INDUSTRIAL PROJECT: UTIL LABS GENERIC TEST JIG FRAMEWORK

A Dissertation

by

JACOB JACOBUS GREEFF

209329085

Submitted in partial fulfillment of the requirements for the degree

BACHELOR TECHNOLOGIAE

in

Electrical Engineering

Department of Electrical Engineering

Faculty of Engineering and the Built Environment

TSHWANE UNIVERSITY OF TECHNOLOGY

Supervisor: R Aylward

Co-Supervisor: J Olwagen

Tshwane

July 2011

DECLARATION

“**I** hereby declare that this dissertation submitted for the degree Bachelor Technologiae at the Tshwane University of Technology, is my own original unaided work and has not previously been submitted to any other institution or higher education. I further declare that all sources cited or quoted are indicated and acknowledge by means of a comprehensive list of references”**.**

_____

JJ Greeff

ACKNOWLEDGEMENTS

I would like to sincerely thank those who made this thesis possible, specifically my wife Carol who gave me the moral support I required to complete it in time. My two supervisors Mr. Ron Aylward and Mr. Jan Olwagen who helped me with the research material and guiding me to the end result. I also would like to make special reference to Mr. Joe Paul who is the CEO of Util Labs who was willing to allow me to take a chance on the test equipment and let me have free reign creatively in the production department, as well as Mr. Cobus Greyling who helped out with the mechanical design and assembly for the mechanical jig components.

Without the cooperation of the people above, this thesis would not have been possible.

ABSTRACT

AUTHOR: Jacob Jacobus Greeff

DEGREE: Btech: Electrical Engineering

TITLE: Industrial Project: Util Labs Generic Test Jig Framework

UNIVERSITY: Tshwane University of Technology

FACULTY: Faculty of Engineering and the Built Environment

SUPERVISOR: Mr. Ron Aylward

CO-SUPERVISOR: Mr. Jan Olwagen

DATE: July 2011

KEYWORDS: Manufacturing, Testing, Production,

a test jig is a device used to test some component or assembly. In this report, the unit to test is a populated circuit board, and the test jig an electronic devices using a bed of nails interface to the unit under test. This bed of nails is made up of a number of spring loaded pins protruding through some isolating medium that are connected via suitable wiring to the heart of the test jig which allows measurements to be taken from and stimuli to be injected into the unit under test. Using this interface, the correct functionality of the unit under test is verified by comparing the measured values from the board with a defined set of test values with a suitable tolerance. Since the test jig is in many ways a reference point determining which products pass and which products fail in a manufacturing run, it must never be in question as to whether the jig is reliable, accurate or repeatable.

Manufacturing in Util Labs relies on a set of test jigs that have proven to be not only unreliable and difficult to maintain, but due to the lack of documentation and source code, have proven difficult to keep up to date and expand. As the Load Limiter project has progressed, different devices have been added to the product spectrum of the company and since different companies at different times have been contracted to create test jigs for each of these products, they are all different and suffer from different weaknesses. This project aims to

rectify this situation by creating a solid test jig framework, that can be expanded to suit the requirements of the specific products being tested, but is also suited to the style of manufacturing employed at Util labs (using multiple contract manufacturers). Since this framework will be used in the future to create all test equipment for circuit boards at Util Labs, new test jigs based on this framework should be as easy as possible to create.

Table of Contents

DECLARATION i
ACKNOWLEDGEMENTS.....ii
ABSTRACT.....iii
GLOSSARY OF TERMS.....xii
CHAPTER 1:.....1
1.1 INTRODUCTION AND BACKGROUND.....1
1.2 PROBLEM STATEMENT.....2
1.2.1 Initial cost of test jig and Scalability.....2
1.2.2 Maintenance turn around time.....2
1.2.3 Design integration (pre and post).....2
1.2.4 Development turn around time.....3
1.2.5 Traceability.....3
1.2.6 Documentation.....4
1.3 USER REQUIREMENTS SPECIFICATIONS.....5
1.3.1 Stakeholders.....5
1.3.2 User Requirements Specification.....6
1.4 PROJECT OBJECTIVES.....9
1.5 SCOPE.....11
1.6 TIMELINE AND BUDGET.....13
1.6.1 Timeline.....14
1.6.2 Budget.....16
1.7 CONCLUSION.....17
CHAPTER 2.....18
2.1 INTRODUCTION.....18
2.2 LITERATURE SURVEY.....19
2.3 PROPOSED PRACTICAL DESIGN.....23
2.3.1 Test Jig Pcb.....23
2.3.2 Test Jig Pc.....24
2.3.4 Bed of Nails.....25

2.4 CONSTRAINTS....27
2.5 TEMPLATING....29
2.6 GENERAL USE CASE....30
2.7 PRODUCT SPECIFICATION....32
2.8 CONCLUSIONS....34
CHAPTER 3....35
3.1 INTRODUCTION....35
3.2 THE PROTOTYPE....36
3.2.1 Prototype Characteristics....36
3.2.2 Detailed Design Description....37
3.2.2.1 Power Supplies....37
3.2.2.2 Controllers....39
3.2.2.3 Isolated Bi-Directional Ports....44
3.2.2.4 Analog Measurement....46
3.2.2.5 UUT Comms....47
3.2.2.6 Storage....48
3.2.3 The Test Jig Creation Process....49
3.2.4 Design Review....50
3.3 THE SECOND ITERATION – FIRST PRACTICAL IMPLEMENTATION....55
3.3.1 Iteration 2 Characteristics....56
3.3.2 Test Jig Creation Process....57
3.3.3 Design Review – Changes Made To Iteration 2 To Create Iteration 3....60
3.4 THE THIRD ITERATION....63
3.4.1 Iteration 3 Characteristics....64
3.4.2 Test Jig Creation Process....65
3.4.3 Design Review – Changes Made To Iteration 3 To Create Iteration 4....68
3.5 THE FOURTH ITERATION....72
3.5.1 Iteration 4 Characteristics....73
3.5.2 Test Jig Creation Process....75
3.5.3 Design Review – Changes Made to Iteration 4 to Create Iteration 5....79
3.6 THE FIFTH ITERATION....81

3.6.1 Iteration 5 Characteristics....82
3.6.2 Detailed Design Description....83
3.6.2.1 Power Supply....83
3.6.2.2 ICoupler....89
3.6.2.3 Output Driver....91
3.6.2.4 Controller....92
3.6.2.5 Light Sensor....93
3.6.2.6 Impedance Measurement And Power Test....94
3.6.3 Firmware....96
3.6.3.1 Inputs and Outputs....97
3.6.3.2 Fast Inputs....98
3.6.3.3 ADC....99
3.6.3.4 Class Summary....101
3.6.4 Test Jig Creation Process....103
3.7 THE JIG MECHANICS....110
3.8 TEST PC JAVA SOFTWARE ARCHITECTURE....112
3.8.1 Creating the Test Boards....114
3.8.2 Creating the Implementation....115
3.8.3 Add The Implementation To The Detector....117
3.10 CONCLUSION....118
CHAPTER 4....119
4.1 INTRODUCTION....119
4.2 RESULTS – USER REQUIREMENTS SPECIFICATION....120
4.3 RESULTS – PRODUCT SPECIFICATION....122
4.3 CONCLUSION....124
CHAPTER 5....125
5.1 INTRODUCTON....125
5.2 CONCLUSIONS....126
5.3 RECOMMENDATIONS....128
5.4 PROPOSED FURTHER WORK....129

APPENDIX A.....I
Generic Test Point Placement.....I
Test coverage.....I
Test Point Checklist.....II
Specifications on placement.....IV
APPENDIX B.1 VI
APPENDIX B.2.....XXI
APPENDIX C XXII
APPENDIX D XXXIII
APPENDIX E.....XLV
APPENDIX F.1 XLVI
APPENDIX F.2 LXXXI
APPENDIX G.....LXXXII
System architecture overview.....LXXXII
EOS Architecture.....LXXXV
REFERENCES.....XCI

Illustration Index

Illustration 1: Project Plan.....13
Illustration 2: Floating Probe System.....19
Illustration 3: JTAG programmer.....21
Illustration 4: Proposed Test Jig System.....23
Illustration 5: Bed of Nails layers.....25
Illustration 6: Different types of test pins.....27
Illustration 7: The standard test flow.....31
Illustration 8: Isolated +3.3V and +5V power supply set.....38
Illustration 9: Master Processor schematic.....39
Illustration 10: Slave Processor Schematic.....40
Illustration 11: Jtag Interface.....41
Illustration 12: Inter Processor Bus.....42
Illustration 13: RS485 driver chip schematic.....43
Illustration 14: Connector To The LCD.....44
Illustration 15: Isolated Input Schematic.....44
Illustration 16: Isolated Output Schematic.....45
Illustration 17: Isolated Analog Front End.....46
Illustration 18: UUT Comms Circuit.....47
Illustration 19: Storage on SD card and Flash.....48
Illustration 20: Test Jig Design Process.....49
Illustration 21: Partially populated prototype board during test.....50
Illustration 22: Test Jig Creation Process.....57
Illustration 23: Creating The Test Jig Stack In Iteration 2.....57
Illustration 24: Test Jig Stack Mock Up.....59
Illustration 25: Test Jig Generic Base Mock Up.....59
Illustration 26: Iteration 2 Implementation Bottom.....61
Illustration 27: Iteration 2 Implementation Top.....61
Illustration 28: Test Jig Stack Creation In Iteration 3.....65
Illustration 29: Guide board showing a cut-out made for a dense number of test points.....68

Illustration 30: Generic Test Jig Rev C before population.....69
Illustration 31: Iteration 4 Test Jig Stack Creation.....75
Illustration 32: Stubs On ICoupler On Stopper Template.....77
Illustration 33: Optional Switch Mode Power Supply On Stopper Template.....77
Illustration 34: Fitting Iteration 4 Boards On the Mechanical Base Plate.....79
Illustration 35: Mutually Exclusive Placement of R44 and D6.....84
Illustration 36: 3.3V Switch Mode Power Supply.....86
Illustration 37: The 12V And 5V On Board Power Supplies.....87
Illustration 38: ICoupler Schematic.....89
Illustration 39: ICoupler CMOS Transformer technology.....89
Illustration 40: Vias Added And Copper Over Pads To Increase Heat Sync.....90
Illustration 41: The 40 pin Interface Connector And Output Driver.....91
Illustration 42: Schematic For One Channel Of The ULN2003A driver.....91
Illustration 43: STM32F103 Microprocessor Generic Test Board Controller.....92
Illustration 44: Light sensor.....93
Illustration 45: Impedance Measurement Circuit.....94
Illustration 46: Input and Output Actions (Only Relevant Methods and Members Shown).....97
Illustration 47: Pulse Test Actions (Only relevant methods and members).....98
Illustration 48: ADC Test Actions (Only relevant methods and members).....100
Illustration 49: Iteration 5 Test Jig Stack Creation.....103
Illustration 50: Board To Test Created As .brd File.....104
Illustration 51: Close Up Of Heat Sync And Stubs On ICoupler On Stopper Template.....107
Illustration 52: Outline Board Created Using Positioner.....109
Illustration 53: Generic Test Boards and Test Jig Stack Bolted To Lifting Plate.....111
Illustration 54: The Java Architecture Overview.....112
Illustration 55: Test Board Specific Implementations.....114
Illustration 56: Test Jig Implementation In The Framework.....115
Illustration 57: Adding The Implementation To The Detector.....117
Illustration 58: Virtual Message Bus Showing An UPWARDS Routed Message.....LXXXII
Illustration 59: What Each Device Connected To The Virtual Message Bus "Sees".....LXXXIV
Illustration 60: EOS Operating System Modules Interactions.....LXXXV

Index of Tables

Table 1: Manufacturing test solutions.....22
Table 2: Hardware designs associated with the prototype.....36
Table 3: comparing JTAG and serial bootloader programming on the test jig.....53
Table 4: Hardware designs associated with iteration 2.....56
Table 5: Hardware designs associated with iteration 3.....64
Table 6: P1 and P2 mappings on temp-*boardname*.brd.....66
Table 7: Maximum UUT Board Size For Iteration 4.....75
Table 8: Hardware designs associated with iteration 5.....82
Table 9: Components To Place For External vs On Board Power Supply.....84
Table 10: Actions Performed By The Generic Test Jig Controller.....92
Table 11: Results - User Requirements Specifications.....121
Table 12: Functional Specifications Results.....123
Table 13: Electrical Specifications Results.....124
Table 14: Interface Specifications.....124
Table 15: Test Point Check List.....IV

GLOSSARY OF TERMS

TUT: Tshwane University of Technology. The home page for the university can be located at:
www.tut.ac.za

URS: User Requirements Specification

MCU: Measurement and Control Unit, the core measurement product in the Util Labs family of products.

Test Jig: An electronic device used to test a unit or assembly that is being manufactured or tested against some known reference(s). In the scope of this project, the test jig will be a bed of nails type test device (or similar) used to test manufactured circuit boards.

Test Pin: a spring loaded pin, generally made from copper with a gold plating that acts as the interface point on the bed of nails between the circuit board being tested, and the test jig. [11] The Test pins used throughout this project are from Ingun [12].

Bed Of Nails: An interface to a circuit board used on the test jig that consists of a number of spring loaded pins that, when actuated will connect to the circuit board to be tested so that reference stimuli can be injected into it and measurements of electronic signals can be made on the board.

JTAG: The acronym for the Joint Test Action Group, that is now synonymous with the debug interface described in the standard IEEE 1149.1-1990. This port is used to debug, test and program various processors and other programmable devices like FPGAs and DSPs.[5]

iCoupler: iCoupler technology is a registered trademark of Analog Devices [24] that defines a way of getting signals across an electrical isolation barrier. It is one of the core technologies in this project that allows electrical isolation between the unit under and the control pc of the test jig.

ADC: Analog to Digital conversion. The act of measuring an analog voltage signal and converting it to a representative digital signal using appropriate electronic hardware. In the scope of this project, analog to digital

	conversion is achieved either by using a dedicated ADC chip (AD7918) [21] or using the on board peripheral of the ARM7 STM32F103 micro processor[19].
LDR:	Light Dependant Resistor. A resistor that changes its resistance based on the amount of light that is incident on it at any point in time. In the scope of this project, LDRs are the main technique of measuring if an optical component (like a lamp or LED) on the unit under test is functional or not. [27]
Unit Under Test:	The device that is presently being tested by a test jig. In the scope of this project, the unit under test will be the circuit board that has been placed on the bed of nails and is being tested.
Test Jig Stack:	The test jig stack is the set of base, guide, stopper and outline pcbs that make up a single bed of nails for a specific type of unit to be tested.
Cost per Test:	Cost per test is a value used in production to factor the cost of testing equipment and testing time into the cost of a manufactured product.
BOM:	Bill Of Materials that contains all of the components and sub-systems required to manufacture a system or sub-system.
GUI:	Graphical User Interface
Kicad:	An open source suite of EDA (Electronic Design Automation) software used to capture schematics and create pcb artwork. Kicad is composed of 5 main components: • Kicad Project manager • Eeschema schematic capture • PCBnew pcb artwork creator • Cvpcb used to match schematic components to artwork • Gerbview gerber viewer. Kicad is used extensively in this project in all schematic and pcb work. The home page for the Kicad project is http://iut-tice.ujf-grenoble.fr/kicad/

Eclipse: Eclipse is a generic open source IDE (Integrated Development Environment) used to write, build, run and debug code written in a number of languages, but optimised specifically Java. In this project, all code written in Java has been written and tested using the Eclipse IDE. The home page of the eclipse project is: http://www.eclipse.org/

Crossworks: Rowley Crossworks is a commercial IDE used for specifically writing, building and testing code for embedded applications in C/C++. All code written in this project on the STM32F103 series of processors was done in Crossworks. The home page for Rowley is: http://www.rowley.co.uk/

CHAPTER 1:

1.1 INTRODUCTION AND BACKGROUND

Util Labs is a fairly new company that is rapidly entrenching itself as a large player in the energy sector in South Africa by offering a holistic approach to metering, energy balancing and smart grid load and appliance control. With a national footprint of 30 000 metering points, with planned expansion in 2011 to 80 000 metering points, the company aims to be the de facto standard for load control in South Africa, as well as keeping an eye on the international market by forming agreements with companies in various countries, including Brazil, Poland, India, Italy etc.

All products in the Util Labs range, form part of a larger system that can be expanded to meet the requirements of the customer (in the case of South Africa, the customer is Eskom or the local municipalities that supply electricity to all people in the country). These products are all developed in house and manufactured in South Africa. The style of manufacture used is one of contract manufacturing, where Util Labs is responsible for supplying a contract manufacturer with Datapacks, assembly instructions, components and test equipment needed to assemble the product.

This system works very well, but there are a number of problems that have been faced in the past production runs due to the fact that early on in the project, all the design of test equipment to be used was outsourced to an external company, and the test equipment received has proven to be expensive, rigid, unreliable and not scalable.

It is with an eye on solving the production problems faced in the manufacturing of product that this project was embarked on.

1.2 PROBLEM STATEMENT

As mentioned in the introduction, the testing phase of the manufacturing of product at external contract manufacturers has been identified as a problem. The following issues have been raised and need to be addressed by this project:

1.2.1 Initial cost of test jig and Scalability

One of the problems faced is that there is a very large initial cost to getting a test jig made, especially around the machining of the bed of nails itself. The problem with this is that there is no chance to have small production runs that are tested to the same level as large production runs as it is not economically feasible to buy a test jig when only 1000 units are created. This limits the ability to prototype new products in large volume, as only a small number can be created when they need to be tested by hand. Additionally, the test jigs are fast enough as single test jigs for medium sized production runs, but when production gets ramped, the jigs are a bottleneck, as there is no way to speed up the throughput on a single jig, and buying additional test jigs will be as expensive as the original.

1.2.2 Maintenance turn around time

Due to the cost of the jigs, there are no hot-swappable backups available – this means that when there is a breakage on the jig during a manufacturing run, it causes a long line stoppage while the test jig is debugged and fixed. To overcome this, a set of hot-swappable jigs need to be created in the framework for each PC board with the goal of dropping line stoppage time to below 20 minutes.

1.2.3 Design integration (pre and post)

Currently the process to create a test jig involves capturing the design gerber files and marking on the schematic where the test pins should go and then sending this to an external company. The engineer at the external company uses the hand markings on the schematics and gerber files to program a CNC drilling machine to create the bed of nails front face. Once the front face is milled and drilled, the test pin holders are inserted and the pins are wired up to a number of relay controllers, SCADA devices, power supplies etc. depending on the requirements of the board to be tested. Once the hardware is created, a custom application is written for the PC to control the jig.

In addition to being laborious, this process has so many steps in it, all with human intervention, that the chances of a bug creeping into the eventual bed of nails are quite large. When this process is followed from start to finish all at once it is not much of a problem to have so many steps, but later on in the design cycle of products when changes need to be made to the bed of nails (for example the moving of a test point, or the addition of a test point) It makes it very difficult to make any modifications to the test jig as all of the steps need to be followed again due to the heavily propriety nature of the design (each test jig has a different internal layout, and a different PC application). This is a costly process due to the level of interaction involved, but also adds the risk of damaging the bed of nails, as to make the changes the current test jig needs to be disassembled, modified and reassembled. Often due to the timelines involved because of line stoppages, quick and dirty (generally undocumented) changes are made to the existing jigs which cause problems in the long run.

The main problems identified here are the lack of integration of process linking the design of the product and the design and manufacture of its corresponding test jig, and the large differences between test jigs.

1.2.4 Development turn around time

As an add on point to the previous point, due to the back and forth interaction between the company and it's test jig manufacturer, the development of a new test jig can be a slow process that can take a couple of months to develop. This time is spent on the test jig once the product requiring the jig has already been designed, as it is only once the product is completed that the full test regime for it can be set up. This therefor increases the time it takes between the development department releasing a design for manufacture and the final product arriving in stores, by the length of time it takes to design and manufacture the test jig.

1.2.5 Traceability

Currently, all the test jigs used respond with a simple pass/fail interface that puts the onus on the operator of the jig to sort the boards that failed to go to the different rework stations, or in the case of boards that cannot be repaired to write them off. This is a flawed process as it allows for human error to creep in and places too much responsibility in the process of sorting the boards on the operator.

1.2.6 Documentation

Since all of the test jigs currently in production are different, there is no central document archive for them. Each jig has differing levels of design, implementation, installation and operating documentation ranging from none at all to some. In addition to the updating of the processes used to creating test jigs, one of the secondary goals of this project is to get a full set of documentation (design document, manufacturing document, installation document and operations manual) for each of the test jigs created.

1.3 USER REQUIREMENTS SPECIFICATIONS

1.3.1 Stakeholders

The stakeholders of the framework proposed in this project are not restricted to the individual operators of the test jigs, so it is important to list the interested parties here before outlining the user requirements specification:

- **The production manager.**
  The production manager is concerned with keeping failure rates and efficiency on the production line optimized. As such, it is in his/her interest to have reliable test equipment that can not only identify individual failed units, but also allows for the detection of failure trends throughout the production process that will point to possible inefficiencies in the process.
- **The operations manager.**
  The operations manager is concerned with the flow of components and orders to the factory, as well as the flow of product to the customer from the factory. The benefit that the operations manager will get from this project is directly related to the cost per test and level of scalability that the project can achieve.
- **The operator.**
  The operator will be the the person that actually handles the test jig and tests the individual PC boards that will form part of the manufactured product. The benefit that the operator should get from the project is a simpler interface and also having the process of sorting failed/passed product simplified.
- **The Test Engineer.**
  The test engineer is responsible for overseeing the test equipment and processes followed on all of the products manufactured at a factory. The test engineer will also be responsible for fault finding a failed test jig. The benefit the Test engineer will get from this project is hot swappable jigs, with a simpler interface, that should minimize downtime.
- **The development engineer**
  The development engineer is responsible for the designing of products (in this case in the Util Labs family of products). His/Her interaction with the production department is such that when he/she has completed the design, It is handed over to

the production department so that a set of test jigs can be created for the individual PC boards/units as well as the product as a whole. The benefit the developer/engineer should see from this project is a much higher level of integration where he/she has a greater input into the testing process and can see the testing of his/her product done much faster.

- **Integration Testing Department**

  The Integration testing department is responsible for continually testing the full system of Util Labs products whenever a new hardware/software version is released to ensure that integrity is maintained across the full product line. The Integration testing department is the gatekeeper between the development department and the production department. By supplying the department with a set of test jigs that can be filled with products that have a known failure, the system can be tested to see what the reaction is in different failure modes.

- **Quality officer.**

  The quality officer is responsible for ensuring that the processes followed in manufacturing the product conforms to the highest quality standards, in the case of Util Labs using the principles in the ISO9000 framework [1]. The benefit that the quality officer should gain from the project is the traceability in failures that should allow him/her to optimise out all problems in the process with the help of the operations and production managers, as well as a set of documentation that should conform to the quality requirements set out in the ISO9000 framework.

With these users in mind, the user requirements specification can be set up.

1.3.2 User Requirements Specification

The user requirement specifications need to meet with the following points:

- **System Description**

  The result of this project should be a framework that will allow the creation of test jigs that can accept a manufactured PC board, test it against a predetermined test sequence, and indicate to the operator whether the tested PC board passed or failed based on the test criteria.

- **Cost**

  The framework put into place should be such that the test jigs generated should be significantly cheaper to manufacture than the current set of test jigs. This does not take into account the development time taken to create the jig, just the physical cost of components and time to manufacture it. The goal price is set to < R20 000 per full testing station compared to the current cost of between R20 000 and R100 000 per test station.

- **Scalability**

  Scalability in this context is the ability to add more test equipment to test units in parallel and thereby ramp up production. The way in which this is achieved is not as important as the ability to test a number of units all at once. Additionally the cost of the test jig must be such that a number of test jigs can be bought for a specified budget and used in parallel. The cost of this scalability should be such that if production is required to double throughput (and by extension double the testing capabilities of the factory), that the cost of the operation would be less than R2 on each unit for a 10 000 unit production run. In other words, doubling the testing station should cost less than R20 000.

- **Design Flexibility**

  As much hardware and software as possible should be re-used and shared between test jigs to bring down development time as well as increasing the reliability of individual components in the test jigs. Code and hardware should therefor be designed to be as flexible as possible so that it can be applied to a number of different situations. The design flexibility should be such that for a moderately sized test jig with:

  - 6 analog inputs
  - 10 digital inputs
  - 10 switched outputs
  - 4 optical inputs
  - 2 pulsed inputs

  The design time would be less than 1 month for both the hardware and software (including testing and documentation).

- **Integration**

  The integration of the test jig design into the design process of new products is necessary to minimise the time required to create a test jig for a new product. The changing of a current test jig must also be easy enough and cheap enough to make changes on test jigs that have already been designed viable. The integration should be such that to add/move/remove a test point from a pre-existing test jig design to take into account new changes to it should take less than 2 man days.

- **Maintenance**

  The maintenance of the test jig should be such that it doesn't cause line stoppages of more than half an hour. The cost should also be such that hot swappable test jigs are available, and the price of a secondary set of hot swappable jigs should be less than R2 on each unit for a 10 000 unit production run. In other words, a hot swappable testing station should cost less than R20 000.

- **Support traceability**

  In order to increase traceability on the line of faults in the production process, the test jig must also make provision for a system that will remove the responsibility to sort the different failed units from the operator and increase automation in the process. The final product should implement a system where each individual test on the test jig is regarded as a separate failure mode, with its own recommended course of action.

1.4 PROJECT OBJECTIVES

The main design objective in this project is to replace the current family of test jigs used at Util Labs with a new generic family of test jigs that will support the style of manufacture as well as the wide array of different products manufactured and sold by the company.
Secondary objectives are as follows:

- **Design Integration**

  The author hopes to achieve a greater level of design integration with the design process as implemented in Util Labs. This integration should allow test points to be added to a design either during the design phase, or to take a previous design and add test points to it and then create a test jig for the board directly from the design files. The objective would be to achieve a full design turnaround time of one month or less on a test jig. The test jig framework of creating new test jigs should be designed in such a way as to support this objective.

- **Simplicity**

  The framework should generate test jigs that are simple enough to use that with only simple button or lever interfaces an operator can go through the entire test sequence for a single unit, as well as allowing for a simple way to sort failed units so that they may be reworked or thrashed depending on the nature of the failure, to support the quality department in goal of producing the highest possible quality of product.

- **Scalability**

  The framework should create test jigs that are scalable by virtue of their very low cost, as well as the ability to reuse as many components as possible. This same cost requirement would support the creation of hot swappable backup test jigs to be made available to the factory manufacturing on behalf of Util Labs.

- **Reliability**

  The test jigs created in the frame work should be reliable enough to do a run of at least 10 000 units without failing, and in the event of failure, should require no more work to swap out or repair than can be done in half an hour to get the production line back up to speed again.

- **Ease of installation**

  The set up of the test jigs created in the framework should be simple enough that the product being manufactured can be changed, and it should take no more than 1 hour to re-set up the next set up jigs for the production run of the new product.

1.5 SCOPE

The scope of this project is the generation of a test jig framework, as well as any test jigs that are created within the framework during the development of it. This includes:

- The hardware that will run the actual tests, injecting into and measuring signals from the UUT.
- The hardware platforms of the individual beds of nails that hold the pins map to the test points on the UUT.
- Any firmware created on the hardware and bed of nails platforms including tasks, gateways and any device drivers coded specifically for the purpose of the generic test jig framework.
- The processes involved in creating the beds of nails from the design files of a product to be tested.
- The application code required to support the generic test jigs, including any classes required to communicate to the test jigs, collecting and interpreting the test values and controlling the flow of testing.
- The porting of the EOS operating system to support the test jig firmware.
- The expansion of the existing Java swing (swing is part of the core set of Java test jig technologies, used in this project to create a graphical user interface to the operator) [2] framework to support the interface to the operator.
- The expansion of the message catalog and supporting firmware and software to support the test jig specific messages on the standard message bus.
- The expansion of the label printing capabilities of the Java swing framework.

Some parts of the design of this framework however do NOT form part of this project, owing to the fact that they had already been in place before the commencement of this project, and although the author was involved in their development in many instances they form part of the current code base of Util Labs:

- The EOS base operating system, including its file system, message bus implementation and existing base hardware interfaces.
- The Java swing GUI framework itself.

- The common message bus structure implementation that allows all devices in the Util Labs family of products to communicate to the server with the same message catalog.
- the JTAG Class interfacing to the command line interface of OpenOCD, other than the small changes made in support of the project.
- The Base mechanical design specific to the generic test jig framework. (Although this work is done specifically for the framework, it Is done in collaboration with the mechanical engineering arm of Util Labs, and as such all mechanical designs done in Solidworks are done by them, and not the author.)

1.6 TIMELINE AND BUDGET

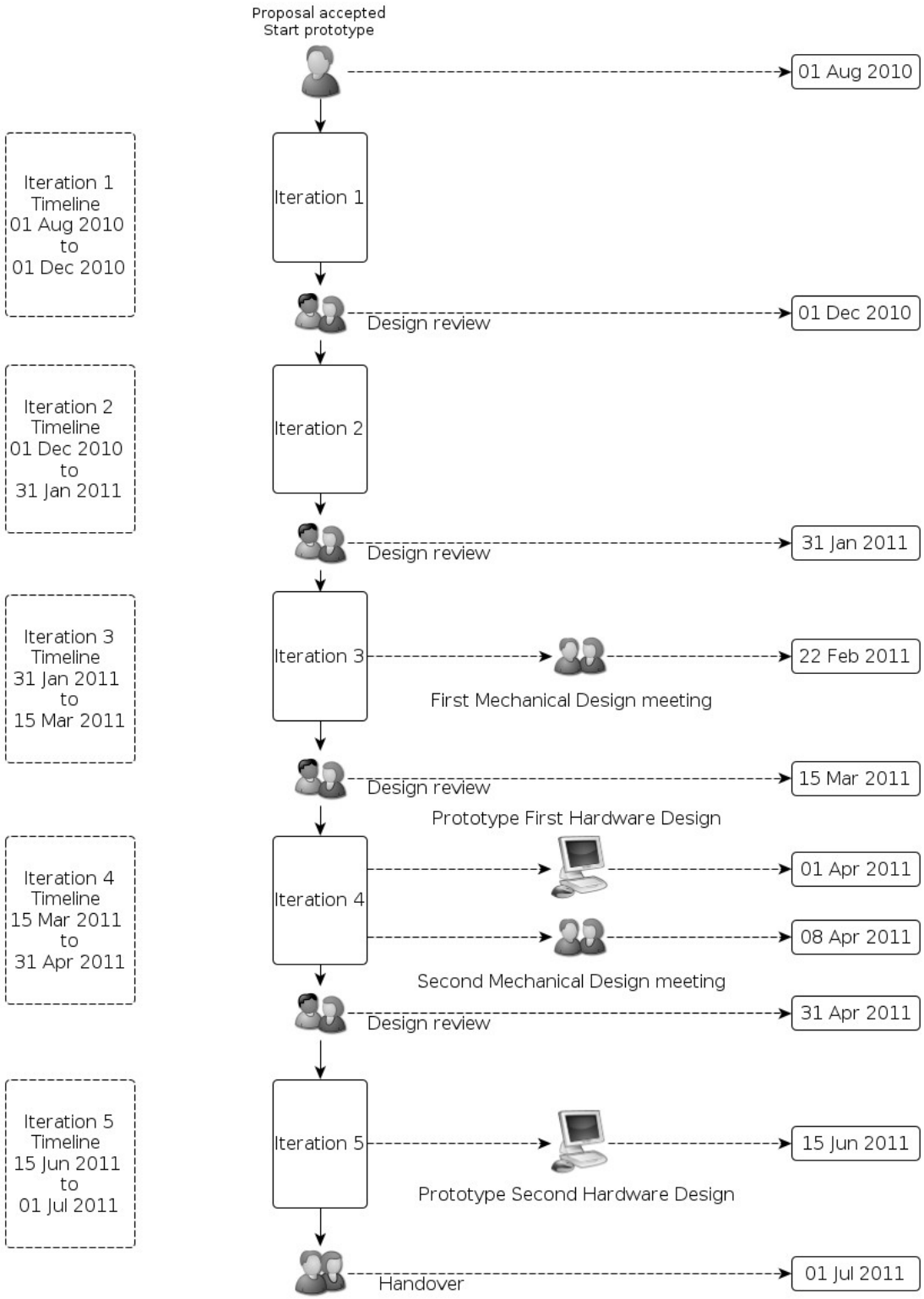


Illustration 1: Project Plan

1.6.1 Timeline

The project will use an iterative design approach, and as such the timeline is not completely set in stone at the start, but the following milestones are used as a guide:

- **milestone:** Project Started (Proposal In) **01 Aug 2010**

First Iteration (approximately 3 months)

- Test jig prototype schematic capture complete +-3 Weeks
- Test jig prototype PC board layout done +-2 Weeks
- Initial application design (design Interfaces) +-2 Weeks
- Prototype test jig PC board and fix any problems +-1 Week
- Documentation +-1 Week
- **milestone:** Design Review **01 Dec 2010**

Second Iteration (approximately 8 weeks including December holidays)

- Use first iteration design review to change hardware schematics and PCBs, add first test jig stack +-2 Weeks
- Expand application to include standard tests +-2 Weeks
- Documentation +-1 Week
- DECEMBER HOLIDAY +-2 Weeks
- **milestone:** Design Review **31 Jan 2011**

Third Iteration (approximately 6 weeks)

- Use second iteration design review to change hardware schematics and PCBs, update test jig stack +-2 Week
- Expand application to include EOS standard tests and add signal injection +-3 Weeks
- Documentation +-1 Week
- **milestone:** Meet with Mechanical Engineering dept for first mechanical iteration design meeting **22 Feb 2011**
- **milestone:** Design Review **15 Mar 2011**

Fourth Iteration (approximately 6 weeks)

- Use third iteration design review to change hardware schematics and PCBs, update test jig stack +-2 Week
- Expand application to include programming and label printing functionality +-2 Weeks
- **milestone:** Prototype hardware with first mechanical prototype **01 April 2011**
- **milestone:** Meet with Mechanical Engineering dept for second iteration design meeting after prototyping **08 April 2011**
- **milestone:** Design Review **31 April 2011**

Fifth Iteration (approximately 8 weeks)

- Use fourth iteration design review to change hardware schematics and PCBs, update test jig stack +-2 Week
- Expand application to auto detect communications ports and add network printing +- 2 Weeks
- Documentation +- 1 Week
- Write operators manual, installation manual etc. +- 2 Weeks
- **milestone:** test full system using second iteration hardware design and final design of generic test jig and test jig stacks **15 June 2011**
- **milestone:** handover to production **01 July 2011**

1.6.2 Budget

The budget for the prototype system of the generic test jig will be R40 000. This value does not take into account the salary of the author or any of the people he interacts with. It does include the following:

- Components worth R5000 (mostly from the Util Labs in house store, see individual BOMs for more information about individual components) R5 000

- The first set of jig mechanics R20 000
- The cost of the PC used on the test jig prototype R6 000
- The cost of manufacturing the PC boards depending on boards

Once production ready jigs can be created, a manufacturing cost can be calculated for each individual jig type created in the framework.

1.7 CONCLUSION

Testing of product is a such a core function in the manufacturing process that is currently causing a large amount of time and money to be lost on design time and downtime due to the current set of test jigs. Expansion of the Util Labs product portfolio is ongoing and as such it is necessary to increase the testing capabilities of the company.

Once implemented, this project should create not only allow for the creation of reliable test jigs, but jigs that can in the case of failure be replaced or repaired in no more than half an hour to minimize cost and loss of production time. The created test jigs should also increase the manufacturing capabilities of Util Labs by supporting the scalability of production, as well as creating tools that will allow for greater quality to be built into the product. The design time of new test equipment should be kept to a minimum by using a solid base of reliable design sub-components.

