Remote Control and Monitoring of a Hybrid Power System

Prepared by Tristan Phillips

Supervised by Michel Malengret

University of Cape Town

2008

Declaration

I declare that this thesis is my own work. Where collaboration with other people has taken place, or material generated by other researcher is included, the parties and/or material are indicated in the acknowledgements or references as appropriate.

This work is being submitted for the Master of Science Degree in Electrical Engineering at the University of Cape Town. It has not been submitted to any other university for any other degree or examination.

I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own.

_____

Tristan Phillips

Abstract

Increasingly, South Africa is developing a need for grid-independent power systems. Many options exist already in the form of diesel generators, solar panels and wind generators. This project is to do with marrying the various sources of power with efficient and intelligent control. Specifically, the terms of reference describe an existing solution that has been developed. The focus of this thesis is to design a replacement system for the existing system which incorporates many communication technologies and additional features. The design centres on a central digital signal processor board. The redesign of this hardware and software will form the bulk of the content of this document. The scope of the design encompasses all the communications and new features including remote programming, control and monitoring software and robustness. The new system is to form the infrastructure or base hardware and software for many different power-related applications. A specific example of such an application, the hybrid power system, is discussed in detail to demonstrate the success of the design.

Acknowledgements

I would like to thank my supervisor, Mr Michel Malengret, for all the help with power electronics and for the motivation to complete this thesis.

Thank you to MLT Drives who made it financially possible to produce prototypes and supplied resources where necessary.

The following engineers worked closely with me get this project off the ground:

- Mr Richard Parry
- Mr Martin Becker
- Mr Stanley Adams
- Mr Len Wright

Thank you to Mr Daniel Willemse for building test rigs, and for helping with technical issues.

Table of Contents

List of Figures

List of Tables

1. Introduction

Every day, we see technological leaps, but behind the scenes, the driving forces of our modern infrastructure are our power systems and communication networks. Power generation is often seen as a static technology that is fully developed and can not make any drastic improvements. It is this misconception that sometimes prevents further research and development into improving power systems.

Recently, especially in South Africa from 2006 due to power shortages, many efforts were made to improve efficiency in power generation and demand side management. Furthermore, reliable power is critical to the economy, both in developed and developing countries.

A power system will inevitably fail if left alone for periods of time. However if there were a proper warning system in place, failures can be predicted and prevented and hence give improved reliability. What most power systems today lack is a proper interface to the advanced communication networks we have around us.

In many developing countries and rural areas, much development is taking place. Infrastructures are being built up to improve the quality of life for local communities. When power is brought into rural areas, micro-enterprises emerge and collectively help to bring wealth to these areas and thus grow the economy of the country.[26] The cost of connecting a rural area to the national power grid is often too high and results in alternative solutions being investigated. One such solution that is becoming increasingly popular is the mini grid.[27]

According to Mauch, Ayoub and Jacquin[24], a mini grid is powered by a combination of photo voltaic (PV), wind, micro-hydro, fossil fuel gensets, and other sources. They typically supply multiple users and may or may not be connected to the distribution grid of the local electric utility.

Mini grids are ideal infrastructures for connecting renewable energy sources. Some of the typical issues related to mini grids are coordination of power, sustainability and metering. In scenarios such as the mini grid there is often a lack of centralised control

and monitoring due to its distributed nature. This presents problems in terms of fault finding, and allows for relaxed rules in terms of what electric supply equipment can be connected to the mini grid and the quality of the power it exports onto the mini grid.

The project detailed in this document is related to coordinating power between loads and multiple sources in a controlled and efficient way, and building a product that connects renewable energy sources to grids of any size.

The application for this project is discussed in detail nearer to the middle of this document. It is necessary however to inform the reader of the basics of the application so that the design can have some direction.

The hybrid power system project was started in 2005 by MLT Drives and Optimal Power Solutions (OPS). MLT Drives and OPS specialise in the design of power electronic equipment. The author is part of the team of several engineers working on this project.

The end result is a commercial product that connects several power sources together with batteries and a load in the most efficient and autonomous way. The product is able to interface with a power grid and monitor exported and imported power.

Figure 1 depicts the block diagram of the physical layout of the system:

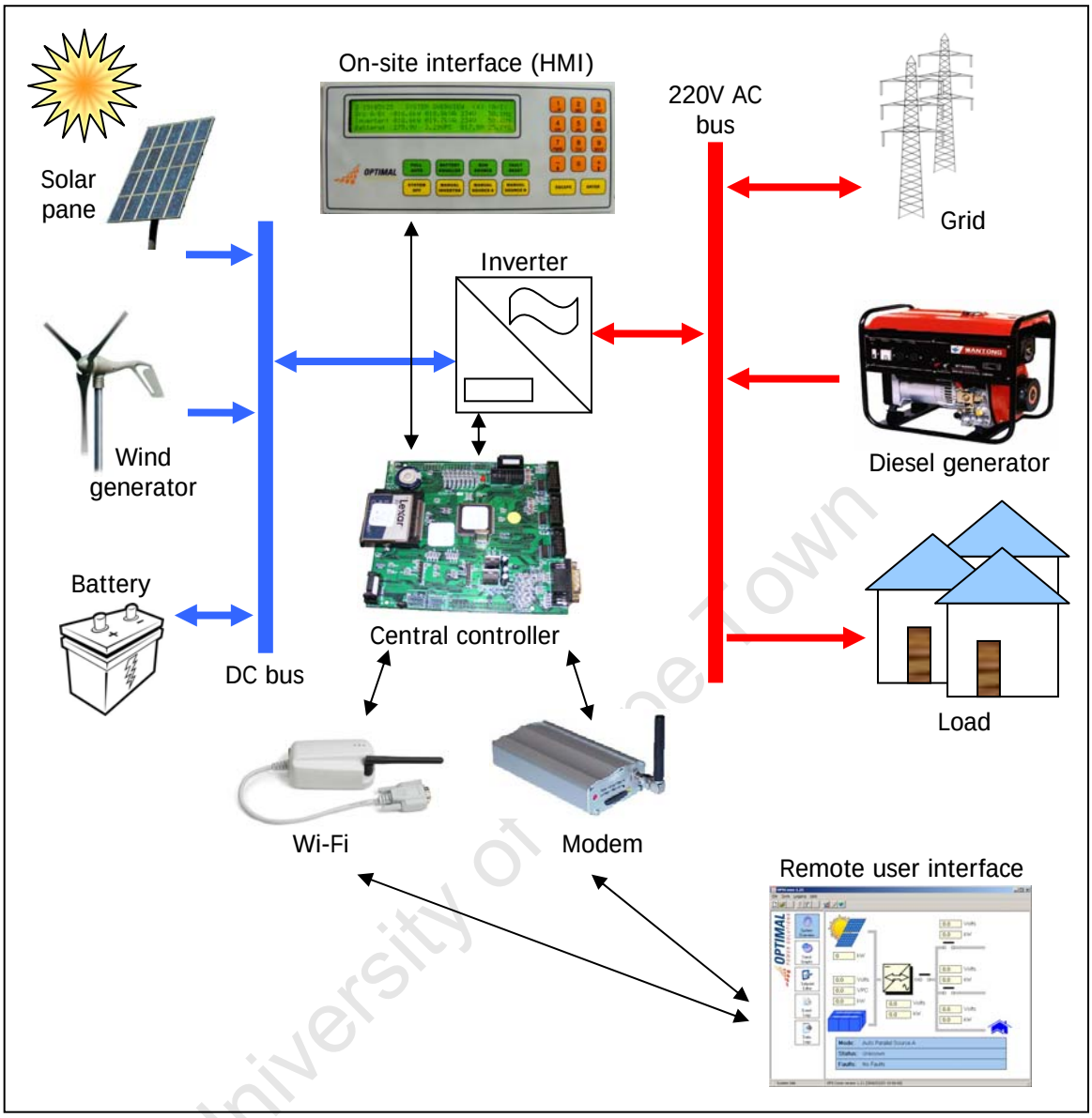


Figure 1 – Components of the hybrid power system

The various components shown here are all connected in terms of power and control.

The central controller manages the flow of power between all the power components. It can charge the batteries from any source, or multiple sources. It can drain the batteries at specific times, or during peak demands. The solar and wind power inputs can be used to charge the batteries. Various modes can be selected to give priority to specific sources of power. The central controller can provide uninterrupted power to the load by automatically disconnecting faulty components in the system.

The hybrid power system is scalable from small inverter sizes of 4kVA to larger inverters capable of delivering up to 300kVA.

An important requirement of this project is to provide an effective solution for managing power systems remotely, and detecting faults before they cause downtime. This is done primarily by designing a central controller printed circuit board that is capable of implementing many different communication technologies.

It is of utmost importance that the remote management system be effective and reliable as the product is targeted at customers who cannot afford to loose power. The central controller board must be capable of handling all local controls and include as many of the most up to date communications technologies. One such technology is being able remotely update software or firmware. When a system in out in the field, it is far more desirable and cheaper to fix a fault or update software remotely than sending an engineer or technician out to the site.

The controller hardware must be robust, compact and cost as little as possible so that it can be scaled over a wide range of system sizes.

2. Literature Review and Background

The object of this thesis is to produce a new, working printed circuit board (PCB) with new features and expandability. Although the author's focus is not on the actual generation and management of electrical power, some knowledge of power was required to adequately design the PCB that controls the power. Thus, an investigation into the current trends, deficiencies and short falls of modern power systems was done. An investigation into optimal design of robust, reliable and noise immune PCBs was of great importance to secure the success of this project.

2.1. Investigation into Power Systems

The focus of this investigation is on the current trends towards decentralisation of power, mini grid technology, and renewable energy sources and their implications.

Mini grids are becoming increasing popular in remote areas where the infrastructure of a national power grid is not present.[2] The technology for building mini grids is more readily available now, especially given the trends in the affordability renewable energy sources.

As stated in the introduction, Mauch, Ayoub and Jacquin[24] define a mini grid as being powered by a combination of photo voltaic (PV), wind, micro-hydro, fossil fuel gensets, and other sources. They typically supply multiple users and may be connected to the distribution grid of the local electric utility.

Strong emphasis is placed on metering and synchronisation. When mini grids increase in size or even join together, a need arises for some form of communication between different segments of the mini grid.

Figure 2 shows the layout of a simple mini grid. It can be assumed that adequate metering is in place to support the buying and selling of energy. The solar panel connection might be placed there to sell energy to users of the grid, i.e. load 1

and 2. The same applies to the wind generator. Another entity might use the bi-directional inverter and batteries to store energy when it is being sold at a low cost, then supply energy when the demand is high, to make a profit.

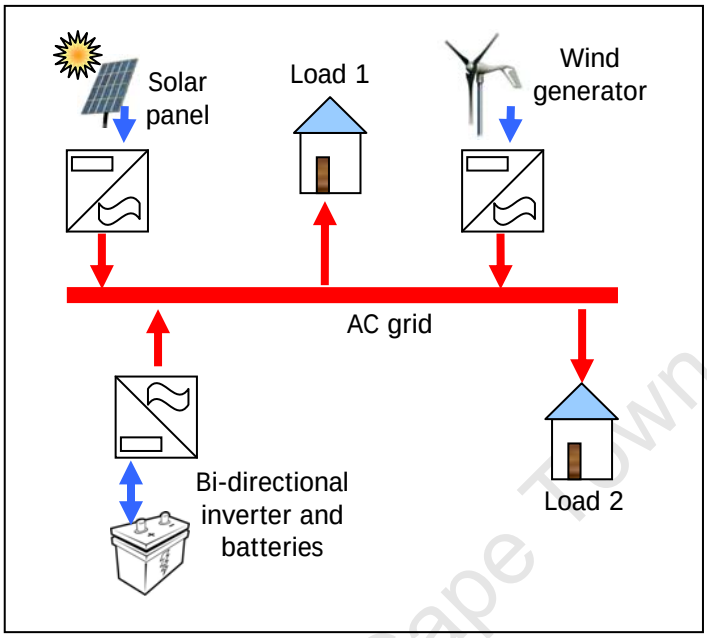


Figure 2 – A simple mini grid layout

Another place where mini grids are becoming popular is in new housing developments. Developers often have to provide "green"[1] power to build on certain sites that are in or near nature reserves and other protected land.

In other cases, expensive housing developments in areas where no nation grid exists require power and can not wait for the national utility to extend the grid to them.

Some existing problems with small power systems in mini grids are:

1. Most power systems do not have remote communications.
2. The local communications they do have are primitive.
3. They are extremely difficult to fault find, as no information is logged.

[1] "Green" is defined as power that is derived from a renewable source.

It is the intention of the author to demonstrate a successful execution of a design and the building of a power system that can interface with a multitude of different communication technologies.

Modern trends towards renewable energy are being initiated in the more developed countries. As the prices of photovoltaic (PV) (or solar) power decrease it is not far away from being cheaper than nation grid power in some countries.

One drawback with most renewable energy sources is that the supply of energy is not sustainable or not present at all times. PV power is only available in sunlight, whilst wind power is only available when there is wind. Even hydro-electric generators, which generate power from the flow of rivers or draining of dams driving turbines, may only provide power in rainy seasons. It can thus be concluded that a reservoir of energy is needed to store energy when it is available in order to provide a constant source of power. The most common such reservoir is an array of lead-acid batteries.

A simple setup for a typical PV power source with a battery is shown in Figure 3. There is an inverter connected to the battery to provide 220V AC power to a load.

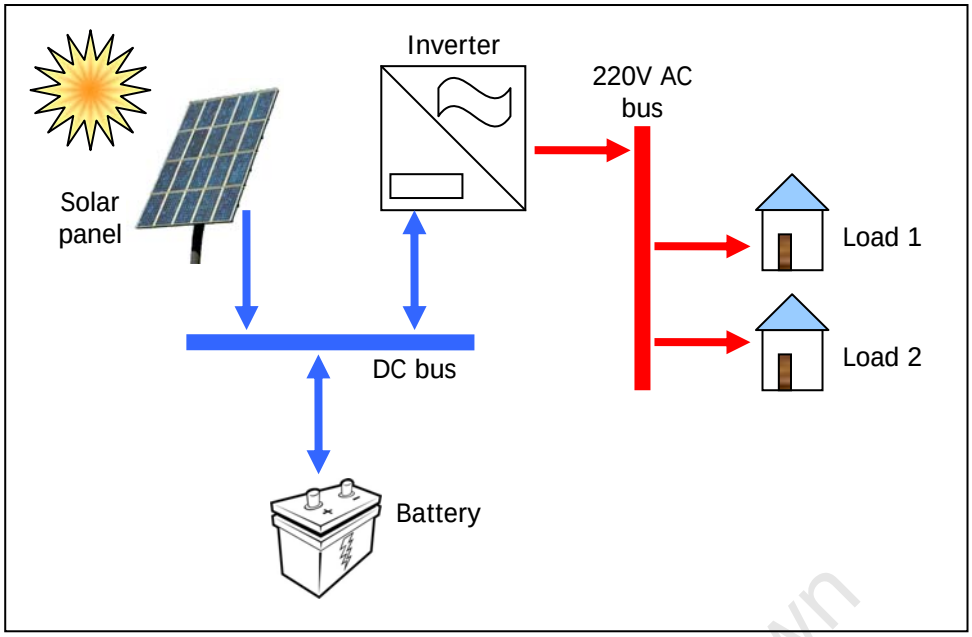


Figure 3 – A typical solar panel setup

A similar setup exists for wind generators.

Some of the first steps in improving efficiencies in such systems are:

More advanced solar systems would include a maximum power point tracking (MPPT) controller. This would draw current from the solar panel at the optimal point where the most power can be drawn. MPPT controllers usually provide an input line to disable disconnect the solar power from the DC bus. This can be done when the batteries are fully charged.

Charging batteries at optimal rates, and depleting them to certain minimum levels is another feature that is found on larger systems. The batteries then last longer and the ratio of energy imported to energy exported is closer to unity.

A bi-directional inverter can supply power as well as draw power, for example to charge batteries from the grid. The inverter can then discharge the batteries to decrease the power required from a power grid during peak demands.

These power components and concepts are all brought together to make up the hybrid power system described later.

2.2. PCB Design with Noise Immunity

After reviewing several papers and guides on printed circuit board design, the following techniques were noted for reducing noise emission and susceptibility:

2.2.1. Using Four-Layered Boards

Using four-layered boards as opposed to two is more expensive, but it can help reduce ground bounce, and supply sag, which are effectively voltage fluctuations throughout a board on the supply. Generally, the middle two layers are used for a ground and a supply plane. These two layers form capacitance, and thus help to decouple supply noise throughout the entire board.

2.2.2. Routing Techniques

When routing tracks on a board, the tracks must be wide enough to carry the current that is needed. This applies especially to ground and supply tracks as it is undesirable to have resistance in the supply path.

For tracks carrying digital or high frequency signals, care must be taken to never use a right-angle corner as high-frequency noise will be emitted at this point. Always use 45° angled corners.

Tracks must always be as short as possible, and at high frequencies, long tracks become inductive.

Plated drill holes that conduct from one layer to another (vias) are also inductive, especially the smaller ones. Always use large vias when carrying high currents, or alternatively use multiple smaller vias, or microvias.

In general routing should be done on the outside two layers in a four-layered board. Some manufactures allow for the use of buried vias. These are vias that are drilled between any numbers of layers, and not necessarily all layers. These are difficult to work with in design, but can improve noise issues. It was found that usually buried vias are used in boards with six or more layers.

2.2.3. Planes

The use of ground (and supply) planes is essential for managing noise. Ground tracks and planes in a circuit form a barrier for noise. Paths of currents in ground planes vary according to frequency. The return paths of high-frequency currents will take the path of least inductance, while the return path of low-frequency currents will take the path of least resistance.[16] Care must then be taken to ensure that ground planes have clear straight paths for currents.

It is usually advisable to separate digital and analogue devices and have separate planes for these components.[16] [1] The planes can be joined at points where supplies come onto the board or where large capacitance is placed across the supply. The same goes for high-speed and low-speed devices which are discussed in the next section.

2.2.4. Component Placement

As with analogue and digital planes being separated, the components themselves must also be separated physically. It is best to divide the components into group using the high-speed and low-speed classifications[5], and then dividing further into digital and analogue devices. This will keep high-speed signals with transient or switching noise from corrupting slow analogue signals.

Also, electro-magnetic noise from inductors in a switch mode power supply for example can induce noise in sensitive circuits. These, too must be placed carefully.

2.2.5. Supply Noise

It is important to ensure that the supply voltages are clean of noise at all frequencies.[5] A star-type formation can be used for the supply and ground planes or tracks that branch off for each group of components.[16] All the tracks or planes can come together at a single node where a large capacitance must be placed. This will ensure that return paths of current for noisy devices will not come into contact with- or induce current in any other devices. It is advisable to use ferrite beads in series with the positive supply tracks for analogue and digital devices to suppress high-frequency noise.[1] Capacitors must be placed after these beads to provide a reservoir for high current demands.

Each device like an operational amplifier or microcontroller should have its supply decoupled as closely to the device as possible.

When connecting cables to devices, ensure that enough ground wires are provided to carry the return current, otherwise a voltage will develop across the ground wire, which could raise or lower the ground reference on a board, or even create ground loops.[16]

When routing supply tracks, it is advisable to keep them close to one another in parallel, as noise on the lines will be opposite in polarity (if they are perfectly balanced in current) and therefore cancel out which results in no emitted noise.[16]

2.2.6. Suppressing Noise in Signals

Some manufacturers advise to put small resistors (less than 100Ω) in series with high-speed lines to provide a small amount of inductance. [16]

This is good for reducing switching noise. It is better to put the series resistor as close to the driving source as possible, as it eliminates resonance effects which emit noise.[1]

For physically long tracks or even cables carrying signals, it is better to push more current through the signal than less. Noise is induced in the form of current, so the more current the signal carries, the higher the signal-to-noise ratio. Therefore signals should generally be terminated at the receiving side with a resistor that draws enough current.[1]

Ensure that all components chosen are rated as low-noise devices.[4] [7]

For differential pair signals, such as speaker outputs or microphone inputs, the two tracks must be routed close together and in parallel so as to cancel out the noise.[16] This also forms a small amount of capacitance. The same applies to cables. If possible, a twisted pair should be used for a differential pair of signals.

3. Terms of Reference

This document describes the migration from an existing hybrid power system to a more advanced system with many communications options. Specifically, it describes the main controller boards that control the power system.

The existing system has been developed and is running successfully in the field. Despite its success, it has several limitations which prevent further additions, modifications and features.

The existing system can be described as a printed circuit board with a Texas Instruments digital signal processor (DSP). The DSP has several peripherals which are routed to connectors. The DSP board plugs into an interface board which conditions signals and buffers digital and analogue inputs and outputs. These boards are shown in Figure 4. A third component is a power supply, and lastly there is a separate human-machine interface (HMI) board, which provides visual displays of information, and a keypad for modifying settings. Further details are discussed in the section on existing functionality and setup.

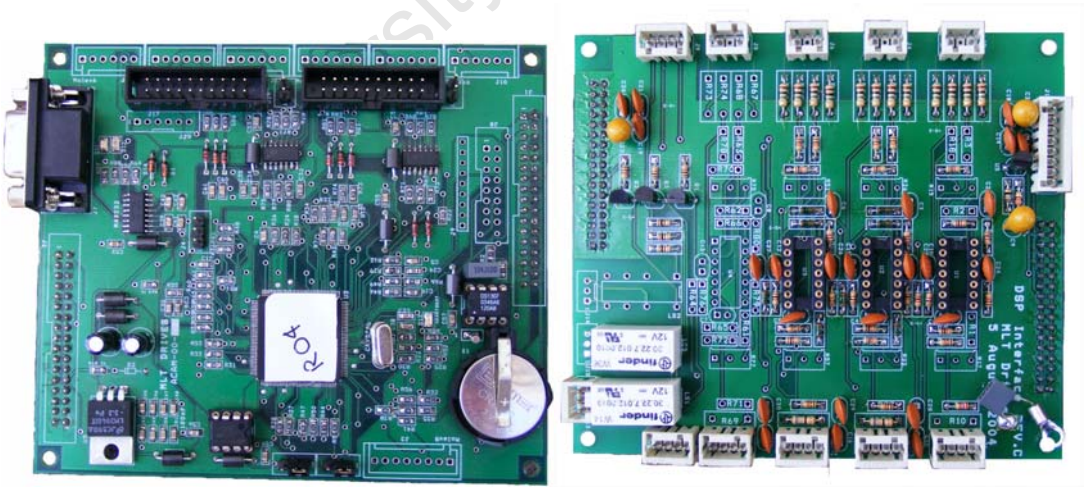


Figure 4 - Photos of the existing DSP board (left) and its interface board (right)

A new system needs to be developed that will allow for far more features to be added. This new system needs to be fairly compatible with the existing system. The strongest emphasis on compatibility is in the area of physical size, layout, and connectors. To

some degree a preference was given to maintain a Texas Instruments series DSP as the core processor.

A fair amount of freedom was given in terms of which features to add, as long as they achieve some main goals:

- The new design must be generic so as to allow for many future additions and modifications.
- The new system must support remote access.
- The new design must be optimised for production in large quantities.
- The physical printed circuit boards must be far more noise immune than their predecessors.

The goals of this thesis are therefore to deliver on the objectives set out in the terms of reference, and to produce a suitable solution for the next generation hybrid system controller.

4. Design Overview

In reality, two systems were designed, but only the second system is discussed in detail here. The first was an intermediate step in the migration and is a watered down version of the second.

As this document shows the migration from an existing system to a more advanced system, the existing system's functionality and shortcomings will be discussed first. Next, additional functionality and/or improvements that are required for the new system will be listed.

4.1. Existing System before Author's Involvement

The existing system is powered by a digital signal processor (DSP) board together with an interface board (see Figure 4). The interface board is for signal conditioning, input and output buffering, and incorporates three relays.

The processor, the TMS320LF2407A from Texas Instruments, is the only processor on the board. *It handles the following:*

- Sampling the 10 analogue channels on the board
- Some primitive communications with a human interface board
- Digital inputs and outputs
- SPI communications with a digital to analogue converter (DAC) for debugging only
- Three complimentary PWM outputs

Some features of the DSP processor, namely the TMS320LF2407A are listed here according to Texas Instruments [23]*:*

- 2.5k 16-bit words of RAM
- Maximum clock speed of 40MHz
- 40 MIPS
- 32k 16-bit words of FLASH
- 144-pin PGE quad flat package (QFP)

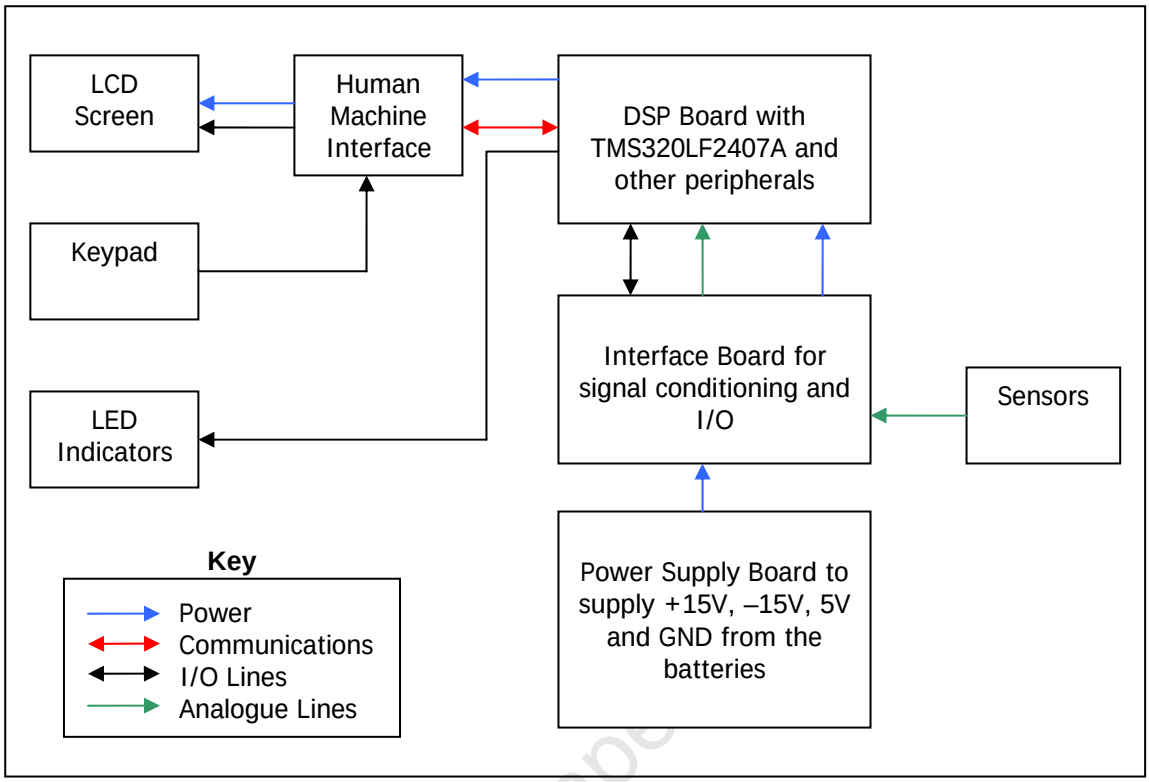


Figure 5 - Existing system

The surrounding hardware required to drive the power equipment is not shown here. It is assumed that there is an inverter, batteries, a load, and an AC voltage source surrounding this system.

