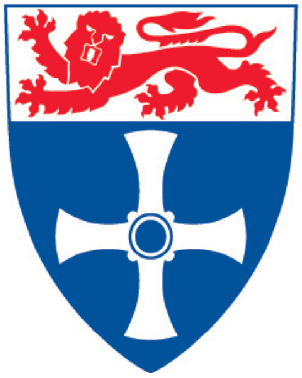

Adaptive Torque-Feedback Based Engine Control

Steven Clugston

SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Faculty of Science, Agriculture and Engineering,
Newcastle University.

May 2012

Abstract

The aim of this study was to develop a self-tuning or adaptive SI engine controller using torque feedback as the main control variable, based on direct/indirect measurement and estimation techniques. The indirect methods include in-cylinder pressure measurement, ion current measurement, and crankshaft rotational frequency variation. It is proposed that torque feedback would not only allow the operating set-points to be monitored and achieved under wider conditions (including the extremes of humidity and throttle transients), but to actively select and optimise the set-points on the basis of both performance and fuel economy. A further application could allow the use of multiple fuel types and/or combustion enhancing methods to best effect. An existing experimental facility which comprised a Jaguar AJ-V8 SI engine coupled to a Heenan-Froude Dynamatic GVAL (Mk 1) dynamometer was adopted for this work, in order to provide a flexible distributed engine test system comprising a combined user interface and cylinder pressure monitoring system, a functional dynamometer controller, and a modular engine controller which is close coupled to an embedded PC has been created. The considerable challenges involved in creating this system have meant that the core research objectives of this project have not been met. Nevertheless, an open-architecture software and hardware engine controller and independent throttle controller have been developed, to the point of testing. For the purposes of optimum ignition timing validation and combustion knock detection, an optical cylinder pressure measurement system with crank angle synchronous sampling has been developed. The departure from the project's initial aims have also highlighted several important aspects of eddy-current dynamometer control, whose closed-loop behaviour was modelled in Simulink to study its control and dynamic response. The design of the dynamometer real-time controller was successfully implemented and evaluated in a more contemporary context using an embedded digital controller.

Acknowledgements

This work has been funded by the School of Mechanical & Systems Engineering, the University of Newcastle-upon-Tyne.

I would like to thank my supervisor Dr Robert Bicker for his unwavering patience, support, and accessibility throughout this project. I would also like to extend my gratitude to Richard Burnett as through his interest and enthusiasm for all matters related to electronics I have learnt many things and who has always made time to discuss many of my electronic design issues when I have needed it on too numerous occasions to recall.

I would also like to thank Professor James Burdess for his insight into state-space control, and also Professors Alan Jack and Barrie Mecrow from the school of Electrical, Electronic and Computer Engineering for discussing dynamometer magnetisation with me.

Contents

1 Introduction **1**

1.1 Motivation for Research 2

1.2 Research Methodology 3

1.3 Reseach Objectives 5

1.4 Contribution and Research Roadmap 5

2 System Identification and Adaptive Control **9**

2.1 Self-Tuning Control (STC) 11

2.2 Model Reference Control (MRAC) 12

2.3 Model Based Predictive Control 14

2.4 System Identification and Parameter Estimation 16

2.5 Intelligent Control 18

3 Literature Review **24**

3.1 Dynamometer Operation and Control 25

3.2 Engine Testing and Calibration 36

3.2.1 Automated Engine Mapping 37

3.2.2 Sweep Mapping and Automated Engine Testing 37

3.2.3 Model Based Calibration Tools 38

3.2.4 Capturing Cylinder Pressure Data in Real-Time 39

3.2.5 Reduction of Cylinder Pressure Data 40

3.3 Engine Modelling 42
3.3.1 First-Principle Phenomenological Modelling 43
3.3.2 Mean Value Engine Model (MVEM) 43
3.4 NARMAX Models 44
3.5 Other Models 48
3.5.1 Sliding Mode and Constant Gain Extended Kalman Filter Models 48
3.6 Torque Estimation Techniques 49
3.6.1 Torque and IMEP Estimation using Crankshaft Rotational Frequency Variation 49
3.6.2 Torque Estimation using Engine Block Angular Acceleration . . . 50
3.6.3 Torque to Cylinder Pressure Correlation 50
3.7 Air-Fuel Ratio Control Strategies 50
3.7.1 Fuzzy Logic 51
3.7.2 Sliding Mode Observer 51
3.7.3 Event-Based Observer 52
3.7.4 Artificial Neural Network (ANN) Augmented Controllers 52
3.7.5 Model Predictive Control 58
3.7.6 Direct Inverse Model 59
3.8 Spark Ignition Timing Control 60
3.8.1 Self-Tuning Optimisation 60
3.8.2 Peak Pressure Position Control for Maximum Brake Torque . . . 60
3.9 Instrumentation for Torque Determination 61
3.9.1 In-cylinder Pressure Transducers 61
3.9.2 Spark Plug Ion Sensing 63
3.9.3 Torque determination via Piezoelectric Spark Plug Load Washer or Engine Mount Strain Measurement 63

3.9.4 Strain Gauge Fitment to a Drive Shaft 64
3.9.5 Magnetostrictive and Magnetoelastic Torque Measurement 65
3.9.6 Surface Acoustic Wave Measurement 66
3.10 Software for Engine Control 67
3.10.1 Ford's High Level Pascal-F Engine Control Software 67
3.10.2 Ford's Automatic Code Generator 69
3.10.3 BASEMENT 71
3.10.4 OSEK/VDK 72

4 Development of a Dynamometer Controller and Automated Engine Test System 73

4.1 Overview of Test Bed Work 74
4.2 The Dynamatic Dynamometer 81
4.2.1 Assessment of Controllability 84
4.3 Development of a Replacement Current Controller 90
4.3.1 Phase Angle and Current Controller 90
4.3.2 Current Controller Simulation 95
4.3.3 Controller Transient Behaviour 97
4.3.4 Speed Control Tests using the Current Controller 100
4.3.5 Dynamometer Response Characterisation 103
4.3.6 Improved Response Characterisation 107
4.3.7 State-space Estimation and Control of the Field Windings 118
4.3.8 Discretisation and Software Implementation of Flux Current Estimator 121
4.3.9 Standstill Testing of the Digital Current Controller 131
4.4 The Control Architecture 136

4.4.1 Control Hardware Selection 136
4.4.2 Arcom Viper 137
4.4.3 Human Machine Interface and Controls 138
4.4.4 Diamond Systems DMM32AT Data Acquisition Card 140
4.4.5 CAN-bus interface 141
4.5 Dynamometer Instrumentation for Control 141
4.5.1 Tachometer Circuit 141
4.5.2 Load Cell and Amplification Circuit for Torque Measurement . . 144
4.5.3 Protection Circuit 145
4.6 Automation Hardware 146
4.6.1 SSR Engine Power Management and Opto-Coupled Interface . . 146
4.6.2 Plint Volumetric Fuel Meter and Digital Retrofit 146
4.6.3 Cylinder Pressure Acquisition System 147
4.6.4 Thermocouple Interface 152
4.7 Software Architecture, Selection, and Development 154
4.8 Control Software 155
4.9 Application Software 155
4.9.1 The mseDyno Application 156

5 Development of an Engine Controller 160

5.1 Nippon Denso ECU 163
5.2 The MPC555 Microcontroller 167
5.3 The Time Processor Unit 169
5.4 The PATI Platform 170
5.5 Porting eCos to PATI 172
5.6 ECU Hardware Development 172

5.6.1 Fuel Injector *Peak-and-Hold* Driver Circuitry 172
5.6.2 Ignition Driver Circuitry 174
5.6.3 Engine Speed and Phase Sensors 177
5.6.4 Lambda Sensor Signal Conditioning Circuitry 181
5.6.5 Knock Detection Circuitry 181
5.6.6 Electronic Throttle Unit and Controller 186
5.6.7 Temperature Measurement 199
5.6.8 Intake Air Mass Flow Meter 201
5.6.9 Establishing the Ignition, Fuelling Sequence and Base Calibration 201

6 Conclusions **205**

6.1 Conclusions 205
6.1.1 Dynamometer Control 206
6.1.2 Engine Controller 207
6.2 Recommendations for Future Work 207

References **210**

A Circuit Schematic Diagrams **222**

B Embedded Control **241**

B.1 TPU Mask A 241
B.2 Linux as an Embedded Operating System 243
B.2.1 Booting Embedded Linux from Flash 247
B.3 Control Software 249
B.3.1 Diamond Device Driver 249
B.3.2 Phillips SJA-11001 CAN4Linux Driver 251

B.4 Application Software 252
B.4.1 Selection of a Widget Toolkit 252
B.5 The MPC555 and Time Processor Unit 257
B.5.1 Time Slicing 257
B.5.2 Channel Priority Levels 258
B.5.3 Code Development for the TPU 258
B.5.4 The Standard Masks 261
B.5.5 TPU Emulation Mode 265
B.5.6 PMM and PSP TPU Functions 266
B.5.7 The PCI 9056 Interface 270
B.5.8 The U-Boot Bootloader 275
B.5.9 The MPC555 Address Map Problem 276
B.6 Porting eCos to PATI 278
B.6.1 Real-time Performance 280
B.6.2 Open Source Architecture 280
B.6.3 Configurability and Scalability 281
B.6.4 Serial Interrupt Delay Problem 281
B.6.5 Creation of a TPU Device Driver for eCos 283

Chapter 1. Introduction

Conventional electronic engine management systems for spark ignition engines fitted to production road vehicles have limited control feedback. In addition, they require a high calibration effort to satisfy multiple objectives such as meeting emission legislation constraints, the driver's expectations (such as responsiveness, drivability, and acceleration performance), noise vibration and harshness (NVH), fuel economy, and longevity of the powertrain. For an SI engine, the two main manipulated inputs are duty of the fuel injectors (mass of injected fuel) and the spark ignition timing (ignition advance).

The control strategies typically employed on mass produced engines have limited adaptivity to accommodate real-world factors such as: variations due to manufacturing tolerances, component wear/deterioration and variations in fuel type and quality. Feedback learning occurs in the form of adaptive look-up tables which are updated on the basis of feedback from the oxygen sensor, for fuel correction and combustion knock events are used for correction of ignition advance timing. Learning algorithms can be as simple as varying ignition advance fuel injection quantities by fixed increments based on trial-and-error at a particular operating point. The last-known-good, or latest best estimates are stored in look-up tables which are written to non-volatile memory at key-off so that they can be reused to initialise the process after the next power cycle. The learnt values are constrained to within calibratable maximum and minimum values for reasons of meeting requirements for functional safety and long term reliability.

A recently produced SI engine might only include two main forms of sensory feedback, neither of which directly measure the engine's performance (torque or brake specific fuel consumption) to ensure it is optimal at the current operating point. The first feedback signal is the oxygen (or lambda sensor) which is used to infer air/fuel ratio by comparing the oxygen content in the exhaust gas to that of the ambient air surrounding the sensor. The fitment of this sensor has become commonplace since legislation has mandated the fitting of catalytic converters to the exhaust systems of gasoline powered vehicles. For correct operation, the catalytic converter requires that the air/fuel ratio is cycled close to stoichiometric. The second sensor type (which is less commonly fitted) is the piezo knock sensor. This sensor is used to detect the onset of combustion knock (or pinking) which is the phenomenon of the fuel mixture self-igniting under compression either before the ignition spark has taken place or before the combustion flame front has reached the unburned gas region in the cylinder. By sensing the onset of combustion knock (detected by measuring structure borne acoustic noise) using this sensor, the ignition timing can be controlled closer to the optimum for operating regions which are limited from reaching the optimum by the onset of knock.

An operating point is considered to be *knock limited* when the angle of spark advance required to locate the peak-pressure-position at the optimum for maximum brake torque (MBT) cannot be achieved without combustion knock occurring. The extents of the knock limited region will vary with the prevailing conditions, such as localised engine hot spots in the combustion space, and the humidity of the air intake charge. It is because of this unpredictable variability that either feedback is required (normally in the form of one or more piezo knock sensors), or worst-case conservative ignition timing must be used to accomodate poor fuel quality (lowest octane rating) used on an aged engine which has a build up of carbon deposits that result in a compression ratio increase. Historically, piezo sensory feedback has been used only for turbo-charged SI engines (due to the substantially increased knock risk arising from forced induction) whilst naturally aspirated engines were set to worst case for the fuel grade available in the market to which the vehicle was supplied. This has meant that many millions of vehicles have been operating sub-optimally due to the absence of additional feedback. More recently, rising fuel costs and environmental concerns have resulted in knock feedback becoming more commonplace on naturally aspirated SI engines.

For any given engine there will be a crank angle which produces the maximum brake mean torque where the in-cylinder combustion peak pressure, termed the peak-pressure-position (PPP) is located. The PPP which results in maximum brake torque (MBT) typically occurs at around 12° after-top-dead-centre, but this is engine dependent and may also vary under different operating conditions. Production engines are indirectly optimised for MBT by varying the ignition angle around the point which yields MBT. Since there is a nominally parabolic sensitivity to variations in spark advance, this amounts to a gradient search optimisation procedure. The main problem with this approach is that intake air humidity and cycle-to-cycle combustion variability causes the optimimum advance to be stochastic in nature which in turn causes the PPP to vary for fixed ignition angles. The use of knock feedback does not address this issue, but only allows for operation closer to MBT or optimal PPP for operating regions which are knock limited. The issue can be addressed by continuing the ignition advance to torque gradient search on-line or by directly measuring and controlling the PPP using individual cylinder pressure measurements and by ideally referencing this controlled pressure position to the output torque to ensure that the set-point is also optimal. Both of these approaches need to be robust to transient variations in the engine operating point (load and speed) and other disturbances, in order to be successful.

1.1 Motivation for Research

When an engine is used in either research or motorsport contexts, it is likely that several aspects of the engine's operation will need to be modified and the control objectives

changed. The calibration tables or *maps* used by the control strategies are often not normally available if the engine is of a new design, or the OEM controller is to be replaced and the calibrations have not been released to the researchers or developers. Additionally, the interdependence of the OEM supplied engine ECU upon other vehicle systems (such as security and immobilisation) is growing with time which further increases the complexity of using these vehicle ECUs outside of there intended context since they may require surrogate CAN signals to be provided as stimulus to allow normal operation. If this information is not readily available (usually in the form of a CAN signal database), then normal operation may not be possible using the OEM ECU and so it must be replaced by an alternative. For these reasons the OEM supplied engine controller is generally unsuitable for any application other than its intended purpose and a replacement must be sourced. The main issues with using an aftermarket controller are the initial setup effort in getting it calibrated to the point where the engine is in a runable state, then thereafter there may be insufficient flexibility to achieve particular control objectives. It would therefore be highly desirable to reduce the initial setup effort of the replacement controller and its optimisation for specific objectives thereafter. This work attempts to address these issues through the development of a flexible open-architecture modular engine ECU onto which a self-tuning algorithm can be developed to automate the initial calibration effort and provide in-service optimisation.

1.2 Research Methodology

The main output of an SI engine is the torque that it produces, however this is seldom measured on-line as a basis for automotive engine feedback control. It is proposed to design a self-tuning, or adaptive engine controller which uses torque feedback from direct measurement as the major variable which the controller can use for tuning or adaptivity.

Torque measurement and estimation are not new concepts and can be achieved through a variety of direct and indirect means. Torque measurement techniques include spark plug ion sensing (Nielsen & Eriksson, 1998), engine mount load monitoring, crankshaft rotational frequency variation (Rizzoni, 1989) and magnetostrictive torque sensing (Fleming, 1989). It can be assumed that none of the techniques alone have so far been deemed adequate for use on production vehicles which may be because of cost and reliability for a particular method. Also there may be no perceived *added value* from a customer-sales perspective as fuel economy benefits may be outweighed by the up front expensive and complexity of additional instrumentation. Rising fuel costs have increased the emphasis on fuel economy from a sales perspective and the addition of a torque sensor to production engines seems more tangible when considering recent advances in

torque sensing technology (such as the ABB Torductor) which mean that a sufficiently cost effective and reliable torque sensor could be fitted.

For this work, direct torque measurement was the preferred sensing method since the inferred measurements involve significant additional complexity, are usually model dependent, and have more than insignificant margins of error associated with them. It is proposed that with 'knowledge' of the system's main output, the closed-loop controller can become self-optimising and therefore a significant performance enhancement may be realisable for a given particular engine over a wide range of operating conditions, when compared to the fixed pre-calibrated control strategies that are commonly used in these applications. In particular, it is expected that this approach will result in an improved rejection, or even opportunistic optimisation, of disturbances arising due to changing environmental conditions, such as intake air humidity, or other unmeasured factors including those related to in-service ageing. The use of direct torque feedback not only allows the operating set-points to be monitored and achieved under wider conditions (including the extremes of humidity and throttle transients), but also allows for actively selecting and optimising the set-points (such as A/F ratio and PPP) on the basis of both performance and fuel economy. Further application of this approach may allow the use of multiple fuel types and/or combustion enhancing methods (such as water injection or onboard oxyhydrogen generation) to best effect. There is also the possibility to perform condition monitoring by checking for significant differences between the expected (leaned) output torque and the actual output torque.

This work is not the first to consider the approach of using torque-feedback. However, it has been the particular aim of this project to establish if a controller can be constructed which initially relies on having only minimal information about the characteristics of the engine which it is to control (it's general configuration, number of cylinders etc) to remove the need for having a so-called *base calibration*. This is of particular benefit for situations where the calibration tables or *maps* either do not exist, are not available, or do not apply to the intended mode of operation. This is typically the case when an engine is to be used for research or motorsport applications. The work does not consider the automatic determination of sensor charateristics such as thermistors, mass airflow meter, or determining control parameters for electronic throttle since it will be asssumed that these are already known or can be determined independently.

