USER GUIDE

gen4-FTDI SERIES

gen4-FT812-43T
gen4-FT812-50T
gen4-FT812-70T
gen4-FT813-43CT-CLB
gen4-FT813-50CT-CLB
gen4-FT813-70CT-CLB

gen4-FTDI Display Series

4D SYSTEMS
MAKING HUMAN INTELLIGENCE SMARTER

gen4-FTDI Series

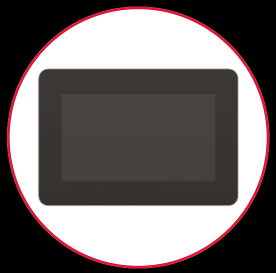
SCREEN SIZE		RESOLUTION	TOUCH TYPE			FOR ARDUINO	FOR RASPBERRY PI
inches	mm		Non-Touch	Resistive	Capacitive		
3.2	81.28	240 x 320		◆			◆
3.5	88.90	320 x 480		◆		◆	◆
4.3*	109.22	480 x 272		◆	◆	◆	◆
5.0*	127.00	800 x 480		◆	◆	◆	◆
7.0*	177.80			◆	◆	◆	◆

*Also available in Cover Lens Bezel (CLB) version.

VARIANTS:
Resistive Touch (T)
Capacitive Touch (CT)
Capacitive Touch with Cover Lens Bezel (**CT-CLB**)

This user guide will help you started using the gen4-FTDI display modules along with the WorkShop4 IDE. It also includes a list of essential project examples and application notes.

What's In The Box



gen4-FT812-43CT-CLB

gen4-Breakout

15-way to 30-way Flat Flex Cable (FFC)

Supporting documents, datasheet, CAD step models and application notes are available at www.4dsystems.com.au

Contents

Introduction

This User Guide is an introduction to becoming familiar with the gen4-FT812/FT813. This manual should be treated only as a useful starting point and not as a comprehensive reference document. Refer to Application Notes for a list of all the detailed reference documents.

In this User Guide, we will briefly focus on the following topics:

- Hardware and Software Requirements
- Connecting the Display Module to the Host
- Getting Started with Simple Projects
- Reference Documents

The **gen4-FT81X-XXT/CT-CLB** is part of embedded SPI display manufactured by 4D Systems. The module features a 4.3", 5.0", or 7.0" colour TFT LCD display, with resistive or capacitive touch. It is powered by the FT812/FT813 Video Engine, which targets high quality graphics displays with Widget support, designed to offload the Host Processor and provide a variety of graphics features.

Intelligent display modules are low-cost embedded solutions used in various applications in the medical, manufacturing, military, automotive, home automation, consumer electronics, and other industries. In fact, there are very few embedded designs on the market today that do not have a display. Even many consumer white goods and kitchen appliances incorporate some form of display. Buttons, rotary selectors, switches and other input devices are being replaced by more colourful and easier-to-use touch screen displays in industrial machines, thermostats, drink dispensers, 3D printers, commercial applications - virtually any electronic application.

System Requirements

The following sub-sections discuss the hardware requirements for this manual.

1 Hardware Requirements

1.1 Intelligent Display Module and Accessories

The gen4-FT812/FT813 (adaptor board and flat flex cable) are included in the box, delivered to you after your purchase from our website or through one of our distributors. Please refer to the section "What's in the Box" for images of the display module and its accessories.

1.2 Host Controller

The gen4-FT812/FT813 is a display module which is driven by a host controller. The host controller will send the commands using SPI communication. It is entirely up to the user to select their host controller and use it to drive the display.

NOTE: *The host controller is a device such as an Arduino or a Raspberry Pi. These devices are not supplied by 4D Systems.*

2 Software Requirements

The gen4-FT81x range is NOT compatible with 4D Systems Workshop4 IDE, it requires FTDI tools to be used, which is found on the FTDI/Bridgetek website.

Website: https://brtchip.com/ft81x

Datasheet: https://brtchip.com/wp-content/uploads/Support/Documentation/Datasheets/ICs/EVE/DS_FT81x.pdf

Programmers Guide: http://brtchip.com/wp-content/uploads/Support/Documentation/Programming_Guides/ICs/EVE/FT81X_Series_Programmer_Guide.pdf

Examples: http://brtchip.com/SoftwareExamples-eve/

Connecting The Display Module

This section shows the complete instructions for connecting the display to the host controller.

HARDWARE

SOFTWARE

Display Module

FFC Cable

gen4-Breakout

Host Controller

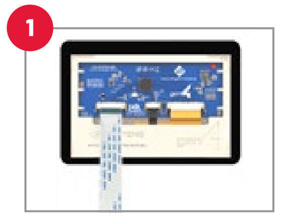
SW

Any Software which can drive your host controller

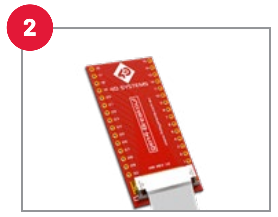
Connection Options

1. Connect one end of the FFC to your module's 30-way ZIF socket with the metal contacts on the FFC facing up on the latch.
2. Connect the other end of the FFC to the 30-way ZIF socket on the gen4-Breakout board with the metal contacts on the FFC facing up on the latch.
3. Connect the pins of the gen4-Breakout to the pins of the Host Controller.

