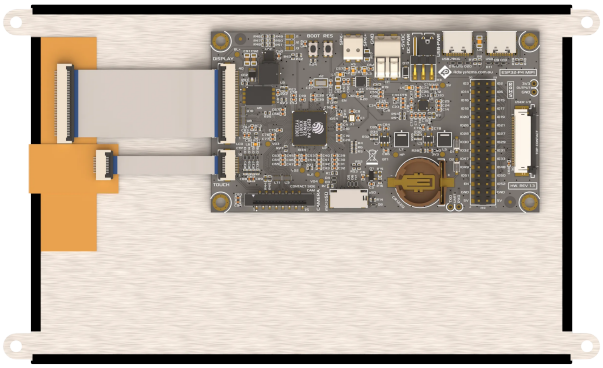
ESP32-P4-70 Front



ESP32-P4-70 Rear



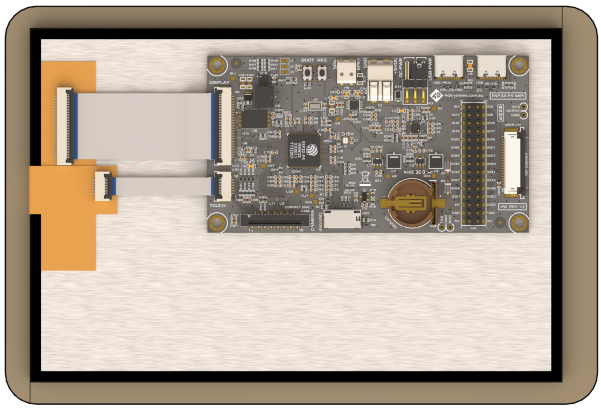
ESP32-P4-70CT Front



ESP32-P4-70CT Rear



ESP32-P4-70CT-CLB Front



ESP32-P4-70CT-CLB Rear

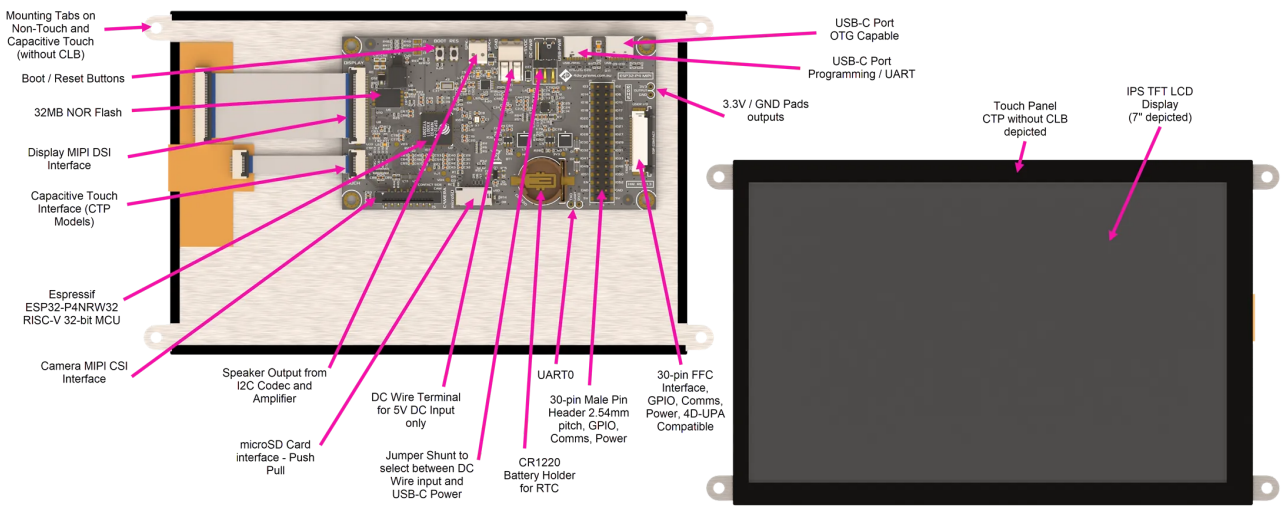
2. Features

- Powerful ESP32-P4 RISC-V 32-bit dual-core Graphics Processor by Espressif, with low power single-core.
- 1280x800 resolution MIPI displays.
- 16-bit RGB565 (recommended) or 24-bit RGB888 (*4D libraries use 16-bit colors*)
- IPS TFT LCD display, in 7", 8", 9" and 10.1" sizes.
- Available in Non-Touch, Capacitive Touch, and Capacitive Touch with Cover Lens Bezel (CLB).
- 32MB of External Quad SPI NOR Flash.
- 32MB of Internal PSRAM.
- 128KB HP ROM, 16KB LP ROM, 768KB HP L2MEM, 32KB LP SRAM, 8KB TCM
- Up to 28 GPIO
- On-board Real Time Clock with battery backup (CR1220) - *battery not included*
- I2S Audio Codec and 3W Mono Class-D Amplifier to drive a speaker - *speaker not included*
- 2x USB-C Ports, 1 for Programming/UART and 1 for OTG applications
- DC Push Terminals for 5V/GND direct wiring for powering the module
- Jumper Shunt to select between DC Terminal or USB-C Powering of the module
- 30pin FPC connection, for power, communications, GPIO and UART programming via 4D-UPA Programmer.
- 30pin male pin header, for power, communications, and GPIO.
- Push-Pull micro-SD memory card connector on SDIO bus for multimedia storage and data logging purposes.
- Display full colour images, animations, icons and video clips.
- 4.0V to 6.0V range operation (single supply), 5V recommended. ** CHECK **
- 4x metal mounting tabs built into the LCD for mechanical mounting using bolts/screws (non CLB models only).
- 3M Adhesive around perimeter of Cover Lens Bezel for mounting the CTP-CLB model.
- RoHS and REACH compliant.
- CE/EMC and UKCA compliance pending.
- PCB is UL 94V-0 Flammability Rated.
- Module dimensions:
    - (7.0" non-Touch): 181.7 x 109.3 x 15.5mm
    - (7.0" Capacitive Touch): 181.7 x 109.3 x 17.2mm
    - (7.0" Capacitive Touch w/ CLB): 184.7 x 124.2 x 17.2mm

- (8.0" non-Touch): 208.4 x 122.7 x 15.5mm
- (8.0" Capacitive Touch): 208.4 x 122.7 x 17.2mm
- (8.0" Capacitive Touch w/ CLB): 206.2 x 137.6 x 17.2mm
- (9.0" non-Touch): 230.2 x 136.1 x 15.5mm
- (9.0" Capacitive Touch): 230.2 x 136.1 x 17.2mm
- (9.0" Capacitive Touch w/ CLB): 231.8 x 153.7 x 17.2mm
- (10.1" non-Touch): 249.2 x 148.4 x 15.5mm
- (10.1" Capacitive Touch): 249.2 x 148.4 x 17.2mm
- (10.1" Capacitive Touch w/ CLB): 252.6 x 167.4 x 17.2mm

- Weighing (approximately):
  - (7.0" non-Touch): ~ TBA g
  - (7.0" Capacitive Touch): ~ 230 g
  - (7.0" Capacitive Touch w/ CLB): ~ TBA g
  - (8.0" non-Touch): ~ TBA g
  - (8.0" Capacitive Touch): ~ 285 g
  - (8.0" Capacitive Touch w/ CLB): ~ TBA g
  - (9.0" non-Touch): ~ TBA g
  - (9.0" Capacitive Touch): ~ 345 g
  - (9.0" Capacitive Touch w/ CLB): ~ TBA g
  - (10.1" non-Touch): ~ TBA g
  - (10.1" Capacitive Touch): ~ 415 g
  - (10.1" Capacitive Touch w/ CLB): ~ TBA g

3. Hardware Overview



Hardware Layout (7.0" CTP Module depicted)

3.1. 30-way FFC Interface Pinout

Pin	Symbol	I/O	Description
1	GND	P	Supply Ground
2	GPIO2	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
3	GPIO3	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
4	GPIO6	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
5	GPIO14	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
6	GPIO15	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
7	GPIO16	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
8	GPIO17	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
9	GPIO18	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
10	GPIO19	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
11	GPIO20	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
12	GPIO21	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
13	GPIO26	I/O	General Purpose I/O. This pin is 3.3V Logic only.
14	GPIO27	I/O	General Purpose I/O. This pin is 3.3V Logic only.
15	GPIO28	I/O	General Purpose I/O. This pin is 3.3V Logic only.
16	GPIO29	I/O	General Purpose I/O. This pin is 3.3V Logic only.
17	GPIO30	I/O	General Purpose I/O. This pin is 3.3V Logic only.
18	GPIO31	I/O	General Purpose I/O. This pin is 3.3V Logic only.
19	GPIO35	I/O	Available after boot, GPIO35 is used for Programming and Boot Strapping. General Purpose Input/Output. This pin is 3.3V Logic only.
20	3.3V	P	3.3V Output for User, connected to system 3.3V bus. Excessive draw will affect system stability. 100mA-200mA draw is OK
21	GND	P	Supply Ground

Pin	Symbol	I/O	Description
22	ESP-EN	I	Master Reset/Enable signal. Internally pulled up to 3.3V via a 10K resistor. Low will disable the chip, High/floating will activate the chip. Used by a UART programmer for programming sequence.
23	GPIO38	I/O	Asynchronous Serial Receive pin, 3.3V TTL level. Connect this pin to the Transmit (Tx) signal of other serial devices. Used in conjunction with the U0TXD pin for UART programming or communications. This pin is 3.3V Logic only.
24	GPIO37	I/O	Asynchronous Serial Transmit pin, 3.3V TTL level. Connect this pin to the Receive (Rx) signal of other serial devices. Used in conjunction with the U0RXD pin for UART programming or communications. This pin is 3.3V Logic only.
25	GND	P	Supply Ground
26	5V IN	P	Voltage Supply +ve input pin. The range is 4.0V to 6.0V, nominal 5.0V.
27	5V IN	P	Voltage Supply +ve input pin. The range is 4.0V to 6.0V, nominal 5.0V.
28	5V IN	P	Voltage Supply +ve input pin. The range is 4.0V to 6.0V, nominal 5.0V.
29	N/C	-	Not Connected
30	GND	P	Supply Ground

Note

1. **I** = Input, **O** = Output, **P** = Power, **A** = Analog Input
2. It is recommended to connect all 3 5V IN pins to a stable 5V DC power supply, as well as at least 3 or more GND pins, when powering over the FFC cable.
3. Please refer to the Espressif ESP32-P4 datasheet for specific detail on the capability of the ESP32-P4 GPIO, in conjunction with the schematic of this module.

3.2. 30-way Male Pin Header Pinout

Pin	Symbol	I/O	Description
1	5V IN	P	Voltage Supply +ve input pin. The range is 4.0V to 6.0V, nominal 5.0V.
2	5V IN	P	Voltage Supply +ve input pin. The range is 4.0V to 6.0V, nominal 5.0V.
3	GND	P	Supply Ground
4	GND	P	Supply Ground
5	GPIO52	I/O	General Purpose I/O. This pin is 3.3V Logic only.
6	ESP-EN	I	Master Reset/Enable signal. Internally pulled up to 3.3V via a 10K resistor. Low will disable the chip, High/floating will activate the chip. Used by a UART programmer for programming sequence.
7	GPIO50	I/O	General Purpose I/O. This pin is 3.3V Logic only.
8	GPIO51	I/O	General Purpose I/O. This pin is 3.3V Logic only.
9	GPIO48	I/O	General Purpose I/O. This pin is 3.3V Logic only.
10	GPIO49	I/O	General Purpose I/O. This pin is 3.3V Logic only.
11	GPIO33	I/O	General Purpose I/O. This pin is 3.3V Logic only.
12	GPIO47	I/O	General Purpose I/O. This pin is 3.3V Logic only.
13	GPIO31	I/O	General Purpose I/O. This pin is 3.3V Logic only.
14	GPIO32	I/O	General Purpose I/O. This pin is 3.3V Logic only.
15	GPIO29	I/O	General Purpose I/O. This pin is 3.3V Logic only.
16	GPIO30	I/O	General Purpose I/O. This pin is 3.3V Logic only.
17	GPIO27	I/O	General Purpose I/O. This pin is 3.3V Logic only.
18	GPIO28	I/O	General Purpose I/O. This pin is 3.3V Logic only.
19	GPIO21	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
20	GPIO26	I/O	General Purpose I/O. This pin is 3.3V Logic only.
21	GPIO19	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
22	GPIO20	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
23	GPIO17	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
24	GPIO18	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
25	GPIO15	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
26	GPIO16	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
27	GPIO6	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
28	GPIO14	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
29	GPIO2	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.
30	GPIO3	I/O/A	General Purpose I/O. This pin is 3.3V Logic only.

Note

1. **I** = Input, **O** = Output, **P** = Power, **A** = Analog Input
2. It is recommended to connect both 5V IN pins to a stable 5V DC power supply, as well as both GND pins.
3. Please refer to the Espressif ESP32-P4 datasheet for specific detail on the capability of the ESP32-P4 GPIO, in conjunction with the schematic of this module.

4. Hardware Interface - Pins

This section describes in detail the hardware interface pins of the module.

4.1. Serial Ports - 3.3V TTL

The ESP32-P4 is capable of up to 5 (HP) UART's via the programmable GPIO pins, however the main UART0 is found on GPIO38 and GPIO39 which is utilised for the 4D-UPA on the 30-way FFC connector, or on the bottom edge of the PCBA as pads. UART0 is where all of the boot and debug messages are sent by default, and is useful when diagnosing issues or troubleshooting your code. The module is capable of turning the available native GPIO into more serial UART's, based on the functionality of the ESP32-P4 processor. Please refer to the Espressif ESP32-P4 Datasheet for more details.

The module can be programmed over UART (rather than USB-C) utilising the U0RXD/U0TXD pins, in conjunction with ESP-EN and GPIO35. This is fully compatible with the 4D Systems 4D-UPA Univeral Programmer, via the 30-way FFC cable. Please refer to the 4D-UPA Programmer, in the Hardware Tools section.

4.2. Communications - I2C / SPI etc

The ESP32-P4 is capable of all sorts of communications via its GPIO ports. To best utilise these, and for the most up to date information, please refer to the Espressif ESP32-P4 datasheet.

4.3. General Purpose I/O

There are up to 28 general-purpose Input/Output (GPIO) pins available to the user on these ESP32-P4 modules.

Please refer to the Espressif ESP32-P4 datasheet for information on how to configure the GPIO for various functions, such as Digital-Input/Digital-Output, Analog Input, I2C, UART, SPI, etc.

5. Module Features

The ESP32-P4 Series is designed to accommodate a wide variety of applications. Some of the main features of the module are listed below.

5.1. ESP32-P4NRW32 Processor

The module is designed around the ESP32-P4NRW32 Processor from Espressif. This model of ESP32-P4 has 32MB of built in PSRAM, and utilises 32MB of External NOR Flash for application storage. Media is typically stored on a micro-SD card, however some types of media can be stored in SPI Flash, but it is limited for capacity. Depending on your application and code requirements, it is entirely possible to fit the entire application and its resources in the 32MB of NOR Flash.

The ESP32-P4 is capable of outputting I2S Audio, which is utilised on these modules. It is fed into an I2S Codec, and then into a Mono 3W Class-D Amplifier, which is capable of driving a 4-ohm to 8-ohm speaker.

5.2. Chipsets used

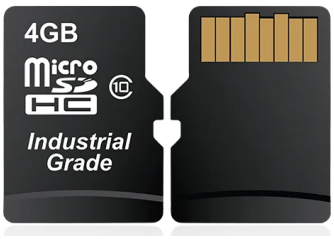
The ESP32-P4 Series of modules utilises a few chipsets from various manufacturers, in order for these modules to operate. Please refer to the Schematic for connection details.

- The main processor is an Espressif ESP32-P4NRW32, as mentioned in the previous section.
- The MIPI LCDs feature a JD9365 driver. It interfaces with the ESP32-P4 via its dedicated DSI pins. The reset pin is connected to GPIO23. For timing requirements and initialization code, please refer to the provided software libraries and examples.
- The backlight is controlled using a Silergy SY7023 backlight driver, which takes On/Off/PWM input from the ESP32-P4 using GPIO22.
- For Capacitive Touch models, the chipset for the capacitive touch is I2C driven (GPIO 7/8), and the chipset is found on the Display flex itself. These utilise Goodix GT928 controllers.
- The Real Time Clock is part of the ESP32-P4 itself, utilising the LP Processor inside the chipset. This is battery backed up with a CR1220 coin cell battery. *Battery is not included.*
- The Quad SPI Flash memory used on these modules, is the Giga-Devices GD25Q256EYIGR, which is 32MB in capacity, and interfaces to the ESP32-P4 on dedicated FLASH pins.
- The micro-SD card interface, while not a chipset, is still worthy of note. It utilises a dedicated SDIO bus from the ESP32-P4 (GPIO 39-46).
- The I2S Audio Codec is an Everist Semiconductor ES8311, which connects to the ESP32-P4 over the I2S Bus (GPIO 9-13), as well as the I2C Bus (GPIO 7/8).
- The Audio Amplifier is an NSIWay NS4150B 3W Class-D Mono Amplifier, and connects to the output of the I2S Audio Codec.