A parametric approach can lead to a parameter tuning effort similar to the lookup tables, so a blackbox approach is preferred since this will allow a more automated identification process to take place. Applying a continuous perturbation signal to inputs then slowly taking the engine through its operating range may provide sufficient time for cycle averages to settle upon local optima.

1.3 Reseach Objectives

The aim of this study is to determine if directly or indirectly measured torque output from an SI engine can be utilised to improve either or both the short term dynamic performance and longer term calibration drift. To achieve this aim it was proposed to develop and assess an SI engine control algorithm(s) with a self-adaptive optimisation or learning capability. To achieve this aim the following objectives were outlined:

- Review past and current research in direct and indirect torque measurement, and to decide how best to utilise the instantaneous torque measurement to benefit control.
- To set up an automated engine test environment to facilitate the assessment of the torque controller.
- Assess the benefit of using Torque-feedback for SI engine control as the main variable for the adaptation or learning.
- Instrument an existing Jaguar AJ26-V8 1998 MY engine for automation of testing and assessment of the said engine control algorithm.
- Study how existing ECU hardware and control strategies work and identify any possible deficiencies.
- Design a proposed engine controller hardware with the required flexibility to allow implementation of a torque based algorithm.
- Retro-fit a dynamometer controller to an existing facility to meet the above requirements.

Hypothesis Can torque feedback be used to perform an on-line identification and optimisation of an SI engine to produce a worthwhile improvement in performance and disturbance rejection, compared with other estimation techniques.

1.4 Contribution and Research Roadmap

The emphasis of this study was to create an engine controller which has as little *a priori* knowledge as possible about the engine which it is to control. This necessitates a black box model based approach and therefore the work has been carried using a real engine rather than an engine model so as to preserve the stochastic and noise prone characteristics of the real plant.

The NARMAX model approach has been considered, a blackbox nonlinear time series method, which can describe the development of torque with respect to time in response to changes in inputs. Significant time delays occur in the system due to a variety of time constantants and propagation delays, for example due to fuel puddling evaporation, intake air propagation, exhaust to lambda gas transport. Most of the system time constants are dependent on engine speed and may also be effected by other factors such as engine load. Broadly speaking, there can be two main approaches to using a blackbox model to solve this particular problem. One of the two approaches is to identify the direct inverse so that the inputs and outputs are essentially reversed. This enables the desired real system output (torque) to be used as an input to the model which will then provide the necessary real inputs (as outputs) ahead of time as feed-forward control input so that disturbances may be rejected. This approach suffers from the problem of invertability, for example a particular torque may be obtainable from more than one combination of air-fuel ratio and ignition advance angle and therefore there is no single inverse solution. It also relies on the fact that the desired optimal torque must be known under all conditions an identified in the model and therefore a continuous process of on-line identification would be required. The alternative approach is to identify the forward model and use it as a part of an optimal control scheme such as a model predictive controller. This approach was favoured since it removes the issue of invertability and also allows the direct incorporation of subjective constraints such as the trade-off between maximum brake torque and maximum brake specific fuel consumption.

The design of an eddy current dynamometer control system has been a requirement of the work undertaken during this project to support the control algorithm development. A number of dynamometer control challenges were met during the project including:

- Timely control of current/magnetic flux of the eddy current dynamometer
- Effective decoupling of interdependent torque and speed control loops
- Tracking of transient or varying torque/speed set points

The body of scientific literature which describes the operation of eddy-current dynamometers is actually written on the subject of the theory and operation of eddy-current couplings that were in wide spread use around the middle of the last century. The two classes of machine have essentially the same principle of operation where the eddy-current dynamometer appears to have been developed as a special application of the equivalent design of coupling.

Roadmap

- The review of relevant literature falls into three categories related to engine modelling, control, and engine testbed infrastructure and test techniques. This process is on-going throughout the work, but the main outcomes are to be known after two years from the project start with work being undertaken in parallel with development of the test infrastructure.

- Work on the engine test environment infrastructure to be largely completed after one year with subsequent work on-going to support particular research requirements.

- Development of a modular ECU to allow flexibility in the way the engine control is implemented is to be completed within one year to allow sufficient remaining time to complete the experimental aspects of the research.

- The final year is to be spent evolving the general concept of an adaptive, self-learning engine controller. Initially tests are to be performed on offline data collected from the actual engine to establish the correct modelling approach. This is to be followed by repeating the model structure search and parameter identification on-line to establish if measurement noise and the stochastic nature of engines prevents convergence to a model comparable with the initial offline attempt. With either the on or offline models, the next stage is to embed it into a control scheme which can use it for feed-forward control to reject disturbances. These activities will be carried out in the following systematic way:

  - Work on structure identification techniques to establish the nature of the problem offline using real engine data. This first step shall reveal if a single NARMAX model can adequately describe the entire operating range of the engine, or if separate models are need to be switched in to cover the varying dynamics as the operating point changes.
  - Assuming that a structure for the engine characteristics can be found, the parameters of the model must be identified then tested and validated against further engine data.
  - Once an offline model is proven, the challenge becomes to construct an on-line scheme for structure detection and parameter identification. This forms the basis for the adaptive self learning controller. The engine must be taken slowly enough through its operating range that a solution can converge. This will require a perturbation signal to be applied to each of the controlled inputs for continuous on-line model validation.
  - With the system model known, or obtainable on-line, then a model predictive control scheme can be implemented and tested around the model. At this stage constraints can be tested such as the limitation of air/fuel ratio

excursions during transients, maximum power performance, or maximum fuel economy. The controller can then be compared to the original OEM fixed calibration unit to see if there has been any net gain in control performance.

- If constraints can be successfully applied and met, then this can be extended to allow dynamic modification of the constraints to achieve changing objectives. Combustion enhancing methods such as the injection of water or hydrogen into the intake stream may then be then be usefully applied to achieve new optimum operating points yielding either higher specific power output or lower specific fuel consumption.

Chapter 2. System Identification and Adaptive Control

In order to be able to consider some of the engine control techniques that have been presented in the literature, it is first necessary to review some of the supporting theory. Conventional control and signal processing techniques assume that processes and systems have fixed parameters. The proportional integral derivative (PID) controller is commonly used in both continuous and discrete applications as it is widely understood and tolerant of slight non-linearities, can be applied to systems where there is virtually no knowledge of the plant model, and new tuning techniques are being developed all the time. However if the controlled process is highly non-linear it may be necessary to retune the controller for each set point to maintain the desired system response. For non-linear plants which can be considered approximately linear in the region of a chosen set point, it is possible to construct a table of gains and other parameters for each set point. This approach is usually adopted for highly non-linear plants in systems that have frequently changing set points. This technique is called *Gain Scheduling* (Figure 2.1) and is used in production engine management systems in the form of look-up tables often referred to as *maps*. A simplified diagram of the use of a calibrated map for fuel injector duty to maintain air/fuel ratio is shown in Figure 2.2.

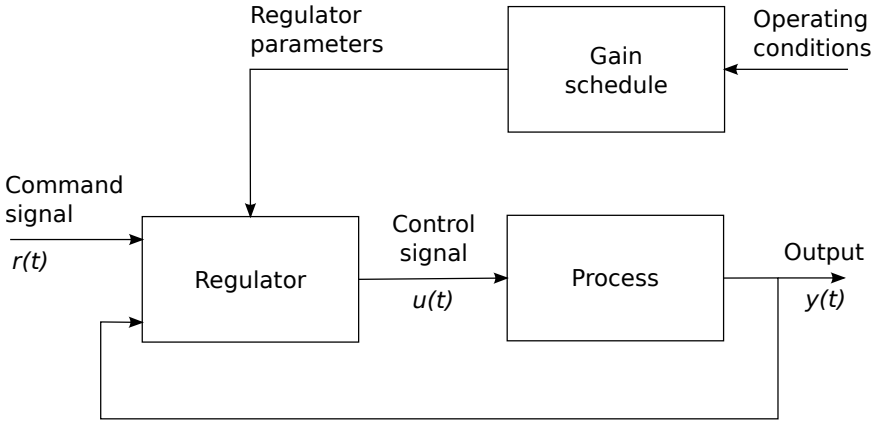

Figure 2.1: A generalised Gain Scheduling scheme (Lelic & Gajic, 2002)

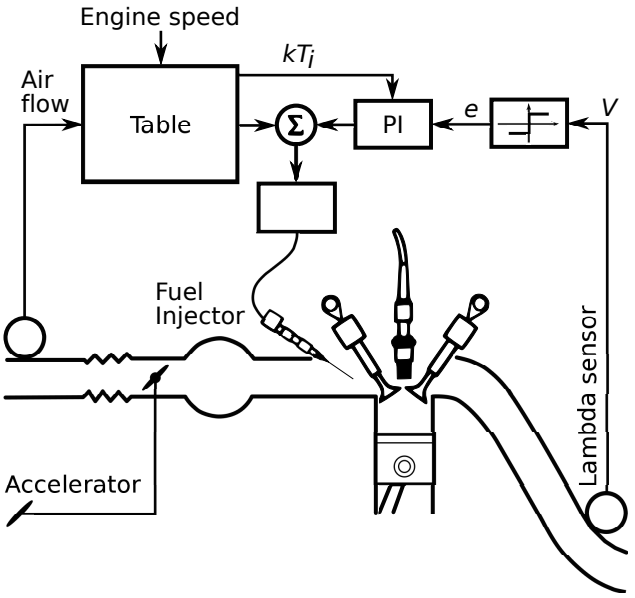

Figure 2.2: Engine mapping shown as a form of Gain Scheduling

When in addition to non-linearities, a process has variable time delays, varying parameters, and is disturbed by time-varying noise (as is the case with the SI engine), then the PID controller may not be capable of controlling the plant to the desired performance. It is in such cases that self-tuning and adaptive control techniques are required. Adaptive controllers have evolved over many years and different approaches have been combined so that they can now be categorised into three generalised groups:

1. Indirect or explicit adaptive controllers use online estimates of plant parameters to adjust the control law. This type of controller is often referred to as a self-tuning controller (STC).

2. Direct or implicit adaptive controllers make no effort to identify the plant parameters. The control law is adjusted to minimise the error between the plant output or states and those of a chosen model. The model used has the desired plant response and is used to constrain and modify the actual plant response. This type of controller is often referred to as a model reference adaptive controller (MRAC).

3. Intelligent adaptive controllers make use of black-box techniques such as neural networks, wavelets or fuzzy logic. With these techniques it may be possible to directly incorporate expert (or heuristic) knowledge and to effect adequate control of a very complex system without having any phenomenological knowledge to construct the plant model.

It should be noted that adaptive controllers are inherently non-linear and stability analysis can only be determined analytically for certain limited cases. It has been shown that if the gain of a system is changed dynamically within the fixed-gain stability margins

then the resulting system may not be stable. So the addition of an adaptive algorithm can potentially destabilise an otherwise stable system. This can have implications for safety, particularly if there is a burden of proof to show that a system can not ever become unstable in the field. MISRA (1994) gives some guidelines on the use of adaptive control on production automotive applications, which it considers to be a form of continuous on-line optimisation. It recommends limiting the number of adaptive variables to restrict the number of degrees of freedom and to use adaptivity in conjunction with look-up tables to compensate for significant nonlinearities. It also states that adaptivity should be considered during failure modes and effects analysis (FMEA). Neural networks are also covered by MISRA (1994), in which they are considered as an *emerging technology* for which it makes the particular recommendation to turn off the learning process once training has been completed. In effect, this means that training must not be performed at all in a production control unit to compensate for mechanical wear, for example. It also highlights the difficulty in demonstrating stablity under all operating conditions which would be required as part of the FMEA process performed for production vehicles.

2.1 Self-Tuning Control (STC)

Self-tuning control or indirect adaptive control differs from conventional control because it introduces algorithms with coefficients that can vary with time. The idea behind STC is to find an algorithm that can adjust its parameters to achieve a particular performance objective especially when the plant to be controlled is subject to time varying changes. The process parameters are estimated in real-time and these parameters are applied without consideration of their uncertainty. STC is usually used to compensate for slow performance degrading plant changes rather than attempting to track fast moving changes. Since adaptive controllers are inherently non-linear there is no way to guarantee stability except for certain restricted cases. Also it is not always certain that the parameters will converge. A general block diagram for a STC scheme is shown in Figure 2.3:

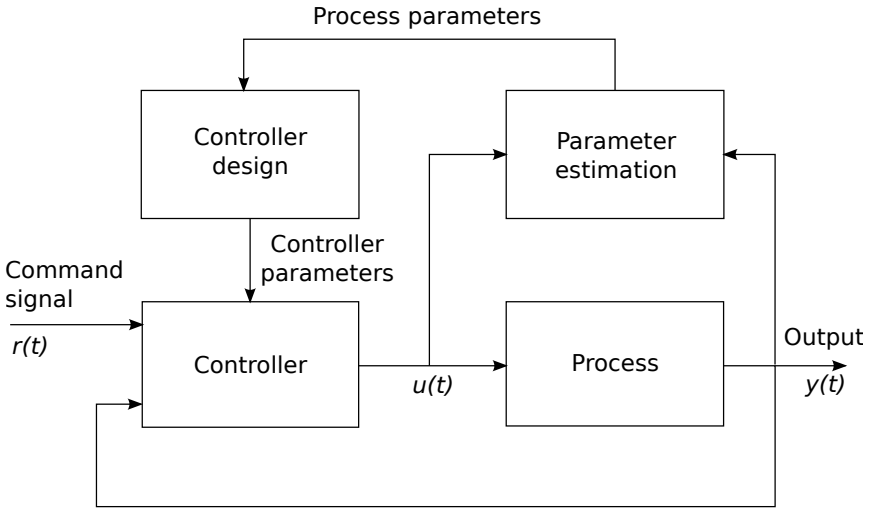

Figure 2.3: Self-tuning control scheme (Lelic & Gajic, 2002)

2.2 Model Reference Control (MRAC)

A Model Reference Control or direct adaptive control scheme makes use of a model which has the desired response to a set point. The desired response is compared with the output of the system to form an error signal. The controller parameters are adjusted accordingly to minimise this error. A general MRAC block diagram is illustrated in Figure 2.4.

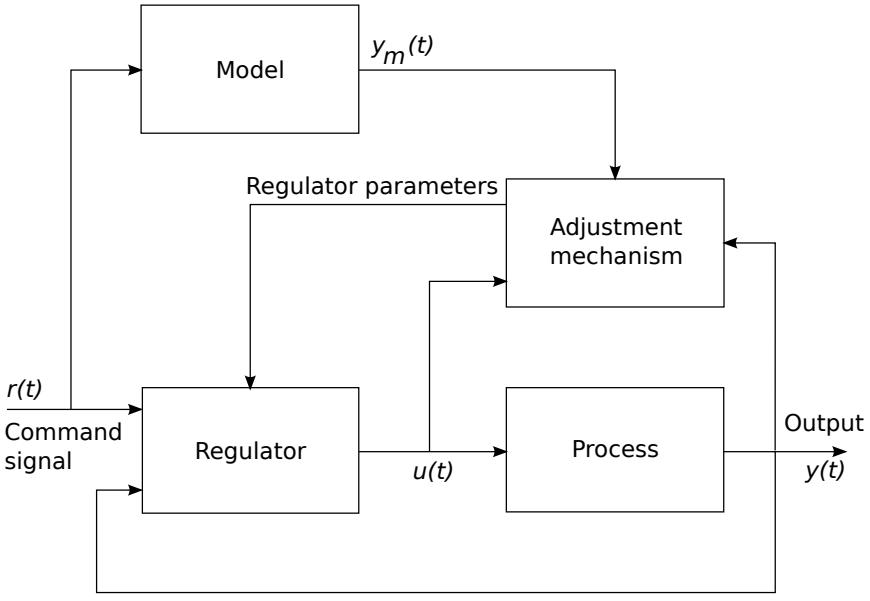

Figure 2.4: Model reference control scheme (Lelic & Gajic, 2002)

The are several modified adaptation rules which make use of *Lyapunov* stability techniques, but otherwise the stability of the algorithm is not guaranteed for gradient type methods of parameter adjustment. A simple example of a continuous time model

for a MRAC system is shown in 2.5 whereby a first order model is used as a reference to control a second order plant.