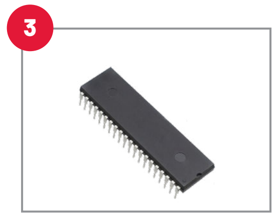
1



2



3



Getting Started With A Simple Project

After successfully connecting the display module to the host controller you are using, you can now start creating a basic application. This section shows how to show a simple keyboard on the display module using the example uploaded on the FTDI chip website.

NOTE: *For the sake of the discussion, the example used on this project is based on the Arduino sketch provided by the FTDI website. With these, the host controller will be the Arduino UNO.*

Follow the following steps below:

1. Download the Example 7 - FT_App_Keyboard(+EVE2) originally from the FTDI chip website.
2. Extract the file and navigate to App_Keyboard > Project > Arduino > App_Keboard. Open the **App_Keyboard.ino**.
3. Edit the Platform.h and add the following configuration needed by the display to run the application.

(a.) On line 41 of the Platform.h - /* Module specific configurations *, add the display definition that suits your module (or add them all and comment out the ones not being used)

```
#define GEN4FT812_43
//#define GEN4FT813_43
//#define GEN4FT812_50
//#define GEN4FT813_50
//#define GEN4FT812_70
//#define GEN4FT813_70
```

and uncomment the display previously defined

```
// #define VM800P43_50
```

(b.) Add the display definition by copying the code below and pasting it on the Platform.h around line 49 (after the defines above). Paste the ones you need, or all of them, the defines above set which of the next section is active.

```
#ifdef GEN4FT812_43
#define DISPLAY_RESOLUTION_WQVGA                (1)
#define FT812_ENABLE                            (1)
#define ENABLE_SPI_SINGLE                       (1)
#define ARDUINO_ATMEGA328P_I2C                  (1)
#define RTC_PRESENT                             (0)
#define FT800_CS                                (9)
#define SDCARD_CS                               (10)
#define FT800_INT                               (7)
#define FT800_PD_N                              (8)
#define ARDUINO_PRO_SPI_CS                      FT800_CS
#define ARDUINO_PLATFORM_SPI                    (1)
#define ARDUINO_PLATFORM_COCMD_BURST            (1)
#define RESISTANCE_THRESHOLD                    (1800)
```

```
#ifdef GEN4FT813_43
#define DISPLAY_RESOLUTION_WQVGA               (1)
#define FT813_ENABLE                           (1)
#define ENABLE_SPI_SINGLE                      (1)
#define ARDUINO_ATMEGA328P_I2C                 (1)
#define RTC_PRESENT                            (0)
#define FT800_CS                               (9)
#define SDCARD_CS                              (10)
#define FT800_INT                              (7)
#define FT800_PD_N                             (8)
#define ARDUINO_PRO_SPI_CS                     FT800_CS
#define ARDUINO_PLATFORM_SPI                   (1)
#define ARDUINO_PLATFORM_COCMD_BURST           (1)
#define RESISTANCE_THRESHOLD                   (1800)
#endif

#ifdef GEN4FT812_50
#define DISPLAY_RESOLUTION_WVGA                (1)
#define FT812_ENABLE                           (1)
#define ENABLE_SPI_SINGLE                      (1)
#define ARDUINO_ATMEGA328P_I2C                 (1)
#define RTC_PRESENT                            (0)
#define FT800_CS                               (9)
#define SDCARD_CS                              (10)
#define FT800_INT                              (7)
#define FT800_PD_N                             (8)
#define ARDUINO_PRO_SPI_CS                     FT800_CS
#define ARDUINO_PLATFORM_SPI                   (1)
#define ARDUINO_PLATFORM_COCMD_BURST           (1)
#define RESISTANCE_THRESHOLD                   (1800)
#endif

#ifdef GEN4FT813_50
#define DISPLAY_RESOLUTION_WVGA                (1)
#define FT813_ENABLE                           (1)
#define ENABLE_SPI_SINGLE                      (1)
#define ARDUINO_ATMEGA328P_I2C                 (1)
#define RTC_PRESENT                            (0)
#define FT800_CS                               (9)
#define SDCARD_CS                              (10)
#define FT800_INT                              (7)
#define FT800_PD_N                             (8)
#define ARDUINO_PRO_SPI_CS                     FT800_CS
#define ARDUINO_PLATFORM_SPI                   (1)
#define ARDUINO_PLATFORM_COCMD_BURST           (1)
#define RESISTANCE_THRESHOLD                   (1800)

#ifdef GEN4FT812_70
#define DISPLAY_RESOLUTION_WVGA                (1)
#define FT812_ENABLE                           (1)
#define ENABLE_SPI_SINGLE                      (1)
#define ARDUINO_ATMEGA328P_I2C                 (1)
#define RTC_PRESENT                            (0)
#define FT800_CS                               (9)
#define SDCARD_CS                              (10)
#define FT800_INT                              (7)
#define FT800_PD_N                             (8)
#define ARDUINO_PRO_SPI_CS                     FT800_CS
#define ARDUINO_PLATFORM_SPI                   (1)
#define ARDUINO_PLATFORM_COCMD_BURST           (1)
#define RESISTANCE_THRESHOLD                   (1800)
#endif

#ifdef GEN4FT813_70
#define DISPLAY_RESOLUTION_WVGA                (1)
#define FT813_ENABLE                           (1)
#define ENABLE_SPI_SINGLE                      (1)
#define ARDUINO_ATMEGA328P_I2C                 (1)
#define RTC_PRESENT                            (0)
#define FT800_CS                               (9)
#define SDCARD_CS                              (10)
#define FT800_INT                              (7)
#define FT800_PD_N                             (8)
#define ARDUINO_PRO_SPI_CS                     FT800_CS
#define ARDUINO_PLATFORM_SPI                   (1)
#define ARDUINO_PLATFORM_COCMD_BURST           (1)
#define RESISTANCE_THRESHOLD                   (1800)
#endif
```

