coreboot

Test interfaces for coreboot automated distributed test system

www.coreboot.org

v1.0

10 Oct 2013

Acknowledgements:

Thanks to Stefan, Dave, Marc, Peter and the folks at the #coreboot IRC channel for their continuous feedback. Special thanks to Google for promoting development of free and open source software through the Google Summer of Code (GSoC) program. This document is related to a project done for coreboot under GSoC 2013.

Feedback and questions:

IRC: http://www.coreboot.org/IRC

Mailing list: http://www.coreboot.org/Mailinglist

or visit www.coreboot.org to know more about coreboot project

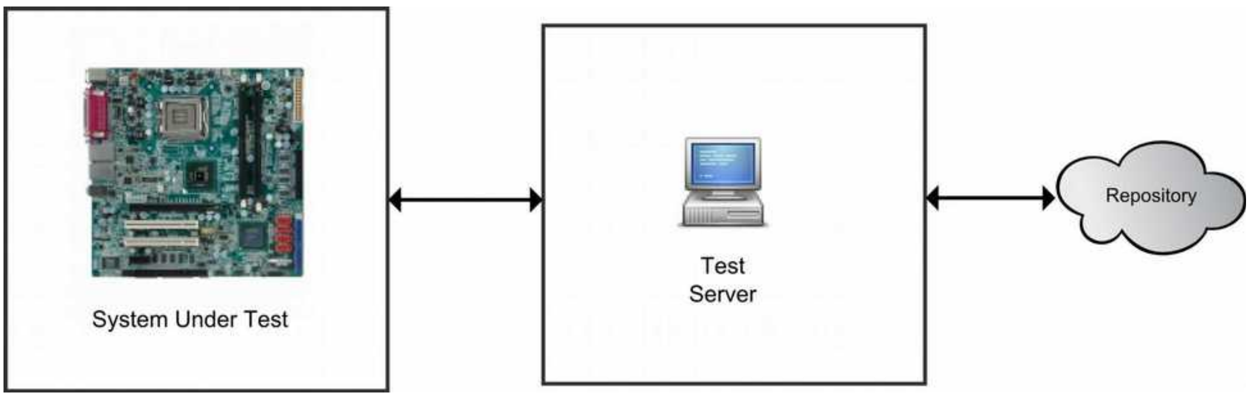
Contents

Change log

Date	Ver.	Author	Changes
10-Oct-13	1.0	Ayush Sagar	Initial

Motivation

Let's say we have a motherboard which is the system under test (SUT) and it's connected in 'some way' to a computer (test server) that performs tests on it. The computer has access to a local or remote coreboot development repository.



A simplistic view of the test set-up

To perform automated testing, it should follow a sequence like the following:

- Power off the SUT
- Get latest untested build from the repository and save it.
- Burn the coreboot test build on the motherboard.
- Power on the SUT, perform basic tests, generate a standardized report file.
- While the SUT is on, dump it's serial port data (if available) on a file.
- Upload/save the test report and serial dump associated with the build in the same or separate repository.

To perform the above tasks there's a need to design the necessary software and hardware interfaces. Further it is desirable that these interfaces are efficient and extensible for different testing needs.

Developing the test interfaces

The test-server<->repository interface is mostly software based, but on the other hand the SUT<->test server interface is more demanding as it also needs some hardware to be designed. There are two devices that have been specially designed to implement the SUT<->test server interface.

1. USB Power Strip: A controllable power strip to control mains power to the SUT.

The initial task is to power off the SUT completely by switching off the mains power. This is a required step to ensure that the SUT can be safely programmed. Powering off also helps to initialize the test conditions.

2. Test Interface Board: a way to talk to the SUT

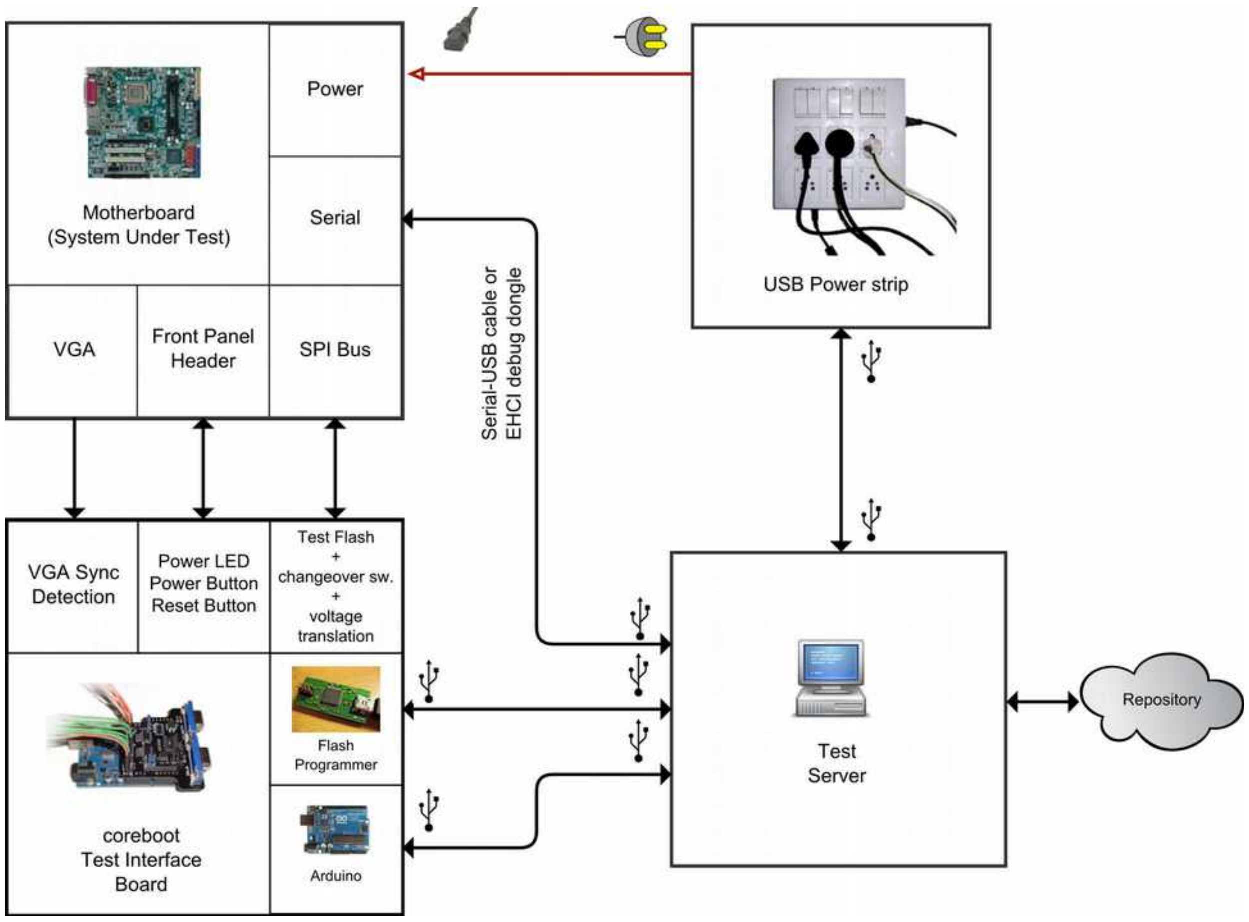
Many PC motherboards do not have In Circuit Programming (ICP) support for its BIOS flash. Although there are workarounds that are specific to each motherboard there is a need for standard and well defined methods. The test interface board provides ICP for SPI flash since it is the preferred flash type in modern boards. There are workarounds for other flash types but those are beyond the scope of this document.

