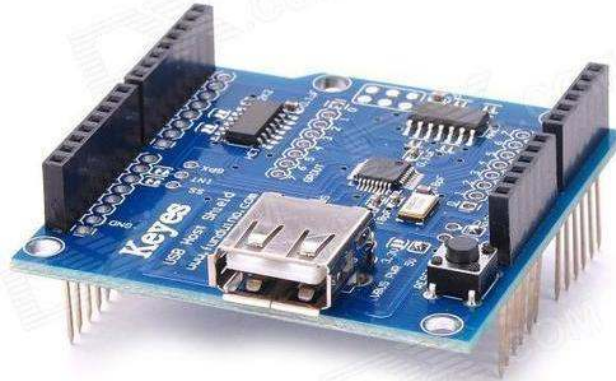


## **USB Host Shield for Arduino**



There is a huge variety of shields that are available which can be stacked on top of an Arduino. Often there are more than one manufacturer for a single type of shield itself. In a way this is good, because as a user you are going to have multiple options. But it becomes a problem when the shield is pretty complex (like the USB Host Shield) and you have to use a library and the shields are not compatible with each other. One such shield which has many incompatible versions is USB Host Shield and in this post I am going to tell you how you can select the proper shield and also the changes that you have to do to make even the incompatible shields work with the library.

### **What is an USB Host Shield?**

Before we start, let's first understand what is an USB Host Shield. It is a shield which provides USB Host support for Arduino.

So, what is USB Host support? The USB protocol defines two types of devices. One is called the host (or server) and the other one is called peripheral (client). The Host device controls the peripheral device and also provides power to it. When you connect any USB device like a mouse or a keyboard to your computer, your computer acts as the host and controls (or polls) the client device (keyboard or mouse or even an Arduino). For a successful communication to happen using USB protocol, you need at least one of the device to be the host, which means that you cannot connect two keyboards together and expect them to communicate with each other.

The USB Host shield has a separate chip (usually Max3421E), which provides USB Host support. Once you have this shield, your Arduino board can act as USB Host and you can connect other USB devices like keyboard, mouse or even an Android phone and communicate with the device from Arduino itself.

## General Description

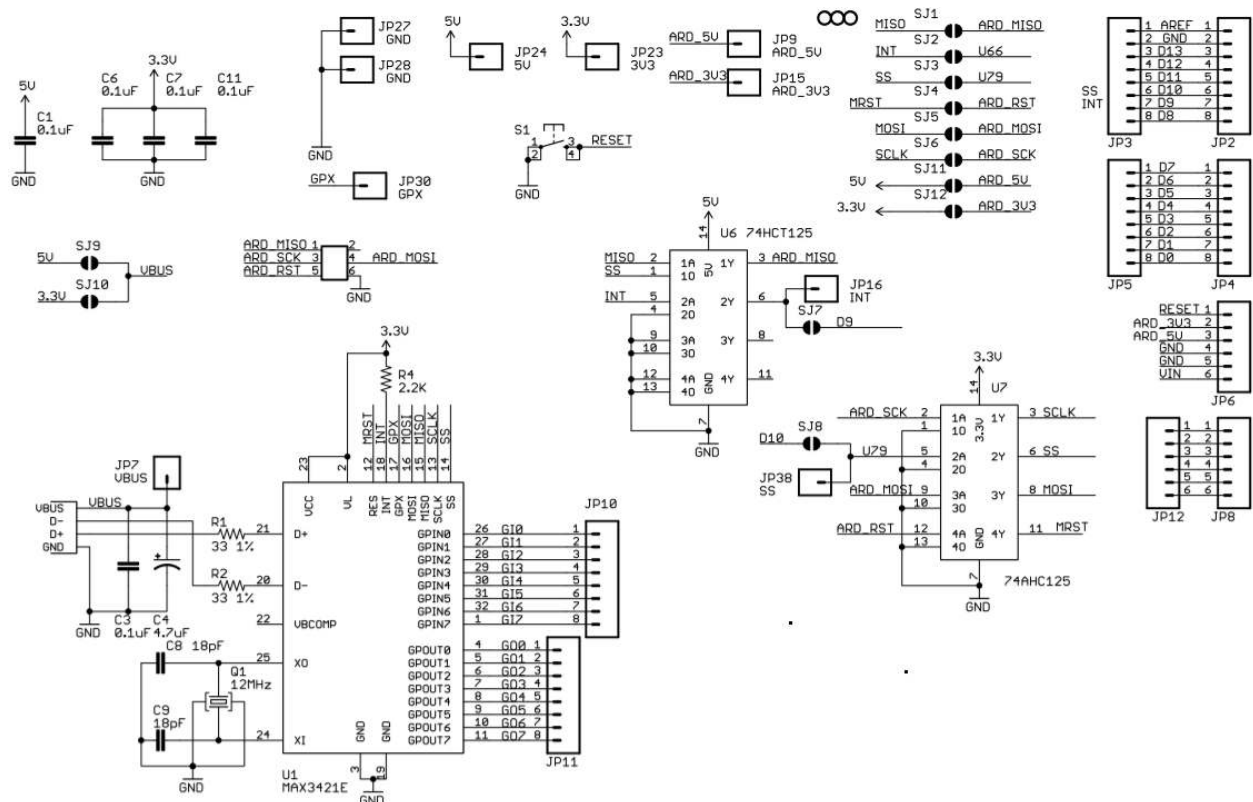
The Keyes USB Host Shield allows you to connect a USB device to your Arduino board. It is based on the MAX3421E, which is a USB peripheral/host controller containing the digital logic and analog circuitry necessary to implement a full-speed USB peripheral or a full-/low-speed host compliant to USB specification rev 2.0.