The software on the DSP exists already and does an adequate job of controlling the inverter, batteries and source, but it requires human intervention when anything goes wrong. This system has no remote accessibility. When something does go wrong, someone physically has to press a reset button on the keypad.

There is no mass storage component in this system, so no real logging can be done. When something has gone wrong, one must interrogate the human machine interface (HMI). Often, a one-line error message is not enough to properly find the fault that occurred.

The system has access to a real-time clock (RTC) on the HMI board, but if the communications between the DSP board and HMI board fail, then the DSP

looses track of the time, which causes battery equalisation and other time-based features to fail.

As a result of the small LCD screen and, a very simple human interface exists. There is far more configurability that is necessary in running large systems.

There is no interface to connect to a PC to view real-time information or faults.

The PWM outputs on this DSP board are limited to 3 complementary pairs, which places limits on the choice of switching schemes for three-phase systems, because one cannot run three-phase systems as three single-phase inverters. One must use state-space PWM instead, which is not always optimal.

The analogue inputs on this DSP board are limited to 10. This board with the interface board can read 4 voltages, 4 currents and 2 temperatures. The DSP chip has in fact another 6 analogue inputs which are not routed to the interface board. This is unfortunate as there are special applications where one needs more than the above-mentioned range of analogue inputs.

The DSP board is a 2-layer printed circuit board. It is evident that much time has been spent making the board immune to noise. Beaded inductors are strategically placed about the board. Unfortunately, in field tests, the DSP still does freeze or reset occasionally when in particularly noisy environments.

A good feature of this board is that the spare input/output pins of the DSP are routed to plugs which make custom projects that have additional requirements more readily possible.

Optically-isolated PWM outputs can optionally be used and have been proven successful in noisy environments in past projects.

In conclusion, this existing system is adequate, and it provides a decent base for custom projects. The disadvantage is that it is not totally up to date with the new technology available now. It is restrictive in processing power, I/O range, and

memory. It is therefore necessary to design a new base system that has the capacity and processing power to embrace new technology.

4.2. Required Functionality

The improvements or additional functionality for the new system are discussed here.

In terms of the application, the hybrid system must be a generic, configurable, robust power system. It should be able to manage two grids, wind generators, solar panels, and an inverter with battery management.

The power electronics company, MLT Drives (see http://www.mltdrives.com/ for details), builds these hybrid power systems commercially. Parts of the terms of reference are to improve the existing system and make it more commercially viable.

The hybrid system has to be extremely robust to improve viability. This requirement is most important in diagnosing problems and allocating responsibilities for malfunctions. This requirement necessitates the need for remote monitoring and control. In order to be reliable, a system must be remotely monitored. It is in both the buyer's and the seller's best interest to keep the system running at all times.

To remotely monitor effectively, one needs the most up to date means of communications available. This can be achieved on site with direct interface locally, or nearby the system. Lastly the system must be accessible from anywhere in the world. With all these forms of communications, the system should then be able to be more effective in warning management of any possible failures, or situations that need attention.

For all the above reasons, it was found that a more powerful DSP was required.

From a power electronics point of view, much better control of three-phase power can be done if the inverter is controlled as three independent single-phase inverters. With such control, one can then alter the phase, voltage or current of any of the phases independently, hence achieving better balanced voltages, currents and a better power factor. This therefore requires more processing power and 6 independent PWM outputs, (as opposed to 3 pairs of complementary outputs.) Having 6 independent PWM outputs as previously mentioned allows for more choice in switching techniques.

Some form of remote access is essential to remotely fix, debug, monitor or control systems in the field. Furthermore, remotely uploading software in a fail-safe system would be a useful feature.

For this, several common options exist:

1. RF radio communications
2. Cellular network
3. Public switching telephone network

RF radio communications were decided to be too unreliable, especially for large distances.

The last two options can be incorporated into one, if the system has a modem. This can be a standard land-line modem or a GSM modem. Communications can then be established across long distances, even across countries. GSM coverage is improving greatly at present, and is now present in most remote places.

With modem communications, the high cost of transferring data is outweighed by the coverage of telephone and cellular networks. The costs of these communications, although high by some standards, are minor in comparison to down time in a power system and in addition, text messages can be sent as alerts for system faults with a GSM modem.

For this, a PC interface is needed. Fortunately, most laptops (and now some mobile phones) have a Wi-Fi card built in. The standard physical interface for equipment is normally RS-232 or now more commonly, USB. Laptops are more likely to be used for debugging in the field. For this reason, a wireless interface is sufficient. Bluetooth is commonly used. It has been found that Bluetooth, for the purposes of this project, can be unreliable and its performance compares unfavourably to that of Wi-Fi. Wi-Fi was chosen over Bluetooth as a wireless interface. It was decided to retain an optional RS-232 interface.

In terms of protocol and software, there are several control systems available on the market for controlling and monitoring systems like this one. It was decided to implement an industry-standard protocol, namely MODBUS as it is widely-used in SCADA systems, and is very common in control engineering.[10] This then gives the system the ability to be controlled and monitored by standard SCADA software or custom software that implements the same MODBUS protocol.

It is also required that the system be able to poll certain devices like amp-hour meters in order to extract and log additional information. MODBUS was specified to interface with the system by some of MLT Drives' customers. Hence MODBUS was also selected to be used for the system to interrogate slave devices.

The ability to log voltages, currents, and other data is necessary. Events or faults that occur need to be logged in sequence with timestamps. It is also necessary to log all user commands and inputs. That is to say that whenever anyone changes a setting or issues a command to the system, it is logged with a timestamp. Using this information, failures can be traced better in the case of human error.

A more interactive human machine interface (HMI) was needed locally on the system. A larger screen was also necessary to display more data without being confusing. The keypad need to incorporate several buttons to execute common operations as well as a number pad for entering values into the system. A custom keypad was therefore specified as a requirement. This section of the

design was not done by the author and was handled by other members of the project team.

Security is required for accessing the system remotely and locally. Pin codes are adequate. Some configurability needs to be hidden from the end-user, such as calibration values for the analogue inputs.

Noisy immunity is a very important requirement for a system controlling large amounts of electrical power. The DSP cannot afford to freeze or not resolve from a loop. For this reason, the DSP board must be designed in such a way as to be fail-safe in this respect. If the DSP fails in some way, there should be surrounding “safety nets” for the critical outputs controlled by hardware. Most importantly the printed circuit board must be designed in the most noise immune way. It was decided to use ground and power planes where possible on all layers. Four layers were used rather than two for improved noise immunity. This allows the board to route power tracks on the inner two layers using wide tracks which would otherwise take up much needed space on the outer two layers.

A further requirement of the system was to be able to interface with a solar controller or maximum power point tracking (MPPT) controller. These MPPT controllers are widely used in photovoltaic (PV) systems and usually provide some control lines which can be driven by the hybrid system. Provision was made for the possibility of incorporating the MPPT controller into the software on the system.

Future additions need to be possible with the new system. Thus all inputs and outputs and especially analogue inputs needed to be routed to some form of header or connectors. This allows flexibility for custom projects where more hardware may be needed to interface with.

Due to the nature of business relationships with manufacturers, suppliers and customers, it is necessary to protect the intellectual property in some way. The protection should be enough to prevent third parties who have access to the

software from copying the hardware, programming the software and reselling the product illegally. Some form of password and/or licensing needed to be in place to activate each system.

4.3. Scope of Design

This section is focused on the design of the controlling and monitoring software, hardware, and the communications. The algorithms and hardware for controlling and managing the actual power system and its details are of secondary importance to the author and have been handled by other individuals.

4.4. Hardware Design

The following sections detail the design of all the aspects of the hardware of the central printed circuit board. An interface board similar to that of the existing system was designed in parallel by other members of the project team, to meet the requirements and expectations of the new system.

4.4.1. Central Digital Signal Processor

The micro processor used before was the TMS320LF2407A. Several options for a new micro processor were considered.

Firstly the preferred brands available were Texas Instruments (TI) and Microchip. The performance of Microchip's digital signal processors was found to be significantly lower than that of TI, and it was thus decided to concentrate on TI processors.

In the end, two possible candidates were found to meet the requirements. One had more memory, and was faster, but lacked an analogue to digital converter, which meant an external chip would have to be connected to it. The other, the TMS320F28xx series had less memory and power, but was more than adequate. More importantly, it had and on-chip analogue to digital converter capable of 80ns conversions at full speed. It was decided

to use the latter as the performance was an order of magnitude ahead of the previous processor and the analogue to digital converter was very fast and of high precision.

Table 1 highlights the differences between the existing system's microprocessor and the new one:

	TMS320LF2407A (existing)	**TMS320F2812 (new)**
Clock speed	40MHz	150MHz
Instruction cycle	25ns	6.67ns
FLASH program memory	32k × 16b	128k × 16b
RAM	2.5k × 16b	18k × 16b
UART ports	1	2
SPI ports	1	1
Analogue to digital ports	16 (10-bit) (375ns)	16 (12-bit) (80ns)
I/O pins	40	56
External memory interface	1	1
PWM outputs	12	16
Supply voltages	3.3V	1.8V and 3.3V
Package	144-pin surface mount quad flat pack	176-pin surface mount quad flat pack

Table 1 - Comparison of DSPs

As can be seen from the table, the processing power is greatly increased, and the sampling period for the analogue to digital converters is reduced by more than a factor of 4. Both these improvements allow for faster and closer monitoring of the voltages and currents in order to control them more tightly.

The additional program space allows more functionality to be included in the system.

4.4.2. Printed Circuit Board Design

The shape and size of the main printed circuit board (PCB) was partially influenced by the existing PCB. Due to the greater number of components and connectors on the new PCB, the size of the board was extended in width.

The placement of the components on the PCB was based on keeping logical and functional groups of components, connector placement, and noise issues.

The placement of the main DSP was in the centre as it requires the most connections to peripherals. The rest of the component groups were placed around the DSP. In general, the component groups associated with connectors were placed near those connectors.

According to the specifications in the requirements, a four-layer board was used to alleviate noise issues.

All the techniques described in the literature review on designing PCBs were used in this design.

More detail on the final result of the PCB will follow in the results section.

4.4.3. Compact Flash Interface

The onboard Compact Flash (CF) card will be used by the logging system, and software upgrading. In the software design section, reasons are given to justify keep the CF card onboard, i.e. not making it removable.

Physically CF cards have a 50-pin interface which conforms to the True IDE standard.[12] [13] Some of these pins' functions are detailed below.

D0..D15

These pins form the 16-bit bidirectional data bus. All 16 are connected to the DSP.

A0..A10

These pins are the address bus. Only the first 3 are used because the True IDE interface is used. The others must be tied to GND according to the specification.[12] [13]

CS0..CS1

These pins are function select pins and are effectively used with A0..A2 to address registers.

RD and WR

These are the read and write strobe lines. The DSP drives these to do a read or write.

RESET

This is a normal reset pin which the DSP drives when any actions fail on the CF card.

The rest of the pins are not used in True IDE mode, or don't need explanations.

The schematic in Figure 6 shows the connection of the CF card:

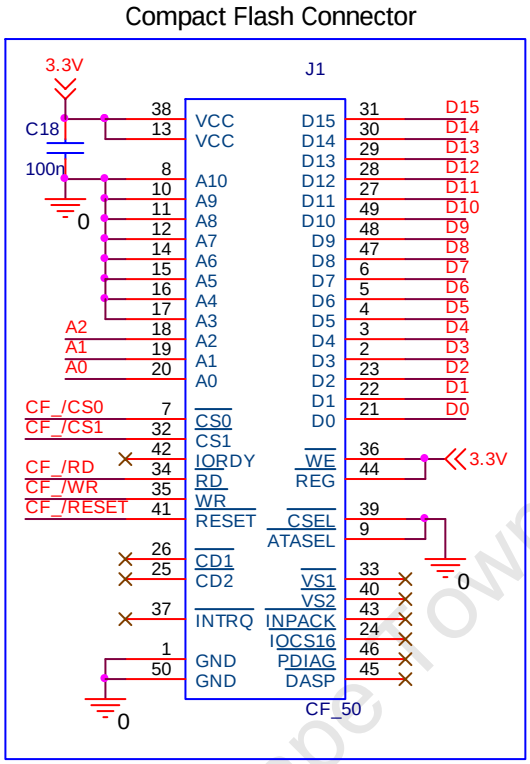


Figure 6 - CF card connector schematics

The DSP provides a dedicated external memory interface (EMIF) for connections like this one to the CF card.[20] [21] It provides a 19-bit address bus and 16-bit data bus, with an assortment of strobe and select lines. Several address zones exist which drive different chip select lines. Care was taken to select an appropriate address range to connect the CF card to. The new DSP's EMIF differs vastly to that of the old TMS320LF2407A which had no zones, and only a write/not read line with an enable strobe.[19] With the old DSP (on the intermediate design), an address decoder chip had to be used in between the DSP and CF card. This chip has become redundant in the new system.

4.4.4. Inter-Processor Bus

The inter-processor bus forms the core of the communications between all the 5 processors in the system. It is important that it be fast and robust.

The algorithms for the communications are discussed in far more detail in the software design section on this bus.

For the hardware, the requirements are that the signals be buffered and multiplexed from the slaves.

Figure 7 shows the way the UART bus is connected in terms of data flow:

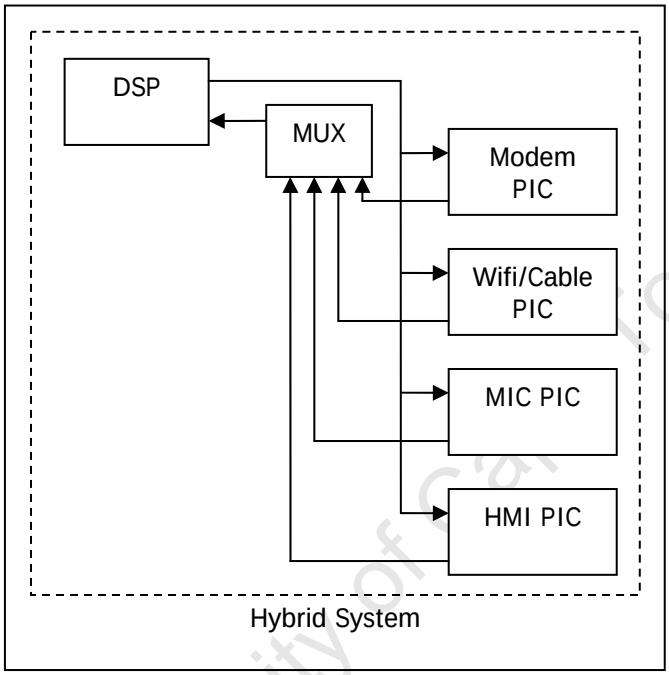


Figure 7 - Data flow of the UART bus

When the DSP sends data, it is sent to all the slaves. When the slaves send data, their data is multiplexed and sent to the DSP. There are some basic rules for this bus setup:

1. When the DSP sends data, all the slaves receive it.
2. The slaves must only send data one at a time.
3. It is up to the DSP to coordinate the slaves' sending of data.

The schematic in Figure 8 *shows how the UART multiplexer works:*

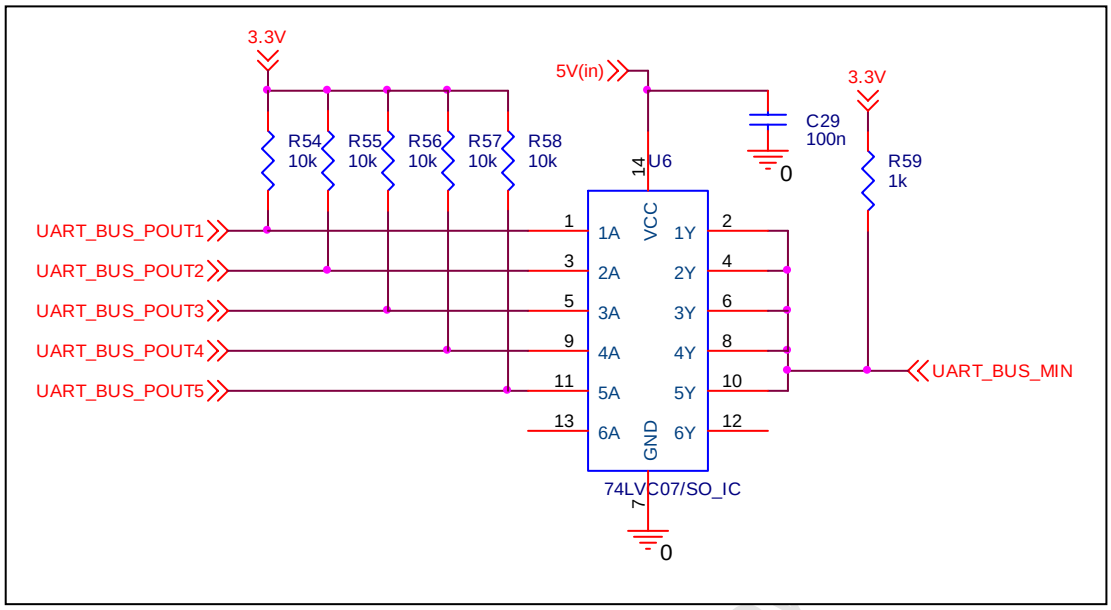


Figure 8 - Schematic of UART multiplexer

All the inputs on the left are the four slaves' output (TX) and one spare for future use. Notice that if any of these inputs is not connected, or floating, the 10k pull-up resistors ensure that they stay in an idle state. The buffer chip has open collector outputs. All the outputs are tied together so that if any one input goes to a low state, the output to the DSP goes low. Thus for this reason, care must be taken to ensure that the slave devices do not attempt to send data simultaneously.

Thanks to the open collector outputs, the DSP's output (TX) can also be buffered and using a pull-up resistor to 5V rather than 3.3V, the logic levels can be raised for a 5V interface to the bus. This is used for the HMI interface which operates at 5V. The HMI's TX line is also buffered to 3.3V thanks to the input range of the buffer chip being 5V compatible.

A spare interface to the bus is provided so that it may be plugged into a PC and monitored for debugging. It may also be used to interface with an addition processor which might be added to the system in the future.

4.4.5. Wi-Fi Interface

Wi-Fi serial bride modules are available and it was decided to use the Airborne Direct Serial to Wireless LAN module from DPAC Technologies, particularly the ABDE-SE-DP series.[15]

Figure 9 - Photo of the DPAC Wireless LAN module

It provides a high level interface that bridges RS-232 to TCP/IP over Wi-Fi. It listens on a configurable port for an incoming TCP/IP connection over the network. When one comes in, it connects it. The data sent from TCP/IP is sent out on the RS-232 port at a configurable baud rate, and data received on the RS-232 port is sent back to the TCP/IP connection. In this way, the design of the TCP/IP stack is effectively abstracted. This relieves the system of the processing of raw TCP/IP data, but is still versatile enough by way of custom commands to configure most aspects of Wi-Fi and RS-232.

The module also has a special design software suite that allows designers to serve web pages on a local wireless network. Although these features will not be used on this design, it means that the module will allow for future developments that give direct access to the hybrid system via a local web page. This could show graphs and logs.

The following pins of the module's DB-9 connector are used:
2 – RX
3 – TX
5 – GND

Flow control on this communication medium is not necessary as the buffers on the module are sufficiently larger than the largest packet that needs to be sent.

4.4.6. Modem Interface

The modem interface needs to communicate with both PSTN modems and GSM/GPRS modems. Most modems use a standard 9-pin RS-232 interface. The hybrid system design allows for UART 3V transmit (TX) and receive (RX) pins to be brought to a standard ribbon cable connector. The TX and RX pins are then buffered to RS-232 levels using an isolated RS-232 board. Unfortunately the isolated RS-232 board allows for only the 2 most widely used pins, TX and RX, to be connected. Any modem interface needs more than that.

Due to the communication networks being generally slower than the interfaces to the devices that send and receive, it is necessary to use flow control. That is to say, for example, a modem interface might be at 115200 baud, but the connection to the communication network might be 44000 baud. Therefore, if a host sends data at 115200 baud, the device in the middle needs to buffer data until it can be sent on the slower network. The buffer can only be a finite size, and therefore at some point, the device needs to indicate to the host to stop sending. The two RS-232 standard pins for this are called *Ready to Send* (RTS) and *Clear to Send* (CTS).

An addition pin called *Data Terminal Ready* (DTR) is also needed. The name of this pin has some history attached to it, but the basic functionality of it allows for switching between online and command modes on modems. It determines where data from the host must be directed, either to the communication channel, or to be interpreted as a modem command.

It is therefore necessary to have a 6-pin RS-232 implementation. A standard non-isolated RS-232 level shifter (MAX232 chip) is needs to be used on the main DSP board as follows:

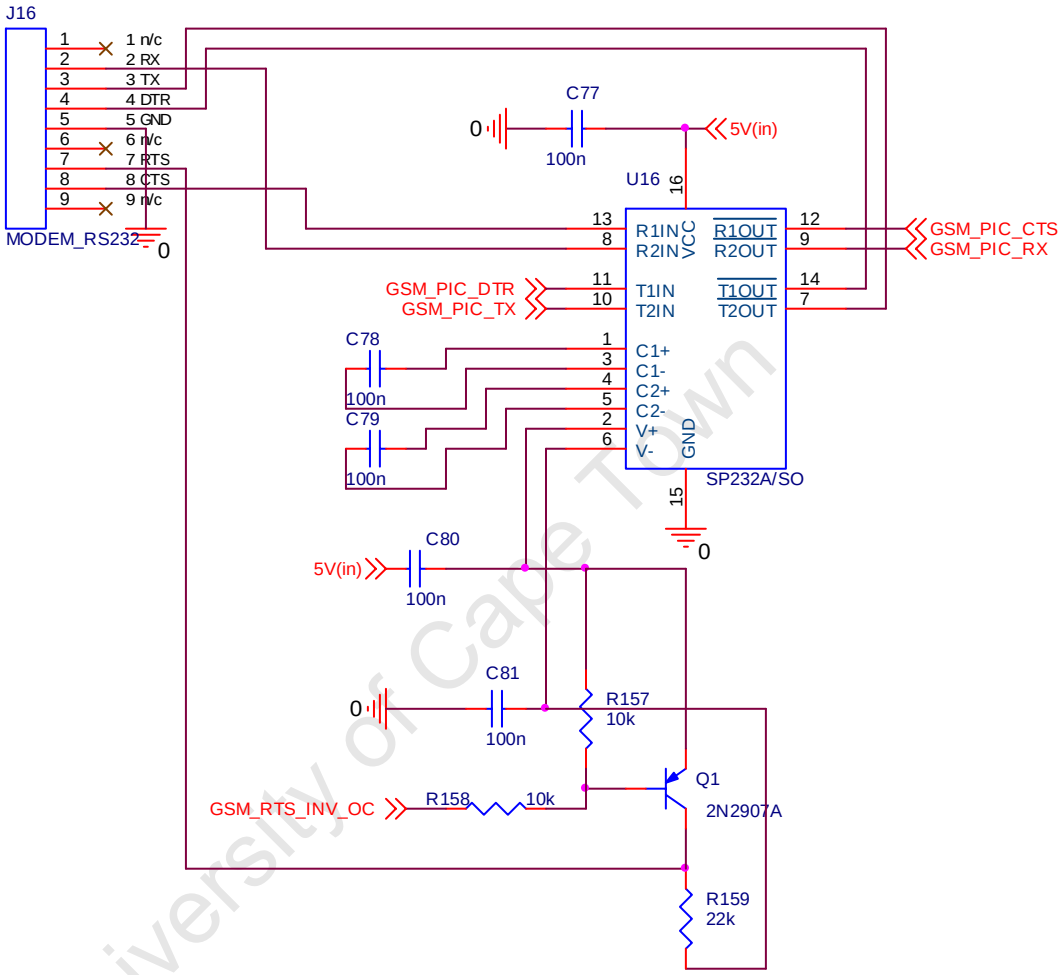


Figure 10 - Schematic of the RS232 level shifter

The PSTN modems are very standard in terms of implementation of AT commands and RS-232 interfaces. Therefore, no specific requirements are placed on exactly which PSTN modem brands are to be used.

GSM/GPRS modems are slightly more specific. Unfortunately, the implementation of GPRS particularly is different across GSM/GPRS modem manufacturers.[9] It was therefore decided to specify one of the most well-known and widely used brands, Wavecom, as the only supported brand to keep software design simple.

The cable that needs to be used between the modem and the DSP board needs to be a full RS-232 9-pin cable, rather than a low-cost 3-pin cable (which is usually RX, TX and GND).

4.4.7. PWM Outputs

The DSP chosen for the design supports 6 pairs of complementary PWM lines as well as 2 pairs of auxiliary PWM line for general use. It was decided to bring all of these PWM lines to connecters. The 6 pairs are broken up into 3 pairs per connecter. This way, either 2 state-space three-phase inverters can be switched or if special switching schemes are needed, the 6 PWM lines from both connecters can be used, ignoring each line's compliment.