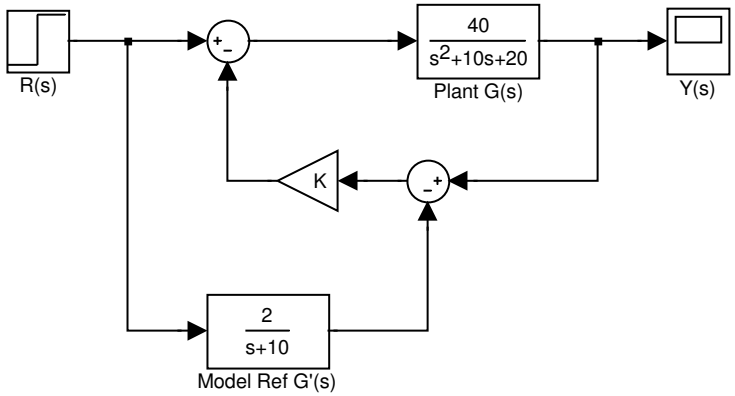

Figure 2.5: SIMULINK block diagram of a simple example MRAC scheme (Dutton et al., 1998)

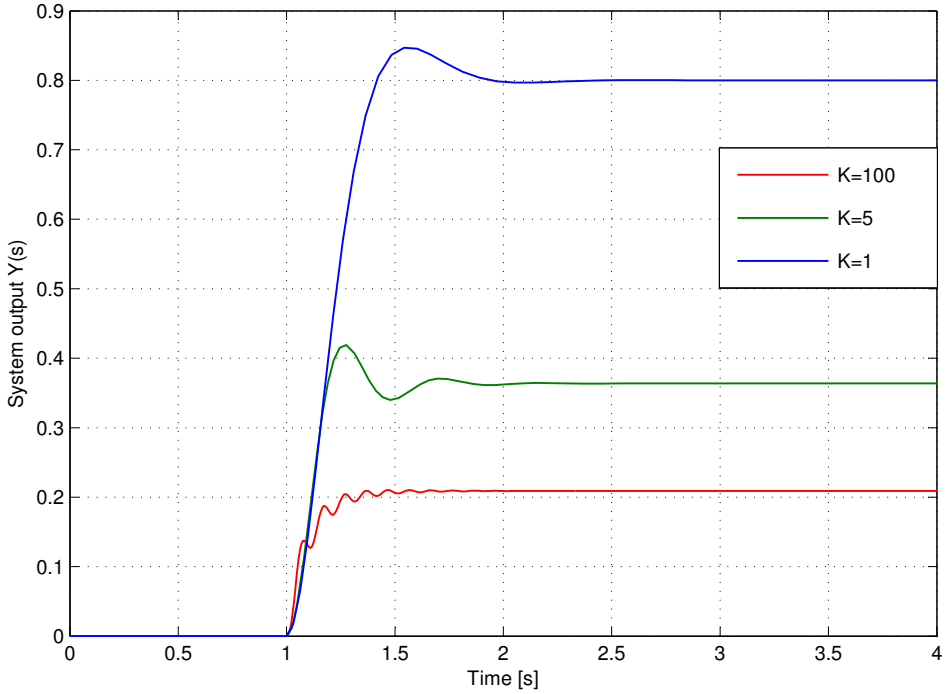

Figure 2.6: Example MRAC step responses for various gains showing decaying higher order oscillation (Dutton et al., 1998)

Figure 2.6 shows the step response of the system for different gains. High order oscillations caused by interaction can be seen as the gain is increased. This is often a problem for MRAC schemes as the oscillations can excite un-modelled higher order terms in the plant which may lead to stability problems. For this reason STC is often preferred to MRAC for many practical applications.

2.3 Model Based Predictive Control

The term model predictive control (MPC) does not designate a particular control strategy (Camacho & Bordons, 2003), but refers to a range of methods which make explicit use of a model of a process to obtain the control signal by minimising an objective function. Most of the techniques encompass the idea of using a process model to predict the output at future time instants up to a finite limit called a horizon. A control sequence is calculated (on-line) to minimise an objective function, and there is the notion of a receding horizon since at each control interval the horizon is displaced towards the future. There are a variety of MPC schemes in existence, not all of which are published as the details of them are considered proprietary as they form a part of commercial software packages. One such commercial implementation which is often discussed and documented is called Dynamic Matrix Control (DMC) and has been used widely by petrochemical industries. Later, an open MPC implementation was published, known as Generalised Predictive Control (GPC) (Clarke et al., 1987a,b) and has since become a reference implementation which is often used and cited in the literature. GPC is said to be able to overcome problems that minimum variance control has with process dead-time in one algorithm, being capable of stable control of process with variable parameters, variable dead-time, and changes in model structure provided that the inputs and outputs are excited in such a way to allow the model identification process to be run on-line in parallel with the controller. GPC is often used with black box linear time series models such as ARMAX. This type of model can be identified and parameter fitted statistically without knowledge of the process, which has allowed software packages to be written to construct a model and run an MPC controller for an arbitrary plant (provided it can be represented approximately linearly).

Some features of MPC are as follows:

- Explicit handling of constraints (useful for safety)
- Control of multiple input multiple output systems (MIMO) is possible (provided a plant model exists)
- Explicit handling of process time delays and dead-time
- Can operate closer to constraints and safe operating region boundaries than conventional control, resulting in improved performance
- Non-linear possible but stability proofs for limited cases only just becoming available which may limit is use
- Use of cost functions or objective functions to subjectively influence how the controller behaves

- Tuning is performed using: a control horizon, a prediction horizon, and weighting matrices

Figure 2.7 shows the general structure of a MPC scheme.

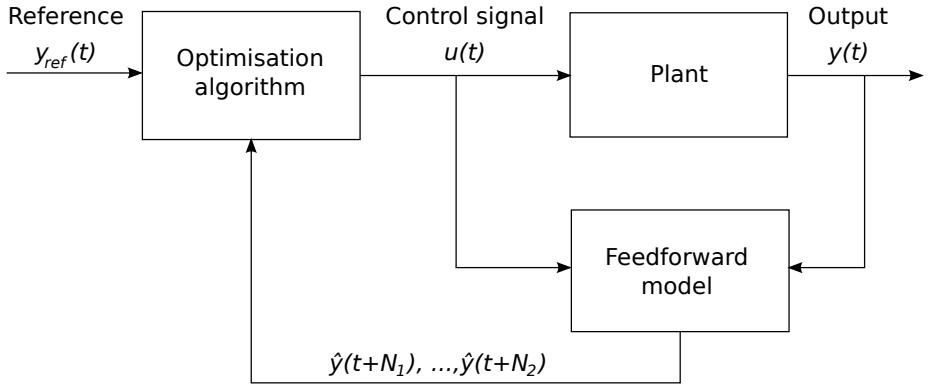

Figure 2.7: General structure of a model-based predictive controller (Barreto & Araujo, 2004)

A factor which has been blamed for the relatively low level of uptake of MPC outside the chemical and process industries is the lack of formal proof of stability, particularly for the non-linear model case. This matter is being addressed in the literature, but some assurance can be gained from the use of constraints to contain the actions of the controller within safe regions of operation, although this does not constitute an absolute proof that instability will not occur. It should be noted that even if a linear model is used, that if constraints are used, then the resulting controller will be non-linear due to the possible discontinuity in it control action which could in itself excite non-linear plant dynamics. Nuno & Biegler (1994) deals with the effects of adding constraints to MPC and the potential for inducing instability.

The subject of recent research interest has been that of non-linear MPC, or NMPC. Chikkula et al. (1995) attempt the use of a Volterra time series model within a conventional MPC framework and strategies for robustness are discussed. Chen et al. (2000) consider that MPC applied to non-linear systems requires the on-line solution of a non-convex, constrained nonlinear optimisation problem, which can lead to a large increase in the computational effort and loss of optimality arising in the optimisation procedures. The authors have developed a MPC scheme for nonlinear systems which it is claimed can guarantee asymptotic stability. Mrabet et al. (2002) have attempted a NARX model combined with a structure detection algorithm and a gradient decent based evaluation of a cost function for control. Bai & Coca (2008) attempt a predictive controller based around NARMAX models and try to show the robustness of the approach.

2.4 System Identification and Parameter Estimation

The purpose of mathematical modelling, for a control objective, is to construct a mapping of inputs to outputs to reflect the relationship between them to a reasonable accuracy. Experimental data based modelling is known as system identification, which requires a suitable model that should be made up of the simplest model structure, and the smallest number of input variables and adjustable parameters, that can achieve the desired accuracy.

A system's input and output at time t can be denoted by $u(t)$ and $y(t)$ respectively. It is useful to describe a system in discrete time as it is usually observed by sampling. If the sample period is assumed to be a unit of time then for linear systems the following generalisation can be made as a linear difference equation (2.1):

$$y(t) - a_1 y(t-1) + \ldots + a_n y(t-n) = b_1 u(t-1) + \ldots + b_m u(t-m) \tag{2.1}$$

This is known as the ARMA model (Auto-Regressive Moving Average) because the a coefficients give a dependence on previous outputs (hence auto-regressive) and the b coefficients give a dependence on present and past inputs (hence moving or weighted average). The following column vectors may be defined by collecting terms (2.2):

$$\theta = [a_1 \ldots a_n \quad b_1 \ldots b_m]^T \tag{2.2}$$

$$\psi = [-y(t-1) \ldots -y(t-n) \quad u(t-n) \ldots u(t-m)]^T \tag{2.3}$$

Equation 2.3 can be written in a simplified form, rearranged to express the new output y in terms of the previous inputs and outputs (2.4):

$$y(t) = \hat{y}(t \mid \theta) = \psi^T(t)\theta \tag{2.4}$$

The notation $t \mid \theta$ in (2.4) is used to emphasize that y is a function dependent on both time and the parameter set of θ. It is usually the case for complicated systems that the parameters represented by θ are not known *a priori* and must be determined by experiment. A data set of input and output values obtained over a period of time can be used to estimate θ. The least squares linear regression method is often used and the following loss function can be defined based on N samples (2.5):

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y})^2 \tag{2.5}$$

The value of $\hat{y}$ is a model estimate of the actual output y for each corresponding input. Since the function J is a quadratic its minimum can be obtained by solving for its differential equated to zero. This method allows parameters to be estimated from a predetermined data set of N observations, but if a new input/output data pair were to be added to improve the estimate, it would require that the whole calculation be repeated each time. This necessitates that all previous data are stored and the memory requirement and the number of calculations needed to obtain the result would increase indefinitely with each iteration. Often data is observed sequentially and in the case of adaptive control the calculation has to be performed online (although an initial estimate can be provided from previous experimentation). It is therefore necessary to rearrange the computations so that they may be performed recursively in separate iterations. This enables a result previously obtained for N observations to be used to calculate an updated estimate for $N+1$ observations (2.6) and (2.7).

$$P_n = P_{n-1} - \frac{P_{n-1}x_{n-1}x_{n-1}^T P_{n-1}}{1 + x_{n-1}^T P_{n-1} x_{n-1}} \tag{2.6}$$

$$\theta_n = \theta_{n-1} - \frac{P_{n-1}x_{n-1}\left(x_{n-1}^T \theta_{n-1} - y_n\right)}{1 + x_{n-1}^T P_{n-1} x_{n-1}} \tag{2.7}$$

To cover the case where abrupt parameter changes occur, P_n can be periodically modified or reset by a decision making algorithm. A *forgetting* factor (or discounting factor) can be used for slow time varying parameters to add a time weighting to reduce the effects of old data. This is usually achieved by multiplying equation (2.6), P_n by a factor less than one. This factor does not have to be a fixed value but can be a function such as an exponential or variable which is dynamically altered in real time by some other process such as a neural network. Other recursive estimation techniques exist including recursive extended least squares (RELS), for use with the ARMAX model (see below), and recursive maximum likelihood (RML). The least squares method can be used for non-linear identification but is restricted to models that are linear in the unknown parameters.

An extension of the ARMA model is the ARMAX (Auto-Regressive Moving Average with eXternal input) model which adds a disturbance term in the form of a moving average of white noise. The ARMA and ARMAX models are just two permutations of a family of models that must be selected to best suit the application. When non-linear black-box models are considered it is most common to use only measured quantities

rather than estimated ones. In the case of engine control the complexity arises from unmeasured disturbances which requires a different approach to be taken. A model that relies upon estimation that is finding application in engine control is the Nonlinear ARMAX or NARMAX (Nonlinear Auto-Regressive Moving Average with eXogenous inputs) model. Equation (2.8) shows the NARMAX form where $F(\cdots)$ is some non-linear function. It is generalised as far as possible for a finite non-linear function and its similarity to the ARMAX expression is apparent.

$$y(k) = F\left(y(k-1), \ldots, y(k-n_y), u(k-1), \ldots, u(k-n_u)\right) \tag{2.8}$$

The NARMAX model is valid for non-linear systems that can be considered linear when operated close to an equilibrium or set point. If the set point is changed then the whole model structure might change and have to be re-evaluated. The NARMAX approach considers all possible permutations of the linear combination of input and output terms. The significant combinations have to be identified and various methodologies for this are presented in the literature such as orthogonal least squares. Once the reduced set of parameters has been chosen, then values for them are obtained from past data. As with other empirical model types, the system data must be significantly rich enough which requires that the system has been stimulated by a variety of inputs for the model to be valid. A process of validation is used to check the model against a set of criteria.

Leontaritis & Billings (1987) prove that if a pseudo random binary sequence (PRBS) is used to excite a non-linear system , it can cause loss of identifiability. This fact is reiterated when work to identify an automotive turbo-charge diesel engine is carried out (Billings et al., 1989). PRBS are a popular technique to excite plant for identification linear ARMAX models as the signal can be imposed on top of the control output of a closed loop system which is particularly useful for identifying system which are open-loop unstable or require control for practical reasons. PRBS can still be used provided that the level or amplitude is also varied so that the plant being identified is being stimulated or excited by a sufficiently frequency rich signal to expose it's dynamics. For a closed-loop system, this may require that the set-point is varied in such a way to provide additional stimulus to the system over and above that of the PRBS signal added to the control output.

2.5 Intelligent Control

Intelligent control methods can be loosely grouped into three categories of so-called black box techniques that are: Artificial Neural Networks (ANN), Wavelets, and Fuzzy Logic. Neural networks have become a popular model structure in recent years and the

name refers to structural similarities with the neural synapse systems in humans and animals.

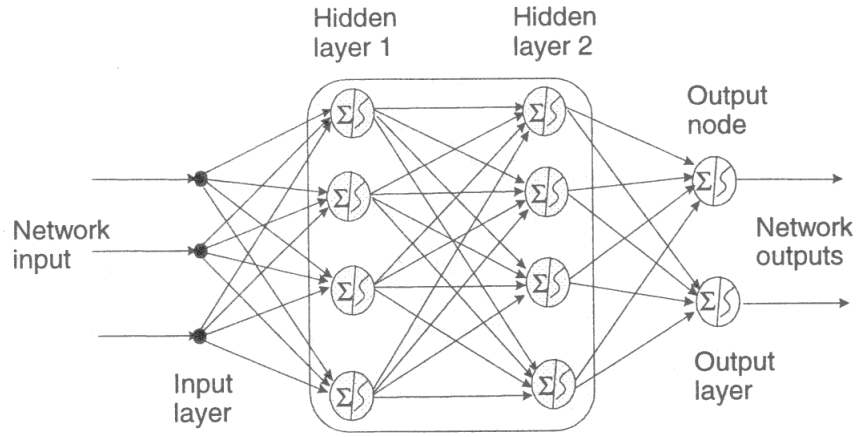

Figure 2.8: MLP neural network configuration

The Multi Layer Perceptron Neural Network (MLP) type of ANN (Figure 2.8) is supervised which means that the system updates it's weighting parameters in response to feedback it gets on its performance measured by the difference between the network output and the desired output provided by the network trainer. A series of layers of nodes are connected in a network fashion. An MLP network with two hidden layers is shown in Figure 2.8. Each node performs a particular nonlinear operation to a weighted sum of its inputs. The node output is then fed into the input of the node in the next layer. The output is affected by all of the weights. MLP is therefore computationally intensive for responding and learning. In particular complex systems require lots of nodes to achieve a reasonable accuracy and so MLPs are often not suitable for real-time applications such as engine control.

Radial Basis Function Networks (RBF) are another topology for neural networks. RBFs are based upon the notion that an arbitrary function $y(x)$ can be approximated as a linear superposition of a set of localized basis functions. RBFs look much like the common feed-forward architecture used with back-propagation training as shown in Figure 2.9.

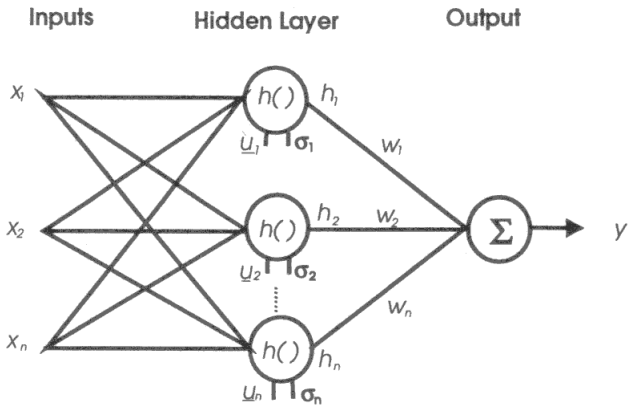

Figure 2.9: RBF network structure

There are three layers which comprise the input layer, hidden layer, and output layer. Each input is exhaustively connected to every hidden neuron via un-weighted links. Each neuron calculates the following exponential shown as equation (2.9):

$$y = e^{\frac{-|x-u_i|}{2\sigma_i}} \tag{2.9}$$

where x is an input vector, u_i (the output of the i^{th} neuron) is a vector representing the centre of the i^{th} basis function and σ_i is the width of its Gaussian. The output of the network is the weighted sum of the outputs from the hidden layer neurons. RBFs are capable of rapid training, generality, and simplicity but the number of nodes required to perform complicated tasks can be very large. This drawback is often referred to as the *curse of dimensionality*.

The Cerebella Model Articulation Neural Network (CMAC) algorithm consists of two mappings and an output computation for determining the value of a complex function. It has been shown in the literature that CMAC is a Gauss-Seidel iterative scheme for solving linear equations to an arbitrary accuracy.