CHAPTER 2

2.1 INTRODUCTION

In this chapter different testing systems are evaluated and an idea is created of the solutions available in the current testing market. Using the information gained in the literature survey, an initial design is proposed to conform to the requirements set forth in the user requirements specification. The electrical and functional requirements for the design are stated at the end of the chapter.

2.2 LITERATURE SURVEY

The concept of bed of nails testers is not new and as such there is a wealth of information available on their design and implementation. A lot of the systems referenced however do suffer from the problems outlined above, although to differing degrees. An elegant alternative solution put forth is to use a fixture-less in circuit test solution called a flying probe tester. An example of a flying probe tester can be seen from Seica Technologies[3]:

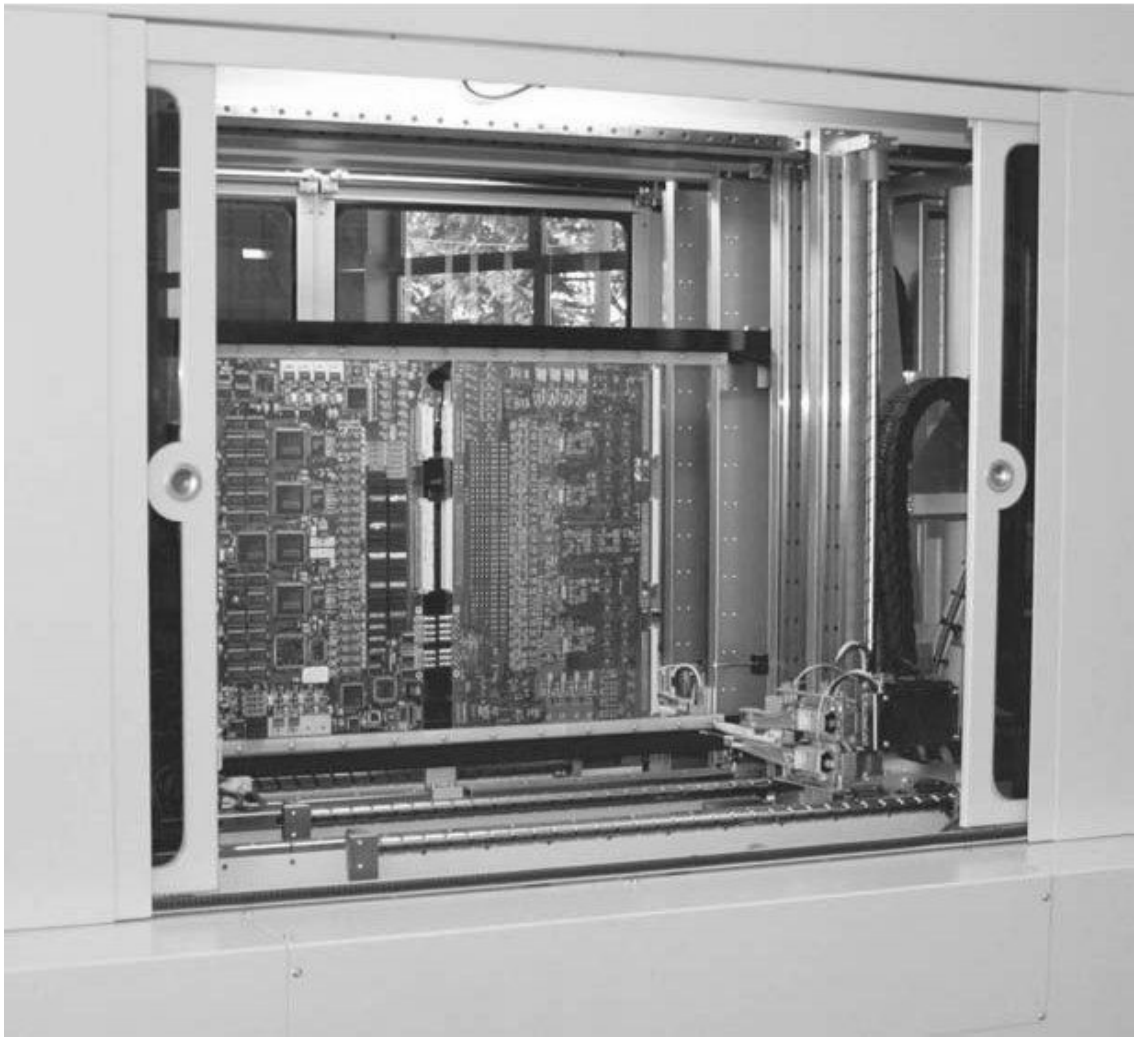


Illustration 2: Floating Probe System

(image retrieved from [3])

In this system instead of having a bed of nails physically, a virtual bed of nails is created with a number of points that need to be tested and this virtual bed of nails is programmed into the system. During the test, the test probes of the machine are physically moved to each of the x-y coordinates of the test points in turn, injecting the appropriate signals, and testing each point to a predefined pass criteria before moving on to the next. The

advantage of this system and other types of fixture-less in circuit test solutions is that they address the flexibility requirement put forth in the problem statement as well as having a very short development turn around time [4], however the down side of this system is that the initial costs are exorbitant and the system cannot scale at all due to very slow testing times per board. These kinds of system are generally used to test prototypes, or in some cases to test bare circuit boards on panels.

A second alternative option is also available in the form of boundary scan technology. This technique of testing is best known through its JTAG implementation by the Joint Test Action Group (JTAG) which is widely used in debugging and programming of microprocessors and programmable logic chips [5]. In a test system that uses boundary scan technology, each of the input and output pins of the chips connected on the JTAG scan chain are scanned via a set of registers and their respective levels compared to known good values during various phases of the test. The disadvantage is that the PC board being tested using this technique needs to be designed with this technique in mind, and it is very difficult to implement on to an older design if JTAG testing was not taken into account from the very beginning. Another disadvantage is that using the scan chain, only those levels that are visible through processor I/O pins are measurable, and some parts of the system cannot be tested. The advantage to this system though is a very low test jig implementation cost and a very high level of test integration with the design. Although I do not wish to follow the route prescribed by JTAG testing, I do feel a lot can be learned from the implementation and in the initial design I will be using a JTAG programmer to program the microprocessors of the units under test on the system.

Illustration 3: JTAG programmer

(image retrieved from [6])

The third and final alternative option is to have designs with built in self test . Systems that have this as part of the design have the highest level of system integration during the design phase, as during the design process each unique test needs to be added to the system as another “component”. It minimizes the testing time, as well as the need for complex test hardware, but greatly increases design time and product cost. This sort of system however is widely used in military and medical systems where a very high level of testing, as well as continual testing is required. It is however not really suited to more commercial products as the additional expenditure can normally not be justified [7].

The following table shows a summary of some of the advantages/disadvantages of the different kinds of testing solutions:

Solution	***Advantages***	***Disadvantages***
Bed of nails	Simple operation, handles very high volume, very rugged, very wide test coverage	High initial cost, little scalability, cannot easily be modified, can damage the PC board with sharp pins
Fixture-less testing	Very flexible (once initial cost is paid subsequent changes are negligible), easy to retrofit, very wide test coverage.	More complex operation, cannot handle high volume, slow test speed, very high initial cost
Boundary scan	Low initial hardware cost, high system integration	High design cost (must be designed into system), very difficult to retrofit, not 100% test coverage
Built-in test	Very high system integration, low hardware cost	Very high design costs, must be designed into all parts of the system, Cannot be retrofitted

Table 1: Manufacturing test solutions

The system I aim to develop in this project takes from all of the above solutions. I aim to combine the ruggedness of the bed of nails and add some of the flexibility of fixture-less testing and combine this with boundary scanning technology for programming and allow for a high degree of self test in designs. By taking the strongest points of each of these approaches I feel that a flexible, scalable and affordable test jig system can be created.

2.3 PROPOSED PRACTICAL DESIGN

The basic system outline is shown in the diagram below:

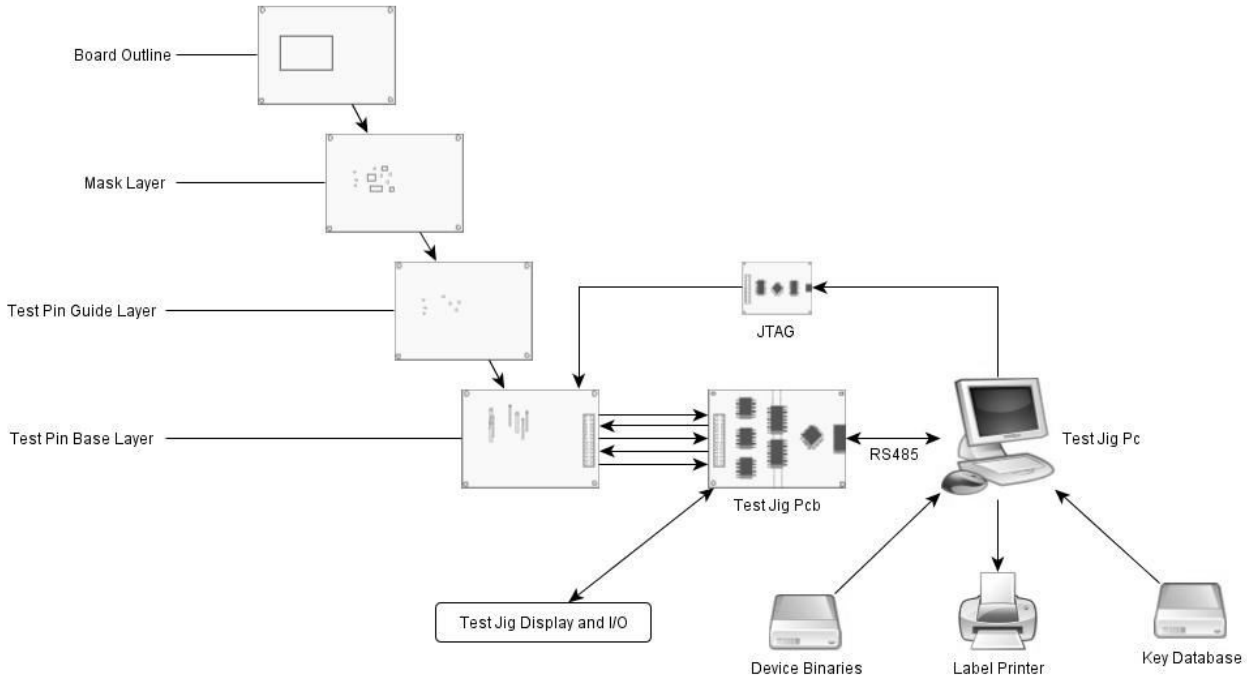


Illustration 4: Proposed Test Jig System

As can be seen in the above image, in the proposed system a standard PC will be used to interface to a generic test jig PCB, which will house all of the measurement hardware, which will in turn interface to a bed of nails that contains the signal conditioning hardware, and is also the main interface to the unit under test. The PC will also interface to the programming hardware, as well as some standard peripherals. The individual components are described in more detail in the pages to follow.

2.3.1 Test Jig PCB

The test jig it self will be based around an ARM7 Cortex processor running a variant of the EOS embedded messaging operating system. The Jig will also need to conform to the following specifications (these however may change to add new requirements as needed):

- 10 binary inputs
- 10 voltage inputs
- 4 frequency inputs
- 2 counter inputs

- 2 serial communication lines to unit under test
- 10 binary outputs (for signal injection)
- Jig information interface for when jigs are running in parallel
- keypad or buttons to operator the jig
- RS-485 or equivalent for communication with PC
- Permanent storage for test results and test scripts (flash memory or SD card)
- Permanents storage must be accessible via Windows/Ubuntu (FAT file system)

These requirements may however at this point not reflect a complete overview of what is required, and certain I/Os may need to be added to ensure full test coverage. If insufficient I/O is available, then an additional processor may be added with one processor acting as a master, and the second acting purely as a slave I/O processor.

The I/O to the unit under test needs to be electrically isolated from the rest of the system to ensure that during fault conditions the rest of the system is protected. In addition to being electrically isolated, all of the ICs that are electrically connected to the unit under test need to be socketed through-hole ICs so that in the event of a failure, they are easy to swap out very quickly.

The permanent storage will need to hold the test jig script that determines what will be tested on which I/O lines to the unit under test. This script needs to be transferable easily to a second jig in the case of failure to speed up the hot-swap process on the manufacturing line. The scripting language used will be either an open source standard scripting language, or a custom simple scripting language that is specific to the project, depending on whether a scripting language is found that has all of the features required and has a small enough footprint to be placed on the ARM processor. An alternative solution can be to create the tests to be run as a component in the test jig firmware – although this will need the test jig code to be recompiled for each test and the jig itself will need to be reprogrammed, this will allow for a MUCH faster swap out time, as an identical test jig(s) can be kept on stand-by in case of failure. Both options should be prototyped.

2.3.2 Test Jig PC

The test jig PC will run a Java application with a Swing GUI implementation that shows the current status of the Jigs(s). The PC will provide storage to serialized binary files for the unit under test and communication to a central production server. By splitting the

programming section away from the test jig itself, different kinds of programming can be supported by the system and a simple interface can be maintained between the test jig and the PC. This same interface will also allow access to normal PC peripherals like label printers.

2.3.4 Bed of Nails

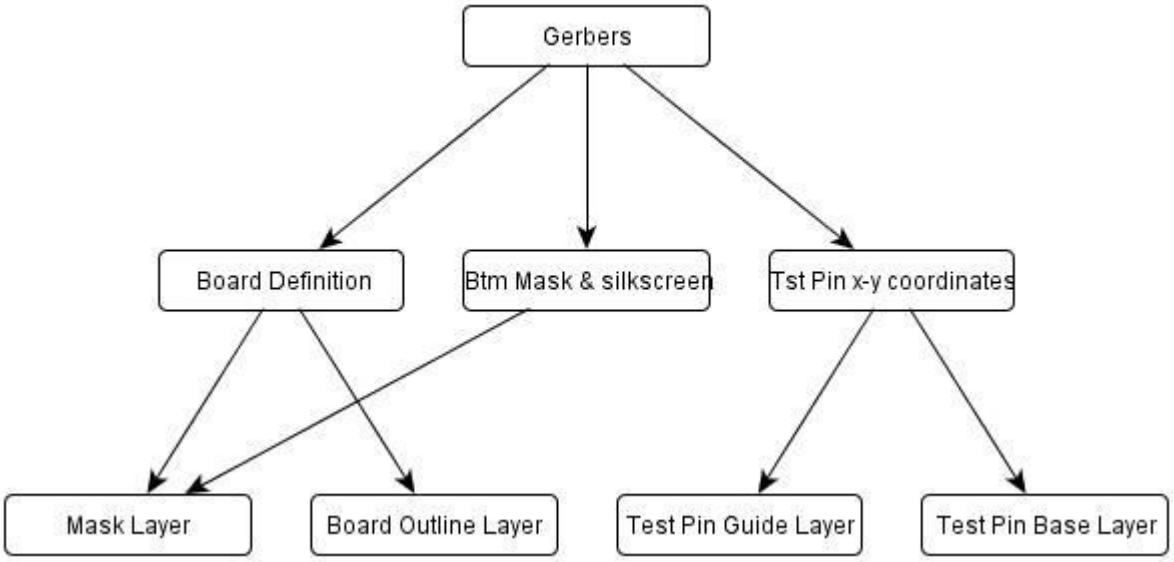


Illustration 5: Bed of Nails layers

The bed of nails forms the interface between the unit under test and the test jig system, and in many ways is the core component that will solve (or attempt to solve) many of the problems put forward in the problem statement. The bed of nails consists of the following 4 components:

- Test Pin Base Layer
- Test Pin Guide Layer
- Mask Layer
- Board Outline Layer

Each of the layers is generated off of the gerber files for the PC board that is to be tested. The functions of each layer and how they are generated are as follows:

- **Board Outline Layer**

  The board outline layer acts as a physical guide to align the unit under test with the test pins. It is generated from the board definition layer of the gerbers by drawing an outline around the board and using it to make a cut out on the board outline layer PCB.

- **Mask Layer**

  The mask layer guides the test pins to the right test pads on the unit under test, and acts as a base for the unit under test to rest on. It is generated off of the bottom mask layer and silkscreen gerbers that define where there are component leads and physical components. Cut outs need to be made on this layer for components on the bottom layer, and holes need to be drilled for test pins to go through, but also for through hole component leads that are fitted on top of the board.

- **Test Pin Guide**

  The test pin guide gives stability to the spring loaded test pins and forces them to stand up straight and align with the unit under test correctly. The board is generated directly from the x-y coordinates of the test pins so a hole is made for each one of them.

- **Test Pin Base**

  The test pin base is where all of the test pins are soldered down and are routed to a standard connector that interfaces with the test jig PC board. The signal conditioning for the signals to and from the unit under test is also on this board. The board is generated from the x-y coordinates of the test pins. The test jig script will identify which test pins go to which pins on the connector and what voltages etc need to be measured there. Different Test Pins will be standardized on depending on what interface is available to the unit under test. The pins generally used can be seen in the following image:

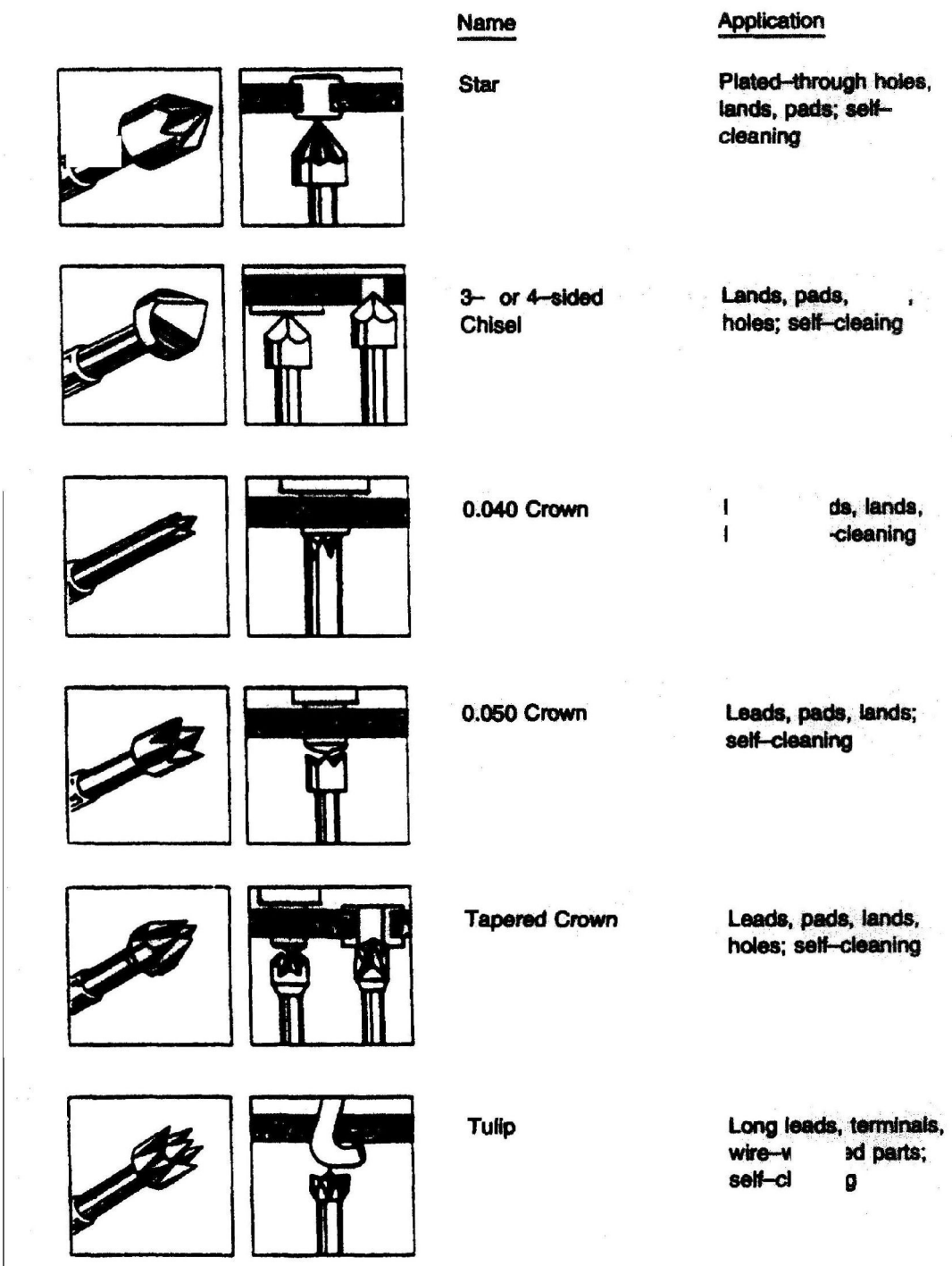


Illustration 6: Different types of test pins

Image retrieved from [8, page 74]

2.4 CONSTRAINTS

In order to ensure that the framework performs as required by the URS, the following 3 constraints are put in place to ensure that the process of creating a new test jig from

existing design files works as smoothly as possible:

- **All the test points must be available from 1 side of the unit under test**
  This will ensure that the simplest test pin interface can be used, as only 1 axis of movement is required to bring the test pins into contact with the unit under test. If there are test points that need to be measured from the other side, additional design time will be required.
- **The communication interface must run on the standard message bus**
  This ensures the maximum amount of re-use of the communications interface, and allows for fast turn around times in test jigs. If however a second message catalog needs to be implemented with a network bridge in between, it will require additional design time.
- **Movement of test points between versions should be minimized.**
  If the test points stay stationary between versions, then the same beds of nails can be used for the different versions. If however any test points move, all the boards in the bed of nails stack need to be changed, and this requires additional time in designing and prototyping.

Appendix A is a guideline to placing test points on future designs to ensure that they are easy to use.

2.5 TEMPLATING

In order to facilitate the fast development of hardware and software in the framework, the parts of the design that are common between all bed of nails implementations need to be split off into hardware and software templates. The main templates that need to be created are:

- **The bed of nails stack boards**

  Each of the boards in the bed of nails stack (outline, mask, guide, base) always perform the same tasks on all of the bed of nails implementations. As such they can all have a standardized outline as well as standardized input and output connections (for those ones that have input and output connections), and just have the positions of the test points changed between different test jig implementations.
- **Generic board interfaces**

  Each of the bed of nails boards perform the same tasks, so many of the functions can be grouped together in software into a template and just the hardware mappings changed for individual boards. This templating forms the basis of the genericTestBoard class in the software framework, where each descendant class will just add the hardware mapping.
- **Test Jig implementations**

  Each of the test jig implementations will obviously run different tests since they are each for different products, but many of the values that are tested are done so in standard ways (voltage measurement, state measurement etc) and these should be grouped together intelligently so that only the reference values and other essential parameters need to be specified when a test sequence is set up. These test functions will form part of the GenericTestJig class, where each descendant class just adds the specific tests needed to run and the sequence.

2.6 GENERAL USE CASE

The main person to interact with the test jig will be the operator, and as such, his interaction is the major use case against which the test jig will be tested. The test process r will loosely follow the structure shown in the diagram below:

- **Insert PCB (operator action)**
  - Before the operator inserts the unit under test, the LCD panel must indicate to him/her that the jig is ready to accept a unit to test, or else show some error code that indicates it is not ready.
  - If it is ready the unit under test is inserted into the test jig, and is physically clamped down using a standard physical clamp.
- **Start Test (operator action)**
  - Once the unit under test is inserted, the test is started by pressing an appropriate button on the interface.
- **Run Pre-tests**
  - A number of tests are sequentially run against the UUT, (testing voltages of power supplies, checking oscillators etc), and once all the pre tests are run, the test jig sends a message to the test jig PC that it is time to program the unit under test.
- **Program**
  - The UUT is programmed by the test jig PC using the JTAG interface.
- **Run Post-tests**
  - Once the unit is programmed through the PC, a second set of tests are run that checks all of the functionality of the unit (can data be written and read to the memory, can serial data be sent and received etc.) and once these tests are passed, a result is displayed to the operator
- **Result**
  - A station pass label is printed for the unit. If at any point in the process the unit fails, then the operator is informed, a label indicating what test has failed is printed so that the unit can be fed back into the system for rework.
- **Store Result**
  - A log is stored of the results attained.

- **Remove UUT (operator action)**
    - The tested board is is removed from the test jig and sorted according to the printed label.

The test flow only requires the operator to perform 3 actions nl.:

- Load the board to be tested
- Start the test
- Remove board and sort

If there are additional actions required of the operator in an implementation of a test jig, then they should be of a similar nature.

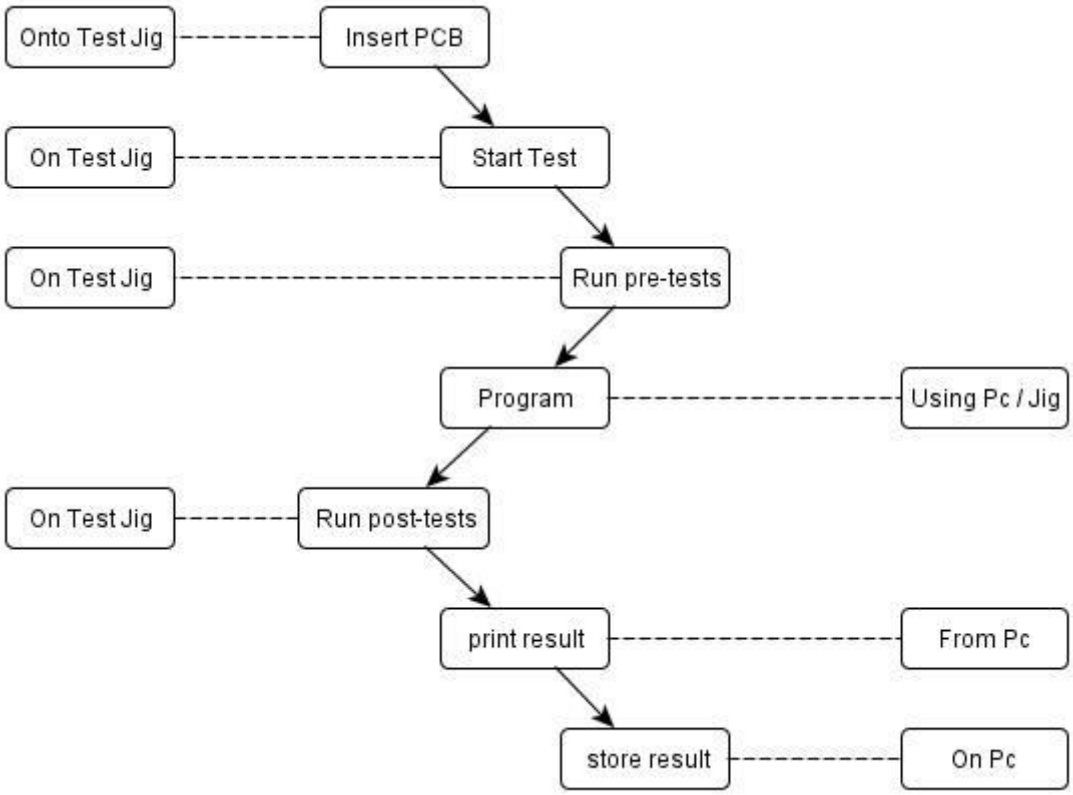


Illustration 7: The standard test flow

2.7 PRODUCT SPECIFICATION

Using the requirements and restraints placed on the design above, below is a preliminary specification of the system:

Test Jig Functional Specification:

- The test jig must be able to measure at least 10 binary inputs.
  - Input should be able to tolerate voltages up to 20V
  - Input impedance of > 100kΩ
- The test jig must be able to measure are least 10 voltage inputs.
  - 1% or better accuracy
  - Input should be able to tolerate voltages up to 20V
  - Input impedance of > 100kΩ
- The test jig must be able to measure are least 4 frequency inputs.
  - Must be able to measure square wave frequency up to 2kHz
  - Input should be able to tolerate voltages up to 20V
  - Input impedance of > 100kΩ
- The test jig must be able to measure are least 2 digital counter inputs.
  - Must be able to count signals of up to 2000 pulses per second
  - Input should be able to tolerate voltages up to 20V
  - Input impedance of > 100kΩ
- The test jig must have at least 2 serial communication lines to the unit under test.
  - Standard UART communication is 115200 baud, 8bits, no parity, 1 stop bit
- The test jig must be able to inject 5 or more signals.
  - Signals up to 1kHz must be able to be injected
  - Injection lines must be able to handle 230Vac
  - Injection lines must be able to handle up to 1A continuous

Electrical Specifications:

- The test jig will be supplied with 230Vac mains voltage to power it, and must require no external power supply.
- Power to the test jig will be supplied via a standard IEC power connector.
- The test jig, when not injecting signals must draw less than 2W of power

- Isolation between the UUT and all external communications lines must be > 1kV
- Input impedance on all inputs must be > 100kΩ
- All inputs must be 20V tolerant
- The test jig must be fused
- No Voltages should be present anywhere a user can physically access during testing

Interface Specifications:

- Test result storage must be non-volatile and be able to store at least 10 000 results.
- Test results need to be accessible to a windows or Linux operating system
- Once Installed, the required user interaction should consist of no more than single actions, for example: close lid, push button, scan label etc
- RS232/RS485 communication will take place at 115200 baud, 8 data bits, no parity bit, 1 stop bit
- Graphical communication to the user must all be in English

2.8 CONCLUSIONS

Using the literature survey and the user requirements specification set up earlier, a proposal was put forward for a design framework that should address the problems as set forth in the problem statement. The design will use a standard PC to run an application that controls a standard test jig board that is generic for all of the test jigs created within the framework, which in turn interfaces to a stack of 4 boards that create the bed of nails. This assembly will fit inside of a standard enclosure which will allow for the test jigs to be hot-swappable as well as giving them a standard interface that an operator can use to initiate the tests. Using the iterative approach, this design will be built up in stages and should, if implemented correctly address all of the needs of the relevant stakeholders.

CHAPTER 3

3.1 INTRODUCTION

In this project, the design methodology followed will be an iterative design process, which will include a substantial testing component during the development cycle, as well as a testing phase at the end of development to ensure that the best solution is come to.

The design is to be implemented with a specific guinea pig board in mind, so as to test all parts of the process. The end result should be the generic test jig framework, as well as the jig for the guinea pig board. A second jig will also need to be implemented to ensure that the “speed” requirements of the project have been fulfilled.

As the design process was followed through the individual iterations, at the end of each iteration, the interested parties were consulted (as they were available) to ensure that the project was still on track for what they initially wanted out of it in the design meetings. As time progressed, a number of changes were made to the way in which the test jig is designed, manufactured, used and tested, based in the input from the interested parties. In a product orientated environment, this design methodology that continually takes feedback from the stakeholders is the best methodology to follow, as it doesn't necessarily deliver a product that matches exactly the initial design specification, but it does deliver the product that the stakeholders need and that is ultimately what is important.

In documenting the individual iterations, only those parts that are specific only to that iteration will be mentioned in this document to ensure there is no repetition. The final iteration will contain all of the design information for the system in its completed form.

3.2 THE PROTOTYPE

The initial prototype was based on the design concept outlined in chapter 2.

3.2.1 Prototype Characteristics

Board	Rev
Generic Base (Proto)	V1
MCU bed (Proto)	V1/V2

Table 2: Hardware designs associated with the prototype

The prototype is charaterised by the following main points:

- RS485 communication between PC and test jig
- operator interfaces to the jig using buttons
- LCD screen to give feedback to the operator
- test scripts are stored on the test jig on an SD card
- test results are stored on the test jig on an SD card
- a number of isolated ports and isolated power supplies
- isolated analog measurement – isolated port to an external chip
- fixed interface to a standard test jig bed of nails base.
- Debug communication is handled by the test jig
- configurable I/O ports (input/output)
- 2 processors with a inter-processor bus (explain how exposing the message bus works to create an inter processor bus.)

The main work done was to create the first hardware platform, as well as the first bed of nails stack. The initial port of the EOS operating system to the board was done, and All of the firmware to support the RS 485 communication , inter processor bus and basic I/O was implemented on the prototype. For additional information on the RS 485 and interprocessor bus firmware see the following files in Appendix B.2:

- drv_Rs485.h and drv_Rs485.cpp

- brg_Rs485.h and brg_Rs485.cpp
- brg_processor.h and brg_processor.cpp
- brg_processorMaster.h and brg_processorMaster.cpp
- brg_processorSlave.h and brg_processorSlave.cpp

This iteration also allowed for the testing of the initial design process to create a set of test jigs and the code to support it.

3.2.2 Detailed Design Description

For the full schematics and gerbers of the Generic Base prototype board V1, see Appendix B.1.

The design of the generic base board can be broken up into the following parts:

- Power supplies
- Controllers
- Isolated bi-directional ports
- Analog
- UUT comms
- Storage
- LCD

Each of the individual sections are described below.

3.2.2.1 Power Supplies

The power supplies on the generic test jig board are made up of 4 isolated sets of +3.3V and 5V power supplies. The sets are isolated so that difference Analog front ends and bi-directional ports can interface to UUT that have different floating ground potentials. Three of the isolated power supplies are used to connect to the UUT, and the fourth is used for the on board controllers and communication. Using the FEA (Front End A) supply as an example in the illustration below:

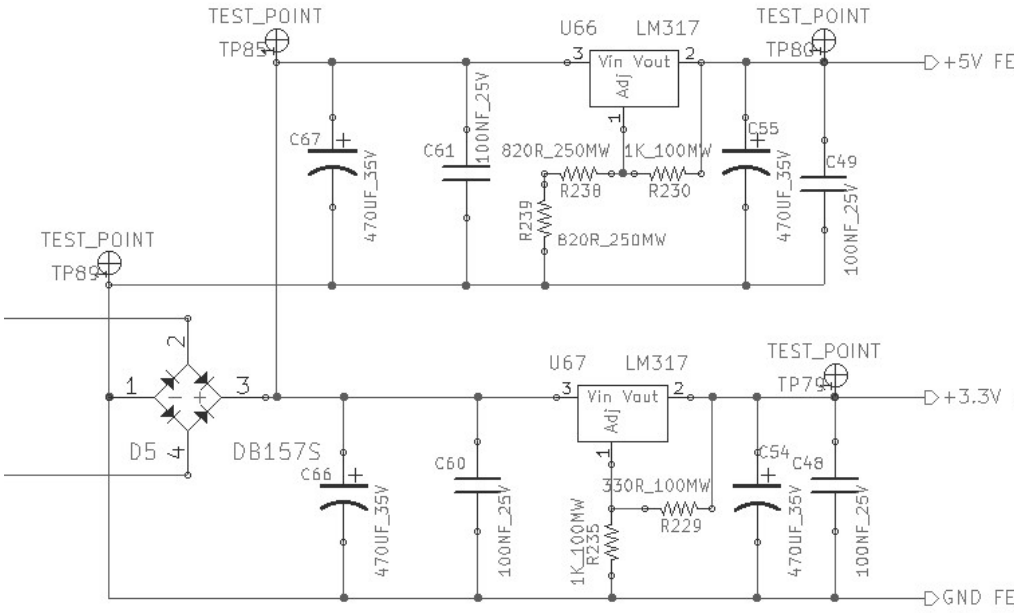


Illustration 8: Isolated +3.3V and +5V power supply set

Each isolated power supply set is created using an isolated secondary 12Vrms winding from one of the two Zettler BV302D12028 transformers (L1 and L2), and then feeding the voltage into a DB157S/B380S bridge rectifier (D5) to get a ripple DC voltage out with an approximate peak of 16.8V. The ripple DC is then smoothed out using two 470uF 35V electrolytic capacitors (C66 and C67) and fed into two separate LM317 (U66 and U67) linear regulators with two 0.1uF decoupling capacitors placed close to the regulator input pins (C60 and C61). The output voltage from the LM317 is determined by:

$$Vout = Vref\left(1 + \frac{R2}{R1}\right) + IadjR2$$

Where:

$Vout$ = *the output voltage*
$Vref$ = 1.25V *reference voltage*
$R2$ = *ground resistor value*
$R1$ = *output resistor value*
$Iadj$ = 100uA *leakage current*

In the above power supply for +3.3V FEA:

- R1 = R230 = 1kΩ and
- R2 = R238 + R239 = 820 + 820 = 1.64kΩ
- calculated Vout of 3.3V

and for +5V FEA

- R1 = R229 = 330Ω

- R2 = R235 = 1kΩ
- calculated Vout of 5.04V

Output ripple is controlled with two more 0.1uF decoupling capacitors (C48 and C49) and two more 470uF 35V reservoir capacitors (C55 and C54).
Test points are placed to measure +5V output, +3.3V output, Ripple DC input and GND using test points TP79, TP80, TP85 and TP89 respectively.

