RX-TM-070845A

UltraTouch 7” Touch Display Module

with optional AD board

RXC-FL0725

Application Note: System Overview

Rev 1.1 October 2024

Quick Start Guide

RX-TM-070845A with optional AD board (RXC-FL0725) is ‘plug and play’. The display requires 12V DC power via a 5.52/2.5mm barrel socket, centre positive and video feed via a mini HDMI connector, type A. The LED backlight is powered directly from the AD board. PCAP (XY) and force touch (Z) operate though a powered I2C (6-pin PCB header) or a powered USB (micro-B) interface and will be reported by the host OS in ‘Device Manager’ / ‘Udev’ as a touch screen. Finger location (XY) and force (Z) are part of the HID reports sent to the host OS, with force detection reported in the “Pressure” container. Consistent force detection requires the touch display module (TDM)is firmly supported and affixed in a housing using the pre-applied VHB tape.

If the optional AD board has not been purchased, please refer to the Datasheet for information on the LVDS timing diagram, LED backlight driving and general precautions to observe when handling and using the parts.

Contents

Key Features

- 7" diagonal full color IPS LCD display
- HD resolution 1280 x 720
- 10 finger XY capacitive touch (default 5)
- UltraTouch Z-force sensitivity (350gf default)
- Programmable Z-force threshold (100-600gf)
- Nitrile glove operation up to 2mm thick
- Sunlight readable 1000 cd/m^2
- Chemically strengthened cover glass
- Anti-glare, Anti-reflective surface
- Mounting at any angle
- Landscape or portrait orientation
- Fully symmetric geometry and black print
- Flush mount to enclosure
- USB/I2C HID plug-and-play
- Industrial reliability & temperature range
- Tolerant to vibration from plant and machinery
- IP* & IK rating with suitable mounting and enclosure
- RoHS/REACH compliant
- Pre-configured for feature and firmware over the air upgrades

*As-supplied the PCAP controller is set to provide moisture rejection. It has the capability to be re-tunned to support rain and even a running water environment. The PCAP controller manufacturer should be consulted on the settings to use in these instances.

Parts List

RX-TM-070845A	UltraTouch 7” Touch Display Module, including
	UltraTouch PCBA incorporating PCAP controller
RXC-FL0725	AD board (optional)

Components List

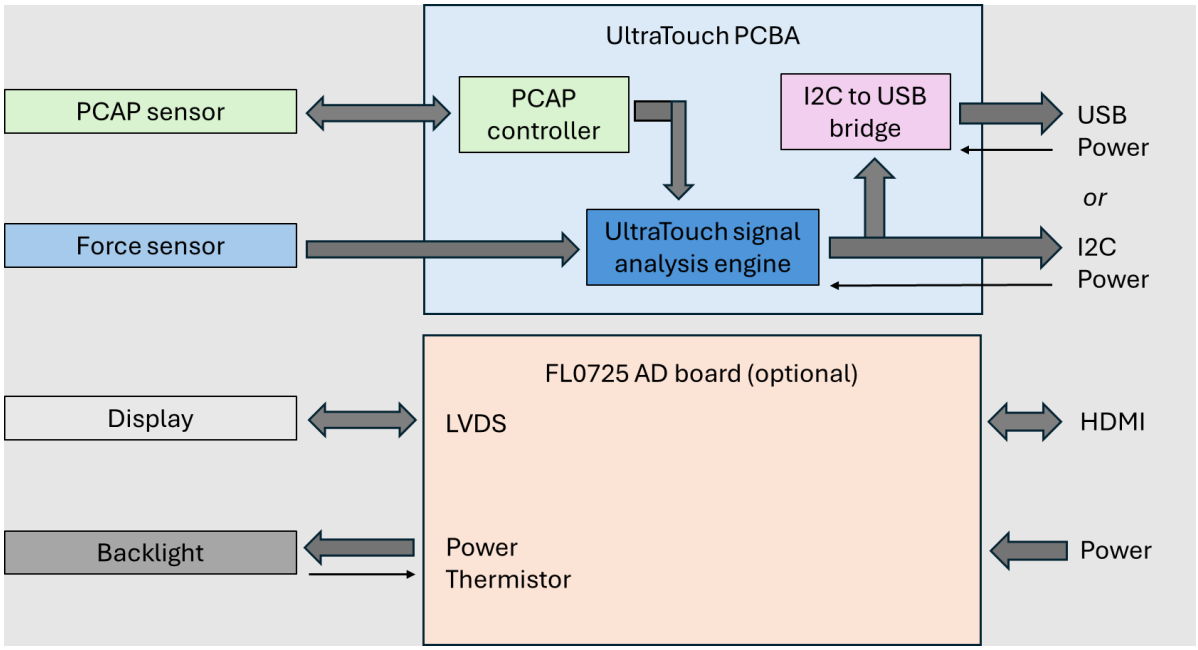
Part Number	Description	Picture
RXC-GF070854A-3.0	7 inch Touchscreen with force sensor technology	
RXL070135-B	7 inch high brightness LCD Display	
RXC-GF070854A-PCB-1.0	CTT PCBA fitted to the back of the display (control board for the touchscreen)	
RXC-FL0725	AD PCBA fitted to the back of the display (control board for the display)	