In addition to ICP for SPI flash it also combines the following functions:

- Power/Reset Button control
- Read Power LED pin to know the power state of the board
- Detect video signal presence to know if it's displaying anything
- Read the serial dump - from a serial port or by using the EHCI debug dongle

Possible test set-ups

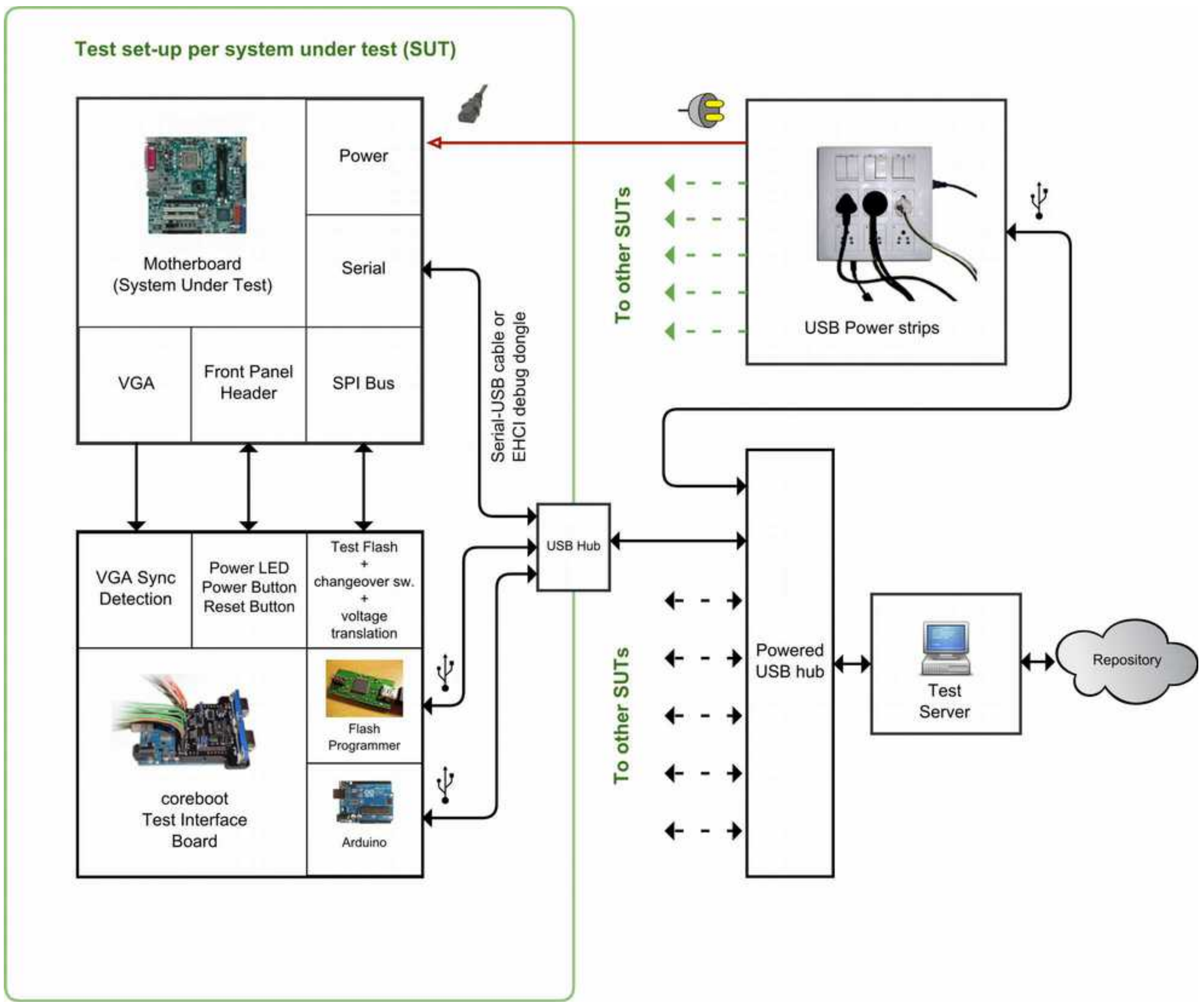
A. Test set-up for single SUT



Test set-up for single SUT

B. Test set-up distributed at test server level

We can extend this system to test multiple SUTs. In general such a system may look like this:


Test set-up per system under test (SUT)
Motherboard (System Under Test)
Power
Serial
VGA
Front Panel Header
SPI Bus
VGA Sync Detection
Power LED Power Button Reset Button
Test Flash + changeover sw. + voltage translation
coreboot Test Interface Board
Flash Programmer
Arduino
Serial-USB cable or EHCI debug dongle
USB Hub
To other SUTs
USB Power strips
Powered USB hub
To other SUTs
Test Server
Repository

Test set-up distributed at the test server level

Notice that this kind of set-up is distributed at test server level. It has a benefit of being less expensive when testing more SUTs at a local level. There are two special requirements in addition to the extra wires and USB hubs:

1. Ability to reliably identify each device uniquely

Each device is required to have a unique serial id string. The test interface board and the USB power strip are based around the Arduino Uno development boards which come with a unique serial id string from the factory. But there are two major road blocks:

- The chosen flash programming devices need to have a unique serial id string otherwise this scheme won't work.
- A way to force flashrom (the open-source flashing software) to select from different devices hasn't been implemented yet. This needs to be done.

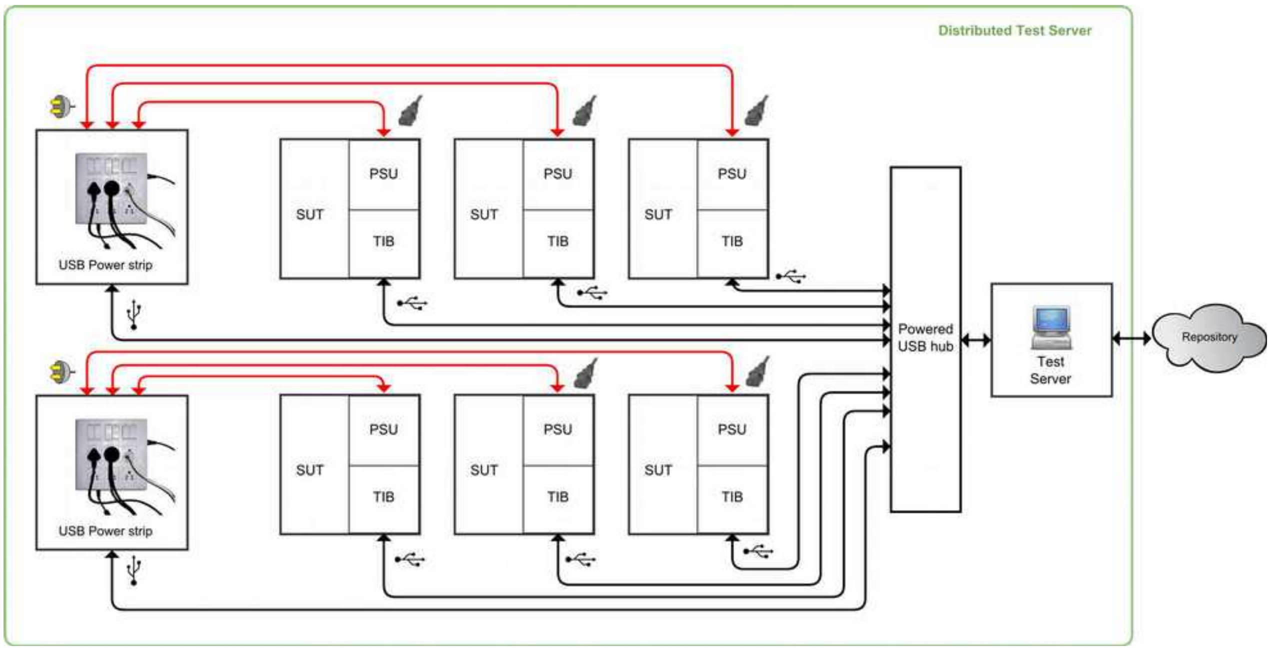
The test logic will be implemented in linux and most modern linux distributions use udev device manager. We can create udev rules to create bindings of these devices to symbolic names based on their USB device descriptors - VID, PID, and serial string.