5.3. SD/SDHC Memory Cards

The ESP32-P4 modules use off-the-shelf standard SDHC/SD/microSD memory cards with up to 4GB capacity usable with FAT16 formatting, and much higher with FAT32 formatting. For any FAT file-related operations, before the memory card can be used it must first be formatted. The formatting of the card can be done on any PC system with a card reader.

Cards with a FAT16 formatting (4GB or under partition) are capable of operating faster on this display module, compared to the same card (16GB for example) with a single FAT32 partition, due to the nature of FAT16 vs FAT32 file transfers. If your application media can fit inside a 4GB partition, it is recommended to use FAT16 to gain the maximal speed possible.



RMPET, a 4D Systems Tool found in the Workshop4 IDE, is capable of repartitioning and formatting microSD cards for FAT16, to be the appropriate type and format. This tool should be used for all cards as it also employs an offset which is critical when using Industrial microSD cards which feature Read Disturb Prevention firmware, which is a special firmware inside the microSD card designed to prevent Read Disturb occurring on NAND based Flash media. Further discussed in the note.

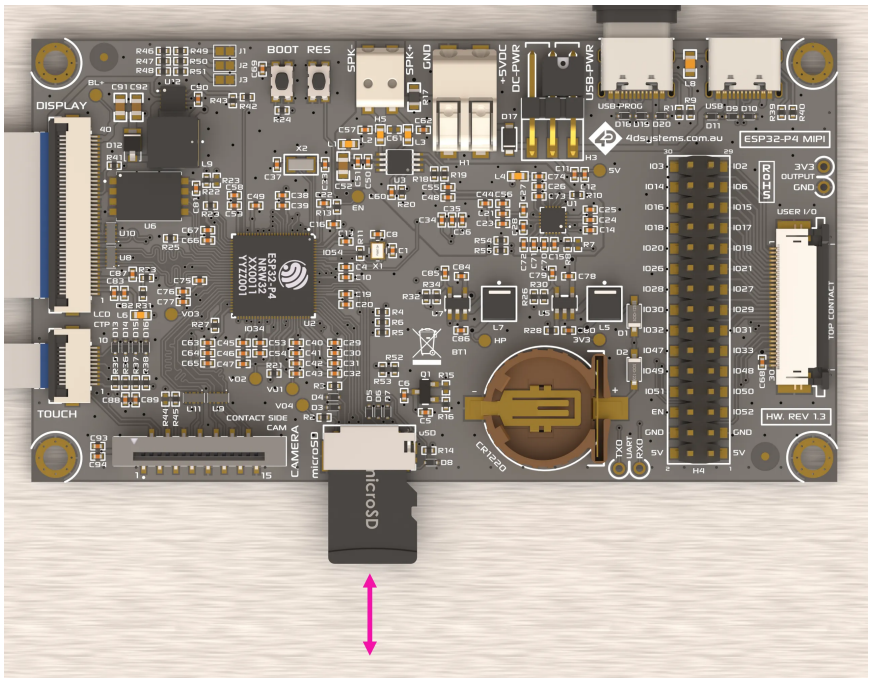
Note

Read disturbance is a well-known issue with flash memory devices, such as microSD cards, where reading data from a flash cell can cause the nearby cells in the same memory block to change over time. This issue can be prevented by using industrial-grade microSD cards with read disturb protection. Industrial-grade microSD cards have firmware that actively monitors the read operation and refreshes areas of memory that have high traffic and even move data around to prevent read disturb error from occurring. Furthermore, manufacturers may choose to implement read disturb protection on a specific part of the flash memory only, such that the beginning part of the memory might not be protected. The RMPET utility in Workshop4 is designed to create the first partition at an offset from the start of the microSD card to account for this situation. It is therefore recommended to always partition and format an industrial microSD card using the RMPET utility before using it with 4D Systems modules. Many commercial grade cards designed for Cameras etc, do not handle read disturb well at all, and therefore it is always recommended to use an Industrial grade microSD card with 4D modules. 4D offers one that is tried and tested, on our website.

5.4. microSD Socket Usage

On the ESP32-P4 modules is a Push-Pull type microSD socket. The use of this socket is easy, and is accessed from the outside of the PCBA.

Push the microSD card into the socket with the contacts facing down towards the LCD, and press the microSD card home into the socket. To remove it, simply pull it out.



microSD Socket with microSD card inserted

5.5. FAT16 vs FAT32

FAT16 is capable of having a partition with up to 4GB capacity usable. While this might seem like a limitation, it still offers the best performance for small processor systems such as the ESP32-P4. Larger partitions are possible with FAT32 formatting, however smaller cluster size results, giving slightly worse performance.

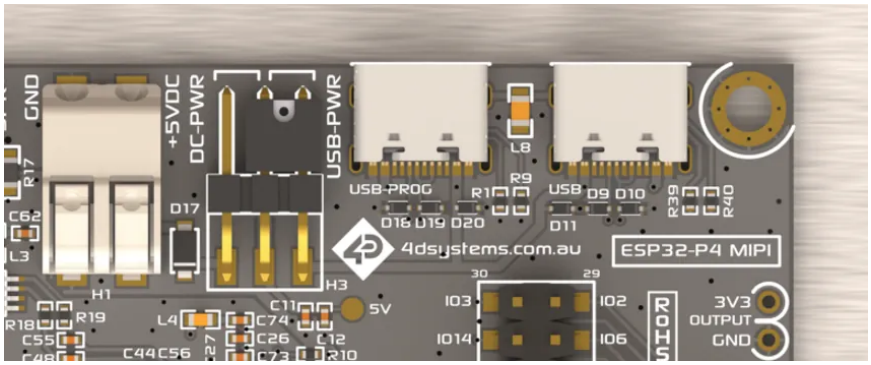
For any FAT file-related operations, before the memory card can be used it must first be formatted correctly. Built into Workshop4 is a tool created by 4D, called RMPET (please refer to the Tools menu, in any Environment, inside the Workshop4 IDE). RMPET allows the User to easily partition and format microSD cards, to make their file system ready to be used with 4D Systems modules. The formatting of the card can be done on any Windows PC system with a card reader.

5.6. Power Selection Jumper

The H3 jumper at the top centre of the PCBA, is for selecting the source of Power to run the module.

When the Jumper Shunt is between the Centre and Right Pin, Power to drive the module comes from the USB-C Ports. Both USB-C ports are in parallel for power, so either could be used to supply power.

When the Jumper Shunt is between the Centre and Left Pin, Power to drive the module comes from the DC Terminal header (H1), to the left of the Power Selection Jumper header. These are spring terminals where you simply press down on the top and insert/remove wires as required. The Right side is Positive (+) and the Left side is Negative (-). This DC Terminal header (H1) is reverse polarity protected by a diode - thus the 5V rail will be 0.2 to 0.3V lower than the input voltage - this will cause no problems on the module itself.



DC Terminal header (H1), Power Selection Jumper (H3) with Shunt, USB Ports (USB-PROG and USB)

It is also possible to power the module using the 5V and GND pins found on the 30-way FFC connector, as well as the 30-way male pin header. These bypass the Power Selection jumper and connect directly to the 5V rail. These have no reverse polarity protection, so take note, however they can also be 5V output if there is a need to power something else from this module, such as a sensor or daugher board.

5.7. DC Terminal Header (H1)

The DC Terminal Header (H1), as mentioned in the Power Selection Jumper section, enables wires to be connected directly to the module and to power the module with 5VDC (or within the specification input voltage).

Simply push in sold core or soldered wire ends, and gently pull out to remove or more ideally pushing down on the release button on the top of the connector as you remove the wire.

The H1 terminal connector powers the 5V rail via a diode, for limited reverse voltage protection. Always be sure to connect the wires with the correct polarity, however the diode is there to prevent perminant damage to your module. Please refer to the PCB Markings to show the Positive (+) and Negative (-) sides of the connector.

5.8. Audio Amplifier / Speaker Output

The H5 header at the top of the PCBA, to the right of the BOOT and RES buttons, is to enable the easy connection of a small speaker to the ESP32-P4 module. Simply push in sold core or soldered wire ends, and gently pull out to remove.

The Audio Amplifier is a Mono Class-D amplifier, and is capable of outputting up to 3W, depending on speaker load. A 4ohm to 8ohm speaker is recommended.

The Audio Amplifier gets its audio information from the I2S Audio Codec, which is connected via I2S and I2C buses to the ESP32-P4 itself.

5.9. CSI MIPI Camera Interface

The CSI MIPI Camerea Interface is a standard 15pin 1.0mm interface, designed for cameras such as the Raspberry Pi Camera REV 1.3 and compatible variants, using chipsets such as the OV5647.

Please refer to the ESP32-P4 Datasheet for up to date information on what cameras and chipsets are compatible.

The 15 pin connector on the 4D Systems ESP32-P4 modules are oriented as such that the contacts on the cable face upward towards the ESP32-P4 chip on the PCBA. It is marked on the PCBA indicating this also. Pin 1 on the left is 3.3V, while Pin 15 on the right is GND. Please refer to the Schematic for the detailed pinout.

The connector is a vertical connector, similar to what is found on Raspberry Pi's. The top surface of the connector is a lock, and can be raised up about 1mm or so, which unlocks the connector. The cable can then be put into the connector, and then the top surface of the connector pushed down again to lock it in place.

5.10. Real Time Clock - Battery

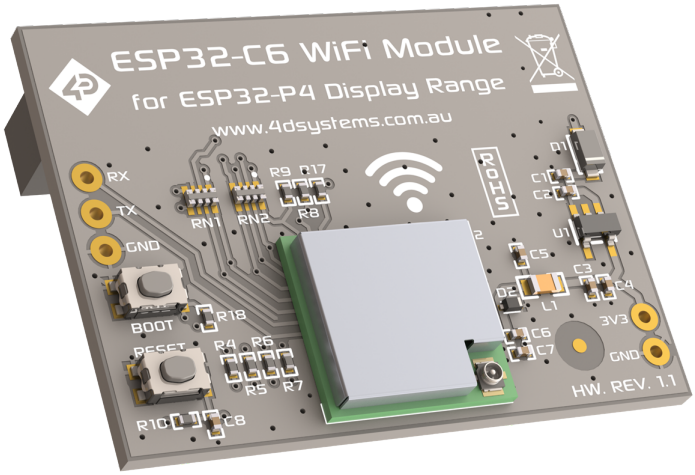
On the board is a CR1220 battery holder, desighed for 3V CR1220 coin cell batteries. The batteries cannot be charged in the ESP32-P4 modules, however should last a long time given their sole purpose is to provide the ESP32-P4 with 3V to keep the Real Time Clock operational when the main display module power is removed. This battery is not for powering the module in any other way, and is only tied to the VBAT pin on the ESP32-P4.

The battery is **NOT INCLUDED** with the module, only the battery holder is supplied as part of the PCBA.

Please refer to the appropriate Espressif documentation on the operation of the Real Time Clock.

5.11. WiFi / BT Support

The ESP32-P4 display module itself does not support WiFi or BT, however with the aid of a 4D Systems ESP32-C6 based Add-On module, WiFi and BT are available. The module connects directly to the 30-pin male pin header on the back of the ESP32-P4 display modules, and communicates to the ESP32-C6-MINI-1U WiFi module using the 4bit SDIO bus, achieving the maximum speeds possible with this combination pair.



ESP32-P4-C6-WiFi module

This module is **NOT INCLUDED** with the base ESP32-P4 display module, but is available as an Add-On / Accessory and can be purchased separately, or possibly in a Kit. Please refer to the 4D Systems website.

For more information, please refer to the ESP32-P4-C6-WiFi module datasheet, available on the Resource Centre on the 4D Systems website.

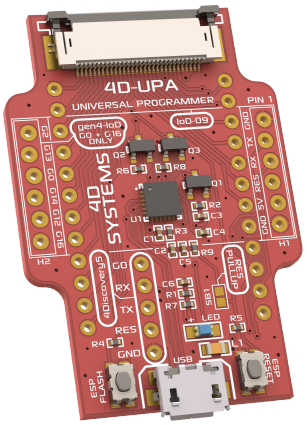
6. Display/Module Precautions

- Avoid having to display the same image/object on the screen for lengthy periods. This can cause a burn-in which is a common problem with all types of display technologies. Blank the screen after a while or dim it very low by adjusting the contrast. Better still; implement a screen saver feature.
- Moisture and water can damage the display. Moisture on the surface of a powered display should not cause any problems, however, if water is to enter the display either from the front or from the rear, or come in contact with the PCB, it will damage. Wipe off any moisture gently or let the display dry before usage. If using this display module in an environment where it can get wet, ensure an appropriate enclosure is used.
- Dirt from fingerprint oil and fat can easily stain the surface of the display. Gently wipe off any stains with a soft lint-free cloth.
- The performance of the display will degrade under high temperatures and humidity. Avoid such conditions when storing.
- Do not tamper with the display flex cable that is connected to the control board. This may affect the connection between the display and the driving circuitry and cause failure.
- Displays are susceptible to mechanical shock and any force exerted on the module may result in deformed zebra stripes, a cracked display cell and a broken backlight.
- Always use the mounting holes on the module's to mount the display where possible, or mount using the CLB for CLB based modules (either with the tape supplied or with suitable sealant - please contact sales for a custom order without tape).
- Display modules have a finite life, which is typically dictated by the display itself, more specifically the backlight. The backlight contains LEDs, which fade over time. In the Specifications section is a figure for the typical life of the display, and the criteria are listed.

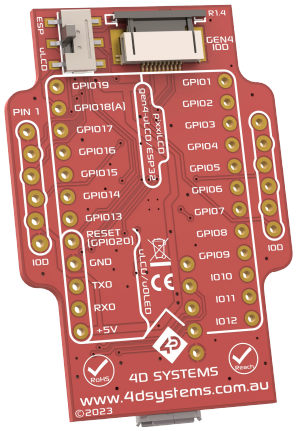
7. Hardware Tools

The following hardware tools are recommended for full control of the ESP32-P4 Display Modules.

7.1. 4D-UPA



4D-UPA Front

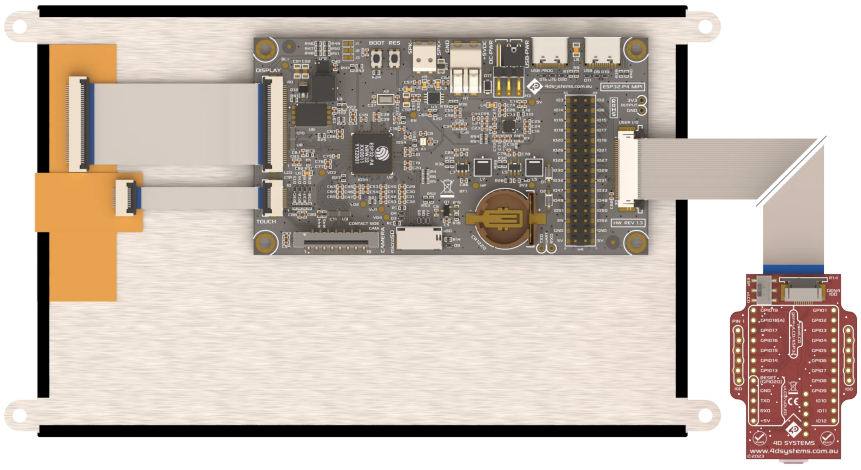


4D-UPA Back

The 4D-UPA minimizes the connections and modules required for programming the ESP32-P4 modules via its UART. The 4D-UPA utilises a micro USB interface, and has DIP style pads for GPIO breakout of all the signals used on the ESP32-P4's 30-way FFC interface, which is useful for development or final product use.

The GPIO naming convention on the 4D-UPA does not reflect the GPIO naming of the actual display module, due to the 4D-UPA being universal and able to be used with many 4D Products. Please review the latest 4D-UPA datasheet for information on mapping the GPIO naming from this module, with the GPIO naming on the 4D-UPA, to ensure you connect to the correct pins you desire.

Typically, a 4D-UPA should not be required for programming the ESP32-P4 series of modules, as they have USB-C on board, however if problems occur or situations arise that the USB-C is non-functional or 4D-UPA programming is more desired, then the 4D-UPA is a good tool to have on hand.



The 4D-UPA is connected to the ESP32-P4 module using the supplied 30-way FFC Cable. The connectors on both the ESP32-P4 module, and the 4D-UPA, are Top-Contact, meaning the FFC cable pins should be facing upwards, and the blue stiffener should be facing down towards the LCD.

Note

1. 4D-UPA REV 1.4 and higher is required to program a ESP32 processor. Anything < REV 1.4 will not work for programming ESP32 based products. Please be sure to refer to the 4D-UPA Datasheet.
2. If using the 4D-UPA, only the supplied FFC cable (or same type) can be used. The type of cable supplied, as described in the FFC Cable section, is an Opposite type (contacts on opposite sides to each other at end end). If a straight cable (contacts on the same side at both ends) is sourced, this will NOT work when connecting to the 4D-UPA, as the connections will be swapped. Please refer to the information provided for more detail.