3.2.2.2 Controllers

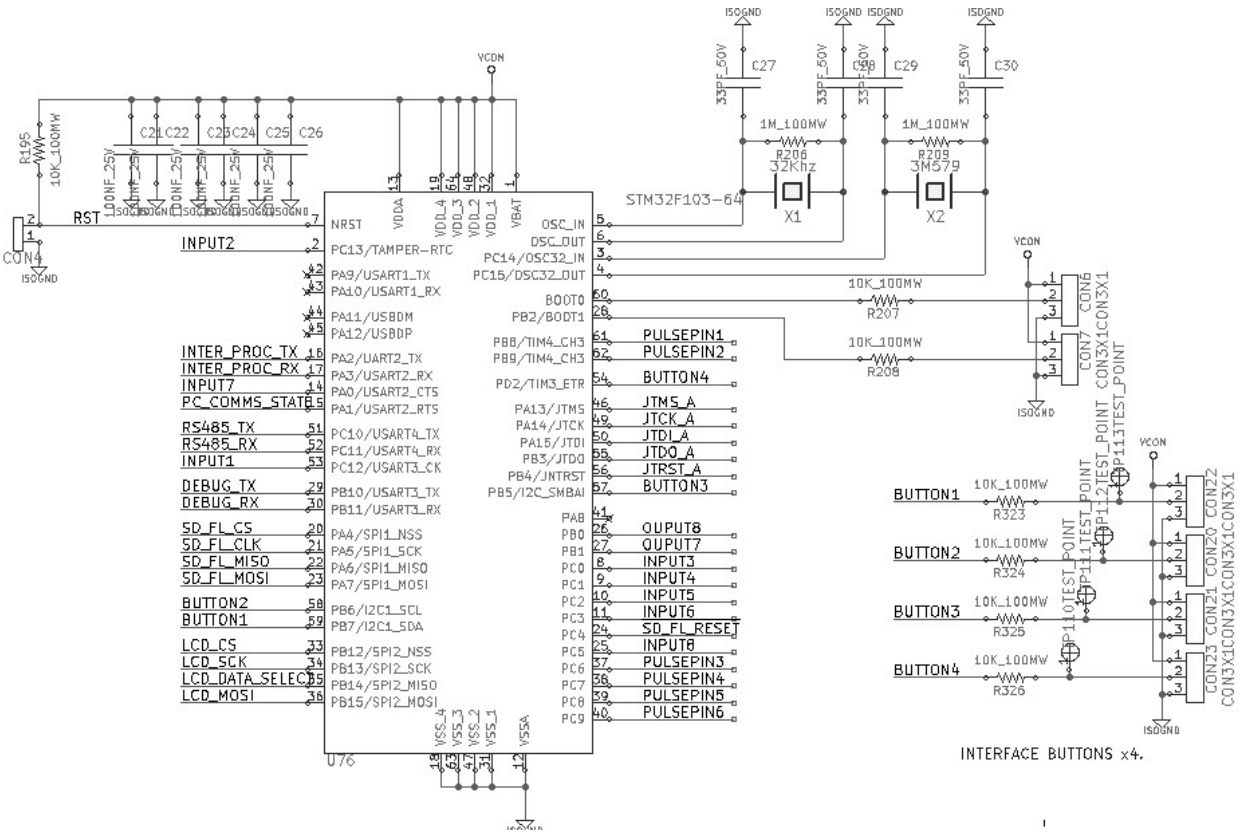


Illustration 9: Master Processor schematic

The controller chips are two STM32F103 64 pin ARM 7 cortex processors. These two processors control all of the generic test jig functionality. The two processors are designated master (U76) and slave (U75). The Master processor is responsible for RS485 communications with the Host PC using the RS485 driver chip (U58), interfacing to the user through the 4 SPDT switch connectors (CON20, CON21, CON22 and CON23), writing to the LCD through the SPI2 port, writing to and reading from the SD card and measuring signals on pulsed inputs. The Slave processor is responsible for measuring analog voltages using the isolated SPI port to the AD7918 ADC chips (see section 3.2.2.4). The processors share the responsibility of measuring the isolated inputs and outputs.

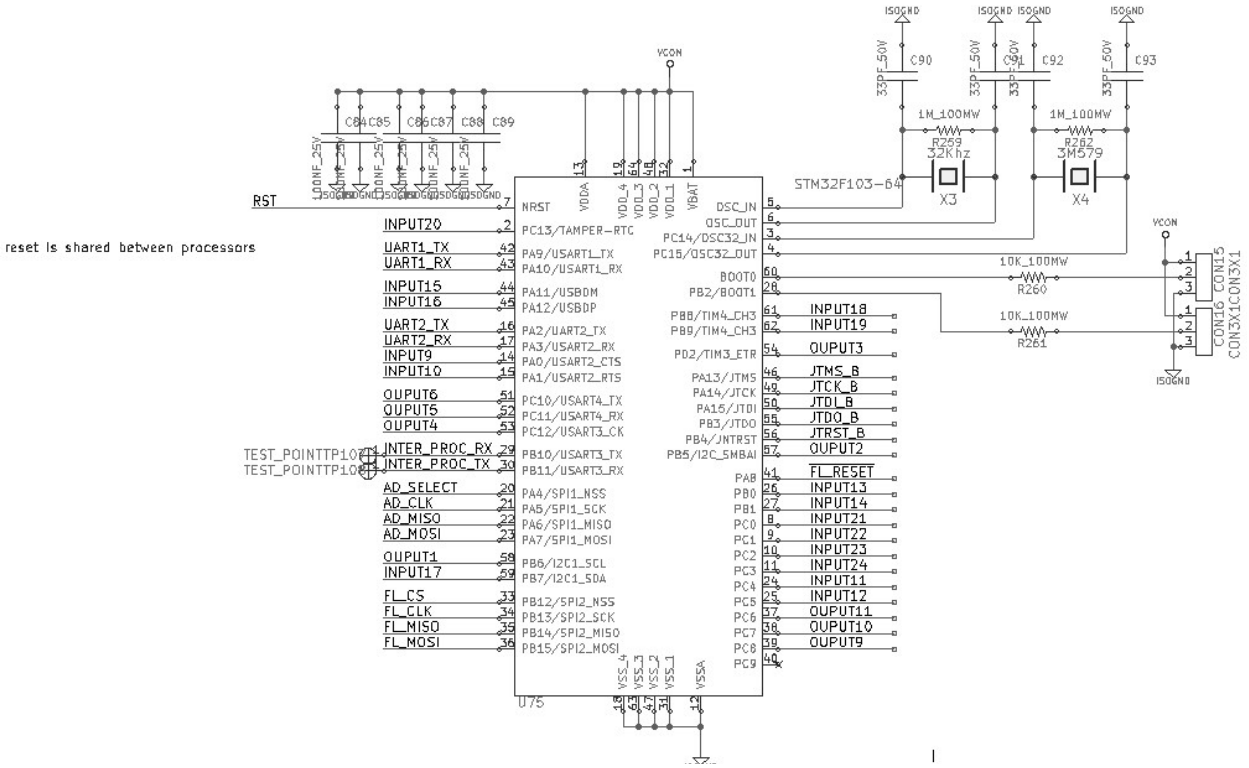


Illustration 10: Slave Processor Schematic

Each of the processors have their own 32 kHz watch crystals (X1 and X3) and their own 8MHz clock crystals (X2 and X4). Programming (and debugging) of both of the processors is done via JTAG interfaces on P1 and P6:

In this set up however the pull up and pull down resistors are optional, as the STM32 processor has internal pull-ups and pull-downs on the JTAG port [10, page 1050 section 31.4.3]

The boot state of the processors can be set individually using jumpers (CON6,CON7 and CON15,CON16). To boot from internal flash on both processors these jumpers all need to be set to GND (shorting pin 2 and 3). If the BOOT0 pin is set low (CON6 and CON15) then the state of BOOT1 is irrelevant and the processor will boot from internal flash [9,page 6 table 1], but the jumper is placed on BOOT1 to allow provision for times when the processor should not be booted from the internal flash.

The power supply rails to the processors are decoupled using 0.1uF capacitors (C21-C26 and C84-C89).

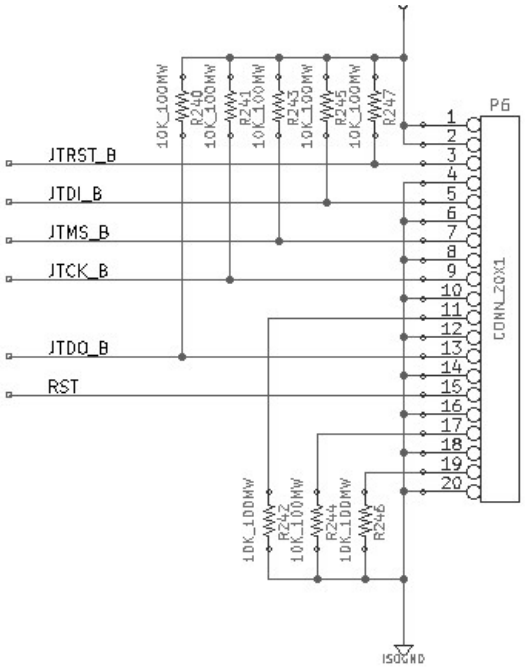


Illustration 11: Jtag Interface

The two processors act as a single unit, since their internal message busses are virtually tied together using the inter processor bus (Signals INTER_PROC_RX and INTER_PROC_TX on test points TP107 and TP108 respectively). To understand how this works some understanding of the nature of the EOS message bus is required (information on the EOS operating system is available in Appendix G).

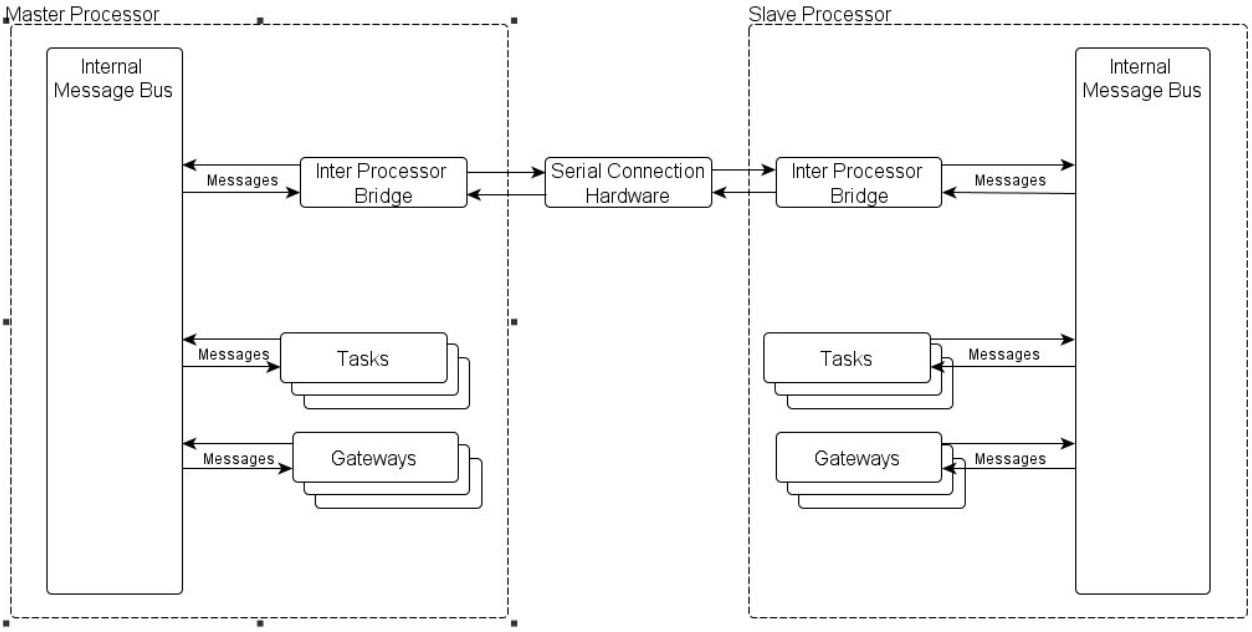


Illustration 12: Inter Processor Bus

In this network bridge implementation, the internal message busses of the two processors are tied together via a UART on each processor controlled by an Inter Processor bridge that just reads messages on the message bus it is connected to, filters them and sends them to the other processor. This means that messages sent by tasks and gateways on the slave processor can be seen by tasks and gateways on the master processor and visa versa (provided they do not get filtered by either bridge's filter method). This system allows the two processors to react as 1 processor sharing all of the processing tasks between them. The reason this system was implemented on the prototype was to allow for the sheer number of input and output pins required.

Communication on the prototype was to be done using a standard MAX485 RS485 driver chip to the host PC's RS485 network.

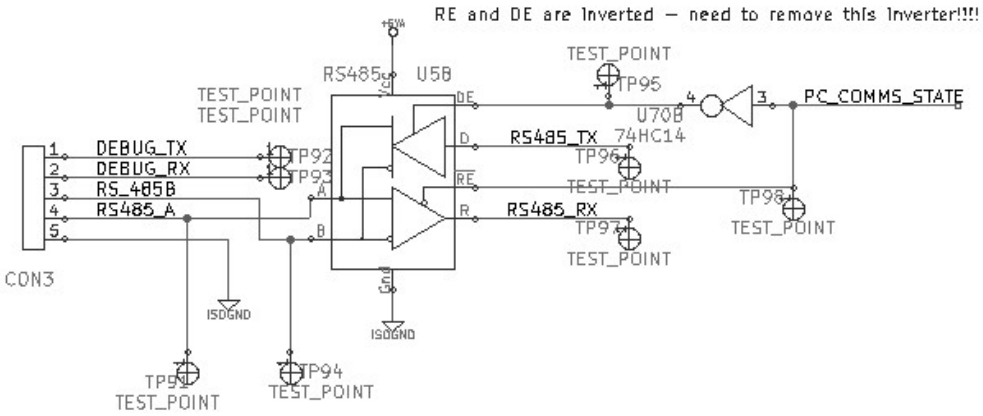


Illustration 13: RS485 driver chip schematic

The RS485 chip (U58) works like a normal RS232 receiver when in “receive” mode (by pulling PC_COMMS_STATE low) which allows the host PC to broadcast messages on the RS485 bus, and also to address individual test jig systems. When a request is sent to a specific test jig system, then the protocol will allow that test jig time to then be the transmitter on the bus and respond. To respond, PC_COMMS_STATE is pushed high to enable the data on the RS485 driver and put it in “send” mode and then data is sent as with a normal RS232 connection. In Version 1 of the prototype there was a mistake in the circuit diagram as port 2 of the 74HC14 inverter U70 was used to invert the PC_COMMS_STATE signal to allow the driver chip to be either in “send” or “receive” mode, but the RE line is already inverted, so the line needed to be cut and the inverter bridged out. The impedance of the max485 rs485 receiver is only ¼ of a normal receiver (48kΩ as opposed to 12kΩ), allowing up to 128 devices to be driven off of the same bus

[23, page 8]

Provision is made for an LCD interface to the user on CON1.

The LCD that provision has been made for Util Labs for general use and is a 128x64 graphical LCD with drivers already implemented in EOS across a standard SPI port.

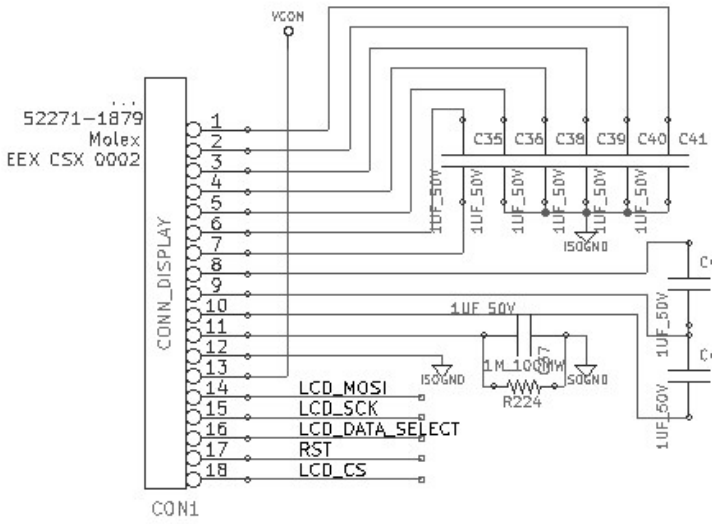


Illustration 14: Connector To The LCD

3.2.2.3 Isolated Bi-Directional Ports

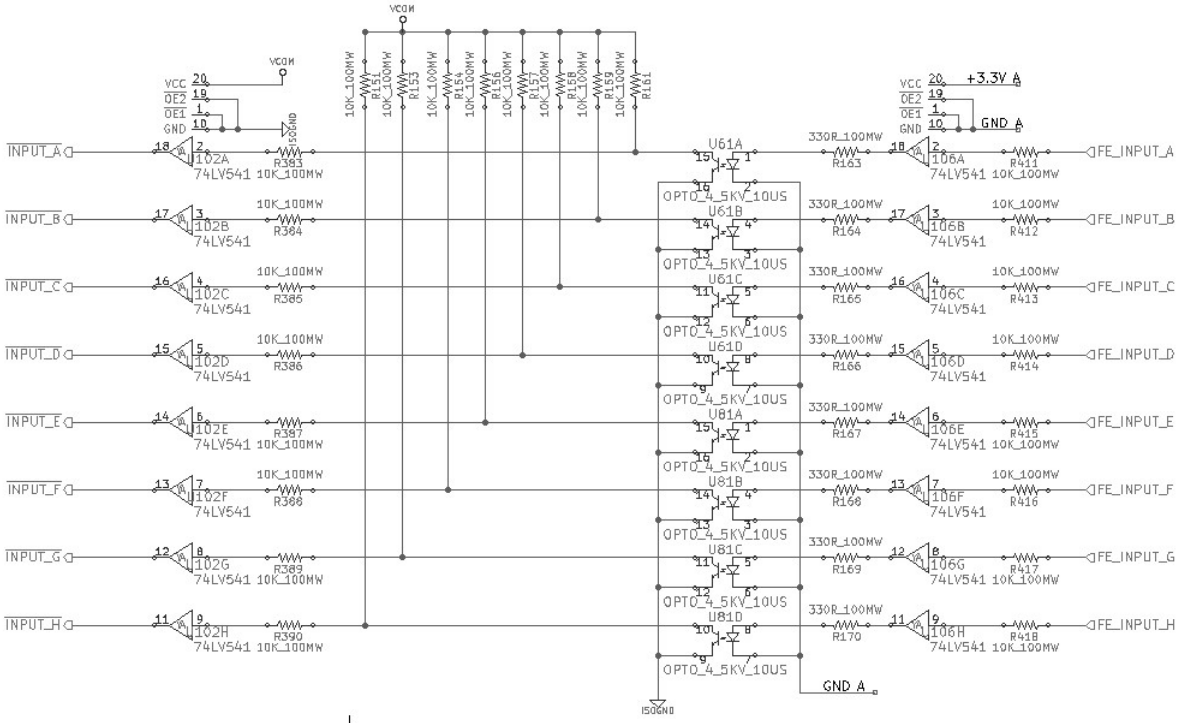


Illustration 15: Isolated Input Schematic

Because the test jig will be used to test PC boards that are often powered by mains voltages, all of the inputs to and outputs from the test jig need to be isolated (This will be a central theme in the design). The binary inputs and outputs on the test jig are isolated to

2.5kV using the CNY74-4 opto coupler. The inputs are buffered using a 74HC541/74LV541 non-inverting buffer on both the input to the opto coupler as well as the output. The input to the opto coupler is buffered so as to not put load on the UUT (the opto coupler is driven at 5mA, whereas the 74LV541 has a leakage current of 1uA [18, page 4 table 6]) and the output of the opto coupler is once again buffered as the opto coupler collector is limited with a 10kΩ pull up resistor to compensate for any aging effects in the opto coupler (the minimum Current transfer ratio for the opto coupler is specified at minimum 50% [20, page 2, table CURRENT TRANSFER RATIO], but over time aging in the opto coupler can lower this value [14]). Using this system however, the input value is inverted through the opto coupler and this fact needs to be taken into account in the firmware. The same logic is applied in reverse to the output circuitry:

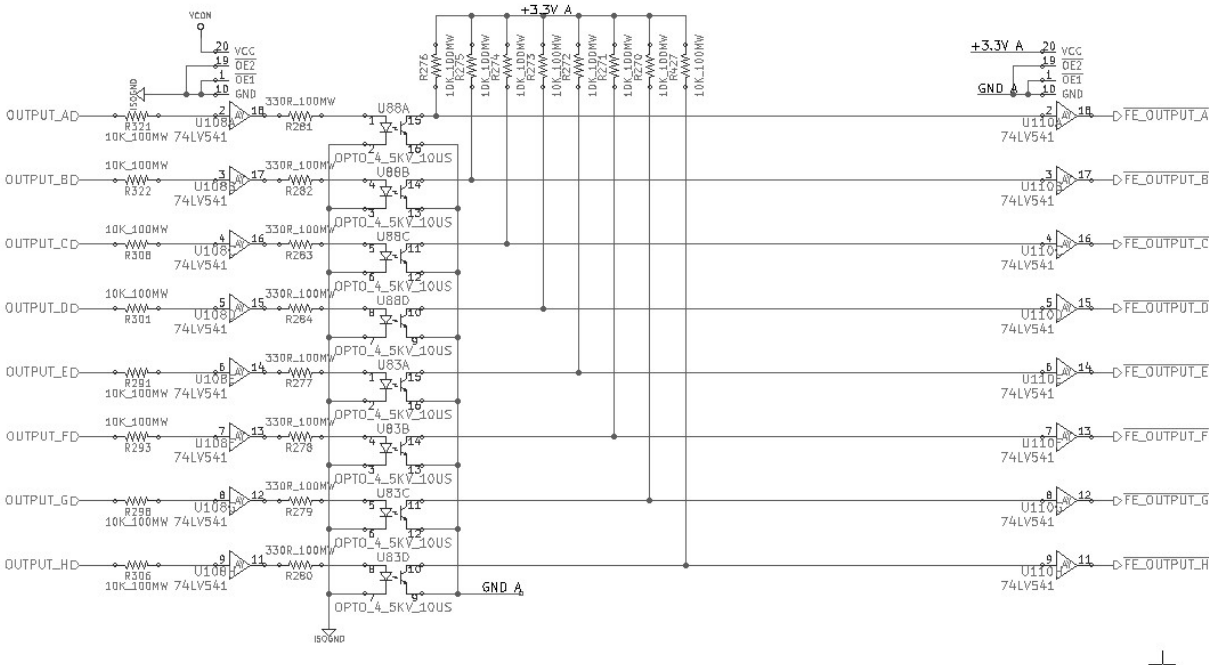


Illustration 16: Isolated Output Schematic

In the output circuitry, the exact same buffer chip is used to also drive the opto couplers (although the maximum current that can be sourced through the VDD pins of the STM32F103 chip is 150mA[19, page 41 table 8], with all of the outputs on the device driving opto couplers, this will cause excessive heating of the device without buffering) on the input side of the opto couplers, and then on the output side of the opto couplers the same buffering is used again to compensate for any aging effects. As with the input circuitry, the logical level of the output will be inversed through the isolation circuit, so this needs to be taken into account by the firmware.

The prototype board has 24 designated inputs to the test jig, 6 designated pulse measurement inputs and 11 outputs split across 3 ground potentials. These ground potentials can either be tied to the same point on the test jig stack if all of the signals on the UUT are referenced to the same ground potential, or they can be used individually as 3 separate circuits completely isolated from each other (and the test jig PC.)

3.2.2.4 Analog Measurement

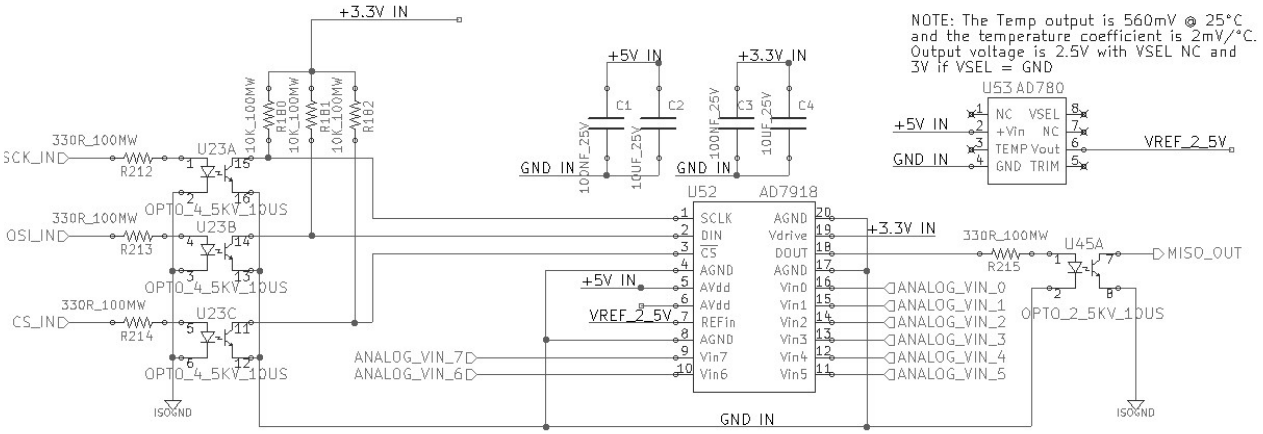


Illustration 17: Isolated Analog Front End

Analog voltage measurement on the test jig is achieved by floating an analog to digital converter chip and separate reference on one of the ground potentials to the UUT, and then isolating the SPI port communicating with that ADC chip. The chip in use is the AD7918 analog to digital converter from analog devices. The AD7918 offers 10 bits of resolution with a 0.5 bit error. Channel to Channel isolation is given as -85dB. The ADC converter has a low input impedance of 1uA and should not unnecessarily load the UUT [21, page 5, table 2]. The AD7918 will measure any voltage between 0V (the reference in this case that it is tied to on the UUT) and the supplied reference which in this case is fed from the AD780 reference chip. The output voltage used is 2.5V since the VSEL pin on the device is left not connected. The drift on the reference is given as 3ppm/°C [22, page 1]. A temperature measurement output is supplied on the reference chip via pin 3, but is unused since the ultra low drift is adequate for the accuracy required on the test jig.

On the prototype board a board modification was required to allow for communication to the chip as the opto couplers invert the signals on the SPI bus, so the opto coupler input tracks had to be cut and the pull up resistors changed to 0Ω links, and the SCK, MOSI and

CS lines had to be taken from the emitter connection on the opto coupler with an added pull down resistor on each line. The same had to be done on the MISO line, except with the modification being on the isolated controller side, rather than on the ADC side.

3.2.2.5 UUT Comms

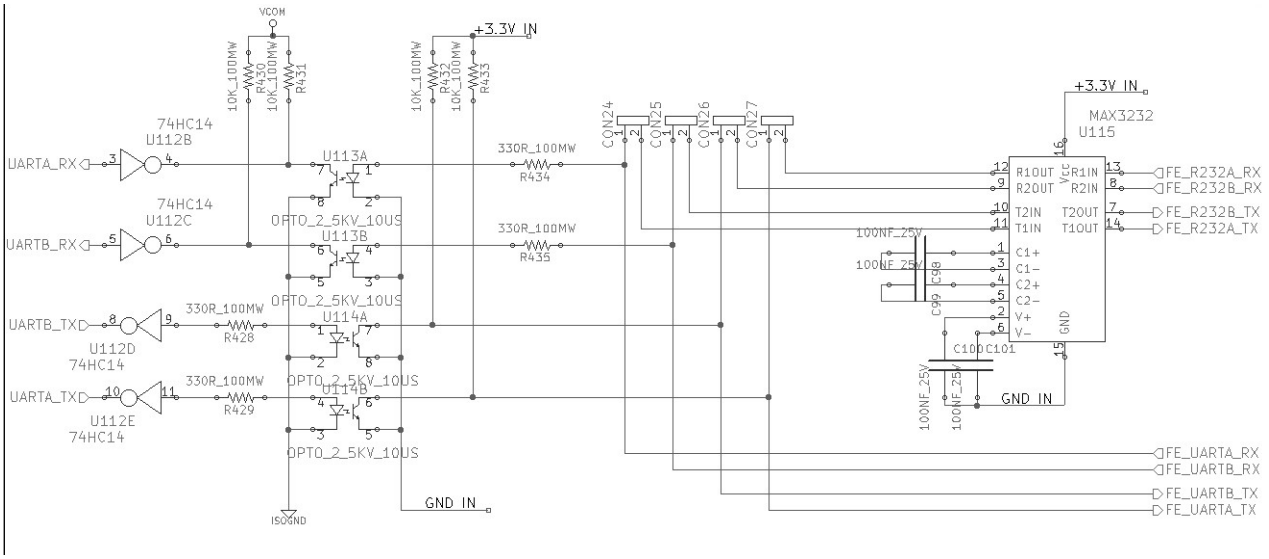


Illustration 18: UUT Comms Circuit

Comms between the controller and the UUT is isolated to 2.5kV through the opto coupler, both directly and through a MAX3232 RS232 converter to allow for but RS232 communication as well as standard UART communication. Two ports are allowed so that debug communication can be used in addition to serially bootloading the UUT if required.

3.2.2.6 Storage

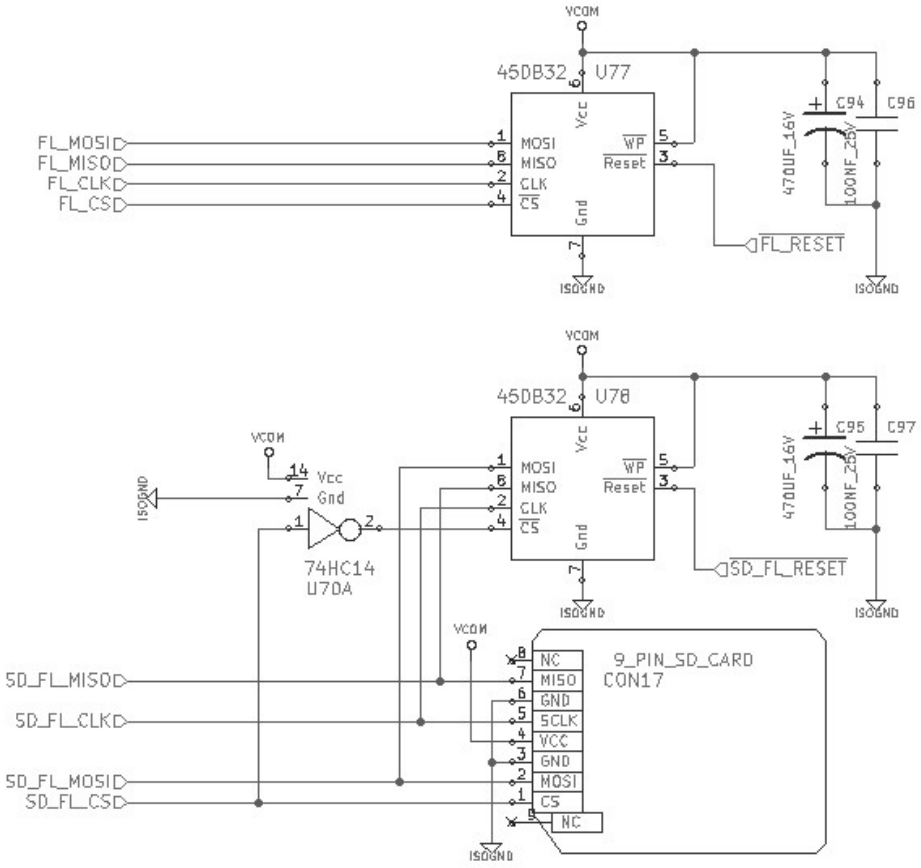


Illustration 19: Storage on SD card and Flash

Part of the requirements for the EOS operating system used is a Flash file system that is implemented on the AT45DB32 32Mb serial dataflash so settings can be stored. Provision was made for a standard SD card holder to allow for the settings, logs and test scripts for the test jig system.

3.2.3 The Test Jig Creation Process

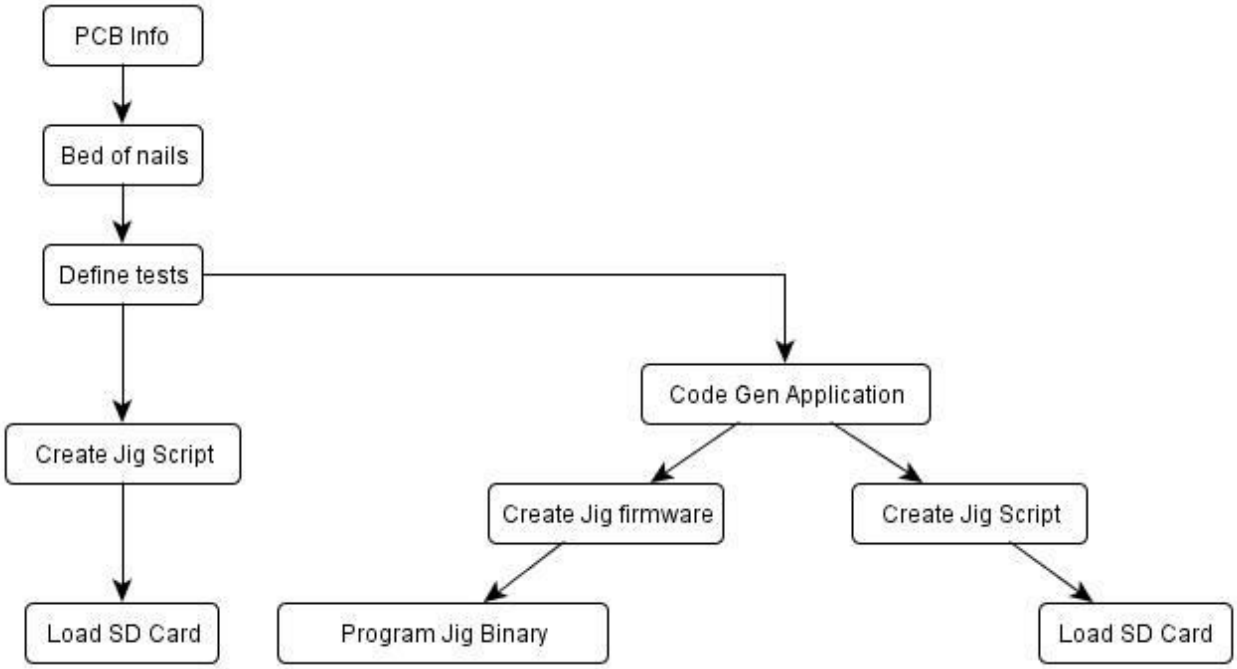


Illustration 20: Test Jig Design Process

The test jig creation process follows a number of predefined steps. On the prototype, the full system was not yet formulated, but the initial test jig board stack creation process was explored. Once a test jig stack was created from the design files of the unit to test, then the test sequence for that unit could be determined. Once the test sequence is defined, it was envisioned that the test sequence could be either coded manually in a scripting language and loaded on the SD card provided on the generic test jig board or the defined tests could be loaded into a code generating application to generate a binary file that could be bootloaded onto the generic test jig, or could generate the test script automatically. Both of these system were however later abandoned when the test sequence was incorporated into the Java swing framework (described later in this chapter). For the prototype initially the option of using tests pre-compiled into the test jig binary files was used.