This is based on revision 2.0 of USB Host Shield. Thanks to new interface layout it is now compatible with more Arduinos - not only UNO and Duemilanove, but also Mega and Mega 2560 work with Standard variant of this shield out of the box. No more SPI re-wiring and code modifications - just solder included stackable connectors (2x3 ICSP connector's female side should be facing down), plug and play!

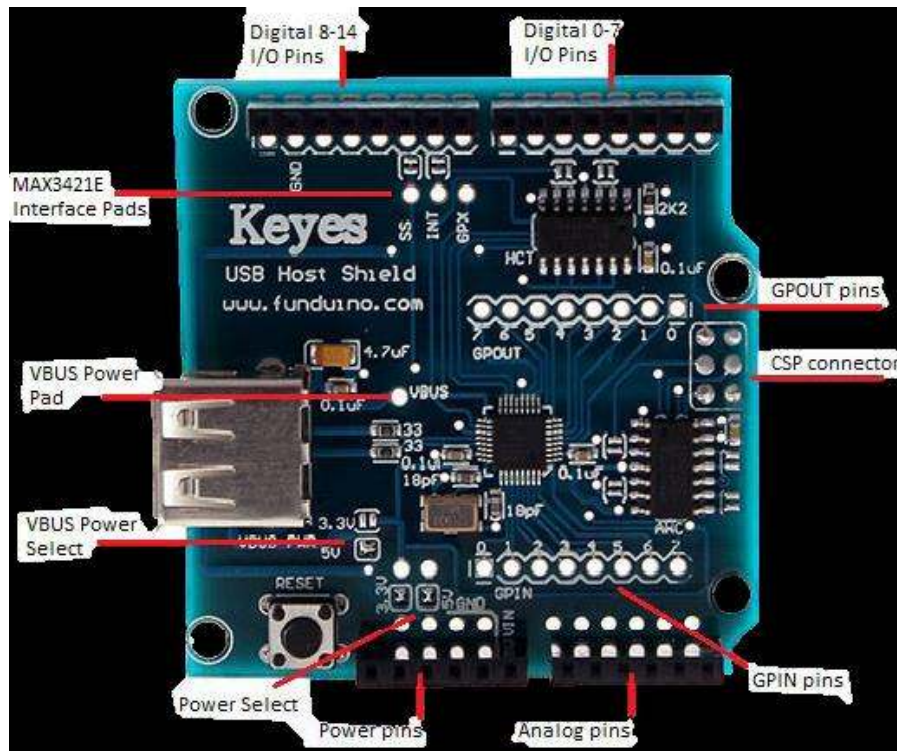
## Specifications

- Works with standard (dual 5/3.3V) and 3.3V-only (for example, Arduino Pro) boards.
- Operates over the extended -40°C to +85°C temperature range
- Complies with USB Specification Revision 2.0 (Full-Speed 12Mbps Peripheral, Full-/Low-Speed 12Mbps/1.5Mbps Host)
- The following device classes are currently supported by the shield:
- HID devices, such as keyboards, mice, joysticks, etc.
- Mass storage devices, such as USB sticks, memory card readers, external hard drives (FAT32 Type File System - Arduino Mega only)

## Schematic



## Shield Overview



- **Power Select** 2 solder jumpers marked “5V” and “3.3V”. They are used for different power configurations. The configuration shown, when both jumpers are closed, is suitable for official Arduinos, such as UNO, Duemilanove, Mega and Mega 2560. See Power Options section for detailed explanation.