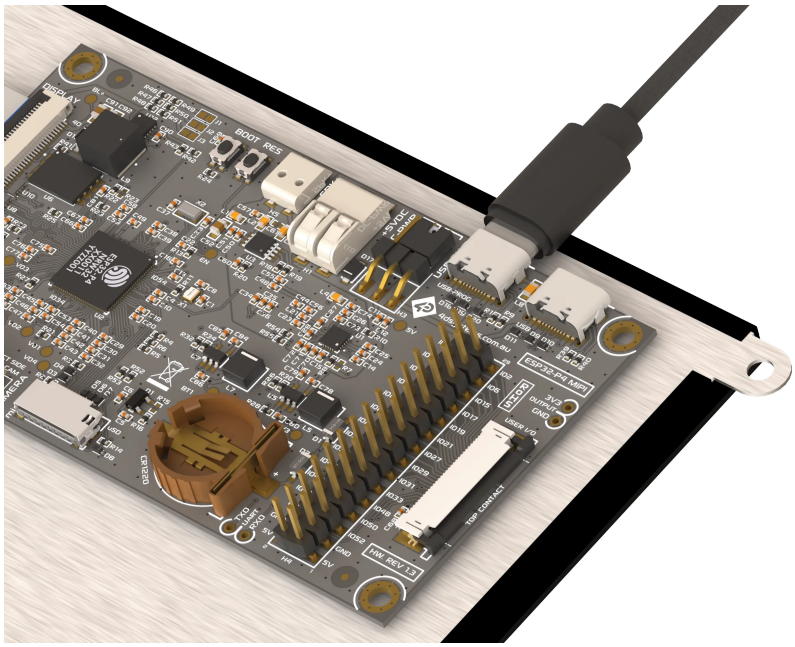
7.2. USB-C Cable

A USB-C Cable is the primary way to program a ESP32-P4 module from 4D Systems, aside from using a 4D-UPA, as described previously.

A USB-C cable is not supplied with the modules, as they can be sourced from any computer or hardware store, and come with most Cell Phones these days too. Be sure to use a cable that offers both Power and Data, as some (rare) cables are Power only.

Connection of the USB-C cable to the module is simple, and simply plugs into the USB-C "USB-PROG" connector on the board. There is a second USB-C connector which is primarily for OTG applications, which is simply labelled as "USB".

The USB-C provides power as well as USB Data communications, for while developing software on the module and programming it. The USB-C cable can be used in the end product use if desired, or the use of the FFC-Cable directly to the projects main Application PCB as an alternative. Similarly the 30-way male pin headers offer power input also, as well as the DC wire terminal - so there are many options for end use powering of these modules.



USB-C Connection into USB-PROG connector

8. Software Tools

8.1. Workshop4 IDE

Workshop4 is a comprehensive software IDE that provides an integrated software development platform for all the 4D family of processors and modules, as well as some 3rd party processors such as the Espressif ESP32.

The IDE provides an Editor and WYSIWYG design area for ESP32-P4 based modules, to develop complete application code with various widgets and media references as required. All user application code is developed within the Workshop4 IDE, and is easily coupled with graphics and media, so is a one stop shop for development with these modules.

The Workshop4 IDE utilises the Arduino IDE 2.x CLI to handle the compiling, linking and downloading of ESP32 based projects, using the ESP32 Arduino Core and associated libraries, without having to interface with the Arduino IDE at all.

When using Workshop4, the library **GFX4dESP32P4** is used. This library maintains compatibility with older Espressif chip based products such as the gen4-ESP32 (ESP32-S3) series and gen4-IoD (ESP8266) series, through the libraries **GFX4dESP32** and **GFX4d**. This means that older projects can be easily converted to this product line, with minimal to no modifications, as long as the hardware supports all functionality.

> **Note**
>
> Arduino IDE 2.x and Arduino CLI are not included when installing Workshop4 and must be installed separately.
> For complete development setup instructions, please refer to the Workshop4 ESP32 Development Manual.

8.2. Workshop5 IDE

Workshop5 is the new IDE from 4D Systems, which will eventually take over Workshop4 when it is fully developed. It is still in development, however will be actively used for new applications for many of the 4D Systems modules, including the ESP32-P4 modules. It is a comprehensive software IDE that provides an integrated software development platform for all the 4D family of processors and modules, as well as some 3rd party processors such as the Espressif ESP32, and the Raspberry Pi RP2350 modules. Support for the ESP32-P4 inside Workshop5 is expected inside Q2 2026 - until then, please use Workshop4.

The IDE provides an Editor and WYSIWYG design area for ESP32-P4 based modules, to develop complete application code with various widgets and media references as required. All user application code is developed within the Workshop5 IDE, and is easily coupled with graphics and media, so is a one stop shop for development with these modules.

The Workshop5 IDE utilises the Arduino CLI to handle the compiling, linking and downloading of ESP32 based projects, using the Arduino-ESP32 Core and associated libraries, without having to interface with the Arduino IDE at all.

When using Workshop5, the library **Graphics4D** is used. This library maintains compatibility with other Workshop5 compatible products that are programed in C++, such as the 4D Systems' RP2350B products. This means that C++ graphics code used in Workshop5 project can be easily used to this product line, with minimal to no modifications.

Note

1. Arduino CLI is included when installing Workshop5, so there is no need to install it separately.
2. Workshop5 compatibility with the ESP32-P4 range is still **Coming Soon**, and should be available sometime in Q2 of 2026. For the time being, please use the Workshop4 IDE which is fully featured.

8.3. Workshop Built-in Tools

Built into Workshop4 and Workshop5 are a number of tools which are available to aid the programming of the ESP32-P4 series of displays.

Terminal, as the name implies, is a terminal application that can be used to communicate with the display module and is primarily used for basic debugging. It displays incoming Serial messages from the display module in ASCII and HEX format. It is capable of sending character or hex strings as well as keystrokes to the display.

RMPET is a partitioning and formatting tool, used to correctly set up a micro-SD card for use with 4D Systems products. This is further discussed in the SD/SDHC Memory Cards section

9. Programming Language

The programming language used in the Workshop4 and Workshop5 IDEs to program the ESP32-P4 series of modules, is C++, which is the same as native Arduino IDE written code.

The Arduino programming language is a user-friendly coding system tailored for Arduino microcontrollers. It simplifies microcontroller coding, bridging the gap between users and hardware. Its approachability and community support make it ideal for various projects.

10. ESP-IDF Development

While it's recommended to use Workshop4 or Workshop5 IDEs to create applications for these displays, having ESP32-P4 at its core allows any user to develop applications directly through the official development framework from Espressif.

When using ESP-IDF, users can utilise search and utilize various IDF components by searching through the ESP Component Registry. These serves as libraries simplifying development similar to Arduino. However, these are mostly written in C rather than C++.

To aid with setting up our ESP32-P4 displays for ESP-IDF development, here are some recommended components for this product line:

Peripheral	Component
LCD	espressif/esp_lcd_jd9365 *Refer to section LCD Initialization Codes for the initialization code for each size*
Touch	espressif/esp_lcd_touch_gt911
I2S (Audio)	espressif/esp_codec_dev

10.1. LCD Initialization Codes

All LCDs in this ESP32-P4 series utilizes JD9365 for the LCD. Each size require a particular init code which is as shown below:

7-inch

```
const jd9365_lcd_init_cmd_t esp32p4_4d_init_cmds[] = {
    {0xE0, (uint8_t[]){0x00}, 1, 0},
    {0xE1, (uint8_t[]){0x93}, 1, 0},
    {0xE2, (uint8_t[]){0x65}, 1, 0},
    {0xE3, (uint8_t[]){0xF8}, 1, 0},
    {0x80, (uint8_t[]){0x01}, 1, 0},
    {0xE0, (uint8_t[]){0x01}, 1, 0},
    {0x00, (uint8_t[]){0x00}, 1, 0},
    {0x01, (uint8_t[]){0x3D}, 1, 0},
    {0x03, (uint8_t[]){0x10}, 1, 0},
    {0x04, (uint8_t[]){0x43}, 1, 0},
    {0x0C, (uint8_t[]){0x74}, 1, 0},
    {0x17, (uint8_t[]){0x00}, 1, 0},
    {0x18, (uint8_t[]){0xEF}, 1, 0},
    {0x19, (uint8_t[]){0x01}, 1, 0},
    {0x1A, (uint8_t[]){0x00}, 1, 0},
    {0x1B, (uint8_t[]){0xEF}, 1, 0},
    {0x1C, (uint8_t[]){0x01}, 1, 0},
    {0x24, (uint8_t[]){0xFE}, 1, 0},
    {0x37, (uint8_t[]){0x09}, 1, 0},
    {0x38, (uint8_t[]){0x04}, 1, 0},
    {0x39, (uint8_t[]){0x08}, 1, 0},
    {0x3A, (uint8_t[]){0x12}, 1, 0},
    {0x3C, (uint8_t[]){0x78}, 1, 0},
    {0x3D, (uint8_t[]){0xFF}, 1, 0},
    {0x3E, (uint8_t[]){0xFF}, 1, 0},
    {0x3F, (uint8_t[]){0x7F}, 1, 0},
    {0x40, (uint8_t[]){0x06}, 1, 0},
    {0x41, (uint8_t[]){0xA0}, 1, 0},
    {0x43, (uint8_t[]){0x14}, 1, 0},
    {0x44, (uint8_t[]){0x11}, 1, 0},
    {0x45, (uint8_t[]){0x24}, 1, 0},
    {0x55, (uint8_t[]){0x02}, 1, 0},
    {0x57, (uint8_t[]){0x69}, 1, 0},
    {0x59, (uint8_t[]){0x0A}, 1, 0},
    {0x5A, (uint8_t[]){0x29}, 1, 0},
    {0x5B, (uint8_t[]){0x10}, 1, 0},
    {0x5D, (uint8_t[]){0x70}, 1, 0},
    {0x5E, (uint8_t[]){0x5B}, 1, 0},
    {0x5F, (uint8_t[]){0x4B}, 1, 0},
    {0x60, (uint8_t[]){0x3F}, 1, 0},
    {0x61, (uint8_t[]){0x3B}, 1, 0},
    {0x62, (uint8_t[]){0x2D}, 1, 0},
    {0x63, (uint8_t[]){0x2F}, 1, 0},
    {0x64, (uint8_t[]){0x18}, 1, 0},
    {0x65, (uint8_t[]){0x2F}, 1, 0},
    {0x66, (uint8_t[]){0x2C}, 1, 0},
```

```
    {0x67, (uint8_t[]){0x2B}, 1, 0},
    {0x68, (uint8_t[]){0x47}, 1, 0},
    {0x69, (uint8_t[]){0x34}, 1, 0},
    {0x6A, (uint8_t[]){0x3B}, 1, 0},
    {0x6B, (uint8_t[]){0x2E}, 1, 0},
    {0x6C, (uint8_t[]){0x2A}, 1, 0},
    {0x6D, (uint8_t[]){0x1F}, 1, 0},
    {0x6E, (uint8_t[]){0x11}, 1, 0},
    {0x6F, (uint8_t[]){0x02}, 1, 0},
    {0x70, (uint8_t[]){0x70}, 1, 0},
    {0x71, (uint8_t[]){0x5B}, 1, 0},
    {0x72, (uint8_t[]){0x4B}, 1, 0},
    {0x73, (uint8_t[]){0x3F}, 1, 0},
    {0x74, (uint8_t[]){0x3B}, 1, 0},
    {0x75, (uint8_t[]){0x2D}, 1, 0},
    {0x76, (uint8_t[]){0x2F}, 1, 0},
    {0x77, (uint8_t[]){0x18}, 1, 0},
    {0x78, (uint8_t[]){0x2F}, 1, 0},
    {0x79, (uint8_t[]){0x2C}, 1, 0},
    {0x7A, (uint8_t[]){0x2B}, 1, 0},
    {0x7B, (uint8_t[]){0x47}, 1, 0},
    {0x7C, (uint8_t[]){0x34}, 1, 0},
    {0x7D, (uint8_t[]){0x3B}, 1, 0},
    {0x7E, (uint8_t[]){0x2E}, 1, 0},
    {0x7F, (uint8_t[]){0x2A}, 1, 0},
    {0x80, (uint8_t[]){0x1F}, 1, 0},
    {0x81, (uint8_t[]){0x11}, 1, 0},
    {0x82, (uint8_t[]){0x02}, 1, 0},
    {0xE0, (uint8_t[]){0x02}, 1, 0},
    {0x00, (uint8_t[]){0x5E}, 1, 0},
    {0x01, (uint8_t[]){0x5F}, 1, 0},
    {0x02, (uint8_t[]){0x57}, 1, 0},
    {0x03, (uint8_t[]){0x58}, 1, 0},
    {0x04, (uint8_t[]){0x48}, 1, 0},
    {0x05, (uint8_t[]){0x4A}, 1, 0},
    {0x06, (uint8_t[]){0x44}, 1, 0},
    {0x07, (uint8_t[]){0x46}, 1, 0},
    {0x08, (uint8_t[]){0x40}, 1, 0},
    {0x09, (uint8_t[]){0x5F}, 1, 0},
    {0x0A, (uint8_t[]){0x5F}, 1, 0},
    {0x0B, (uint8_t[]){0x5F}, 1, 0},
    {0x0C, (uint8_t[]){0x5F}, 1, 0},
    {0x0D, (uint8_t[]){0x5F}, 1, 0},
    {0x0E, (uint8_t[]){0x5F}, 1, 0},
    {0x0F, (uint8_t[]){0x42}, 1, 0},
    {0x10, (uint8_t[]){0x5F}, 1, 0},
    {0x11, (uint8_t[]){0x5F}, 1, 0},
    {0x12, (uint8_t[]){0x5F}, 1, 0},
    {0x13, (uint8_t[]){0x5F}, 1, 0},
    {0x14, (uint8_t[]){0x5F}, 1, 0},
    {0x15, (uint8_t[]){0x5F}, 1, 0},
    {0x16, (uint8_t[]){0x5E}, 1, 0},
    {0x17, (uint8_t[]){0x5F}, 1, 0},
    {0x18, (uint8_t[]){0x57}, 1, 0},
    {0x19, (uint8_t[]){0x58}, 1, 0},
```

```
    {0x1A, (uint8_t[]){0x49}, 1, 0},
    {0x1B, (uint8_t[]){0x4B}, 1, 0},
    {0x1C, (uint8_t[]){0x45}, 1, 0},
    {0x1D, (uint8_t[]){0x47}, 1, 0},
    {0x1E, (uint8_t[]){0x41}, 1, 0},
    {0x1F, (uint8_t[]){0x5F}, 1, 0},
    {0x20, (uint8_t[]){0x5F}, 1, 0},
    {0x21, (uint8_t[]){0x5F}, 1, 0},
    {0x22, (uint8_t[]){0x5F}, 1, 0},
    {0x23, (uint8_t[]){0x5F}, 1, 0},
    {0x24, (uint8_t[]){0x5F}, 1, 0},
    {0x25, (uint8_t[]){0x43}, 1, 0},
    {0x26, (uint8_t[]){0x5F}, 1, 0},
    {0x27, (uint8_t[]){0x5F}, 1, 0},
    {0x28, (uint8_t[]){0x5F}, 1, 0},
    {0x29, (uint8_t[]){0x5F}, 1, 0},
    {0x2A, (uint8_t[]){0x5F}, 1, 0},
    {0x2B, (uint8_t[]){0x5F}, 1, 0},
    {0x2C, (uint8_t[]){0x1F}, 1, 0},
    {0x2D, (uint8_t[]){0x1E}, 1, 0},
    {0x2E, (uint8_t[]){0x17}, 1, 0},
    {0x2F, (uint8_t[]){0x18}, 1, 0},
    {0x30, (uint8_t[]){0x07}, 1, 0},
    {0x31, (uint8_t[]){0x05}, 1, 0},
    {0x32, (uint8_t[]){0x0B}, 1, 0},
    {0x33, (uint8_t[]){0x09}, 1, 0},
    {0x34, (uint8_t[]){0x03}, 1, 0},
    {0x35, (uint8_t[]){0x1F}, 1, 0},
    {0x36, (uint8_t[]){0x1F}, 1, 0},
    {0x37, (uint8_t[]){0x1F}, 1, 0},
    {0x38, (uint8_t[]){0x1F}, 1, 0},
    {0x39, (uint8_t[]){0x1F}, 1, 0},
    {0x3A, (uint8_t[]){0x1F}, 1, 0},
    {0x3B, (uint8_t[]){0x01}, 1, 0},
    {0x3C, (uint8_t[]){0x1F}, 1, 0},
    {0x3D, (uint8_t[]){0x1F}, 1, 0},
    {0x3E, (uint8_t[]){0x1F}, 1, 0},
    {0x3F, (uint8_t[]){0x1F}, 1, 0},
    {0x40, (uint8_t[]){0x1F}, 1, 0},
    {0x41, (uint8_t[]){0x1F}, 1, 0},
    {0x42, (uint8_t[]){0x1F}, 1, 0},
    {0x43, (uint8_t[]){0x1E}, 1, 0},
    {0x44, (uint8_t[]){0x17}, 1, 0},
    {0x45, (uint8_t[]){0x18}, 1, 0},
    {0x46, (uint8_t[]){0x06}, 1, 0},
    {0x47, (uint8_t[]){0x04}, 1, 0},
    {0x48, (uint8_t[]){0x0A}, 1, 0},
    {0x49, (uint8_t[]){0x08}, 1, 0},
    {0x4A, (uint8_t[]){0x02}, 1, 0},
    {0x4B, (uint8_t[]){0x1F}, 1, 0},
    {0x4C, (uint8_t[]){0x1F}, 1, 0},
    {0x4D, (uint8_t[]){0x1F}, 1, 0},
    {0x4E, (uint8_t[]){0x1F}, 1, 0},
    {0x4F, (uint8_t[]){0x1F}, 1, 0},
    {0x50, (uint8_t[]){0x1F}, 1, 0},
```

```
    {0x51, (uint8_t[]){0x00}, 1, 0},
    {0x52, (uint8_t[]){0x1F}, 1, 0},
    {0x53, (uint8_t[]){0x1F}, 1, 0},
    {0x54, (uint8_t[]){0x1F}, 1, 0},
    {0x55, (uint8_t[]){0x1F}, 1, 0},
    {0x56, (uint8_t[]){0x1F}, 1, 0},
    {0x57, (uint8_t[]){0x1F}, 1, 0},
    {0x58, (uint8_t[]){0x40}, 1, 0},
    {0x5B, (uint8_t[]){0x30}, 1, 0},
    {0x5C, (uint8_t[]){0x07}, 1, 0},
    {0x5D, (uint8_t[]){0x30}, 1, 0},
    {0x5E, (uint8_t[]){0x01}, 1, 0},
    {0x5F, (uint8_t[]){0x02}, 1, 0},
    {0x60, (uint8_t[]){0x30}, 1, 0},
    {0x61, (uint8_t[]){0x03}, 1, 0},
    {0x62, (uint8_t[]){0x04}, 1, 0},
    {0x63, (uint8_t[]){0x6A}, 1, 0},
    {0x64, (uint8_t[]){0x6A}, 1, 0},
    {0x65, (uint8_t[]){0x35}, 1, 0},
    {0x66, (uint8_t[]){0x0D}, 1, 0},
    {0x67, (uint8_t[]){0x73}, 1, 0},
    {0x68, (uint8_t[]){0x0B}, 1, 0},
    {0x69, (uint8_t[]){0x6A}, 1, 0},
    {0x6A, (uint8_t[]){0x6A}, 1, 0},
    {0x6B, (uint8_t[]){0x08}, 1, 0},
    {0x6C, (uint8_t[]){0x00}, 1, 0},
    {0x6D, (uint8_t[]){0x04}, 1, 0},
    {0x6E, (uint8_t[]){0x00}, 1, 0},
    {0x6F, (uint8_t[]){0x88}, 1, 0},
    {0x75, (uint8_t[]){0xBC}, 1, 0},
    {0x76, (uint8_t[]){0x00}, 1, 0},
    {0x77, (uint8_t[]){0x0D}, 1, 0},
    {0x78, (uint8_t[]){0x1B}, 1, 0},
    {0xE0, (uint8_t[]){0x04}, 1, 0},
    {0x00, (uint8_t[]){0x0E}, 1, 0},
    {0x02, (uint8_t[]){0xB3}, 1, 0},
    {0x09, (uint8_t[]){0x60}, 1, 0},
    {0x0E, (uint8_t[]){0x4A}, 1, 0},
    {0x37, (uint8_t[]){0x58}, 1, 0},
    {0x2B, (uint8_t[]){0x0F}, 1, 0},
    {0xE0, (uint8_t[]){0x05}, 1, 0},
    {0x15, (uint8_t[]){0x1D}, 1, 0},
    {0xE0, (uint8_t[]){0x00}, 1, 0},
    {0x11, (uint8_t[]){0x00}, 1, 120},
    {0x29, (uint8_t[]){0x00}, 1, 5},
    {0x35, (uint8_t[]){0x00}, 1, 0},
};
```