The auxiliary PWM lines have no function at present, but a 10-pin connector will be assigned to these for future use.

Optionally, pull-up or pull-down resistors can be used by soldering links on the board. This is desired for different interfaces to switching hardware.

4.4.8. Analogue Inputs

The DSP has 16 analogue inputs which must all be 0 to 3V in range. As required, all of these inputs will be connected and used. Care is taken in keeping the signals in the correct range to prevent damage to the DSP.[8][17]

The circuit for each analogue input is as follows:

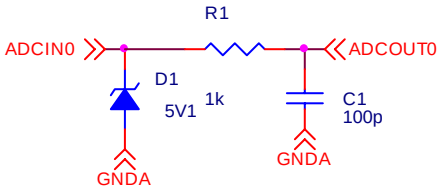


Figure 11 - Schematic of the analogue input filter

The signal comes from the left side of the circuit and is immediately limited by the zener diode to the range -0.6V to about 4V. The zener diodes start conducting long before their nominal voltage, so one must rate them about 50% high than the voltage limit. The resistor and capacitor have two functions. Firstly they form a low-pass filter which is selected to have a time constant approximately twice that of the sample rate for the analogue to digital converter.

The formula used is as follows:

$$f_{cutoff} = \frac{2\pi}{R \cdot C} = \frac{2\pi}{1\times10^{3}\Omega \cdot 100\times10^{-12}F} = 1.59MHz$$

Secondly the resistor limits the current flowing into the analogue pin of the DSP which helps to prevent damage when voltages are out of specification. The capacitor forms a reservoir of voltage for when the analogue to digital converter suddenly draws current to do the sample, thus avoiding distorting the signal.

The 12-bit sampler results in 4096 different possible values for a sample, and gives 244µV/V resolution.[11] That means that in this 3V range, the

resolution is 244µV/V × 3V = 732µV. Running at 80ns, the sampler results in 12.5 million samples per second. That means that it can sample all 16-channels at a rate of 781.25 kHz. This is far more than adequate for the new system.

4.4.9. SPI Interface

An SPI interface was incorporated into the design for two reasons:

1. Backward compatibility
2. Communications for the real-time clock and onboard EEPROM chips

The previous system also had an SPI bus which was used for communications with a digital to analogue converter primarily used for debug. In the new system there are two chips which need SPI to communicate with the DSP. The intermediate design, which is not discussed much here, used SPI as the inter-processor bus. For backward compatibility with the intermediate design only, the SPI bus can also be used, but it is not recommended.

The SPI interface must have three select lines which drive the chip select lines of the various chips to be communicated with. The SPI bus must also come out to a connector for future additions. The SPI bus will use 3.3V logic signals. Because of this, care must be taken to ensure that the onboard SPI chips can operate at 3.3V instead of the nominal 5V.

4.4.10. Onboard EEPROM or FLASH

An EEPROM or FLASH chip, 25AA256 (SO8)[22], was included in the design for use in smaller hybrid systems which do not require a compact flash card. The EEPROM or FLASH chip can be used to store smaller amounts of logs and settings. This is not used in the demonstration of the hybrid system.

4.4.11. Real-Time Clock

The DS1306 from Dallas will be used to keep track of time. It proved difficult to find a 3.3V compatible real-time clock (RTC) that used SPI rather than I^2C. The choice was narrowed down purely due to availability. The RTC uses a 32kHz crystal and either a super capacitor or a lithium-ion battery. Provision was made for either to be soldered onto the board. In practise, later it was found that the super capacitor was a better solution, as the battery's voltages were sometimes too high and caused issues with using a 3.3V supply voltage instead of 5V. A 220,000µF super capacitor provided 5 days of time retention.

4.4.12. Debug Port

The DSP has two UART ports. One of which is used for programming and communication with the inter-processor bus. The other is not used in the system, so it must become a debugging interface. It is important to have a console-type debug with a large scale project like this. The author is also not the only programmer to be using this system, so care must be taken that sufficient debug is available that other persons will use the design in the future.

As will be explained in the software design section, a query-based debug console will be designed, as well as an unsolicited message output system. Various levels of debug must exist which can be enabled or disabled dynamically.

In terms of hardware, the debug UART comes to a connector with the inter-processor bus UART which then connects via ribbon cable a UART to RS232 converter board.

4.4.13. JTAG

The JTAG (IEEE Standard 1149.1[1]) connector on the board is necessary for in-circuit debugging and emulation, and can be used for faster programming of the DSP. The Texas Instruments compatible JTAG cable is expensive, so this must be viewed as an alternative to the serial programming interface with debug UART. Having said that, if a JTAG is available, debugging methods like stepping through code, and dynamic variable and register interrogation exist, which can be vastly more useful than a console-type debug.

Figure 12 depicts the schematic for the JTAG connection:

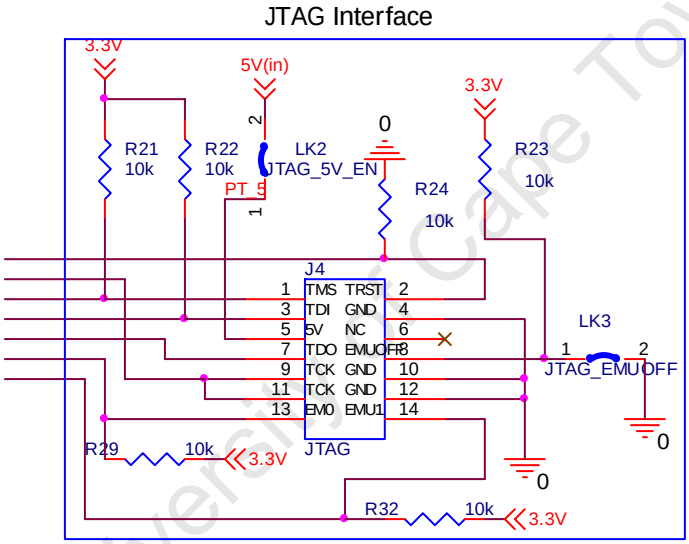


Figure 12 - Schematic of the JTAG interface

To be compatible with different voltages, the 10kΩ pull-up resistors are present. The JTAG and DSP both use an open-collector output system. Depending on JTAG interfaces, the links LK3 and LK2 (see Figure 12) can be soldered for different configurations.

4.4.14. Considerations for Remote Programming

Initially it was planned to have one of the onboard PIC microcontrollers program the DSP using the inter-processor bus path. It was necessary then to connect the inter-processor bus to the UART of the DSP which it uses to program. In order to put the DSP in programming mode, it must be reset and the programming line must be held high while it starts up. The design was such that any device on the board can drive the reset line. If done so, all processors in the system are reset with the global reset line. Provision was made for each PIC to hold its reset line high briefly to inhibit its own reset when resetting the DSP.

All these provisions were made, but then in practise, it was decided to write a custom bootloader for the DSP which programmed itself from the data stored on the compact flash (CF) card. For this method, the DSP does not need to be put into programming mode, and it can reset itself so these provisions are partially redundant now.

4.4.15. Production

When designing the hardware, specifically the printed circuit board, one must consider what measures need be in place or what compromises must be made to ensure that the mass production of the hardware is as simple, fast and inexpensive as possible. One must also make sure that there is as little room for error as possible.

From reviewing some literature, the following tips were established:

- Avoid using odd capacitor and resistor values that are multiples of 1000s of each other in different places of the board. For example, do not use 22pF capacitors in one section, then 22nF in another. Statistics show that the pico and nano versions get mixed more frequently than if the numbers were different. One solution to this (if the tolerance of the circuit allows) is to use 15pF and 22nF instead.

- Use as many surface mount components as possible. Through-hole components need to be hand-placed and sometimes hand-soldered. The more manual work the board needs, the more room for error. Most surface mount components can be automatically placed, and wave-soldered by machine. Only where surface mounts are much more expensive or heat dissipation is a problem, use through-hole components.
- Do not place heat-radiating components (such as voltage regulators) near heat-sensitive components, or whose reliability will be affected by abnormally warm environments. For example, if an electrolytic capacitor is placed near a regulator (which are both common components of a power supply circuit), the capacitors expected life time will be dramatically reduced if the regulator keeps it hotter than the normal environment.
- Overrate all components. Do not operate components near their maximum operating conditions in the datasheet.
- Use components that are readily available.
- Setup up a test rig, where each and every produced board can be completely tested before it is used in a system. This includes testing every input and output pin, all analogue measurements, and making sure all the supply voltages are what they should be. Care should be taken to ensure that the board has sufficient hardware to allow the software to test its own voltage supplies.

4.4.16. Choice of Connectors

Some peripherals have standard connectors that the design must accommodate. They are:

- Compact flash card
- JTAG
- Modem interface

Other peripherals allow for a range of connectors. It was decided to choose 100-mil[2] IDC connectors with ribbon cable as they are cheap and widely available. They also stay connected fairly well compared to other connectors. The disadvantages of ribbon cable are firstly its low-current capacity and secondly, most ribbon cable is not shielded.

Thus, any power connectors, or any high-current demands will use other connectors. Any noise-sensitive connectors will have to have a shielded connector and cable.

The following table shows the different connectors and their footprints:

Name	Connector Type
SPI (For future use for the SPI protocol.)	6-pin Wago connector. Allows for shielding, and medium currents.
Bus IO (Connection to HMI board.)	10-pin IDC for ribbon cable
PWM-A	14-pin IDC for ribbon cable
PWM-B	14-pin IDC for ribbon cable
PWM-C	10-pin IDC for ribbon cable
Debug	10-pin IDC for ribbon cable
Interboard-1	34-pin IDC male
Interboard-2	34-pin IDC male
Modem interface	Standard RS-232 serial cable. Allows for shielding, and medium currents.
Compact flash	50-pin IDE interface for CF
JTAG	Standard JTAG connector (14-pin keyed IDC)

Table 2 - Board connectors

The choice of the two inter-board connectors was based on the compatibility requirements in the terms of reference. This board must be

[2] *mil* is the abbreviation for thousandths of an inch. i.e. 100 mil is 0.1 inches.

backward compatible, thus allowing it to plug straight onto older interface boards. With this option available, one can swap out an old DSP board in the field with a new DSP board.

A new interface board is still a requirement for the design of this system, but strong emphasis has been placed on compatibility in the terms of reference.

4.5. Software Design

Together with the hardware, the software must be designed in parallel. It was decided to write code in the c++ programming language because it is a well used standard. The software on the PICs will use the Hi-Tech PIC18 compiler, while the Texas Instruments DSP code will be compiled in the Code Composer IDE using Texas Instruments' own C compiler.

It is important to lay down good foundations for writing software in a large project like this, as the author will not be the only writer of code once the project is finished. The design of the code will be kept modular so that each piece of code can be completed and packaged. The infrastructure of the software will be set up and documented well so that third parties writing code will conform to the infrastructure and keep the code neat.

The various sections or modular blocks of software are discussed in the following text.

4.5.1. System Variables and Settings

Inside the system, many variables need to be stored to be used during processing and also for viewing on the user interface. There are setpoints which control the way the system runs, and there are calibration values which are adjusted at production. It was decided that all these variables be put into logically separated tables, which are generally 16-bit unsigned integer arrays.

The following tables are present in the system:

- Normal setpoints
- System setpoints
- Modem setpoints
- Wi-Fi setpoints
- Data points
- Fault flags
- System status
- Inputs
- Outputs
- Summations

These tables hold most of the information about the system. The data points, fault flags, system status and summations are retrieved by the HMI and user interface software to provide information about the status of the system to the user. All the setpoints-type tables are settings which control the system. Normal setpoints are generally for the user to edit, while system setpoints contains calibrations and other settings that are password protected and for more advanced users or technicians.

For the process of data logging, having an entire snapshot of the system is easy with tables. All the data points and some other tables can be stored without much processing.

Summations are for storing cumulative values such as run time in hours and total kilowatt-hours produced by the inverter.

Input and output tables are mostly mapped directly to the inputs and outputs of the DSP and also of some other processors. It is because the other processors share the input and output loading that it is necessary to have tables for these values.

All the tables are usually directly accessible using the application layer protocol discussed later.

4.5.2. Design of an Inter-Processor Bus Protocol

In the system design, there are five processors that need to communicate with one another.

Several important requirements for the communications are:

1. They must run at a sufficient speed for the application.
2. They must have sufficient error checking to be noise immune.
3. They must be expandable to more processors, or more data.
4. The algorithms for processing bus packets must not be processor intensive.
5. If one or more processors on the bus fail, or stop responding, the bus must continue to run.

It was decided to run the bus protocol using UART for the data layer. As the DSP is the central, most powerful processor, it was made the bus master. Each of the other processors, were made slaves.

In an early stage, a previous design was experimented with using SPI. As with the newer design the DSP was the master. Unfortunately there are some problems with using SPI as a bus protocol.

1. It is designed so that the master sends a byte as well as receiving a byte at the same time.
2. A slave can only send data when the master sends data to it.
3. The master must send dummy bytes out to receive data from the slaves at a rate equal to that of the worst-case response time.

Because of these problems, the newer design uses UART.

The UART was configured as follows:

- 1 start bit
- 1 stop bit
- 8 data bits
- No parity bits
- Baud rate of 115200 bits/s

Just as a reminder the diagram from the hardware design section is repeated here for reference.

The following diagram shows the way the UART bus is connected in terms of data flow:

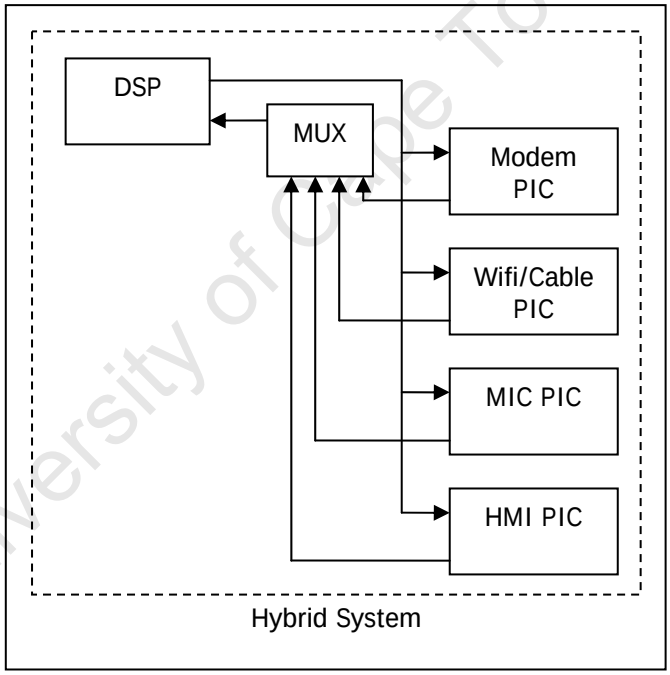


Figure 13 - Data flow for the UART bus

The polling algorithm for the bus works as follows:
Each slave is polled to give it a chance to send any data. If it has data to send, it sends it. Next it must send an end-of-data marker. This marker can follow directly after the poll if there is no data to send.

The master can send data at any time as there is no multiplexer in this direction. Should the master need to interrogate a slave, it polls it, but

indicates that the slave must not send the end-of-data marker. This allows the slave to respond to a request by the master. Once this is finished the master sends an indication that the slave must stop sending data, and the process is complete.

Should a slave not send an end-of-data marker after being polled, or send its own data, the master will eventually send its own indication that the slave must stop sending data. It then moves onto the next slave.

Should a slave not respond at all to a poll, the master notes this and stops polling it for a certain amount of time. The reason for this is to not put unnecessary delays on the bus. The master will then poll this faulty slave only from time to time until it responds again.

This feature allows the slave devices on the bus to be hot-swappable, that is to say that any slave device may be removed from the bus, and the bus's performance will not be significantly affected. Any new slave device can join the bus at any time as well.

The following flow diagram shows how the master polls the slaves:

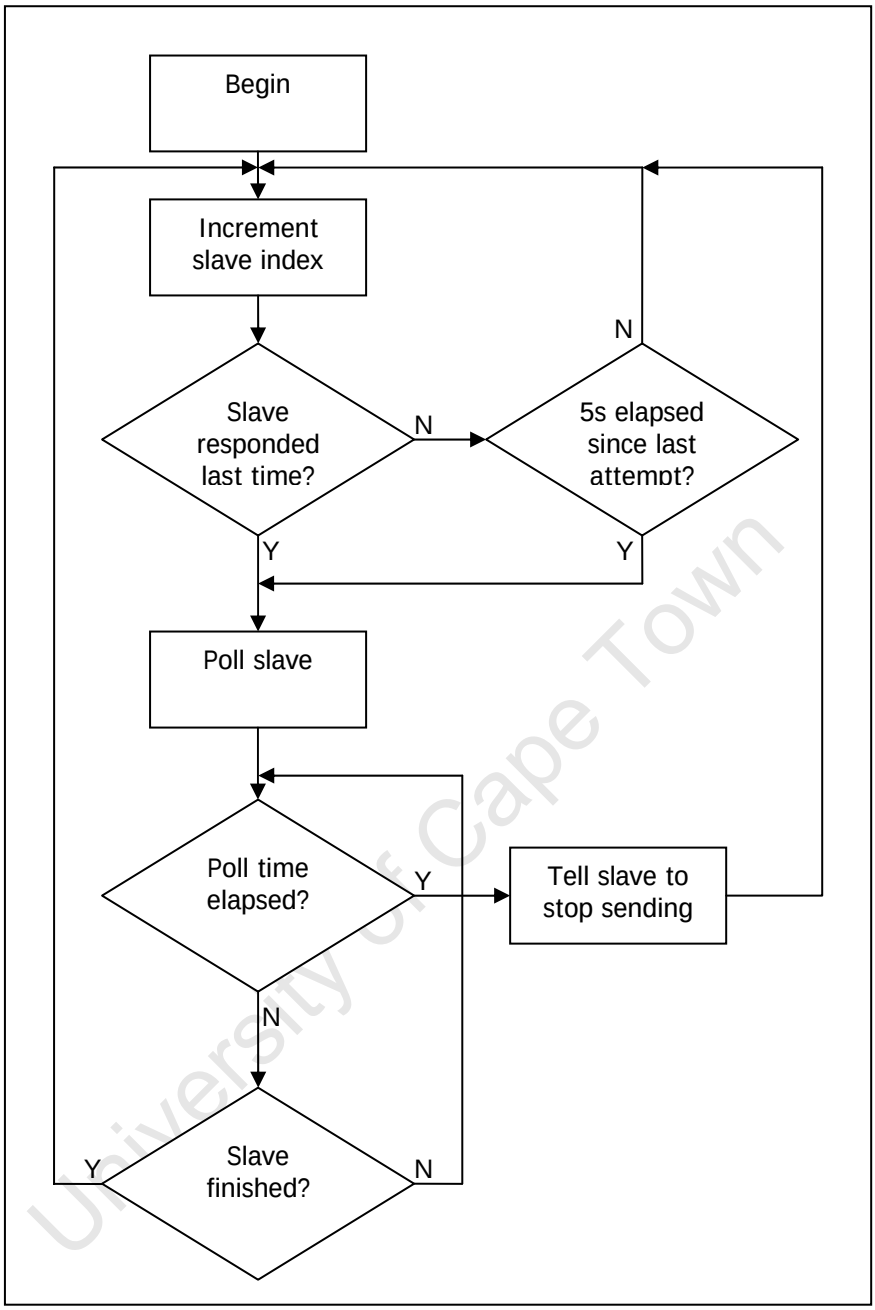


Figure 14 - Flow chart of the polling algorithm

The following message diagram shows the polling process with different types of interactions between the master and the slaves:

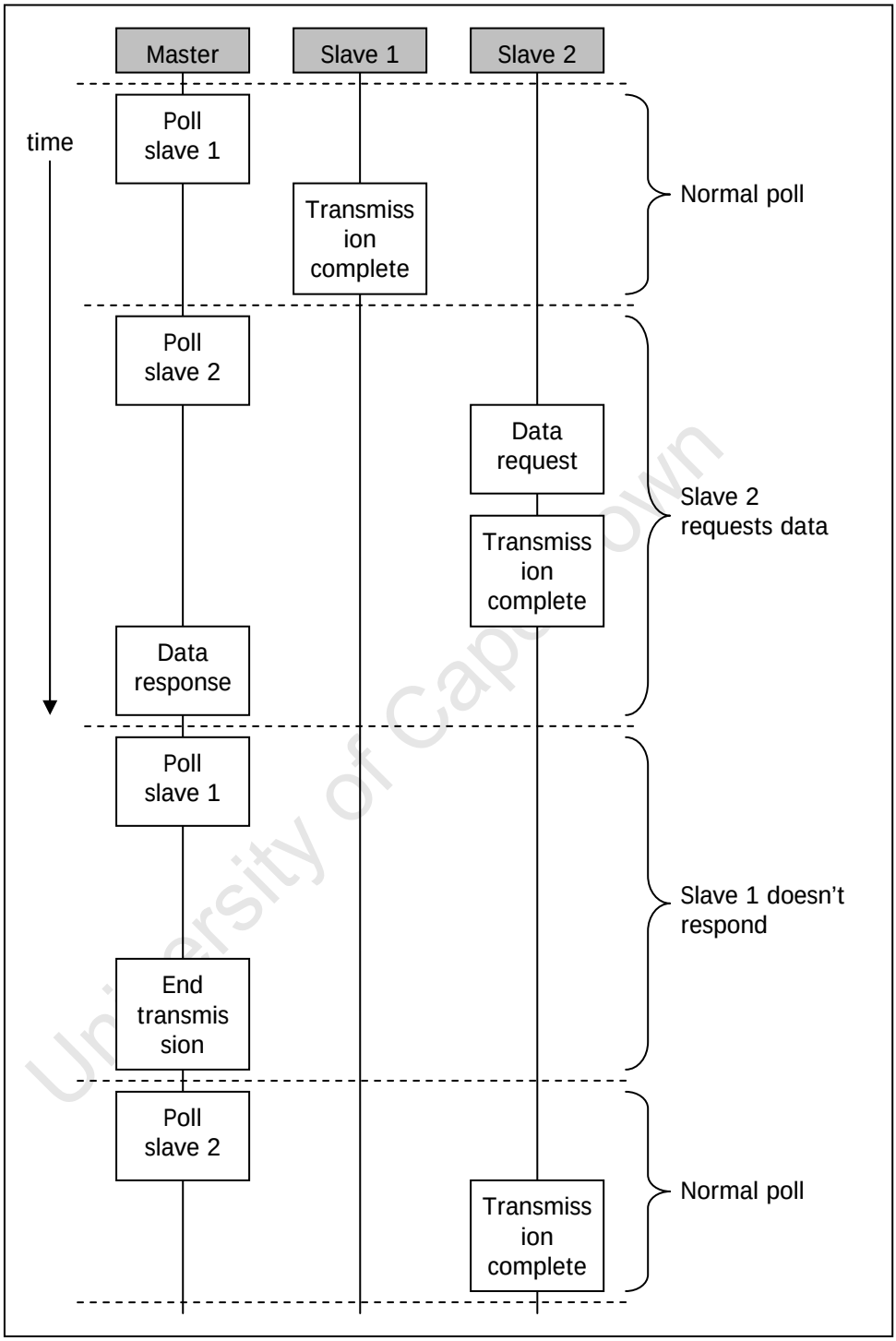


Figure 15 - Sample message diagram from the inter-processor bus

More information on this protocol at the network layer can be found in section 10 (Appendix III – Paper on the Inter-Processor Bus).

4.5.3. Bus Protocol in this Application

Now that the generic design of the protocol is completed, the specific details on the data layer need to be designed. Also, the actual data packets necessary for the application layer are discussed only briefly here.

To begin a packet, the sending device must send the byte *AA*. Please note that all *italic* numbers are in hexadecimal format. This indicates the start of any packet. Next an address byte in the form *50+i* where *i* is the slave number being send to, or receiving from. For example, the master sending a packet to slave 2 would send *AA 52*.

At the end of each packet, a checksum byte is transmitted. This byte is the XOR operation performed on all the bytes excluding the first two bytes in the message. This checksum byte must be verified against the calculated value during reception to confirm the integrity of the packet. Should a mismatch occur, the packet must be ignored completely.

To poll a slave, the master sends only one byte, in the bit format *1110 hsss*. The *h* field is a hold bit. This indicates to the polled slave that it must not send the transmission complete message as the master desires to make a request to which the slave must respond. The *sss* field is a slave number, where *0* is reserved for indicating that no slaves are allowed to send. The values for *sss* are then 1 to 7. During packets, to avoid confusion, the bytes *E0* to *F0*, which cover all poll bytes are reserved. *F0* is used when in a data packet any of these reserved values are to be sent. Firstly *F0* is sent, then another byte which is a value from *00* to *10*, which translates to the range, *E0* to *F0*.

Note that transmissions from the slave to the master do not have to escape the range above as the polling bytes are never sent to the master.

The standard message format can now be shown as follows:

AA	*5x*	[application layer data …]	[checksum byte]

Now the application layer data packet structure needs to be defined. It was decided to use a request and response type structure. Some typical example packets that need to be transmitted between processors are as follows:

- Switch inverter off
- Request various voltages and currents to display on HMI
- Requests to retrieve logs
- Unsolicited responses for input and output states

It was decided to always contain one application layer data packet inside one lower layer packet to simplify the software. An important aspect to decoding packets in software is knowing the expected length of the packet. The length must be determined by reading the third byte in the lower layer packet, which is the first byte in the application layer packet. This byte indicates which type of packet it is. For most types, the length is set for that type. For example, a data point request will be indicated by a byte, *3F*, at which point the software must know that for this type the length is always 2. The next byte is the data point index number, and the packet is complete. For more complicated packets, multiple setpoints may be sent, in which case the software must know that the length is calculated by a length byte in the packet. An example of such a packet follows:

AA	*5x*	*33 00 03 11 11 22 22 33 33*	[checksum byte]

The application layer packet is decoded as follows:
33 indicates the packet type *Multiple Setpoint Response*.
00 is the start of the range to update, i.e. setpoint zero.
03 is the length of the range to update, i.e. setpoints 0 to 2 will be updated.
11 11 is the 16-bit value to be written to setpoint *00*.
22 22 is the 16-bit value to be written to setpoint *01*.
33 33 is the 16-bit value to be written to setpoint *02*.