Hardware setup

Anti-static precautions should be taken at all times while handling the parts.

The display will be received with the purchased boards attached and interconnecting cables in place. Before installation, a visual check should be made of the hardware integrity to ensure there has not been any disturbance during transit.

System Block Diagram

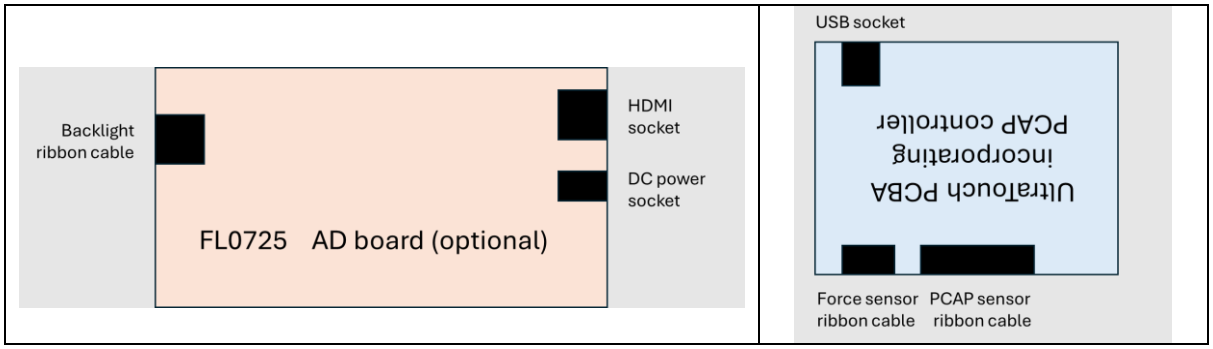


The UltraTouch PCBA handles the communication between the host OS and the PCAP and force sensors. UltraTouch communicates internally with the PCAP controller and combines the location and force information in the HID touch screen report.

The optional AD board is highly recommended since it undertakes the conversion of HDMI into LVDS and ensures the correct power-on timing sequence is followed. Inaddition, components for conversion of voltage to supply the LED backlight are included.

Key Port Locations

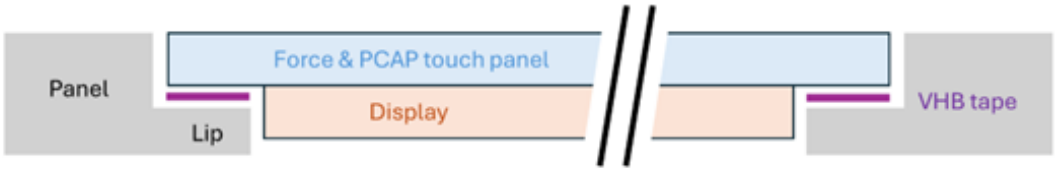
The key ports on the UltraTouch PCBA and optional AD board are located as indicated on the diagrams below. The boards are depicted with the silk screen side facing the viewer. Note that the UltraTouch PCBA is installed rotated 180 degrees relative to the optional AD board.



The UltraTouch PCBA is powered by default through the USB recptacle and is factory pre-set. An on-board jumper is provided if power through the I2C is preferred. Please consult with the manufacturer to avail of this option. Communication with the UltraTouch PCBA is selectable between USB (default) and I2C by a slide switch on the PCBA.

TDM Mounting Arrangement

The UltraTouch TDM is extremely versatile. It can be mounted in any orientation, at any angle and either flush-to-panel, recessed-in-panel, on-panel or behind panel. The drawing below illustrates the front-face, flush-mount, arrangement. The TDM is supplied with suitable Very High Bond (VHB) tape pre-applied to the perimeter, protected by a peel-off strip.