8-inch

```
const jd9365_lcd_init_cmd_t esp32p4_4d_init_cmds[] = {
    {0xE0, (uint8_t[]){0x00}, 1, 0},
    {0xE1, (uint8_t[]){0x93}, 1, 0},
    {0xE2, (uint8_t[]){0x65}, 1, 0},
    {0xE3, (uint8_t[]){0xF8}, 1, 0},
    {0x80, (uint8_t[]){0x01}, 1, 0},
    {0xE0, (uint8_t[]){0x01}, 1, 0},
    {0x00, (uint8_t[]){0x00}, 1, 0},
    {0x01, (uint8_t[]){0x37}, 1, 0},
    {0x17, (uint8_t[]){0x00}, 1, 0},
    {0x18, (uint8_t[]){0xF7}, 1, 0},
    {0x19, (uint8_t[]){0x01}, 1, 0},
    {0x1A, (uint8_t[]){0x00}, 1, 0},
    {0x1B, (uint8_t[]){0xF7}, 1, 0},
    {0x1C, (uint8_t[]){0x01}, 1, 0},
    {0x35, (uint8_t[]){0x23}, 1, 0},
    {0x37, (uint8_t[]){0x09}, 1, 0},
    {0x38, (uint8_t[]){0x04}, 1, 0},
    {0x39, (uint8_t[]){0x00}, 1, 0},
    {0x3A, (uint8_t[]){0x01}, 1, 0},
    {0x3C, (uint8_t[]){0x70}, 1, 0},
    {0x3E, (uint8_t[]){0xFF}, 1, 0},
    {0x3F, (uint8_t[]){0x7F}, 1, 0},
    {0x40, (uint8_t[]){0x06}, 1, 0},
    {0x41, (uint8_t[]){0xA0}, 1, 0},
    {0x43, (uint8_t[]){0x1E}, 1, 0},
    {0x44, (uint8_t[]){0x0B}, 1, 0},
    {0x45, (uint8_t[]){0x28}, 1, 0},
    {0x55, (uint8_t[]){0x02}, 1, 0},
    {0x57, (uint8_t[]){0xA9}, 1, 0},
    {0x59, (uint8_t[]){0x0A}, 1, 0},
    {0x5A, (uint8_t[]){0x2D}, 1, 0},
    {0x5B, (uint8_t[]){0x1A}, 1, 0},
    {0x5C, (uint8_t[]){0x15}, 1, 0},
    {0x5D, (uint8_t[]){0x7F}, 1, 0},
    {0x5E, (uint8_t[]){0x6F}, 1, 0},
    {0x5F, (uint8_t[]){0x62}, 1, 0},
    {0x60, (uint8_t[]){0x57}, 1, 0},
    {0x61, (uint8_t[]){0x54}, 1, 0},
    {0x62, (uint8_t[]){0x46}, 1, 0},
    {0x63, (uint8_t[]){0x4B}, 1, 0},
    {0x64, (uint8_t[]){0x35}, 1, 0},
    {0x65, (uint8_t[]){0x4E}, 1, 0},
    {0x66, (uint8_t[]){0x4C}, 1, 0},
    {0x67, (uint8_t[]){0x4A}, 1, 0},
    {0x68, (uint8_t[]){0x65}, 1, 0},
    {0x69, (uint8_t[]){0x4E}, 1, 0},
    {0x6A, (uint8_t[]){0x4C}, 1, 0},
    {0x6B, (uint8_t[]){0x36}, 1, 0},
    {0x6C, (uint8_t[]){0x31}, 1, 0},
    {0x6D, (uint8_t[]){0x24}, 1, 0},
```

```
    {0x6E, (uint8_t[]){0x12}, 1, 0},
    {0x6F, (uint8_t[]){0x02}, 1, 0},
    {0x70, (uint8_t[]){0x7F}, 1, 0},
    {0x71, (uint8_t[]){0x6F}, 1, 0},
    {0x72, (uint8_t[]){0x62}, 1, 0},
    {0x73, (uint8_t[]){0x57}, 1, 0},
    {0x74, (uint8_t[]){0x54}, 1, 0},
    {0x75, (uint8_t[]){0x46}, 1, 0},
    {0x76, (uint8_t[]){0x4B}, 1, 0},
    {0x77, (uint8_t[]){0x35}, 1, 0},
    {0x78, (uint8_t[]){0x4E}, 1, 0},
    {0x79, (uint8_t[]){0x4C}, 1, 0},
    {0x7A, (uint8_t[]){0x4A}, 1, 0},
    {0x7B, (uint8_t[]){0x65}, 1, 0},
    {0x7C, (uint8_t[]){0x4E}, 1, 0},
    {0x7D, (uint8_t[]){0x4C}, 1, 0},
    {0x7E, (uint8_t[]){0x36}, 1, 0},
    {0x7F, (uint8_t[]){0x31}, 1, 0},
    {0x80, (uint8_t[]){0x24}, 1, 0},
    {0x81, (uint8_t[]){0x12}, 1, 0},
    {0x82, (uint8_t[]){0x02}, 1, 0},
    {0xE0, (uint8_t[]){0x02}, 1, 0},
    {0x00, (uint8_t[]){0x50}, 1, 0},
    {0x01, (uint8_t[]){0x55}, 1, 0},
    {0x02, (uint8_t[]){0x55}, 1, 0},
    {0x03, (uint8_t[]){0x52}, 1, 0},
    {0x04, (uint8_t[]){0x77}, 1, 0},
    {0x05, (uint8_t[]){0x57}, 1, 0},
    {0x06, (uint8_t[]){0x55}, 1, 0},
    {0x07, (uint8_t[]){0x4E}, 1, 0},
    {0x08, (uint8_t[]){0x4C}, 1, 0},
    {0x09, (uint8_t[]){0x55}, 1, 0},
    {0x0A, (uint8_t[]){0x4A}, 1, 0},
    {0x0B, (uint8_t[]){0x48}, 1, 0},
    {0x0C, (uint8_t[]){0x55}, 1, 0},
    {0x0D, (uint8_t[]){0x46}, 1, 0},
    {0x0E, (uint8_t[]){0x44}, 1, 0},
    {0x0F, (uint8_t[]){0x40}, 1, 0},
    {0x10, (uint8_t[]){0x55}, 1, 0},
    {0x11, (uint8_t[]){0x55}, 1, 0},
    {0x12, (uint8_t[]){0x55}, 1, 0},
    {0x13, (uint8_t[]){0x55}, 1, 0},
    {0x14, (uint8_t[]){0x55}, 1, 0},
    {0x15, (uint8_t[]){0x55}, 1, 0},
    {0x16, (uint8_t[]){0x51}, 1, 0},
    {0x17, (uint8_t[]){0x55}, 1, 0},
    {0x18, (uint8_t[]){0x55}, 1, 0},
    {0x19, (uint8_t[]){0x53}, 1, 0},
    {0x1A, (uint8_t[]){0x77}, 1, 0},
    {0x1B, (uint8_t[]){0x57}, 1, 0},
    {0x1C, (uint8_t[]){0x55}, 1, 0},
    {0x1D, (uint8_t[]){0x4F}, 1, 0},
    {0x1E, (uint8_t[]){0x4D}, 1, 0},
    {0x1F, (uint8_t[]){0x55}, 1, 0},
    {0x20, (uint8_t[]){0x4B}, 1, 0},
```

```
    {0x21, (uint8_t[]){0x49}, 1, 0},
    {0x22, (uint8_t[]){0x55}, 1, 0},
    {0x23, (uint8_t[]){0x47}, 1, 0},
    {0x24, (uint8_t[]){0x45}, 1, 0},
    {0x25, (uint8_t[]){0x41}, 1, 0},
    {0x26, (uint8_t[]){0x55}, 1, 0},
    {0x27, (uint8_t[]){0x55}, 1, 0},
    {0x28, (uint8_t[]){0x55}, 1, 0},
    {0x29, (uint8_t[]){0x55}, 1, 0},
    {0x2A, (uint8_t[]){0x55}, 1, 0},
    {0x2B, (uint8_t[]){0x55}, 1, 0},
    {0x2C, (uint8_t[]){0x01}, 1, 0},
    {0x2D, (uint8_t[]){0x15}, 1, 0},
    {0x2E, (uint8_t[]){0x15}, 1, 0},
    {0x2F, (uint8_t[]){0x13}, 1, 0},
    {0x30, (uint8_t[]){0x17}, 1, 0},
    {0x31, (uint8_t[]){0x17}, 1, 0},
    {0x32, (uint8_t[]){0x15}, 1, 0},
    {0x33, (uint8_t[]){0x0D}, 1, 0},
    {0x34, (uint8_t[]){0x0F}, 1, 0},
    {0x35, (uint8_t[]){0x15}, 1, 0},
    {0x36, (uint8_t[]){0x05}, 1, 0},
    {0x37, (uint8_t[]){0x07}, 1, 0},
    {0x38, (uint8_t[]){0x15}, 1, 0},
    {0x39, (uint8_t[]){0x09}, 1, 0},
    {0x3A, (uint8_t[]){0x0B}, 1, 0},
    {0x3B, (uint8_t[]){0x11}, 1, 0},
    {0x3C, (uint8_t[]){0x15}, 1, 0},
    {0x3D, (uint8_t[]){0x15}, 1, 0},
    {0x3E, (uint8_t[]){0x15}, 1, 0},
    {0x3F, (uint8_t[]){0x15}, 1, 0},
    {0x40, (uint8_t[]){0x15}, 1, 0},
    {0x41, (uint8_t[]){0x15}, 1, 0},
    {0x42, (uint8_t[]){0x00}, 1, 0},
    {0x43, (uint8_t[]){0x15}, 1, 0},
    {0x44, (uint8_t[]){0x15}, 1, 0},
    {0x45, (uint8_t[]){0x12}, 1, 0},
    {0x46, (uint8_t[]){0x17}, 1, 0},
    {0x47, (uint8_t[]){0x17}, 1, 0},
    {0x48, (uint8_t[]){0x15}, 1, 0},
    {0x49, (uint8_t[]){0x0C}, 1, 0},
    {0x4A, (uint8_t[]){0x0E}, 1, 0},
    {0x4B, (uint8_t[]){0x15}, 1, 0},
    {0x4C, (uint8_t[]){0x04}, 1, 0},
    {0x4D, (uint8_t[]){0x06}, 1, 0},
    {0x4E, (uint8_t[]){0x15}, 1, 0},
    {0x4F, (uint8_t[]){0x08}, 1, 0},
    {0x50, (uint8_t[]){0x0A}, 1, 0},
    {0x51, (uint8_t[]){0x10}, 1, 0},
    {0x52, (uint8_t[]){0x15}, 1, 0},
    {0x53, (uint8_t[]){0x15}, 1, 0},
    {0x54, (uint8_t[]){0x15}, 1, 0},
    {0x55, (uint8_t[]){0x15}, 1, 0},
    {0x56, (uint8_t[]){0x15}, 1, 0},
    {0x57, (uint8_t[]){0x15}, 1, 0},
```

```
    {0x58, (uint8_t[]){0x40}, 1, 0},
    {0x5B, (uint8_t[]){0x10}, 1, 0},
    {0x5C, (uint8_t[]){0x06}, 1, 0},
    {0x5D, (uint8_t[]){0x40}, 1, 0},
    {0x5E, (uint8_t[]){0x00}, 1, 0},
    {0x5F, (uint8_t[]){0x00}, 1, 0},
    {0x60, (uint8_t[]){0x40}, 1, 0},
    {0x61, (uint8_t[]){0x03}, 1, 0},
    {0x62, (uint8_t[]){0x04}, 1, 0},
    {0x63, (uint8_t[]){0x6C}, 1, 0},
    {0x64, (uint8_t[]){0x6C}, 1, 0},
    {0x65, (uint8_t[]){0x75}, 1, 0},
    {0x66, (uint8_t[]){0x08}, 1, 0},
    {0x67, (uint8_t[]){0xB4}, 1, 0},
    {0x68, (uint8_t[]){0x08}, 1, 0},
    {0x69, (uint8_t[]){0x6C}, 1, 0},
    {0x6A, (uint8_t[]){0x6C}, 1, 0},
    {0x6B, (uint8_t[]){0x0C}, 1, 0},
    {0x6D, (uint8_t[]){0x00}, 1, 0},
    {0x6E, (uint8_t[]){0x00}, 1, 0},
    {0x6F, (uint8_t[]){0x88}, 1, 0},
    {0x75, (uint8_t[]){0xBB}, 1, 0},
    {0x76, (uint8_t[]){0x00}, 1, 0},
    {0x77, (uint8_t[]){0x05}, 1, 0},
    {0x78, (uint8_t[]){0x2A}, 1, 0},
    {0xE0, (uint8_t[]){0x04}, 1, 0},
    {0x09, (uint8_t[]){0x11}, 1, 0},
    {0x0E, (uint8_t[]){0x48}, 1, 0},
    {0xE0, (uint8_t[]){0x00}, 1, 0},
    {0x11, (uint8_t[]){0x00}, 1,120},
    {0x29, (uint8_t[]){0x00}, 1, 20},
};
```