Here it can be seen that the application layer packet's length can be calculated with the following formula:

$$length = 3 + (2 \times buffer[2])$$

These formulas need to be hard coded into the software so that the expected length of every type of packet can be known.

Note that all 16-bit values and 32-bit values are sent as most significant byte first and least significant last.

In terms of error detection, there are a few different methods here:

1. The checksum at the end of every packet must match the calculated value.
2. There may not be any delays of more than a certain amount inside packets.
3. Each packet must start with a specific sequence.

For all error detects, it is necessary to delete the packet or portion of a packet from the input buffer in the processor. Any subsequent bytes received that are not polls, or start sequences should be deleted as well.

In general there are two types of transactions on the application layer: Firstly there is the request and response type. This is a typical interaction between two devices where one needs data from the other. Secondly there is a response and acknowledgement type. This is where one device needs to inform another of some event, or update some data without the other device requesting it. The other device will then acknowledge the response, indicating that it has accepted the response.

There is an application layer data packet called *Error/Acknowledgement*. It contains two additional bytes, one of which indicates success (0) or returns an error code (other than 0). The other byte is a reference to which response it is acknowledging. This is a packet type indicator.

Further details of the application layer protocol are not discussed in this document. They are not really relevant to proving the success of the design.

4.5.4. Implementation of MODBUS Protocol

In order to make the system compatible with industry standard SCADA interfaces, it was necessary to implement some industry standard protocol. MODBUS was chosen as it was the simplest and widely used.

Generally a SCADA interface on a PC is a master in the context of the MODBUS protocol, and the system (in this case, the hybrid system) is the slave.

In the MODBUS protocol, a master polls a slave and the slave responds once only. The slave may not send unsolicited data to the master or any other slave.[10]

MODBUS supports multiple slaves, so the hybrid system(s) could be part of a large MODBUS network.

According to the MODBUS protocol, a master and read and write from and to selected address ranges in order to communicate with the slave. Many different operations are possible with MODBUS. Only a subset of these operations is implemented in the hybrid system.
The operations or "function codes" implemented are:

- 3 – Read multiple words from memory addresses 40001 to 50000.
- 4 – Read multiple words from memory addresses 30001 to 40000.
- 6 – Write a single word or memory in the address range 40001 to 50000.

In order to setup a SCADA interface, the memory ranges that the MODBUS function codes read and write need to be mapped to the various tables of variables and settings on the DSP. A mapping was created and documented. This mapping is stored across all the processors that need it.

As with the hybrid system being the MODBUS slave device, it must also be the master device in a separate MODBUS network. This is

implemented to interrogate MODBUS compatible meters and any other slave devices that might need to interface with the hybrid system.

4.5.5. Wi-Fi Interface

There is dedicated PIC microcontroller for controlling a Wi-Fi interface. This PIC also controls the RS-232 interface. The Wi-Fi and RS-232 mediums will allow the user interface software to connect to the system using the industry standard MODBUS protocol as discussed above.

The PIC software must be able to retrieve settings from the main DSP and then use those settings to initialise the Wi-Fi module accordingly. If the Wi-Fi module is not detected on start-up, the PIC must know that the connection will be a simple RS-232 one. Given that, there is not much difference, as the Wi-Fi module pipes TCP/IP data through the RS-232 port.

Apart from supporting MODBUS over the Wi-Fi or serial connection, another protocol, namely the remote programming protocol, needs to be supported. This protocol exists between a custom piece of software and the PIC. The software has been written for transmitting compiled software over TCP/IP or over a modem connection. This is discussed more in the Remote Programming section below.

4.5.6. Modem / GSM / GPRS Interface

The hybrid system has a modem connected to one of the PIC microcontrollers in the system as discussed in the hardware section.

The software to interface with the modem has to implement all the following functions:

1. Dial-up to a SCADA interface that runs MODBUS over the connection.
2. GPRS connection to a server on the internet.
3. Remote software uploading.
4. SMS control.

The software must be able to accept incoming data calls to establish a connection between a PC and the system. This connection will run the MODBUS protocol as discussed above. The software needs a secure mode and non-secure mode, i.e. checking the incoming phone number or just accepting any incoming call. A pin code is still required when any setpoint changes are made to any system.

For future developments, a server connected to the internet could be set up. Using GPRS (which is normally much cheaper than a data call), the system can connect and periodically upload data and event logs. Anyone with a username and password could then log onto the server from any internet connection (with no custom software) and view logs and change settings. This is a much more elegant solution to the remote access, but requires an infrastructure of a server.

Apart from supporting MODBUS over the modem connection, the remote programming protocol needs to be supported. This is discussed more in the Remote Programming section below.

SMS control entails being able to receive text messages from a cell phone that contain commands such as “system off” or “stop inverter”. This can

be done by programming certain phone numbers into the system. Only messages from these phone numbers will be interpreted for security. Acknowledgement messages can be sent back. The user should also be able to interrogate the system's status with a text message.

4.5.7. Bootloaders for Remote Programming

To allow for remote programming, custom bootloaders must be written for each processor type. These bootloaders must be able to program the memory that stores the main program that runs on that processor.[3] All the PIC microcontrollers in the system are remote programmed the same way, and therefore have the same bootloaders.

Figure 16 shows the sequence for entering the bootloader:

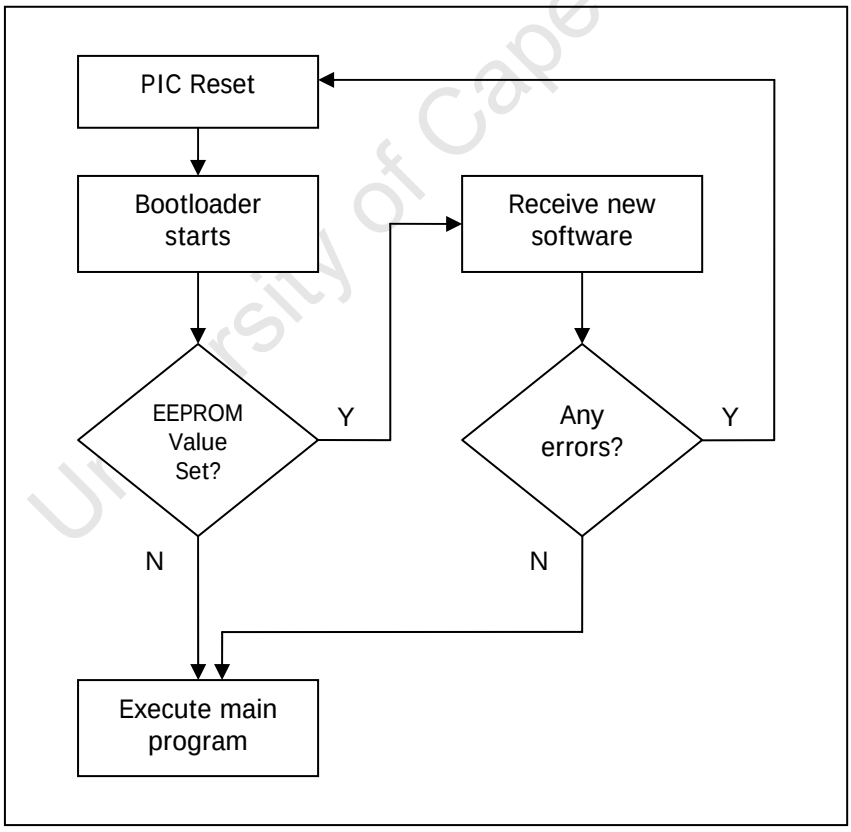
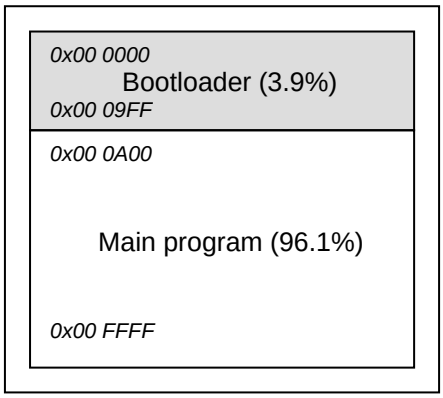


Figure 16 - Flow chart of the operation of the bootloader on the PIC processor

The PIC's program memory is organised as follows:



The bootloader uses either of the PIC's serial ports to upload the new software. The PIC microcontrollers are able to program their own FLASH whilst executing program code in FLASH. This simplifies the bootloader software.

The FLASH must be erased in rows of 64 bytes, and programmed in groups of 8 bytes. The bootloader reads in rows of 64 bytes until its RAM is full. The PICs have 3.8kB of RAM, and the bootloader automatically detects this. Once this RAM is full, the bootloader then erases and programs row by row. When all the rows of the main code are programmed, the bootloader indicates so with an acknowledgement, then branches to the main program, starting at address *0x00 0A00*.

Unlike some systems, this remote programming feature needed to be fail-safe. To combat this, it was decided to buffer the entire new software on the compact flash card on the board. Since the DSP is the processor assigned to communicate with the compact flash card, the program had to be sent to the DSP to be stored first. The DSP now writes the whole program for the specific PIC to its compact flash card. Then, the remote connection can even be dropped entirely, and the DSP will then program that PIC with the software it has just written to the compact flash card. This makes the upgrading process far more reliable and robust.

The upgrading of the DSP presents a more complex challenge. It is manufactured with a standard bootloader in its memory. This bootloader uses a special protocol. In contrast the PIC has no bootloader by default. A special programming tool is used to put the bootloader on the PIC. The DSP's bootloader uses the serial port like the PIC's, but to implement the robust, reliable methods like on the PICs, it is necessary for the DSP to read from the compact flash card and program its own FLASH. For this reason, a custom bootloader needed to be written to access the compact flash card.

The DSP's program memory is organised as follows:

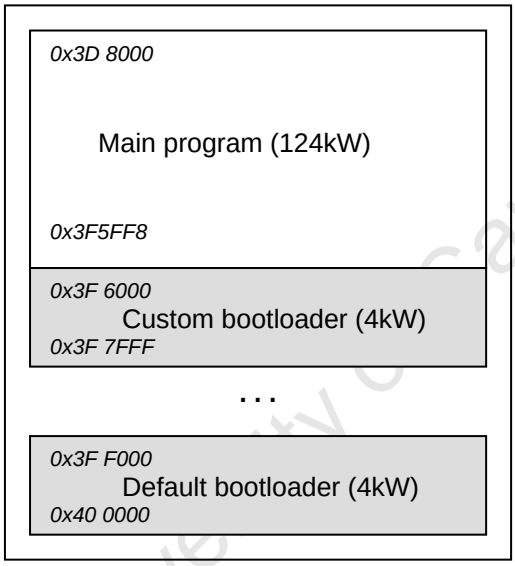


Figure 17 shows the operation of the two bootloaders that run on startup:

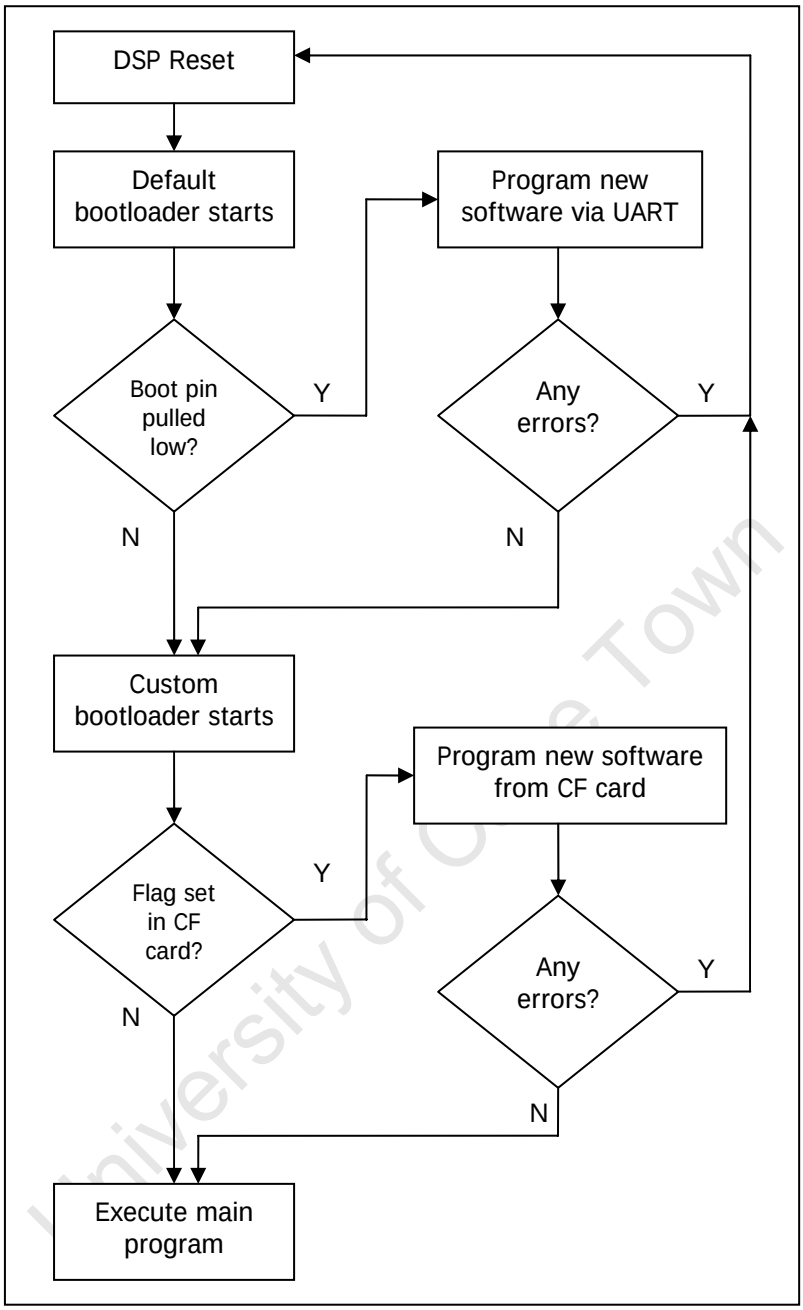


Figure 17 - Flow chart of the operation of the bootloader on the DSP

Figure 18 shows the typical communication setup:

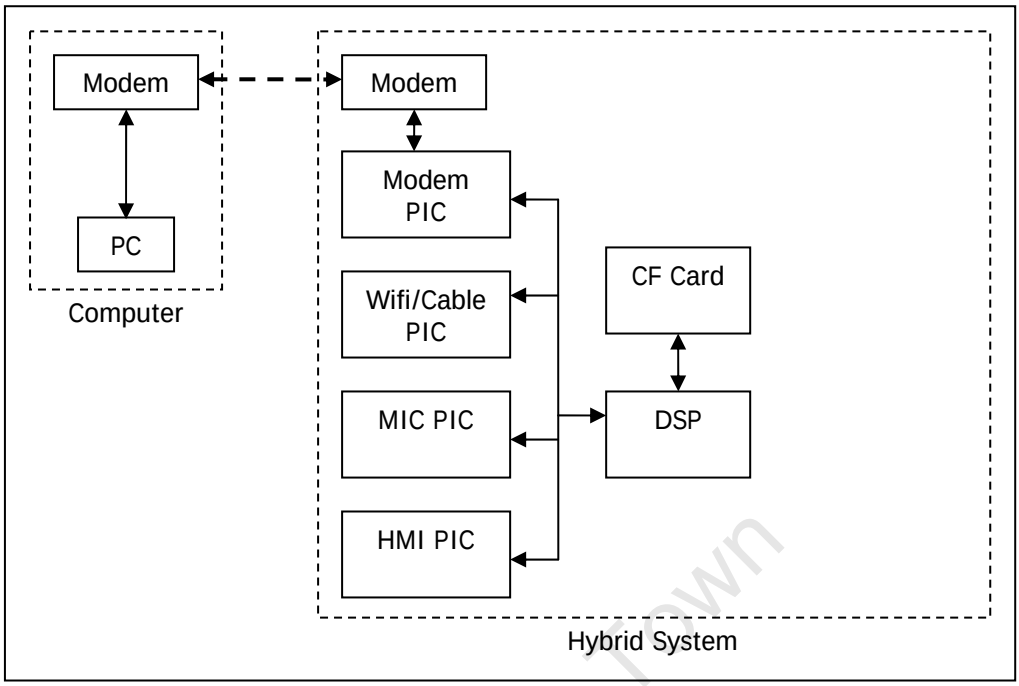


Figure 18 - Communications between the computer and the hybrid system

A protocol exists between each of the components in this setup that need to communicate directly with one another.

Firstly the PC to modem communication is done using the standard AT command set. This same protocol is used by the Modem PIC to communicate with its modem. The PC establishes the connection over the modems, and the Modem PIC accepts. Once the connection is established, the modems become transparent and the PC then communicates directly with the Modem PIC. The protocol between these two (or between the Wi-Fi connection and the Wi-Fi PIC) works as follows:
The PC sends a request to write to the software buffer memory on the compact flash (CF) card. This is acknowledged by the PIC. The PC then starts sending 128-byte packets of data with the corresponding address and various checksums for data integrity. It sends up to 16 packets without any acknowledgements. The PIC receives the packets and acknowledges each one individually. This method of sending packets in advance is called a sliding window.

The reason to do this is that there are often high round-trip times (RTT) on modem connections, even though the data throughput is fast. That is to say that sending 1-byte can take 1 second, whereas sending 256 bytes might take 1.01 seconds. So there is an inherent delay in sending a packet and getting an acknowledgement. Thus if the sender blindly sends packets out, and they get acknowledged later, the effects of this problem can reduced dramatically. If there are any errors, all the packets that have not been acknowledged yet, must be resent. In most cases, this never comes up, so this method works well.

Once the Modem PIC has received enough data, it sends that data to the DSP over the inter-processor bus.

The DSP then stores the data on the compact flash card using the 16-bit IDE interface as described in the Compact Flash Specification (Version 3.0).

Once all the code has been sent over the link and stored in the CF card, the PC sends the go ahead for the DSP to start programming the target PIC, or itself. The link can then be dropped and the programming will continue. The DSP is the device that programmes the PIC. It uses the physical inter-processor bus to program the PIC, but does not send inter-processor bus data packets, instead it sends the data required by the custom bootloader on the PIC. The other PICs are “told” in advance not to confuse this data with the inter-processor bus data. Once programming has completed, the DSP writes flag bits to the status values on the CF card. When a connection is re-established to a PC, the PC can confirm that the new software has been programmed onto the PIC by reading these flags.

When the actual DSP is to be reprogrammed, the process is different after the software has been stored in the CF card. The DSP will write certain flags in the status values on the CF card. It then resets itself. When it boots up, the default bootloader runs as per normal and loads what it considers to be the main program. In actual fact, the custom bootloader is started.

This custom bootloader reads the flags from the CF card. If certain flags are set, it begins reading from the CF card and writing to its own FLASH memory. As with the PIC programming, it finishes by writing flags in the status values on the CF card which will be read by the remote PC to confirm the software upgrade.

The custom bootloader written for the DSP is more complicated than that of the PIC as the DSP can not write to its FLASH memory whilst executing code from that FLASH memory. The PIC can do this, but with the DSP, the code needed to program the FLASH has to be copied in the volatile RAM memory, and then execution is transferred to the RAM. Once the FLASH is modified, it can resume execution from the code in the FLASH.

After the DSP bootloader was working, it was found that to program the DSP during development, it was not feasible to use remote programming every time as it was significantly slower that using the default bootloader to serially program the DSP. It was then decided to update the custom bootloader to include a serial loader like that of the default bootloader. It was found that the default bootloader's serial interface was very sensitive and would often take up to 5 DSP resets to get a successful connection. This motivated the team even more to make serial programming a requirement for this custom bootloader. The author then built this functionality into the custom bootloader, and after completion, it was found that the new serial programming was slightly faster than that of the default bootloader.[3] The custom bootloader was also found to be far more reliable, and also had the advantage of not having to physical power down and reset the DSP manually.

[3] Test results on loading a 144kB program onto the DSP were as follows:

1. Default bootloader: 90.3s
2. Custom bootloader 79.8s

It can thus be seen that there was an improvement in speed.

Figure 19 shows a typical screenshot of the PC software written in Visual Basic .NET by the author for programming PIC and DSP processors, either remotely or locally by connecting to the custom bootloaders.

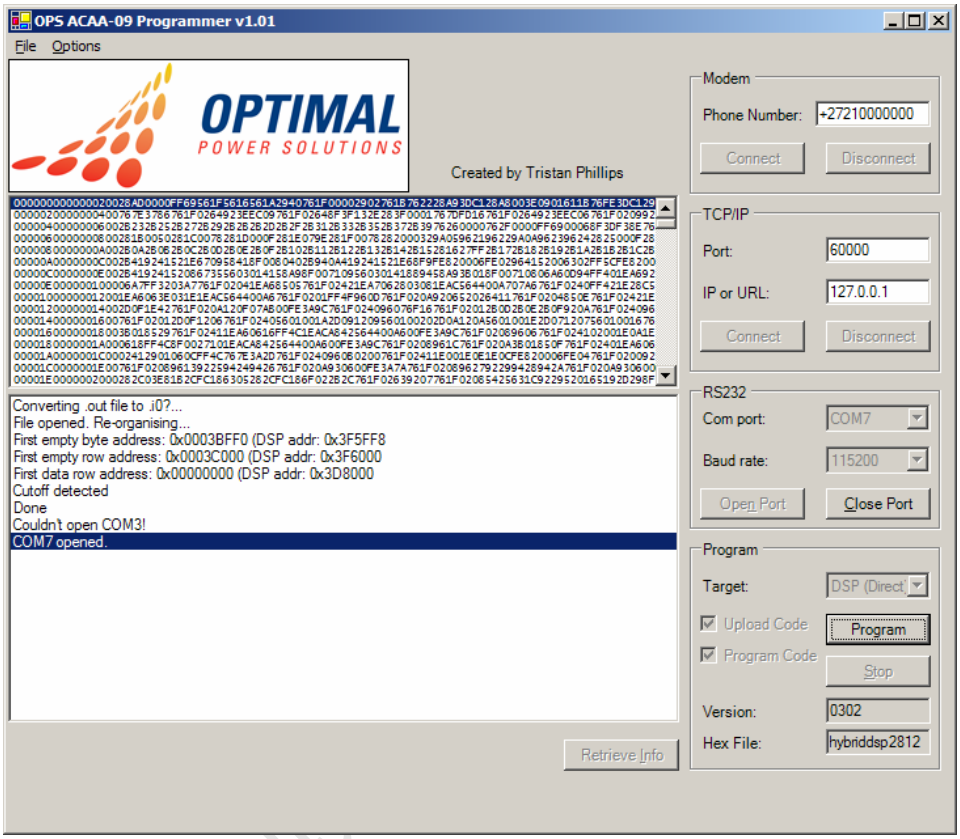


Figure 19 – The hybrid all in one programmer

4.5.8. Data and Event Logging

All the logging, data and events, need to be stored on the Compact Flash (CF) card. The CF card should be connected all the time for this. It was decided that the CF card should not be removable for retrieving logs locally. The retrieval method should always be through one of the many communication options. Therefore, the CF card becomes a static component on the DSP board.

The CF card is organised into sectors of 512 bytes.[12] The 16-bit interface is used to control the CF card through the reading and writing of registers.

This system conforms to the True IDE standard.[13] [20] This is the same standard that is used to communicate with hard disk drives.

Initially it was decided to use the FAT16 or FAT32[14] standard to write "files" onto the CF card. This would then be readable by PCs, just like a USB FLASH drive would. Due to the decision to keep the CF card permanently on the board, it was not seen as an advantage. The overheads of FAT16 and FAT32 are such that for the small amounts of data used in this logging system, up to 3 times as many reads and writes were necessary to create a log compared to not using any FAT format. It was thus decided to adhere to a simpler, custom indexing system.

Source code from various Texas Instruments application notes were used to create the software for reading and writing to the CF card.[18] [20] [21]

Generally, the standard sizes of CF cards start at 512MB at the time of design. The formats of the two log types are detailed below:

Data log

Field Size (bytes)	Field Name
4	Data log index (primary key of the log)
6	Time/date
2	Data field 1
2	Data field 2
…	…
2	Data field N
4	Summation Value 1
4	Summation Value 2
…	…
4	Summation Value N
22	Debug information
4	Event log reference
2	Trigger code

The total size of the data varies between 400 and 500 bytes depending on the number of data fields logged. It was decided to allocate 1 sector (512 bytes) to each data log to simplify code and to support future increases in size.

Event log

Field Size (bytes)	Field Name
5	Time/date
1	Type of event
1	Index of event
1	Trigger code

The total size of the event log is 8 bytes. It was decided to compact 64 event logs into each sector.

$$(\frac{512}{8} = 64)$$

Note that the event log type does not support future increases in size. Having more event logs available was favoured over scalability in this case.

The event log number is not stored in the actual event. Instead, the start number, and count of event logs are stored in an index sector of the CF card. The data log start number and count are also stored in this index sector for redundancy.

The way the logging works, is that a data log is created regularly at a predefined configurable interval. Additionally, when an event occurs, a data log is created.

The event logs will be created whenever anything changes in the system, for example, when the battery voltage goes too low, or when someone enters a command to switch off the system.

Having this detail of logging will allow someone to trace back to the time of a fault and see exactly what happened.

The arrangement of the CF card's sectors is as follows:

Sector number	Use
0	Backup of setpoints
1	Index sector for logs
2	Index sector for code storage for remote programming
3	Backup of setpoints (continued)
4 to 262147	Code storage for remote programming (DSP code (256KB))

262148 to 393219	Code storage for remote programming (PIC code (128KB))
393220 to 524291	Code storage for remote programming (PIC code (128KB))
524292 to 655363	Code storage for remote programming (PIC code (128KB))
655364 to 786435	Code storage for remote programming (PIC code (128KB))
786436 to (0.75N – 1)	Data log space
(0.75N) to (N – 1)	Event log space

Note: N is the total number of sectors available on the CF card. (Typically N would be approximately 1×10^6 for a 512MB CF card.)