(c.) Below this in Platform.h add the configuration for the display in the /* Custom configuration set by the user */

The completed code should be like this:

```
#if (!defined(VM800P43_50) && !defined(VM800P35)
&&!defined(VM801P43_50) &&!defined(VM800B43_50)
&&!defined(VM800B35) &&!defined(VM801B43_50)
&&!defined(GEN4FT811_43) &&!defined(GEN4FT812_43)
&&!defined(GEN4FT813_43) &&!defined(GEN4FT812_50)
&&!defined(GEN4FT813_50) &&!defined(GEN4FT812_70)
&&!defined(GEN4FT813_70))
```

4. Save the following changes on the file and upload the program on the Arduino using the Arduino IDE.

NOTE: *This is a modification to an existing FTDI example. For different platforms you will need to study the FTDI documentation and potentially find examples on the FTDI / Bridgetek website. FTDI examples are subject to change at any time, and contact should be made with FTDI / Bridgetek if you have queries about their demos or software.*

Reference Documents

The software support for the FT812/FT813 is provided by the FTDI. If you want to further check the FT81x graphic controller IC, please refer to their website. The datasheet for the module itself can be downloaded from the 4D systems website. Below are the necessary documents to help you get started with the display module.

FT81x Graphics Controller IC

This product page contains the Key Hardware Features, Performance Improvements, Applications Areas, Product Information, and other related products of the FT81x Series.

FT81x Embedded Video Engine Datasheet

This document contains the features of the FT81x Video Engine. It covers the Part Numbers, Block Diagram, Device Pin Out and Signal Description, Function Description, Memory Map, Devices Characteristics and Ratings, and Application Examples of the FT81x Series.

gen4-FT81x Datasheet

This document contains detailed information about the gen4-FT81x display module.

NOTES

GLOSSARY

Hardware

1. **Graphics Controller** – A chip also known as a graphics coprocessor and like a microprocessor ordinarily found on graphics accelerator cards. The graphics controller processes the graphics received by the computer and creates the dots and lines on the screen to generate the picture.

2. **Embedded System** – A programmed controlling and operating system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts.

3. **Microcontroller** – A compact integrated circuit designed to govern a specific operation in an embedded system. A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip.

4. **FFC** – Flexible flat cable, or FFC, refers to any variety of electrical cable that is both flat and flexible. It used to connect the display to a programming adaptor.

5. **Adaptor Board** – Adapter Board is the dedicated interface to connect the Board to the specific hardware application.

6. **Raspberry Pi** – A series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote teaching of basic computer science in schools and in developing countries.

7. **Arduino** – An open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices.

8. **Resistive Touch Panel** – A touch-sensitive computer display composed of two flexible sheets coated with a resistive material and separated by an air gap or microdots.

9. **Serial Peripheral Interface (SPI)** – A synchronous serial communication interface specification used for short distance communication, primarily in embedded systems.

10. **Capacitive Touch Screen** – A control display that uses conductive touch of a human finger or specialized device for input.

Software

1. **Arduino IDE** – The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. It is used to write and upload programs to Arduino compatible boards, but also, with the help of 3rd party cores, other vendor development boards.

2. **GUI** – A form of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces, typed command labels or text navigation.

Visit our website at:	www.4dsystems.com.au
Technical Support:	www.4dsystems.com.au/support
Sales Support:	sales@4dsystems.com.au