9-inch

```
const jd9365_lcd_init_cmd_t esp32p4_4d_init_cmds[] = {
    {0xE0, (uint8_t []){0x00}, 1, 0},
    {0xE1, (uint8_t []){0x93}, 1, 0},
    {0xE2, (uint8_t []){0x65}, 1, 0},
    {0xE3, (uint8_t []){0xF8}, 1, 0},
    {0xE0, (uint8_t []){0x01}, 1, 0},
    {0x00, (uint8_t []){0x00}, 1, 0},
    {0x01, (uint8_t []){0x4E}, 1, 0},
    {0x03, (uint8_t []){0x00}, 1, 0},
    {0x04, (uint8_t []){0x65}, 1, 0},
    {0x0C, (uint8_t []){0x74}, 1, 0},
    {0x17, (uint8_t []){0x00}, 1, 0},
    {0x18, (uint8_t []){0xB7}, 1, 0},
    {0x19, (uint8_t []){0x00}, 1, 0},
    {0x1A, (uint8_t []){0x00}, 1, 0},
    {0x1B, (uint8_t []){0xB7}, 1, 0},
    {0x1C, (uint8_t []){0x00}, 1, 0},
    {0x24, (uint8_t []){0xFE}, 1, 0},
    {0x37, (uint8_t []){0x19}, 1, 0},
    {0x38, (uint8_t []){0x05}, 1, 0},
    {0x39, (uint8_t []){0x00}, 1, 0},
    {0x3A, (uint8_t []){0x01}, 1, 0},
    {0x3B, (uint8_t []){0x01}, 1, 0},
    {0x3C, (uint8_t []){0x70}, 1, 0},
    {0x3D, (uint8_t []){0xFF}, 1, 0},
    {0x3E, (uint8_t []){0xFF}, 1, 0},
    {0x3F, (uint8_t []){0xFF}, 1, 0},
    {0x40, (uint8_t []){0x06}, 1, 0},
    {0x41, (uint8_t []){0xA0}, 1, 0},
    {0x43, (uint8_t []){0x1E}, 1, 0},
    {0x44, (uint8_t []){0x0F}, 1, 0},
    {0x45, (uint8_t []){0x28}, 1, 0},
    {0x4B, (uint8_t []){0x04}, 1, 0},
    {0x55, (uint8_t []){0x02}, 1, 0},
    {0x56, (uint8_t []){0x01}, 1, 0},
    {0x57, (uint8_t []){0xA9}, 1, 0},
    {0x58, (uint8_t []){0x0A}, 1, 0},
    {0x59, (uint8_t []){0x0A}, 1, 0},
    {0x5A, (uint8_t []){0x37}, 1, 0},
    {0x5B, (uint8_t []){0x19}, 1, 0},
    {0x5D, (uint8_t []){0x78}, 1, 0},
    {0x5E, (uint8_t []){0x63}, 1, 0},
    {0x5F, (uint8_t []){0x54}, 1, 0},
    {0x60, (uint8_t []){0x49}, 1, 0},
    {0x61, (uint8_t []){0x45}, 1, 0},
    {0x62, (uint8_t []){0x38}, 1, 0},
    {0x63, (uint8_t []){0x3D}, 1, 0},
    {0x64, (uint8_t []){0x28}, 1, 0},
    {0x65, (uint8_t []){0x43}, 1, 0},
    {0x66, (uint8_t []){0x41}, 1, 0},
    {0x67, (uint8_t []){0x43}, 1, 0},
```

```
    {0x68, (uint8_t []){0x62}, 1, 0},
    {0x69, (uint8_t []){0x50}, 1, 0},
    {0x6A, (uint8_t []){0x57}, 1, 0},
    {0x6B, (uint8_t []){0x49}, 1, 0},
    {0x6C, (uint8_t []){0x44}, 1, 0},
    {0x6D, (uint8_t []){0x37}, 1, 0},
    {0x6E, (uint8_t []){0x23}, 1, 0},
    {0x6F, (uint8_t []){0x10}, 1, 0},
    {0x70, (uint8_t []){0x78}, 1, 0},
    {0x71, (uint8_t []){0x63}, 1, 0},
    {0x72, (uint8_t []){0x54}, 1, 0},
    {0x73, (uint8_t []){0x49}, 1, 0},
    {0x74, (uint8_t []){0x45}, 1, 0},
    {0x75, (uint8_t []){0x38}, 1, 0},
    {0x76, (uint8_t []){0x3D}, 1, 0},
    {0x77, (uint8_t []){0x28}, 1, 0},
    {0x78, (uint8_t []){0x43}, 1, 0},
    {0x79, (uint8_t []){0x41}, 1, 0},
    {0x7A, (uint8_t []){0x43}, 1, 0},
    {0x7B, (uint8_t []){0x62}, 1, 0},
    {0x7C, (uint8_t []){0x50}, 1, 0},
    {0x7D, (uint8_t []){0x57}, 1, 0},
    {0x7E, (uint8_t []){0x49}, 1, 0},
    {0x7F, (uint8_t []){0x44}, 1, 0},
    {0x80, (uint8_t []){0x37}, 1, 0},
    {0x81, (uint8_t []){0x23}, 1, 0},
    {0x82, (uint8_t []){0x10}, 1, 0},
    {0xE0, (uint8_t []){0x02}, 1, 0},
    {0x00, (uint8_t []){0x47}, 1, 0},
    {0x01, (uint8_t []){0x47}, 1, 0},
    {0x02, (uint8_t []){0x45}, 1, 0},
    {0x03, (uint8_t []){0x45}, 1, 0},
    {0x04, (uint8_t []){0x4B}, 1, 0},
    {0x05, (uint8_t []){0x4B}, 1, 0},
    {0x06, (uint8_t []){0x49}, 1, 0},
    {0x07, (uint8_t []){0x49}, 1, 0},
    {0x08, (uint8_t []){0x41}, 1, 0},
    {0x09, (uint8_t []){0x1F}, 1, 0},
    {0x0A, (uint8_t []){0x1F}, 1, 0},
    {0x0B, (uint8_t []){0x1F}, 1, 0},
    {0x0C, (uint8_t []){0x1F}, 1, 0},
    {0x0D, (uint8_t []){0x1F}, 1, 0},
    {0x0E, (uint8_t []){0x1F}, 1, 0},
    {0x0F, (uint8_t []){0x5F}, 1, 0},
    {0x10, (uint8_t []){0x5F}, 1, 0},
    {0x11, (uint8_t []){0x57}, 1, 0},
    {0x12, (uint8_t []){0x77}, 1, 0},
    {0x13, (uint8_t []){0x35}, 1, 0},
    {0x14, (uint8_t []){0x1F}, 1, 0},
    {0x15, (uint8_t []){0x1F}, 1, 0},
    {0x16, (uint8_t []){0x46}, 1, 0},
    {0x17, (uint8_t []){0x46}, 1, 0},
    {0x18, (uint8_t []){0x44}, 1, 0},
    {0x19, (uint8_t []){0x44}, 1, 0},
    {0x1A, (uint8_t []){0x4A}, 1, 0},
```

```
    {0x1B, (uint8_t []){0x4A}, 1, 0},
    {0x1C, (uint8_t []){0x48}, 1, 0},
    {0x1D, (uint8_t []){0x48}, 1, 0},
    {0x1E, (uint8_t []){0x40}, 1, 0},
    {0x1F, (uint8_t []){0x1F}, 1, 0},
    {0x20, (uint8_t []){0x1F}, 1, 0},
    {0x21, (uint8_t []){0x1F}, 1, 0},
    {0x22, (uint8_t []){0x1F}, 1, 0},
    {0x23, (uint8_t []){0x1F}, 1, 0},
    {0x24, (uint8_t []){0x1F}, 1, 0},
    {0x25, (uint8_t []){0x5F}, 1, 0},
    {0x26, (uint8_t []){0x5F}, 1, 0},
    {0x27, (uint8_t []){0x57}, 1, 0},
    {0x28, (uint8_t []){0x77}, 1, 0},
    {0x29, (uint8_t []){0x35}, 1, 0},
    {0x2A, (uint8_t []){0x1F}, 1, 0},
    {0x2B, (uint8_t []){0x1F}, 1, 0},
    {0x58, (uint8_t []){0x40}, 1, 0},
    {0x59, (uint8_t []){0x00}, 1, 0},
    {0x5A, (uint8_t []){0x00}, 1, 0},
    {0x5B, (uint8_t []){0x10}, 1, 0},
    {0x5C, (uint8_t []){0x06}, 1, 0},
    {0x5D, (uint8_t []){0x40}, 1, 0},
    {0x5E, (uint8_t []){0x01}, 1, 0},
    {0x5F, (uint8_t []){0x02}, 1, 0},
    {0x60, (uint8_t []){0x30}, 1, 0},
    {0x61, (uint8_t []){0x01}, 1, 0},
    {0x62, (uint8_t []){0x02}, 1, 0},
    {0x63, (uint8_t []){0x03}, 1, 0},
    {0x64, (uint8_t []){0x6B}, 1, 0},
    {0x65, (uint8_t []){0x05}, 1, 0},
    {0x66, (uint8_t []){0x0C}, 1, 0},
    {0x67, (uint8_t []){0x73}, 1, 0},
    {0x68, (uint8_t []){0x09}, 1, 0},
    {0x69, (uint8_t []){0x03}, 1, 0},
    {0x6A, (uint8_t []){0x56}, 1, 0},
    {0x6B, (uint8_t []){0x08}, 1, 0},
    {0x6C, (uint8_t []){0x00}, 1, 0},
    {0x6D, (uint8_t []){0x04}, 1, 0},
    {0x6E, (uint8_t []){0x04}, 1, 0},
    {0x6F, (uint8_t []){0x88}, 1, 0},
    {0x70, (uint8_t []){0x00}, 1, 0},
    {0x71, (uint8_t []){0x00}, 1, 0},
    {0x72, (uint8_t []){0x06}, 1, 0},
    {0x73, (uint8_t []){0x7B}, 1, 0},
    {0x74, (uint8_t []){0x00}, 1, 0},
    {0x75, (uint8_t []){0xF8}, 1, 0},
    {0x76, (uint8_t []){0x00}, 1, 0},
    {0x77, (uint8_t []){0xD5}, 1, 0},
    {0x78, (uint8_t []){0x2E}, 1, 0},
    {0x79, (uint8_t []){0x12}, 1, 0},
    {0x7A, (uint8_t []){0x03}, 1, 0},
    {0x7B, (uint8_t []){0x00}, 1, 0},
    {0x7C, (uint8_t []){0x00}, 1, 0},
    {0x7D, (uint8_t []){0x03}, 1, 0},
```

```
    {0x7E, (uint8_t []){0x7B}, 1, 0},
    {0xE0, (uint8_t []){0x04}, 1, 0},
    {0x00, (uint8_t []){0x0E}, 1, 0},
    {0x02, (uint8_t []){0xB3}, 1, 0},
    {0x09, (uint8_t []){0x60}, 1, 0},
    {0x0E, (uint8_t []){0x2A}, 1, 0},
    {0x36, (uint8_t []){0x59}, 1, 0},
    {0xE0, (uint8_t []){0x00}, 1, 0},
    {0x11, (uint8_t []){0x00}, 1, 120},
    {0x29, (uint8_t []){0x00}, 1, 120},
    {0x35, (uint8_t []){0x00}, 1, 0},
};
```