4.5.9. USB Interface (Future)

The addition of USB slave functionality can easily be added to the system. For the current design it was felt to be redundant, having so many other communication mediums available. However, very easy-to-use USB to UART chips are readily available. These chips use all 8 signals defined in RS-232 on the processor side, and on the USB side it emulates a serial port on a PC. That is to say, when the cable is connected to the PC, a new serial port is detected, and can then be used like any other serial port. Thus, the standard user interface software could still be used.

These USB to UART chips usually provide some storage space where custom strings can be stored so that on the PC side the device can be detected with the correct company and product name rather than some default name, which makes the connection look more professional.

4.5.10. Security

As discussed in the requirements, it is necessary to implement some form of security to prevent illegal replications of the system being sold illegally.

One cannot simply use a standard password to activate each system because if the password becomes known, the exercise is useless.

It was initially decided to use the serial numbers of the compact flash cards in the systems to produce a unique number for each system. This number would then be given to the holders of the intellectual property who would feed the number into an algorithm which produced another number. This number would be returned to the buyer who would then input the number to the system. The system would then use the same algorithm to verify the number. This method was implemented successfully, but then in the testing phase, it was found that cheaper (and possibly illegal) compact flash cards could be bought from less reputable suppliers. These cards would all have the same serial number. The problem then arose that all the systems required the same number to activate them and this obviously defeats the purpose.

After that failure, it was necessary to find another source of a (more or less) unique number. The following method was investigated and proven to be successful in producing almost unique numbers every time:
The DSP measures all 16 of its analogue to digital converters. It then takes each of the 12 bits of each reading and reverses the bits. Now the least significant bits become the most significant. This means that small changes in the values propagate large changes in their reversed bits. These 16 readings are then each multiplied by pseudo-random prime numbers and added together. All in all the final number is a 32-bit integer which appears to be random and mostly unique for each system. This number is only ever produced once when the system first starts up. The number will not be produced if specific voltages are applied to its analogue pins in order to get it to produce a known number. Once this number is produced, it is saved in EEPROM that is not accessible by any means other than by the software on the chip. This value is then read on start-up and modified slightly. The user can then view this number and quote it to the intellectual property holders. The algorithm is applied as before and a resulting number produced and given to the user. The user then enters this 32-bit integer into the system and it is stored. So long as the number entered is verified (continuously) the system will continue to run.

An initial period of a month is allowed without a correctly verified number purely for the testing phase of each system.

To verify the uniqueness of the numbers stored on the system, a test was done. 1000 numbers were produced and no two were the same.

The verification algorithm can unfortunately not be revealed in this document for security reasons, but was also tested to produce about 1,000,000,000 unique results from 4,294,967,296 input numbers. This means that the algorithm can produce one result for multiple inputs. This is good as it means the algorithm is a one-way algorithm, so it can not easily be reverse engineered. It still produces enough numbers to thwart any attempts to crack the algorithm.

A website that produces the verification numbers was created by the author. Licences can be purchased online. The website is connected to a MySQL database running on a server. The web content is generated by PHP code. More details on this licensing are out of the scope of this document. This has been running with no problems and it can thus be concluded that the security appears to work.

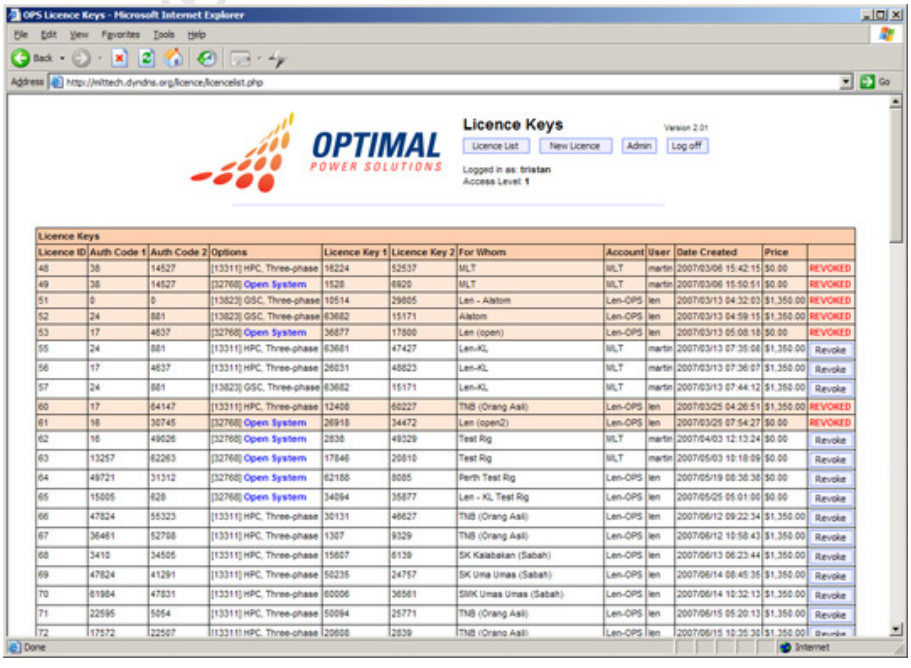


Figure 20 – A typical view of the licensing website

5. Results and Field Testing

This section describes the finished product of the design and requirements. These results are discussed in detail in the next two sub-sections.

Following that, some different tests and metrics will be discussed which outline the aims and goals for this application. Then, the actual tests will be done and performance measured and analysed.

5.1. Finished Hardware

This section will focus on the main DSP printed circuit board. The other components of the hardware will only be mentioned as necessary in the section about the application.

5.1.1. Printed Circuit Board

The main result of this entire thesis is the result of the design of the main DSP printed circuit board. This board has been designed to meet all the requirements as well as suggestions laid out in the terms of reference.

Much time and effort was put in to ensure that the board is noise immune and robust. Also, the ease of production and testing was taken into account.

The board is printed on FR-4 material 1.6mm thick. It is a four-layer board. The middle two layers are used primarily as ground and power planes, and to a lesser degree for routing certain high current power tracks.

For future production, if the cost of the PCB is a major issue and the interference is not serious, the board can be printed as a cheaper two-layer board. All the tracks and planes on the middle layers are redundant, and only provide better performance under noisy conditions. The power tracks

on the one middle layer are also redundant as they only provide an additional path for current to flow.

To further save costs, the minimum drill size is 0.6mm in diameter. This allows for higher tolerances on milling machines. The closest copper to copper spacing is 8 mil (8 thousandths of an inch, or 0.2mm), which is a reasonable and standard expectation for local manufacturers.

Figure 21 and Figure 22 show the placement of the most important parts of the board.

Figure 21 shows a component overlay of the top side of the board:

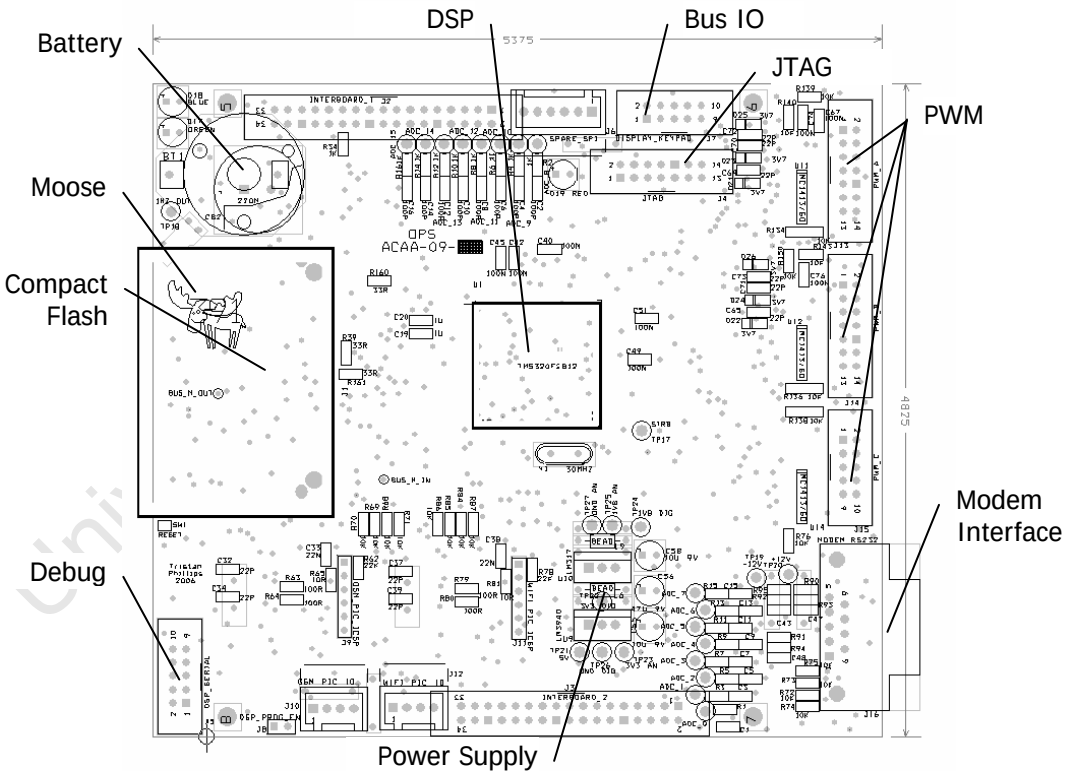


Figure 21 - Overlay of the top side of the DSP board

As can be seen from the diagram, the components are laid out in a logical manner where they are grouped by function as much as possible. This made routing the tracks easier, but more importantly; the tracks are shorter and have less vias, making them less susceptible to interference. The

connectors are placed near to the outsides of the board so as to keep all cables moving away from the board.

The main DSP is placed in the centre of the top of the board as it is connected to the most other components.

Figure 22 shows a component overlay of the bottom side of the board:

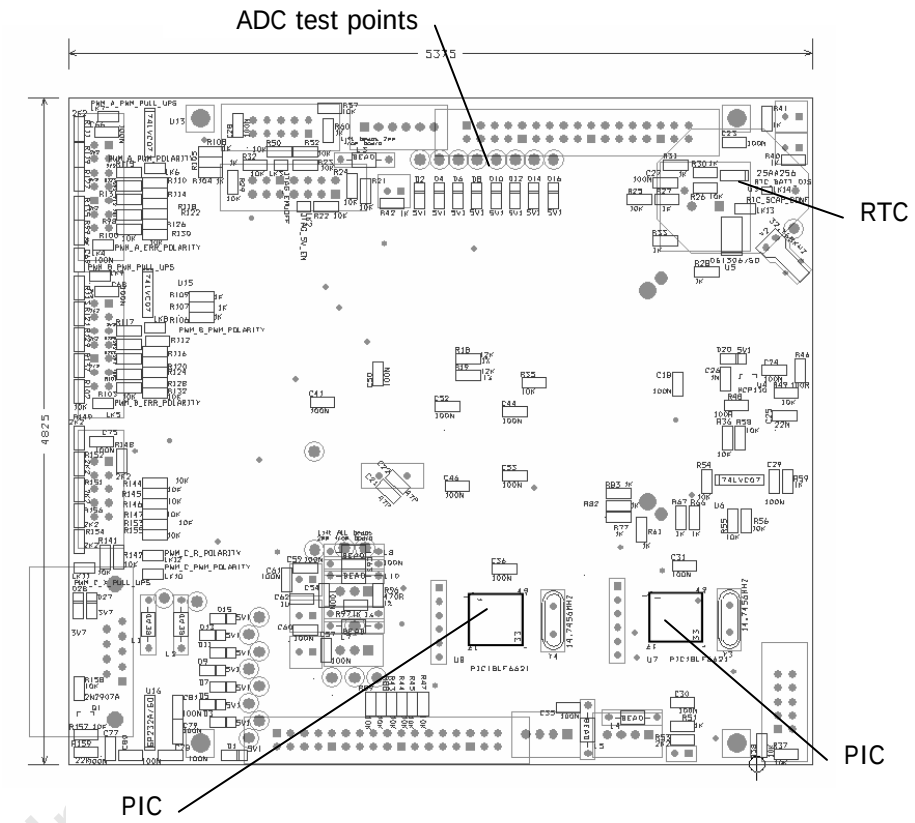


Figure 22 - Overlay of the bottom side of the DSP board

Figure 23 shows a photo of the top side of the board:

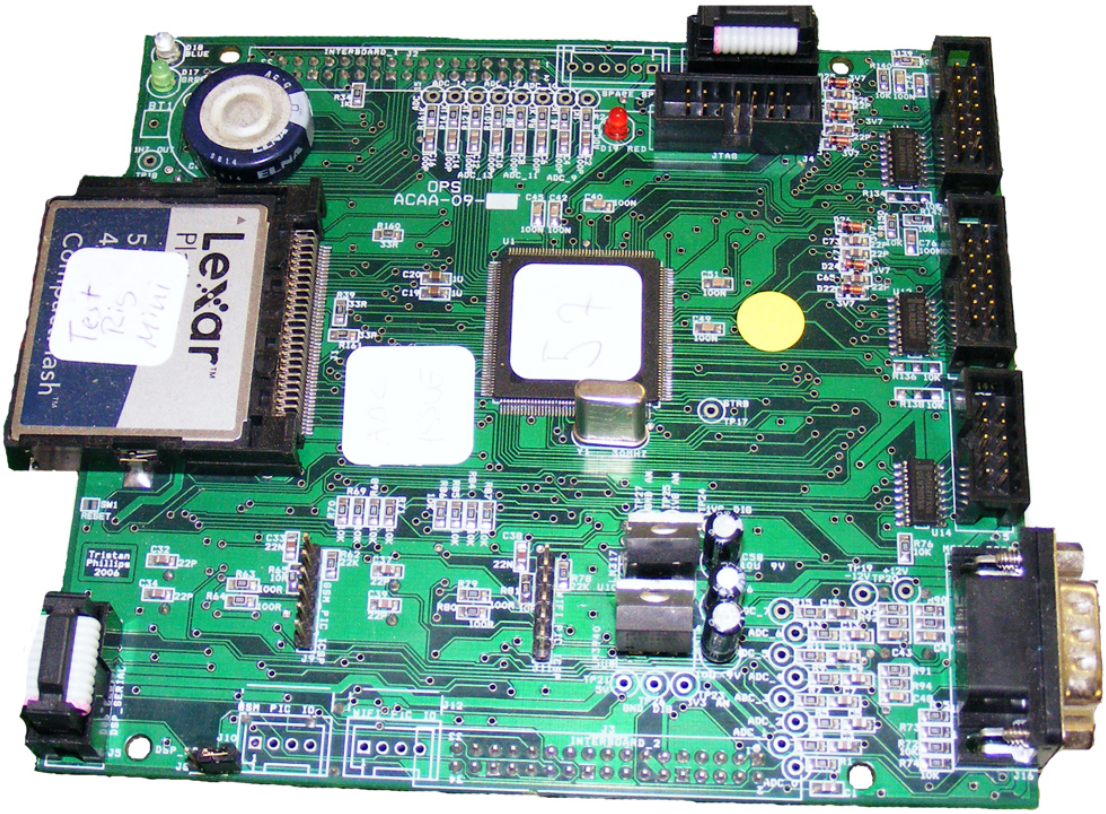


Figure 23 - Photo of the top side of the board

5.1.2. Component Selection and Sourcing

For the most part, the components on this board are cheap and easy to source. No components are obsolete. The most difficult components to source are:

- Serial (SPI) real-time clock DS1306 (SO8)
- Serial (SPI) EEPROM / FLASH chip 25AA256 (SO8)

These are only difficult due to the small quantities ordered during testing.

5.2. Finished Software

Together with the hardware of the DSP board, a package of software has been produced. This software has been designed in layers of differing amounts of abstraction. On the lowest levels the software interacts with the hardware, and on the highest levels, the specific application details are coded.

The diagram in Figure 24 shows these levels of abstraction for the main DSP only:

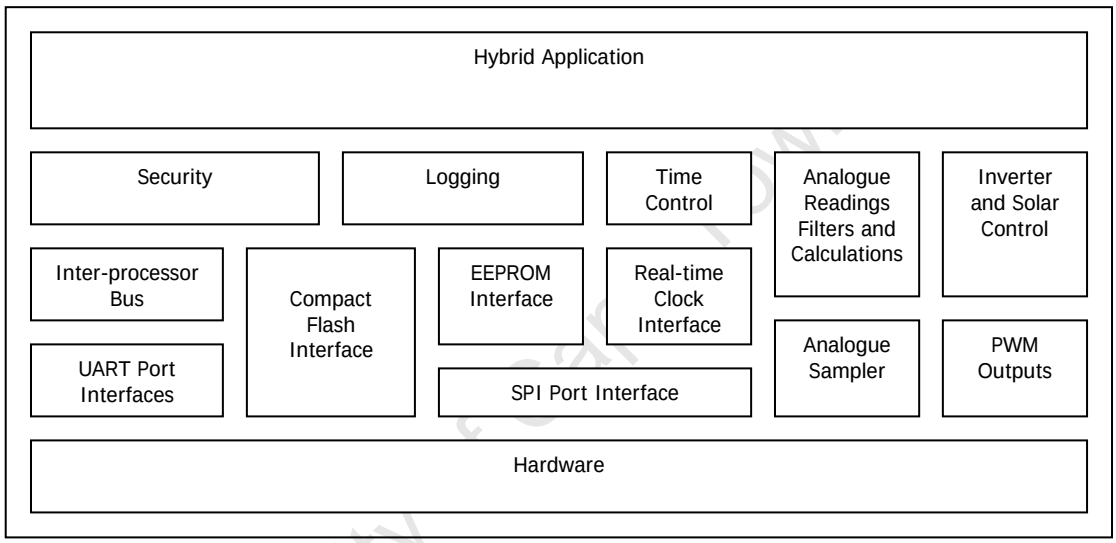


Figure 24 - Software blocks

Each block in Figure 24 represents a logical section of code. The high up the blocks are, the more abstracted they are. Blocks below other block implies that the higher block uses the all blocks below that overlap it. For example, it can be seen that Logging uses Compact Flash Interface as well as EEPROM Interface. The Hybrid Application is at the top as it defines the overall behaviour of the DSP board. Note that it does not interact with any blocks of code not directly below it.

It can thus be seen that the software has been designed in such a way as to allow for a fairly easy change of application. For future developments, this allows a third party to easily change the application block of code on the top, and not worry about any of the lower layers. They can be seen as modular components that can be used as is.

The software for the PICs has been written in a similar fashion. The difference is that there are multiple PIC processors. The code that interacts with the inter-processor bus, for example, is used by all the PIC processors. A package of common code has therefore been put together for all the PIC processors to use in their respective software compilations. The advantage of doing this is that bugs found in any of the common code can be fixed by changing only one set of code, and the changes will be propagated to all the PIC processors once their software has recompiled. Note that the common code is shared at compile time, not during run time. Some examples of the logical blocks in the common code are the MODBUS protocol handling, the inter-processor bus, delay functions, memory reading/writing and remote programming handling.

The final deliverables on the software are as follows:

- All but the top Hybrid Application blocks of code on the DSP
- Common code for all PIC processors
- Specific code for the PIC processor on the interface board
- Specific code for the modem PIC processor
- Specific code for the Wi-Fi / RS-232 PIC processor
- Remote programming user interface on from PC

The missing parts of software have been complete by third parties. The Hybrid Application block of code has been completed by Martin Becker. The specific code for the PIC processor on the human interface board has been completed by Stanley Adams. The software for the user interface that implements MODBUS protocol has been completely written by Richard Parry.

5.3. Implementation of Application

As discussed, the application chosen to demonstrate the design was a hybrid power system, integrating a battery charger, an inverter, a load, solar and wind controllers, and two grid/generator inputs.

The scope of this document covers only the implementation of the hardware and software infrastructure in this application, and not the full details of the power management.

The aim of this section is to document the implementation of the hybrid system as far as necessary to prove the success of the design.

The following components make up a test rig which is a fully working hybrid system that uses the newly designed hardware and software.

5.3.1. The Test Rig Overview

A test rig was built by MLT Drives to test the new hardware and software in a real environment where it can demonstrate a successful hybrid system. The test rig incorporates all the peripherals that the new system can interface with. The inverter is rated to 4kVA. All the readings and sensors are scaled by a factor of 10 so that the readings and logged data will show values that are closer to that of its implementation in the field. The physical layout of the test rig is discussed in detail here.

The inside of the cabinet door is shown here:

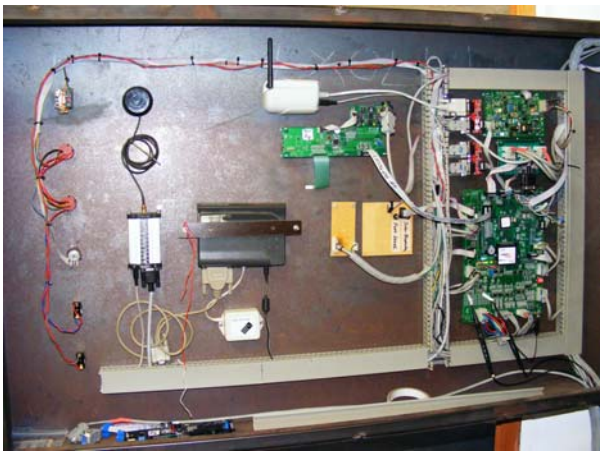


Figure 25 - Inside of the test rig's cabinet door

The central DSP board and interface board are on the right, while the HMI board (green) is in the centre. Above it is the Wi-Fi module. On the left, two modems, one cellular and one PSTN, can be seen. (Only one is connected at a time.) On the far left, some dials, buttons and warning lights are mounted.

This is the inside of the cabinet. The top left section is the power supply. The PWM driver boards are on the top right. It can be seen that there are two driver boards for each phase, for the top and bottom switch. In the centre, power components are mounted for each phase. Below are all the various breakers which stop malfunctions from being dangerous. The lead-acid batteries are at the bottom on the floor.

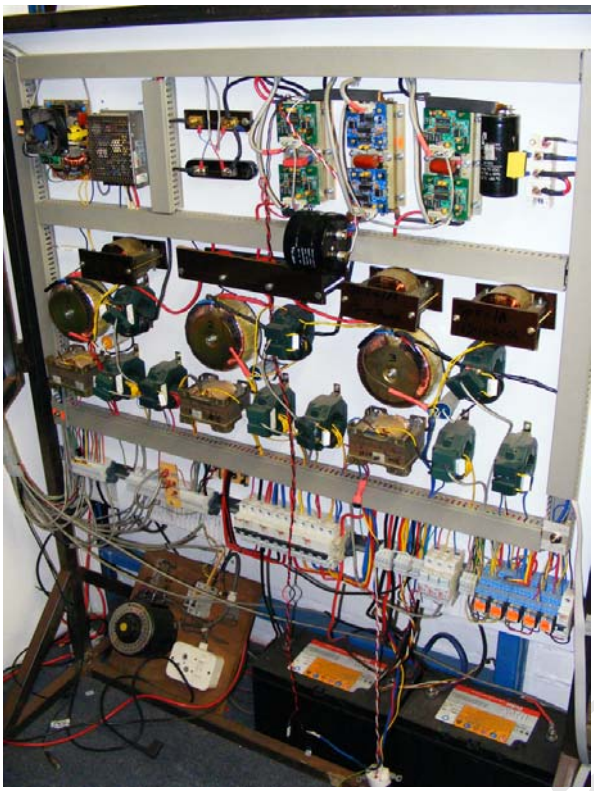


Figure 26 - Inside the test rig

Here the central DSP board together with the interface board can be seen in detail, connected to the test rig through its various connectors.

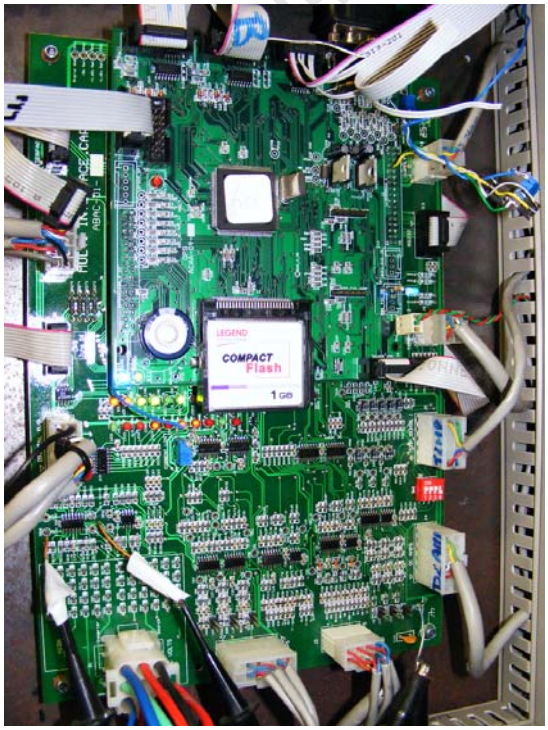


Figure 27 - Placement of the DSP and interface board inside the test rig

The test rig provided months of testing for developing hardware and software in a relatively real environment.

It can not however replace actual field testing, as aspects such as installation techniques and harsh environments can not always be predicted.

5.3.2. Central PCB and Interface Board

Figure 28 shows the central DSP board fitting onto the top side of the interface board. Its two 34-pin IDC inter-board connectors are aligned so it can plug directly onto the interface board.

The author played a minor part in the design of the hardware of the interface board for the hybrid system. Specifically the PIC processor on the board and its surrounding circuitry were designed by the author to conform to the standards used on the DSP board.

This interface board mostly handles the buffering of input and output lines, signal conditioning and filtering for analogue inputs. It has an array of debug LEDs for on-site debugging.

There are connectors riddled about the outside of the board for connecting all the sensors and peripherals. Most connectors are mini-fit type from Molex. There are also some ribbon cable connectors for connections to other printed circuit boards in the system like the isolated RS-232 level shifter.