The perimeter of the touch display module should be affixed to and supported by a lip on the panel. The lip requires a flatness of +/- 0.2 mm and width similar to the black print. The panel lip needs to be sufficiently stiff so it does not distort or yield to any significant extent when the TDM is pushed hard by multiple fingers. Mounting orientation can be either portrait or landscape.

As with all displays of this type it is important there is an air-space behind. This facilitates cooling of the display and must provide sufficient clearance so that application of force to the front face of the display does not cause it to compress as this can cause irreversible damage to the display. An air-space of 10-20mm usually satisfies both requirements.

The detail of the mounting arrangement needs to be designed by the customer to suit the expected operating/storage environments and the required IP and IK ratings for the product. The perimeter of the touch display module can be sealed to the enclosure without significantly affecting the force touch sensitivity, provided the seal width is narrow relative to the black print area.

The stiffness of the display frame and the presence or absence of an edge seal will slightly alter the force touch threshold. For example, switching from an aluminum to a physically identical frame in ABS plastic will increase the force sensitivity by around 25-50gf (i.e. a lighter force is required to exceed the threshold). If necessary, compensation for this can be achieved by altering the default sensitivity as described below.

Unlike many other technologies used for touch screens, UltraTouch is agnostic to the angle at which the display is held. Performance is the same irrespective of whether the display is horizontal, vertical or being moved. This makes it suitable for use on monitor arms, mobile/portable products and high vibration environments.

Power-on Precautions

When used with the optional AD board, there are no constraints to the power-on sequence. Where the display is driven directly via the LVDS interface the timing diagrams (Power ON/Off Sequence) given in the datasheet must be strictly adhered to.

The UltraTouch system performs an auto-calibration routine on each power-on that takes less than 1 second to complete. No force should be applied to the touch display module during this period otherwise the base-line zero force level will be erroneous. If this occurs a simple power cycle of the UltraTouch PCB, with the force removed, will restore normal operation. The UltraTouch system can be forced to auto-calibrate without power cycling by following the procedure given in the following section.

UltraTouch Auto-Calibration

If the user presses with force that is above the minimum threshold for 4 seconds or more **at the same location**, the UltraTouch system will prime to perform an auto-calibration (i.e. same as the power-on calibration routine). This will start when the finger is lifted from the touch screen and will complete in under 1 second.

Host OS Interfacing

Using the optional AD board and HDMI connector, the display will be detected and operate as a HD display (1280 x 720p).

The PCAP and force sensors are connected to the host OS through the UltraTouch controller by either USB (default) or I2C. In both cases the touch system enumerates as an HID touch screen. Under Windows 10 and above and many Linux kernels, including Raspberry Pi OS, the default Microsoft HID descriptor is used. This includes a container for 'pressure', so no additional drivers or HID libraries are required.

The force sensitivity of the touch screen is pre-configured at 350gf. When a PCAP event is detected, the accompanying force is determined. If the applied force is below the threshold (i.e. very light touch) then the HID 'pressure' container is null (contains zero). If the applied force is above the threshold the HID 'pressure' container is populated with a non-zero value. The magnitude of the value is arbitrary and bears no relationship to the force applied. This is because the force sensor is not a simple 'weighing scale' but a dynamic entity with spatial and temporal dependency that further seeks to confirm the touch event is likely to have originated from a human finger. The force threshold can be altered, as described in the following section.

Some examples of code used to read an HID touch container, including 'pressure' are given in the Appendix.

The RX-TM-070845A touch display module is configured to report a pseudo-weighted average 'pressure' from all the fingers pushing on the display. Other UltraTouch TDMs can determine the location and force of individual fingers, either in combination with the PCAP system or purely from the force sensor.

Force Threshold Adjustment

UltraTouch can be configured to respond to a wide range of force, from a very light touch (approximately 100gf) to a very firm push (approximately 600gf). Eleven thresholds (0-10) across this range can be set are available to suit the use case and user preference. Access to this parameter can be achieved using an API that is available from Cambridge Touch Technologies. This service reads the current value and provides the ability to write a new value. The new force threshold is permanent and preserved on power cycling. The API includes the license manager for the product and the encryption necessary to communicate with the UltraTouch firmware. A change to the force threshold is achieved in less than one second so that it can be changed quickly and easily if required by the use case.

LED Backlight Dimming

The LED backlight is nominally set to operate at maximum brightness. A standard NTC 10K thermistor built into the module is used to sense the temperature of the LEDs. If it is required to alter the backlight current and consequent brightness as a function of temperature, the manufacturer can provide a custom AD board with the necessary circuitry.