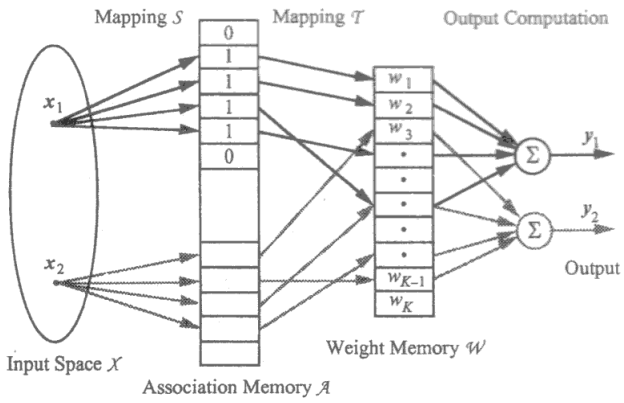

Figure 2.10: Diagram of CMAC topology

A CMAC network layout is shown in Figure 2.10. There are two mappings that are denoted by S and T. The mapping S relates the input vector to C locations in the association memory A. The mapping constructs overlapping receptive fields. If an input falls within the i^{th} receptive field then that field becomes active and the i^{th} location of A has a value of one or zero. This bears some resemblance to the fuzzification process or membership function used with fuzzy logic. The second mapping T relates each location of A to a particular adjustable value in the weight memory W. The quantisation and mapping structure allows nearby inputs to activate one or more of the same C weights which can produce similar outputs. This creates a degree of generality which means that training for one set of conditions can be applied to another similar but non-identical set of conditions. This is more computationally efficient than some of the other schemes and lends itself well to real-time applications. The general learning algorithm is (2.10) where β_I is a learning rate, d is the instantaneous desired output, and y is the actual output:

$$\Delta w = \beta_I \frac{d-y}{C} \tag{2.10}$$

Small values for β_I, less than one are used (e.g. 0.25) to ensure smooth convergence.

Recurrent Neural Networks (RNNs) are another type of ANN that are suited to simulating the dynamic behaviour of a controlled system. RNNs are derived from Multi Layer Perceptron Feed Forward (MLPFF) networks by introducing feedback connections amongst the neurons. The non-linear mapping features of MLPFF allow black-box non-linear dynamic modelling. An external recurrent neural network feedback typology can be written as (2.11):

$$\hat{y}(t,\theta) = F\left(\hat{y}(t-1 \mid \theta), \ldots, \hat{y}(t-n \mid \theta), u(t-1), \ldots, u(t-m)\right) \tag{2.11}$$

where $\hat{y}$ is the neural network output, u is the input and θ is the parameter vector of the model. The indices n and m are set once the lag space of both the external input u and the feedback variables are fixed. The similarity to the describing equation (2.8) of the NARMAX model is noticeable. Where neural networks are to be used as a controller they usually fall into one of two major classes which are Direct Control Systems (DCS) and Indirect Control Systems (ICS). With DCS the network is used as a controller and its outputs directly control the actions. With ICS the network outputs are processed using an optimisation algorithm to determine the control actions. The DCS approach acts as a controller which processes the signals coming from a real system. The control signals are evaluated as a function of the target value of the controlled variable. DCS can be further sub-classified into two types which are the Direct Inverse Model (DIM) and Internal Model Control (IMC). DIM works by training the ANN (parameter

identification) to simulate the inverse dynamics of the real process then the inverse model becomes the controller. The dynamics of a system can be described by (2.12):

$$\hat{y}(t+1) = F\Big(\hat{y}(t), \ldots, \hat{y}(t-n+1), u(t-1), \ldots, u(t-m+1)\Big) \tag{2.12}$$

The inverse RNN is an estimate of the input that caused the current output in terms of the current output and previous inputs. The most recent control input can be isolated to obtain the inverse RNN as (2.13):

$$u(t) = F^{-1}\Big(y(t+1), y(t), \ldots, y(t-n+1), \ldots, \hat{u}(t-m+1)\Big) \tag{2.13}$$

After training the network can be used as a controller by replacing the output $y(t+1)$ with the required target value $r(t+1)$.

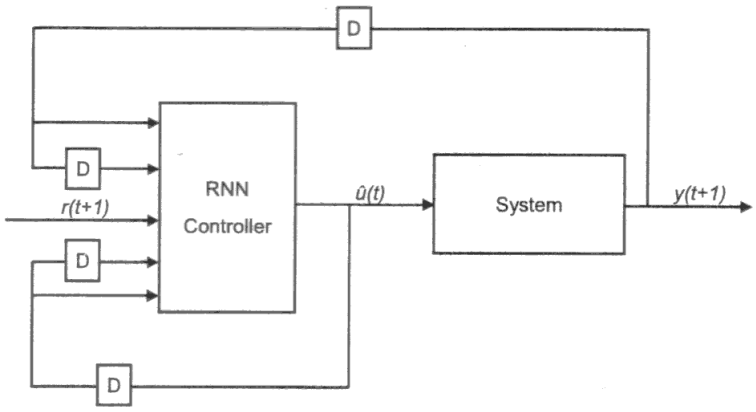

Figure 2.11: Direct Inverse Model structure: The controller performs the control action one time step later than the desired target

The DIM structure is shown in Figure 2.11. The advantages of using DIM are its simple implementation and dead-beat inbuilt control properties which results in a fast response for wide and sudden variations of the state variables. The main disadvantage is that the training is performed offline and so it cannot be considered an adaptive scheme without extending it to have an online training methodology. IMC is derived from DIM in which a predictive model is added to work in parallel with the inverse controller. The error between the predictive model and the real plant is fed back as an additional controller input as can be seen in Figure 2.12.

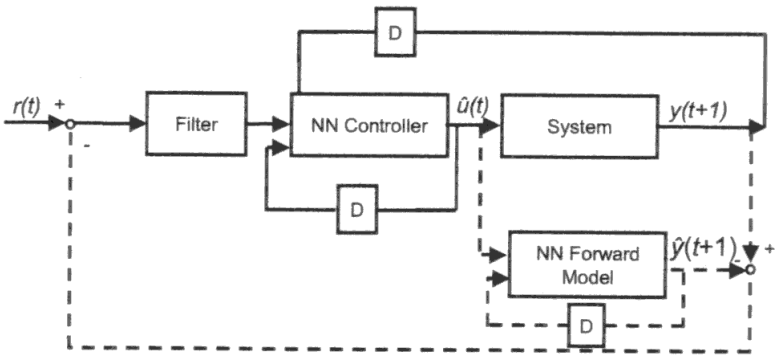

Figure 2.12: Internal Model Control structure

A filter is used to improve robustness. For the training a goal driven approach is used and the network is trained to minimise the difference between the output of the predictive model and the target value.

Chapter 3. Literature Review

Historically there has been much work published related to engine control issues. Much of it claims some break-though in the approach, but in practice it usually represents an incremental step along the road of development. Novel 'new' control strategies are seldom taken up by vehicle manufacturers. This review is concerned with the work to-date that relates to an improvement upon the trusted but limited capabilities of the lookup controller that is in wide-spread use. There is also consideration of work which relates to the instrumentation and modelling challenges that are present when performing work in this field.

It is not feasible to produce an accurate model of a spark ignition (SI) engine directly from first principles without the inclusion of empirical data to augment it. Models which attempt a first principle only approach have been shown to be of limited use due to the parameter calibration effort and their sensitivity to an inherent inability to cope with time-varying parameter change. The individual components of the SI engine processes such as air induction, fuel flow, and combustion each constitute significant ongoing research effort. Some of the many control strategies are essentially just robust linear designs with accommodation for plant uncertainty as a coping mechanism for the inherent non-lineararities. Transient response predictors have also been embedded, although none of these approaches offer a unified solution and often suffer from issues of robustness, especially to parameter changes of a real engine used on the road. The results of many of the strategies developed are not tested on an actual engine let alone different engines; they are instead tested against computer simulation models that have been verified against an actual engine. Although this is both convenient and desirable as a fast means of testing a particular strategy, this approach does not demonstrate how robust a real implementation of a controller would be and how it may interact with or react to higher order effects that may excite any part of the plant and induce stability problems or just degrade the claimed performance in some way.

It is proposed to adopt and extend one or more of the techniques for air/fuel ratio control presented here and to combine this with other strategies for improved spark timing control that have not so far been discussed here. A significant amount of work may relate to the study of the interaction between the techniques presented for air/fuel ratio control and optimum ignition timing control. It is generally accepted that the peak cylinder pressure should occur at around 12° after top dead centre for optimum power generation. This will vary on a per engine basis and also vary throughout the operating range of a particular engine. By directly controlling the peak pressure point and allowing the air/fuel ratio to be more accurately controlled and varied, the synergistic combination of

the two techniques may yield beneficial results that have not so far been published. In order to obtain the optimum control of both air/fuel ratio and spark timing in an online real-time manner it will be necessary to utilise torque feedback as this is the main metric by which to measure the effect of an online parameter change so that the optimisation may be performed. Torque measurement and estimation is not a new concept and can be achieved through a variety of direct and indirect means and may well require using a combination of some of them. These techniques include spark plug ion sensing (Nielsen & Eriksson, 1998), engine mount load monitoring, crank shaft rotational variance (Rizzoni, 1989) and magnetostrictive torque sensing (Fleming, 1989). It can be assumed that none of the techniques alone have so far been deemed adequate on the grounds that they have not been adopted for use on production vehicles with few exceptions. This may be because of cost and reliability for a particular method such as there may be no perceived *added value* from a customer-sales perspective as fuel economy benefits can already be obtained by purchasing smaller and often cheaper vehicles rather than a more expensive one that is made even more expensive by improved instrumentation.

The rest of this chapter is broken down into several main sections which cover the areas of interest which are those related to dynamometer control, engine calibration and testing, engine modelling techniques, engine instrumentation techniques, and a review of existing software implementations for engine control.

3.1 Dynamometer Operation and Control

The development of an eddy current dynamometer control system has been a required part of the work undertaken during this project. Designing such a system has raised a number of control challenges to which is it has been necessary to consult the literature to find the previous and current states-of-the-art. The technical challenges include:

- Timely control of current/magnetic flux of the eddy current dynamometer
- Effective decoupling of interdependent torque and speed control loops
- Tracking of transient or varying torque/speed set points

It has been found that the body of literature which explains the operation of eddy-current dynamometers actually is found with research into the theory and operation of eddy-current couplings that were in wide spread use around the middle of the last century. The principle of operation and design of the two classes of machine are very similar and the eddy-current dynamometer appears to have been developed as a special application of the equivalent design of coupling.

In a PhD study of transient behaviour in eddy-current couplings, Wright (1972) (an employee of dynamometer manufacturer Redman Heenan Froude Ltd.) states that 80% of (power transmission) couplings produced in the U.K. were of the eddy-current design. No reference or data source was given to support this, but coupling sales quantities will have been known to his employer. Figure 3.1 shows the basic design of the inductor pole coupling, which of the two main types of eddy-current coupling, represents the larger power rated and has closest correspondence to that of the dynamometer. The design consists of a shaft mounted toothed solid iron rotor which has an appearance like a coarse pitched plain spur gear. There is an annulus in the rotor into which a field coil is wound. The rotor is held within a *loss* drum (also of solid construction) with a small air gap clearance between the rotor's poles and the inner surface of the drum, and they are free to rotate relative to each other. When the field coil is energised, a toroidal field is established around the rotor which passes through the air gap into the surface of the loss drum then back through the air gap to the rotor. Figure 3.2 shows the air gap flux distribution relative to the rotor's poles. When there is relative motion between the rotor and drum (termed slip-speed), the flux distribution is *dragged* around the drum so that any fixed position on the drum sees an alternating flux intensity. It is this slip-speed proportional change in flux which induces electrical currents in the drum close to it's internal surface which are known as *eddy* currents. The eddy currents in turn produce their own magnetic field which is aligned in the opposite direction to the rotor field and hence opposes it's motion requiring torque to overcome it. Since the drum is of solid construction, the electrical eddy currents conduct through the material forming a short circuit which produces heat. Thus a part of the input power is converted directly to heat in the loss drum in proportion to the field strength and the slip speed. For the case of the dynamometer, the slip speed is equal to the input shaft speed and all of the input power is converted to heat. Either the rotor or the drum side of the coupling can be used as the input, but the convention is that the drum is used so that it's higher speed assist convection cooling of the generated heat. Also a minor benefit is that if slip rings are used for the field coil then the wear rate will be lower if the rotor is the slower member of the coupling. For dynamometers, the opposite convention is used where the drum is kept stationary and the rotor is connected to the input shaft.

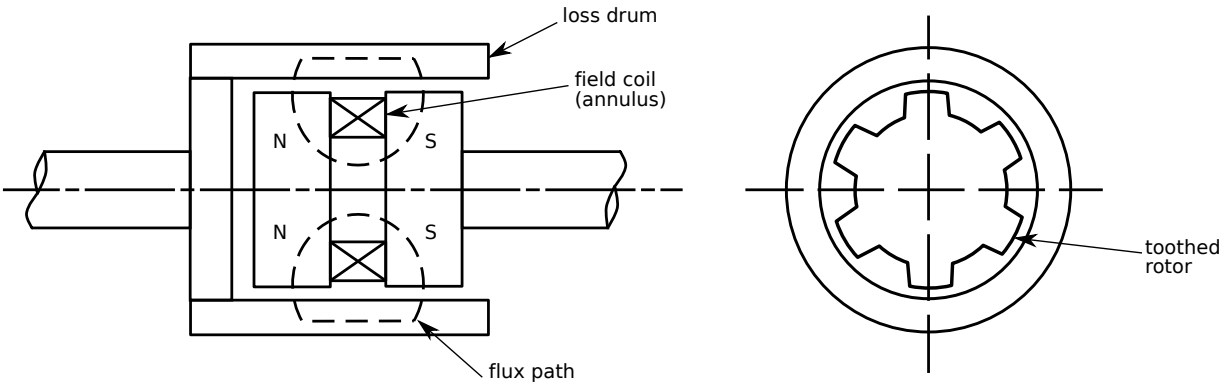

Figure 3.1: Schematic of an eddy-current coupling (Davies, 1966)

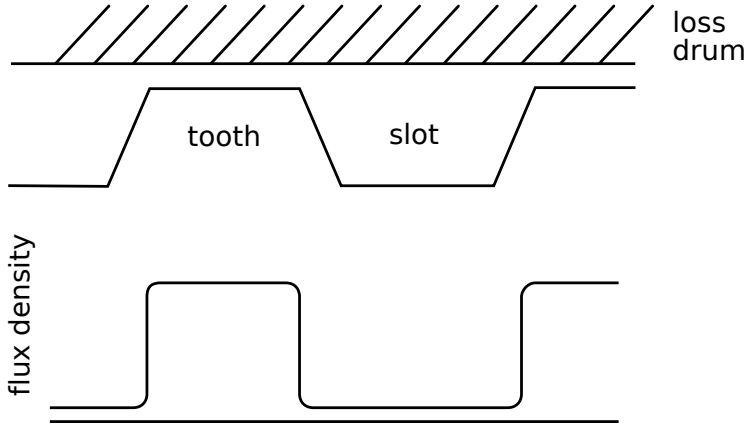

Figure 3.2: Eddy-current coupling static air-gap flux distribution (Davies, 1966)

The magnetic circuit is completely unlaminated. It may be feasible to laminate a part of the stationary iron, this was (and is) not done in practice as it would be difficult and more expensive. Part of the stationary iron must be unlaminated and so must the poles. As a result, these machines normally have a slow torque response. An effective means of modifying the torque-slip curves can be achieved by brazing copper endrings onto the extremities of the loss drum or by coating the internal surface of the drum with a thin sleave of copper. The endring method gives little control over the shape of the torque-slip curve since it only eliminates the end region loss. The copper facing method allows control over the curve shape by control of the facing depth. The addition of copper facing to the loss drum has negligble effect on the transient torque response (for constant airgap) except at very low slip speeds where the copper facing reduces the torque response time by 5-15% depending upon the rotor design. Wright reports that there was no published literature on the transient performance of eddy-current couplings. All of the published literature on the subject deals with the steady-state performance of eddy-current couplings with homogeneous solid drums.

> “Even though the coupling is clearly not a single order linear system, torque response is usually specified in terms of the “torque time constant”, i.e. the

time to reach 63% of the steady-state torque, following a step of rated field voltage. The torque time constant of a commercial coupling ranges from 0.20 s for a fractional kilowatt machine to approximately **10 s for a large dynamometer**."
– Wright (1972)

The majority of eddy-current couplings are manufactured with solid iron rotors and eddy-currents are expected in the loss drum as the principle mechanism of operation of these machines, but eddy-currents are also induced in the other parts of the magnetic circuit (particularly the rotor) during periods of transient flux. The these eddy-currents produce their own field which tends to delay or damp the change (increase *or* decrease) in air gap flux by opposing the main field which created them. Wright describes the potential task of a direct analysis of the damping eddy-current paths as being formidable even for simple geometry due to the interdependence of the inducing flux and the resulting distribution of eddy-currents (a later attempt is made by Jamieson (1968) for an unslotted iron rotor), although the subsequent substantial performance improvements in modern computers and the adoption of finite element techniques may make this a more tangible prospect (an early example for the solid iron rotor being Demerdash & Nehl (1979)).