2. Robust concurrency control

- The test logic must handle operations well.

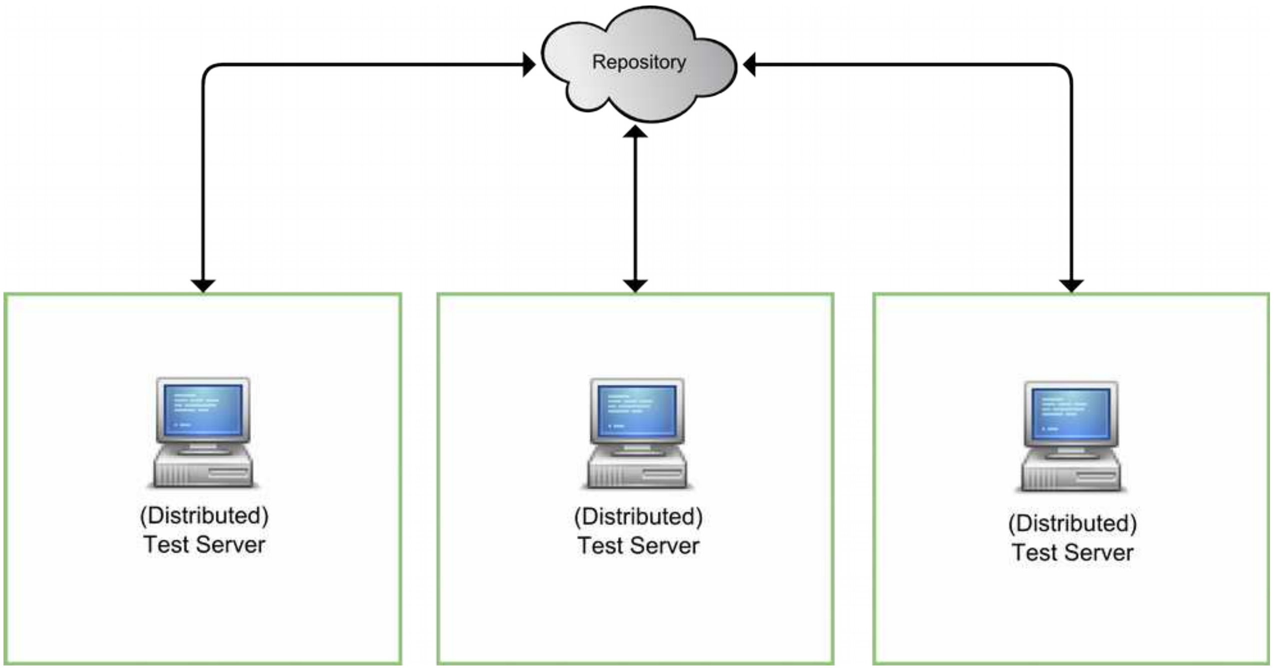
- Further, flashrom will need to be tested for reliability when running its multiple instances.

Here's an example of how this test set-up could look like



An example of test set-up distributed at test server level containing 6 SUT

C. Test set-up distributed at repository level


Repository
(Distributed) Test Server
(Distributed) Test Server
(Distributed) Test Server

Test set-up distributed at repository level

Following the similar trend we can extend our system in a way that multiple test severs can share their repository.

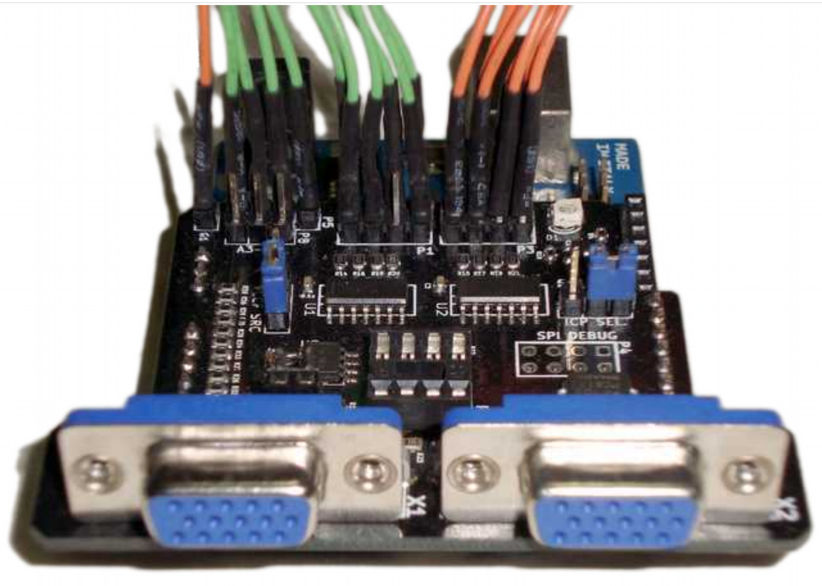
There can be two sub types:

- **C1**: Test set-up with distributed test server and distributed repository.
- **C2**: Test set-up with non-distributed test-server and distributed repository.

The C2 type has a distinct advantage of making the system cost effective at small scale, considering that the test routines are not very CPU intensive and we can use cheap single board computers like raspberry Pi. Moreover, a separate flash programming device is not required because the raspberry Pi has in-built flash programming capability. http://www.flashrom.org/RaspberryPi. It's readily implementable at present. The speed of flashing through raspberry Pi is slow however.

Coreboot Test Interface Board

Overview



Prototype of the coreboot test interface board

The coreboot test interface board (TIB) provides convenient electrical interfaces for programming/re-programming firmware on a PC motherboard and performing basic tests on it.

When combined with an Arduino Uno and suitable test logic on a PC, it can perform automated tests and automated SPI flash programming. The test logic is able to interact with this board using an included command line interface program.

Features

- Provides SPI In-Circuit-Programming (ICP) solution with support for logic level translation in the range 1.65V-5.5V to maximize compatibility.
- Provides a stable and safer alternative to test-clip flashing.
- Augments test-clip flashing and provides a non-invasive method for testing coreboot on a motherboard using the HOLD function on the SPI flash. There’s no need to remove the flash chip from the motherboard or to remove the stock

firmware from it.

- Power and reset button control, power-on detection and video detection when combined with an Arduino Uno or a compatible board
- Test flash IC may be replaced conveniently on the test interface board. Test flash of different voltages can be used. *(Note: Flash V_{CC} must be equal to or lower than V_{CC} of motherboard SPI and programmer device used due to limitations of the translator IC)*
- Protection against damage from electrostatic discharge (ESD) from normal handling on all ports.