10.1-inch

```
const jd9365_lcd_init_cmd_t esp32p4_4d_init_cmds[] = {
    {0xE0, (uint8_t[]){0x00}, 1, 0},
    {0xE1, (uint8_t[]){0x93}, 1, 0},
    {0xE2, (uint8_t[]){0x65}, 1, 0},
    {0xE3, (uint8_t[]){0xF8}, 1, 0},
    {0x80, (uint8_t[]){0x01}, 1, 0},
    {0xE0, (uint8_t[]){0x01}, 1, 0},
    {0x00, (uint8_t[]){0x00}, 1, 0},
    {0x01, (uint8_t[]){0x26}, 1, 0},
    {0x03, (uint8_t[]){0x10}, 1, 0},
    {0x04, (uint8_t[]){0x2F}, 1, 0},
    {0x0C, (uint8_t[]){0x74}, 1, 0},
    {0x17, (uint8_t[]){0x00}, 1, 0},
    {0x18, (uint8_t[]){0xD7}, 1, 0},
    {0x19, (uint8_t[]){0x01}, 1, 0},
    {0x1A, (uint8_t[]){0x00}, 1, 0},
    {0x1B, (uint8_t[]){0xD7}, 1, 0},
    {0x1C, (uint8_t[]){0x01}, 1, 0},
    {0x24, (uint8_t[]){0xFE}, 1, 0},
    {0x37, (uint8_t[]){0x19}, 1, 0},
    {0x35, (uint8_t[]){0x28}, 1, 0},
    {0x38, (uint8_t[]){0x05}, 1, 0},
    {0x39, (uint8_t[]){0x08}, 1, 0},
    {0x3A, (uint8_t[]){0x12}, 1, 0},
    {0x3C, (uint8_t[]){0x78}, 1, 0},
    {0x3D, (uint8_t[]){0xFF}, 1, 0},
    {0x3E, (uint8_t[]){0xFF}, 1, 0},
    {0x3F, (uint8_t[]){0x7F}, 1, 0},
    {0x40, (uint8_t[]){0x06}, 1, 0},
    {0x41, (uint8_t[]){0xA0}, 1, 0},
    {0x43, (uint8_t[]){0x1E}, 1, 0},
    {0x44, (uint8_t[]){0x0B}, 1, 0},
    {0x55, (uint8_t[]){0x02}, 1, 0},
    {0x57, (uint8_t[]){0x6A}, 1, 0},
    {0x59, (uint8_t[]){0x0A}, 1, 0},
    {0x5A, (uint8_t[]){0x2E}, 1, 0},
    {0x5B, (uint8_t[]){0x1A}, 1, 0},
    {0x5C, (uint8_t[]){0x15}, 1, 0},
    {0x5D, (uint8_t[]){0x7F}, 1, 0},
    {0x5E, (uint8_t[]){0x5C}, 1, 0},
    {0x5F, (uint8_t[]){0x4B}, 1, 0},
    {0x60, (uint8_t[]){0x3E}, 1, 0},
    {0x61, (uint8_t[]){0x39}, 1, 0},
    {0x62, (uint8_t[]){0x2B}, 1, 0},
    {0x63, (uint8_t[]){0x2F}, 1, 0},
    {0x64, (uint8_t[]){0x1A}, 1, 0},
    {0x65, (uint8_t[]){0x33}, 1, 0},
    {0x66, (uint8_t[]){0x32}, 1, 0},
    {0x67, (uint8_t[]){0x32}, 1, 0},
    {0x68, (uint8_t[]){0x51}, 1, 0},
    {0x69, (uint8_t[]){0x3F}, 1, 0},
```

```
    {0x6A, (uint8_t[]){0x45}, 1, 0},
    {0x6B, (uint8_t[]){0x37}, 1, 0},
    {0x6C, (uint8_t[]){0x33}, 1, 0},
    {0x6D, (uint8_t[]){0x25}, 1, 0},
    {0x6E, (uint8_t[]){0x13}, 1, 0},
    {0x6F, (uint8_t[]){0x02}, 1, 0},
    {0x70, (uint8_t[]){0x7F}, 1, 0},
    {0x71, (uint8_t[]){0x5C}, 1, 0},
    {0x72, (uint8_t[]){0x4B}, 1, 0},
    {0x73, (uint8_t[]){0x3E}, 1, 0},
    {0x74, (uint8_t[]){0x39}, 1, 0},
    {0x75, (uint8_t[]){0x2B}, 1, 0},
    {0x76, (uint8_t[]){0x2F}, 1, 0},
    {0x77, (uint8_t[]){0x1A}, 1, 0},
    {0x78, (uint8_t[]){0x33}, 1, 0},
    {0x79, (uint8_t[]){0x32}, 1, 0},
    {0x7A, (uint8_t[]){0x32}, 1, 0},
    {0x7B, (uint8_t[]){0x51}, 1, 0},
    {0x7C, (uint8_t[]){0x3F}, 1, 0},
    {0x7D, (uint8_t[]){0x45}, 1, 0},
    {0x7E, (uint8_t[]){0x37}, 1, 0},
    {0x7F, (uint8_t[]){0x33}, 1, 0},
    {0x80, (uint8_t[]){0x25}, 1, 0},
    {0x81, (uint8_t[]){0x13}, 1, 0},
    {0x82, (uint8_t[]){0x02}, 1, 0},
    {0xE0, (uint8_t[]){0x02}, 1, 0},
    {0x00, (uint8_t[]){0x52}, 1, 0},
    {0x01, (uint8_t[]){0x5F}, 1, 0},
    {0x02, (uint8_t[]){0x5F}, 1, 0},
    {0x03, (uint8_t[]){0x50}, 1, 0},
    {0x04, (uint8_t[]){0x77}, 1, 0},
    {0x05, (uint8_t[]){0x57}, 1, 0},
    {0x06, (uint8_t[]){0x5F}, 1, 0},
    {0x07, (uint8_t[]){0x4E}, 1, 0},
    {0x08, (uint8_t[]){0x4C}, 1, 0},
    {0x09, (uint8_t[]){0x5F}, 1, 0},
    {0x0A, (uint8_t[]){0x4A}, 1, 0},
    {0x0B, (uint8_t[]){0x48}, 1, 0},
    {0x0C, (uint8_t[]){0x5F}, 1, 0},
    {0x0D, (uint8_t[]){0x46}, 1, 0},
    {0x0E, (uint8_t[]){0x44}, 1, 0},
    {0x0F, (uint8_t[]){0x40}, 1, 0},
    {0x10, (uint8_t[]){0x5F}, 1, 0},
    {0x11, (uint8_t[]){0x5F}, 1, 0},
    {0x12, (uint8_t[]){0x5F}, 1, 0},
    {0x13, (uint8_t[]){0x5F}, 1, 0},
    {0x14, (uint8_t[]){0x5F}, 1, 0},
    {0x15, (uint8_t[]){0x5F}, 1, 0},
    {0x16, (uint8_t[]){0x53}, 1, 0},
    {0x17, (uint8_t[]){0x5F}, 1, 0},
    {0x18, (uint8_t[]){0x5F}, 1, 0},
    {0x19, (uint8_t[]){0x51}, 1, 0},
    {0x1A, (uint8_t[]){0x77}, 1, 0},
    {0x1B, (uint8_t[]){0x57}, 1, 0},
    {0x1C, (uint8_t[]){0x5F}, 1, 0},
```

```
    {0x1D, (uint8_t[]){0x4F}, 1, 0},
    {0x1E, (uint8_t[]){0x4D}, 1, 0},
    {0x1F, (uint8_t[]){0x5F}, 1, 0},
    {0x20, (uint8_t[]){0x4B}, 1, 0},
    {0x21, (uint8_t[]){0x49}, 1, 0},
    {0x22, (uint8_t[]){0x5F}, 1, 0},
    {0x23, (uint8_t[]){0x47}, 1, 0},
    {0x24, (uint8_t[]){0x45}, 1, 0},
    {0x25, (uint8_t[]){0x41}, 1, 0},
    {0x26, (uint8_t[]){0x5F}, 1, 0},
    {0x27, (uint8_t[]){0x5F}, 1, 0},
    {0x28, (uint8_t[]){0x5F}, 1, 0},
    {0x29, (uint8_t[]){0x5F}, 1, 0},
    {0x2A, (uint8_t[]){0x5F}, 1, 0},
    {0x2B, (uint8_t[]){0x5F}, 1, 0},
    {0x2C, (uint8_t[]){0x13}, 1, 0},
    {0x2D, (uint8_t[]){0x1F}, 1, 0},
    {0x2E, (uint8_t[]){0x1F}, 1, 0},
    {0x2F, (uint8_t[]){0x01}, 1, 0},
    {0x30, (uint8_t[]){0x17}, 1, 0},
    {0x31, (uint8_t[]){0x17}, 1, 0},
    {0x32, (uint8_t[]){0x1F}, 1, 0},
    {0x33, (uint8_t[]){0x0D}, 1, 0},
    {0x34, (uint8_t[]){0x0F}, 1, 0},
    {0x35, (uint8_t[]){0x1F}, 1, 0},
    {0x36, (uint8_t[]){0x05}, 1, 0},
    {0x37, (uint8_t[]){0x07}, 1, 0},
    {0x38, (uint8_t[]){0x1F}, 1, 0},
    {0x39, (uint8_t[]){0x09}, 1, 0},
    {0x3A, (uint8_t[]){0x0B}, 1, 0},
    {0x3B, (uint8_t[]){0x11}, 1, 0},
    {0x3C, (uint8_t[]){0x1F}, 1, 0},
    {0x3D, (uint8_t[]){0x1F}, 1, 0},
    {0x3E, (uint8_t[]){0x1F}, 1, 0},
    {0x3F, (uint8_t[]){0x1F}, 1, 0},
    {0x40, (uint8_t[]){0x1F}, 1, 0},
    {0x41, (uint8_t[]){0x1F}, 1, 0},
    {0x42, (uint8_t[]){0x12}, 1, 0},
    {0x43, (uint8_t[]){0x1F}, 1, 0},
    {0x44, (uint8_t[]){0x1F}, 1, 0},
    {0x45, (uint8_t[]){0x00}, 1, 0},
    {0x46, (uint8_t[]){0x17}, 1, 0},
    {0x47, (uint8_t[]){0x17}, 1, 0},
    {0x48, (uint8_t[]){0x1F}, 1, 0},
    {0x49, (uint8_t[]){0x0C}, 1, 0},
    {0x4A, (uint8_t[]){0x0E}, 1, 0},
    {0x4B, (uint8_t[]){0x1F}, 1, 0},
    {0x4C, (uint8_t[]){0x04}, 1, 0},
    {0x4D, (uint8_t[]){0x06}, 1, 0},
    {0x4E, (uint8_t[]){0x1F}, 1, 0},
    {0x4F, (uint8_t[]){0x08}, 1, 0},
    {0x50, (uint8_t[]){0x0A}, 1, 0},
    {0x51, (uint8_t[]){0x10}, 1, 0},
    {0x52, (uint8_t[]){0x1F}, 1, 0},
    {0x53, (uint8_t[]){0x1F}, 1, 0},
```

```
    {0x54, (uint8_t[]){0x1F}, 1, 0},
    {0x55, (uint8_t[]){0x1F}, 1, 0},
    {0x56, (uint8_t[]){0x1F}, 1, 0},
    {0x57, (uint8_t[]){0x1F}, 1, 0},
    {0x58, (uint8_t[]){0x40}, 1, 0},
    {0x5B, (uint8_t[]){0x10}, 1, 0},
    {0x5C, (uint8_t[]){0x06}, 1, 0},
    {0x5D, (uint8_t[]){0x40}, 1, 0},
    {0x5E, (uint8_t[]){0x00}, 1, 0},
    {0x5F, (uint8_t[]){0x00}, 1, 0},
    {0x60, (uint8_t[]){0x40}, 1, 0},
    {0x61, (uint8_t[]){0x03}, 1, 0},
    {0x62, (uint8_t[]){0x04}, 1, 0},
    {0x63, (uint8_t[]){0x6C}, 1, 0},
    {0x64, (uint8_t[]){0x6C}, 1, 0},
    {0x65, (uint8_t[]){0x75}, 1, 0},
    {0x66, (uint8_t[]){0x08}, 1, 0},
    {0x67, (uint8_t[]){0xB4}, 1, 0},
    {0x68, (uint8_t[]){0x08}, 1, 0},
    {0x69, (uint8_t[]){0x6C}, 1, 0},
    {0x6A, (uint8_t[]){0x6C}, 1, 0},
    {0x6B, (uint8_t[]){0x0C}, 1, 0},
    {0x6D, (uint8_t[]){0x00}, 1, 0},
    {0x6E, (uint8_t[]){0x00}, 1, 0},
    {0x6F, (uint8_t[]){0x88}, 1, 0},
    {0x75, (uint8_t[]){0xBB}, 1, 0},
    {0x76, (uint8_t[]){0x00}, 1, 0},
    {0x77, (uint8_t[]){0x05}, 1, 0},
    {0x78, (uint8_t[]){0x2A}, 1, 0},
    {0xE0, (uint8_t[]){0x04}, 1, 0},
    {0x00, (uint8_t[]){0x0E}, 1, 0},
    {0x02, (uint8_t[]){0xB3}, 1, 0},
    {0x09, (uint8_t[]){0x61}, 1, 0},
    {0x0E, (uint8_t[]){0x48}, 1, 0},
    {0x2B, (uint8_t[]){0x0F}, 1, 0},
    {0x37, (uint8_t[]){0x58}, 1, 0},
    {0xE0, (uint8_t[]){0x00}, 1, 0},
    {0x11, (uint8_t[]){0x00}, 1, 120},
    {0x29, (uint8_t[]){0x00}, 1, 20},
};
```

11. Display Module Part Numbers

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

Examples:

- ESP32-P4-70
- ESP32-P4-80CT
- ESP32-P4-90CT-CLB

where:

```
ESP32-P4 - ESP32-P4 Display Family
70 - Display size (70 = 7.0", 80 = 8.0", 90 = 9.0" and 101 = 10.1")
CT - Capacitive Touch
CLB - Cover Lens Bezel
```

Note

- A product without a CT in the part number is a non-touch variant.
- 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. This is available for capacitive touch only.

12. 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 well-known and strong 3M 300LSE adhesives. The double-sided adhesive has a thickness of 0.175mm once the backing has been removed.

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

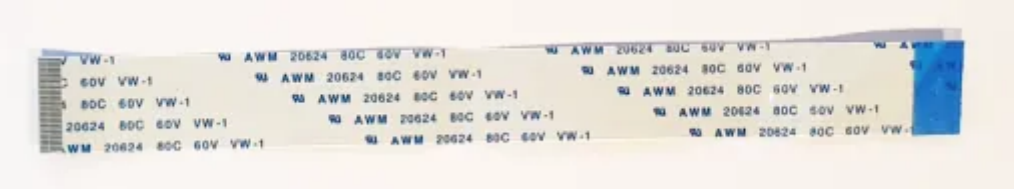
If you have an application where sealant is a better solution, please make contact with our Sales team, and discuss options for customising your order without 3M Tape, or anything else required.

13. FFC Cable

The FFC cables supplied by 4D Systems (included with products) have the following specifications:

- **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)

You can get different cable lengths from the 4D Systems website.



Note

If you are interfacing with this module directly to your product via the 30-way FFC rather than utilising a breakout board or 4D-UPA, suitable connectors are readily available from many electronics suppliers, such as Digikey, Mouser, Farnell, RS, etc.

A standard 30-pin, 0.5mm pitch, 0.3mm thick FFC, FFC connector. They are available in Top Contact and Bottom Contact, so depending how you orientate the cable on your product, will determine which one you need. Please however take care of the pinout and how it flows from the display module, through the FFC and into your product, to ensure Pin1 and Pin30 are where you expect them to be.

14. Starter Kit

4D Systems highly recommends all first-time buyers of 4D Systems' displays, to purchase the Starter Kit when purchasing their first 4D Systems display solution.

The Starter Kit provides all the hardware that is required to get the User up and running.

Starter Kits typically include:

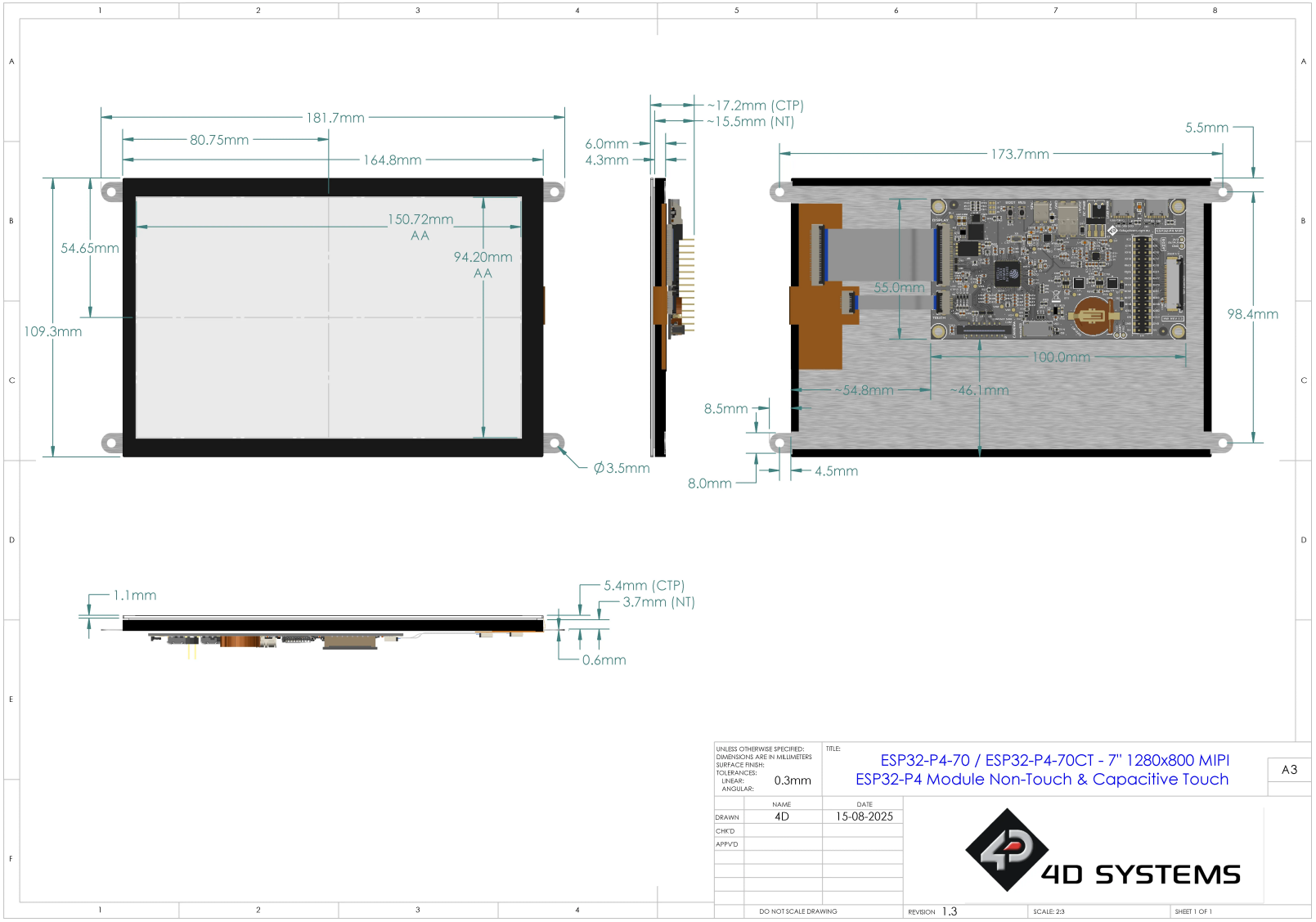
- ESP32-P4 Display Module
- gen4 Breakout Board (gen4-Breakout)
- 4D Universal Programming Adaptor (4D-UPA REV1.4 or higher)
- 4GB micro-SD Card
- 150mm 30-way FFC cable for connecting display to gen4-Breakout, 4D-UPA, or Users System

Please refer to the 4D Systems website for the current components included in the Starter Kit.

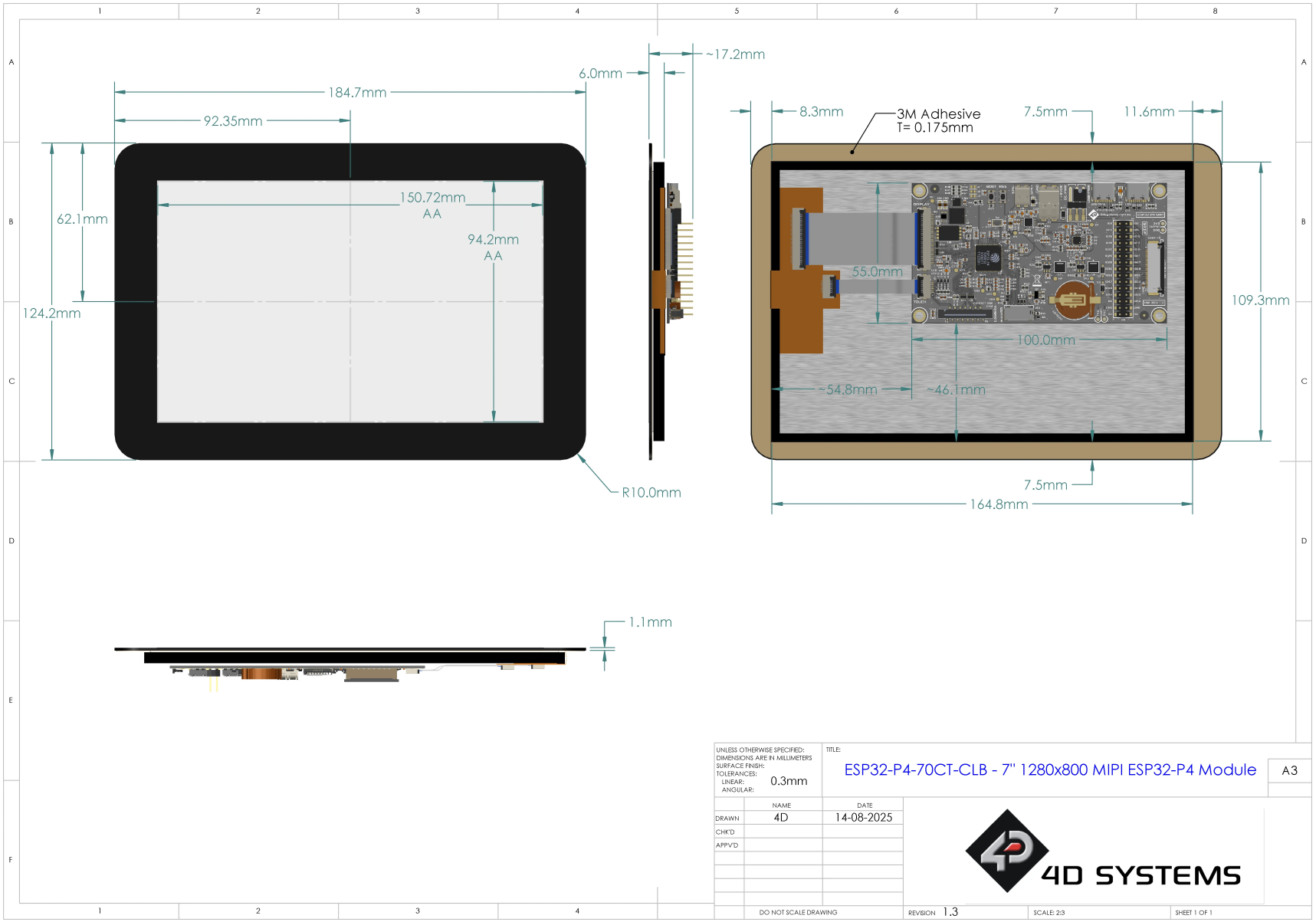
Simply select the Starter Kit option when purchasing the chosen display module on the 4D Systems shopping cart, or from your local distributor.