The following diagram shows how the DSP board slots into the interface board:

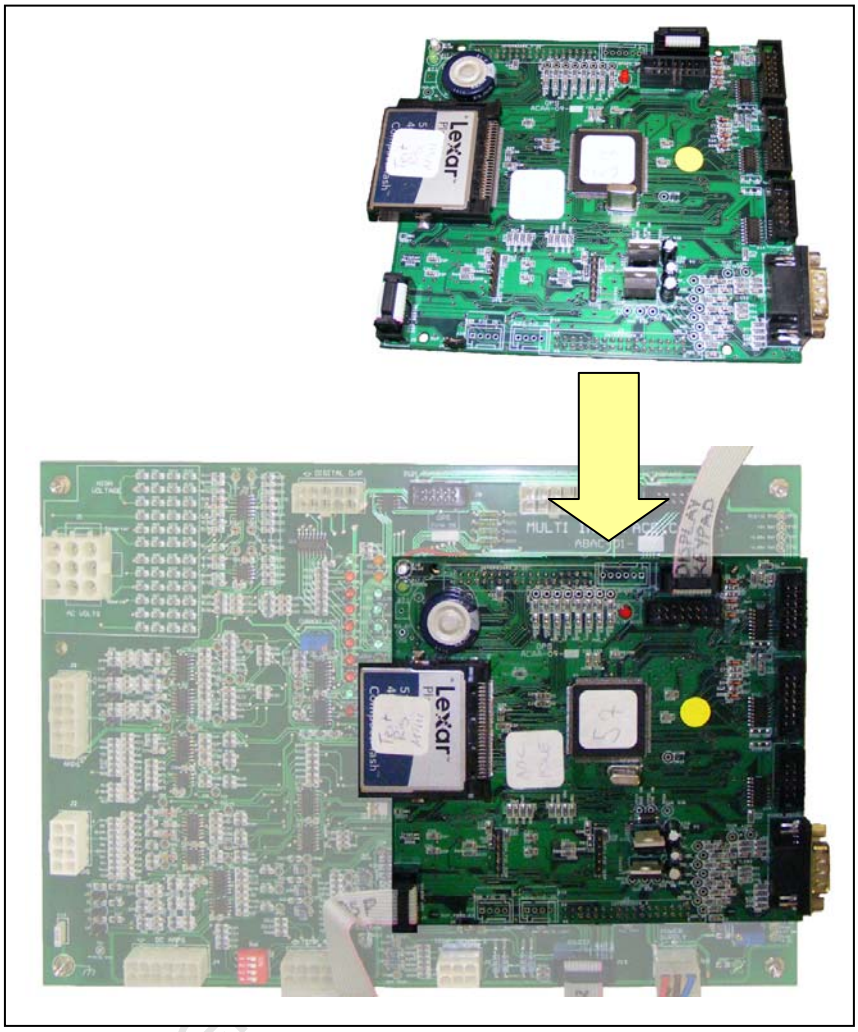


Figure 28 - Mating of the DSP and interface boards

5.3.3. Human Machine Interface

The design of the local human-machine interface (HMI) is out of the scope of this document. The HMI hardware is assumed to be laid out functionally as shown in Figure 29:

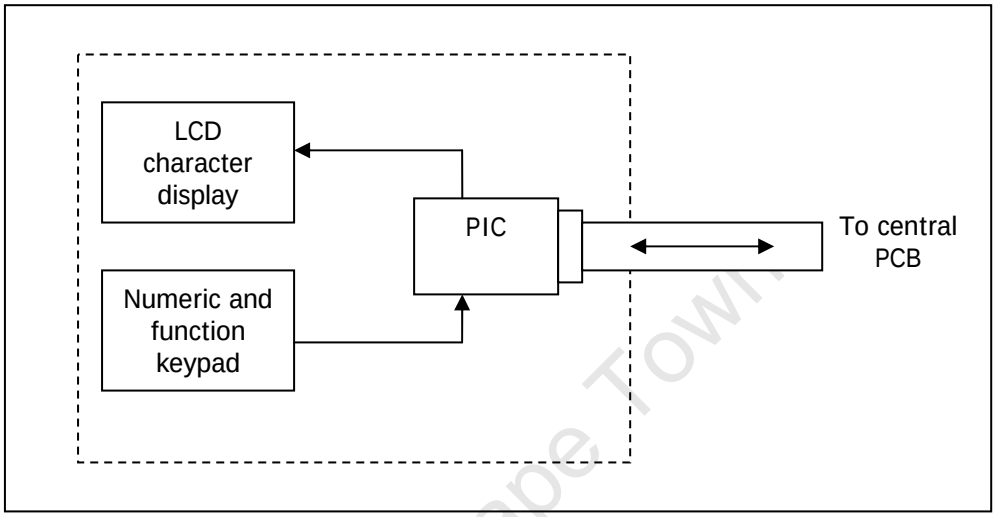


Figure 29 - Logical layout of the HMI

A photo of the HMI board in shown below in Figure 30:

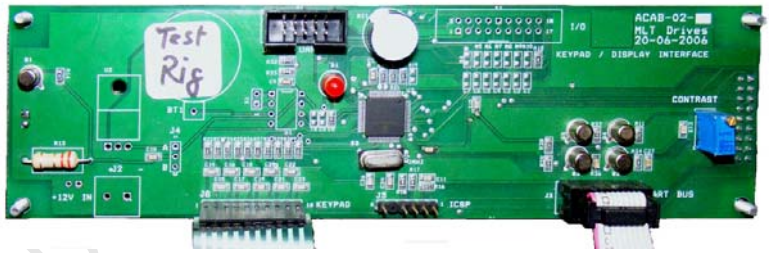


Figure 30 - Photo of the HMI board

On the photo, one can see the 10-pin IDC ribbon connector for connection to the DPS board. Amongst these pins are the inter-processor bus transmit and receive, and power.

Figure 31 shows a snapshot of the typical layout of the keypad and LCD screen on the front of a hybrid system.

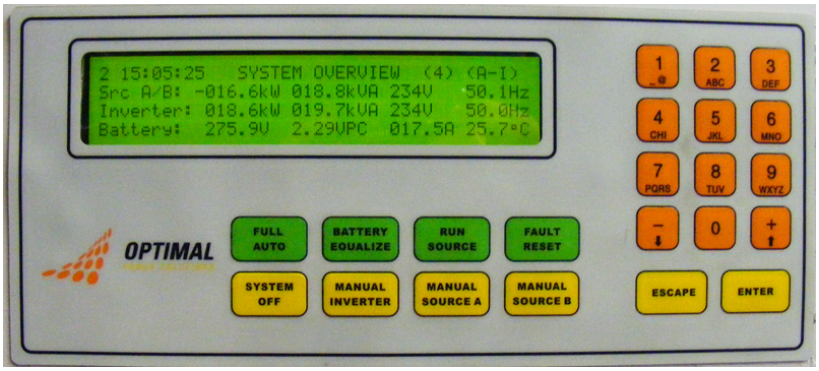


Figure 31 - Typical screenshot of the HMI

Some typical screen shots follow which show various readings and status values.

5.3.4. Power Supply

The power supply needs to supply several voltages with current protection at different levels. This supply provides power for the central PCB, the human-machine interface, the driver boards, and other peripherals such as the modem and the Wi-Fi serial bridge. The design of the power supply is out of the scope of this document.

The specifications are detailed in Table 3 here for reference:

Voltage	Current Limit	Maximum peak to peak ripple allowed at current limit
+12.0V	1A	200mV
–12.0V	1A	200mV
GND	-	-
+5.0V	2A	100mV

Table 3 - Power supply specifications

All other operating voltages will be generated by the specific printed circuit boards that require them.

5.3.5. Driver Boards

Again, this section is out of scope and it can be assumed that driver boards interface with the PWM signals to drive the inverter's switches.

Figure 32 is a photo showing the top and bottom driver boards for each phase in the test rig:

Figure 32 - Photo of driver boards

5.4. Metrics

Several metrics have to be defined in order to measure the performance of the system.

5.4.1. Speed

Several measurements of speed can be taken. Some of which are:

- Emergency button response
- Synchronising the inverter to the grid
- Power driver error response time
- Speed of remote programming

5.4.2. Noise Immunity

The ability of the system to continue to run smoothly in electrically noisy environments is very important in power systems as the switching signals for the power drivers inject much noise in any circuit close to them. The DSP board's four-layer design will be put to the test here. The DSP board's analogue inputs need to also be noise immune.

5.4.3. Ease of Use

With all the complicated software running on the system, it is a challenge to make the end user's interface easy to use. There is a compromise between customisability and simplicity. There are some products similar to this application which are so difficult to setup and use that advanced training needs to be done in order to operate the system. This raises the cost, and the accessibility.

5.4.4. Remote Monitoring and Control

The system must be able to be monitored remotely. This means that someone or some software must know exactly what the system is doing at

any point. The system must be able to store logs of any events of interest, be it faults, or normal running events. These logs must be sufficiently detailed and regular such that someone who is trying to find a fault in a system will have enough information to pinpoint the problem.

The system must be able to be controlled remotely as well. This means that commands can be issued to the system that would produce the same results of that of someone interfacing directly with the system on site.

5.4.5. Autonomous Operation

The design philosophy included a certain degree of robustness and autonomy. Therefore the ability of the system to continue to run normally after something unexpected happens is very important. This autonomy has been implemented mostly in software. Examples are:

- Automatic retry when the inverter fails to synchronise to the grid for any reason.
- Automatic reconnect on many different over-current faults.
- Fully automatic restart

Although many of the autonomous features are implemented in software outside of the scope of this document, they are all made possible by the flexible hardware, powerful processor, and reliable communications.

5.5. Performance

Once several hybrid systems were commissioned over the world, it was a good opportunity to test the various aspects of performance in the real world.

5.5.1. External Communications

Systems in countries such as Malaysia and India have been successfully remotely controlled and monitored now for several months. Modem dial-

up connections to the systems were possible and the user interface program was able to retrieve data logs and send commands successfully.

In another installation, a connection was made from the hybrid system to the local Wi-Fi network. This network was then connected to the internet. Through the internet, the system was able to be controlled remotely.

Although for both connection types, the data transfer rate was low compared to that of today's standards, the concept worked.

Typical performance figures are listed here:
Using the Wi-Fi connection locally, a maximum transfer rate of 3 data logs per second was achieved, (each data log consists of 512 bytes), while over a modem, about 1 log per second was achieved. These relatively low speeds can be attributed to having to use the MODBUS protocol, and having to transport information across the inter-processor bus.

5.5.2. Internal Communications

The internal communications, namely the inter-processor bus, perform well. The bus runs on UART at 115200 bits/s. Thanks to using high-priority interrupts in software, the DSP can poll all 4 slave processors in 10ms. Due to the abundance of speed, the bus was limited to 15ms between polling the same slave device. This gives the slave devices time for background processes.

The interface board's processor sends current and voltage data to the DSP regularly. Without loading the bus too much, this processor can send this information approximately 10 times per second.

The bus is able to recover from any number or errors. This was tested by using a very long cable to the human machine interface. This cable was routed through noisy switching devices. Using an oscilloscope, it was obvious that there was a tremendous amount of noise present. The DSP

detected the noise and continued to communicate just with the other slaves until the noise dissipated.

To program the DSP locally, custom PC software was written as discussed in section 4.5.7 - Bootloaders for Remote Programming on page 53. The process of programming the DSP locally was improved by about 10s from a typical 90s program time by using a custom bootloader. The reliability of this process also increased four-fold. Typically, every fourth attempt to program the DSP was successful with the default bootloader, and now it is up to 100%.

5.5.3. General

The general performance in terms of power management has been reported to be more than satisfactory by customers and testers in the project team. Unfortunately as the scope of this document does not cover the management of power, the exact figures will not be included. However, it is safe to say that the under-lying infrastructure developed is adequate to allow the power management to perform well.

6. Conclusions

It is the opinion of the author that the new architecture developed is a viable platform for the hybrid system project. All the objectives in the terms of reference have been delivered upon.

The following noticeable improvements over the old system can be observed:

6.1. Speed

The processing power of the new system is far greater than that of the old, namely almost 4 times. This means the new system can calculate readings like RMS values far quicker than the old. Calculations like this form a significant part of the processors job of controlling inverters. Often crude approximations for RMS values are used, thus affecting accuracy and performance. Now the proper calculations can be done, and in faster times so the control algorithms can run faster and react quicker to short circuits, and high currents. The improvements in the analogue to digital converter's speed are also a major advantage in the new system and match the increases of the processing power adequately.

The increased power also leaves room for many peripheral functions like real-time logging of more than 400 readings at regular intervals without affecting critical tasks.

6.2. Noise Immunity

The noise immunity of the new board would seem to be greatly improved. It has been tested in harsh environments where power stage switching noise is very strong, so much so that input pins receive spurious inputs and communications channels receive errors. Yet, the processor has continued to run, accommodating for all spurious inputs with filtering, and error correction on its inter-processor communication bus. In this situation the previous system has been observed to freeze and/or reset, which interrupts the power to the load.

6.3. Reliability

As a result of being noise immune and being able to predict faults, the new system has been proven to be far more reliable. It has been left running an inverter supplying power to a 3kVA load for over a month without an interruption in power.

In the field, the new system is more reliable than the old because it can be monitored by both the customer and manufacturer. Any heat problems, over currents, imbalanced loads or abuse can be detected and stopped before they cause faults and downtime.

In addition, the reliability of the production of the central DSP board has proven a success. To date[4], 200 DSP boards have been manufactured and tested. The layout of the DSP board lends itself to self testing which leads to increased reliability. Many hybrid systems have been commissioned in several countries.

6.4. Ease of Use

Mostly due to the user interface software written by Richard Parry, the new system is far easier to control and far easier to see its status than the old. This ease is amplified by being able to dial into any system from any computer in the world with a modem.

The human machine interface (HMI) designed by Stanley Adams has provided a larger set of functionality to the user on-site, without compromising on easy-of-use. This has been done by increasing the size of the display to clearly show information. Also a new keypad has been custom designed to provide shortcuts to common commands. Much of the configurability that should only be changed by the advanced user or manufacturers has been hidden from the user but is still available with password security.

[4] The date of printing was February 2008.

6.5. Remote Monitoring and Control

Once again, the user interface software written by Richard Parry (shown below in Figure 33) has played a large role in the remote monitoring and control. The remote control and monitoring conforms to industry standard protocol MODBUS. Already the system has been proven to be able to be controlled and interfaced with common proprietary control engineering software successfully.

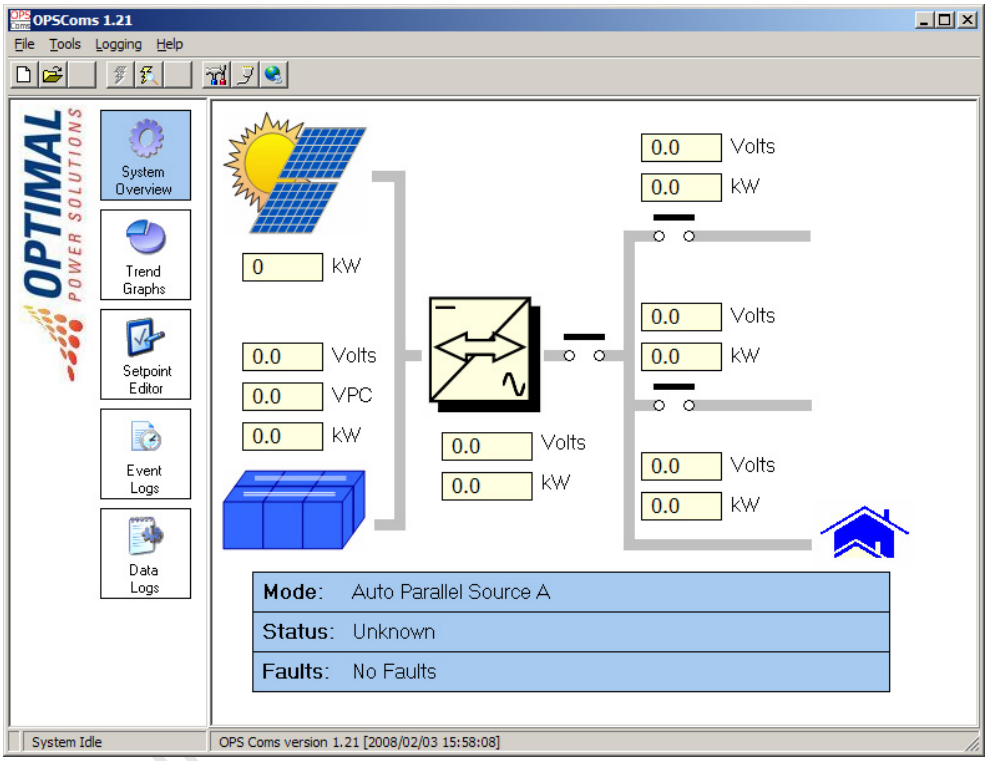


Figure 33 – A typical real-time screenshot of the user interface software

The new system can be controlled by anyone with a computer and a modem from anywhere in the world. Faulty equipment can now be debugged remotely and possibly even fixed remotely. Detailed logs of user-intervention, standard events and regular reading logs provide endless information about the system. These logs, depending on the size of the storage, can provide year's worth of information which can be graphed in the user interface software and trends can be observed.

Software can be upgraded remotely safely thanks to the storage on the compact flash card. Any potential software bugs or modifications can be done without incurring expensive travelling costs for engineers.

6.6. Intellectual Property Protection

Thanks to the security features and licensing system implemented for this system (as discussed in section 4.5.10 on page 63), it is possible to rest assured that it is extremely difficult to replicate illegal systems. As a commercial venture, the new hybrid system is made more safe and viable.

6.7. Expandability

The new system has been designed with the future in mind. No system's design stays static. The new system has sufficiently catered to future modifications and add-ins. The software, too, has been written so that many more communication devices can be attached to the inter-processor bus and thus interact with the DSP. The way that the PIC microcontrollers shared any common code makes improving software a clean documented process.

The DSP has many pins which are not used in this design, but have been routed to external connectors for future use. Not a single input/output pin is left unconnected.

6.8. Scalability and Cost

The detachable interface board makes it possible to develop different interface boards for different products or projects, thus saving cost in two ways:

1. The interface board can be developed and populated with only the necessary components and circuits.
2. The central DSP board is standard and does not need to be redesigned for each product or project.

The central DSP board has been designed in a modular way, such that certain modules (collection of components on the board) can be left unpopulated in production if not needed. The cost of the unnecessary components is then saved. The physical board has been designed to operate as either a 2-layer or 4-layer board to save costs on smaller systems.

It can thus be said that the DSP board and the associated interface board for each project can be customised to suite all systems from the smallest to the largest.

7. References

[1] Altera Corporation, *High-Speed Board Layout Guidelines*, 2005

[2] Bendel, C. Nestle, D: *Decentralized Electrical Power Generators in the Low Voltage Grid - Development of a Technical and Economical Integration Strategy*, Institut für Solare Energieversorgungstechnik (ISET), 2004

[3] Hudson, S. *Embedded Boot-loaders*, Texas Instruments Inc., 2002

[4] Baker, B. Curtis, K. *ETP-248: Managing Power, Ground, and Noise in Microcontroller/Analog Applications*, Microchip Technology Inc.

[5] Baker, B. *Hardware and Firmware Noise Reduction Techniques in Embedded Systems*, Microchip Technology Inc., 2005

[6] Baker, B. *Predict the Repeatability of Your ADC to the BIT*, Microchip Technology Inc., 2004

[7] Baker, B. *Select the Right Operational Amplifier for your Filtering Circuits*, Microchip Technology Inc., 2003

[8] Baker, B. *Techniques that Reduce System Noise in ADC Circuits*, Microchip Technology Inc., 2004

[9] *ETSI TS 101 356 v7.2.0*, 2001

[10] Modicon, Inc., Industrial Automation Systems. *Modicon Modbus Protocol Reference Guide (Rev. J)*, 1996

[11] Microchip Techonolgy Inc. *Analog-to-Digital Converter Design Guide*, 2004

[12] Compact Flash Association, *CF+ and CompactFlash Specification Revision 3.0*, 1994

[13] SanDisk Coporation, *SanDisk CompactFlash Memory Card Product Manual (Version 11.0)*, 1996

[14] Microsoft, *FAT32 File System Specification (Version 1.03)*, Microsoft Extensible Firmware Initiative, 2000

[15] DPAC Technologies Corp. *Airborne Wireless LAN Node Module Data Book*, 2004

[16] Infineon Technologies, *EMC Design Guideline for Microcontroller Board Layout*, 2001

[17] Texas Instruments, *TMS320F2812 EzDSP Development Kit Documentation*, 2003

[18] Hunter, I. *Interfacing TMS320C20x DSPs to Intel 28Fxxx Flash Memories*, Texas Instruments, 1999

[19] Crankshaw, J. *Understanding the TMS320F240 External Memory Interface*, Digital Signal Processing Solutions, 1998

[20] Hermandez, M. *CompactFlash Memory Card Interface to the TMS320VC54x*, Texas Instruments, 2001

[21] Castille, K. *TMS320C6000 EMIF to External Flash Memory*, Texas Instruments, 2002

[22] Kvasnicka, M. *Interfacing SPI Serial EEPROMs to Microchip PICmicro Microcontrollers*, Microchip Technology Inc., 2004

[23] Texas Instruments, *Data Manual for the TMS320F2812 DSP*, 2005.

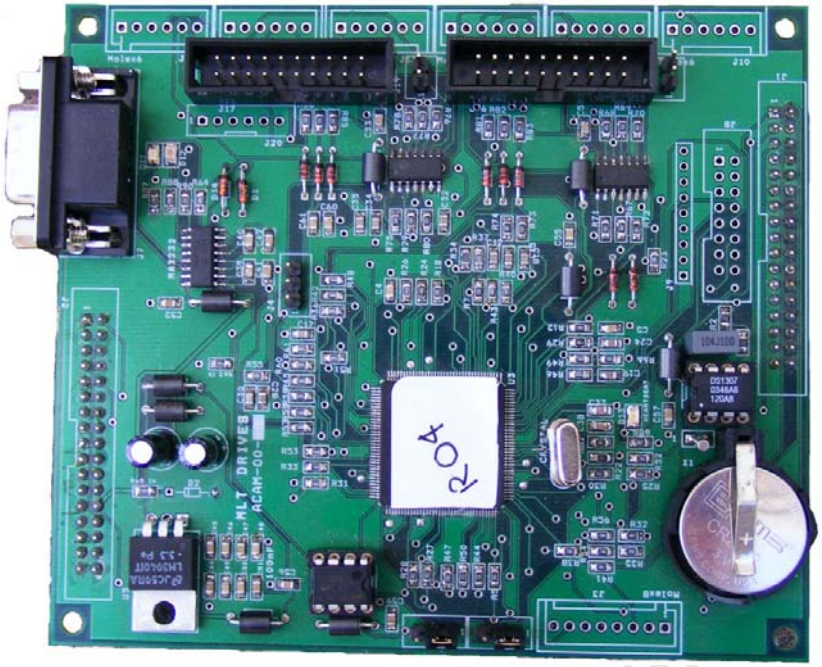
[24] Mauch, K. Ayoub, J. Jacquin, P. *PV Hybrid Systems within Mini Grids – IEA PVPS Task 11.* 3rd European Conference PV-Hybrid and Mini-Grid, Aix-en-Provence, 2006.

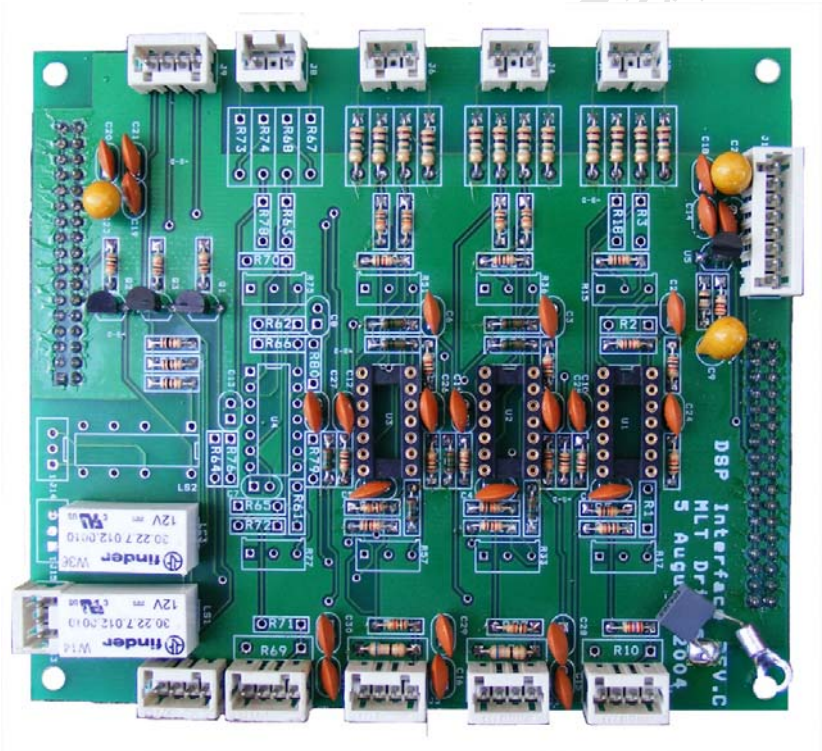
[25] The World Bank Group – Energy Unit, Energy Transport & Water Department. *Technical and Economic Assessment of Off-Grid, Mini-Grid and Grid Electrification Technologies Summary Report.* 2006.

[26] Croxford, B. Rizig, M. *Is Photovoltaic Power a Cost-Effective Energy Solution for Rural Peoples in Western Sudan?* 2006.