The test interface board when coupled with an Arduino and connected to a computer provides the following services:

- Query/set ICP mode: if ICP jumper set to internal.
- Report whether VGA sync is present by measuring vertical and horizontal sync frequencies and checking whether it lies in the acceptable range. A frequency counter has been implemented using the hardware interrupts.
- Query power LED state to know whether the board has been successfully turned on/off.
- Operate Power and Reset buttons.
- Query VCCP, VCCF, VCCM to check for correct operation of the system. It uses statistical mode filter to filter out intermittent behavior of the ADC and the readings are referenced to the Atmega’s internal voltage reference instead of the power supply. So it gives accurate voltage reading reliably even with a single query.
- Query A3, A4, A5 analog inputs. These are extra ADC input for measuring any voltage in range 0 – 24.5VDC for Arduino Uno/Duemilanove and 0 – 56.5 VDC for Arduino Leonardo

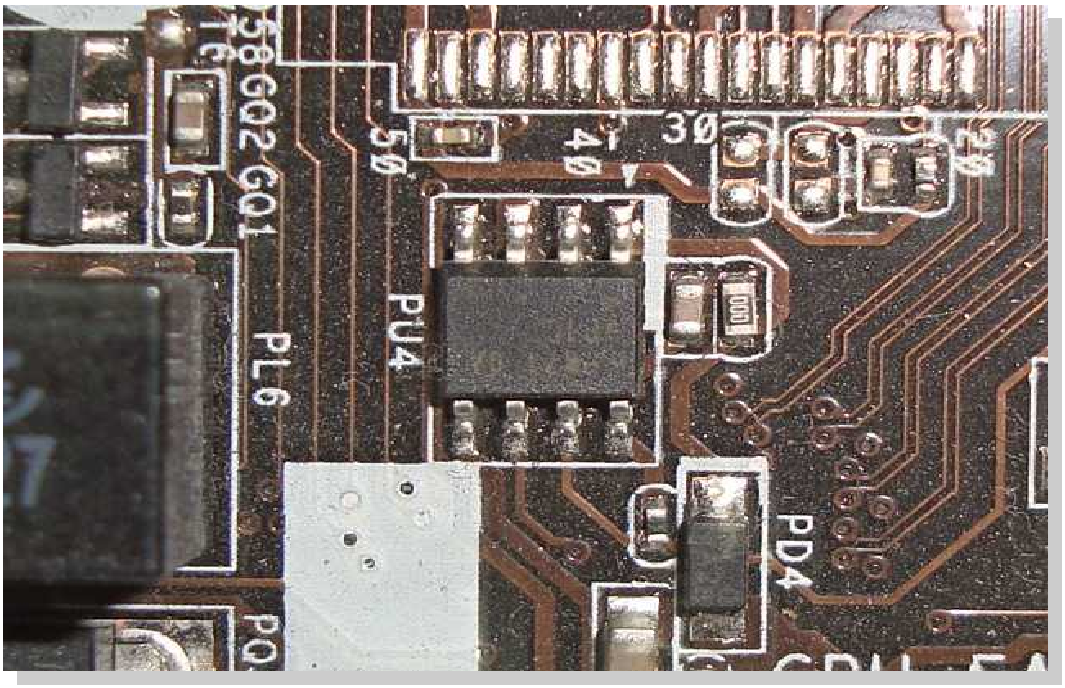
Usage

Connecting to motherboard's SPI bus

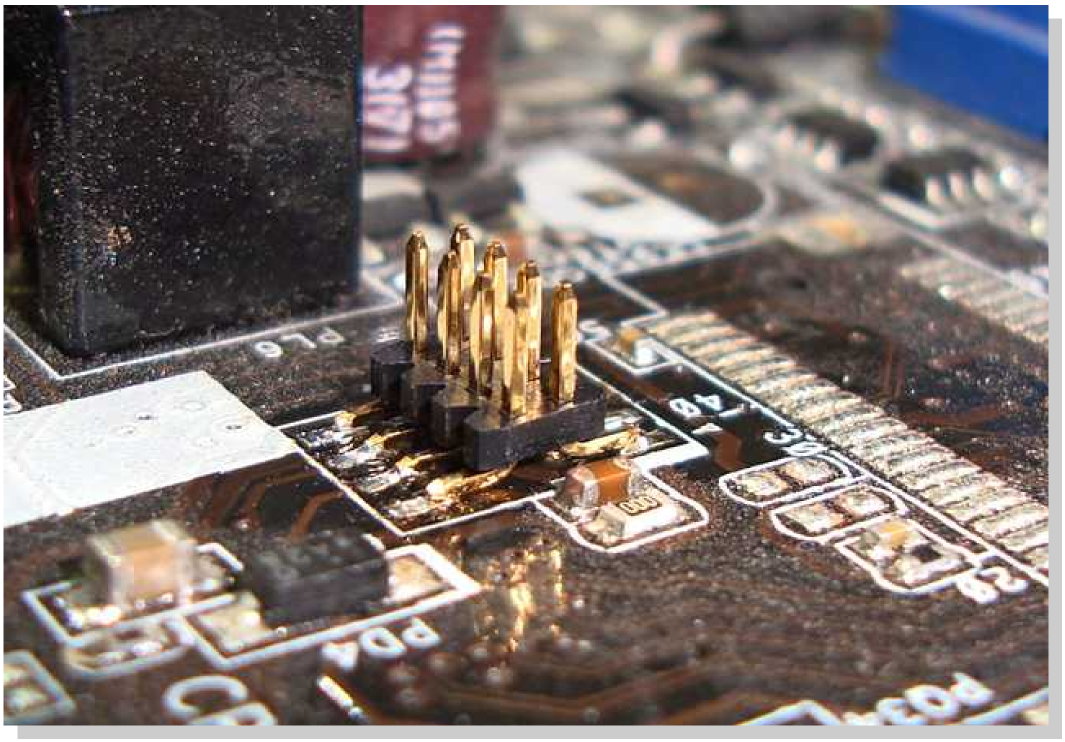
There are two ways to connect a motherboard's SPI bus to the test interface board. First is the *fixed method* that requires de-soldering the flash from the motherboard if it doesn't have a socket and soldering a male SMD header on its place to make an easy connection with the test interface board. Second is the test clip method which is non-invasive. It requires connecting a test-clip over the flash IC and enabling the SPI hold on the flash so that the motherboard talks only to the test flash. Here is a comparison:

	Fixed method	**Test clip method**
Ease	Requires de-soldering flash chip and then soldering a header on its footprint if it’s a socket-less board	Quick and easy
Long term convenience	Convenient. Electrically and mechanically more stable	Less convenient in long term. Not stable.
Suitability	When performing repeated testing over a longer time on the same board	Good for performing quick tests and when a non-invasive method is required.