15. Mechanical Details

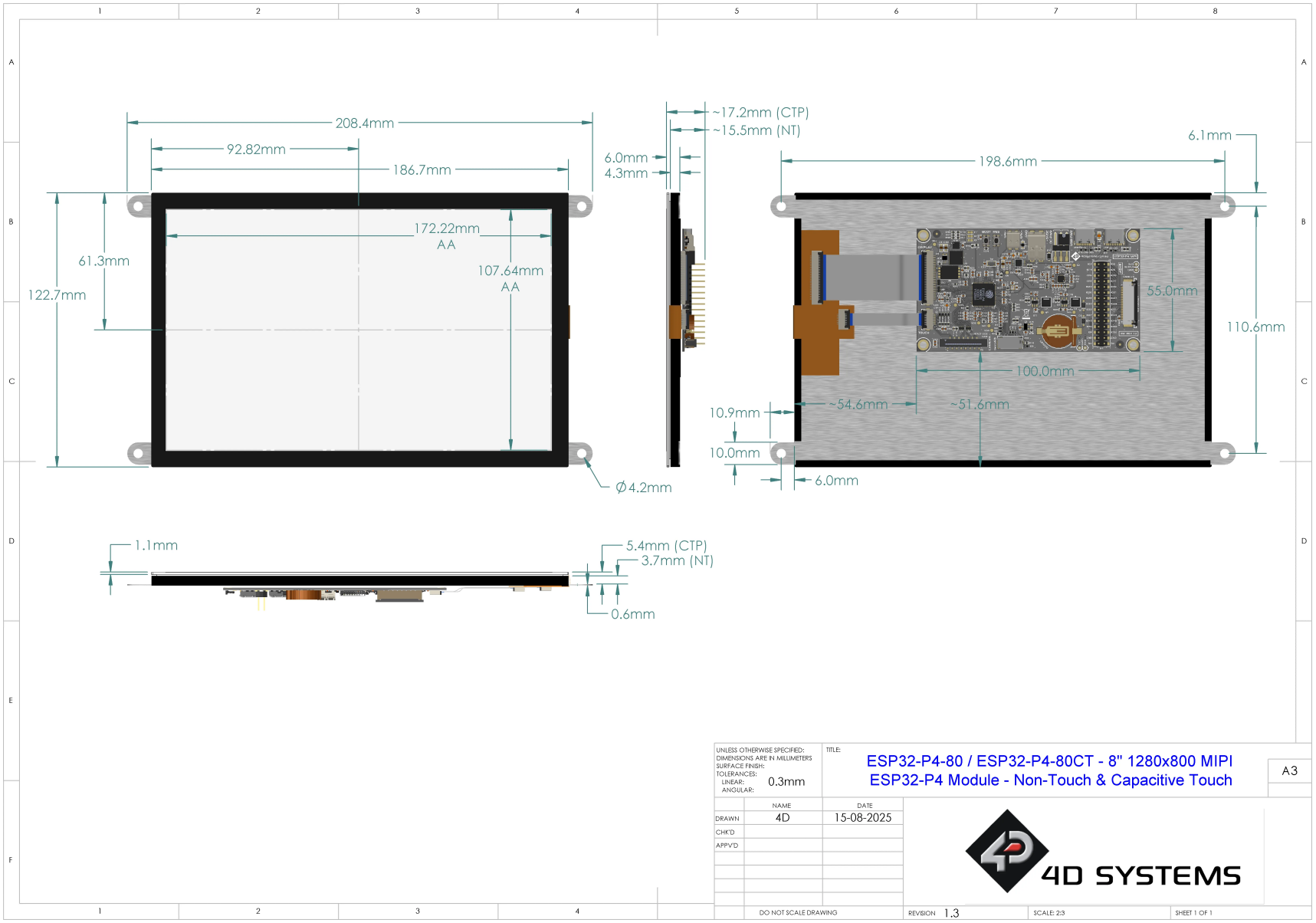
15.1. 7" Non-Touch, Capacitive Touch



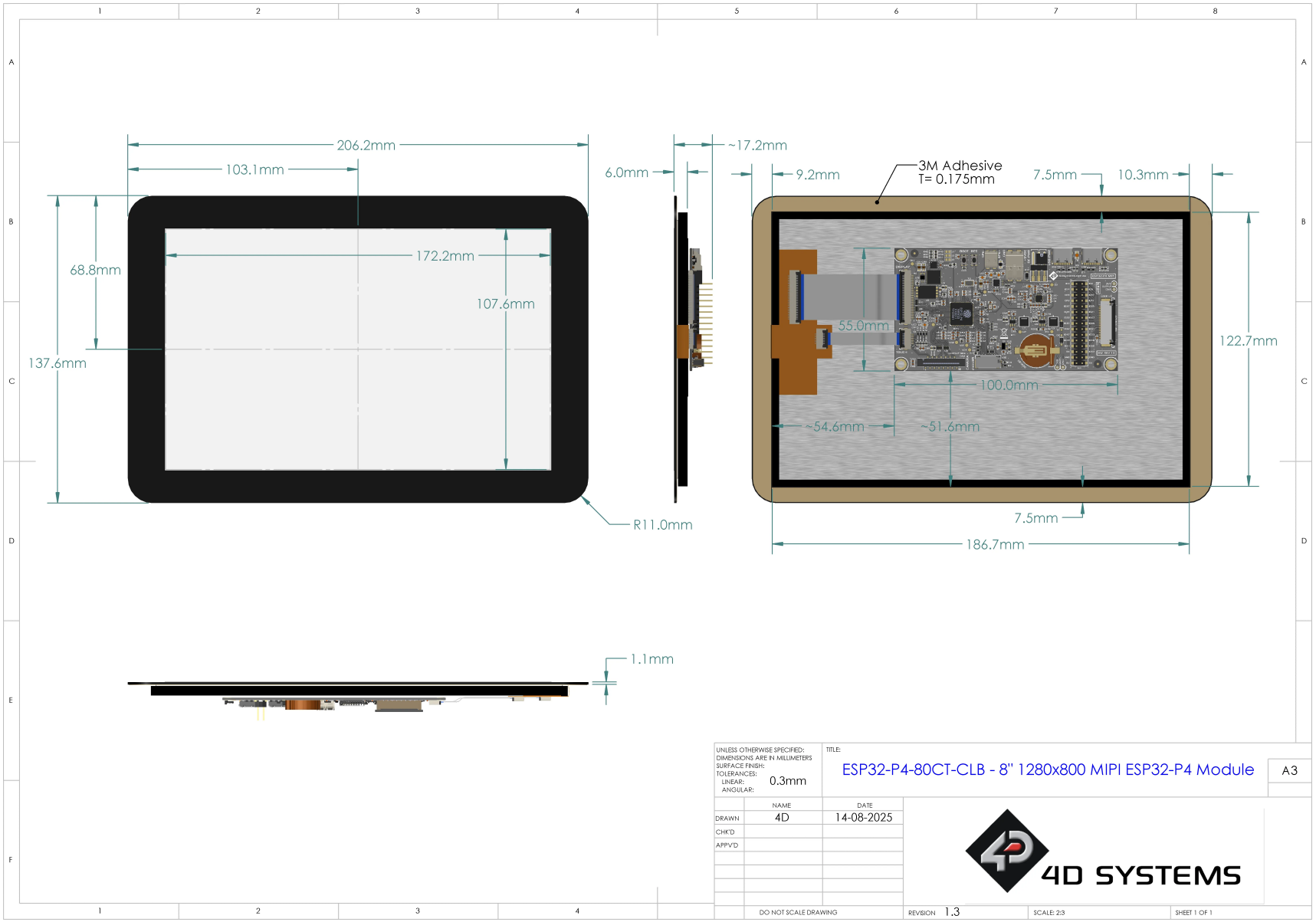
15.2. 7" Capacitive Touch w/ CLB



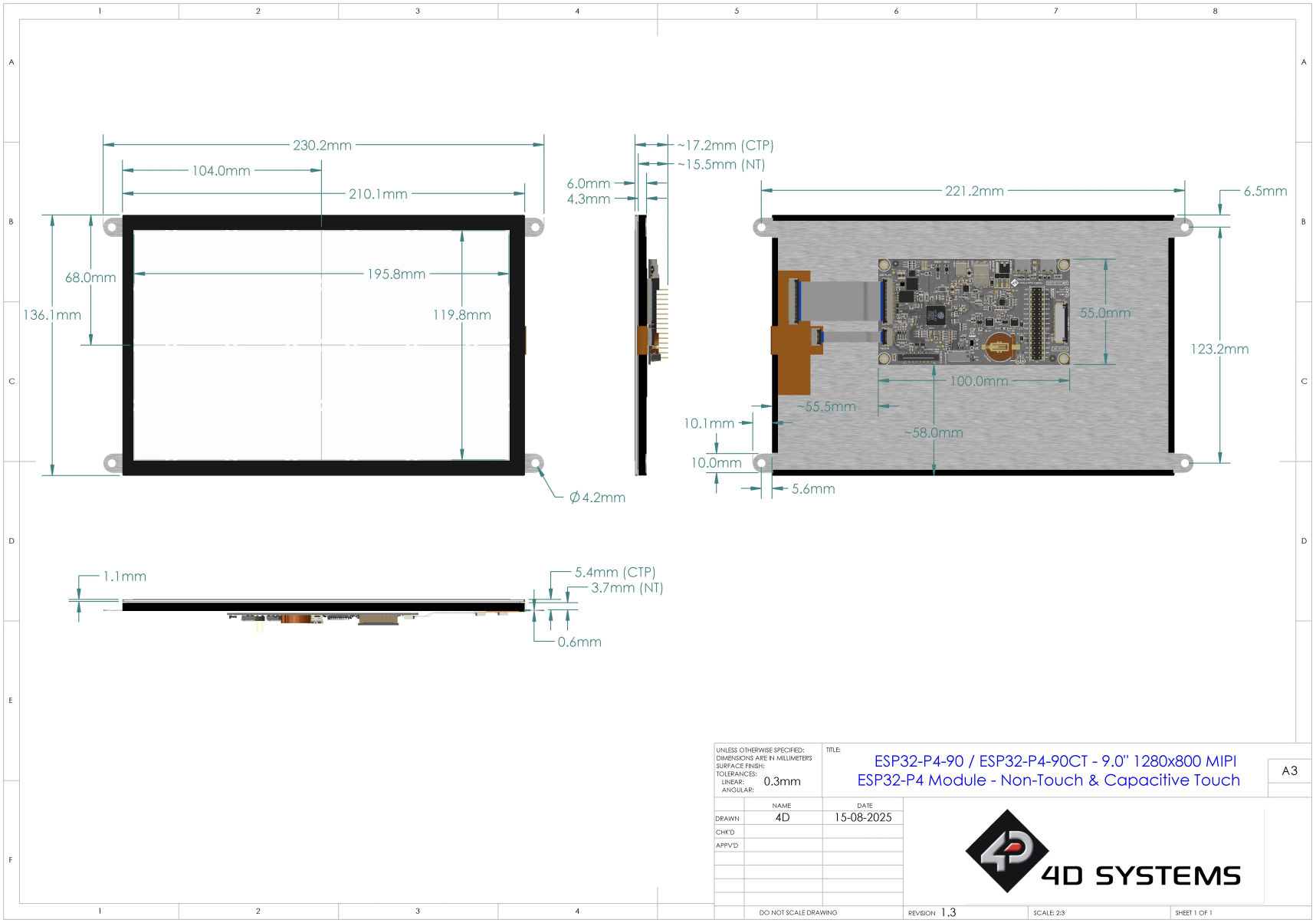
15.3. 8" Non-Touch, Capacitive Touch



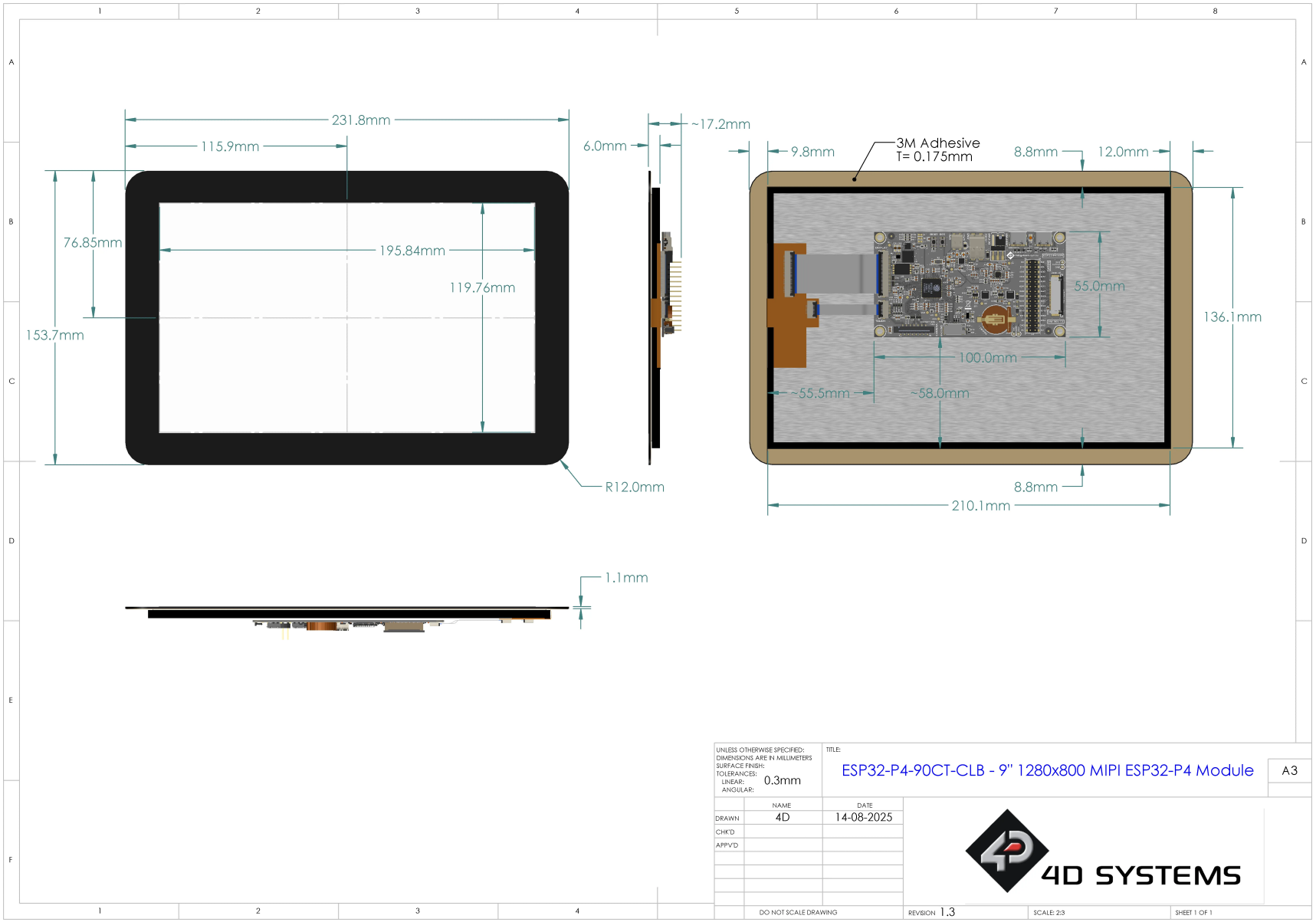
15.4. 8" Capacitive Touch w/ CLB



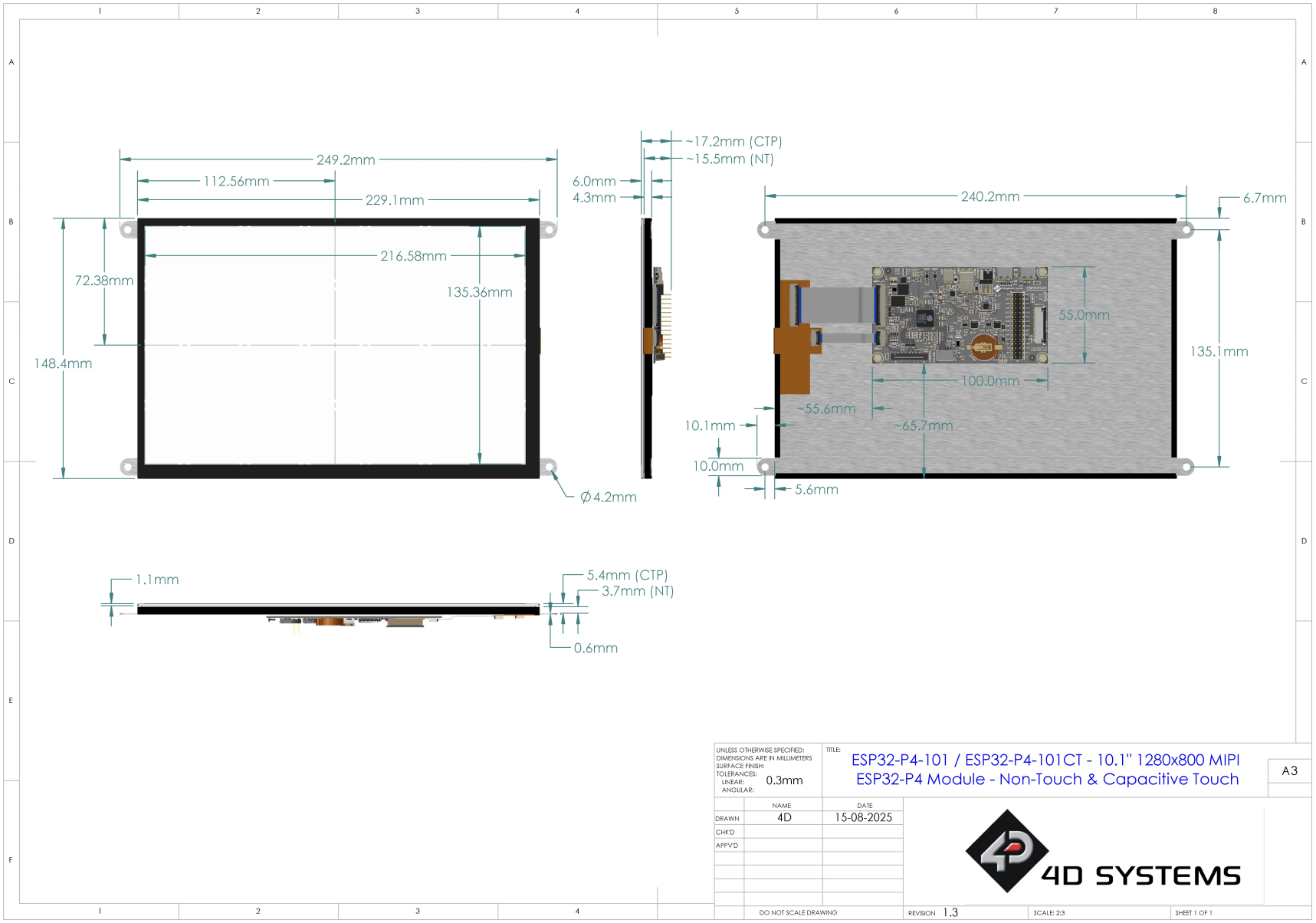
15.5. 8" Non-Touch, Capacitive Touch



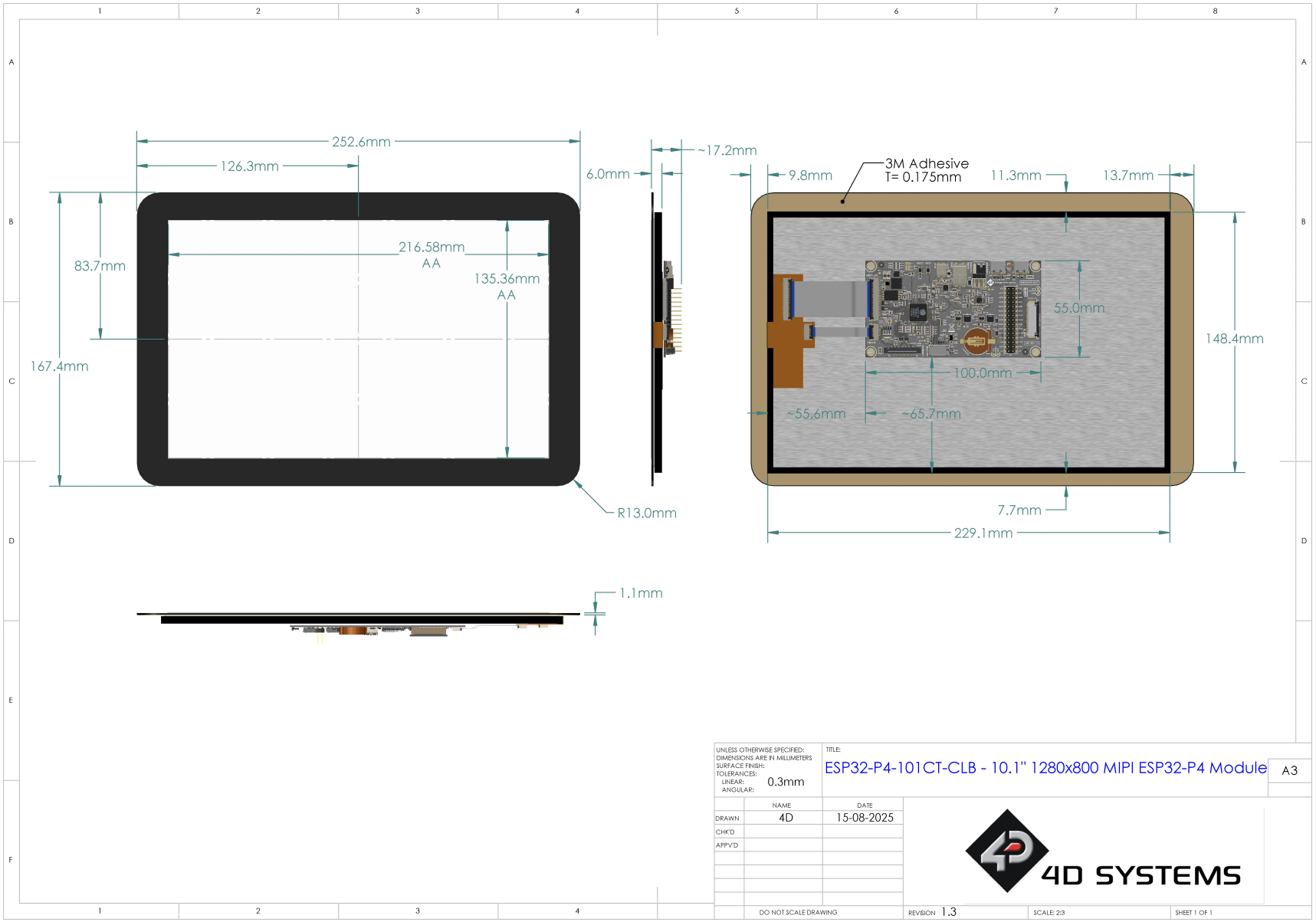
15.6. 9" Capacitive Touch w/ CLB



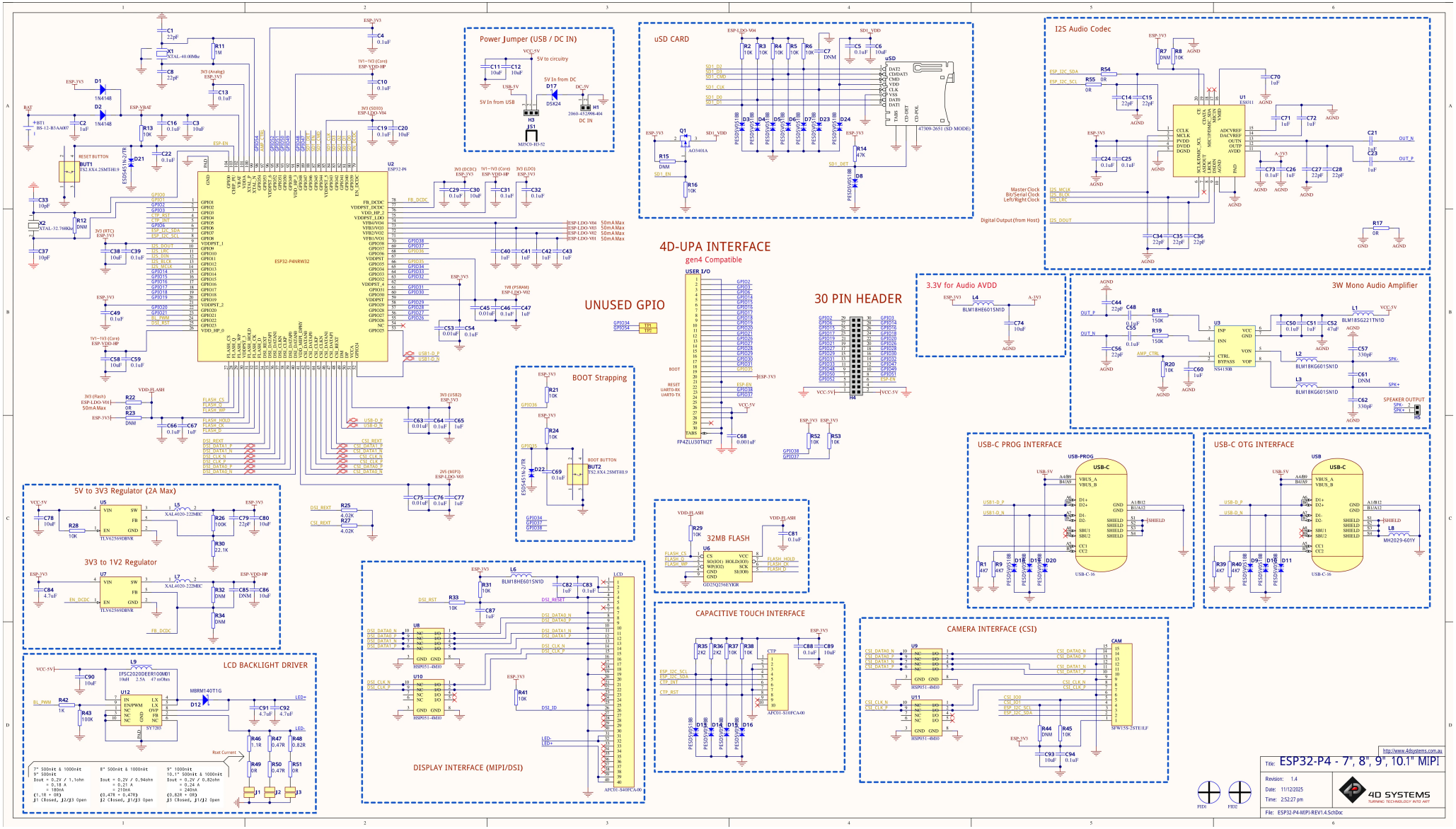
15.7. 10.1" Non-Touch, Capacitive Touch



15.8. 10.1" Capacitive Touch w/ CLB



16. Schematic Circuit Details REV 1.4



17. Specifications

Absolute Maximum Ratings

Operating ambient temperature	-20°C to +70°C (see note 1)
Storage temperature	-30°C to +80°C
Voltage on any digital input pin (ESP32) with respect to GND	-0.3V to 3.6V
Voltage on module VCC with respect to GND	-0.3V to 6.5V

Note

1. Temperature range for Ambient and Storage, are determined by a combination of components used on these modules. While some components may be capable of exceeding these temperatures, some are not, so the minimums/maximums are determined by the weakest device on the modules. The 'weakest' component on the module is the TFT LCD, which is capable of -20°C to 70°C Operating Temp.
2. Stresses above those listed here may cause permanent damage to the device. This is for 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 (VCC)	Stable external supply required	4.0	5.0	6.0	V
Processor voltage (VP)		–	3.3	–	V
Input Low Voltage (VIL)	all pins	0	–	0.25VP	V
Input High Voltage (VIH)	non 5V tolerant pins	0.75VP	–	VP+0.3	V
Input High Voltage (VIH)	5V tolerant pins	0.8VP	–	5.5	V
Output 3.3V Voltage	3.3V Output on FFC and Pads	–	3.3	–	V
Output 3.3V Current	Output Current capability (see note 1)	–	–	200	mA

Note

1. The output current of the 3.3V pins (found on the 30-way FFC connector, as well as the pads on the upper right hand side of the PCBA) are capable of approximately 200mA, however this is heavily dependant on what else the module is expected to do at the time. The 3.3V output is on the shared 3.3V rail, which is used by most of the components on the PCBA. If too much power is requested on the 3.3V output pins, it could sacrifice module stability. Keep current draw to lower levels where possible, and if a higher output current is required on 3.3V, consider using a separate power supply instead driven off the 5V pins, or completely independant source.

Global Characteristics Based on Operating Conditions

Parameter	Conditions	Min	Typ	Max	Units
Supply Current (ICC) ***	ESP32-P4-70 (Contrast = 15) @ 5V	–	865	–	mA
	ESP32-P4-70CT / CT-CLB (Contrast = 15) @ 5V	–	890	–	mA
	ESP32-P4-80 (Contrast = 15) @ 5V	–	1025	–	mA
	ESP32-P4-80CT / CT-CLB (Contrast = 15) @ 5V	–	1050	–	mA
	ESP32-P4-90 (Contrast = 15) @ 5V	–	1025	–	mA
	ESP32-P4-90CT / CT-CLB (Contrast = 15) @ 5V	–	1050	–	mA
	ESP32-P4-101 (Contrast = 15) @ 5V	–	1165	–	mA
	ESP32-P4-101CT / CT-CLB (Contrast = 15) @ 5V	–	1190	–	mA
Display Endurance	Hours of operation, measured to when display is 50% original brightness	30000	–	–	H
Touch Screen Transparency	Capacitive Touch	90	–	–	%
CLB Hardness	Cover Lens Bezel & CTP Glass Hardness	–	6	–	H