The initial process for creating a test jig stack is identical to what was defined in section 2.3.4.

3.2.4 Design Review

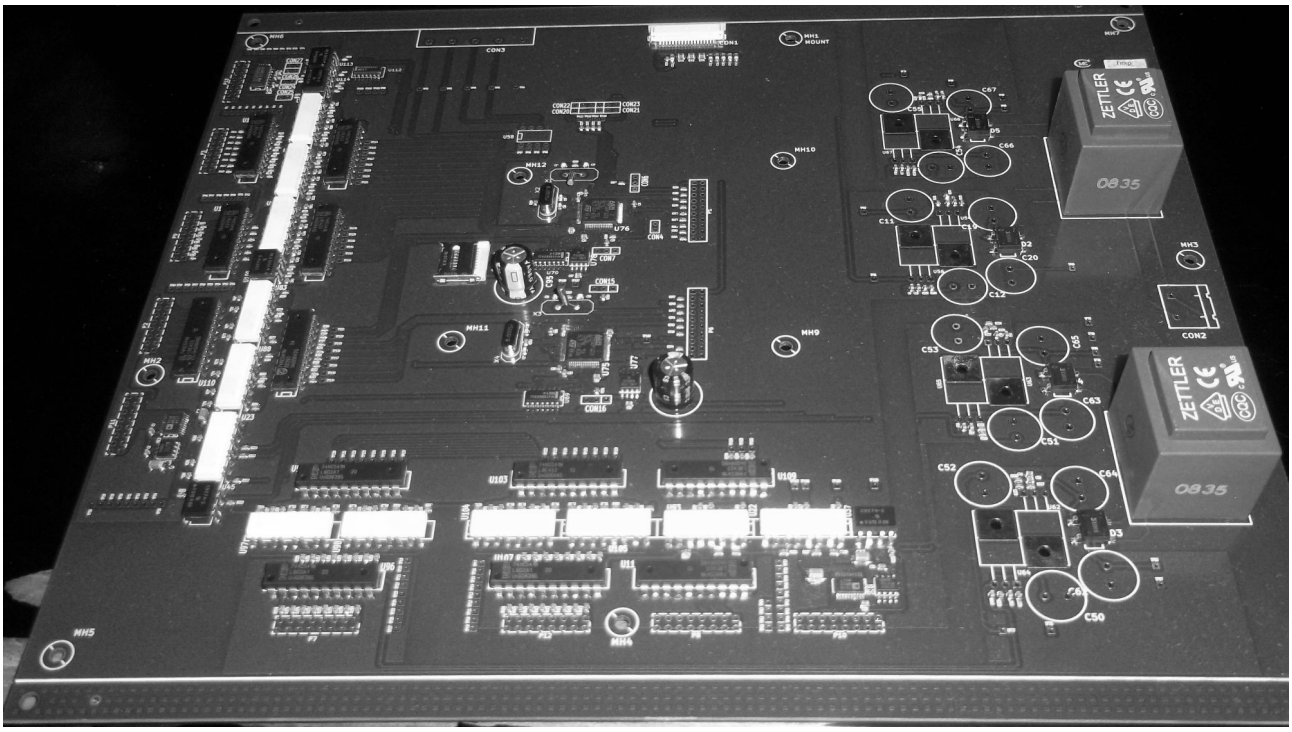


Illustration 21: Partially populated prototype board during test

Some severe problems were found with the initial prototype design, not least of which was the physical size of the test jig board at 295x235mm, as well as the physical size of the test bed of nails base at 800x400mm. There is also a very fixed interface between the two boards, making expandability very difficult. There were a massive number of changes suggested between the first prototype version and the first implementation of the test jig. These changes are discussed below:

- **change the isolation point in the design.**

  Problem: In the prototype isolation was achieved using a very large number of opto couplers. The reason for the sheer number of optical isolators (and support logic chips for the directional selection) is the lack of expandability of boards – the test jig needs to make provision for the maximum number of inputs and outputs (or any combination thereof) for all conceivable designs. This sheer amount of I/O required necessitated the use of 2 processors as opposed to 1 (the processor chosen at the start of the process was one that was already stocked by UTIL LABS, and as such the benefits of economies of scale meant that it was in fact cheaper to use 2 of these processors, than 1 larger processor).

  Solution: By shifting the isolation point to behind the RS232 comms port between

the PC and test board using iCoupler technology, a huge saving in logic pins could be made. This was the most significant change in the design, and was to be the preferred form of communication between the test jig and PC from then on out. This shifting of the isolation point also allowed for the use of the processor's on board peripherals.

- **Add scalability, shrink board.**

  Problem: Trying to make provision for all eventualities by adding a lot of IO to the generic test board seemed like a good idea at the start of the prototype, but It was soon realised that there is no real gain in this approach.

  Solution: make the boards smaller and therefore more scalable so the correct number of inputs, outputs and ADC pins could be chosen for each situation by adding the right number of boards.

- **Connection Standardisation.**

  Problem: The risk of damaging the test jig unit, or some other unit by starting up into the incorrect state was deemed to be too high. The multiple interface points between the base and test bed of nails was a bad idea – its always better to keep the interface simple.

  Solution: The decision was taken to standardize the interface connector between the generic test jig board, and the test jig bed of nails. The interface connector was standardised to the 40 pin connector used till the final version, as opposed to the smaller “single port” ribbon cables on the prototype.

- **Mechanical strain point moved.**

  Problem: The mechanical strain that would be placed on the Base board when the UUT is pressed down would cause some flexing of the board, which could cause solder joints to crack and the test jig to become unreliable. using the Guide board just to keep the pins straight would place all of the strain onto the base board.

  Solution: Test point placement was re-evaluated after the prototype. The placement scheme would be to print a silkscreen version of the outline of the UUT on the test jig bed of nails base, and then position the test points on the exact positions that they would be on the UUT. This allowed test pins to be added to an additional stopper layer in the stack where they could be fastened, such that the stopper board would take up any bending due to the force exerted onto the test pins, and the test pins could be joined to the base of the bed of nails via short

jumper wires. This soldering of wires between the test pins and the base of the bed of nails would nullify the mechanical strain placed on the board.

- **Removed SD card.**

  Problem: The idea of issuing a test jig with an SD card seemed like a good idea, but the problem is that every time a new set of serial numbers needs to be issued from Util Labs to the factory, it would involve re-writing the card. This would mean that either a new SD card needs to be issued from UTIL LABS to the factory for each production run, or the serial numbers would need to first be loaded up to the test jig PC, and then sent down to the jig for storage. Additionally, the possibility of someone placing the wrong SD card into the wrong test jig, becomes problematic, especially since the initial idea was also to house the test scripts for the jig on the SD card as well. An advantage of having the serial keys and firmware on the SD card, was that it would be easier to have the test jig then program the UUT using a serial bootloader. This option however was not particularly attractive though, as all boards designed at UTIL LABS so far had exposed the JTAG port for programming rather than the serial port – and implementing a full JTAG interface on the test jig processor was deemed to bad idea since there was already an existing implementation on the test jig PC that could be modified slightly to work in the framework. Another problem with keeping all of the information on the SD card, is that it is a losable medium. If the SD card is swapped out during production runs (or between production runs) then all of the logging data would be lost.

  Solution: The solution to the problem is to keep the serial numbers and test sequences with the JTAG implementation, which is on the PC. The initial idea, was to have a generic test jig, which could then be fitted with a specific bed of nails/SD card combination, but the SD card offers no real value. By removing the SD card, the need for a fat file system is nullified.

JTAG	Serial bootloader
Needs additional hardware to isolate the JTAG port.	Easy to isolate since the processor is already on the same ground potential
An implementation already exists that can be used in the PC application	Would need to be coded from scratch
No redesign required on previous product at UTIL LABS	Would need some product to be redesigned slightly
Needs only the firmware versions and PC to work	If implemented on the test jig, then the SD card would be needed to house the firmware due to its size on some of the processors
	Risk of losing SD card could stop the production run and badly compromise serial number security

Table 3: comparing JTAG and serial bootloader programming on the test jig

- **Single JTAG per test jig as opposed to JTAG scan chain.**

  Problem: Since the logging, programming, and test script responsibilities were moved to the PC, The JTAG scan chain was investigated as a way of using a central PC to communicate with multiple jigs, but this poses the problem of synchronisation, as only 1 device can be programmed at a time with the JTAG interface, so each of the tests would have to be aware of the status of the other tests, which is overly complex. Additionally by using the JTAG scan chain, the wiring between test jigs become overly complex, and limits the ability of the jigs to be moved around and reconfigured. Isolation between the different jigs is also of concern when running with the scan chain as it is simple to isolate the RS 485 bus, but due to the speed it runs at, isolating the JTAG scan chain become significantly more complex.

  Solution: It was with this in mind, that the RS 485 and scan chain bus were dropped in preference to using low cost pcs that each have a JTAG key device and RS232 assigned to it. This simplifies the logging, programming and all communication problems between jigs, and the cost was deemed to be acceptable for the gain offered.

- **Electrical Protection.**

  Problem: Electrical protection was a problem in the prototype, as the neither the

CMOS inputs or the ADC inputs were protected.

Solution: With this in mind zener diodes and series resistors are added to all input lines.

- **Test Sequence and installation**

  Problem: Initially it was proposed to use scripts to set what would be tested on each test jig, However, with all of the changes proposed above, the test script implementation seemed to be a less attractive solution as it would require the operator to be aware of the test scripts on installation of the jig and he would need to ensure that the correct script is run on the correct jig (if an output that is normally connected to mains for example was switched at the wrong time it could cause massive damage).

  Solution: A better solution is proposed by allowing all of the possible tests to be housed in a single jar file written in Java and using the spring framework add a GUI to it. This way, by adding a few select lines to the bed of nails base, the appropriate jig behaviour can be chosen in software from a list of different test jigs automatically. This way, there is no need by the operator to install the correct script, as It will already be a part of the framework that is required to run the jig. Configurable parts of the system can be added to the .prop resources file. An additional advantage to having this single jar file approach, is that in the event of a new test jig becoming available, if the jig select lines do not conform to a known jig for the version of the jar file on the test jig PC, then the program can display an error instead of running ahead preventing all damage to the jig from incorrect software set up.

3.3 THE SECOND ITERATION – FIRST PRACTICAL IMPLEMENTATION

The second iteration marks the change in the way the test jig framework hardware is organised. From this revision, for each test jig stack there are two (or more) of the generic test jig boards that will perform all of the testing on the UUT. The number of inputs and outputs required will determine the number of boards required. This iteration also marks the first attempt at implementation of the test jig framework to create a test jig for a guinea pig board (the MCU V3 PCB). This guinea pig board will continue to be used in future iterations to ensure that the jig creation process is as simple as possible. Since this was the first board explored with the framework, two new tests that were not anticipated in the original proposal were added, namely the ability to measure light and the ability to test for short and open circuits on the unit under test. In order to tell the test jig stacks apart, select lines were added, that allow each stack of base, stopper, guide and outline boards to have a unique id. The iCoupler isolated RS232 drivers have also been added to isolate the test voltages from the PC through the single isolation point. All of the I/O between the generic test boards and the test jig stack is now handled through 2 (or more) 40-pin ribbon cable connectors. The way test points are handled has been changed significantly by moving them from being mounted on the bottom base board, to the stopper board. By moving these test pins, the mechanical strain is removed from the base board where all of the electronic components are mounted for the test jig stack, and places all of the strain on the stopper board. The connections between the test pins and the base board are done by using solder type test pin holders, attaching wire leads to the test pins and then soldering these wire leads to the base board. Any strain now experienced due to the downwards force from the unit under test will warp the stopper board (which has no soldered components on it), but leave the base board undamaged. The impedance test capabilities of the test jig stack are achieved by placing a low voltage source in place of mains source to power the unit under test and then using the voltage drop across a series resistor to determine how much current is flowing through the unit under test. The optical measurement is achieved by using a light dependent resistor and a Wheatstone bridge measurement circuit to determine if light is present or not. Since the base design of the the hardware for the generic test jig board is similar between iteration 2 and iteration 5 of the design, the hardware will be discussed in iteration 5 in detail.

3.3.1 Iteration 2 Characteristics

Board	Rev
Generic Test Jig	B
MCU bed of nails base	A
MCU bed of nails stopper	A
MCU bed of nails guide	A
MCU bed of nails outline	A

Table 4: Hardware designs associated with iteration 2

- RS232 communication between PC and test jig
- operator interfaces to the jig using the PC
- PC to give feedback to the operator
- test results are stored on the PC
- Debug communication to UUT is handled by PC
- Test application determines the tests to run based on the bed of nails attached to it – automatic configuration for all types of jig in a single jar
- Single points of isolation – RS232 isolated by iCoupler device
- Measurements float on the same ground potential(s) as the UUT
- fixed interface connector to base of bed of nails, but more expandable as more connectors can be added for more generic test jigs beyond the standard 2
- fixed number of inputs/ outputs per generic test board – scalability and configuration achieved by adding more boards if required beyond the standard 2
- single processor on the generic test board and measurements done using the processor's on board peripherals
- protection on ADC and logic inputs achieved with series resistors and zener diodes.
- Additional measurement options are explored – LDR light measurement to measure led light added.
- Provision made for impedance test functionality.
- Select lines added to allow the test jigs to have ids.

The Full Schematics and gerber files for all of the boards in Iteration 2 are all available in Appendix C.

3.3.2 Test Jig Creation Process

The test jig creation process is different from the process outlined in the prototype in that not only has the hardware stack changed, but the way the tests are coded has also been changed. The process for creating the software on the jig in the second iteration is shown in the following diagram.

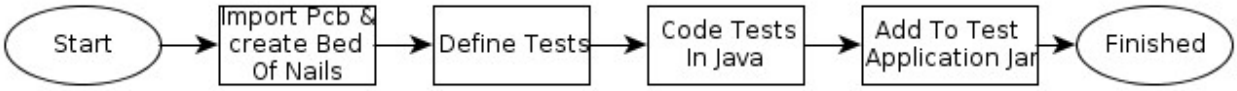


Illustration 22: Test Jig Creation Process

The jig creation process is now slightly simpler than previously envisioned. In this iteration, the process has been refined in that proposed templates for the entire stack have been put forward, and a number of standard tests have been developed in the Java application framework, which are supported by the firmware on the generic test boards. The test jig stack creation process has been changed to be as followed:

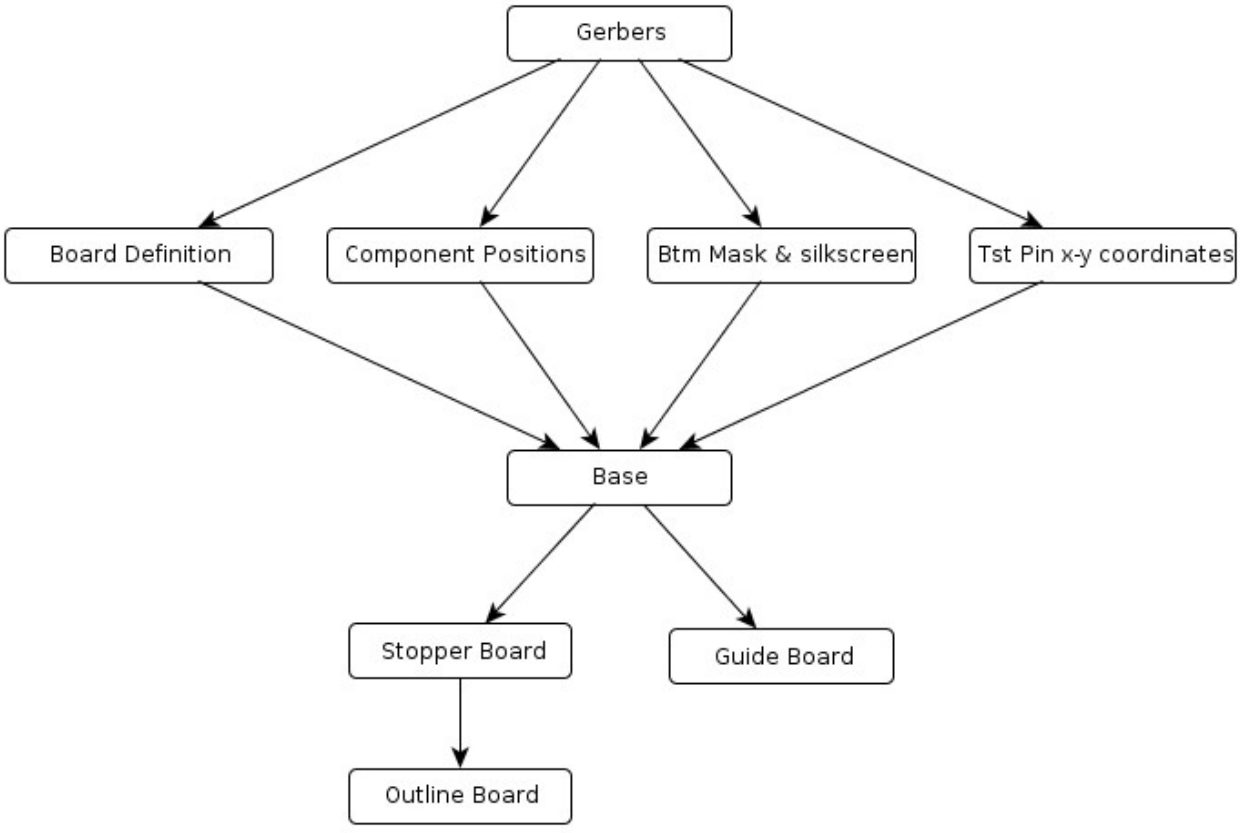


Illustration 23: Creating The Test Jig Stack In Iteration 2

- **Base Board**

  The base board is created by importing the unit under test's gerber files into the template and placing it in the center of the board. From this reference point, the outline of the unit under test's gerbers needs to be traced onto the base board, and test points placed for every point to be measured. These test points won't have the test points soldered to them, only jumper wires coming from the board above it (the stopper board). The base board is also used to do all of the signal conditioning for the test points.

- **Stopper Board**

  The Stopper board is where the test pins are attached to. From the bottom of the test pins, jumper wires are soldered, which then drop down and are soldered to the base board in the test point holes created earlier. To build the stopper board, the base board is just copied, all of the components removed, and the test pin holes are changed to 68 mil holes to hold the test pins. For the LDRs, jumpers need to be dropped down and connected to a second set of jumper holes which will house the LDRs on the stopper board.

- **Guide Board**

  The Guide board is what the unit under test rests on while allowing the test points and all other through hole components to push through it. It is generated by using the imported gerbers used in creating the Base board, and placing appropriately sized holes for each of the test pins and through hole component leads.

- **Outline Board**

  The outline board is created using the stopper board. A cut out is drawn around the silkscreen outline of the unit under test board, approximately 0.5mm bigger in each direction so that the board can drop in easily.

This design was first mocked up mechanically using cardboard and PCB standoffs to attempt to ensure it overcame the problems experienced in the prototype. The board used for the mock up was the generic test jig board itself.

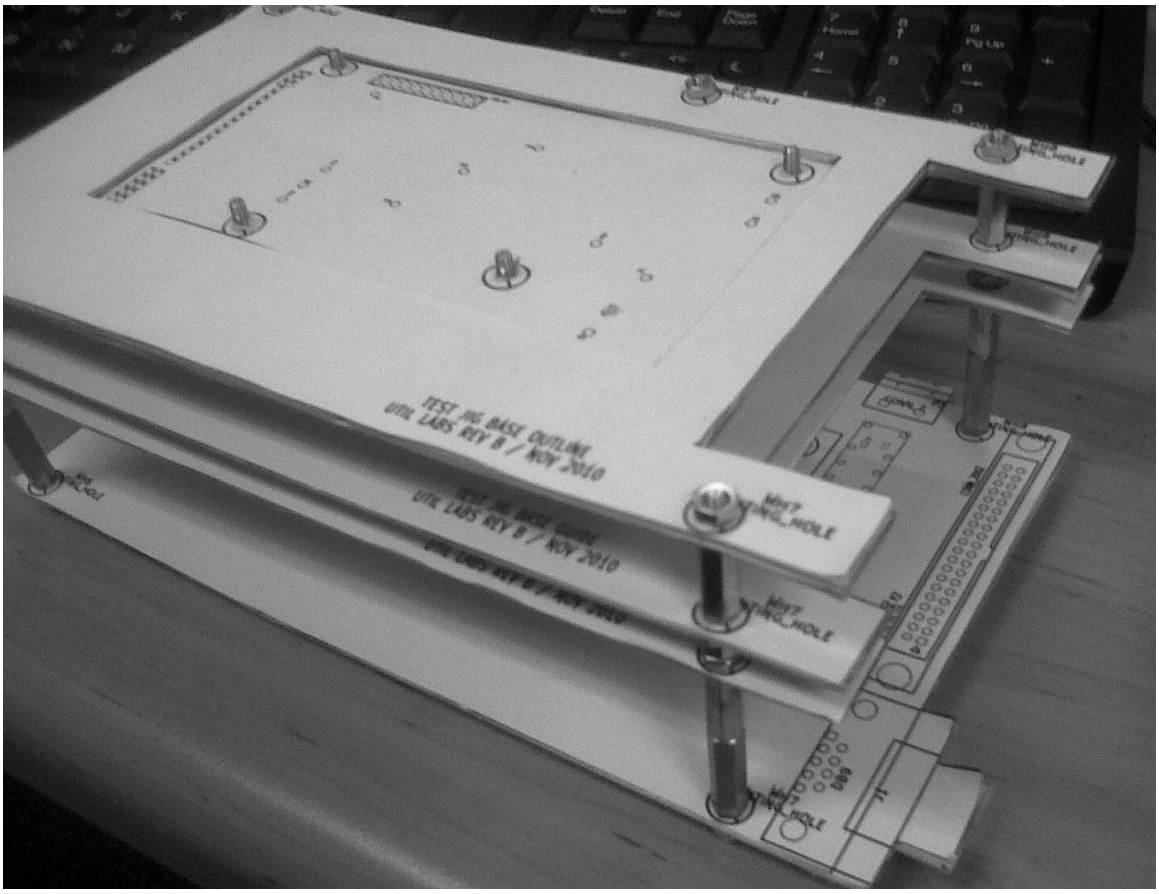


Illustration 24: Test Jig Stack Mock Up

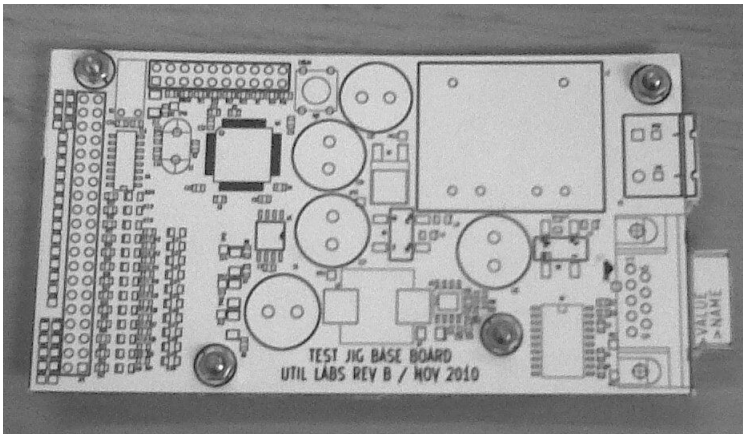


Illustration 25: Test Jig Generic Base Mock Up

3.3.3 Design Review – Changes Made To Iteration 2 To Create Iteration 3

The Changes between the prototype and first implementation were very large, and as such some problems creeped in.

- **Board Layout**

  Problem: There were some problems with the board layout:

  - on the power supply, the 12v regulator footprint (use the standard through hole version) was incorrectly specified as the D2PAK instead of the through hole TO220 in stock.
  - the iCoupler doesn't work – it needs either an additional 5V power supply on the input side, or a 3.3V supply on the output side [24, page 3 table 1]. By just running it on the 3.3V supply on the VCC rail, the iCoupler is run with the DC-DC converter on the isolated side disabled.
  - The transformer used is a little under powered, and runs fairly hot. Use a larger transformer. There is also no use in running two secondary windings on the transformer.
  - There is a problem in the kicad library, in that the db9 connector doesn't have the solder pads masked out. fix solder mask problem in the library.
  - There is a problem in the kicad library, in that the pin arrays (header pins) have pin 1 of the connector smaller than the other pins. Fix footprint.
  - The 0603 resistors used on the board are difficult to solder by hand, and due to the expected number of boards that will be made, it is likely that the boards will be soldered by hand. Change the 0603 resistors to 0805.

- **Test Pin Connection Between Base And Stopper**

  Problem: The initial implementation of the test pin connection between the base board and stopper board proved to be very very difficult to implement mechanically. Although good electrical contact was made between the UUT and the test pins, the physical act of wiring up the test pin holders to the base test pin solder holes made the design inviable, as to allow the base and stopper boards to be split, the wires needed to be <15cm, and this meant a huge amount of wire in between the base and stopper boards when the boards were bolted together. The same action that supplies the protection to the wiring (the fact that it is springy, and doesn't rigidly press against the base board), also proved problematic, as the test pin holders

were now being pushed up.

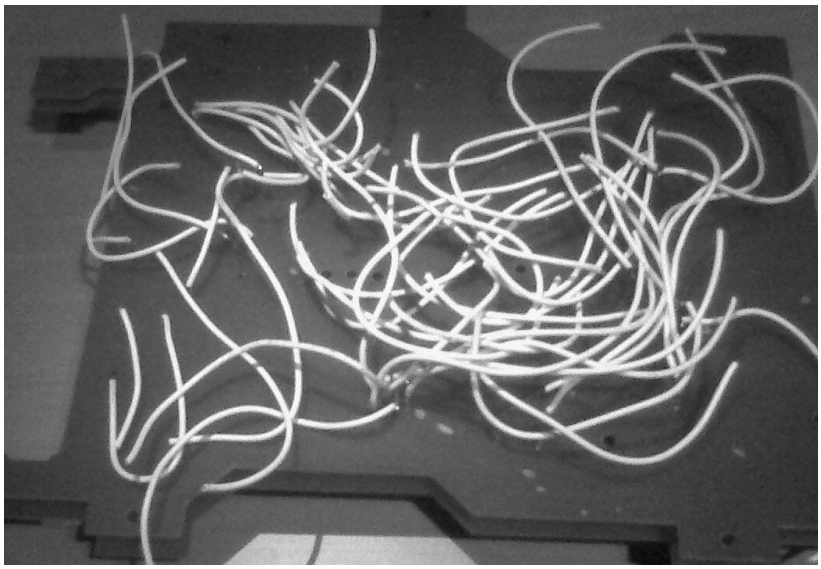


Illustration 26: Iteration 2 Implemetation Bottom

Solution: The solution to this problem was twofold – solder the test pin holders onto the stopper board, and instead of wiring from the back of the test pin holder to the base board, use an additional set of ribbon cable to wire between the base board and stopper board. Since the test pin holders would now be soldered onto the stopper board, a different solution would be required that would stop the board from flexing and breaking the solder connections. This is achieved by using 2 stopper boards, help apart by a number of 5mm spacers. The two boards have significantly more physical strength, and as such prove to not flex under the pressure of all of the test points.

- **Non-Linear Force**

  Problem: The initial idea was to have the test points push through the guide layer, and use bright bar (or similar) guides to allow the operator to push the board down with a clamp onto them. This method however proved very difficult, as it was not possible to ensure that the operator presses down 90 degrees onto the board, and any bending of the test pins cause them to press against the guide layer, and not be depressed properly and in some cases be damaged due to the physical pressure being more on some pins than others.

  Solution: The solution to this, was to rethink the way the pins connect to the UUT, by instead of making the operator apply downward force onto the pins, allow the operator to gently place the board onto the guide layer without the test points

present, using the outline layer as a guide, clamp it down and then pressing the test pins up into the testing position using a lifting plate arrangement. This single axis of movement for the test pins meant there is no chance for the pins to be bent or the operator to apply the pins incorrectly, as it would be applied with the pull of a lever.

- **Jtagconnection**

  Problem: The high speed lines for the JTAG programming need to have shorter track lengths to ensure proper data transfer from the JTAG to the UUT.

  Solution: The arrangement of having all of the connectors on the base board is unnecessary, the debug comms and the JTAG connectors no longer go to the base board and as such can be routed on the stopper board to keep these high speed lines much shorter.

3.4 THE THIRD ITERATION

The third iteration marks the change in the responsibilities of the boards in the test jig stack. The following responsibilities are assigned to each board :

- **Base :** The base board is responsible with the stopper board for the signal conditioning of the signals to and from the test pins. The resistor dividers, signal relays and low voltage sources are placed here.
- **Stopper:** The Stopper board is where the test pins are attached to. There is also some signal conditioning done on the stopper board in the case of the LDRs that sense the light signals from the board and their corresponding Wheatstone bridge circuit. There are two stopper boards placed one on top of the other with 5mm spacers in between to ensure mechanical stability of the board.
- **Guide:** The unit under test rests on the guide board during the test sequence once it has been clamped down. As such, the guide board has holes cut for all of the test pins to pass through it when the lifting plate is actuated, as well as having holes cur out for all of the components mounted on the bottom board and any through hole component pins to ensure that the unit under test rests flush on it.
- **Outline:** The outline board is used to position the unit under test on the guide board before it gets clamped down mechanically for the test.

The connections between the base board and stopper board are now done with the same type of ribbon cable as is used to connect between the generic test jig boards and the base board. This iteration was the first iteration in which the iCouplers were shown to be an effective isolation point, in that no damage was done to the PC that housed the testing software even when mains voltage was present on the test pins on the test jig stack.

3.4.1 Iteration 3 Characteristics

Board	Rev
Generic Test Jig	C
MCU bed of nails base	B
MCU bed of nails stopper	B
MCU bed of nails guide	B
MCU bed of nails outline	B

Table 5: Hardware designs associated with iteration 3

- RS232 communication between PC and test jig
- operator interfaces to the jig using the PC
- PC to give feedback to the operator
- test results are stored on the PC
- Debug communication to UUT is handled by PC
- Test application determines the tests to run based on the bed of nails attached to it – automatic configuration for all types of jig in a single jar
- Single points of isolation – RS232 isolated by iCoupler device
- Measurements float on the same ground potential(s) as the UUT
- fixed interface connector to base of bed of nails, but more expandable as more connectors can be added for more generic test jigs beyond the standard 2
- fixed number of inputs/ outputs per generic test board – scalability and configuration achieved by adding more boards if required beyond the standard 2
- single processor on the generic test board and measurements done using the processor's on board peripherals
- protection on ADC and logic inputs achieved with series resistors and zener diodes.
- LDR light measurement.

The full schematics and gerbers for all files in Iteration 3 are available in Appendix D.

3.4.2 Test Jig Creation Process

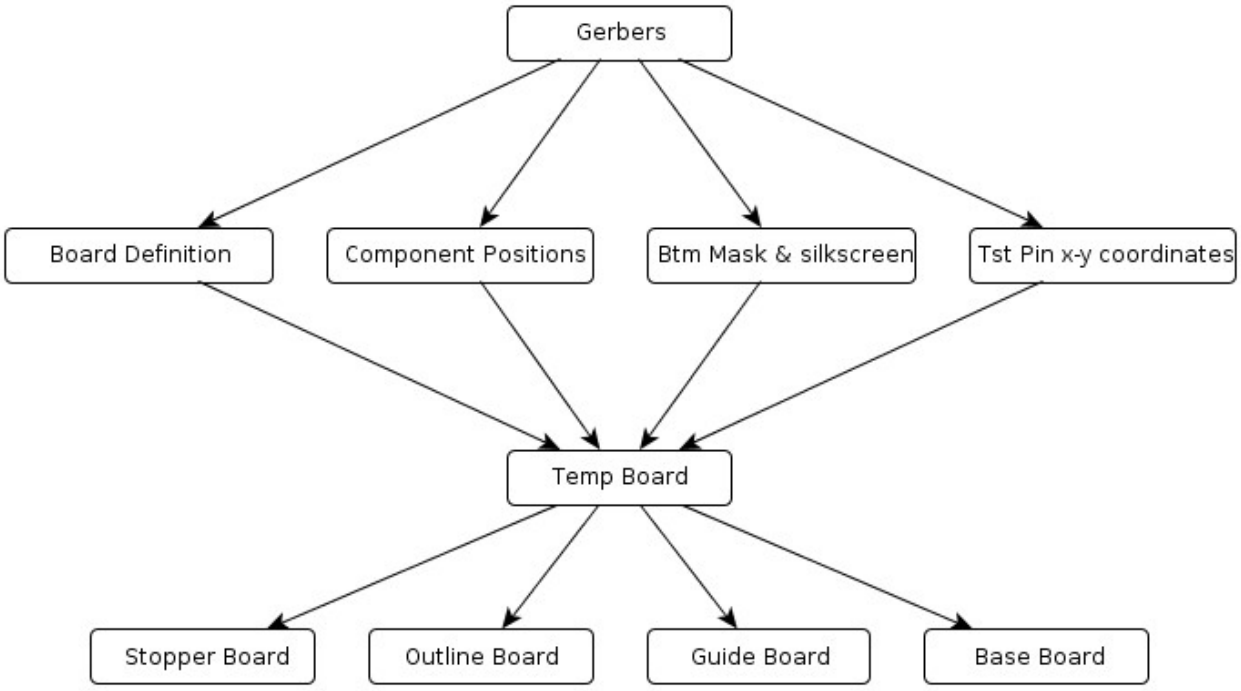


Illustration 28: Test Jig Stack Creation In Iteration 3

In Iteration 3, the use of the template is explored further and updated to take into account the changes in the responsibilities of the different boards in the test jig stack. The gerber files for the UUT are first pulled into a temporary board used for creating the other boards. The board outline of the UUT, test point positions, all through hole components and components mounted on the bottom of the UUT need to be taken into account. This information is placed in the middle of the Temp board, and then saved as temp-*boardname*.brd, where *boardname* is some identifier for the UUT. From the temp-*boardname*.brd file, all the other boards can be created. The schematics for the stopper and base board need to take into account all of the mounting holes on temp-*boardname*.brd, as well as the pre-placed connectors P1, P2, P4 and P5. The stopper board schematic also needs to take into account the select resistors R1-R10. In the test jig stack, 2 stopper boards are used with a number of 5mm spacers in between to give the boards mechanical rigidity. Place sufficient screw holes in temp-*boardname*.brd to keep the boards rigid when they are bolted together. As a rough guide, all of the test points need to be at least 1 inch away from a stabilising screw.