- **Power pins** are used to connect to power pins of Arduino board. RESET, 3.3V, 5V and GROUND signals from this connector are used.

- **Analog pins** are not used by the shield. They are provided to simplify mounting and provide pass-through for shields mounted atop of USB Host Shield in a stack.

- **GPIIN pins.** Eight 3.3V general-purpose digital input pins of MAX3421E. They are used primarily to interface with buttons, rotary encoders and such. GPIIN pins can also be programmed as a source of MAX3421E interrupt.

- **ICSP connector** is used by the shield to send/receive data using SPI interface. SCK, MOSI, MISO and RESET signals from this connector are used.

- **GPOUT pins** are eight 3.3V general-purpose digital output pins of MAX3421E. They can be used for many purposes.

- **Digital I/O pins 0-7**, like already mentioned analog pins are not used by the shield and provided only for convenience.

- **Digital I/O pins 8-13.** In this group, the shield in its default configuration uses pins 9 and 10 for INT and SS interface signals. However, standard-sized Arduino boards, such as Duemilanove and UNO have SPI signals routed to pins 11-13 in addition to ICSP connector, therefore shields using pins 11-13 combined with standard-sized Arduinos will interfere with SPI. INT and SS signals can be re-assigned to other pins (see below); SPI signals cannot.

- **MAX3421E interface pads** are used to make shield modifications easier. Pads for SS and INT signals are routed to Arduino pins 10 and 9 via solder jumpers. In case pin is taken by other shield a re-routing is necessary, a trace is cut and corresponding pad is connected with another suitable Arduino I/O pin with a wire. To undo the operation, a wire is removed and jumper is closed

- **VBUS power pad.** This pad is used in advanced power configurations, described in Power Options section.

### How to Test

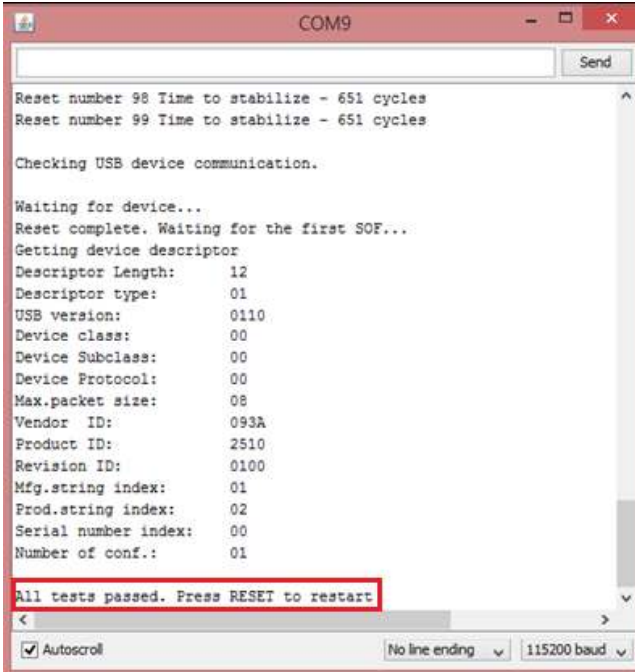
1. Mount the Keyes USB Host Shield into your Arduino.
2. Download USB Host Shield Library & extract it to library folder in your Arduino directory.
3. Open board\_qc then click upload.

You can find this sketch at Arduino IDE File> Examples > USB\_Host\_Shield > board\_qc.

4. Click Serial Monitor and set the baud rate to 115200.
5. When prompted, plug in a USB device to the USB Host Shield's USB port.

```
Checking USB device communication.  
  
Waiting for device...  
Reset complete. Waiting for the first SOF...
```

6. If the shield is fully functional, you should be seeing this Serial Monitor:



```
COM9  
Reset number 98 Time to stabilize - 651 cycles  
Reset number 99 Time to stabilize - 651 cycles  
  
Checking USB device communication.  
  
Waiting for device...  
Reset complete. Waiting for the first SOF...  
Getting device descriptor  
Descriptor Length: 12  
Descriptor type: 01  
USB version: 0110  
Device class: 00  
Device Subclass: 00  
Device Protocol: 00  
Max.packet size: 08  
Vendor ID: 093A  
Product ID: 2510  
Revision ID: 0100  
Mfg.string index: 01  
Prod.string index: 02  
Serial number index: 00  
Number of conf.: 01  
All tests passed. Press RESET to restart
```