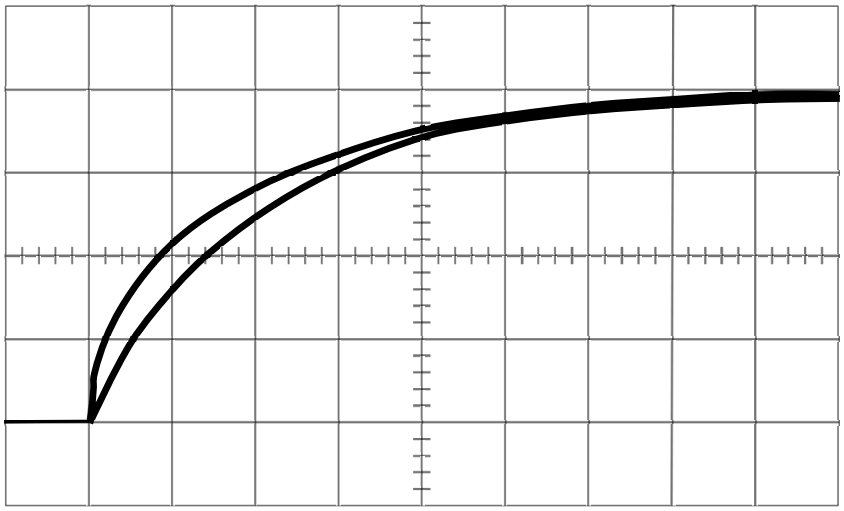

Figure 3.3: Eddy current coupling current (upper trace) and flux (lower trace) response at standstill (Wright, 1972). 20 ms timebase, arbitrary vertical scale

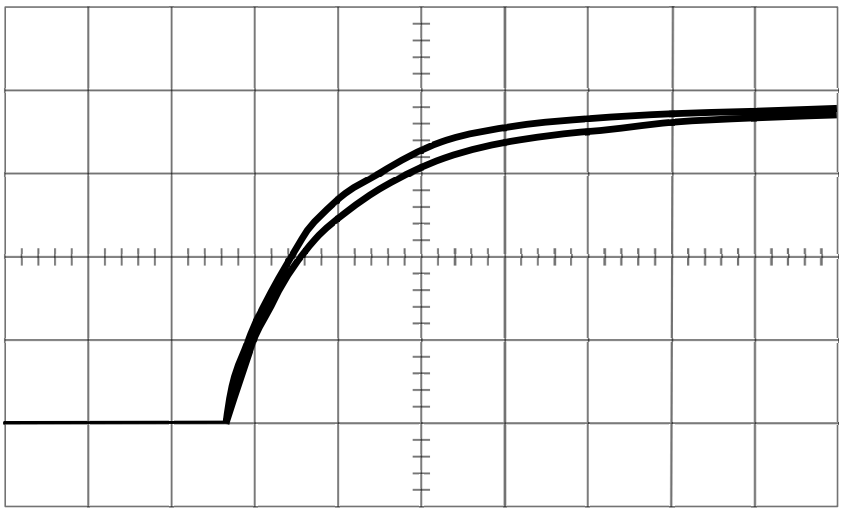

Figure 3.4: Eddy current coupling current (upper trace) and flux (lower trace) response at 100 rpm, 10 Hz pole frequency (Wright, 1972). 20 ms timebase, arbitrary vertical scale

Concerned with establishing an applied change in voltage to transient torque relationship, Wright uses an idea from Fegley (1956) (and others before him referenced therein) to approximate the effects of eddy-current damping by a fictitious damper winding which is magnetically coupled to the main field winding (3.5).

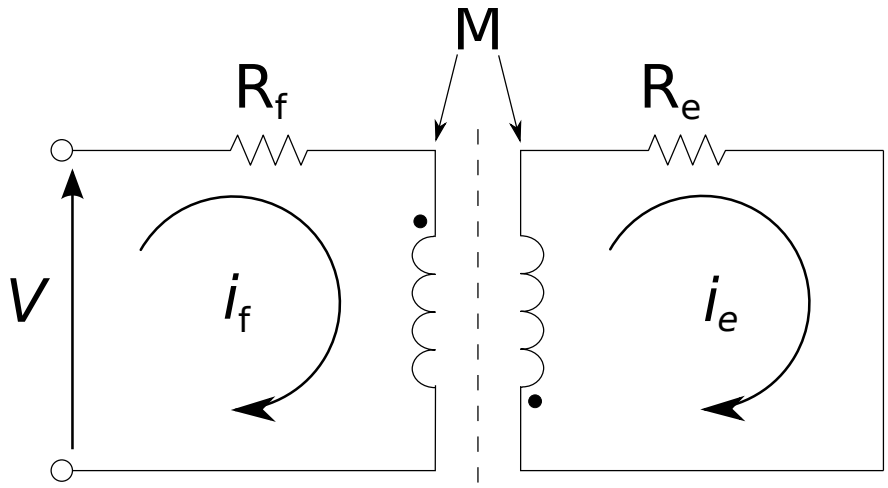

Figure 3.5: Field circuit and fictitious shorted damper winding circuit representing eddy-current path (Wright, 1972; Fegley, 1956)

Fengly states that the analogy holds best for machines having a completely laminated magnetic circuit (as these machines still stuffer from the same eddy-current effect, to a lesser, but not negligible, degree), as in a solid iron section the eddy-current density is highest in the vacinity of the surface (therefore lowest in the centre) creating variable eddy-current density which causes a distortion in the air-gap flux. The reader is also reminded by Fegley and his respondents that the relationship will only hold well for excitation conditions that are far away from the magnetic material's saturation point. A respondent shows data for the predicted flux development alongside measured flux for a

DC machine which demonstrates the fact that the real flux develops more slowly than the estimate. The writer who presented the data proposes that the time constant for both the real and fictitious windings should instead be calculated from the design parameters and mechanical dimensions (respectively) of the machine. For the fictitious winding the technique outlined by Pohl (1949) (who notes eddy-current brakes as an application) might be used for simple geometry, but for a salient pole rotor this must imply forming an estimate of the eddy-current path resistance based upon an knowledge of the actual eddy-current distribution which returns us to the original complexity noted by Wright. Wright is however, slightly dismissive of a detailed analysis of the eddy-current damping effect on the field winding transient current response, stating that it just serves to modify (slow) the flux development time constant without changing it's nature. This may be true, particularly for an unsaturated laminated core/rotor, but for control purposes where field current is the measured variable rather than flux (as it is more easily measured), it would be advantageous to have a transient relationship between current and flux development to allow virtual sensing of flux which in turn would allow either compensated feedback/feedforward or a predictive control technique to be used.

$$k_l = 1 - \frac{M}{\sqrt{L_f L_e}} \tag{3.1}$$

The main result from Fegley (1956) is that for the case where flux leakage is neglected ($k_l = 0$), the initial current step takes place instantaneously, and the response of the field current i_f to an applied voltage can be can be expressed in terms of just the field and damper winding time constants (3.2).

$$i_f = \frac{V}{R_f}\left[1 - \left(1 - \frac{\tau_e}{\tau_f + \tau_e}\right) e^{\left(\frac{-t}{\tau_f + \tau_e}\right)}\right] \tag{3.2}$$

(3.2) implies that when a voltage is applied the field current will rise instantaneously to a value determined by the time constants of the real and fictitious windings:

$$\text{Per-unit jump in } i_f = \frac{\tau_e}{\tau_f + \tau_e}$$

The combined time constant for both windings $\left(\tau_f + \tau_e\right)$ can be determined by plotting the natural logarithm of the transient current reponse against time and measuring the resulting negative and approximately linear gradient. The size of the initial jump in field current can be measured directly from a current-time plot. Hence by capturing a single current transient response, the two field time constants can be determined. This simplified result may be sufficient for control purposes depending upon the amount of

flux leakage the real machine exhibits and whether the step rise time lies inside the control period. A difficulty in applying this techique is that DC couplings and dynamometers usually use solid rotors, not laminated, and saturation is likely to occur in some parts of the magnetic circuit which will distort this simple relationship.

Wright defines a general relationship for the pole flux developed for a given applied voltage and field current as:

$$\frac{\phi_{ac}}{V} = \frac{N\Lambda/(1+A)R}{\left(\tau_l + \frac{\tau_{g_0}}{1+A}\right)s+1} \tag{3.3}$$

$$\frac{\phi_{ac}}{i_f} = \frac{N\Lambda}{1+A} \tag{3.4}$$

where magnetic permeance $\Lambda = \frac{\phi}{NI}$

Wright defines the time constant associated with the field winding as being composed of a coupled and leakage time constants so that $\tau_f = \tau_g + \tau_l$ and derives that following transfer function relationships for the current response to an applied voltage for both the field winding and the fictitious damper winding:

$$\frac{i_f}{V} = \frac{1+\tau_e s}{R_f\left[\tau_e\left(\tau_g+\tau_l\right)k_l\left(2-k_l\right)s^2+\left(\tau_g+\tau_l+\tau_e\right)s+1\right]} \tag{3.5}$$

$$\frac{i_e}{V} = \frac{Ms}{R_f R_e\left[\tau_e\left(\tau_g+\tau_l\right)k_l\left(2-k_l\right)s^2+\left(\tau_g+\tau_l+\tau_e\right)s+1\right]} \tag{3.6}$$

The poles for the two transfer functions are given by solving the quadratic denominator common to both. Letting the two real roots be $-1/\tau_a$ and $-1/\tau_b$ where τ_b is the flux leakage time constant, we have:

$$\frac{i_f}{V} = \frac{1}{R_f} \times \frac{1+\tau_e s}{\left(1+\tau_a s\right)\left(1+\tau_b s\right)} \tag{3.7}$$

Setting the denominators equal to solve the quadratic:

$$\tau_a\tau_b s^2 + \left(\tau_a+\tau_b\right)s+1 \equiv \tau_e\left(\tau_l+\tau_g\right)k_l\left(2-k_l\right)s^2+\left(\tau_l+\tau_g+\tau_e\right)s+1 \tag{3.8}$$

Gives:

$$\tau_a + \tau_b = \tau_g + \tau_l + \tau_e \tag{3.9}$$

and

$$\tau_a \tau_b = \tau_e (\tau_l + \tau_g) k_l (2 - k_l) \tag{3.10}$$

Wright applies the assumption that the flux leakage between the field and damper windings is small so that $\tau_a \gg \tau_b$ which allows τ_b to be dropped from 3.9:

$$\tau_a = \tau_l + \tau_g + \tau_e \tag{3.11}$$

And by substitution back into 3.9:

$$\tau_b = \frac{\tau_e (\tau_l + \tau_g) k_l (2 - k_l)}{\tau_l + \tau_g + \tau_e} \tag{3.12}$$

By substituting for τ_a and τ_b in the damper current transfer function 3.6 and using 3.3:

$$\frac{\phi_{ac}}{V} = \frac{N\Lambda/(1+A)R}{(\tau_l + \tau_g + \tau_e) s + 1} \tag{3.13}$$

Thus if a small flux leakage between the damper and field windings is assumed then the only effect of the damper winding is to increase the overall flux time constant by τ_e.

Wright cites a result from an earlier paper (Davies, 1966) written by his main supervisor that the drum can be assumed to be in a pseudo-steady-state condition during transients. Davies reports that for the loss drum the magnetisation curve the *BH* relationship can be summarised by a piecewise selection of $B = kH^n$ around the *knee* of the curve, then $B = \mu H$ and $B = constant$ for the lower and higher extremes of operation respectively (Davies, 1963, 1966). The flux penetration depth in the drum is affected by excitation frequency which in turn is directly related to the coupling slip speed. Hence the flux will pass from pole to pole though a very shallow surface layer for higher rotational speeds causing saturation localised to the surface layer. Linear theory suggest that torque should be proportional to the square of the excitation current (i.e. $T \propto \phi_{ac}^2$), however saturation in the drum makes this not the case in practise. It is the degree of saturation which is the basis for selecting which of the *BH* relationships describes the current operating point. By defining and index m, transient torque will be affected by flux as:

$$T \propto \phi_{ac}^{\frac{2m}{2m-1}}$$

Wright further concerns himself with attempting to predict the transient torque response to a change in applied voltage, noting that this is directly coupled to the transient flux response which can only lead to an analytical solution for constant $\frac{d\phi_{ac}}{di}$ which corresponds to the lower part of the magnetisation curve which is approximately linear. This implies that the analytical solution holds only for voltages resulting in relatively low excitation currents, but he also states that the error is small for high currents at the low slip speeds which couplings are usually rated for. Dynamometers are used for high slips speeds so would require a piece-wise-linear solution where τ_g is held at a constant for each of the (suggested three) linear solutions.

Koustas & Watson (1984); Koustas (1984) are among the earliest to report a full digital implementation of an eddy current dynamometer controller. They developed a single unit which could can be easily transferred between several dynamometers of different ratings. The controller consists of three 16-bit PC's fitted with data acquisition cards and linked through three bi-directional 16-bit data buses. The controller was implemented in such a way that design variations between dynamometers could be accommodated by using different calibration files for each to retain optimal settings. For DC eddy current dynamometers, the authors cite the disadvantages of no motoring capability and a relatively slow response time, with very low inertia (typically $0.5\,kg.m^2$), compared to DC motor dynamometers. However they are relatively low cost whilst maintaining precision and ease of control.

A commercial thyristor voltage controller excitation unit (BEC 12/45) was used initially with a maximum output of 45 V. This was primarily a voltage controller not a current controller and the authors claim that a slow thyristor response was limiting the load application response of the dynamometer. To improve this the thyristor unit was replaced by a pulsed high voltage current controller which took the form of a DC voltage chopper with current feedback where the mark-space ratio controls the RMS current (no details are provided) from a rectified 240 V AC supply. A switching frequency of 2.5 kHz was used, limited by the switching delays in the power transistors, and the current ripple was maintained <5%. The reported current rise time for this unit was (dynamometer dependent) typically 20 ms. For rapid unloading, reverse polarity is applied to the dynamometer. Although the control of the dynamometer is described in terms of its flux, it was the excitation current which was directly controlled as torque and steady state excitation were assumed to be proportional to flux. The main reason for using flux was so that if torque and speed are mapped into flux the control problem can be reduced to a SISO system without loss of the speed information.

At the time of publication, it was believed by the authors that no existing dynamometer torque controller uses speed as an input so as to avoid the need for multivariable control. Speed was treated as an exogenous disturbance to be rejected by the torque controller. The transient current and flux behaviour was not considered as the excitation unit was

said to be able to achieve the required current within the torque controller period of 40 ms (software adjustable to any multiple of 20 ms), although no data is presented to support this and it is not stated what conditions the excitation unit can achieve this, i.e. for how big a step change in input. An open loop measured flux response is shown and does not reach steady state after 6 seconds. Driving the field coil with a much higher voltage, under control, than is required to maintain a particular steady state flux will reduce this time considerably for small changes in demand. However, a full scale demand change is unlikely to reach the set-point in anywhere near 40 ms even with a high voltage power supply.

An attempt at a theoretical and analytical set of models of varying order for the flux response were made from simplified geometry of the magnetic path and they are shown to make reasonable agreement with the measured flux response of a dynamometer. However, the test conditions are not stated and it is mentioned briefly (in a different section of Koustas (1984) to that which the model is derived) that the second order least-squared fitted flux model (presumably selected as the best candidate model order) was later abandoned due to its inability to accurately provide predictions over the full operating range. Instead a 3-D map relating speed, torque and steady state excitation current to flux was used with a linear interpolation program. The rest of the work assumes flux to be the manipulated input to the dynamometer from which it must be assumed that the required steady-state current to achieve the demand flux is *looked up* for the particular operating point. No direct mention is made of eddy-current damping and its effect on the transient current/flux/torque responses.

Five test run control modes were used from a console user interface:

1. A series of steady-state test points with data-logging at each.
2. A transient tracking mode in which both controllers go through a predefined transient schedule, using preview methods to optimise the torque schedule.
3. A non-tracking version of 2, with no preview for optimisation.
4. A power-speed law such that torque is some predetermined function of speed, using a third order speed polynomial.
5. Speed control by torque. The throttle controller follows a prescribed throttle position whilst the torque controller adjusts torque to achieve the desired speed.

The torque control algorithm was based upon time optimal control using a dynamometer excitation model. For this, a second order model based on the magnetic flux (rather than torque) was selected to simplify the model derivation. For large set point changes a

bang-bang control mode is used until a target range is achieved, then within this range a pseudo-singular mode is activated until the actual target is achieved.

The torque sampling and control loop frequency is 50 Hz. The torque signal was measured using a strain gauge load cell and all the dynamometers are undamped for fast response. Due to the inevitable noise from vibration, the torque signal is first passed through a low-pass filter with a cut-off frequency of 200 Hz and then integrated over the 20 ms sample period. The torque schedule is specified every 0.5 s. Non-tracking modes use a single point look ahead and a four point (2 second) preview for tracking. A stepping motor forms the actuator for the engine governor and is directly controlled open-loop by software generated pulses. The stepper motor scheduling is described in detail, but essentially the fastest possible velocity profile, or acceleration/deceleration curves without the motors missing any steps, is used directly by the time-optimal speed controller.

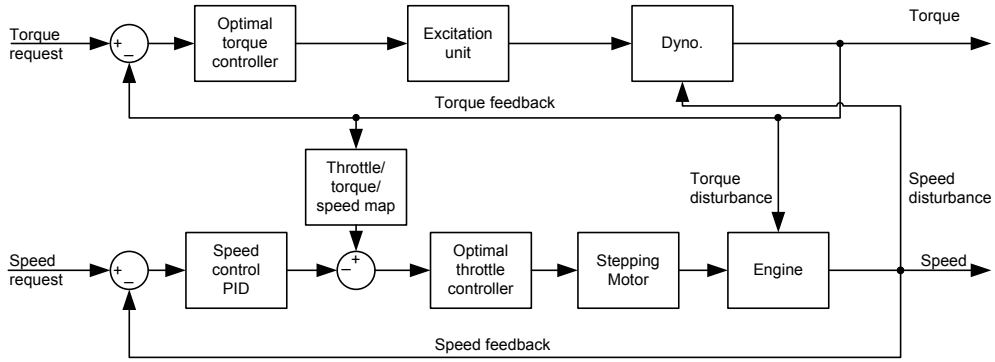

Figure 3.6: Block diagram of modes 1 and 3, no preview. (Koustas & Watson, 1984)

Figure 3.6 shows the schematic diagram of the control system in modes 1 and 3. This shows the software based torque control with torque feedback, the dynamometer's speed variation is treated as an external disturbance, since this alters torque for a given excitation. For speed control a PID controller with velocity feedback is used and the PID output is compensated for the load point using a throttle/torque/speed map.