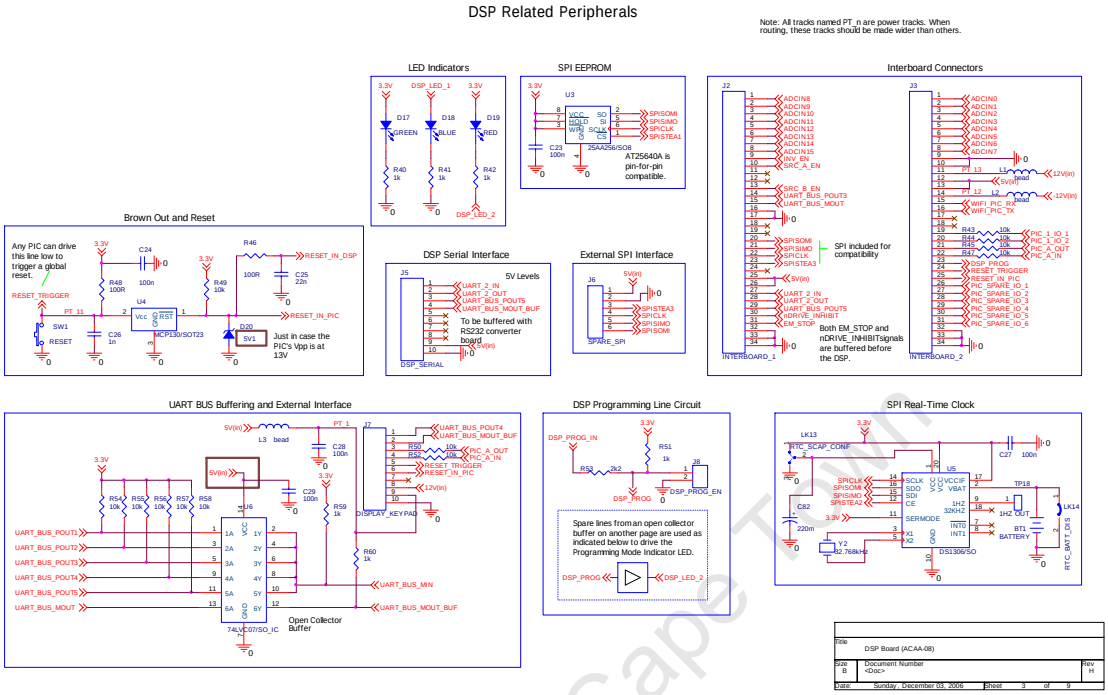
[27] Ketjoy, N. Rakwichian, W. Nathakaranakule, S. *Mini-Grid Concept for Rural Electrification in Thailand.* Post 2004.

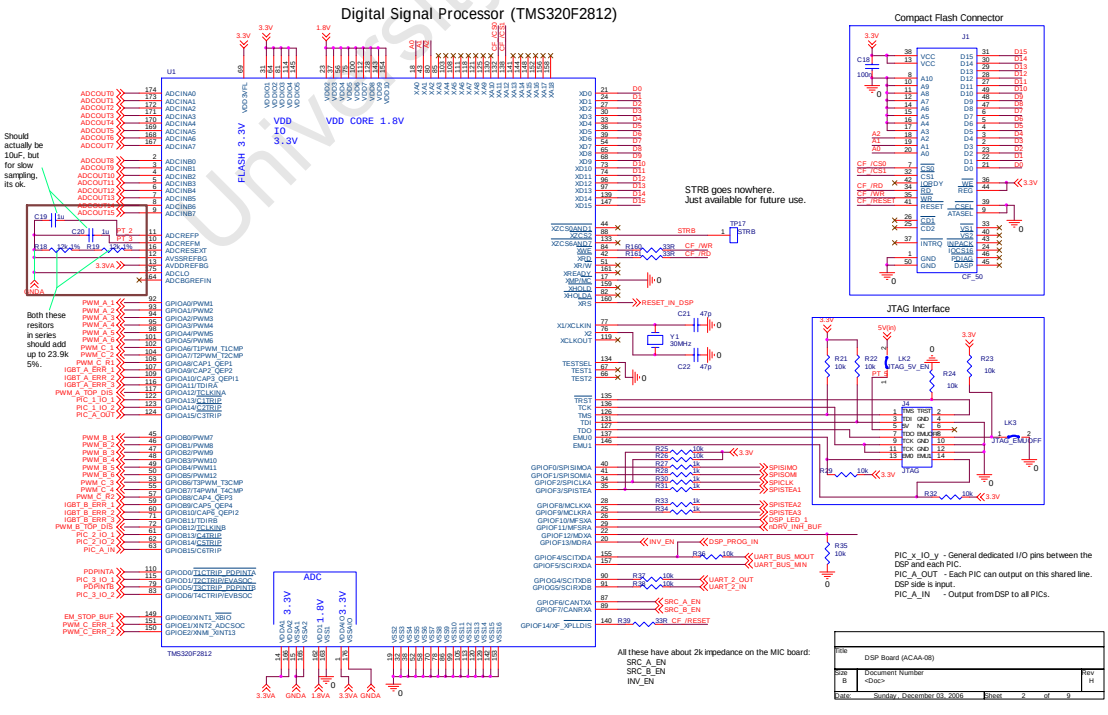
8. Appendix I – Existing DSP Board Photos

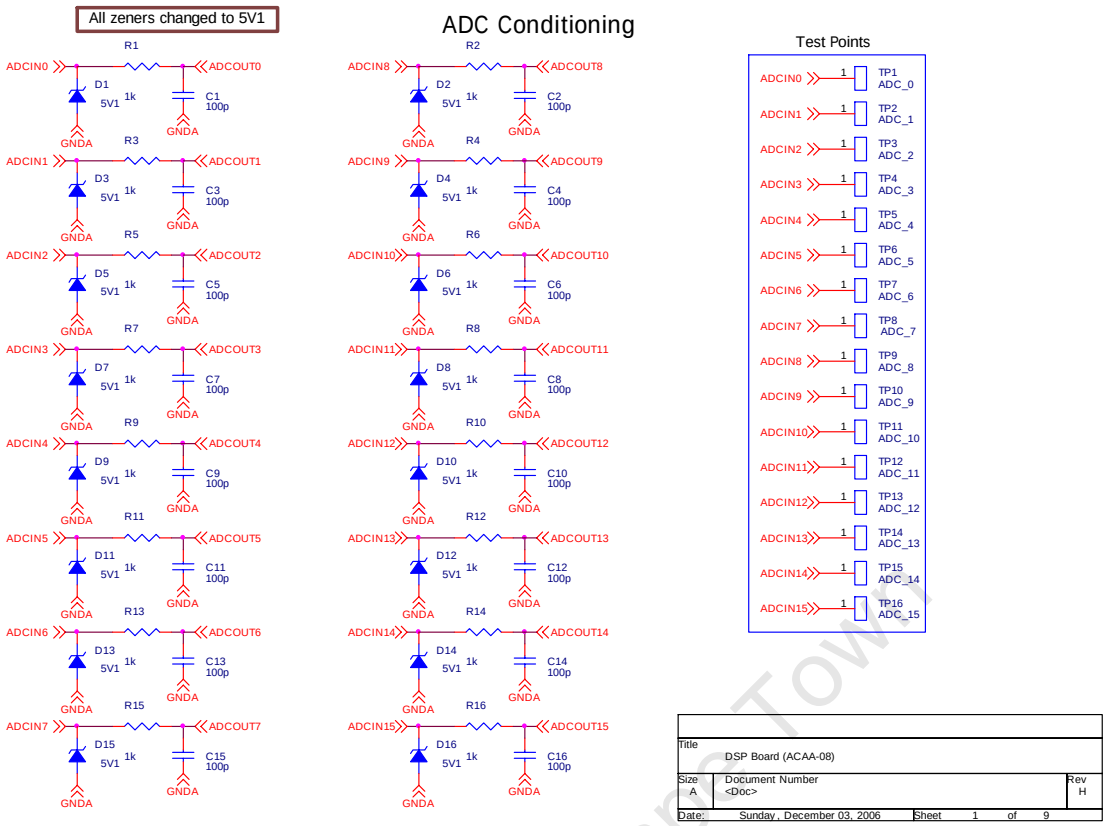


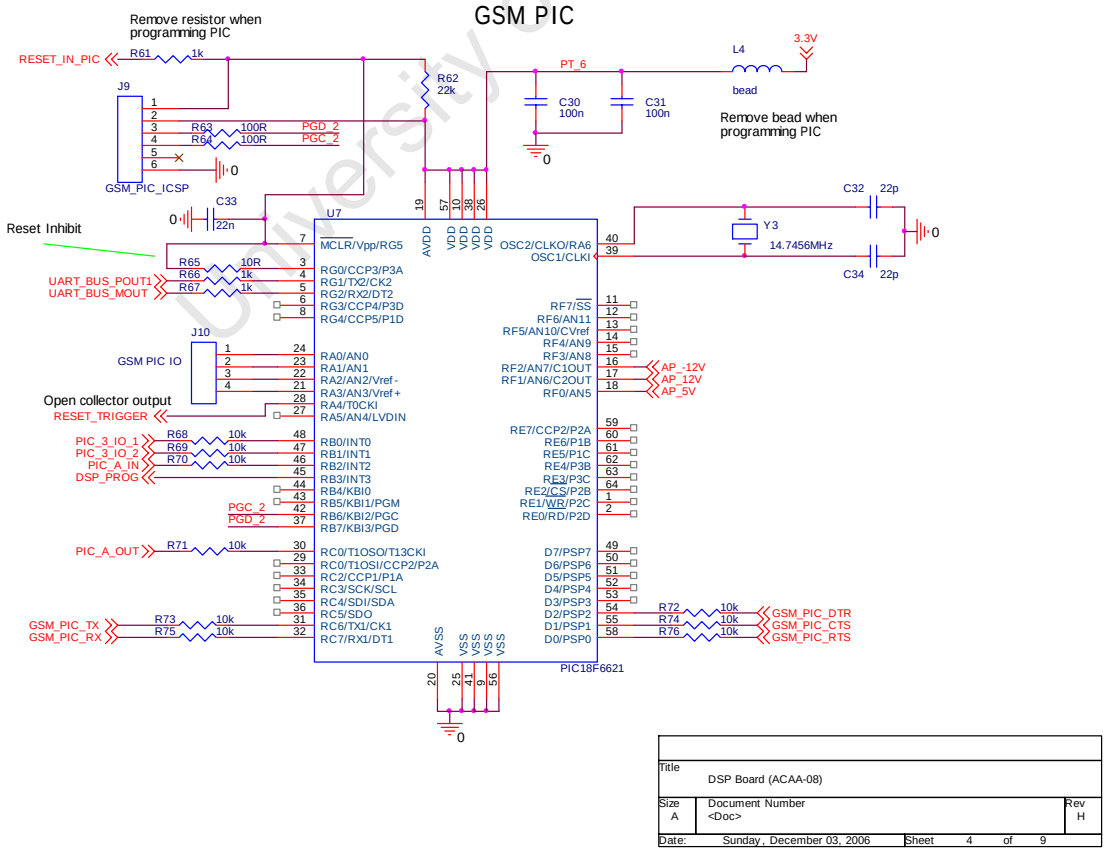


9. Appendix II – New DSP Board Schematics

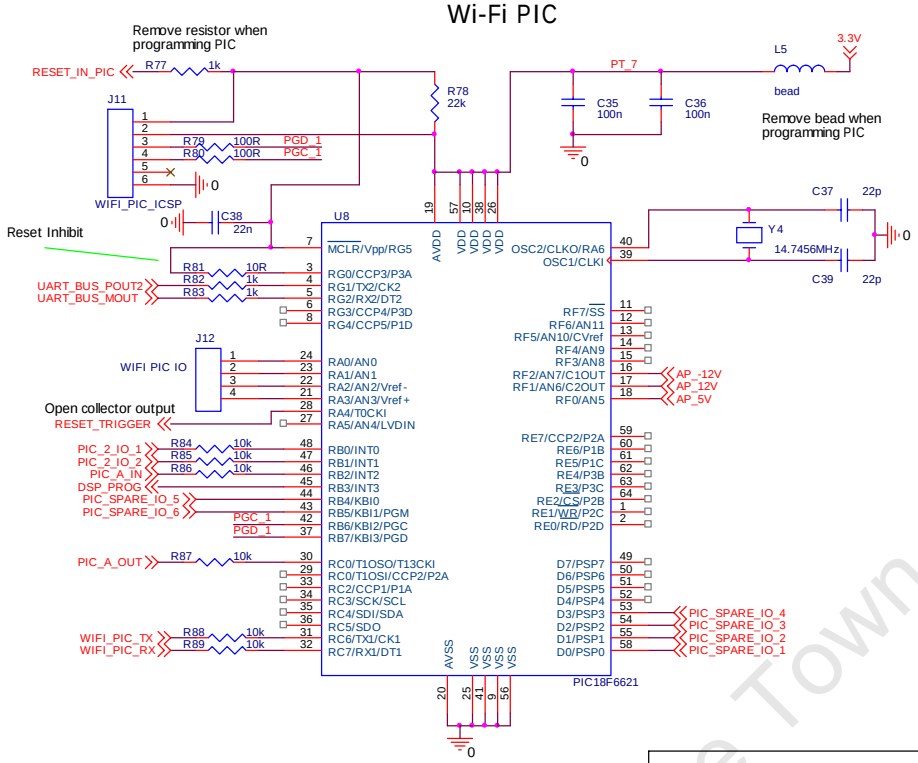








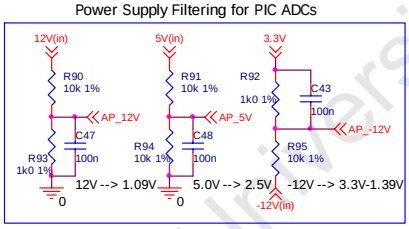
Wi-Fi PIC



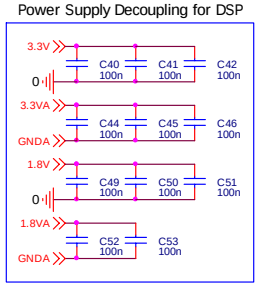
Title	DSP Board (ACAA-08)		
Size A	Document Number <Doc>		Rev H
Date: Sunday, December 03, 2006		Sheet 5 of 9	

Power Supply

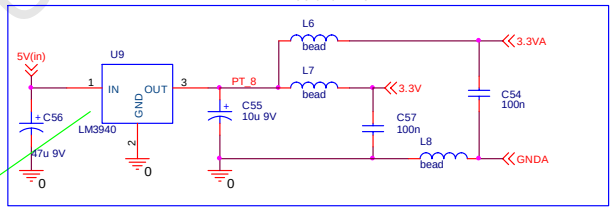
Power Supply Filtering for PIC ADCs



Power Supply Decoupling for DSP

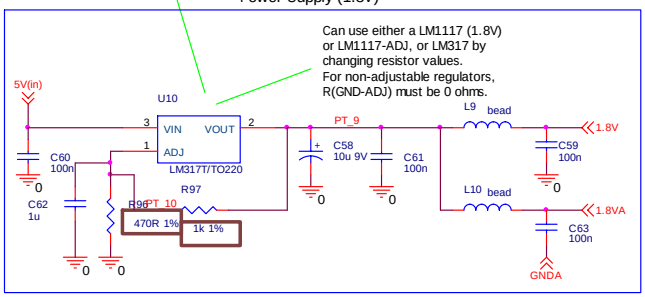


Power Supply (3.3V)

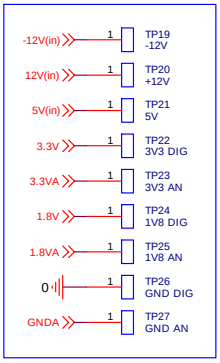


Both the 1.8V and 3.3V regulators should be placed close to each other so they can share the supply capacitors.

Power Supply (1.8V)

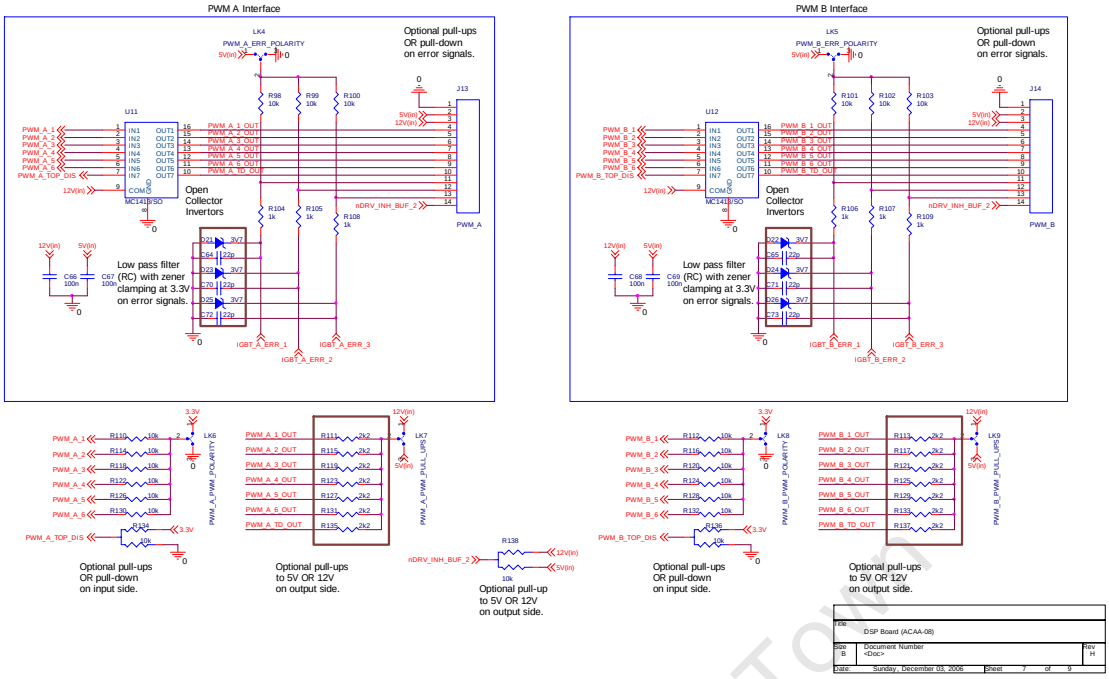


Test Points

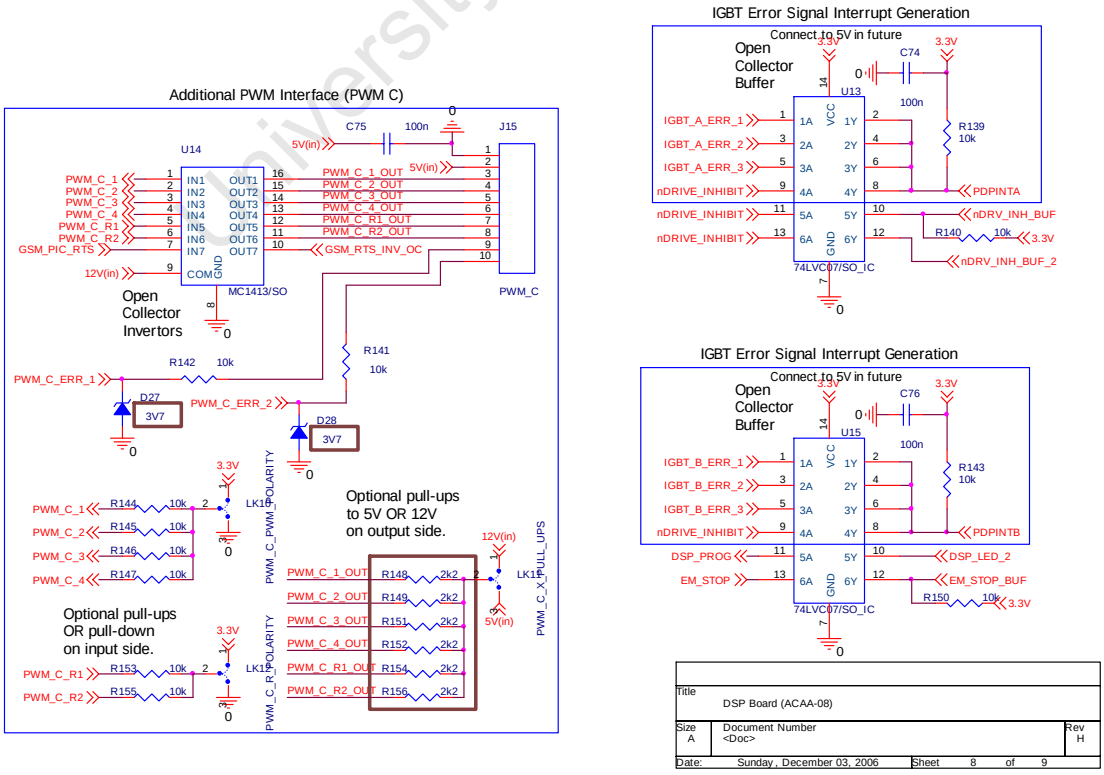


Title	DSP Board (ACAA-08)		
Size A	Document Number <Doc>		Rev H
Date: Sunday, December 03, 2006		Sheet 6 of 9	

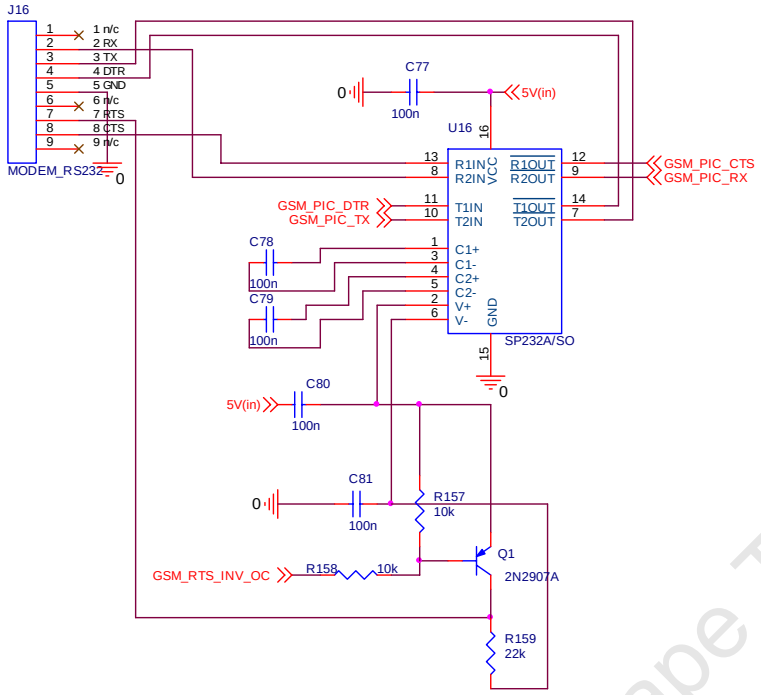
PWM A and B Interfaces



PWM C Interface and IGBT Errors



RS-232 Buffering of Modem Interface



Title	DSP Board (ACAA-08)		
Size A	Document Number <Doc>		Rev H
Date: Sunday, December 03, 2006		Sheet 9 of 9	

10. Appendix III – Paper on the Inter-Processor Bus

A Light-Weight Inter-Processor Network-Layer Protocol over a UART Bus

Tristan Phillips (BSc Eng) 2008, University of Cape Town

10.1. Abstract

This paper describes a protocol at the network layer which can be used to create an inter-processor bus between several devices, connected through their standard universal serial asynchronous receivers and transmitters. The focus is on the method in which the flow of data is controlled, and how errors are handled on higher protocol layers.

10.2. Problem

A fairly generic problem is that of connecting several microcontrollers on a printed circuit board (PCB) together in terms of communications. Several established methods exist already, but each has one or more disadvantages.

Some of these methods are:

- SPI
- I^2C

The disadvantages of using SPI in this scenario are: Firstly, three or even four physical lines are needed, compared to many other solutions that need only two. Secondly, it has a master/slave arrangement where the master must send data and receive data at the same time. The master always solicits data from the slave. Thirdly, in some cases where multiple slaves are present, a dedicated chip select line is needed for each slave.

The disadvantages of using I^2C similarly are: Firstly, it is more prone to noise, and thus limited in how far the physical bus can reach in distance. Secondly, as with SPI, data from a slave is always solicited by a master. At least with I^2C, the master can issue separate read and write commands.

It is apparent that both the above busses are designed to interface fixed-function, non-programmable devices with microcontrollers, rather than provide communications between microcontrollers. The greatest problem faced with using these busses to provide communications between microcontrollers is that the response time of slaves is not fixed. For example, when a master sends a request with either bus type, it sends the request data. Then after a fixed time (determined by the datasheet of the device), the master interrogates the slave for an answer. Since microcontrollers can not be expected to have a constant response time on busses, one has to give enough time for the slowest possible response to have reliable communications.

The author of this paper was faced with such a problem when trying to interface four microcontrollers with a central digital signal processor (DSP). If SPI or I^2C were to be used, the slowest response time was in the order of 20ms. This would place a severe damper on the speed of the communications if this response time was to be used as a round trip time.

10.3. Solution

Typically, a universal asynchronous receiver and transmitter (UART) is used on a physical point-to-point RS-232 line. The fact that it is normally point-to-point means that it does not provide any form of addressing as part of its specification.

In the next few paragraphs, a method of addressing devices and managing the flow of data using UART will be described.

The solution is to have a multiplexer on a UART bus that allows only one slave device to transmit data at any time, and to have a direct connection from the master's transmitter to all the slaves' receivers.

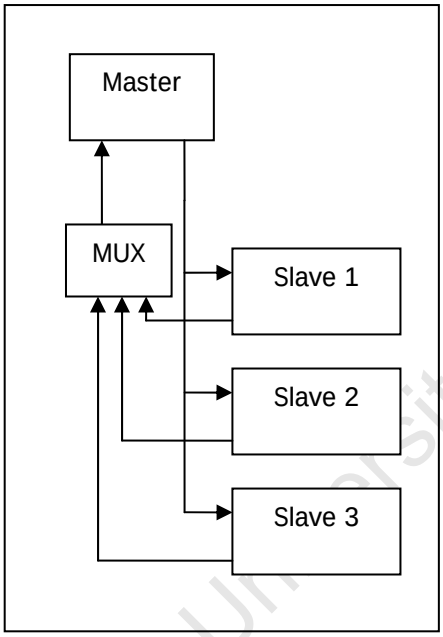
The management of data is done in a polling algorithm instigated by the master device.