Note

Typical Supply Current (ICC) figures are without microSD card inserted, No peripherals connected, No Audio, and using simple display operations only. Any additional load such as GPIO sourcing, will increase this figure. This is a Typical figure only, not a Maximum.

LCD DISPLAY INFORMATION (IPS DISPLAYS)

Parameter	Conditions	Specification
Display Type	All Models	IPS - TFT Transmissive LCD
Display Size	All 7.0" models	7.0" Diagonal
	All 8.0" models	8.0" Diagonal
	All 9.0" models	9.0" Diagonal
	All 10.1" models	10.1" Diagonal
Display Resolution	All Models	1280 x 800 (Landscape/Wide Viewing)
Display Brightness	ESP32-P4-70 (Contrast = 15)	1000 cd/m2
	ESP32-P4-70CT / CT-CLB (Contrast = 15)	900 cd/m2
	ESP32-P4-80 (Contrast = 15)	1000 cd/m2
	ESP32-P4-80CT / CT-CLB (Contrast = 15)	900 cd/m2
	ESP32-P4-90 (Contrast = 15)	1000 cd/m2
	ESP32-P4-90CT / CT-CLB (Contrast = 15)	900 cd/m2
	ESP32-P4-101 (Contrast = 15)	1000 cd/m2
	ESP32-P4-101CT / CT-CLB (Contrast = 15)	900 cd/m2
Display Contrast Ratio	All 7.0" models - Typical	800:1
	All 8.0" models - Typical	1500:1
	All 9.0" models - Typical	1000:1
	All 10.1" models - Typical	1000:1
Display Viewing Angles	Above Centre	80 Degrees
	Below Centre	80 Degrees
	Left of Centre	80 Degrees
	Right of Centre	80 Degrees
Display Viewing Direction	All Sizes	ALL (Wide viewing IPS Display)
Display Backlighting	All 7.0" models - 18V 180mA	6x6 (6 Series, 6 Parallel) LED's
	All 8.0" models - 18V 210mA	6x7 (6 Series, 7 Parallel) LED's
	All 9.0" models - 18V 210mA	6x6 (6 Series, 6 Parallel) LED's
	All 10.1" models - 18V 240mA	6x7 (6 Series, 7 Parallel) LED's
Pixel Pitch	All 7" - Width x Height – Landscape	0.11775 x 0.11775mm (Square pixels)
	All 8" - Width x Height – Landscape	0.13455 x 0.13455mm (Square pixels)
	All 9" - Width x Height – Landscape	0.1497 x 0.153mm (non-Square pixels)
	All 10.1" - Width x Height – Landscape	0.1692 x 0.1692mm (Square pixels)
Pixel Density	Number of pixels in 1 row in 25.4mm, 7.0"	215 DPI/PPI (Horizontal) 215 DPI/PPI (Vertical)
	Number of pixels in 1 row in 25.4mm, 8.0"	188 DPI/PPI (Horizontal) 188 DPI/PPI (Vertical)
	Number of pixels in 1 row in 25.4mm, 9.0"	

Parameter	Conditions	Specification
		166 DPI/PPI (Horizontal) 169 DPI/PPI (Vertical)
	Number of pixels in 1 row in 25.4mm, 10.1"	150 DPI/PPI (Horizontal) 150 DPI/PPI (Vertical)

Note

Relevant for all displays, the Displays used are of the highest rated 'Grade A', which allows for 0-4 defective pixels. A defective pixel could be solid Black (Dead), White, Red, Green or Blue.

18. Revision History

Datasheet Revision

Revision Number	Date	Description
0.1	15/08/2025	Internal Use Only
1.0	11/12/2025	Initial Public Release Version
1.1	20/02/2026	Minor update regarding WS5 - expected support of ESP32-P4 Added some detail about the ESP32-P6-C6-WiFi module

Hardware Revision

Revision Number	Date	Description
1.0	24/04/2025	Initial Internal Version
1.2	12/06/2025	Internal Release Version
1.3	11/08/2025	Internal Release Version
1.4	11/12/2025	Public Release Version

19. Legal Notice

19.1. 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.

19.2. Disclaimer of Warranties & Limitations of Liabilities

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.