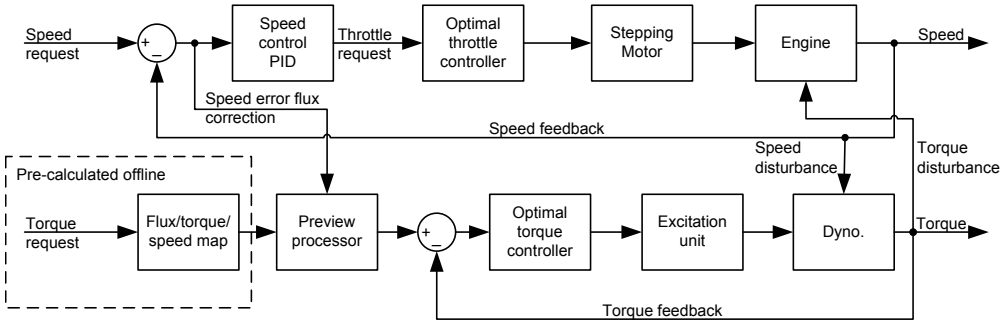

Figure 3.7: Block diagram of mode 2, torque and speed tracking with preview. (Koustas & Watson, 1984)

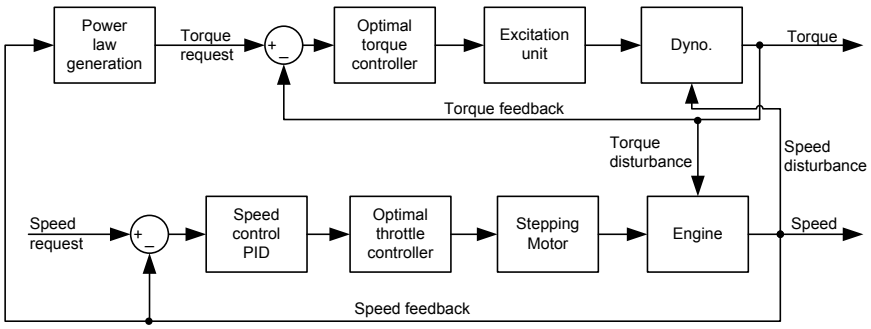

Figure 3.8: Block diagram of mode 4, power-speed law. (Koustas & Watson, 1984)

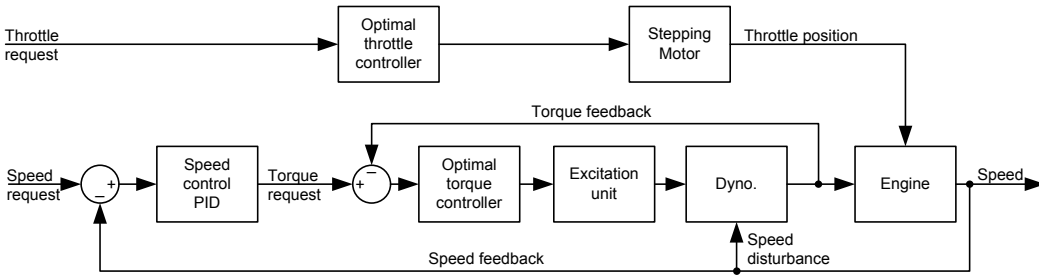

Figure 3.9: Block diagram of mode 5, control of speed using torque. (Koustas & Watson, 1984)

Mode 5 (Figure 3.9) uses torque to control speed and the time optimal speed controller is replaced with a conventional PID controller with a 40 ms control interval and is tuned under step load application at constant throttle. The optimal torque controller requires a steady-state flux map of the dynamometer as input. Mapping is performed as a 3D array of speed, torque and excitation (considered proportional to flux). The map is then used by the controller through linear interpolation.

Other work in this area includes a generalised predictive controller (GPC) (Noble et al., 1988; Beaumont et al., 1988) of Ricardo. Also King et al. (1991) presents a pole placement self-tuning controller for the torque control loop.

3.2 Engine Testing and Calibration

The aim of an adaptive engine controller is to replace some or all of the calibration effort of a conventional one, and/or to optimally track changes in the prevailing conditions whether that is due to internal plant parameter drifting or external climatic change. A further application of engine condition monitoring may arise naturally as a consequence of having a plant model available in the control strategy whether any physical correspondance from the model to the plant is possible or not. When a sudden or unexpected parameter change is detected then this may be indicative of a fault or the onset of a fault (Krishnaswami & Rizzoni, 1995) and allow prognostics and diagnostics to take place.

For work to commence on a control strategy it is necessary to have a baseline calibration with which adaptive algorithms may be compared. Rapid determination of the calibration maps is desirable. For this to be realised there must be some level of automation in adjusting the engine to the various set points and acquiring data. Collection of data is also necessary for off-line model fitting and model validation. Traditional steady state mapping is very time consuming especially if it is performed manually, so it must be performed using an automated procedure, some of the history of which is considered here.

3.2.1 Automated Engine Mapping

Hochschwarzer et al. (1992) has developed and tested an automated engine calibration system. They state that there are large number of control elements in engine control software and they often contradictory requirements, the determination of the best compromise is a complex and labor intensive task.

The setting of ECU calibration variables is achieved using a ROM emulator plugged into the EPROM socket and dual-ported RAM is used to link ECU to PC for real-time monitoring. Constrained optimisation is achieved using Lagrangian techniques and the engine protected from damage by calculation of safe operating regions and monitoring of exhaust temperatures, cylinder pressure and knock. The operator can define an allowable range for each control characteristic, but this cannot gaurantee that an unfortunate combination of control parameters does not cause a limit violatation. The optimisation module is able to cancel a parameter variation if a limit threatens to be violated (such as knock). A custom programming language was developed that they called OPTI, which allows a test scenario to be described in terms of the manipulated variables, constraints and the performace objective. It is not stated, but this implementation would require a dynamometer and torque speed controller capable of reaching and maintaining the desired test operating points independently of whatever calibration exercise was being performed. The engine map calculator produces fully populated maps from a relitively low number of optimised settings. It performs interpolation and extrapolation using a spline function from the optimisation results and provides filters and confidence checks. They claim that the resulting engine maps compared favourably with manually determined maps, with a torque improvement for one cited case.

3.2.2 Sweep Mapping and Automated Engine Testing

Steady state mapping has been commonplace for optimising IC engine operation. Engine data are logged after a predefined time has elapsed and this is repeated systematically for

each operating point that is required. Conversely, the sweep method continuously moves the engine through its operating envelope without dwelling to speed up data capture. This method is not straight forward as thermal and mechanical inertias associated with the engine make instantaneous measurements different to steady state ones and also as the settling time of the instrumentation used must be taken into account. Sweeping the engine through its torque range at constant speed helps to reduce the effects of inertia, rather than allowing it to accelerate through its speed range as is often done to produce torque and power curves.

Research at Bath University in collaboration with Cosworth Technology (formerly, now MAHLE Power Train division) compared the accuracy and repeatability of the sweep approach under experimental conditions, with that of steady state testing (Ward et al., 2002). The authors reported that the errors can be reduced to satisfactory levels by modelling the engine and instrumentation responses. The sweep mapping technique required intensive data processing and test bed sophistication over conventional steady state mapping, and showed the potential to give accuracy comparable to steady state testing, but with much reduced mapping times. The response time of some equipment (particularly relating to air, fuel and emissions) was found to be the limiting factor for the maximum sweep speed.

3.2.3 Model Based Calibration Tools

Lumsden et al. (2004) considered the use of proprietary model based calibration tools for use with what they have termed *complex engines, i.e.* those that have added degrees of freedom created by variable valvetrain mechanisms (lift and timing), variable compression ratio, stratified direct injection, variable intake geometry and charge direction control ability. Such additions to the conventional engine design can increase the calibration effort by orders of magnitude and it may become unfeasible to map the entire operating envelope systematically. For these engines it is necessary to either accept suboptimal operation in known stable regions or employ Design of Experiment (DoE) techniques that have more commonly been applied to the chemical processing industry to improve data quality and reduce testing time. The use and interpretation of results created using DoE tools is not straightforward for engine management and work is ongoing to improve these tools. DoE tools are also used to reduce the amount of experimentation and perform optimisations.

They first tested a software tool by Umetrics called *Modde* which allowed only simple constraints to be applied to the test regions. It is suggested that an n-dimensional hull is required, the construction of which is not a simple task. A typical task might include attempting to minimise BSNOx (brake specific nitrous oxide emissions) at a particular operating point whilst maintaining acceptable combustion stability. An acceptable result

may be obtained but it does not deal with practical considerations such as the need for a smooth transition in control parameters along any given speed-load trajectory. The creation of the subsequent map becomes at least as difficult as the original model construction using DoE. The Quadratic models produced by the software proved useful for prediction and optimisation of the NOx and CoV (combustion stability) responses, but they were not found to be adequate for BSFC (brake specific fuel consumption) or THC (Total Hydrocarbon emissions) modelling.

The Mathworks MBC (Model-Based Calibration) Toolbox was used to develop experiments which allow cubic models to be produced. Tools for directly populating engine calibration maps are included when using Matlab, unlike standalone DoE software. The cubic models showed a better predictive ability than the quadratic models, in particular for the BSFC and THC where it was thought that quadratic model shape was just not suitable for representing the complexity of these two parameters to the desired accuracy. Due to a clustering of data points the model was reworked and an additional radial basis function (RBF) model was tested.

It is stated that it is not yet common for engineers to have complete trust in automated optimisation routines for high numbers of dimensions and that flexible plotting functions are essential so that some opportunity for judgement is still possible. Cosworth Technology have previously used motoring dynamometers only as these allow the engine to keep running during extremely poor combustion that might otherwise cause a stall condition. Being able to keep the engine running is essential during automated testing as an engine stall would cause the experimental program to abort and would require manual intervention to restart it each time. Recent work now allows them to use conventional absorb-only dynamometers presumably by making using of DoE constraints as well as monitoring transitions between operating sites. They are also investigating the use of Genetic Algorithms (GA) to run optimisations in the Matlab environment. By using GA the effective number of evaluated points increases as an exponential function of run time and they hope that this will allow them to efficiently find local and global optima.

3.2.4 Capturing Cylinder Pressure Data in Real-Time

Cylinder pressure data is required to establish effective ignition spark timing and to validate techniques that use pressure measurement or estimation as a means of indicating output torque. It is also useful for misfire detection when starting and operating close to regions of increasingly unstable combustion in an automated experimental system. Early cylinder pressure data capture systems were not fast enough to gather data form a single cycle so they were triggered at fixed a crank angle which was varied with time to build up a time averaged chart of cylinder pressure. It is now possible to obtain transducers that have a sufficient response time and A/D converters which can sample at a high

enough rate such that complete pressure traces can be captured for consecutive engine cycles. However, capturing data in this way presents a storage (and interpretation) problem as large amounts of data can be generated. Data windowing and reduction techniques can be used to discard and reduce data points where high density is not required. This can dramatically reduce the amount of data requiring storage.

Perkins Technology, manufacturers of diesel engines, pioneered their own data capture system at a time when the commercially available systems were inadequate. A 12-bit 500 k samples/second A/D conversion system with an onboard microprocessor and RAM to handle the acquisition, with direct output of data to an oscilloscope, X-Y data plotter and tape storage. It was also coupled to an IBM PC through a RS-232 serial connection and an 8-bit bus which could be used interchangeably. The PC was deemed to be too slow (having a 6303 processor at 4.9152 MHz) to manage the data collection directly and so it was decoupled by having an onboard processor which was programmed using FORTH and sections of assembly language code. There must have been considerable cost and complexity in this custom system and its low level design, but it was considered worthwhile as it enabled them to visualise cold starting performance and conclude that only the charge in one cylinder of four needs to self ignite under starter motor cranking and the other cylinders will follow on.

3.2.5 Reduction of Cylinder Pressure Data

Capturing cylinder pressure data can result in very large file sizes. By way of an example, the aquistion from one cylinder of a four-stroke engine running at 3000rpm, sampling at 1 degree crank angle resolution to 16-bit precision results over a 240 degree interval, results in a data rate of 703 kB/min or 41.2 MB/hour. Reducing the sample interval to 0.5 degrees whilst running the engine at 6000 rpm and sampling the pressure of all eight cylinders of a V-8 engine, increases this to 1.29 GB/hour. Whilst recent advances in manufacture of harddisks mean that storage of over a Terabyte is easily affordable, it is still desirable to keep the amount of data storred to a minimum. One technique to reduce the amount of data storred is to use a variable resolution technique such as that presented by Brunt et al. (2000). Figure 3.10 shows how the variable resolution scheme employed for data reduction against an actual cylinder pressure curve. It is claimed by the authors that by dividing the pressure curve into regions of interest determined by mean cycle characteristics in angle space, allows upto 90% of the original data to be discarded. From Figure 3.10 the amount appears to be closer to 75% for the angular windowed region shown. The figure of 90% may refer to a comparison with continous sampling of a four-stroke cycle or the fact that derived statistical quantities no longer have to be stored alongside the pressure data as they can be recontructed from this data at a later time the reduced data is still representative of the original data.

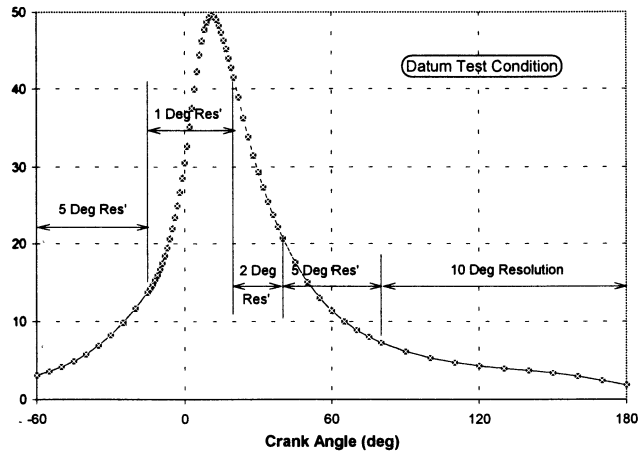

Figure 3.10: Mean cylinder pressure curve for datum test condition for reduced format resolution, showing the stored data points and the interpolated curve (Brunt et al., 2000)

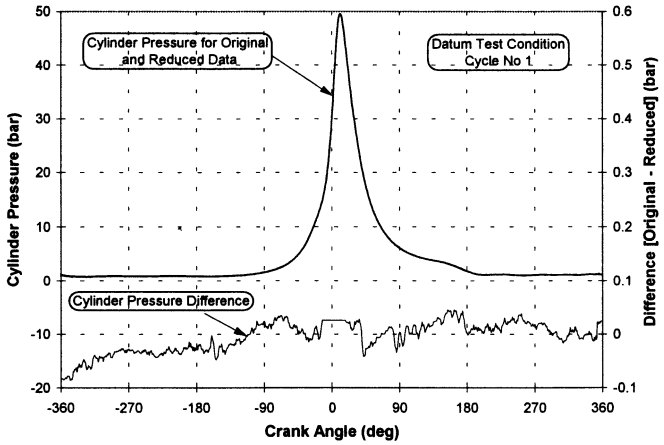

Figure 3.11: Typical comparison of cylinder pressure vs. crank angle curves for original and reduced cylinder pressure, difference is shown in lower region of the plot (Brunt et al., 2000)

Figure 3.11 shows a mean pressure curve and the pressure difference between the original data and the reduced data against crank angle. This shows that the reconstructed mean from the reduced dataset differs from that derived from the original data by at most 800 mbar and by only 50 mbar in the central -90 to +90 degree region of interest where the main pressure rise has occurred. Figure 3.12 shows how the data reductions technique has effectively been filtered and introduced a small offset (of at most 50 mbar) to the original data at a region peripheral to the main pressure rise.

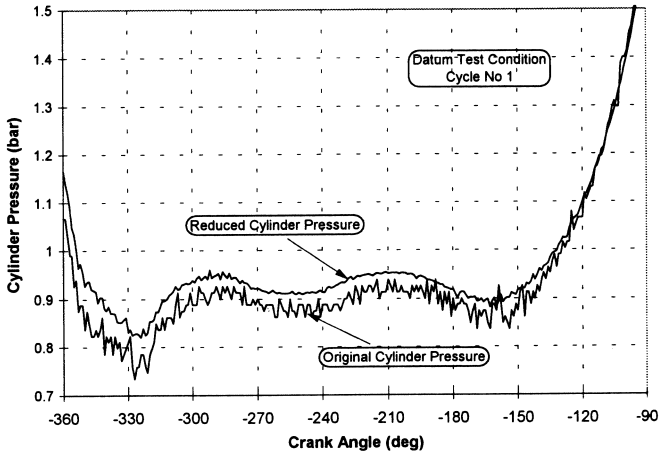

Figure 3.12: Comparison of reduce cylinder pressure and origninal data (Brunt et al., 2000)

3.3 Engine Modelling

This section covers some of the many models which have been proposed for representing engine dynamics either to allow study of behaviour without the need for the physical engine, or for direct inclusion within a control scheme to improve it's performance.

Why Model?

"Essentially, all models are wrong, but some are useful."

– Box & Draper (1987)

A very brief summary of the of the rationale for the use of modelling for engine control is as follows:

- Models can be used for off-line development of controller.
- They can be used for verification.
- They can be used for parameter estimation in a control loop or can be used directly as a control mechanism if converted into code either manually by a programmer or using by a block diagram and an automatic code generator.
- Models vary in complexity and use.
- More complexity usually means that it is harder to generalise and is specific to a particular engine. As the number of parameters increase the effort require in obtaining valid value for the model's coefficients and constants increases.