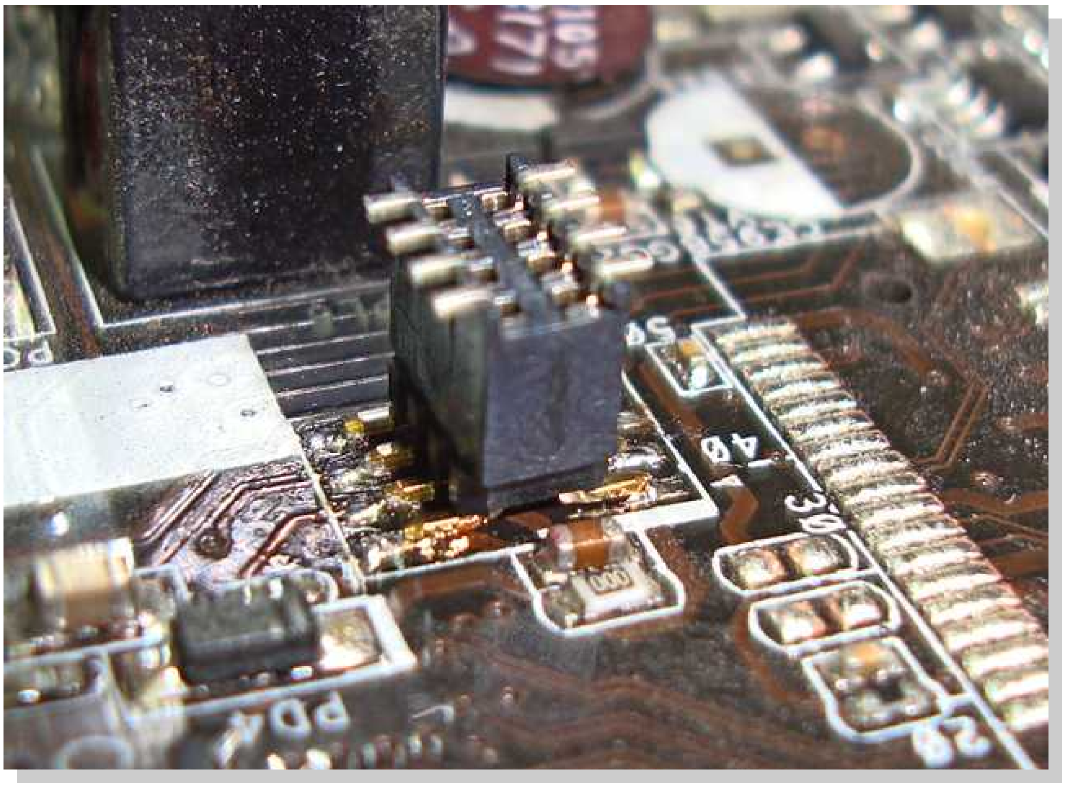
Quick instructions to connect using fixed method

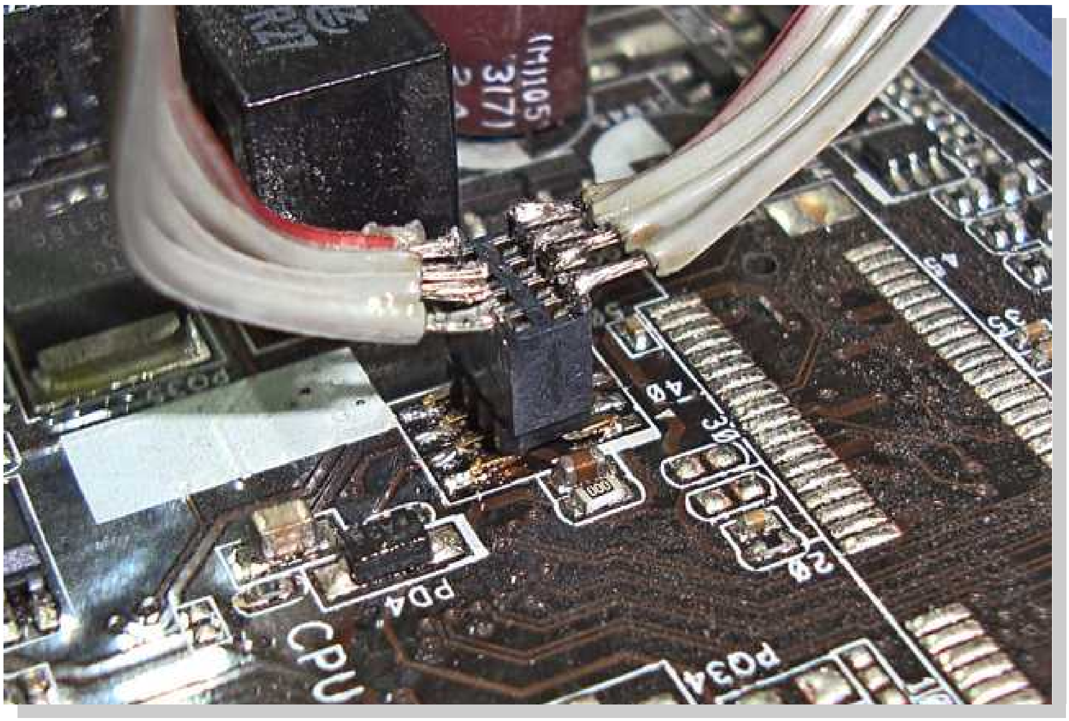


1. Make sure the board is disconnected from power source. Identify the SPI flash



2. Once identified de-solder it and solder a compatible header on it.


3. Plug in a complementary receptacle for the header.


4. The wires coming from MBD SPI port on the test interface board are soldered to a complementary receptacle is used to make connections between the header and the test interface board. Wires should be as short as possible.

Here’s table of recommended headers for different SPI flash IC footprints.

Description	Generic part name	Recommended part
Header for SO8	1.27x1.27MM, SMT, Vertical, Header, 8 Pin	FCI 20021121-00008C4LF
Receptacle for SO8 Header	1.27x1.27MM, SMT, Vertical, Receptacle, 8 Pin	FCI 20021321-00008C4LF
Header for SO16	1.27x1.27MM, SMT, Vertical, Header, 16 Pin	FCI 20021121-00016C4LF
Receptacle for SO16 Header	1.27x1.27MM, SMT, Vertical, Receptacle, 16 Pin	FCI 20021321-00016C4LF

Some motherboards have a DIP8 socket. This can be even more convenient and male-female jumper wires can be used to make connections.

Direct soldering of wires to the footprint is not recommended. The stress from an accidental pull of the wire can peel off the solder pads and damage the motherboard irreversibly.

Test-clip method

An SO8 IC test clip from Pomona electronics

The test clip method relies on the fact that the SPI flash chips have a #HOLD pin to mute its communication and make it invisible on the SPI bus. When HOLD is enabled the MISO line goes to high impedance state. This allows us to establish communication between the motherboard’s south-bridge and the test flash on TIB.

To activate hold, the #HOLD pin needs to be connected to ground since it’s an active-low pin. To prepare for this method you must ensure the following:

1. Most SPI flash support the hold feature but it is recommended to ensure that the particular chip used on your motherboard supports it.
2. Use an ohmmeter to measure resistance between the #HOLD pin and VCC pin of the motherboard flash IC. There can be two cases:
   - Case I: It measures a high resistance in order of kilo-ohms.

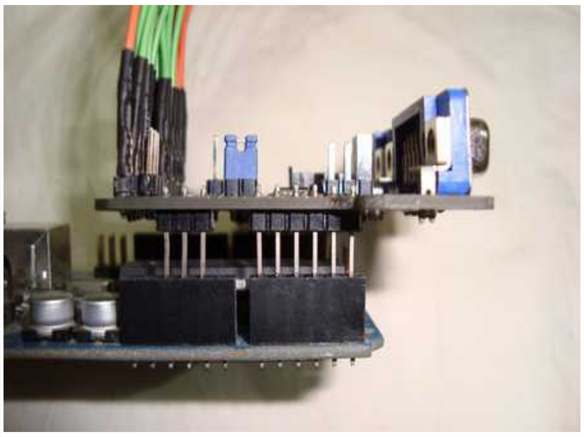
     This is fine, the #HOLD pin can be safely connected to ground.
   - Case II: It indicates a short circuit. You may not be able to use the test-clip method because connecting the #HOLD pin to ground will effectively create a short circuit between the motherboard's VCC and ground and this will damage the board.