- **Base Board**

  The base board is created by removing from temp-*boardname*.brd, all of the components that were imported, as well as the select resistors R1-R10. This leaves only the inter-board connectors P1,P2,P4 and P5, the outline for the UUT, the test points and the stabilising screw holes (not all of these will have pillars down to them, but it is good to keep tracks away from those points if pillars are required later for stability on the stopper board). The outline on the board is now changed from the “board definition layer” to the “top silkscreen” layer so it serves only as a visual indication on the base board as to where the UUT is positioned. From this point the schematic is created (with P1,P2,P4 and P5 present) routing all of the lines that need to go to test points on the stopper board using the mapping for P1 and P2. The mappings for P1 and P2 are as follows:

Pin	P1 (BoardA)	P2(BoardB)	Pin	P1(BoardA)	P2(BoardB)
1	Select1	Select1	**21**	Analog5	Analog5
2	Select0	Select0	**22**	GNDA	GNDB
3	Select3	Select3	**23**	Analog4	Analog4
4	Select2	Select2	**24**	GNDA	GNDB
5	Input0	Input0	**25**	Analog1	Analog1
6	Select4	Select4	**26**	GNDA	GNDB
7	Input1	Input1	**27**	Output0	Output0
8	Input2	Input2	**28**	GNDA	GNDB
9	Input3	Input3	**29**	Output1	Output1
10	Input4	Input4	**30**	GNDA	GNDB
11	Input5	Input5	**31**	Output2	Output2
12	Input5	Input5	**32**	GNDA	GNDB
13	Input6	Input6	**33**	Output3	Output3
14	Input7	Input7	**34**	3.3VA	3.3VB
15	Analog9	Analog9	**35**	Output4	Output4
16	Input8	Input8	**36**	3.3VA	3.3VB
17	Analog15	Analog15	**37**	Output5	Output5
18	Analog14	Analog14	**38**	12VA	12VB
19	Analog7	Analog7	**39**	Output6	Output6
20	Analog6	Analog6	**40**	12VA	12VB

*Table 6: P1 and P2 mappings on temp-*boardname*.brd*

The Analog lines are marked for the channel number on the STM32F103 processor's ADC peripheral. Once all of the signal conditioning has been added to the schematic, the netlist can be imported into PCBnew, and the board can be laid out taking into account the positions of the test pins (to not place components over them). Once the board has been laid out, the test points can be removed. The board designator on the PCB silkscreen should be changed from “Temp” to an appropriate board designator like:

BOARD NAME BASE

COMPANY NAME REV *REVISION* / *MONTH* *YEAR*

- **Stopper Board**

  The Stopper board is created by removing from temp-*boardname*.brd, all of the components that were imported (assuming none of the components on the bottom of the board are higher than 14mm), and leaving only the UUT board outline on the “top silkscreen” layer, the test points, the mechanical stabilising screw holes and the connectors P4 and P5. Using the mappings on P4 and P5 that were created during the Base board creation, create the schematic for the stopper board taking into account P4 and P5 as well as the test points. Once the schematic is created, feed the netlist into PCBnew and lay out all of the required tracks. The board designator should be updated as above using the following format:

  BOARD NAME STOPPER

  COMPANY NAME REV *REVISION* / *MONTH* *YEAR* .

- **Guide Board**

  The Guide board is created from temp-*boardname*.brd by removing all of the added connectors, leaving the outline of the UUT on the “top silkscreen” layer, and then using the “bottom silkscreen” layer of the UUT as well as the test point positions and the pin positions of all through hole components to create a board that the UUT can lie flat on during the test, allowing all of the test pins, components and component leads to pass through it without obstruction. In positions where there is a large density of holes that need to be made, a cut-out can instead be made that will allow groups of pins/components/leads to pass through.

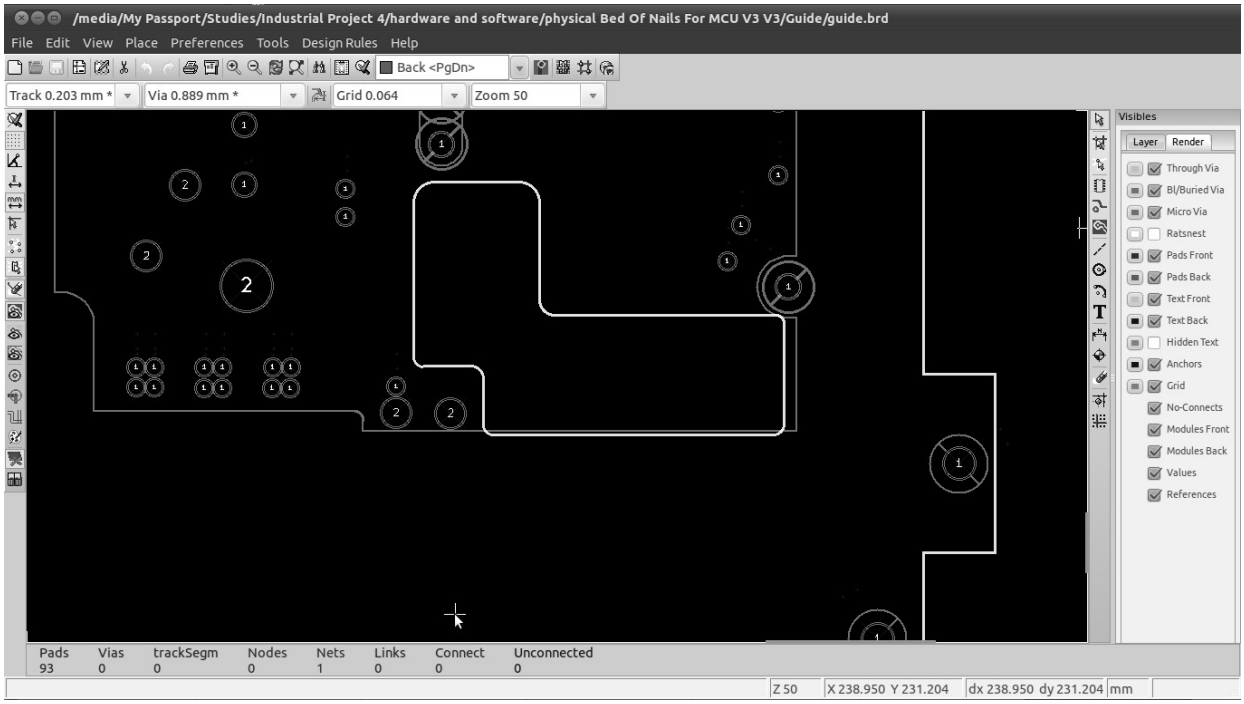


Illustration 29: Guide board showing a cut-out made for a dense number of test points

- **Outline Board**

  The Outline board is created from temp-*boardname*.brd by removing everything but the board definition of the UUT. An outline needs to be drawn in the same around this about 0.5mm away from the original UUT outline. This allows the UUT to rest on the Guide board, but not move forward and backward during the test.

3.4.3 Design Review – Changes Made To Iteration 3 To Create Iteration 4

The Third iteration moved much further towards the final design of the generic test jig system, and allowed the first successful bench tests to be done. The first meeting with the Mechanical department was held, and the outcome was to use the templates of the boards as they stand at the moment and do a first iteration of the physical test jig hardware, including the covering plate for the UUT, as well as the lifting plate arrangement. For more information on the mechanical hardware, see section 3.7. The following 2 iterations (4 and 5) were used to optimise the design, and to fix any latent issues in the generic test jig board and test jig stack. The issues to address in the following iteration are as follows:

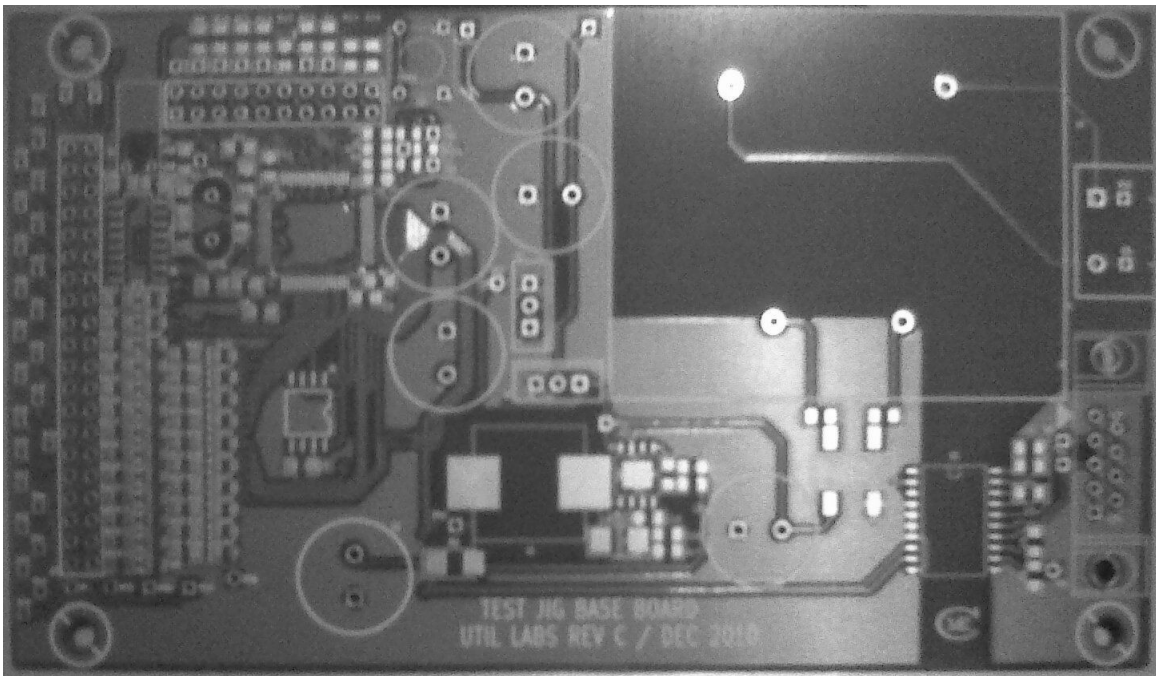


Illustration 30: Generic Test Jig Rev C before population

- **Future expansion limitations due to select lines**

  Problem: In this Iteration the number of boards that the generic test jig can handle is limited to 32 since both of the select line sets reflect the same value allowing you to identify both of the boards as being connected to a jig of ID “00001”. The problem with this arrangement, is that it doesn't allow you to identify which of the two individual generic test jig board is which (as they play different roles) and the 5 bit select lines will only allow a maximum of 32 different test jigs.

  Solution: By combining the two values, and making the ID a 10 bit value with the least significant 5 bits connected to one generic test jig board and most significant 5 bits connected to the other one, there is the opportunity to make almost 1000 jigs (using the full 1024 values wouldn't work, as there are a number of times when jigs would have 2x 5 bit ids that are the same in the 10 bit value, which would lose the advantage of being able to tell the jigs apart). In order to tell apart which generic jig board is which, the “left” jig connecting to the test jig stack will always be connected to the least significant 5 bits.

- **Impedance Test circuit doesn't function correctly**

  Problem: The impedance test circuit doesn't function correctly, and always reads the same value.

  Solution: Re-bias the circuit

- **Confusing naming convention**

  Problem: in the process of mapping each test pin to an input/output on the generic test jig boards, the naming is confusing to follow.

  Solution: fix the logical mapping and wiring of the board (ensure test pins got to the correct positions), by streamlining the naming convention of the inputs/ outputs and specifically the analog measurement pins.

- **Simplify wiring**

  Problem: Wiring the test jig stack is overly complex as no provision is made for a standard way to connect the peripheral hardware in the test jig box to the generic test jig controller.

  Solution: Add connectors for all of the test jig hardware (Interlock, Counter, Indicator Leds RUNNING, PASS, FAIL) to the front of the bed of nails stack allowing all the connectors to be accessible from 1 side. The only connector to keep separate would be the mains input to the impedance measurement circuit, as this will not be fitted for all test jigs, and also simplifies the creapage and clearance requirements. (it is moved slightly though to make the bed of nails compatible with the generic mechanical jig V1.)

- **Zener Diodes leakage affect measurements**

  Problem: The zener diodes were added between the prototype and first revision of the test jig, and were to protect the CMOS chip inputs and ADC inputs on used on the prototype. The problem with the zener diodes are their high leakage current (depending on the voltage applied, as high as 200uA) found in experimentation showed to create a voltage drop of up to 0.2V in the series resistor, which badly affected the analog measurements.

  Solution: Remove the zener diodes. The stm32F103 has internal protection diodes, and with the addition of the series resistor, these diodes offer sufficient protection to over voltage situations. The zeners need not be removed from the PCB, just not populated on the BOM.

- **Excessive heat on high load**

  Problem: When running for a while, 12V, 5V regulators as well as the iCoupler tend to run really hot due to the amount of current consumed by the iCoupled (145mA). This problem is made significantly worse when large numbers of relays are switched.

Solution: Add additional heat sinking to the iCoupler to handle excess heat, and add the capabilities for an external power supply if required (the regulator alone is sufficient for the running of the iCoupler and a few relays, but when switching a large number of times this will become a problem).

3.5 THE FOURTH ITERATION

The 4th iteration was used as a mostly optimization exercise, in that the 3rd iteration was already working but some small problems needed to be fixed with reference to heat and physical wiring. In order to fix the heat generated on the PCB and the strain placed on the transformer, the generic test jig board was changed to allow for either using the on board power supply or an external power supply and by adding significantly more heat syncing on the board for the iCoupler chip. To ensure that only the external power supply or the on-board power supply are used, the components are placed in such a way as to make it impossible for both options to be fitted (this part of the design is covered in detail in the discussion of the Fifth iteration in Section 3.6). In order to simplify the wiring of the jig, connectors to the mechanical system were chosen such that it is difficult to wire the jig incorrectly. There is some responsibility added to the base board to carry the connector to wire the mechanical interlock and counter, and for the stopper board to carry the connector for the wiring of the indicator LEDs. The connectors used to interface between the test jig stack and the mechanical jig are as follows:

- 2-pin: a two pin socket 2EDGRC-7.62-02P is used to allow for the adding of mains to the test jig stack[26]. The mating plug that goes into the socket is 2EDGK-7.62-02P[25].
- 3-pin: the three pin socket 2EDGRC-7.62-03P is used to connect the mechanical counter and mechanical interlock. The mating plug for the socket is 2EDGK-7.62-03P.
- 4-pin: the four pin socket 2EDGRC-7.62-04P is used to connect the indicator lights that show the current testing state to the operator to the test jig stack. The mating plug for the socket is 2EDGK-7.62-04P.
- 40 pin headers: there are 2 x 40 pin ribbon connectors on the test jig base board that go to the generic test jig boards. The length of these ribbon (5cm) cables were chosen such that it is not possible to connect the generic test jig boards incorrectly to the test jig stack.
- 20 pin headers: the 20 pin header is for the JTAG

Using these standard connectors, the base and stopper templates were created to allow for rapid test jig creation, and the process for creating a test jig rapidly was tested out by

creating a test jig for another product. The rapid test jig creation process was a resounding success.

3.5.1 Iteration 4 Characteristics

Board	Rev
Generic Test Jig	D
MCU bed of nails base	C
MCU bed of nails stopper	C
MCU bed of nails guide	C
MCU bed of nails outline	C
Display bed of nails base	A
Display bed of nails stopper	A
Display bed of nails guide	A
Display bed of nails outline	A
Base Board Template	A
Stopper board Template	A

Characterised by

- RS232 communication between PC and test jig
- operator interfaces to the jig using the PC, and with 3 simple test status indicator LEDs.
- PC to give feedback to the operator
- test results are stored on the PC
- Debug communication to UUT is handled by PC
- Test application determines the tests to run based on the bed of nails attached to it – automatic configuration for all types of jig in a single jar
- Single points of isolation – RS232 isolated by iCoupler device
- Measurements float on the same ground potential(s) as the UUT
- fixed interface connector to base of bed of nails, but more expandable as more connectors can be added for more generic test jigs beyond the standard 2

- fixed number of inputs/ outputs per generic test board – scalability and configuration achieved by adding more boards if required beyond the standard 2
- All connectors to the bed of nails now accessible from one side to facilitate easier swapping of beds of nails.
- Physical interlock to stop the mechanical test jig cover from being opened while a test is in progress, and also to help ensuring that the correct label is placed on the correct device when testing in parallel with a single printer (scan to ensure the device and label serial numbers are the same)
- single processor on the generic test board and measurements done using the processor's on board peripherals
- protection on ADC and logic inputs achieved with series resistors and zener diodes.
- LDR light measurement to measure led light.
- impedance test functionality.
- Connectors for external power supplies.
- First rapid test jig created to prove that the project fulfills the initial requirements.
- First test jig stack to be fitted into the mechanical Jigs

The full schematics and gerbers for Iteration 4 are available in Appendix E.

3.5.2 Test Jig Creation Process

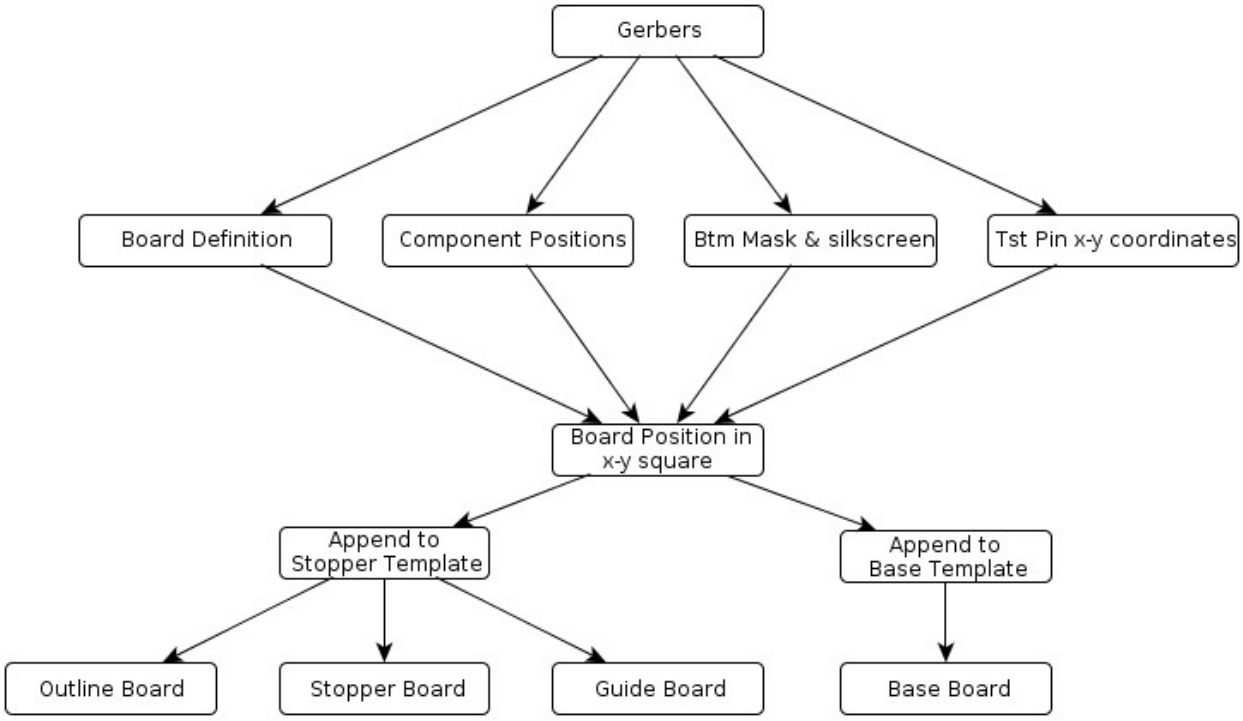


Illustration 31: Iteration 4 Test Jig Stack Creation

The creation of the test jig stack has also been optimised in Iteration 4. Two templates are now used to create the boards in the stack, and the UUT gerbers are “Appended” onto the templates one at a time. Append is a function available in Kicad, which takes one board as a brd file, and places it at the exact same x-y position on another board, and then creates from this an intermediate board called *boardB*-append.brd. To use this, an x-y area is defined, that will allow the UUT to sit in the middle of the test jig stack, and the gerbers are appended to the two templates. To create a .brd file from a set of gerbers, the GerbView utility is used from the Kicad suite. once the brd file is created from the gerbers, the board is moved to the designated x-y coordinates and then appended. The x-y coordinates are defined by a box with the following corners:

Corner	X position	Y position
Top Left	110mm	80mm
Top Right	230mm	80mm
Bottom Left	110mm	230mm
Bottom Right	230mm	230mm

Table 7: Maximum UUT Board Size For Iteration 4

As can be seen, this gives a maximum UUT board dimension of 120x150mm. Once the board has been appended, the boards in the stack are created as follows:

- **Base Board**

  Using the Base-append.brd board created by appending the UUT board, the base board is created by removing everything but the board outline (which is converted to the top silkscreen layer) and the test point positions, so that the schematic can be created (using the base template schematic) that just routes the correct lines from the generic test jig boards to the stopper board where the test points are housed through some intermediate signal conditioning placed on the base board. Once the schematic is created, it can be pulled into the base-append.brd board and laid out taking into account the positions of the test pins. Once the board has been laid out, these test pin positions can be removed. The Base-append.brd board designators need to be updated to show that the board is now a base board for a specific UUT board.

- **Stopper Board**

  The stopper board is created using the Stopper-append.brd created above. The outline of the UUT board is created as a silkscreen and the test pins are populated with test pins in the appropriate positions. Once the test pins have been placed on the board, the schematic can be created by taking the stopper schematic template and adding the test points to it. Any additional signal conditioning not done on the base board (like for example placing transducer hardware to read light as opposed to electrical signals) is now added to the schematic. To communicate with the UUT via a debug port, an ICoupler is added to ensure that the electrical isolation is maintained between the UUT and test PC. The ICoupler is pre placed on the Stopper-append.brd with an appropriate amount of heat syncing and 4 line stubs that are used to connect +5V,gnd,RX an TX to the ICoupler.

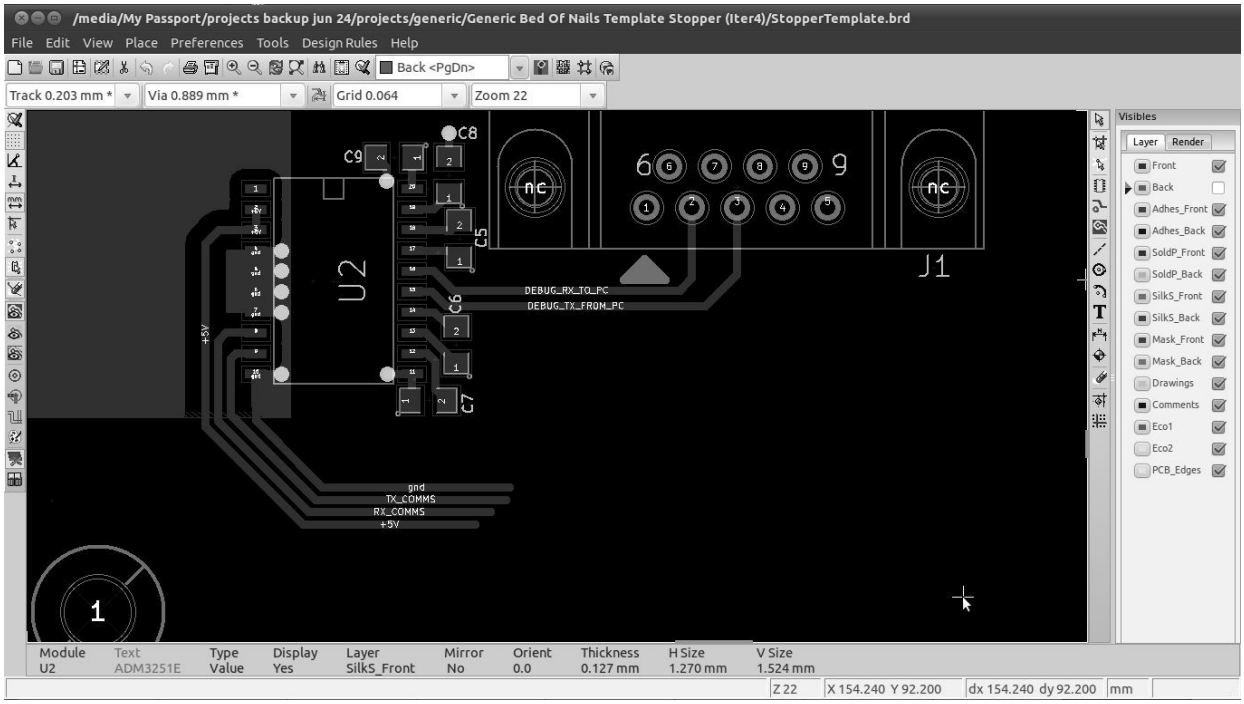


Illustration 32: Stubs On ICoupler On Stopper Template

To power the chip either a pre-laid schematic of a switch mode power supply can be used, that must then be laid out around the test pins, or +5V can be brought up from the base board. If the power supply is brought up from the base board, then the U1 circuit can be removed from the schematic.

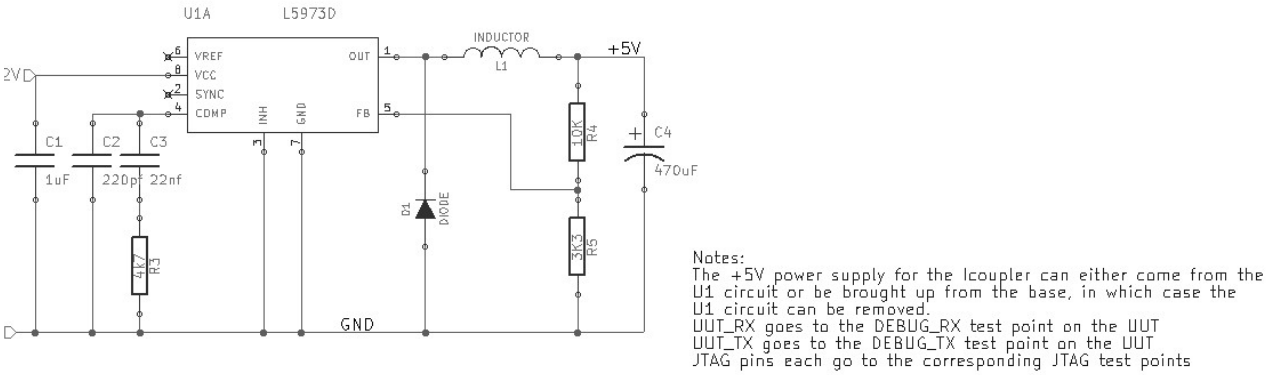


Illustration 33: Optional Switch Mode Power Supply On Stopper Template

Once the schematic is created, the netlist can be pulled into the board and laid out. The Stopper-append.brd board designators need to be updated to show that the board is now a stopper board for a specific UUT board.

- **Guide Board**

  The guide board is created using the Stopper-append.brd created above. The outline of the UUT board is created as a silkscreen, and then the bottom silkscreen of the UUT board is used to create cut outs for any components mounted on the bottom of the board, and the test points and through hole lead positions of other components on the top layer of the UUT are used to create holes for those leads to go through. All of the components on the Stopper append board are then removed. There is no schematic for the guide board and therefore no tracks, and no components are fitted onto it. When manufacturing the Guide board, only the top silkscreen layer need be sent along with the board outline layer so that the manufacturer knows not to add any copper. The Stopper-append.brd designators need to be updated to show this is now a Guide board for a specific UUT board.

- **Outline Board**

  The outline board is also created from the Stopper-append.brd board created above. The only step required for making an outline board is to draw in the Board definition layer an outline around the UUT at a distance of 0.5mm away from it and then removing all of the components on the board. The board designators need to be updated to show this is a outline board for a specific UUT board. The outline board has no schematic or copper layers.

3.5.3 Design Review – Changes Made to Iteration 4 to Create Iteration 5

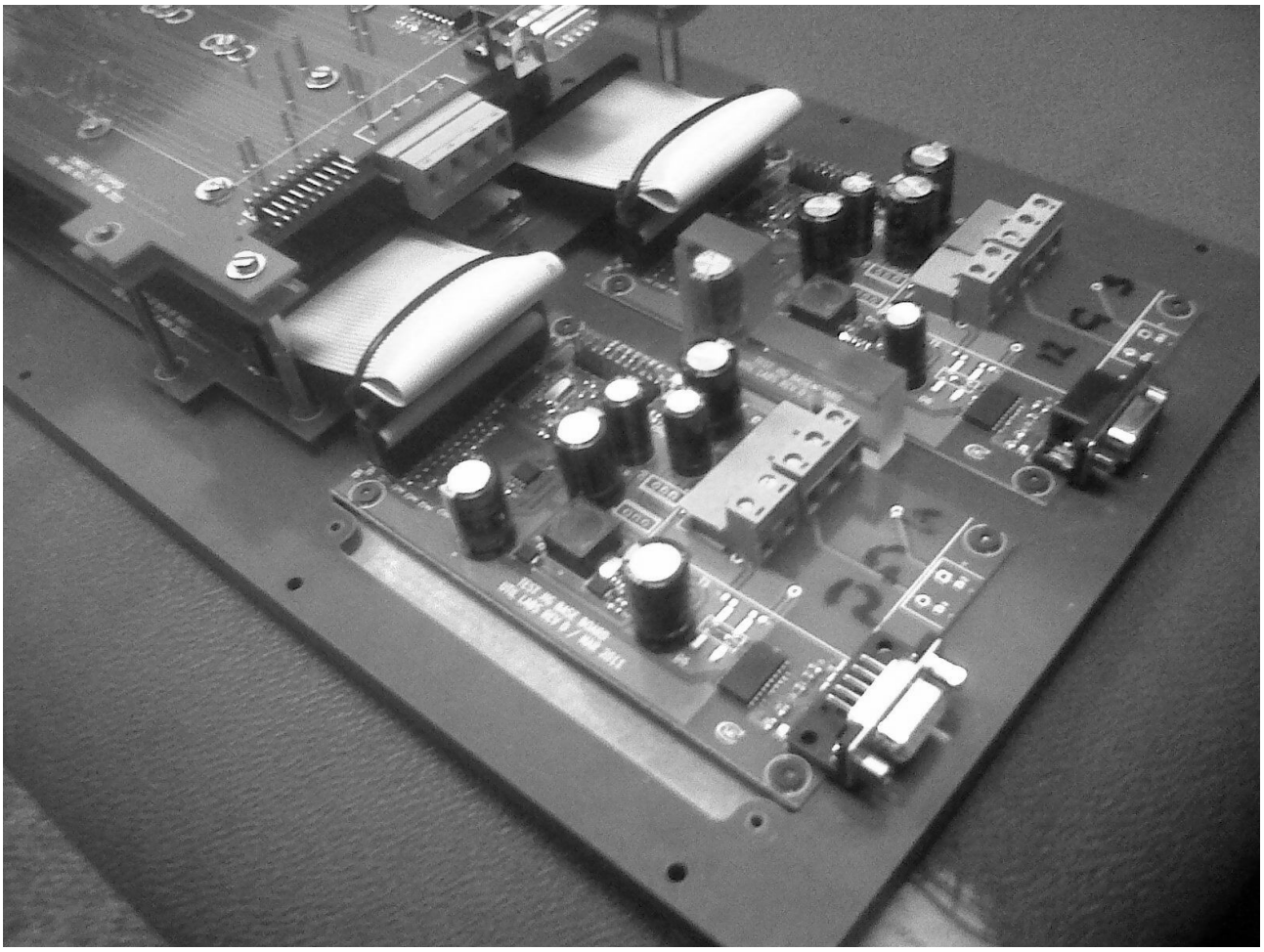


Illustration 34: Fitting Iteration 4 Boards On the Mechanical Base Plate

Version 4 worked well as a test jig stack on the bench, but in attempting to fit it into the 2nd version of the mechanical hardware, there were some problems in seating the stack in the jig. The first revision of the mechanical jig was created around the 3rd and 4th iteration, but a more generic lifting plate test jig frame was found, and with small modifications found to be a suitable solution for the generic test jig. Although the generic mechanical frame was a suitable solution, it was not 100% compatible with the 4th iteration and as such the following mechanical changes need to be made:

- All of the connectors on the stopper board are to be mounted on the bottom layer instead of on top of the board.
- The screws around the edge or the stopper assembly should be placed in set positions so that they can press into the holes provided on the mechanical faceplate.

- Shorter ribbon cables need to be used in the test jig stack to stop mechanical interference.
- The switching probes used to detect board presence are just too expensive, so an alternative solution to presence detection needs to be explored.

3.6 THE FIFTH ITERATION

The 5th iteration is the final iteration in terms of the initial project proposal. It marks the successful implementation of a test jig in a mechanical housing and successful testing of a board supplied from a manufacturer by a qualified operator. The templates and generic test jig boards were finalised into a release version and all documentation done for the creation of test jig stacks and the programming of test sequences. The flexibility of the generic test jig system was also shown, in that the board used as a guinea pig throughout the course of this project (MCU V3) changed during this iteration, and the changes were made in the test jig stack to accommodate both the original MCU V3 board as well as the MCU V3 (Rev D) without any problems. The only major design changes was to stop using switching probes to detect board presence due to the high cost of these pins and their respective holders and to instead opt for a simple micro switch, and the changing of the stopper assembly to now have a bottom board that accommodates the connectors mounted on the bottom of the stopper PCB (Backplane). The templating process was greatly optimised in this iteration, creating a positioner board that allows the UUT under test to be positioned for each board in the stack, and this even allows multiple people to work on a stack at once. Full isolation is also now a reality as all of the communication lines are now as a standard (due to the template) isolated using ICouplers, an the JTAG can be isolated using a JTAG isolation board from Segger (available from Digikey as part number 899-1003-ND)[13]

3.6.1 Iteration 5 Characteristics

Board	Rev
Generic Test Jig	D
MCU Rev D bed of nails base	C
MCU Rev D bed of nails stopper	B
MCU Rev D bed of nails guide	A
MCU bed of nails outline	C
MCU Rev D backplane	A
Base Board Template	B
Stopper board Template	B
Backplane Template	B
Guide Template	B
Outline Template	B
Positioner	A

Table 8: Hardware designs associated with iteration 5

- RS232 communication between PC and test jig,UUT
- Full Isolation between test jig PC and UUT on all communication lines through ICoupler devices or opto couplers.
- operator interfaces to the jig using the PC, and with 3 simple test status indicator LEDs.
- PC to give feedback to the operator
- test results are stored on the PC
- Debug communication to UUT is handled by PC
- Test application determines the tests to run based on the bed of nails attached to it – automatic configuration for all types of jig in a single jar
- Measurements float on the same ground potential(s) as the UUT
- fixed interface connector to base of bed of nails, but more expandable as more connectors can be added for more generic test jigs beyond the standard 2
- fixed number of inputs/ outputs per generic test board – scalability and configuration achieved by adding more boards if required beyond the standard 2

- All connectors to the bed of nails accessible from one side to facilitate easier swapping of beds of nails.
- Physical interlock to stop the mechanical test jig cover from being opened while a test is in progress, and also to help ensuring that the correct label is placed on the correct device when testing in parallel with a single printer (scan to ensure the device and label serial numbers are the same)
- Single processor on the generic test board and measurements done using the processor's on board peripherals
- LDR light measurement to measure led light.
- impedance test functionality.
- Connectors for external power supplies.

All Schematics and gerbers for the boards in Iteration 5 are available in Appendix F.1.

3.6.2 Detailed Design Description

The designs for the generic test jig board are described here as Iteration 5 represents the Final version of the board, and as such the most complete (previous iterations may have had some parts of the circuits only partially implemented, or implemented only to enable the debugging and testing process for the board). Below are the descriptions of the individual parts of the Generic Test Jig Board:

3.6.2.1 Power Supply

The generic test board can either be powered via an on board power supply with a 220V mains input, or using an external 12V/5V power supply unit. The Board is laid out in such a way, that if either one of these options is chosen, the other one physically cannot be implemented. For example, the tracks to the 0Ω resistor R44 which allows the switch mode power supply to run off of the external 5V power supply, is placed in the same physical location as the bridge rectifier D6 that would normally supply the input to the switch mode rectifier. The same applies with resistors R45, R46 and connector P2 which allow the 12V and 5V external supplies to be brought onto the board are placed below the transformer T1 allowing only one option to be populated.