3.3.1 First-Principle Phenomenological Modelling

The overhead in representing the physical plant is considerable and is only useful for engine design and conceptualisation where a complex model is not needed, but a detailed model of the aspect of design under scrutiny is. The overhead is too high for control and may deal with parameters that are immeasurable on a production engine and identification and validation can be difficult to impossible to achieve. Mean Value Engine Models (discussed in the next section) are usually phenomenological, but offer some simplifications to make them more usable in practise.

3.3.2 Mean Value Engine Model (MVEM)

Mean Value Engine Models (MVEMs) describe the main dynamic processes as a function of the most significant engine variables, and were pioneered mainly by the work of Hendricks (Hendricks & Sorenson, 1990), building upon early published work on them such as Dobner (1980); Dobner & Fruechte (1983) and Aquino (1981). Hendricks has given much attention to filling dynamics (Hendricks et al., 1996; Chevalier & Hendricks, 2000) throughout his work, with particular attention paid to validity during throttle transients (Chevalier et al., 2000) as this offers the potential to improve open-loop transient emissions. MVEMs have been used to verify engine control strategies (Silviero et al., 1995) and also have been used to directly control engines as they are computationally simple enough for real-time computation, however they are limited by approximations made during their design and identification. Heat transfer from the intake manifold is, for example, not accounted for and correction factor maps have to be manually tuned. These static maps are used to account for the non-linear dependence of model parameters such as the fuel evaporation time constant. MVEMs take some time and experimentation to fit to a particular engine and do not take long-term temporal factors, such as engine component wear and contamination, into account. Classical adaptive strategies have been proposed around MVEMs without these underlying problems having first been resolved. Despite the limitations outlined, MVEMs represented a leap forward in engine modelling in that offline software simulations can be formulated and run to a reasonable level of accuracy with only a moderate effort required to tune the model to approximate an actual engine. Control strategies can then be tested without going to the full time, effort and expenditure of creating a hardware implementation.

An overview and detailed explanation of MVEMs are given in Hendricks (1997, 2000) and offer an excellent starting point for the interested reader who has just come across them and wishes to understand more about them.

Melgaard et al. (1990) describe an identification process for MVEMs using maximum likelihood estimation, and gives particular attention to the identification of the major time constants of the engine. Data is collected in a time-based fashion rather than in the angular domain and a pseudo random binary sequence if imposed upon the normal demand signals. Although the work was carried out for the purpose of calibrating a MVEM for offline use, the authors state that the noise estimates produced are also useful for construction of Kalman filters for condition monitoring or control applications. The identification results themselves also have potential for condition monitoring and control, although the MVEM is in a continuous form which limits its use outside a simulation environment without further work to discretise and optimise it for implementation on a microcontroller.

3.4 NARMAX Models

NARMAX is an abbreviation for Nonlinear AutoRegressive, Moving Average with eXogenous inputs. Many systems that have nonlinear behaviour may be modelled by a NARMAX model structure. This structure describes both the stochastic and deterministic behaviour of the system. It models the input-output relationship of the system as the non-linear, but linear-in-the-parameters (LIP), difference equation of the form shown in (3.14),

$$y(t) = F\left(y(t-1),...,y(t-n_y),u(t),...,u(t-n_u),e(t-1),...,e(t-n_e)\right)+e(t) \tag{3.14}$$

where y is the output, u is the controlled input (i.e., exogenous variables), and e is the noise input. F is a nonlinear mapping, which may include a variety of nonlinear terms, such as terms raised to an integer power, products of past inputs, past outputs, or cross-terms. The maximum number of terms in a NARMAX model depends on the maximum lag on the inputs, the output, and on the error, the number of the inputs, and the maximum non-linearity order. Even for moderately complex models the number of candidate terms becomes very large. However, the actual number of terms needed to fully describe a NARMAX model may be relatively few, therefore a subset of the total possible candidate terms has to be selected, in the process of structure detection, to construct what is often termed a *parsimonious* model. The danger of using a non-parsimonious model is the same as that of an artificial neural network with too many layers or nodes. A model with a higher order or more terms than the underlying system's structure will attempt to reproduce noise.

The main difficulty with constructing a NARMAX model is that the structure of the model must either been known in advance from *a priori* or expert knowledge, or estimated for a given set of data. The most widely used parameter estimation method for NARMAX models is the *extended least squares* (ELS), which addresses the bias problem by modelling lagged errors to obtain an unbiased parameter estimate.

To assist with the identification of what is normally an initial highly over-parameterised model, a bootstrap-based algorithm presented by Kulreja et al. (1999) has been proposed that consists of structure detection as well as parameter estimation. The paper assumes the order of the model is known which might not always be the case if little is known about the underlying structure of the system to be modelled. If the order of the model is not known then the order can be estimated from frequency response functions or by using the bootstrap method to test different model orders. The frequency response method has been reported by Åkesson & Sällberg (2003) as being the most reliable to estimate the model orders.

The bootstrap method, introduced some years ago by Efron (1979), is a numerical procedure for estimating the distribution of statistical parameters which requires few assumptions, and can be used in situations where regression is not valid. The method has regained popularity in recent years with the widespread availability of high performance computers required for its repetitive nature. The only conditions needed for bootstrap methods are that the errors are independent, identically distributed, and have zero mean. For the usual method of regression analysis, accurate estimates of the parameter variances are needed, which are difficult to obtain unless the model structure is correct. The structure detection problem is to select only a subset of the candidate terms which best describe the output. Several methods for NARMAX structure detection have been proposed including hypothesis testing of the differences between means via *t-test*, stepwise regression and Korenberg's orthogonal structure detection routine. All of these methods can fail in nonlinear system identification for various reasons.

The application of the bootstrap method to structure detection involves first computing a series of parameter replications, then forming percentile intervals for hypothesis testing where the significance of the parameters is determined. Bootstrap data is formed by estimating the residuals of the identified model. These residuals are then re-sampled with replacement data and added to the predicted output to generate bootstrap replications of the output. A number B of bootstrap data sets are generated to estimate B bootstrap parameter replications. The significance of each parameter is determined by checking if zero lies in its interval, if so the parameter is rejected, as illustrated in Algorithm 3.1.

More recently a Matlab toolbox called *detectNARMAX* (Shafai et al., 2003) has been developed to automate the Bootstrap procedure for NARMAX models. The authors have developed a graphical user interface that allows the users to keep track of both the

Algorithm 3.1 The BSD Algorithm

1. Compute an initial estimate of the unknown parameter vector and estimate the residuals
2. Generate B bootstrap data sets and compute the bootstrap parameter replications
3. Form percentile intervals for each parameter by ranking estimates from the B parameter replications in increasing order
4. Estimate the upper and lower bounds of each parameter's confidence interval for a desired level of significance
5. Determine if zero lies in the interval of each parameter in the vector
6. If zero lies in the interval for any parameter(s) remove them from the regression
7. Compute the new estimate of the parameter vector and residuals
8. Go to 2 until convergence

structure detection process and the parameter estimation, which they consider to be most useful in the case of a higher model order with a large number of candidate terms. The program also allows the user to make a pre-selection of candidate terms that may be known in advance from *a proiri* or expert knowledge of the system. A more detailed assessment of the bootstrap method for structure detection was presented and it was compared using Monte-Carlo simulations to the *t-test* method to which it compared favourably.

Diesel Generator NARMAX Representation A relevant application of NARMAX identification was performed by one of the original proposers (Leontaritis & Billings, 1985a,b) who used a difference equation to predict the output of a 4.6 MW 12-cylinder industrial diesel generator (Billings et al., 1988). At the time of its publication, virtually no practical NARMAX applications for real industrial processes had been reported. Numerous experiments were conducted on the diesel generator but only a limited amount of data was recorded in a suitable form for analysis. The sample time interval of 5 seconds was found to be too small so the data was decimated to retain only every 100^{th} point. Two data sets were used, one to estimate the model, and the other to validate the model against. Although the resulting model showed good prediction against the data set (Figure 3.13) it was not subsequently tested against any other data sets collected. By the author's own admission it was not known if the data sets were fully representative of the engine's whole range of operation and excitation.

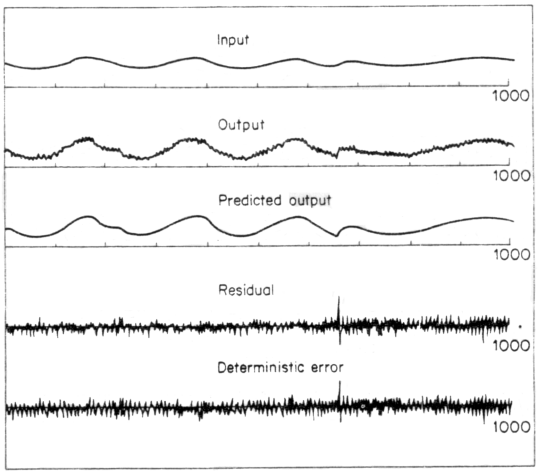

Figure 3.13: Estimation results using NARMAX input-output model compared with the original data set

Idle Speed Controller Designed from a NARMAX representation An engine idle speed controller has been developed Glass & Franchek (1999) using a NARMAX model of the underlying system to help determine an improved controller transfer function. The difference equation (3.15) is an example NARMAX model:

$$y(k) = c_{01,1}y(k-1) + c_{01,2}y(k-2) + c_{02,1}y(k-1)\,y(k-1) + c_{10,1}u(k-1) + c_{20,1}u(k-1)\,u(k-2) \tag{3.15}$$

The subscripts represent the order of the contribution of past inputs, the order of the contribution of past outputs, and the coefficient for a specific regressor structure, respectively. The actual model used for the resulting controller was not given. A statistical evaluation process uses an orthogonal estimator to identify key regressor terms out of the many possibilities. An example of this is a cubic NARMAX model using three past states having 99 different combinations of terms. The model is linear in the coefficients and they can be identified using the least squares method, as with a linear model. The NARMAX least square problem can be formulated as (3.16) where $y(k)$is the model output, $p_i(k)$ are the regressor terms and i are the coefficients for each regressor term and $e(k)$ is the noise model of the system.

$$y(k) = \sum_{i=1}^{M} p_i(k)\,\theta_i + e(k) \tag{3.16}$$

The noise term $e(k)$ has to be minimised so that it is uncorrelated with the input and output signals. To allow the NARMAX model to be used in a controller frequency domain a describing function of the model was developed. The fundamental component

of the system output due to a single frequency harmonic input was considered and higher-order harmonics were neglected due to their much lower amplitude. A harmonic balancing method (Nayfeh & Mook, 1995) was used to recover the fundamental frequency content of the non-linear system response. The controller was designed using a loop shaping approach (Jayasuriya & Franchek, 1991) developed by the authors.

A Ford 4.6 L V8 fuel injected engine was used to test the controller algorithm. The idle speed was controlled using the by-pass idle air valve (BPAV) which at the time was standard on many Ford engines. This electric solenoid valve controls the idle air flow into the engine when the throttle is fully closed. The engine speed was measured using a high resolution optical encoder. The pulse train was passed through a voltage to frequency converter and low pass filtered at 25 Hz to avoid aliasing. A sample rate of 250 Hz was used to capture the engine speed. The controller was implemented using Real Time Workshop C code generator, but the details of the hardware implementation were not given. A disturbance step load of 20 Nm was imposed on the engine by saturating the power steering pump. The controller was set to limit the engine speed deviation to 100rpm and was set to maintain 700 rpm that is also the set speed for the original Ford controller. As can be seen from Figure 3.14, the controller compares favourably to the original production ECU showing both an improved time response and that the drop in engine speed is exactly bounded to the designed limits.

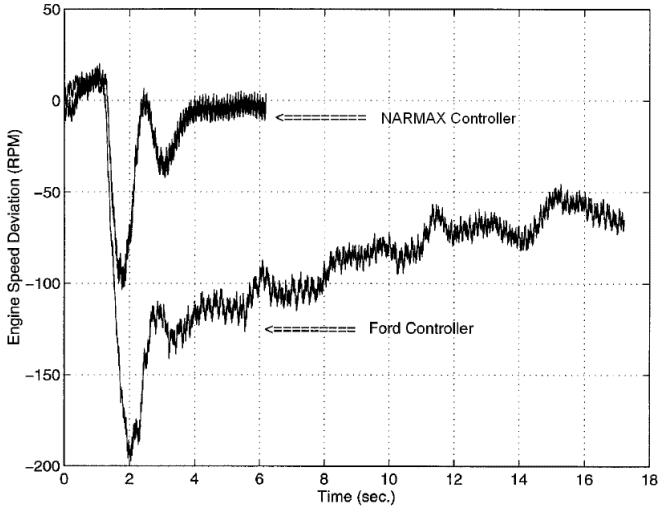

Figure 3.14: Comparison of the experimental response of on-board Ford idle speed controller with a NARMAX controller to a 20 Nm step load torque disturbance

3.5 Other Models

3.5.1 Sliding Mode and Constant Gain Extended Kalman Filter Models

(Kaidantzis et al., 1993) build upon earlier work (Hendricks et al., 1992) which uses a

constant gain extended Kalman Filter (CGEKF) to estimate air mass flow into an engine in combination with a mean value model to estimate the fuel film dynamics. The emergence in the literature of sliding mode observers prompted investigation into a comparison with the CGEKF estimator due to claims of high levels robutness in the literature. An aggressive driving scenario was used for the tests, and it is shown that it has a degree of robustness, but not to a significantly higher level than the CGEKF based observer. It is also claimed that the mode of operation of the sliding mode observer is inherently noisy which degrades its performance when compared to the CGEKF. The authors note that both types of observer can be implemented on microcontrollers in use at the time of publication, so the comparison is one of performance rather than a particular implementational advantage.

3.6 Torque Estimation Techniques

3.6.1 Torque and IMEP Estimation using Crankshaft Rotational Frequency Variation

Rizzoni (1989) has presented a technique for estimating the indicated engine torque by relating cylinder pressure deterministically to net engine torque through the geometry and dynamics of the reciprocating assembly. A passive second-order electrical circuit model with constant parameters is used. It is claimed that the experimental results confirm the validity of the model over a wide range of engine operating conditions, including transients.

Lida et al. (1990) also present an indicated mean effective pressure (IMEP) estimation technique, from flywheel angular-speed fluctuations and has claimed high accuracy at medium to low speeds engine speeds.

Wang et al. (1997) has developed a crank-angle-based model to design a nonlinear event-based observer to estimate the indicated engine torque using one or more measurements of crankshaft angular velocity. They have used crank-angle based implementation for the observer, because the crankshaft angular velocity signal can sampled at discrete crank angle intervals to give more uniform signal than previous time-based approaches and so that the processing of the velocity signal is performed at in a way which is synchronised to the torque pulses being produced by the engine. The test results presented suggest a good correspondence with torque calculated from measurements taken with a cylinder pressure sensor for both steady-state and transient operation. The authors emphasise the benefit of the sliding mode approach particularly for the transient case where a non-linear model would otherwise be required.

3.6.2 Torque Estimation using Engine Block Angular Acceleration

Ball et al. (2000) examines the possibility of detecting misfires via measurements of the angular acceleration of the engine block. They took measurements on a production 4-cylinder engine and modeled it as a single degree of freedom torsional oscillator. The torque waveform was estimated then compared to the indicated torque estimated from cylinder pressure measurements. Their results indicated the results were most reliable at low frequency, but at high frequency preformance was degraded due to modelling limitations and the non-rigid behavior of the block at high engine speeds. From these results it appears that this techique may be useful as another way to detect misfire, but is of limitied practical use for acurate torque estimation for simimlar reasons as crankshaft rotational frequency variation measurement.

3.6.3 Torque to Cylinder Pressure Correlation

Larsson (2003) introduces a parametric crankshaft model based approach for controlling the individual cylinder peak pressure position from the torque data a crankshaft based torque sensor (ABB Torductor) and a parameterised pressure model to reconstruct pressure for peak-pressure-position (PPP) determination and control (Larsson & Schagerberg, 2004).

3.7 Air-Fuel Ratio Control Strategies

Current emissions regulations has meant the normal control objective is to hold the air-fuel ratio near to a constant value assumed to be near stoichiometric, but in practise is an optimum for efficient gas conversion in the catalytic converter. A problem with air-fuel ratio strategies that are not constrained to a constant air/fuel ratio (such as lean-burn engines, or motor sport and off-road applications) is that there is no direct method for the driver to convey the required performance, for example if the maximum possible power (hence acceleration) is required or best fuel consumption. This is sometimes inferred from the absolute accelerator pedal position or its rate of change, but the driver is not necessarily aware of this and so may not adopt the required strategy to obtain best economy or power. Some vehicles have featured a power/economy selector which may change the final drive ratio of the gear box, or directly affect some engine parameter. More modern vehicles also have featured a *sport* switch which tells the ECU and associated systems, such as transmission and body control units of the driver's intentions. None of these systems readily allow the driver's performance requirement to be altered dynamically in response to changing road conditions.

The section looks at some of the model based approaches to air/fuel ratio control that have been documented in the literature. To the authors knowledge, none of these techniques have been used in production or aftermarket engine controllers despite there being some degree of promise and benefit in their approaches, such as the removal of the need for an initial calibration map, and their ability to cope with parameter drift and wear effects.

3.7.1 Fuzzy Logic

The use of fuzzy logic for fuel-injection control has either been less well explored in published literature, or less successful than other techniques. Lee et al. (2003) describe a fuzzy controller developed for a Bosch Suffolk single cylinder 2-stoke 98cc engine. The results were inconclusive as the benefit of fitting fuel injector was not reported and only the resulting fuzzy logic controller results were shown. Small single cylinder engines that are used on lawn mowers usually have carburettors that are single-jetted and don't correct the non-linearity caused by the characteristic of the venturi. In practise this forms an natural speed and power limiting mechanism as the air/fuel ratio leans towards high flow rates. A simple look-up table may have worked just as well or even better than the fuzzy solution.