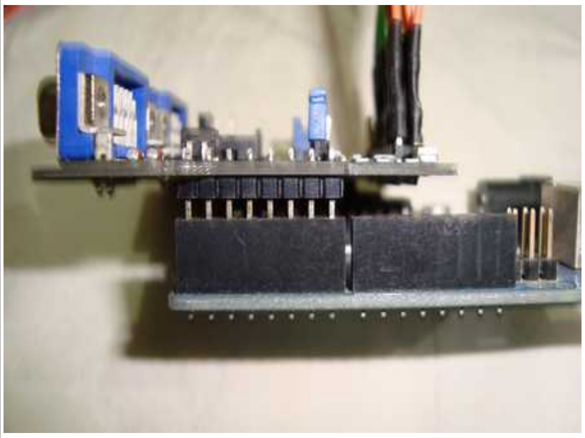
If this method is being used for repeated tests over long term it is advisable to use a glue gun to put hot-melt adhesive between the board and test clip. Ensure that the glue doesn't stick on any of the components on the board.

Direct soldering of wires to the footprint is not recommended. The stress from an accidental pull of the wire can lift off the solder pads and damage the motherboard.

Connecting an Arduino Uno

Arduino Uno, Duemilanove, Leonardo and their clones can work with this board. Arduino Uno is recommended as it comes with a unique USB serial id. USB serial id helps to uniquely identify a device when multiple of the same are connected to a computer.


Test interface board partially inserted into an Arduino in correct alignment. Shows power and analog ports.


Test interface board partially inserted into an Arduino in correct alignment. Shows digital port

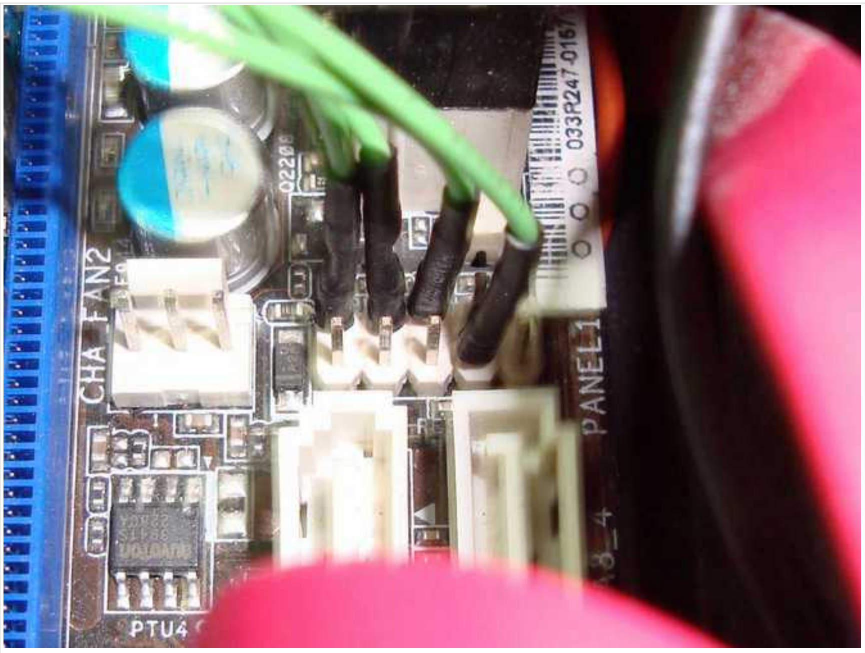
Take care when aligning and connecting the TIB to the Arduino. P7 of the TIB should be inserted to D0-D7 of the Arduino and P9 should be inserted to A0-A5 of the Arduino.

Connecting to motherboard front panel header

Connecting to power button and reset button inputs

The power and reset button pins in the motherboard are inputs that have been pulled up internally. Shorting the input to ground activates the function. In each pin pair for power and reset, one of the pin is for input and the other pin is ground. You need to connect the

TIB's output to the input pin not the ground one. To identify try shorting to ground through a resistor (220 ohms to 1K) one of the pins to ground. If the button activates it's the input, otherwise it's ground.



Test interface board connected to front panel header of a motherboard

The power LED signal is read by the TIB through an optocoupler within the board which is not used for opto-isolation but to emulate an LED. This is needed since the LED output voltage can be less than V_{IH} of Arduino's digital input in some cases, so direct connection without an opto-coupler may not work.

The +ve and -ve terminals are marked on the header label, in case it's not given you can try connecting either way to check which way it works.

Power LED readout is an important function. If power LED output is not available, use a voltage rail such as +5V or +12V from the ATX power supply and connect a resistor in series. The resistance required will be given by the formula:

$R = 50*V_{IN} – 60$.

(derived using the voltage divider rule and the input operating point of PC817 opto-coupler which is 1.2V, 20mA)

Connecting to Video outputs

The test interface board have a VGA pass through. The VGA output can be connected to one of the ports. The other port is optional and could be used to connect it to a monitor without altering the test set-up.

In case the motherboard has a different video interface like HDMI, DVI or DisplayPort use a VGA converter. Inexpensive VGA converters are available to buy on internet.

The test interface board detects whether a VGA sync signal is present or not. In other words, it detects whether a connected monitor would have its status LED green (active) or amber (inactive) at a given time.

Command Line Interface

Use the command line interface program TIB.py (python script) to perform the test operations with the TIB.

DIY assembly

You can order the PCB by sending the gerber files to a PCB fabrication service.

Order the following components and solder them.

Skip the following components if you need only the SPI ICP interface functionality:

R1, R2, R3...

Skip the following components if you need all other test functionality but not the SPI ICP interfacing/

For people new to soldering:

The minimum tools required for building this is a 10-15W micro-soldering iron, no-clean flux, a solder wire with less than 0.3mm cross-section diameter and a pair of tweezers.

Melt solder to one of the pads for every component. Use tweezers to correctly place the component and then solder the terminal where the pad has the added solder on it to fix the components. Now it will be easier to solder the other terminals.

Using as standalone In-Circuit-Programmer

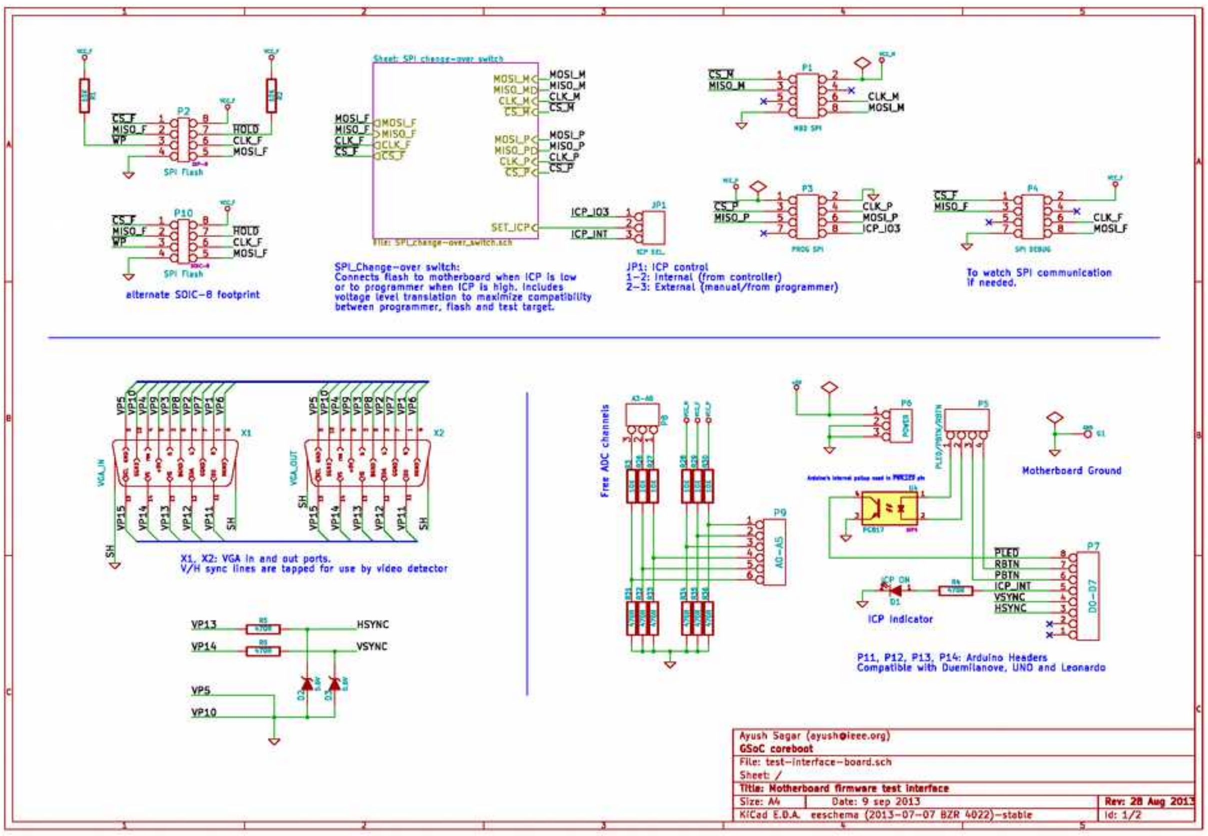
In many cases this board will be used for its SPI features. In this case, the non-relevant components on the PCB can be omitted to save cost and assembly effort.