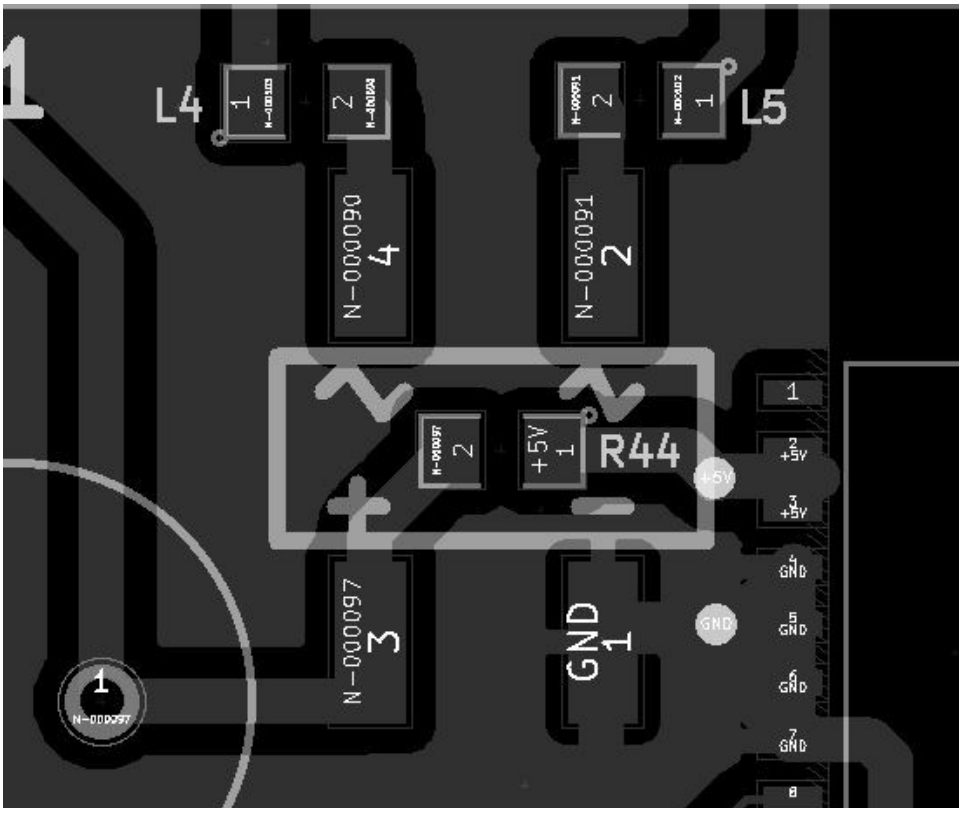


Illustration 35: Mutually Exclusive Placement of R44 and D6

The following table shows which components should be fitted depending on which power supply option is required.

Power Supply	Fit	Do Not Fit
On Board	T1 ,P1,L4,L5,D6 U4,U7,C20,C22,C24	P2 ,R44,R45,R46
External	P2 ,R44,R45,R46	T1 ,P1,L4,L5,D6 U4,U7,C20,C22,C24

Table 9: Components To Place For External vs On Board Power Supply

There are 3 voltages used in the generic test jig system:

- 3.3V used to power the controller and any logic on the test jig stack that is required. 3.3V can also be used to supply power to the UUT, provided it does not overload the power supply. The 3.3V supply is either fed from an external 5V supply, or from the on board 12V transformer.
- 5V used to power the ICoupler isolator as well as any logic specifically requiring 5V. If the on board supply is used, this should not be used to power the UUT. The 5V supply is either supplied externally, or fed from the 12V supply.

- 12V used to switch relays or to power the UUT. The 12V supply is either supplied externally or supplied from the on board transformer.

Transformer

The Transformer on the board is a Jessiva A103 230V transformer with a 12Vac output and a 5VA rating. The current that the transformer can supply is therefore:

$$\begin{aligned} Imax &= \frac{5\text{VA}}{12\text{V}} \\ &= 416\text{mA} \\ &\approx 410\text{mA} \end{aligned}$$

and the transformer internally fuses after this value (500mA fuse point). The output current is shared by the three power supplies. The output voltage is rectified by the DF1510S bridge rectifier (alternative bridge rectifier B380S can also be used) and the the full wave rectified voltage is smoothed by two 470uF electrolytic reservoir capacitors C20 and C21. The rectified output voltage will peak at:

$$\begin{aligned} Vdc &= Vac \times 1.414 \\ &= 12 \times 1.414 \\ &= 16.97\text{V} \\ &\approx 17\text{V} \end{aligned}$$

+3.3V Supply

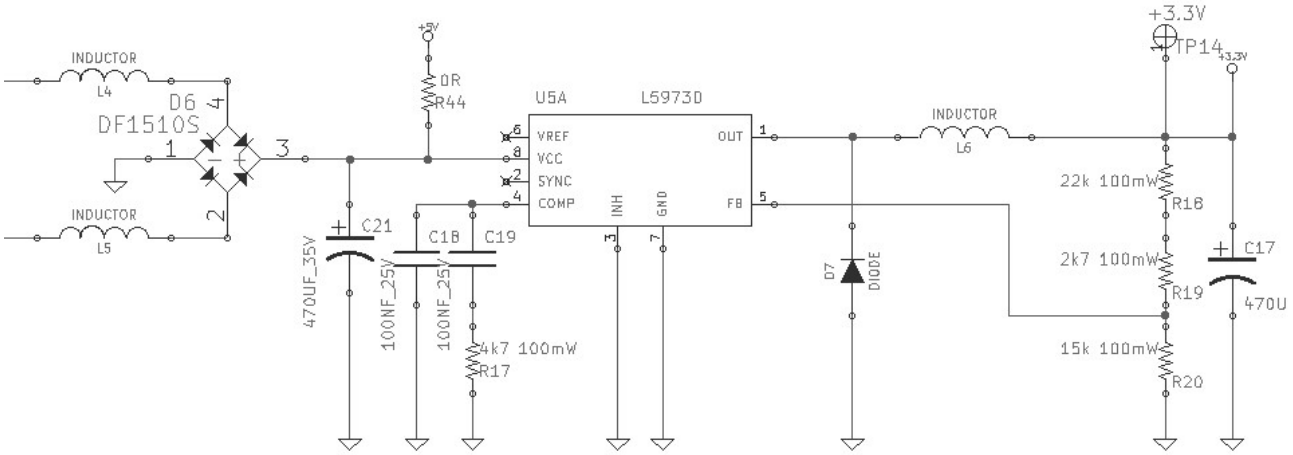


Illustration 36: 3.3V Switch Mode Power Supply

The 3.3V supply is built around a L5973D step down switching regulator. The input voltage range for the chip is from 4V to 36V [28, table 2 page 4]. If the external supply option is used, then R44 is populated and 5V is supplied to the input VCC pin of the regulator. If the on board power option is used, then the voltage is supplied from the on board 12V transformer at ≈ 17Vdc, which is well within the 36V limit. The output of the switch mode power supply is calculated from the feedback resistors R18,R19 and R20. The expected output is:

$$Vout = Vref\left(1 + \frac{(R18 + R19)}{R20}\right)$$
$$= 1.235\left(1 + \frac{(22\text{k} + 2\text{k}7)}{15\text{k}}\right)$$
$$= 3.27\text{V}$$

Where:

$Vout$ = *the output voltage*

$Vref = 1.235\text{V}$ *reference voltage*

Which with values of R18 = 22k, R19 = 2k7 and R20 = 15k gives an output voltage of 3.27V. The 470uF electrolytic capacitor on the output acts as a reservoir.

The inductors L4 and L5 in the circuit to the D6 bridge rectifier are 0805 ferrite filter inductors to stop noise from the switch mode power supply leaking back onto the mains input due to the switching frequency of 250kHz. The Flyback coil L6 was chosen as a shielded 33uH coil as this value was already a stocked value in the Util Labs store.

The only components on the generic test board that draw current continuously from the

3.3V rail are the STM32F103 microprocessor and the AT45DB32 flash chip, which draw in total < 50mA.

+12V and +5V Supplies

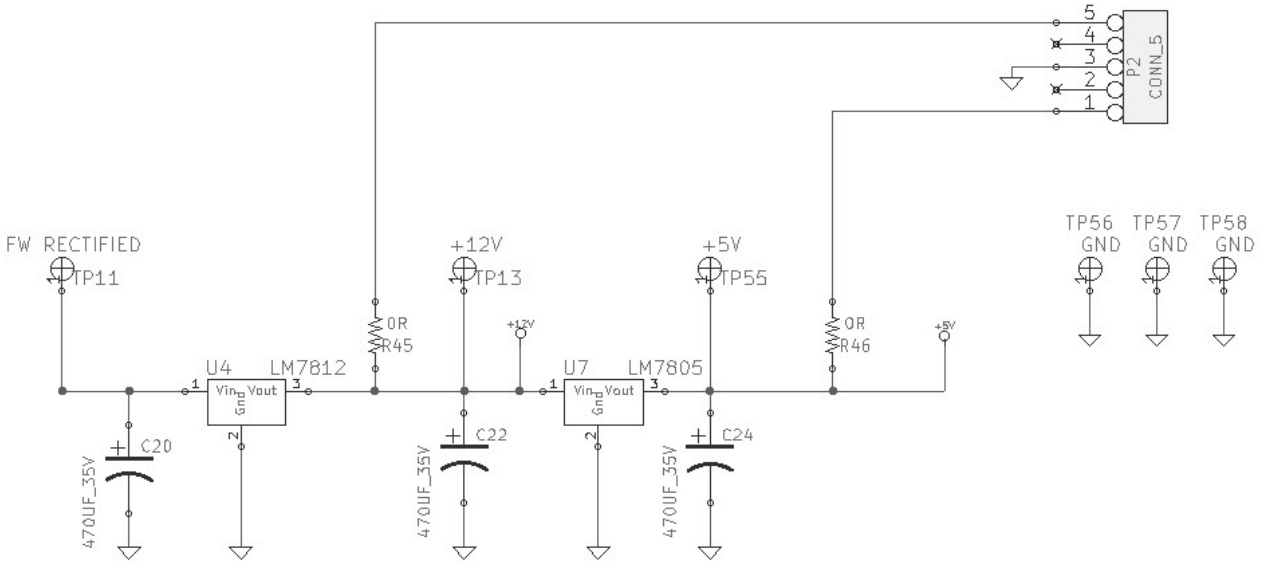


Illustration 37: The 12V And 5V On Board Power Supplies

The on board 12V supply is fed by the on board 12V transformer that supplies it with a rectified voltage of ≈ 17V, and the 5V regulator is fed from the 12V supply. The regulators are standard LM7812 and LM7805 linear regulators in TO-220 packages. The 5V supply is supplying the ADM3251E ICoupler which draws 110mA nominally from the power supply when not connected to anything, and up to 145mA when connected to another RS232 device (dependent on the load resistance that chip places on the transmitter of the ICoupler) [24, page 3 table1]. Since the 5V chip is driven from the 12V rail, the drop across it will be 7V. For the TO-220 package, θJA (Junction to Ambient thermal resistance) is 54 °C/W [29, page 6 note 2]. Assuming an ambient temperature inside of the test jig of 35°C, then to keep the regulator junction below the maximum 150°C (de-rated to 120°C), the maximum power that can be dissipated for both the 12V and 5V supplies is:

$$Pmax = \frac{(Tmax - Ta)}{\theta JA}$$
$$= \frac{(120 - 35)}{54}$$
$$= 1.57\text{W}$$
$$\approx 1.5\text{W}$$

Where:

$Tmax$ = The maximum case temperature in °C
Ta = The ambient temperature in °C
θJA = The thermal resistance of the component junction to ambient

Since the base current (for the ICoupler) drawn from the 5V regulator can be assumed at 145mA, this current will contribute:

$$P5V = Vdrop5V \times I$$
$$= 7\text{V} \times 145\text{mA}$$
$$= 1.015\text{W}$$

Where:

$P5V$ = The dissapation accross the 5V regulator
$Vdrop5V$ = The voltage drop accross the regulator
I = The current drawn from the supply for the Icoupler

and

$$P12V = Vdrop12V \times I$$
$$= 5\text{V} \times 145\text{mA}$$
$$= 0.725\text{W}$$

Where:

$P12V$ = The dissapation accross the 12V regulator
$Vdrop12V$ = The voltage drop accross the 12V regulator
I = The current drawn from the supply for the Icoupler

which means that only 0.5W can be drawn from the 5V supply and only 0.775W from the 12V supply by the test jig stack. If any more power is required than can be supplied by the on board power supplies, an external 5V/12V supply can be used as described above.

3.6.2.2 ICoupler

The ADM3252E ICoupler is the basic isolating component used throughout thus project. It is a UART to RS232 converter chip that allows 2.5kV isolation between UART and RS232 sides of the chip. The main advantage of using the Coupler rather than an opto coupler arrangement is the fact that the ICoupler generates its own power supply for the isolated section by using on chip CMOS transformers

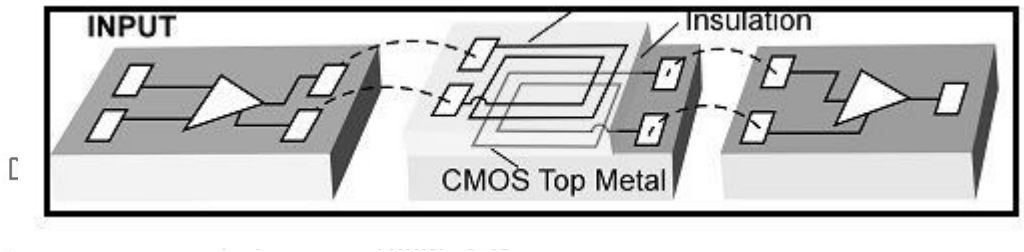


Illustration 39: ICoupler CMOS Transformer technology

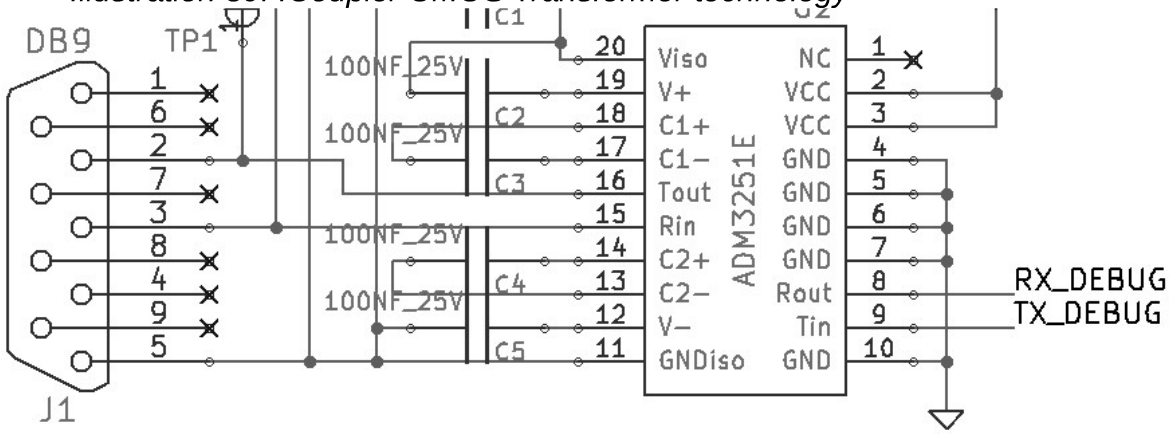


Illustration 38: ICoupler Schematic
image retrieved from [15]

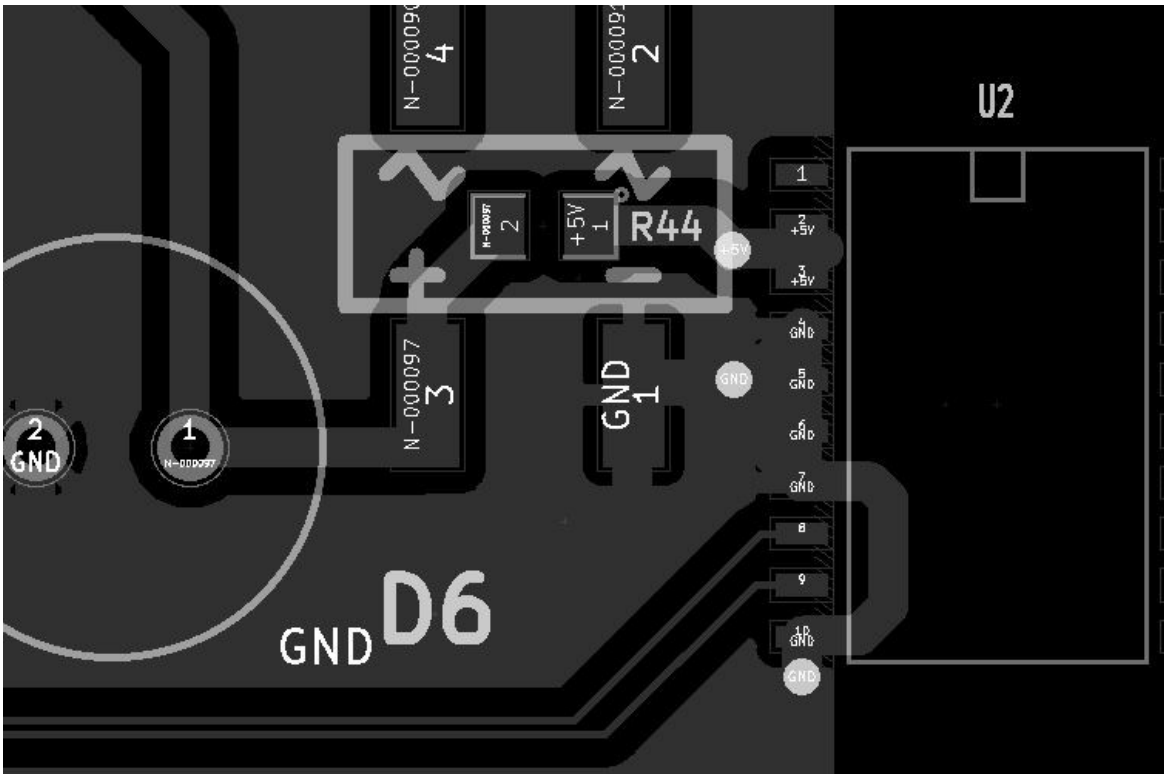


Illustration 40: Vias Added And Copper Over Pads To Increase Heat Sinc

The data is transferred from the input side of the chip to the output side by generating 1ns pulses on state changes in the input. The power is also transferred from the one side to the other by means of a 300Mhz internal oscillator that feeds a CMOS transformer and is rectified and regulated on the Isolated side of the chip[16]. It is due to this high frequency oscillator that the chip has a fairly high current consumption and heat dissipation. To do away with excess heat, the chip has a large amount of heat sink in the form of copper around it on both the top and bottom copper layers of the PCB. Other than for the addition of one 100nF capacitor, the ADM3251E chip schematic is similar to standard RS232 converters like the MAX232. The UART RX and TX signals are fed into the chip on pin 8 and 9, and the isolated RS232 signals are available at pin 15 and 16. C1 is a smoothing capacitor for the isolated voltage that is generated by the ICoupler, and capacitors C2-C5 form part of the charge pump of the chip to create the RS232 voltages.

3.6.2.3 Output Driver

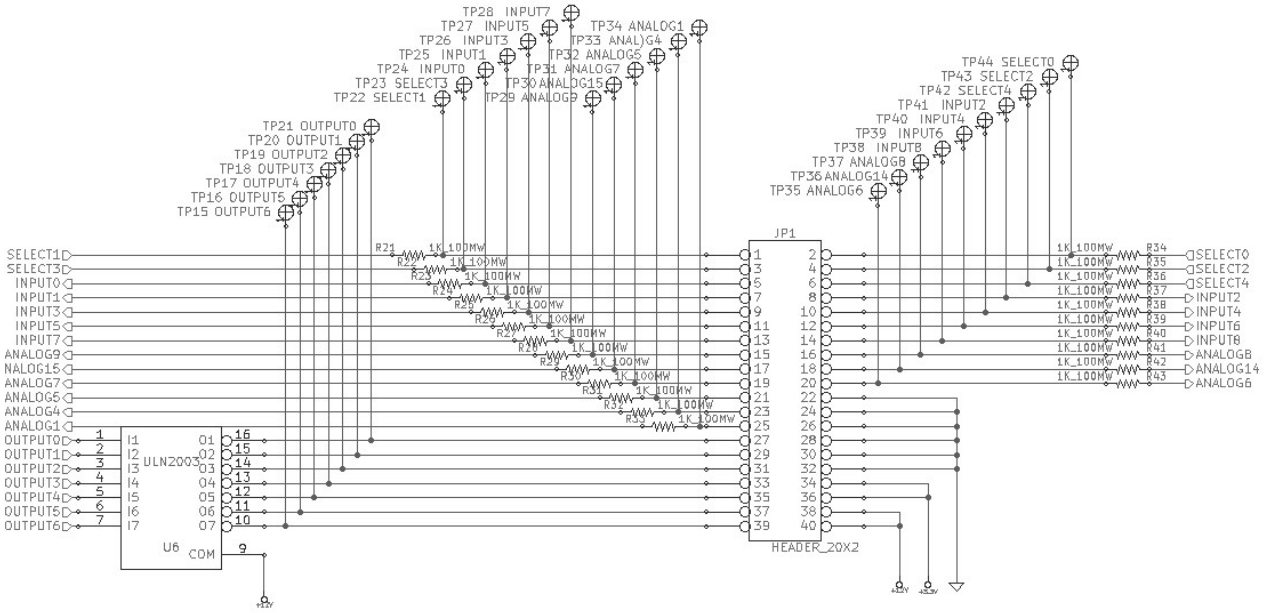


Illustration 41: The 40 pin Interface Connector And Output Driver

The outputs on the generic e jig boards are all driven through a ULN2003A Darlington driver chip. The Darlington driver already has a bias resistor to the input of the transistor pair for each channel, and a diode to the common rail (12V) allowing it to drive inductive loads up to 500mA [30, page 2 table MAXIMUM RATINGS].

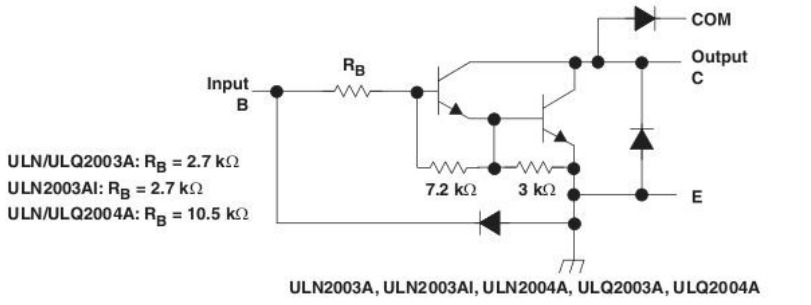


Illustration 42: Schematic For One Channel Of The ULN2003A driver

image retrieved from [30].

The Schematic above also describes the final 40 pin ribbon interface connector pin-outs.

3.6.2.4 Controller

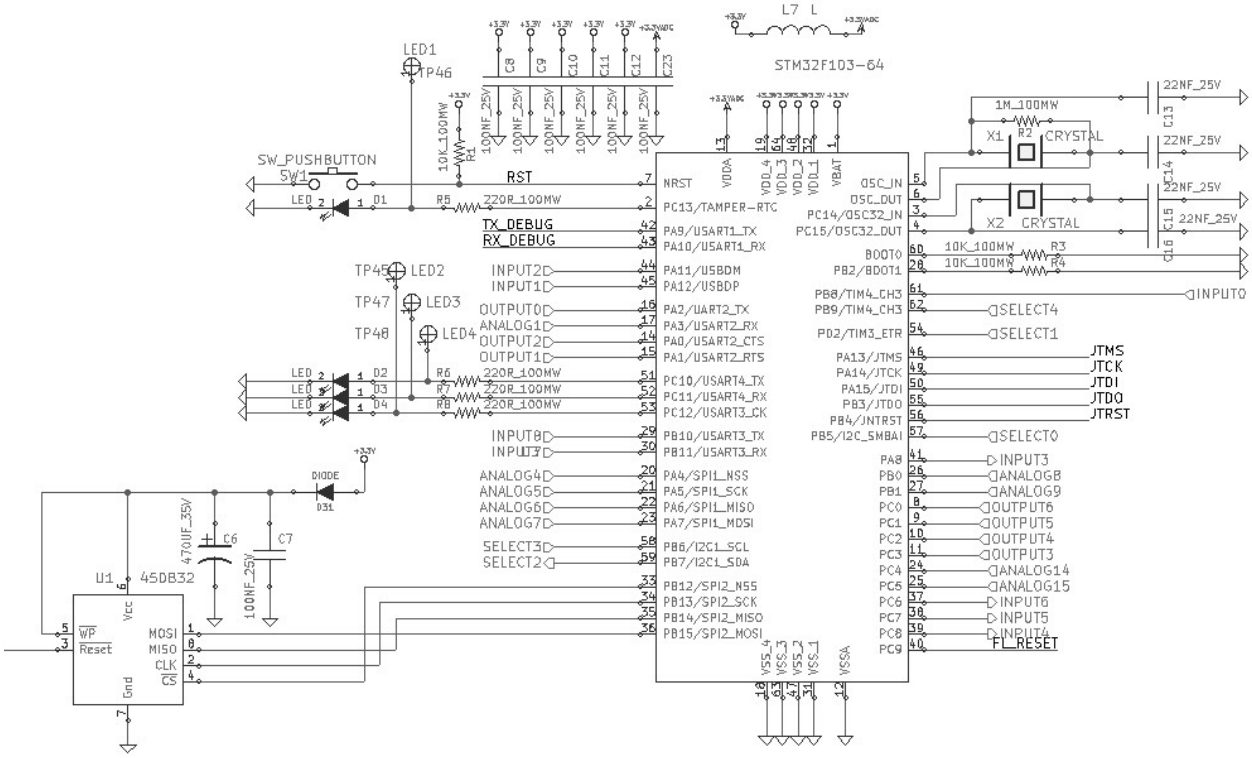


Illustration 43: STM32F103 Microprocessor Generic Test Board Controller

The main controller on the generic test jig board is an STM32F103 microprocessor. All of the measurements on the UUT are eventually done with this microprocessor's internal peripherals (through appropriate signal conditioning on the test jig stack). The AT45DB32 flash memory is a requirement of the EOS operating system. All measurements done by the controller are done on the prompting of the test jig PC through a message, and the results of the measurements are sent back to the test jig PC in a message when the results become available. The main actions performed by the controller are as follows:

Action	Request Message	Response Message
Measure voltage points	0x730	0x731
Measure logic level	0x732	0x733
Start a pulse test	0x734	N/a
Stop a pulse test	0x735	N/a
Count a number of pulses*	0x736	0x737
Measure pulse frequency*	0x736	0x737
Switch an output	0x011	N/a
***Before pulses can be measured, a pulse test must be started**		

Table 10: Actions Performed By The Generic Test Jig Controller

The firmware that controls this process is described in section 3.6.3.

The controller is programmed by a standard JTAG interface.

Analog measurements are done using the on board ADC peripheral, and as such the conversion is possible across the full range from 0V to the Vdda voltage [19, page 103 table 59], tied on the board to the 3.3V rail through the LC filter formed by the ferrite bead L7 and C23. The ADC is used at the slowest possible speed (239.5 cycles per conversion) to make sure that high divider resistors in the signal path have minimal impact on the measurement. The measured value is calculated as follows:

$$Vuut = \frac{Vmeasure \times (R1 + R2||Radc)}{R2||Radc}$$

Where:

$Vuut$ = The voltage to be measured
$Vmeasure$ = The voltage measured at the ADC
$R1$ = The top resistor in the divider
$R2$ = The voltage to be measured
$Radc$ = The input impedance of the ADC *voltage*

The input impedance of the ADC input pins is the ADC channel resistance (Rain) in series with the sampling switch resistance(Radc), in series with the 1k protection resistors (R28-R33 and R41-R43). Rain is given as 50kΩ and Radc as 1kΩ in the datasheet [19, page 103 table 59], which gives an input impedance of 52kΩ.

3.6.2.5 Light Sensor

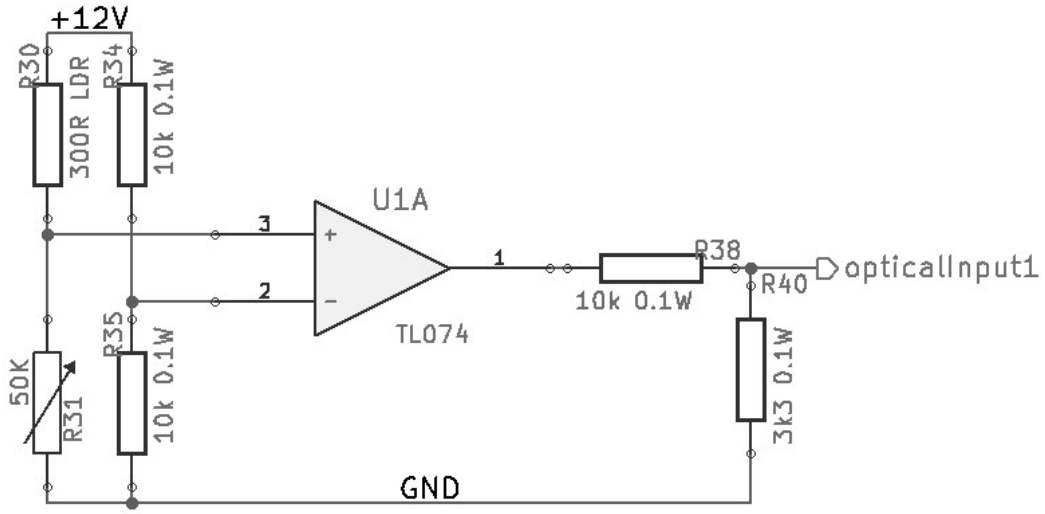


Illustration 44: Light sensor

The light sensor is placed on the stopper board is built around a Jameco 9001 light

Dependant resistor. The LDR has a resistance of 300kΩ in no-light conditions, 20kΩ at 1lux light level, and 50kΩ at 10 lux light level [27, page 2 graph kΩ vs lux]. This LDR is placed in series with a 50kΩ multi turn potentiometer which is then calibrated to the maximum light level that the test jig will see with the UUT placed in the test jig, and the light source on the UUT off. This is then used as the positive input to a TL074 op-amp that is used as a comparator[32]. The negative input is then connected to a resistor divider created by two 10kΩ resistors, which biases the negative input at 6V (half way on the 12V rail). When there is no light source present, the potentiometer will have a lower resistance than the LDR. When the light source goes on, the resistance of the LDR drops (since the LDR is placed right under the light source of the UUT, the 1 lux level will be reached for devices with minimal lumen ratings), and the voltage on the positive input of the comparator will rise above the 6V bias level and change the state of the output. A resistor divider network is added on the output signal to drop the 12V output signal down to an appropriate level for th 3.3V inputs on the generic test jig.

Take note that the LDR has a 25ms decay time and 60ms rise time which means that pulsed signals above 10Hz will not necessarily be measured correctly.

3.6.2.6 Impedance Measurement And Power Test

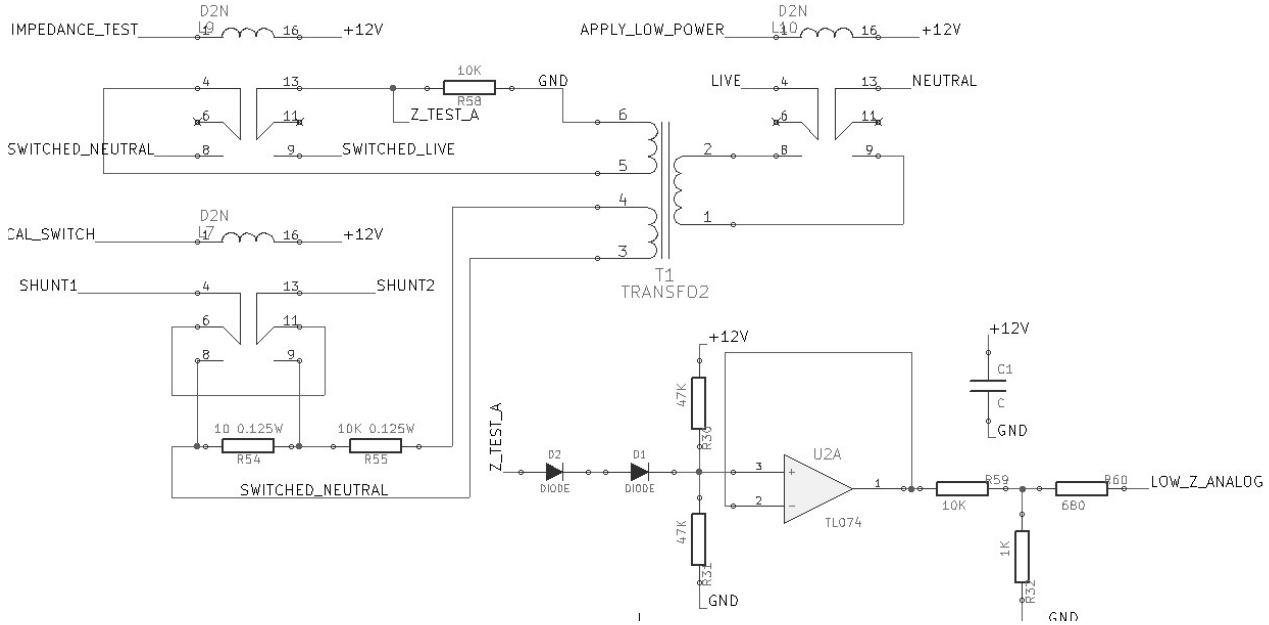


Illustration 45: Impedance Measurement Ciruit

Impedance Measurement

The impedance measurement hardware is placed on the Base board of the test jig stack. The impedance measurement test is run before any of the other tests on the UUT to check if there is either a short circuit or a open circuit across the voltage input pins of the UUT. In the case of the MCU guinea pig board, the power input to the board is 220V mains. The circuit placed into impedance measurement mode by actuating the LOW_POWER and IMPEDANCE test relays (L10 and L9 respectively) which powers the low power transformer T1 and supplies 12Vac to the LIVE and NEUTRAL input pins to the UUT. The 12Vac is in series with a 10kΩ resistor R58. A resistor divider is now formed between the impedance of the UUT and the 10kΩ resistor. The output of this resistor divider is now rectified through diodes D1 and D2 which both rectify the signal and drop it by 1.4V to ensure the TL074 op-amp U2 is not damaged. The op amp is normally biased at 6V through the resistor divider of R30 and R31, but the current flows through the rectifier diodes to ground through R31, this will cause the bias voltage to vary. If there is no UUT board present, or the board measures open circuit, the input voltage on U2 will remain at 6V, but if there is a short circuit, then R58 will see the full voltage of T1 and this will raise R31 closer to the 12V rail. U2 acts as a buffer, with the output going to a resistor divider formed by R59 and R32 which will drop the buffered output of the op-amp to a suitable level for input into the generic test jig. An additional 680Ω series resistor R60 is added to offer additional protection to the input of the generic test jig.

Suitable impedance values for production boards are found experimentally by measuring the impedance of a number of known good boards and setting appropriate boundaries for the measured voltage.

Power Test

Since the MCU guinea pig board has a function to measure series current, a power test circuit was added to force an amount of current through a mock shunt R54 which will mimic the 0.001Ω on the UUT and test that power level is measured to ensure the shunt and power measurement circuit is working correctly. A 10Ω mock shunt is used, as it will simulate a much higher current, since the voltage drop across a 10Ω resistor will be 10000 times higher than the equivalent current through the shunt resistor.

The same transformer is used for the low voltage source as for the impedance test, only this time the CAL_SWITCH and APPLY_LOW_POWER (L7 and L10 respectively) are

actuated. The mock shunt is placed in series with a 10kΩ resistor R55 which causes a current of:

$$Imock = \frac{17\text{V}}{10 + 10\text{k}}$$
$$= 1.698\text{mA}$$
and
$$Vmock = Imock \times 10$$
$$= 16.98\text{mV}$$

Which simulates a current of:

$$Isim = \frac{Vmock}{Rshunt}$$
$$= \frac{16.98\text{V}}{0.001\Omega}$$
$$= 16.98\text{A}$$

This simulated current is measured by looking at the output of the measurement chip on the UUT which has a pulsed output corresponding to the measured amount of current. During the test, the measurement chips are not yet calibrated on the UUT, so there is a wide range of acceptable values for the pulse frequency, but it must send pulses to pass the test.

3.6.3 Firmware

The firmware on the generic test jig allows the actions described in section 3.6.2.4 to be performed by the hardware. To understand the firmware sections, some background on the EOS operating system is required. Information on the operating system is available in Appendix G. The following files are important to the functioning of the generic test jig in iteration 5:

- gw_FastInputs.cpp and gw_FastInputs.h
- gw_Adc.cpp and gw_Adc.h
- gw_TestJigGpio.cpp and gw_TestJigGpio.h
- drv_FastInputs.cpp and drv_FastInputs.h
- drv_Adc.cpp and drv_Adc.h
- tsk_StateTracker.cpp and tsk_StateTracker.h

The individual files are available in Appendix F.2. All of the classes are discussed in section 3.6.3.4.