The backlight may also be dimmed by attaching a physical keyboard, or keyboard emulator to the port provided on the AD board. Please refer to the User Manual for manufacturer part number RXC-FL0725 for details.

Appendix: Code examples to read an HID container including ‘pressure’

The code below is provided without liability, warranty or guarantee. Line numbers and spaces may need to be removed from the code depending on the compiler used.

Code Example: Python3 (evdev module running under Raspberry Pi OS)

```
1   from evdev import *
2   dev = InputDevice('/dev/input/event7')
3
4   print(dev)
5
6   X = 0
7   Y = 0
8   pressure = 0
9
10  for event in dev.read_loop():
11      if event.type == ecodes.EV_ABS:
12          if event.code == ecodes.ABS_X:
13              X = event.value
14          if event.code == ecodes.ABS_Y:
15              Y = event.value
16          if event.code == ecodes.ABS_PRESSURE:
17              pressure = event.value
18          print("X={} Y={} PRESSURE={}".format(X, Y, pressure))
```

Code Example: Qt6 Python (running under Windows/Ubuntu)

Note that some versions of Qt running in limited set of environments do not decode the "pressure" value correctly from kernel instructions. If this problem is encountered the HID-API library can be used as a work-around.

```
1    import sys
2
3    from PySide6.QtCore import *
4    from PySide6.QtGui import *
5    from PySide6.QtWidgets import *
6    from PySide6.QtSvgWidgets import *
7
8    class MainWindow(QMainWindow):
9        def __init__(self):
10           super().__init__()
11
12           self.setWindowTitle("My App")
13
14   button = QPushButton("Press Me!")
15   button.setMinimumWidth(1000);
16   button.setMinimumHeight(800);
17   button.setStyleSheet("font-size: 50px;");
18   button.setCheckable(True)
19   button.clicked.connect(self.the_button_was_clicked)
20
21   # Set the central widget of the Window.
22   self.setCentralWidget(button)

23   def the_button_was_clicked(self):
24       print("Clicked!")
25   def eventFilter(self, watched: QObject, event: QTouchEvent) -> bool:
26       if event.type() == QEvent.TouchBegin:
```

```
27            print(f"touch begin  : {watched.objectName()}")
28             touches = event.points()
29             for t in touches:
30                 pressure = round(t.pressure() * 3)
31                 print(f"touch pressure  : {pressure}")
32                 print(f"x : {t.pos().x()}")
33                 print(f"y : {t.pos().y()}")
34             return True
35         elif event.type() == QEvent.TouchUpdate:
36             print(f"touch update  : {watched.objectName()}")
37             touches = event.points()
38              for t in touches:
39                 pressure = round(t.pressure() * 3)
40                  print(f"touch pressure  : {pressure}")
41                  print(f"x : {t.pos().x()}")
42                 print(f"y : {t.pos().y()}")
43               return True
44             elif event.type() == QEvent.TouchEnd:
45                print(f"touch end  : {watched.objectName()}")
46                return True
47             elif event.type() == QEvent.MouseButtonPress:
48                print(f"MouseButtonPress  : {watched.objectName()}")
49                return True
50             elif event.type() == QEvent.MouseButtonRelease:
51               print(f"MouseButtonRelease  : {watched.objectName()}")
52               return True

53             return super().eventFilter(watched, event)
54
55         def event(self, event: QEvent) -> bool:
56             print(f"event={str(event.type().name)}")
57     ##       if event.type() == QEvent.HoverMove:
```

```
58      ##          print(f"points {event.points()}")
59      ##      if event.type() == QEvent.UpdateRequest:
60      ##          print(f"points {event}")
61
62      app = QApplication(sys.argv)
63      #app.setNavigationMode(Qt.NavigationModeCursorAuto)
64
65      window = MainWindow()
66      window.setAttribute(Qt.WA_AcceptTouchEvents, True)
67      window.setAttribute(Qt.WA_TouchPadAcceptSingleTouchEvents, True)
68      window.setAttribute(Qt.WA_MouseTracking, True)
69
70      window.installEventFilter(window)
71      window.show()
72
73      app.exec()
```

Note that Raspberry Pi currently uses a static library of Qt (up to V6.4.2) which does not report ‘pressure’ correctly. A workaround is to create a python virtual environment to allow the upgrade to newest version of Qt.

Code example: Javascript html (running under Microsoft Edge and Google Chrome. Firefox interprets touch as a mouse click)

```
1       Touch anywhere with mouse or touchscreen or touchpad
2
3       touch:
```