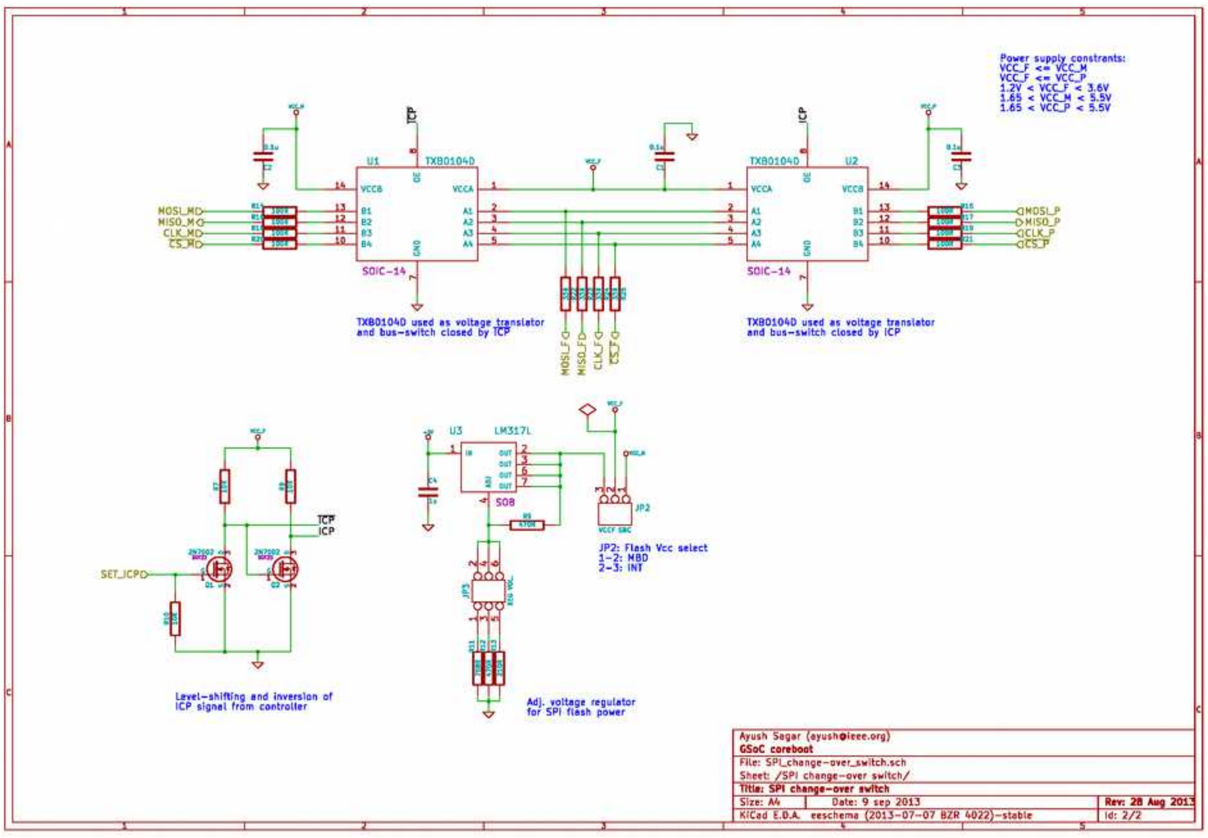
You will however need to connect a 5V supply to P6 which normally takes its power from an Arduino.

Arduino Sketch

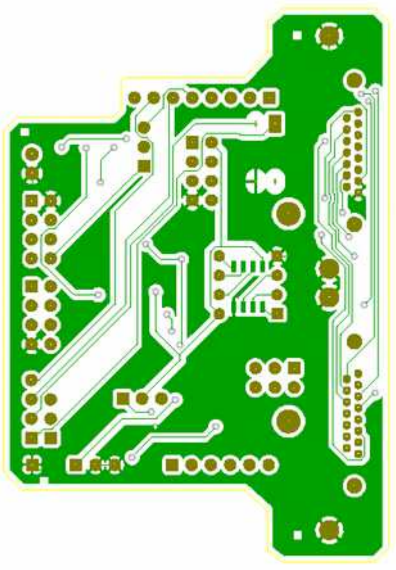
Burn the arduino sketch using Arduino IDE. Make sure to select the correct serial port and correct board type.This was tested to work on Arduino IDE v1.0.3 as well as v1.0.5

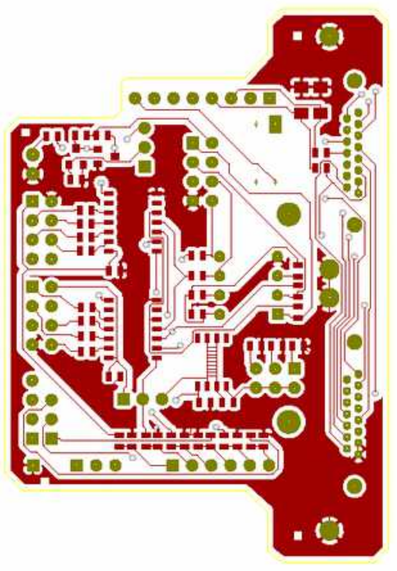
Schematics

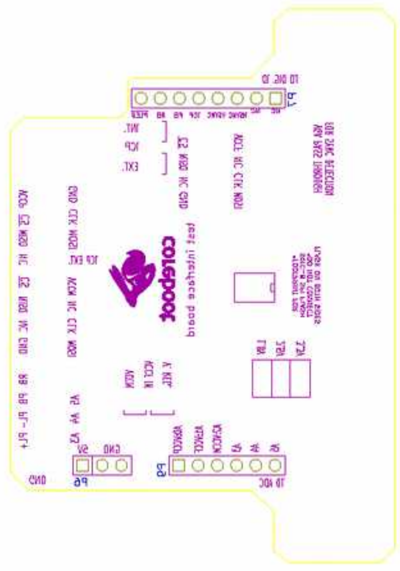

SPI Flash
alternate SOIC-8 footprint
Sheet: SPI change-over switch
File: SPI_change-over_switch.sch
SPI_Change-over switch:
Connects flash to motherboard when ICP is low
or to programmer when ICP is high. Includes
voltage level translation to maximize compatibility
between programmer, flash and test target.
JP1: ICP control
1-2: Internal (from controller)
2-3: External (manual/from programmer)
MBD SPI
PROG SPI
SPI DEBUG
To watch SPI communication
if needed.
X1, X2: VGA in and out ports.
V/H sync lines are tapped for use by video detector
HSYNC
VSYNC
Free ADC channels
A0-A5
Motherboard Ground
ICP Indicator
D0-D7
P11, P12, P13, P14: Arduino Headers
Compatible with Duemilanove, UNO and Leonardo
Ayush Sagar (ayush@ieee.org)
GSoC coreboot
File: test-interface-board.sch
Sheet: /
Title: Motherboard firmware test interface
Size: A4
Date: 9 sep 2013
Rev: 28 Aug 2013
KiCad E.D.A. eeschema (2013-07-07 BZR 4022)-stable
Id: 1/2