3.6.3.1 Inputs and Outputs

Relevant files: tsk_StateTracker.cpp, tsk_StateTracker.h

Relevant Messages: “State Changed” (0x10), “Set State” (0x11), “State Request ” (0x732), “State Response”(0x733)

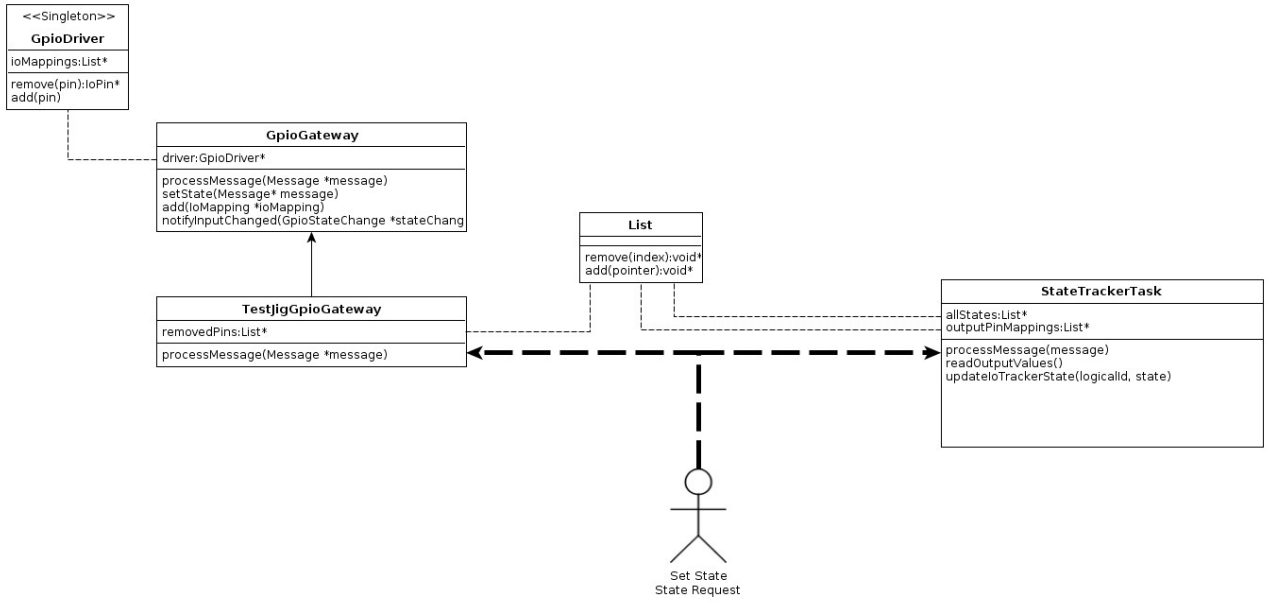


Illustration 46: Input and Output Actions (Only Relevant Methods and Members Shown)

Physical State Change

Whenever a physical pin that is tracked by the GpioDriver changes state, the system is notified of it by the TestJigGpioGateway sending out a “State Changed” (0x10) message. The problem with this, is that the test jig PC may not have been listening when that message was sent as it is not assumed to be connected to the internal message bus of the generic test jig system 100% of the time. So the StateTrackerTask will intercept this state, and track it along with all of the other inputs on the generic test jig.

“Set State” (0x11)

When a set state message is received, the TestJigGpioGateway will instruct the GpioDriver to change the state of the output to the state specified in the “Set State” message

“State Request” (0x732)

When a “state request” message is received for an input on the system, the StateTrackerTask will respond with a “State Response” (0x733) message if the LogicalId specified in the message in its list of known states, or it will ignore the message, assuming it is for another task if it is not in the list.

Using the messaging interface for the switching of IO on the generic test jig allows the test jig to comfortably slot into the existing product line of Util Labs. A lot of code re-use is possible as all system interactions in the ULM system are message based.

3.6.3.2 Fast Inputs

Relevant files: gw_FastInputs.cpp, gw_TestJigGpio.cpp, drv_FastInputs.cpp, gw_FastInputs.h, gw_TestJigGpio.h, drv_FastInputs.h

Relevant Messages: “Start Pulse Test” (0x734), “Stop Pulse Test” (0x735), “Pulse Data Request” (0x736), “Pulse Data Response” (0x737)

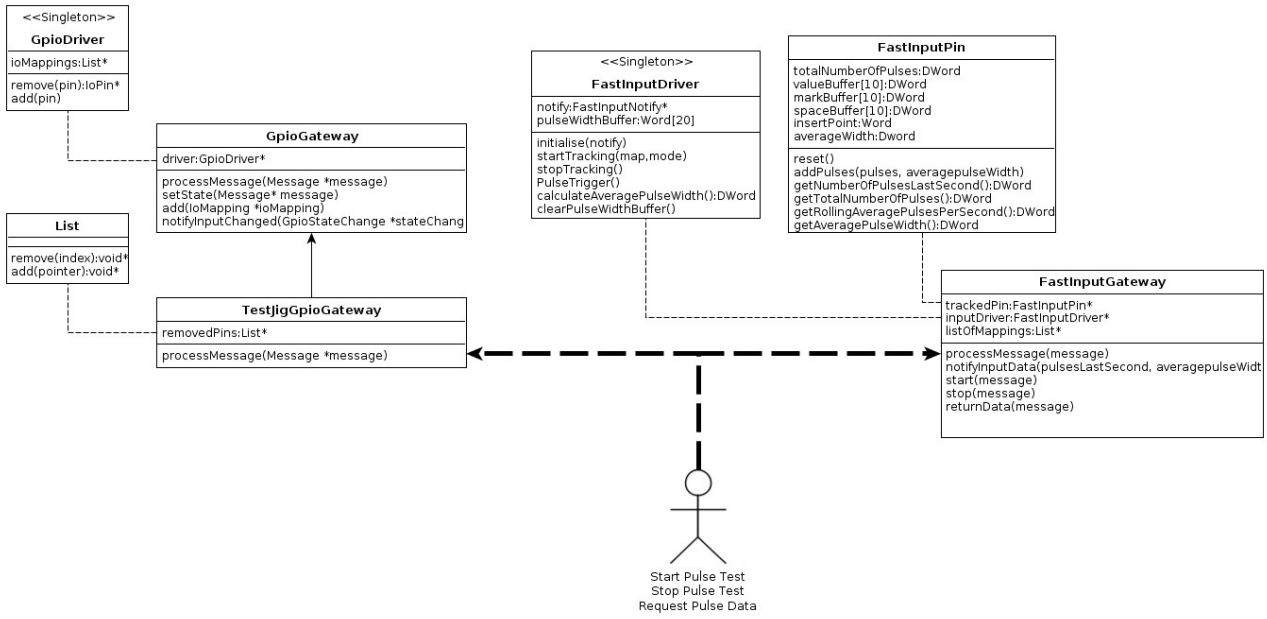


Illustration 47: Pulse Test Actions (Only relevant methods and members)

“Start Pulse Test” (0x734)

When the “Start Pulse Test” message is received by the system, the TestJigGpioGateway disables the appropriate input pin in the and GpioDriver so that it is no longer tracked and debounced. At the same time the FastInputGateway receives the same message (since they are on the same message bus) and instructs the FastInputDriver to set the pin to an

external interrupt, with the appropriate edge detect mode (riding, falling or both), and then the gateway creates a FastInputPin object that will track all of the information for that pin during the test. While the test is running and there is a FastInputPin active, the FastInputDriver will notify the FastInputGateway once a second of the number of pulses received as well as the average pulse length of those pulses. This information is passed from the gateway to the FastInputPin So it can be tracked

“Request Pulse Data” (0x736) and “Pulse Data Response” (0x737)

If there is no active FastInputPin and a “Request Pulse Data” message is received then the gateway will respond with a “Pulse Data Response” that has LOGICAL_ID_INVALAD (0xffff) as its pin Id specifier. However while the pulse test is active, and there is an active FastInputPin requests will be responded to with valid pulse data.

“Stop Pulse Test” (0x735)

Once all of th required data has been collected, a “Stop Pulse Test” message is sent by the test jig PC. Once this message is received, the TestJigGpioGateway will return the pin it removed from it to the GpioDriver which can then once again resume monitoring and debouncing it. At the same time the FastInputGateway will instruct the FastInputDriver to reset the required pin back to a standard input. The FastInputPin is destroyed.

The fast input mode allows all of the inputs on the generic test jig to also act as pulse capture inputs. It also allows for a lot of different tests to be run, as FastInputPin tracks not only the average pulse frequency but also the last second's pulse frequency (allowing changes in frequency to be detected), the total number of pulses since the start of the test and the average pulse width, all of which is contained in the “Pulse Data Response” message. By decoupling the tracking of the data from the hardware pin driver, it becomes very simple to later on add more functionality to the pulse driver by simply adding another message and expanding the FastInputPin tracker.

3.6.3.3 ADC

Relevant files: gw_Adc.cpp, drv_Adc.cpp, gw_Adc.h and drv_Adc.h

Relevant Messages: “Request ADC Measurements” (0x730), “ADC Measurement Response” (0x731)

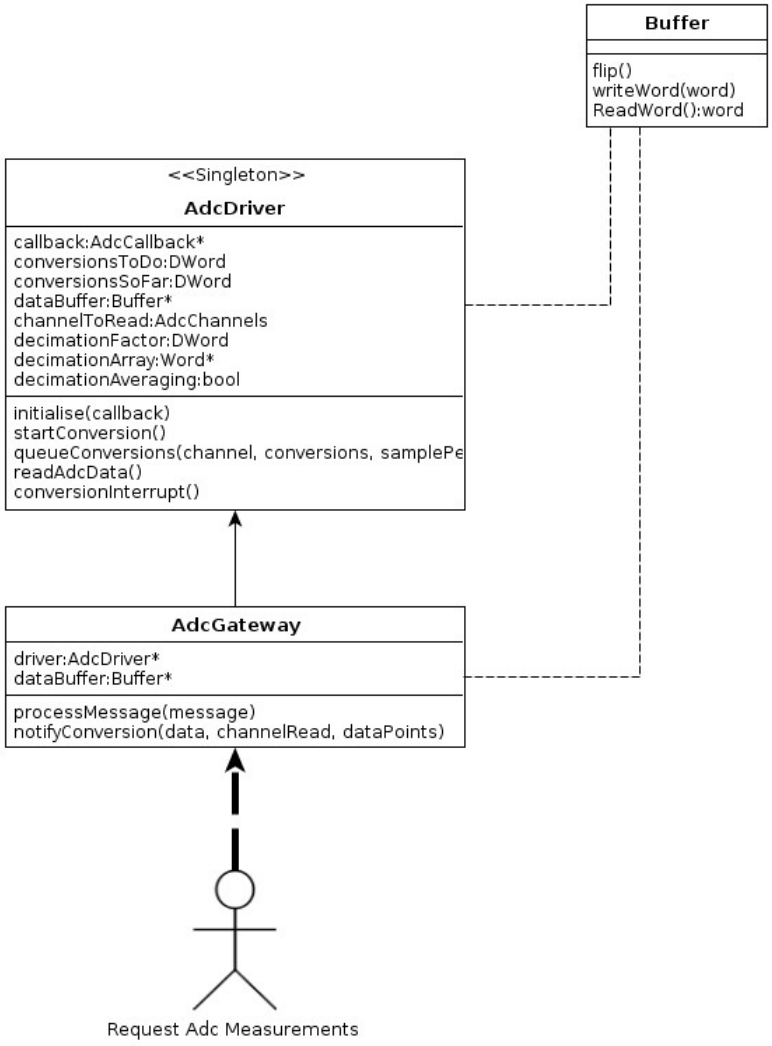


Illustration 48: ADC Test Actions (Only relevant methods and members)

“Request ADC Measurements” (0x730)

When a “Request ADC Measurements” message is received by the AdcGateway, it will instruct the singleton AdcDriver to queue the required number of ADC conversions on the requested ADC channel. The samples are done one at a time, and then once the required number of samples has been reached, the drive will notify the AdcGateway with the data. When the data is received back from the AdcDriver, the AdcGateway will pack it into a “ADC Measurements Response” (0x731) message and send it onto the message bus. To ensure that each set of ADC measurements fits into a standard message frame, no more than 100 measurements should be done at a time.

The sampling speed of the ADC is fixed at 239.5 cycles per conversion to ensure that high impedance signal conditioning circuitry will have minimal impact on the measurements

made by the ADC due to there being insufficient time to charge up the capacitor for the sample and hold circuit on the peripheral. In the “Request ADC Measurements” message the option is also given to implement decimation filtering. Decimation is a 2 step process of first applying a low pass filter to the input signal and then downsampling the data [17]. If the option is chosen to decimate, then the low pass filter needs to be applied on the test jig stack in hardware for the appropriate frequency, and the ADC driver will handle the downsampling.

3.6.3.4 Class Summary

The classes and their interactions are as follows:

- **GpioDriver**

  Is the standard GPIO driver in the EOS system. It is controlled by the GpioGateway and can be instructed to switch any outputs in its list of pin mappings. It will also notify the GpioGateway when any of the pins in it's list of pin mappings changes state. The driver is internally debounced to only check the state of the pins ever 20ms. The GpioDriver is a singleton class (it can only be instantiated once)
- **List**

  Is a generic list of objects and part of the EOS library. Objects added to or removed from the list are cast as void pointers to allow the list to be used on any kind of object.
- **GpioGateway**

  Is the standard GPIO gateway in the EOS system. It sends out notifications when any states change on any of the pins monitored by its GpioDriver, and will instruct the driver to switch an output to a different state if it receives an appropriate message to do so.
- **TestJigGpioGateway**

  A descendant of the GpioGateway used specifically on the generic test jig that allows pins to be removed from the GpioDriver when it received a “Start Pulse Test” Message, and then to return the pin to the GpioDriver when it receives a “Stop Pulse Test” message. The pins removed in this way are stored in a generic list.
- **FastInterruptDriver**

  The fast interrupt driver will set up a hardware pin to fire an interrupt when it changes state when it is instructed to do so by the FastInputGateway. The pin can

be set to fire on the rising edge, falling edge or both edges. It also tracks the width of pulses that are received (the time difference between rising and falling edge) in a rolling buffer to calculate the average pulse width. The external interrupt will call a static function in the FastInterruptDriver class, and as such the driver is created as a singleton (the static method calls the object through a static pointer member). Once a second (if there is an active pin being tracked) the driver will notify the FastInputGateway of the amount of pulses recorded over the second, and the average pulse width during the second.

- **FastInterruptPin**

  The fast interrupt pin object tracks all information about a pin that is currently under test as a fast input. Every time the FastInterruptDriver notifies the FastInputGateway of new pulse information (once a second), this information is passed along to the FastInterruptPin object which will add this data to any previous data. It can then be queried at any time for the pulse width since inception, average pulses per second since inception, average pulse width since inception etc.

- **FastInputGateway**

  The FastInputGateway owns the FastInputDriver, and when it receives a “Start Pulse Test” message will instruct the driver to set up the required pins external interrupt with the correct mode, and then creates a FastInputPin object to track the data. When the Gateway receives a “Stop Pulse Test” it will instruct the FastInputDriver to reset the required pin back to a normal input, and will destroy the FastInterruptPin object. During the test, if a “Request Pulse Data” Message is received, the gateway will populate a “Response Pulse Data” message with all of the information tracked by the FastInputPin.

- **StateTrackerTask**

  The StateTrackerTask sits on the message bus and listens for messages from the GpioGateway for when input and output pins change state. When a “State Change” message is received, it will update a list of known LogicalIds and their respective states. On receipt of a “State Request” message, the StateTrackerTask will look for the LogicalId in the request in its list of tracked values. If the LogicalId is found, then the state is sent out in a “State Response” message, however if it is not found, then nothing will be sent and the request will be ignored (assuming the message is for some other task)

- **AdcDriver**

  When called to do so, the AdcDriver will queue a number of conversions on a specified ADC channel. When all of the conversions have been done, the data is packed into a Buffer and sent to the Notify object (in this case the AdcGateway). The Adcdriver also handles the downsampling for a decimation filter if it needs to be implemented. The AdcDriver is implemeted as a singleton.

- **AdcGateway**

  The AdcGateway will wait for a “Request ADC Measurements” message , and when it receives it will instruct the AdcDriver to queue the required number of conversions on the required ADC channel. Once the conversions have been done, the driver will notify the AdcGateway of the data, at which point the gateway will pack the data into a “ADC Measurements Response” message and send it.

3.6.4 Test Jig Creation Process

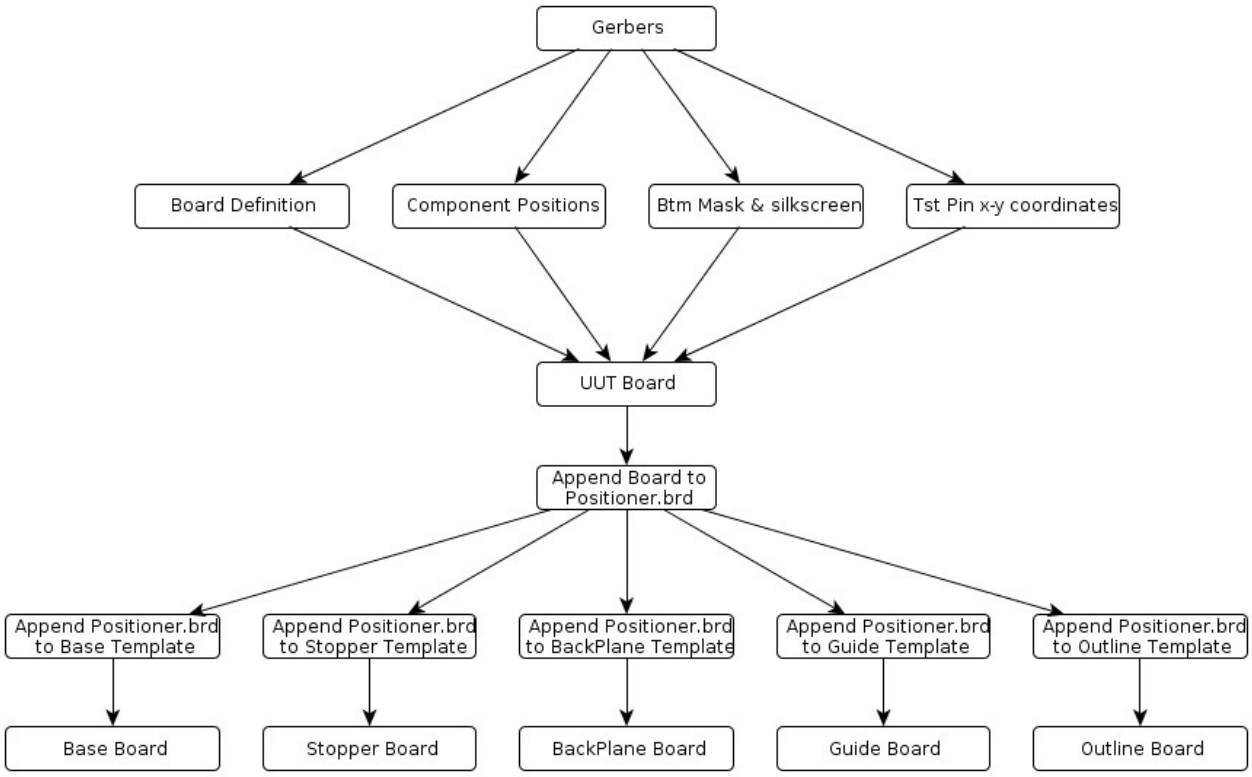


Illustration 49: Iteration 5 Test Jig Stack Creation

The test jig stack creation in iteration 5 further optimised the creation of test jig stacks by building on the work done in iteration 4 around the x-y box placement technique as well as the partially pre-routed template boards. In this iteration, instead of forcing the designer to

place the required board to test's gerbers into a .brd file, and then moving it to predetermined x-y coordinates, a “positioner” board has been created. The postioner board is not a PCB at all, but a virtual indicator of where a UUT can be placed to ensure that it does not interfere with anything. To prepare the board to be tested for positioning, the gerbers are pulled into a .brd file using gerbview:

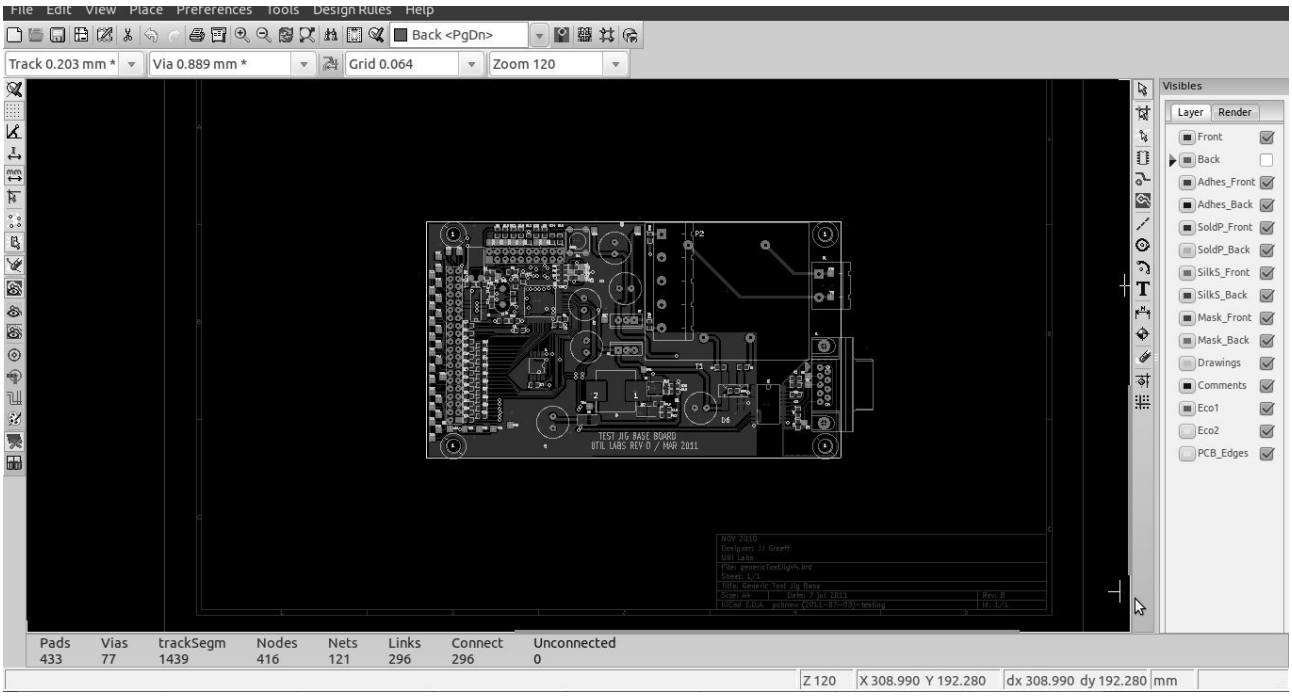
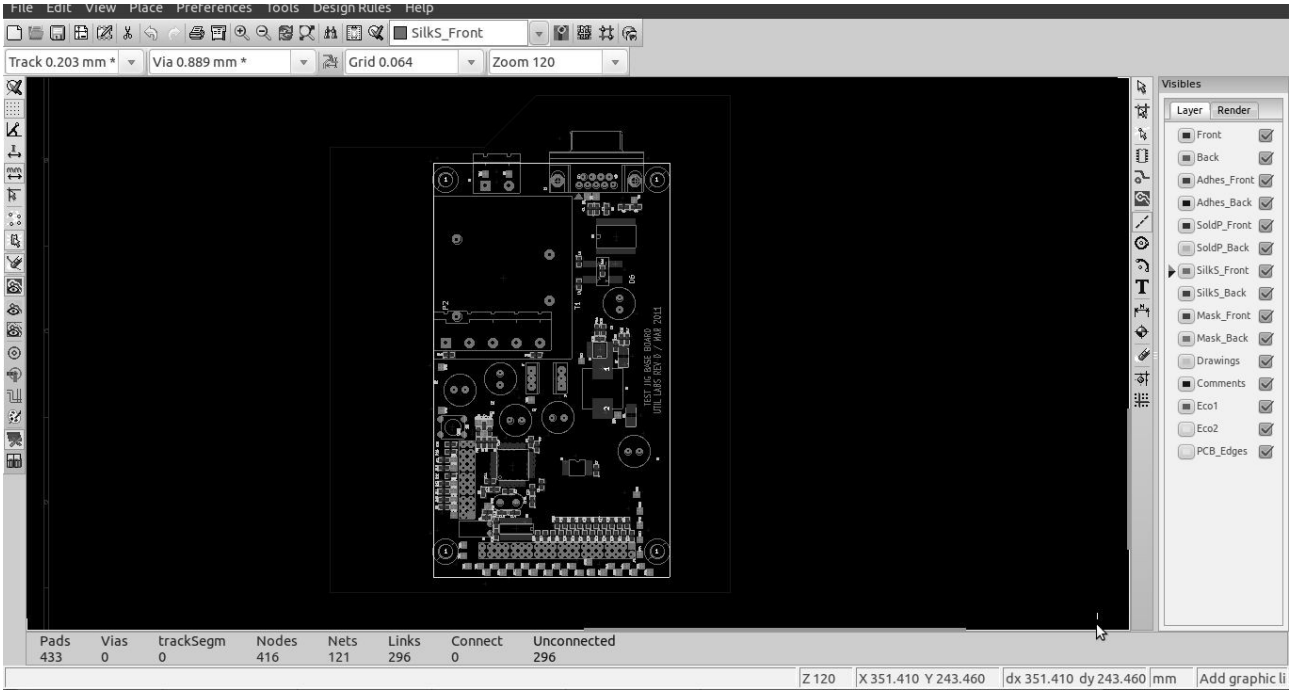
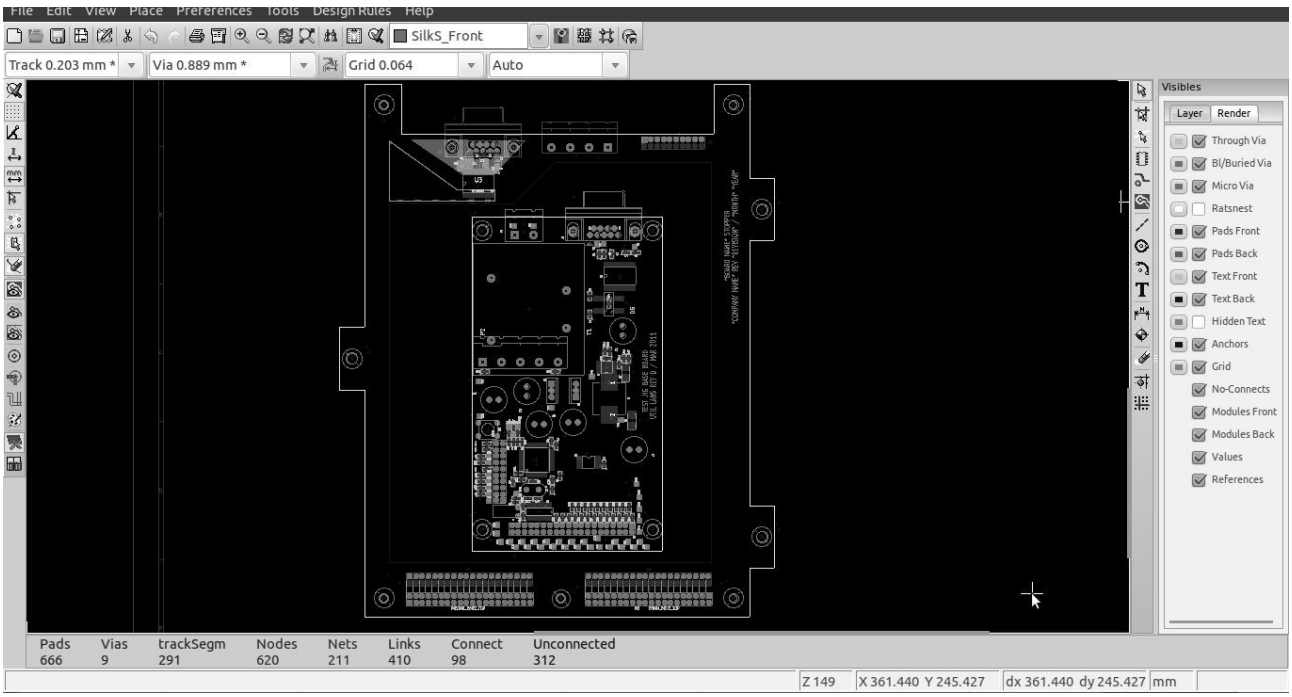


Illustration 50: Board To Test Created As .brd File

This board is now appended to the positioner board, positioner.brd, and all of the copper tracks are removed by using Edit->Global Deletions->Delete Tracks and Edit->Global Deletions->Delete Zones. The board is now positioned within the blue outline of the positioner on the board so that no part of the board protrudes, and all of the required test point objects are placed, as well as sufficient 3.15mm holes close to the test points to act as positioners for stabilising pillars between the stopper and backplane boards:



And once the board has been placed in the positioner, it can be appended to each of the 5 template boards (base.brd, stopper.brd, backplane.brd, guide.brd and outline.brd respectively) in turn:



Using this technique, all mechanical alignment issues are eliminated in the stack – provided the correct templates are used, the boards will align correctly. From there, a similar process is followed to Iteration 4 to complete the 5 boards in the stack.

- **Base Board**

  Using the Base-append.brd board created by appending the positioned board, the base board is created by removing everything but the board outline (which is converted to the top silkscreen layer) and the test point positions, so that the schematic can be created (using the base template schematic) that just routes the correct lines from the generic test jig boards to the stopper board where the test points are housed through some intermediate signal conditioning placed on the base board. The pin assignments of P1 and P2 are the same as iteration 3 and 4 and can be found in table6 in section 3.4.2. Once the schematic is created, it can be pulled into the base-append.brd board and laid out taking into account the positions of the test pins so that no tall components re placed below them, as they can interfere with the assembly of the test jig stack. Once the board has been laid out, these test pin positions can be removed. The Base-append.brd board designators need to be updated to show that the board is now a base board for a specific UUT board.

- **Stopper Board**

  The stopper board is created using the Stopper-append.brd created above. The outline of the UUT board is created as a silkscreen and all of the other components are removed (assuming there are no components mounted on the bottom of the test board that are higher than 14mm, as these will require cut outs on the stopper board). The schematic can be created by taking the stopper schematic template and adding the test point schematic objects to it. Any additional signal conditioning not done on the base board (like for example placing transducer hardware to read light as opposed to electrical signals) is now added to the schematic. To communicate with the UUT via a debug port, an ICoupler is added to ensure that the electrical isolation is maintained between the UUT and test PC. The ICoupler is pre placed on the Stopper.brd template with an appropriate amount of heat syncing and 4 line stubs that are used to connect +5V,gnd,RX an TX to the ICoupler.
  To power the chip either a pre-laid schematic of a switch mode power supply can be used, that must then be laid out around the test pins, or +5V can be brought up from the base board. If the power supply is brought up from the base board, then the U1 circuit can be removed from the schematic as in Iteration 4. The stopper board also contains the presence switch which is just a micro switch that needs to press against the UUT when the test lever is actuated on the test jig. A net just

needs to be assigned to the PRESENSE line on the schematic template.

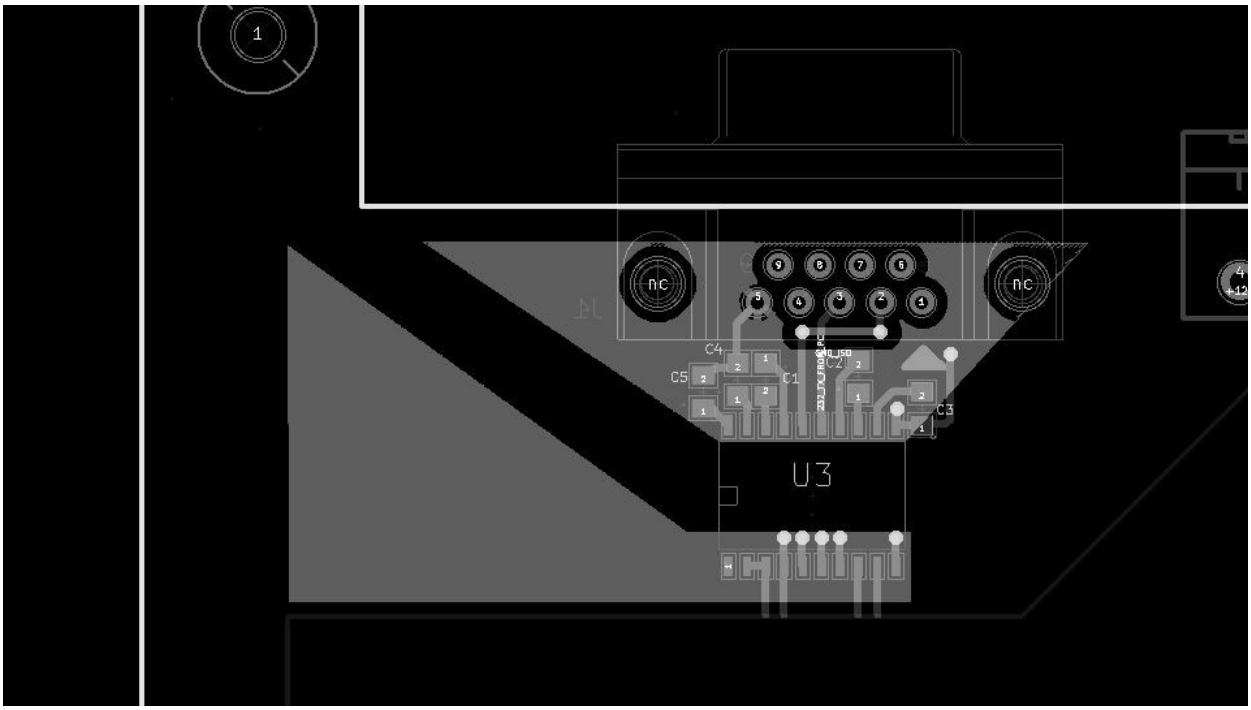


Illustration 51: Close Up Of Heat Sync And Stubs On ICoupler On Stopper Template

Once the schematic is created, the netlist can be pulled into the board and laid out, ensuring not to place any of the signal conditioning components too close to the stabilising screws for the support pillars between the stopper and backplane boards, and placing the presence switch micro switch somewhere that does not interfere with any other components when it is actuated. The Stopper-append.brd board designators need to be updated to show that the board is now a stopper board for a specific UUT board.

- **Backplane Board**

  The backplane board is created from the backplane-append.brd created above. The backplane board serves the same purpose as the 2nd stopper board used to perform in iteration 4 in that it offers stability to the assembly, and creates (together with the stopper board and support pillars) a rigid plate for the test pins to press against during the test. To create the backplane board, the imported outline for the board is moved to the silkscreen layer, and all components but the test pin holes and the stabilising holes are removed from the board. There is no schematic for the backplane board and therefore no tracks, and no components are fitted onto it. When manufacturing the backplane board, only the top silkscreen layer need be sent along with the board outline layer so that the manufacturer knows not to add

any copper. The backplane-append.brd designators need to be updated to show this is now a backplane board for a specific board.

- **Guide Board**

  The guide board is created using the guide-append.brd created above. The outline of the board to test is created as a silkscreen, and then the bottom silkscreen of the board is used to create cut outs for any components mounted on the bottom of the board, and the test points and through hole lead positions of other components on the top layer of the UUT are used to create holes for those leads to go through. All of the components on the board are then removed. There is no schematic for the guide board and therefore no tracks, and no components are fitted onto it. When manufacturing the Guide board, only the top silkscreen layer need be sent along with the board outline layer so that the manufacturer knows not to add any copper. The guide-append.brd designators need to be updated to show this is now a Guide board for a specific board.