10.3.1. Master/Slave Arrangement

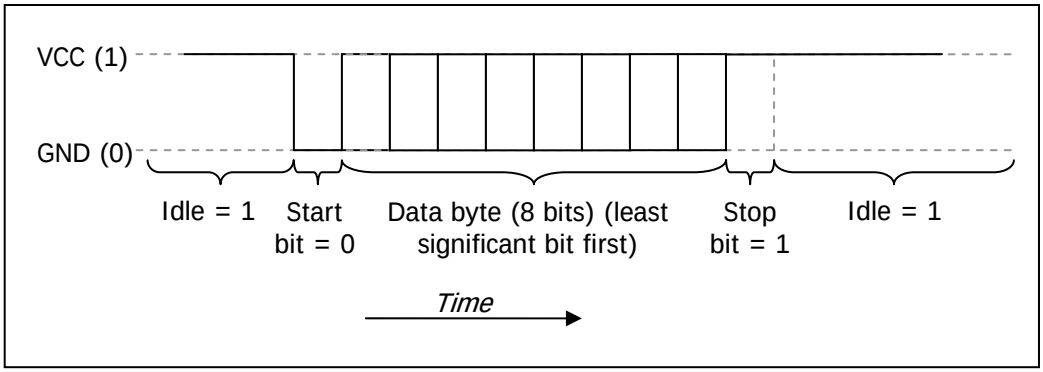
This solution is based on a master and slave arrangement. It is necessary to have one processor to manage the flow of data and poll slave devices.

10.3.2. Physical Layer

The following diagram shows the way the UART bus is connected in terms of data flow:



From the timing diagram below it can be seen that the transmission of a data frame on UART is triggered by a start bit which is always a low signal for the duration of exactly one bit. Following that, the bits that form the data are transmitted in from the least significant to the most significant. Usually (and in this case), 8 bits are used in a data frame to form a byte. A stop bit which is always a high signal is then sent for at least 1 bit duration. The line can then continue to idle by staying at a high signal.

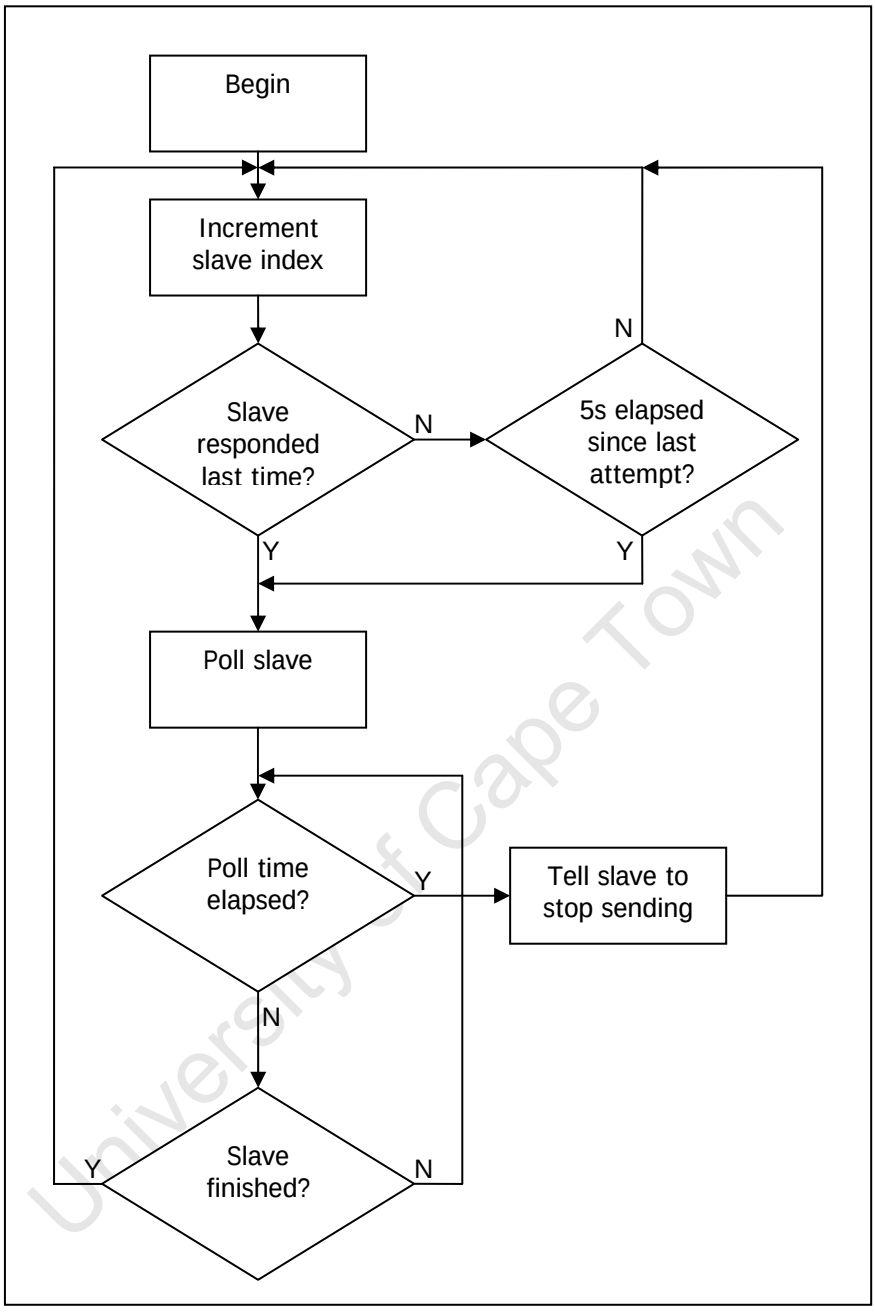


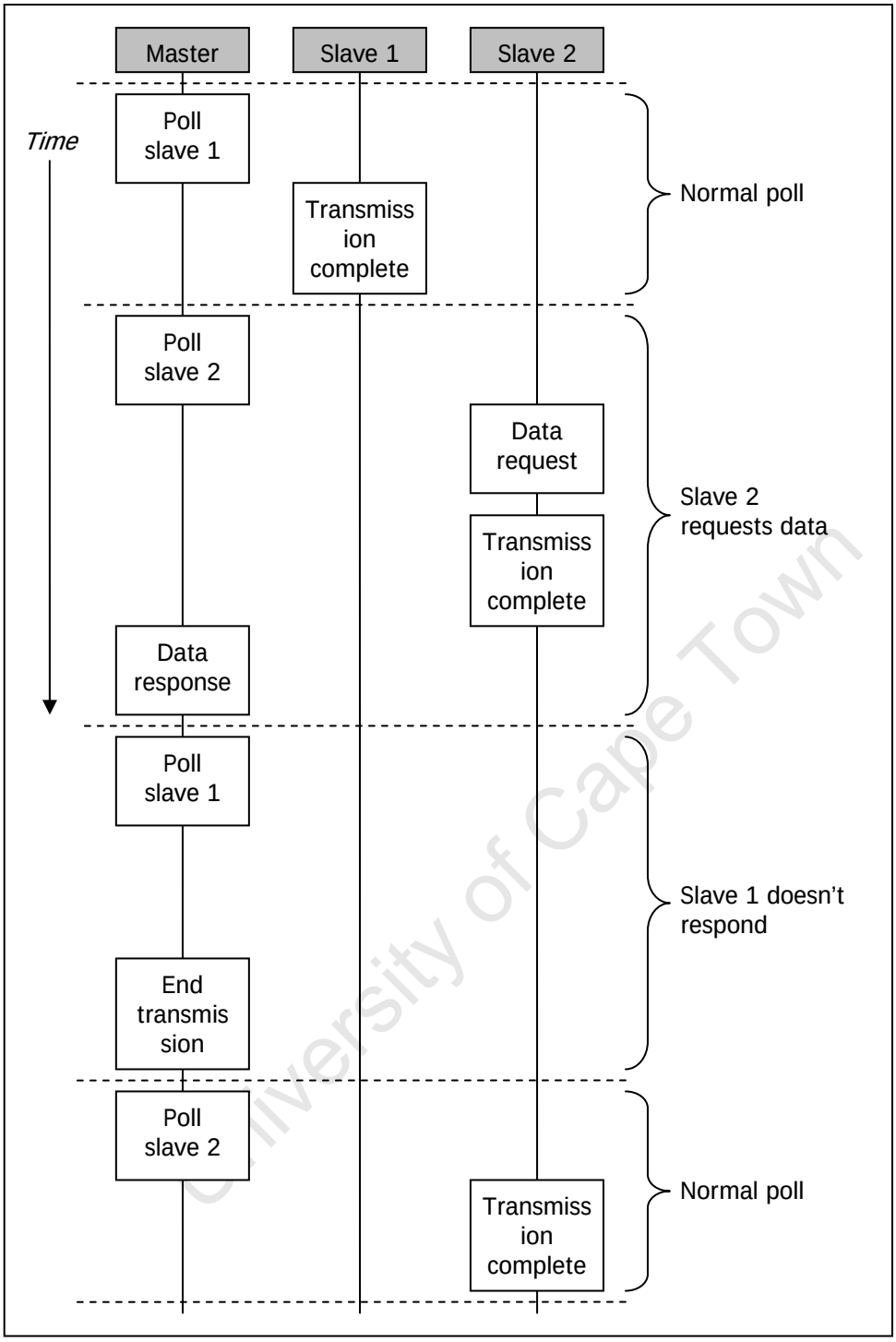
Because the line idles at a high signal, a simple multiplexer can by designed by having each slave's transmitter signal fed into an open-collector output buffer. The outputs of these buffers can be connected together and also to a pull-up resistor. In this arrangement, any one slave device can at any time pull the bus to a low level and send data. In terms of a logic circuit, the signals are all put into one large AND-gate. Care must be taken, of course, to not allow more than one slave device to pull this bus low at the same time. This task is up the manager of the flow of data.

The following logic table shows the various states of the multiplexer given different inputs:

x	***y***	***z***	***x* AND *y* AND *z***	**State**
0	0	0	0	illegal
0	0	1	0	illegal
0	1	0	0	illegal
0	1	1	0	slave *x* active
1	0	0	0	illegal
1	0	1	0	slave *y* active
1	1	0	0	slave *z* active
1	1	1	1	idle

10.3.3. Polling Algorithm





10.3.4. Performance

If one runs this bus at a rate of 115200b/s, and 1 stop bit is used, each frame takes up the time of 10 bits. The byte rate is then 11.52kb/s. The typical time between polls, in an arrangement of 4 slave devices would then be:

$$t = \frac{d}{r} = \frac{(4 \times 2)bytes}{11520bytes \cdot s^{-1}} = \frac{8}{11520} s = 694\mu s$$

It can be assumed that one byte is sent to poll a slave, and one byte is sent to acknowledge the poll, indicating that there is no data to send.

10.3.5. Scalability

The way the protocol is structured, the design is scalable up to 1 master and 6 slaves, but this restriction is purely based on the arrangement of address bits in the protocol. The protocol can be modified to support much larger arrays of slaves. The speed of the UART bus is also not restricted to 115200 bits/s, it can be scaled upwards from there. The only disadvantage to using speeds above the mention rate is that debugging from a conventional PC serial port will not be possible.

10.4. Conclusion

The solution provided here to the generic problem described in this paper adequately addresses the needs of microcontroller design. It is a very simple protocol with a simple polling algorithm and therefore fairly easy to code in software. It is also not very processing intensive, so precious processing time can be used for important tasks in each microcontroller. With error detection like cyclic redundancy checking (CRC) built into the higher layers of the protocol, errors caused by slow processors or by noise can be tolerated. The typical response times between polling a slave device are fairly fast, but scale inversely according to the number of slaves.

A Light-Weight Inter-Processor Network-Layer Protocol over a UART Bus

Tristan Phillips (BSc Eng) 2008, University of Cape Town

Abstract

This paper describes a protocol at the network layer which can be used to create an inter-processor bus between several devices, connected through their standard universal serial asynchronous receivers and transmitters. The focus is on the method in which the flow of data is controlled, and how errors are handled on higher protocol layers.

Problem

A fairly generic problem is that of connecting several microcontrollers on a printed circuit board (PCB) together in terms of communications. Several established methods exist already, but each has one or more disadvantages.

Some of these methods are:

- SPI
- I^2C

The disadvantages of using SPI in this scenario are: Firstly, three or even four physical lines are needed, compared to many other solutions that need only two. Secondly, it has a master/slave arrangement where the master must send data and receive data at the same time. The master always solicits data from the slave. Thirdly, in some cases where multiple slaves are present, a dedicated chip select line is needed for each slave.

The disadvantages of using I^2C similarly are: Firstly, it is more prone to noise, and thus limited in how far the physical bus can reach in distance. Secondly, as with SPI, data from a slave is always solicited by a master. At least with I^2C, the master can issue separate read and write commands.

It is apparent that both the above busses are designed to interface fixed-function, non-programmable devices with microcontrollers, rather than provide communications between microcontrollers. The greatest problem faced with using these busses to provide communications between microcontrollers is that the response time of slaves is not fixed. For example, when a master sends a request with either bus type, it sends the request data. Then after a fixed time (determined by the datasheet of the device), the master interrogates the slave for an answer. Since microcontrollers can not be expected to have a constant response time on busses, one has to give enough time for the slowest possible response to have reliable communications.

The author of this paper was faced with such a problem when trying to interface four microcontrollers with a central digital signal processor (DSP). If SPI or I^2C were to be used, the slowest response time was in the order of 20ms. This would place a severe damper on the speed of the communications if this response time was to be used as a round trip time.

Solution

Typically, a universal asynchronous receiver and transmitter (UART) is used on a physical point-to-point RS-232 line. The fact that it is normally point-to-point means that it does not provide any form of addressing as part of its specification.

In the next few paragraphs, a method of addressing devices and managing the flow of data using UART will be described.

The solution is to have a multiplexer on a UART bus that allows only one slave device to transmit data at any time, and to have a direct connection from the master's transmitter to all the slaves' receivers.

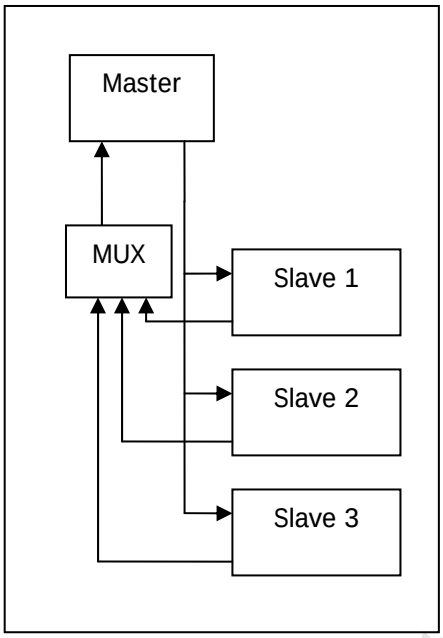
The management of data is done in a polling algorithm instigated by the master device.

Master/Slave Arrangement

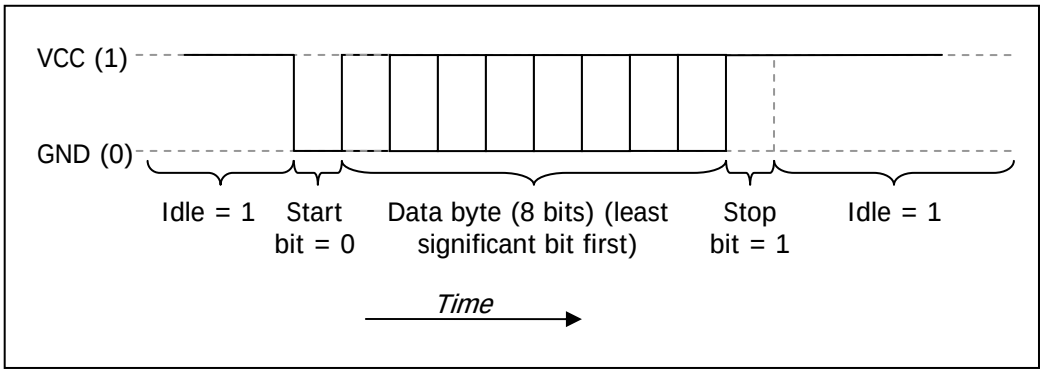
This solution is based on a master and slave arrangement. It is necessary to have one processor to manage the flow of data and poll slave devices.

Physical Layer

The following diagram shows the way the UART bus is connected in terms of data flow:



From the timing diagram below it can be seen that the transmission of a data frame on UART is triggered by a start bit which is always a low signal for the duration of exactly one bit. Following that, the bits that form the data are transmitted in from the least significant to the most significant. Usually (and in this case), 8 bits are used in a data frame to form a byte. A stop bit which is always a high signal is then sent for at least 1 bit duration. The line can then continue to idle by staying at a high signal.



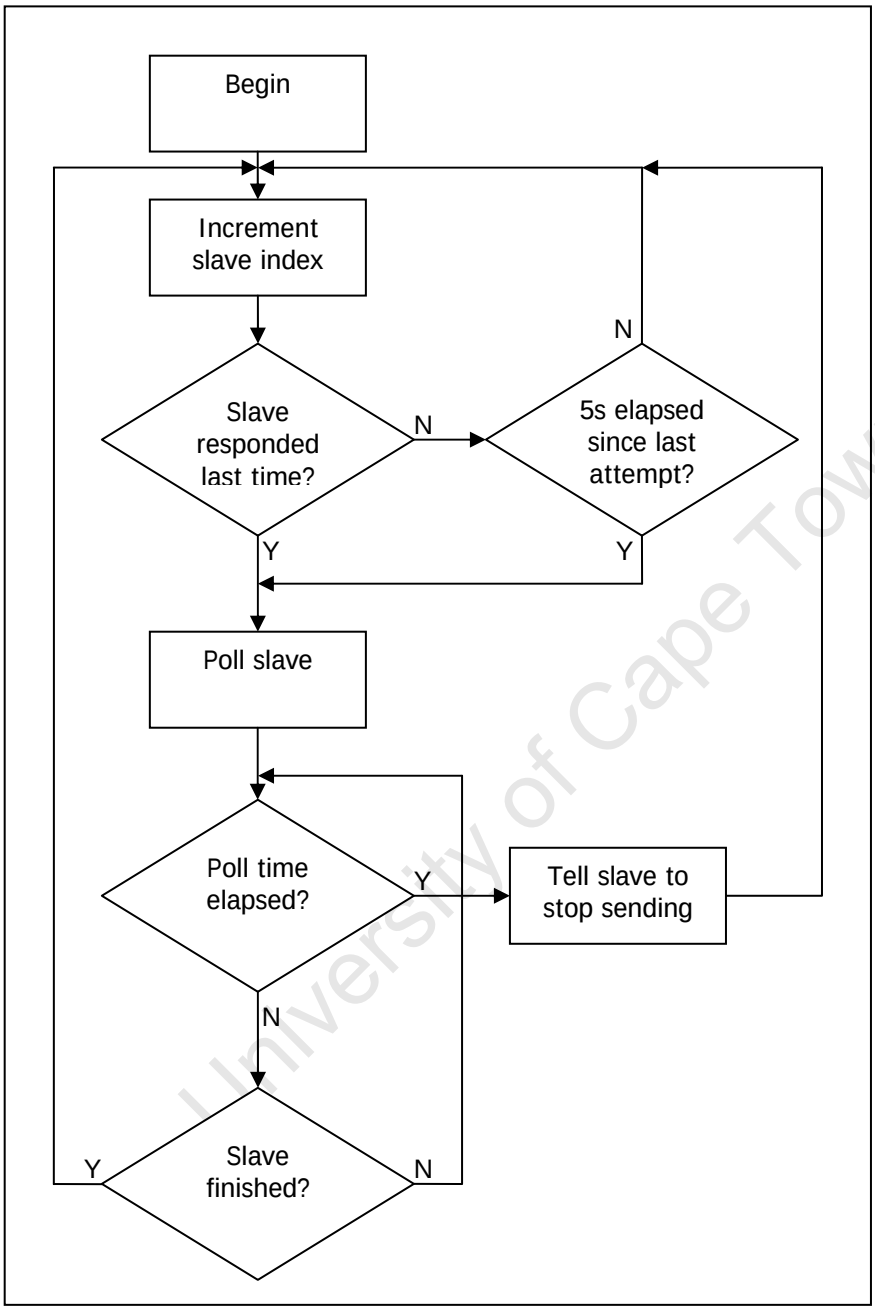
Because the line idles at a high signal, a simple multiplexer can by designed by having each slave's transmitter signal fed into an open-collector output buffer. The outputs of these buffers can be connected together and also to a pull-up resistor. In this arrangement, any one slave device can at any time pull the bus to a low level and send data. In terms of a logic circuit, the signals are all put into one large AND-gate. Care must be taken, of course, to not allow more than one slave device to pull this bus low at the same time. This task is up the manager of the flow of data.

The following logic table shows the various states of the multiplexer given different inputs:

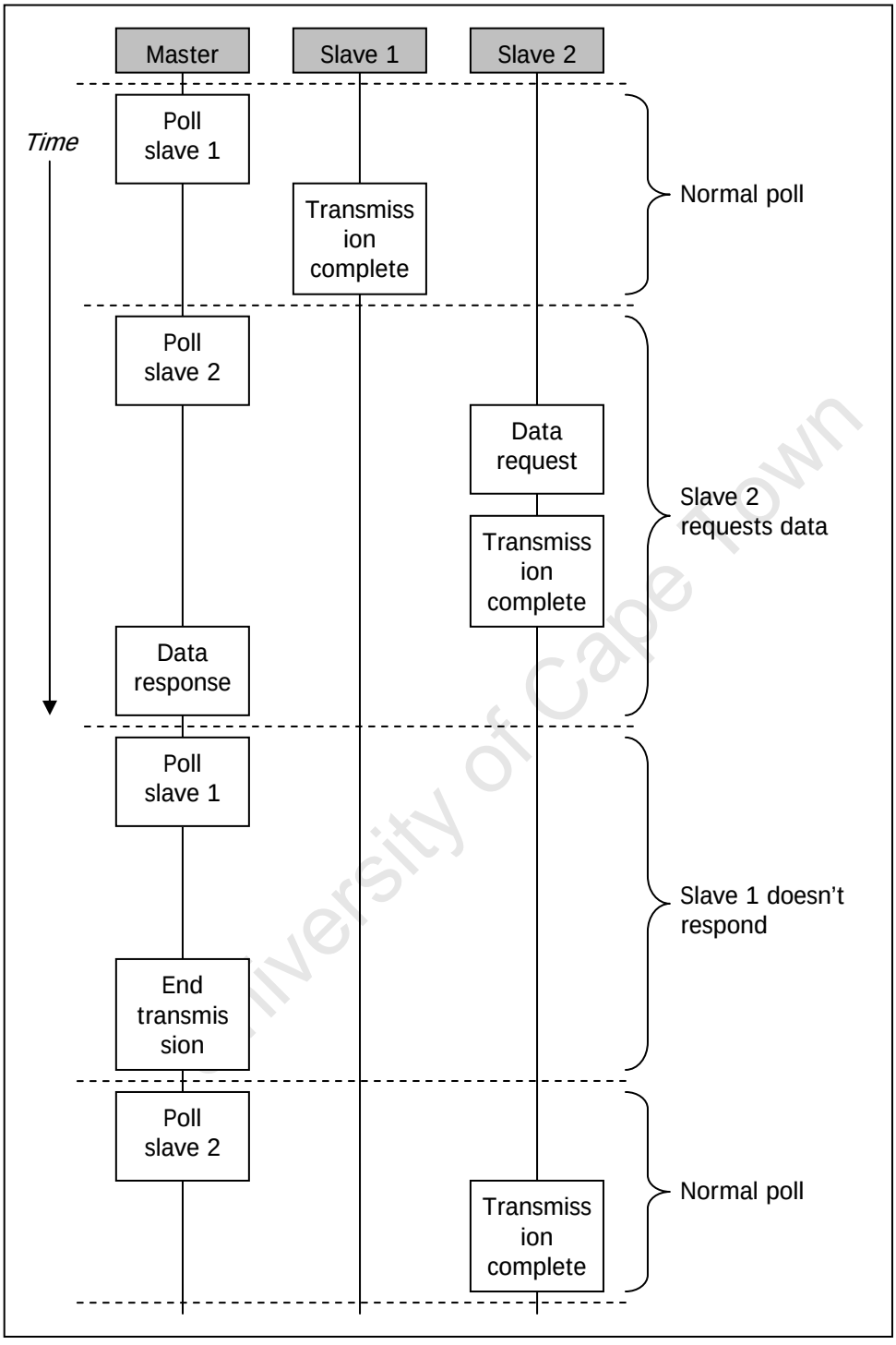
x	***y***	***z***	***x* AND *y* AND *z***	**State**
0	0	0	0	illegal
0	0	1	0	illegal
0	1	0	0	illegal
0	1	1	0	slave *x* active
1	0	0	0	illegal
1	0	1	0	slave *y* active
1	1	0	0	slave *z* active
1	1	1	1	idle

Polling Algorithm

The following flow diagrams shows the steps in the polling algorithm as in the master device:



The following message diagram shows typical interaction between a master and two slave devices:



Performance

If one runs this bus at a rate of 115200b/s, and 1 stop bit is used, each frame takes up the time of 10 bits. The byte rate is then 11.52kb/s. The typical time between polls, in an arrangement of 4 slave devices would then be:

$$t = \frac{d}{r} = \frac{(4 \times 2)bytes}{11520bytes \cdot s^{-1}} = \frac{8}{11520}s = 694\mu s$$

It can be assumed that one byte is sent to poll a slave, and one byte is sent to acknowledge the poll, indicating that there is no data to send.

Scalability

The way the protocol is structured, the design is scalable up to 1 master and 6 slaves, but this restriction is purely based on the arrangement of address bits in the protocol. The protocol can be modified to support much larger arrays of slaves. The speed of the UART bus is also not restricted to 115200 bits/s, it can be scaled upwards from there. The only disadvantage to using speeds above the mention rate is that debugging from a conventional PC serial port will not be possible.

Conclusion

The solution provided here to the generic problem described in this paper adequately addresses the needs of microcontroller design. It is a very simple protocol with a simple polling algorithm and therefore fairly easy to code in software. It is also not very processing intensive, so precious processing time can be used for important tasks in each microcontroller. With error detection like cyclic redundancy checking (CRC) built into the higher layers of the protocol, errors caused by slow processors or by noise can be tolerated. The typical response times between polling a slave device are fairly fast, but scale inversely according to the number of slaves.