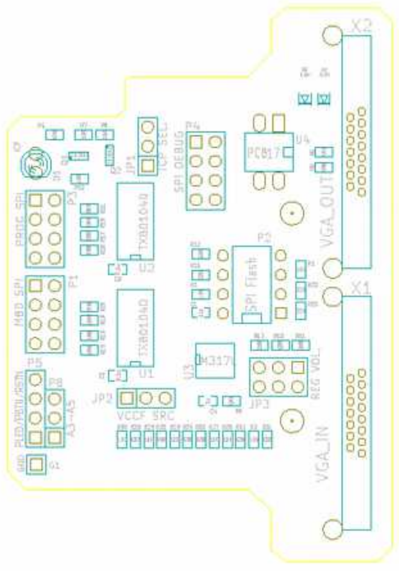

Power supply constrants:
VCC_F <= VCC_M
VCC_F <= VCC_P
1.2V < VCC_F < 3.6V
1.65 < VCC_M < 5.5V
1.65 < VCC_P < 5.5V
U1 TXB0104D
U2 TXB0104D
SOIC-14
TXB0104D used as voltage translator
and bus-switch closed by ICP
MOSI_M
MISO_M
CLK_M
CS_M
MOSI_P
MISO_P
CLK_P
CS_P
MOSI_F
MISO_F
CLK_F
CS_F
U3 LM317L
SO8
JP2: Flash Vcc select
1-2: MBD
2-3: INT
SET_ICP
2N7002
Level-shifting and inversion of
ICP signal from controller
Adj. voltage regulator
for SPI flash power
Ayush Sagar (ayush@ieee.org)
GSoC coreboot
File: SPI_change-over_switch.sch
Sheet: /SPI change-over switch/
Title: SPI change-over switch
Size: A4
Date: 9 sep 2013
Rev: 28 Aug 2013
KiCad E.D.A. eeschema (2013-07-07 BZR 4022)-stable
Id: 2/2

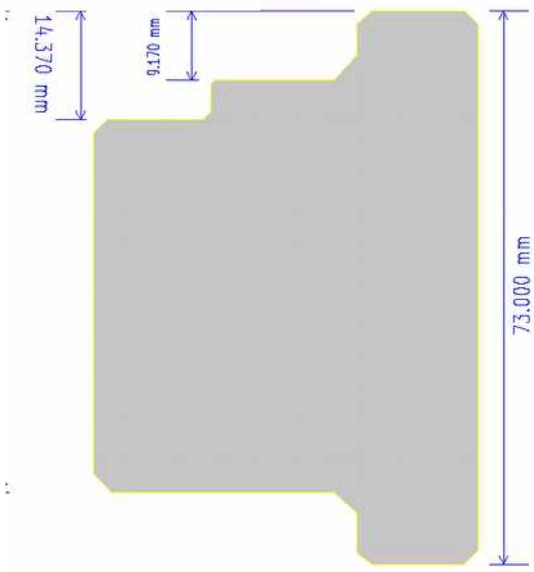
PCB layout



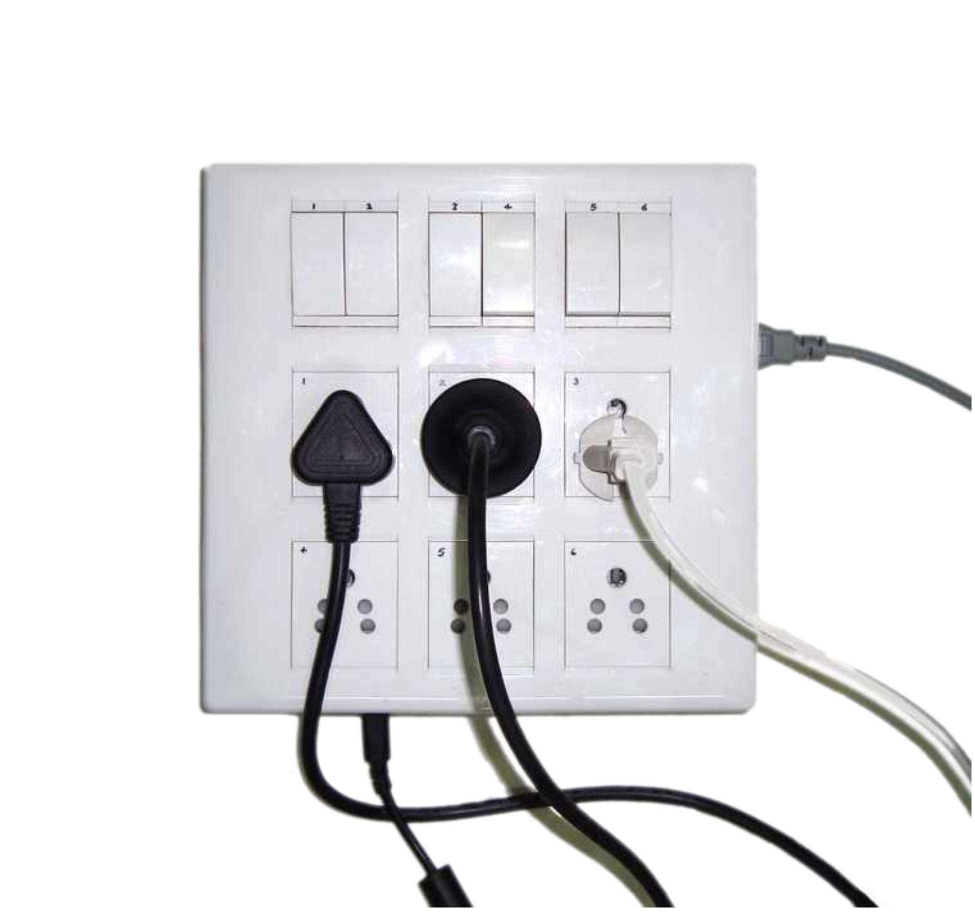





X2
X1
VGA_OUT
VGA_IN
PC817
TXB0104D
TXB0104D
SPI Flash
LM317
SPI DEBUG
PROG SPI
MBD SPI
VCCF SRC
REG VOL
JP1
JP2
JP3
P1
P2
P3
P4
P5
U1
U2
U3
U4


14.370 mm
9.170 mm
73.000 mm

Overview



Prototype of the Coreboot USB power strip (a box actually) made with locally available parts. 220V Indian sockets used.

The Coreboot USB power strip is a convenient and easy to construct means to control mains power to the SUTs. It is USB powered and can be controlled using a command line interface. It works as a norma

Usage

Schematics

Construction