- **Outline Board**

  The outline board is created from the outline-append.brd board created above. The only step required for making an outline board is to draw in the Board definition layer an outline around the board to test at a distance of 0.5mm away from it and then removing all of the components on the board. The board designators need to be updated to show this is a outline board for a specific UUT board. The outline board has no schematic or copper layers.

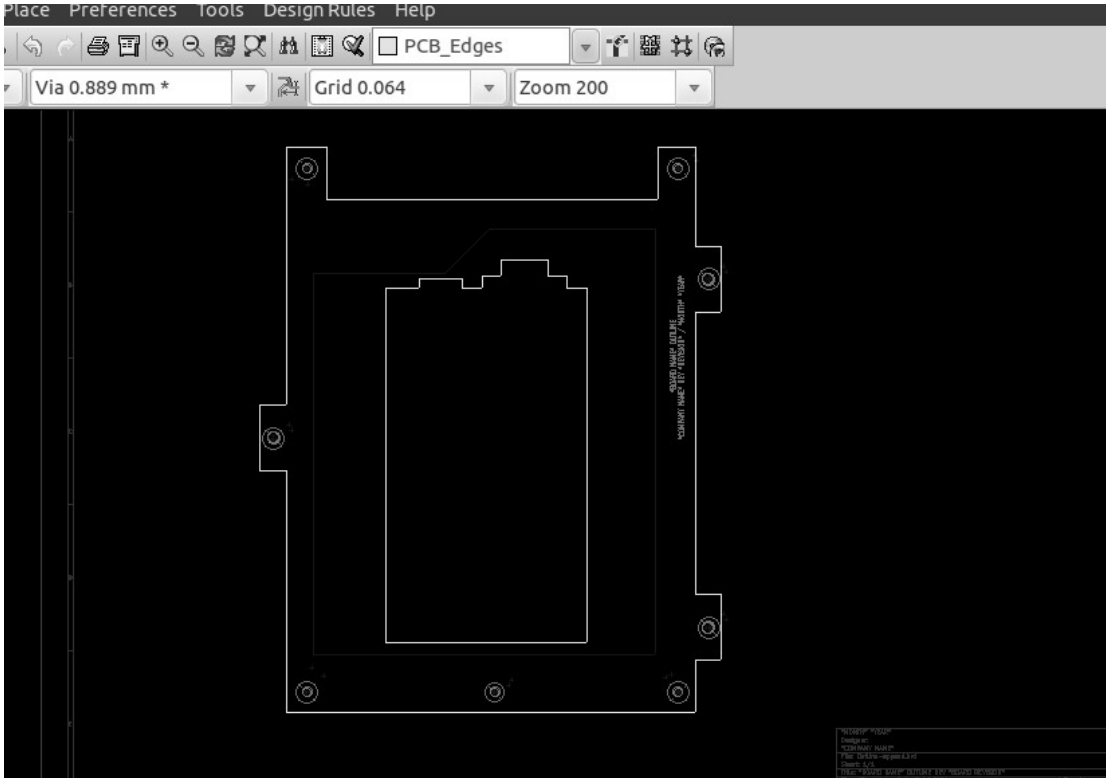


Illustration 52: Outline Board Created Using Positioner

3.7 THE JIG MECHANICS

There were two versions of the test jig mechanical base, an initial version created around the 3rd and 4th iterations of the jig, and then a 2nd mechanical version around with the 5th iteration of the generic test jig system was built. The reason for the change was that the first version was designed in house with a number of mechanical components that would have needed to be custom made, and in the end a previously designed jig that had a similar lifting plate arrangement was found that would work out significantly cheaper than the in house design. The hardware housing of the jig allows the test jig stack to be moved up and down using an actuating arm on the outside of the box, and it is this action that will cause the test pins to either make or break contact with the UUT depending on the position of the lever.

The lifting plate houses all of the boards of the generic test jig system with the exception of the outline and guide boards (they are mounted on the top plate of the jig so the operator can position the UUT), so when the arm is actuated the whole assembly moves up and down so as to minimise strain on any sensitive components on the stack. All of the wiring for the test jig stack can be done from one side of the plate, which is held in place with 8x4mm screws. When swapping between different test jig types, the technician need only drop the lifting plate by removing the 8 screws, and then swapping the test jig stack, outline and guide boards. As templates are used, the wiring for all of the test jig stacks is identical.

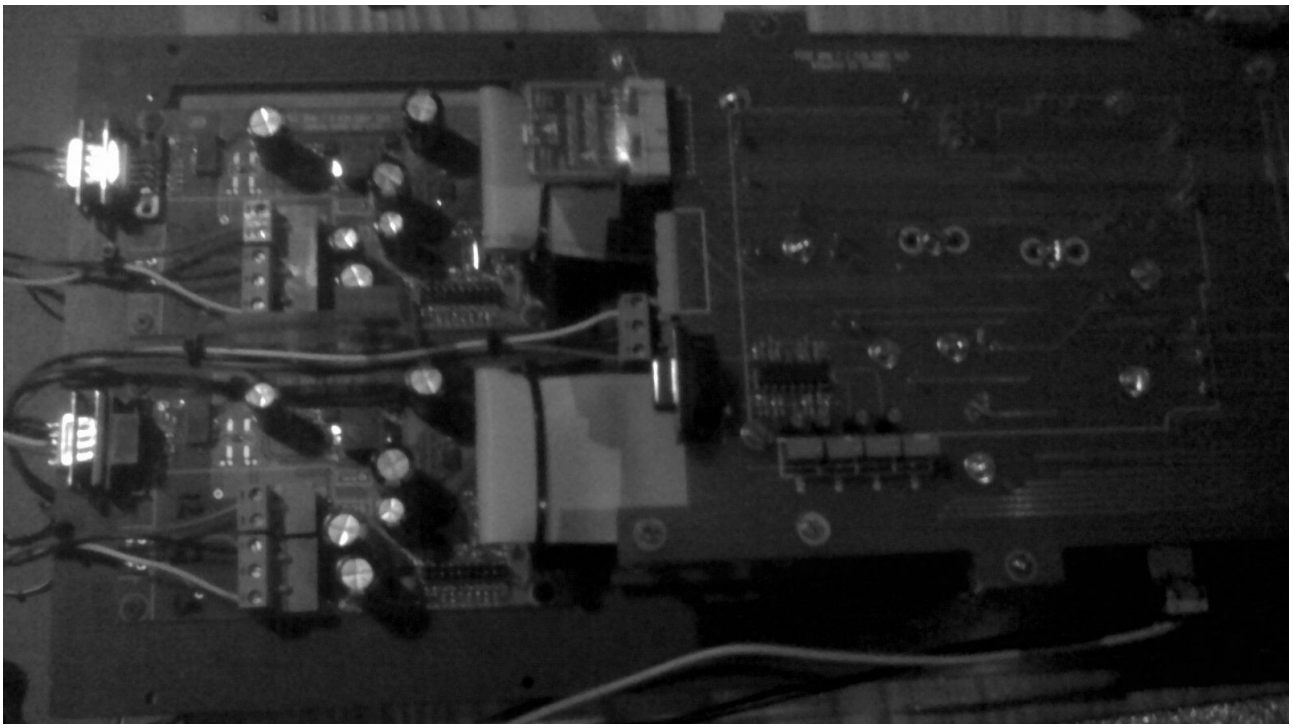


Illustration 53: Generic Test Boards and Test Jig Stack Bolted To Lifting Plate

The mechanical jig also houses some components to assist the generic test jig system in the test process:

- 3 LEDs are added to the front panel of the mechanical jig that are always wired to the stopper board which indicate RUNNING, PASS and FAIL states (orange, green and red respectively)
- 2 power supplies are mounted in the base of the mechanical jig which can supply power to the generic test jig boards.
- The mechanical counter indicating the lifetime of the mechanical jig and the mechanical interlock stops the operator opening the lid of the jig while the test is running, both of which are wired to the base board.
- The lid of the mechanical jig has a number of spring loaded pillars that place downwards force on the UUT when the test pins are pressed against the board to keep it in place.

3.8 TEST PC JAVA SOFTWARE ARCHITECTURE

The application that runs on the test jig PC houses the Java framework as well any implementations that have been done for boards that have generic test jigs created for them. To add a new jig implementation to the framework, the architecture of the framework must first be understood.

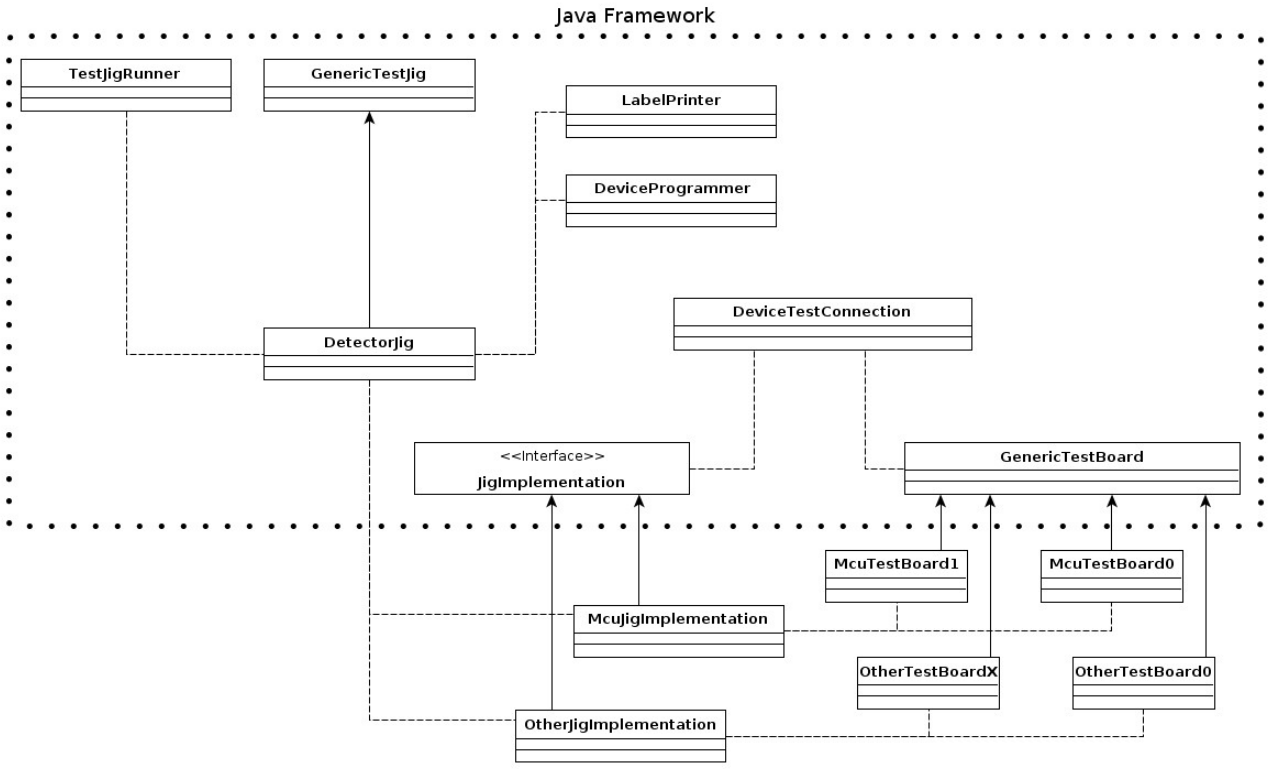


Illustration 54: The Java Architecture Overview

The main classes that need to be taken into account when writing a new test jig implementation are:

- **JigImplementation interface**

  This is the interface that the specific implementation for the test jig being created needs to implement. The JigImplementation interface forces the specific implementation to add a run method, runPreProgramTests method and a runPostProgramTests. The run method is the main loop for the test sequence for the jig being programmed, and is the method that is called by the DetectorJig Class if it is found that the test jig stack corresponds with this jig implementation.

- **GenericTestBoard**

  The GenericTestBoard class represents the a generic test jig board. It allows the

test jig system to invoke any of the actions discussed in section 3.6.3 (reading voltages, reading inputs etc). The class also adds a method to read the select line Id from the generic test jig board, which will allow the detector class to determine which test jig stack is connected to the two (or more) generic test jig board. When creating a new test jig implementation, the GenericTestBoard class is inherited from to create two specific implementations of test jig boards. In the example used throughout this project, the two boards created are McuTestBoard0 and McuTestBoard1. The 0 and 1 represent the 5 bit value of their select lines. The constructor for the McuTestBoard0 and McuTestBoard1 take in a com port string (this is the name of the comm port on the system that will be used to connect to the generic test jig board). Other than the constructor, the only other thing that is added to the McuTestBoard0 and McuTestBoard1 is a number of public constants that indicate what is connected to the inputs and outputs of the generic test jig board - in other words this is where all the logical mapping happens.

- **DetectorJig**

  The DetectorJig is a descendant of a GenericTestJig. The GenericTestJig class represents a physical test jig, which has a test jig stack and a number of generic test boards, a label printer to print labels, etc. It gives the DetectorJig the ability to auto detect which com port is connected to which generic test board, as well as reading th 10 bit of the test jig stack. Once the DetectorJig exists, and is run (this method is called by the TestJigRunner object) the DetectorJig will assume it has a new jig attached to it. It will then read the number of com ports that are attached to the test jig in question from a configuration file (this will usually be 3, for 2 generic test jig boards and a UUT debug connection), and then go through the process of automatically detecting which com ports contain GenericTestJig boards and which are currently not communication (since at this point the UUT will not be powered). Once the com ports are detected, they will be placed in an array where the size of the array corresponds to the number of com ports, the final entry in the array is the UUT debug port, and the other entries in the array the GenericTestBoard com ports sorted in order of their 5 bit ids. From here the 10 bit test jig stack Id is read, and the specific implementation that corresponds to that id is instantiated.

The Generic Test Jig Framework fits on top of an existing code base that has been developed at Util Labs over the last few years. Part of this framework is the DeviceTestConnection communication class, which allows serial communication through

standard com ports. All messages moving through the DeviceTestConnection class will be logged to file on the test jig PC. This means at any time the results of a test, or any part of a test can be traced with a timestamp, allowing very precise tracking of when problems arrive which should allow for very accurate tracking of failure trends while using the generic test jig system.

3.8.1 Creating the TestBoards

Using as an example the implementation of the MCU test board that has been the guinea pig for this project all along, the following two classes were created:

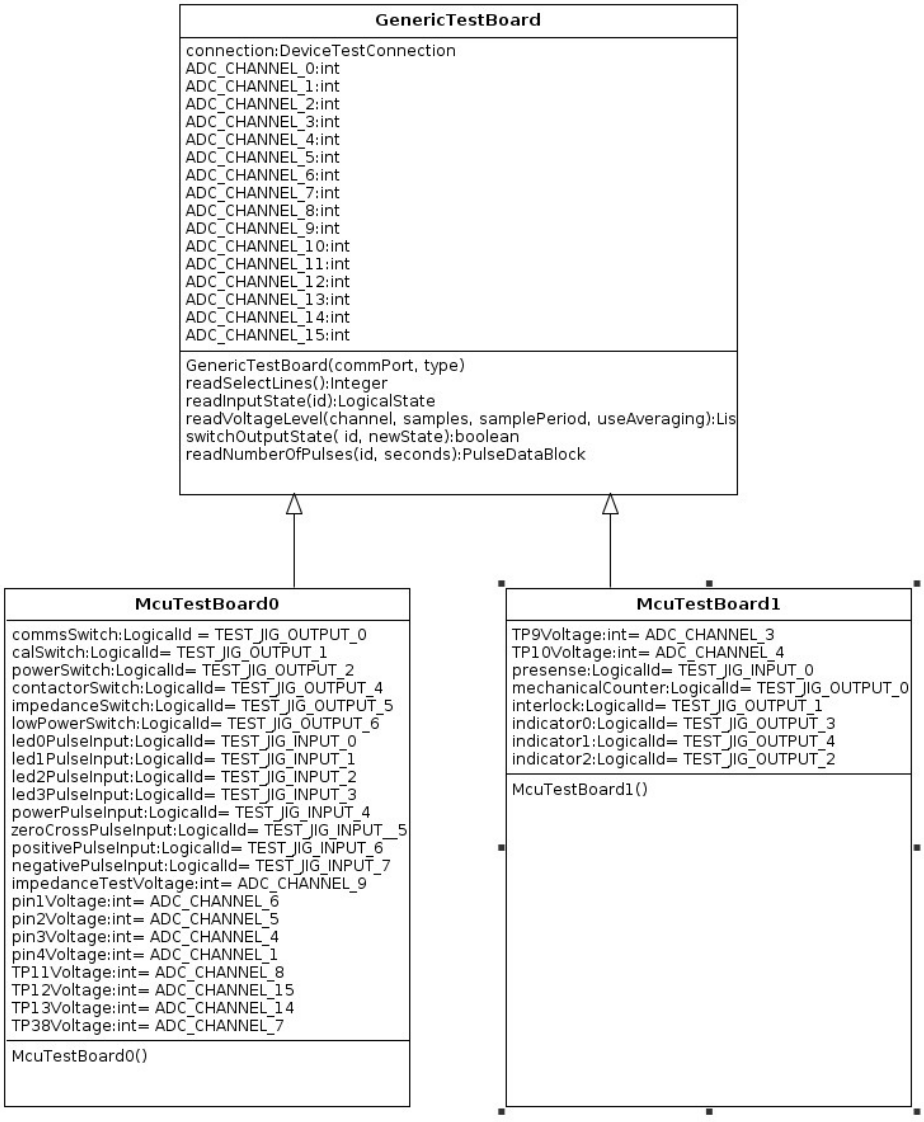


Illustration 55: TestBoard Specific Implementations

The two implementations both overwrite GenericTestBoard and just add a number of constants to it which allow the inputs and outputs on the generic test jig boards to be used

in the McuJigImplementation, and to ensure that the wrong input/output is not referenced on the wrong board (TEST_JIG_OUTPUT_1 exists on both of the McuTestBoard implementations, but those inputs are mapped to calSwitch and mechanicalCounter respectively). The McuJigImplementation will use these constants for all of the interactions with the generic test jig boards.

3.8.2 Creating the Implementation

Once the generic test boards have been mapped out, the actual test sequence code can be written. Once again the MCU test jig example is used.

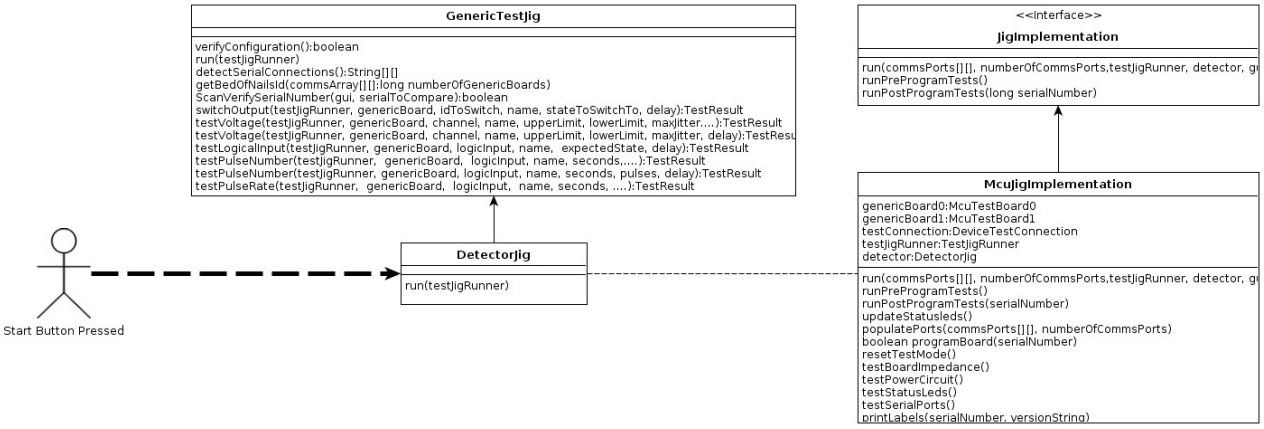


Illustration 56: Test Jig Implementation In The Framework

To add the implementation to the Java system, first the Implementation class needs to be created. This class implements the JigImplementation interface and as such inherits its run methods. The standard tests are already available to the Implementation through the Detector Jig (which is passed into the run method when invoked), but any tests that are not simple measurements and require some set up before the time need to be added to the implementation. As such the following tests are added to th implementation:

- resetTestMode
- testBoardImpedance
- testPowerCircuit
- testStatusLeds
- testSerialPorts

Each of these tests require some set up before they can be run (for example, before the board impedance impedance can be measured through the adc, the test jig stack must first set the low impedance relays to the correct state, and then switch on the low power supply before the measurement can happen). Generally these special tests have 3 phases nl. Set up, measure and tear down. The set up of the test happens when the appropriate preliminary steps are performed before a measurement can be taken (for example setting the relays to the right state in the impedance test example). The measurement happens once the tst has been set up, and the tear down happens once the measurement has already been taken, and the test jig is then returned to its original state.

When the run method of the Implementation is invoked, a TestJigRunner object is passed in. This is the object that all the results of tests should be reported to. The TestJigRunner can be queried to find out if any of the previous tests have failed. Generally in a test sequence, if any one test fails the entire jig fails. On failure, a failure code can be passed into the TestJigRunner as well, which will allow the printer to print the failure code which is then stuck to the board when the operator removes it from the test jig to facilitate failure tracking. The testJigRunner also allows the Implementation to write data to the history log of the test jig, so for example pass or failure conditions can be saved in the file, so that later when the file is analysed for historical data, it is both parseable to read the messages out of its history, but also human readable as the log information will be saved as plain text. The TestJigRunner also allows logging to the console to indicate the status of the tests to the operator.

There are 3 phases that most boards go through when being tested nl. The Pre-programming phase, the programming phase, and the post programming phase. The pre-programming phase handles all hardware on the UUT that required no interaction with programmable devices, for example power supplies, checking impedances, checking opto couplers etc. After the board has been tested that it is safe to start programming the programmable device(s) on it, the programming phase happens. If the device(s) is successfully programmed, then the post programming tests are run. Post programming tests are tests that require some interaction with the processor for them to pass, for example checking that an external memory chip is working by writing a file to it from the programmable device on the UUT and reading it out again. In order to get full test coverage it is important to ensure as much of the interaction on the UUT is tested. For example if a Flash memory chip is being tested before the programming has happened,

the chip may pass the test of reading and writing data, but if there is a bad solder joint on the pins on the programmable device that will normally uses that chip, by running the test through the programmable device, this error will be missed.

3.8.3 Add The Implementation To The Detector

Once the implantation for a test jig stack has been completed, this implementation can be added to the Detector run method.

```
long testJigId = getBedOfNailsId(commsPorts, (numberOfExpectedPorts-1));
TestJigid id = TestJigid.getEnum((int)testJigId);
JigImplementation implementation = null;
switch (id){
   case MCU_TEST_JIG:
      implementation = new McuJigImplementation();
      implementation.run(commsPorts, numberOfExpectedPorts, testJigRunner, this, gui);
   break;
   case INVALID_ID:
   break;
}
```

Illustration 57: Adding The Implementation To The Detector

The code snippet above shows where the new implementation needs to be inserted in the DetectorJig to allow it to be run when it's id is discovered. The test jig ids are stored in the TestJigId enumeration, and a new implementation with a new id needs to be added in there before code can be written for it.

3.10 CONCLUSION

The first successful fully integrated test using a test jig stack made using the generic test jig framework was run on Thursday 9th of June 2011. Previous tests were done but this one was the first one where the generic test jig framework created a board stack that fitted into a mechanical housing, and could be used by an operator to test a PCB (In this case the MCU V3 Rev D board), which passed all of the tests as expected.

The Iterative design approach proved to be very suitable to this project since even though there was an initial specification put forward by the internal customer (in this case the production department of Util Labs), this specification proved to be incomplete and it was only through a process of continual feedback and review that a suitable system was finally designed. Through the 5 iterations, all of the initial design requirements were met, and a number of features not initially anticipated to be necessary were added to create a framework that should greatly increase the test capabilities of Util Labs and its contract manufacturing facilities.

CHAPTER 4

4.1 INTRODUCTION

The generic test jig system over 5 iterations has become the de-facto standard method of testing hardware at Util Labs. The MCU V3 board that was used throughout this project as a guinea pig was chosen simply because it is the most complex board in the Util Labs product line to test. Since Iteration 4, 4 of the Util Labs products have so far been implemented in the generic test jig system (The Eddi display, The LCD front end module to the MCU V3, and the generic test board itself), and the decision has been made to backdate the tests to all existing devices in the product range as well. Design integration is also working very well because all 4 devices are still in development, and every time a new version of hardware is released the test jig associated with that hardware can be updated almost immediately (as can be seen by how quickly the changes to the to the guinea pig board were absorbed in Iteration 4 and 5).

In addition to being the de-facto standard at Util Labs, it has also attracted the attention of other players in industry, and there is an opportunity to commercialise the system by implementing the communication between the Test jig PC and the UUT in more generic way to allow other protocols to be implemented.

This chapter analyses the performance of the generic test jig system when measured against the requirements in chapter 1 and the product specifications set out in chapter 2.

4.2 RESULTS – USER REQUIREMENTS SPECIFICATION

Attribute	Requirement	Result
Cost	Physical cost per text fixture must be < R20 000	The Current Test Fixture comes to less than R20 000 (not including the label printer), and in addition to this the ability to share mechanical jigs between manufacturing runs for different products pushes the cost down even lower
Scalability	Must be able to add more test fixtures to a production run to increase production output at a cost of < R20 000 per test fixture	Since the test fixtures can share a printer on a network, the cost of adding another test fixture is just the cost of the pc and bed of nails stack if a set of spare mechanical fixtures is kept in stock so < R5000
Flexibility	Design must be generic and flexible enough to be able to design the hardware and software for a moderately sized jig in 1 month	The current system of templating allows for a jig that does not require any special tests or protocol changes to be designed in <1 week, without a programmable device on the UUT < 2 days.
Integration	Design should be integrated well enough with current design systems, that if a test pin needs to be moved or a test needs to be re-written that the impact will be less than 2 man days.	Between the 4th and 5th iteration, the guinea pig board changed, and since no additional tests were required (the test pins moved around and the leds on the board were removed), a jig could be made that was compatible with both the new and old version of the board in <2 days.

Attribute	Requirement	Result
Maintenance	Costs should be low enough that a hot swappable set of jigs can be kept on hand for failure conditions for under R20 000, and install time low enough that for line stoppages <30min on jig break down.	With the jig able to auto detect the test jig stack, in the event of a broken pin or electrical damage to the bed of nails, the lifting plate can be dropped and the test jig stack swapped in less than 20 mins with a cost of <R1000. Spare Generic test boards can also be kept on hand for < R300 a set
Traceability	The jig should support traceability in the factory, and remove some of the responsibilities from the operator to sort faulty boards into the correct areas.	By adding the ability to print failure modes on the same printer that prints serial numbers for manufactured units, there is now the ability to trace failures through the factory and easily make sure faulty boards arrive at the right station.

Table 11: Results - User Requirements Specifications

The generic test jig system conforms to all of the requirements set down by the original user requirements specification.

4.3 RESULTS – PRODUCT SPECIFICATION

Functional Specifications:

Attribute	Requirement	Result
Inputs	Jig must have at least10 binary inputs, 100kΩ input impedance and be 20V tolerant on logic pins.	As a base with 2 test boards, 18 inputs are available and can be expanded. The input characteristics have been split away from the generic jig the the stack, and as such the impedance can be pushed into the MΩ range by buffering with op-amps, and the pins have been tested to 20V levels on the MCU test board with no problems.
Voltages	Jig must have at least 10 voltage inputs, 100kΩ input impedance and be 20V tolerant on adc pins with at least 1% accuracy.	As a base with 2 test boards, 18 inputs are available and can be expanded. Signal conditioning on stack can buffer the input impedance to MΩ levels. Pins tested with 20V input voltage successfully. The slowest conversion speed is used on the ADC to ensure the specified ±2bits accuracy on 12 bits is reached.
Pulse in	Jig must be able to measure at least 4 frequency inputs up to 2kHz, voltage tolerant to 20V	Using the external interrupts on the STM32F103 allows every input in the system to also be a pulse input with the same voltage characteristics. The external interrupts can sense pulses of as short as the width of the internal APB2 clock (36Mhz). Pins have been tested successfully with 1kHz signals

Attribute	Requirement	Result
Counters	Jig must be able to keep a count of pulses on at least 2 inputs, up to 2000 pulses per second, voltage tolerant to 20V	The same fast input module that handles the pulse in can also act as a counter with the same characteristics. The counter value is returned as a 32bit DWord value, which will then be the maximum counter value.
Comms	The test jig must have at least 2 serial communication lines to the unit under test @ Standard UART communication of at least 115200 baud, 8 bits, no parity, 1 stop bit	There is as a standard only 1 communication line to the UUT instead of 2, but a second can easily be added if required. The second communications port was meant to be a programming port and this was exchanged for a dedicated JTAG in the test jig.
Signal Injection	The test jig must be able to inject 5 or more signals up to 1kHz, at voltages up to 230Vac (for mains injection) up to 1A.	Outputs are capable of driving 500mA loads, and as such, D2n telecommunication relays were used on the guinea pig board which have a maximum voltage of 230Vac, can handle 3A loads and frequency up to 100Mhz.[31]

Table 12: Functional Specifications Results

Electrical Specifications:

Attribute	Requirement	Result
Supply	The test jig must run on 230Vac and require no external power supply	The Test jig is powered by 230Vac, and if the option is used for an external supply, this supply is external to the test board, but still inside the mechanical Jig.
Connector	Power must be supplied by a standard IEC power connector.	An IEC power connector is used with a built in fuse
Standby	The test jig, when not injecting	When not running tests, the test jig will

Attribute	Requirement	Result
	signals must draw less than 2W of power	draw just over 1.5W (The majority of which is due to the ICouplers.)
Isolation	Isolation between the UUT and all communications lines must be > 1kV	All lines are isolated with ICoupler devices that offer 2.5kV isolation
Fuse	The Test Jig must have a replaceable fuse	The fuse is built into the power socket, and is replaceable.
Physical Isolation	The operator must not be able to touch any of the test pins while the test is running.	There is a mechanical interlock that can be activated during the test to stop the operator opening the jig while the test is running.

Table 13: Electrical Specifications Results

Interface Specifications:

Attribute	Requirement	Result
Storage	Test result storage must be non-volatile and be able to store at least 10 000 results	Results stored to file, size is limited only to hard disk size
Access	Test results need to be accessible to a windows or Linux operating system	Test Jig Application is native to Linux
Interface	Once Installed, the required user interaction should consist of no more than single actions, for example: close lid, push button, scan label etc	As Specified
Comms	RS232/RS485 communication will take place at 115200 baud, 8 data bits, no parity bit, 1 stop bit	The ICouplers and USB-Serial converters both handle at least 115200 baud
Language	Graphical communication to the user must all be in English	As Specified

Table 14: Interface Specifications

4.3 CONCLUSION

As can be seen from the comparison tables, the generic test jig system has conformed to all of the requirements placed at the start of this project.

CHAPTER 5

5.1 INTRODUCTON

The project up until this point has been a resounding success as can be seen by the results obtained in the previous chapter. Throughout the course of the project, a number of ideas have come about that would make the generic test jig framework significantly more powerful, not only as a design tool, but as a commercial product. There were however also some things that could have been done better, and it is with this in mind that the project is reviewed.

5.2 CONCLUSIONS

The Iterative design approach was definitely the best approach to take in this project, as the initial requirements were not all available at the start of the project, and some things had to be added throughout the course of the design. However, since each iteration of the design required changes to be made to the design, a large number of PCBs were made throughout the course of the project. In the context of Util Labs this was not a problem as there is an agreement in place with our preferred PCB manufacturer that tooling costs are not added to the manufacturing cost of prototype board due to the large number of boards we order, but even taking this into account, the project went over budget by about 20%. Iterative design is a very powerful tool, but in a predominantly hardware design project, more time should be spent in the first iteration designing a proper simulation environment (like for example the cardboard mock-up system used in iteration 2 could be used more as it is a cheap and simple way to see if everything fits together properly).

An interesting theme throughout working on this project and interacting with different stakeholders, is that technology as a concept in a design is not confined to the technical quality of the solution. The best ideas to come out of this project have been around Templating, and the ability to cut down on design time, rather than the architectural design of the code or even design of the actual hardware. In future, I think it would be a good idea to spend a little more time getting the stakeholders in the project more involved, specifically the non-technical stakeholders as unique technologies can grow from such interactions.

An interesting side note on the project though is the commercial potential that has arisen. At the start of the project during the literature review, a number of systems were explored, and the two systems that are closest in function to the generic test jig system are flying probe systems and traditional beds of nails. During a normal design cycle, products are designed, and tested by hand, but flying probe systems offer an alternative to this technique, as they can quickly be programmed to become a virtual test fixture for a prototype. However when the product is then completed, a bed of nails permanent test fixture can be designed which then allows the product to go into mass manufacture. The templating process has cut the design time so low on the generic test jig system that there is an opportunity to compete with flying probe systems, as for a simple board, a test fixture can be completed within a week (provided the PCBs arrive from the manufacturer in time),

which puts it in serious competition with these systems. The competitive edge however, is that a flying probe system has no way to migrate from a “prototype” tester to a “production” tester, which is where the generic test jig system fits in. This has been discussed internally, and a few initial discussions have been made with some people in industry on the concept, and the initial feedback is very positive. This will definitely be something that will be explored in future.

5.3 RECOMMENDATIONS

The following recommendations are made by the author to ensure the most is got out of the project from this point forward:

- The generic test jig system is already being established as the de-facto standard for testing at Util Labs, but it would be useful to backdate this process to all of the other products in the Util Labs portfolio. The ability to use the same operators on all test equipment and to rapidly change the production line to a different kind of product is a valuable commercial “edge”.
- The templating process has made the design of test jigs simple enough, that with some training, other departments in Util Labs can take advantage of the technology – specifically the quality department that consistently needs to test and measure on products that come out of the field presenting some unknown problem. A number of jigs could definitely streamline the process of fault finding for known problems.
- Some additional tests are required for upcoming products in the Util Labs portfolio – specifically capacitive sense buttons need to be tested. This functionality should be added soon.
- The commercial value of the generic test jig system in its own right should not be ignored. There are no products currently on the market that can offer the same level of production test capability within the time frame offered by this system!

5.4 PROPOSED FURTHER WORK

As is the nature of iterative design, there are always a new directions to look into to make a product better. This is a list of useful functionality that the author feels would be very valuable to continue developing:

- Camera interface

  It would be useful to have a simple camera interface to do automated visual inspection of LCD screens on products.
- Standardised network bridge

  The current debug information is sent to and from the UUT via RS232, but there is no reason that a small processor cannot be placed as an intermediary between the UUT and the test jig pc which will then allow the conversion of messages from RS232 to UART, SPI, I2C etc etc.
- Programmer support

  There are only so many widely used programming interfaces in industry today, and it would be useful if the generic test jig natively supported SWIM and serial bootloading of the major processors from ST, ATMEL microchip etc.
- Port To ARM11/ARM9

  The current system of having the test application on the test jig PC next to the jig works well, but a better solution would be to run Linux on a small ARM11/ARM9 mini computer board, and build the board directly into the test jig so that it is a single unit.
- Web Server

  A web server should be installed on the test jig PCs to allow for easy retrieval of logs as well as simple updates of the Java application in situations where not test jig implementations are added.
- Reporting

  Create Cron jobs on the system to email a list of people a daily report of the number of devices test on the test jig, and a breakdown of different failures that were found to allow for easy tracking of failure trends.
- Automated templates

  The templating process can be fully automated, and any time cut off of the design will mean faster delivery of test equipment.