Chamaillard & Perrier (2001) present a preliminary investigation into the use of fuzzy logic and perform and evaluation against an MVEM simulation. More work is needed to establish the validity of the technique on an actual engine.

Passaquay et al. (2001) describe the torque based control of air/fuel ratio using a control structure a gasoline direct injection (GDI) engine. They propose an engine controller which uses fuzzy logic to define piecewise linear controllers. A stability analysis of the engine is shown, based on a bifurcation diagrams.

3.7.2 Sliding Mode Observer

Choi and Hedrick have proposed a sliding mode observer fuel injection control scheme initially in Cho & Hedrick (1988), then again a decade later in (Choi & Hedrick, 1998). The sliding mode approach uses the switching of the exhaust gas oxygen sensor (narrow band type) to estimate the rate of airflow into the cylinder. A simplified model that uses a lumped parameter for both the lag and propagation time delay of the oxygen sensor feedback and essentially consists of a three state model for the controller design. The first is the mass of air in the intake manifold, the second is the engine speed, and the third is the fuelling and its dynamics. There are delays incurred from intake to torque and spark to torque. Many states are not modelled (such as exhaust pressure, exhaust gas

recirculation (EGR)), but the authors claim that it holds up well when compared against more vigorous and complicated models. This attempt improved upon previous sliding mode schemes by reducing chatter but still performed relatively badly during fast throttle transitions. The choice of feedback gain was the limiting factor as a value that guarantees stability and minimises chatter holds the scheme suboptimal for fast engine dynamics.

Cho & Oh (1993) also report a variable structure based scheme for controlling SI engine air/fuel ratio.

3.7.3 Event-Based Observer

Chang et al. (1993) have developed a discrete non-linear crank-angle based SI engine model for the design of AFR control algorithms. The engine model includes intake manifold air dynamics, fuel wall-wetting dynamics, and cycle delays. The modelling approach is a conventional first-principal one, for example using the ideal gas law for intake air flow. The continous equations representing the engine dynamics are discretisied using difference equations. A constant gain matrix Kalman filter (Kalman, 1960) is used to estimate the fuel puddle mass in the intake system. Integral control is used in the form of state-space bias estimation to remove steady-state air/fuel ration errors. To speed up the dynamic response, the authors use a form of adaptive feedforward control that they describe as a *learning map* to store the biases that have been previous learned by the integral controller so that under transient conditions the integrator can be reinitialised to the previous bias value for the operating point. During testing of the scheme the ignition timing was independently controlled to maintain the combustion peak-pressure-position to 15° ATDC. The air/fuel ratio shown to be controlled to within 0.5% rms of the demanded value during an arbitary throttle transients and over 10° throttle angle step changes.

A later papers by the same authors (Chang et al., 1995; Powell et al., 1998) revisit this work with more details of the experimental hardware, more consideration to the lambda UEGO sensor's characteristics (differing lean-rich and rich-lean response times), and further description of the engine model formulation and experimental results for lean air-fuel ratios which the controller was able to track to within 0.8% rms.

3.7.4 Artificial Neural Network (ANN) Augmented Controllers

Sliding mode with Gaussian RBF Transient Compensation A paper (Won et al., 1998) which builds on the results of Choi and Hendrick's use of sliding mode observers, takes the sliding mode system further by introducing a Radial Basis Function (RBF) network to provide transient fuelling compensation under fast throttle changes using

dynamic sliding mode control. A degree of robustness is given to engine ageing and the peculiarities of individual engines. This Gaussian implementation was tested alongside a production ECM on a 3.8L V6 sequential port injection engine. A 33 MHz 386 PC was used with Microsoft Quick-C compiler to run the control algorithm within a 10ms loop time. LM322 timer ICs were used to override the ECM and impose the new fuel injector duty cycle. The results appear to have improved upon the original ECM performance for transient conditions with a standard deviation of only 1.56% of the 14.64:1 air/fuel ratio set point that was used. Again this was achieved without the laborious process of engine mapping and so compares favourably to the ECM in the limited test case.

Radial Basis Function Air Mass Flow Estimator Another approach that set out to improve upon the previously referenced sliding mode scheme was outlined by Manzie et al. (2002). The controller's ability was extended by using a wideband oxygen sensor (UEGO) which has a linear response to changes in air/fuel ratio. A RBF neural network was used as an estimator of the air mass flow into the cylinder instead of the sliding mode scheme. It was initially trained using the same speed density laws that are used directly in the sliding mode scheme. This approach maintains adaptivity whilst the estimator is still based upon known calculated quantities. A first order model was used to represent the fuel pooling on the manifold walls. A model predictive control (MPC) algorithm is used to produce an optimal control set that compensates for the fuel film. MPC is believed to be the only methodology that can handle the constraints in a systematic way.

The authors tested the resulting controller by both simulation and experiment. The simulation used the Mean Value Engine Model technique, but with a modification required due to a limitation in the way the fuelling dynamics are represented (Aquino, 1981). The engine tests were performed using a MoTec controller connected to a 4-cylinder 2.4 L Mitsubishi Magna SEFI TE series engine that has no external EGR. The replacement control algorithm was implemented using LabView software running on a Pentium class PC. A National Instruments data acquisition (NIDAQ) PC-LPM16 card was used to interface the PC to the engine sensors and output the fuel injector pulse width. The original MoTec ECU was used to trigger the fuel injector driver circuitry at the correct time so the PC software did not have to calculate the engine timings, but was required to update the fuel demand within an engine cycle. A Bosch UEGO sensor was placed about 1m from the exhaust valves and a MoTec linearising kit was used to convert the output into a DC voltage in the range of 0-1.6 V.

For testing purposes the throttle was changed by 10° over a 200 ms duration. The results show that the proposed controller is able to maintain the air/fuel ratio within a 1% boundary of the desired ratio. This is better than the MoTec but it exhibits far more oscillation within the boundary than the MoTec does. It is suggested that this could be fixed by a low-pass filter on the control signal, but this was not attempted because the

bounded oscillations are not harmful to the catalytic converter and the emissions will be time averaged. As with other proposals little or no consideration was given to the processor speed, or computational effort, that would be required for a production implementation. Also no discussion was assigned to issues of cold starting when the UEGO sensor had not reached operating temperature or is broken. It is likely that a lookup system would still be needed as a fallback in these cases.

Hybrid ARX Controller A hybrid fuel injection controller has been proposed by Wendeker & Czarnigowski (2000), the conventional controller as a single-input single-output (SISO) jump-ramp controller, where the input signal is the amount of fuel and the output signal is the lambda sensor feedback. The core of the system was a mathematical model based upon a multi-cylinder 2198 cc 16-valve engine with multipoint injection. The complete description of this model is not given but an unpublished reference written in Polish is cited instead. The text suggests that a dynamic engine model may contain a submodel of the fuel film behaviour but it is not clear whether or not this is the case for the model that was used.

Adaptive identification was used in the form of a linear ARX (similar to ARMAX but the parameters are considered time-invariant hence the absence of the MA or *Moving Average* terms) model. The identification is used to estimate the parameters of a regulator which controls the value of a coefficient used by the engine model to adjust the value it has determined for fuel injector duration.

A *forgetting factor* is used with the model which must be optimised to find a compromise between the speed and the precision of the adaptation. The factor is adjusted for optimum operation by the use of a neural network which does so as a function of the engine speed, load and change in load. Very few details about the neural network are given except that it uses three groups of nodes or neurons. The input layer has three variables engine speed, intake manifold air pressure (MAP), and a transiency variable delta MAP which is the difference between the current MAP and the average over the previous seven cycles. There is a hidden layer with ten neurons and an output layer which has the forgetting factor value.

The results are presented only for a simulated engine, the data for which was collected from road tests. An example optimal forgetting factor distribution for the three variables was shown. It is not clear how the neural network was trained, either before or during the simulation. The engine model parameters are not adjusted, only the gain is, therefore the scheme does not directly accommodate changes in the plant over time which will inevitably deviate from the model, increasing the amount of correction and adaptation that is required. This approach does however keep the complexity level down which may make it suitable for use in production hardware. The complexity of the engine model

used is unknown so it is difficult to asses the computational overhead of the whole scheme.

Cerebellar Model Articulation Controller (CMAC) Majors et al. (1994) reported their first implementation of a CMAC neutral network for control of the air/fuel ratio of a research vehicle. The control objective was to maintain the air/fuel ratio with a 1% band. The CMAC controller was configured to be fast learning and adapts as the operating point changes. No information is retained about an operating point, and so optimums have to be relearned each time an operating point is revisited. The researcher used a wide-band (linear) lambda sensor and the results compared favorably with the production controller under relatively steady conditions. The limitation found was that the CMAC neural network did not work as well under transient conditions. The authors also cite the use of the more expensive and less widely used wideband sensor as being a practical limitation.

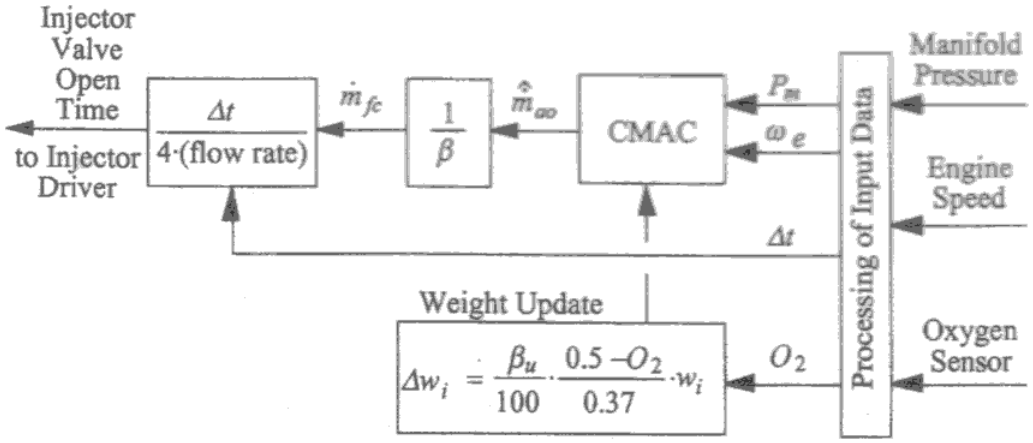

Figure 3.15: Block diagram of a the CMAC fuel injection controller (Shiraishi et al., 1995)

In a continuation of the same work, Shiraishi et al. (1995) proposed another CMAC fuel injection controller. A block diagram of a CMAC fuel injection controller and a block diagram is shown in Figure 3.15. This CMAC implementation also uses engine speed and intake manifold pressure (MAP) senors as inputs with online learning to attempt to maintain a constant air to fuel ratio of 14.64:1. There is no temperature compensation scheme, but the variation of engine temperature is slower than the reaction time of the controller so it will adjust with time without explicit temperature measurement. Past experience of temperature variation is therefore not stored for later reuse to allow faster learning. A standard narrow band (non-linear) lambda oxygen sensor (sometimes refered to as the binary type due to it's sigmodal sensitivity characteristic near to the target measured gas oxygen content) was used for the CMAC controller feedback. The lambda sensor time delay (presumably the combined exhaust gas transport delay and sensor's

own time constant) is compensated for by adjusting the previously activated weights rather than the currently active one which will affect the next output and not the current one.

The test vehicle used to test for drivability was a 1988 *Oldsmobile Cutlass Calais* with a Quad-4 DOHC 4-cylinder engine having a multi-port simultaneous fuel injection system. The CMAC controller was implemented on a PC using MS-DOS with a RTI-815 I/O board fitted. The original engine ECU was left in place and a toggle switch was used to override the ECU injector signal with that of the CMAC controller. A wideband linear oxygen sensor was installed downstream of the normal narrow band lambda sensor to provide an independent means of monitoring the controller's regulation performance.

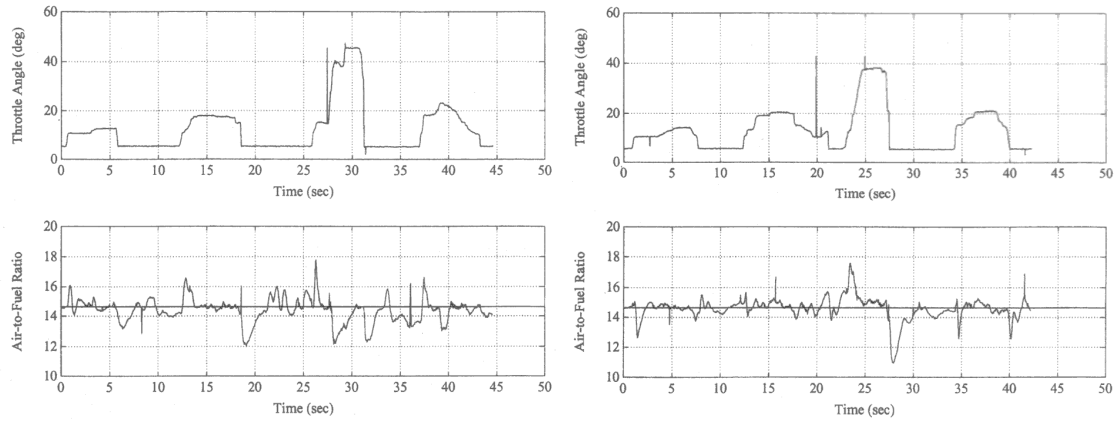

Figure 3.16: Throttle and A/F ratio control using the standard ECM (left) and using the proposed CMAC controller (right) under random driving conditions (Shiraishi et al., 1995)

The new controller performed as well as (or better than) the original ECM, as can be seen by a comparison of the A/F ratio excursions in Figure 3.16. No mapping was required, with only a period of self training. Extra weights had to be used to compensate for transient throttle conditions. It was recognised that the throttle tip-in and tip-out conditions had to be handled differently as they produce opposite mixture excursions. An identified limitation was the quantisation error for the MAP input. A pressure of 0-20 kPa was divided into 40 discrete intervals and this was thought to be insufficient to cope with transient conditions. If the number of intervals had been increased then the computational overhead would have been increased also and the training time extended.

The approximation capabilities of the CMAC network have been improved (Gan & Rosales, 2003) by replacing the constant weights with linear function weights. The improvements in smoothing are demonstrated but it is not reported whether there is a significant penalty in adaptation time. It is thought that CMAC networks are chosen for their simplicity and localisation of weights for quick online adaptation and any increase in complexity acts contrary to this.

Radial Basis Function (RBF) Controller An online trained neural network for fuel injection control that is adaptive and was verified on a real engine was outlined by Park et al. (2003). This made use of a Radial Basis Function Network (RBFN) combined with a conventional linear feedback controller. A wideband oxygen sensor was used for feedback error learning. The feedback controller was fixed and two RBFNs provide a feed-forward path. The first network (designated RBFN_S) was given inputs of engine speed and MAP, the second (RBFN_T) was given the derivative inputs of the first. RBFN_S was optimised in accordance with engine speed and MAP since the air mass flow per stroke is a non-linear function of the engine speed and MAP. The air mass flow rate is proportional to the corresponding fuel requirement. RBFN_T was used to compensate for throttle transients by adding an additional fuel demand to the existing RBNF_S and a fixed base injector duration termed T_{i0}. This demand can be negative and so it will reduce the overall demand when necessary. The RBFN_T has to be trained after RBFN_S in a way that is analogous to conventional engine controller mapping where transient compensation is added after the static maps are calibrated. A block diagram of the system is shown in Figure 3.17.

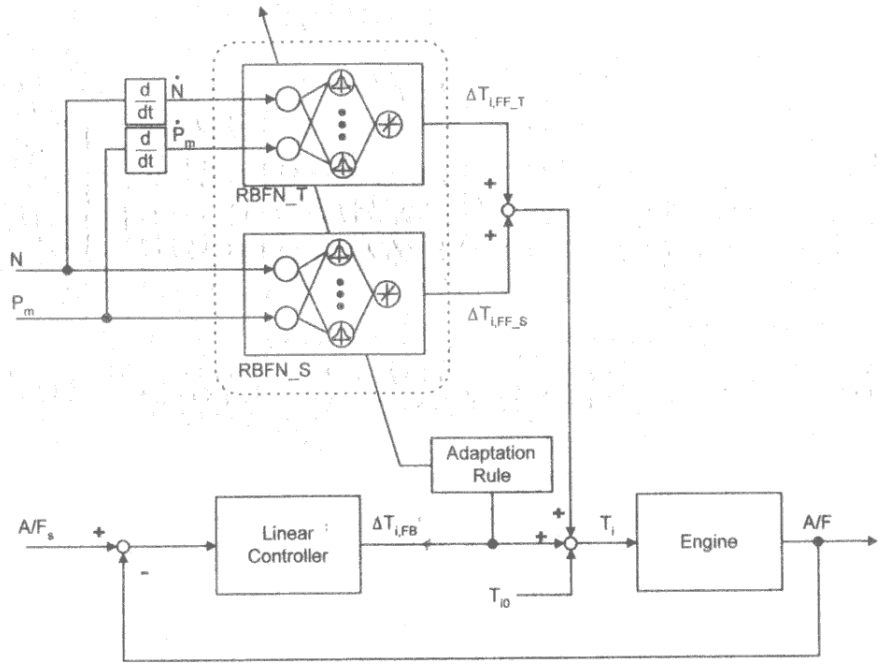

Figure 3.17: Radial Basis Function augmented controller (Park et al., 2003)

The setup was tested using a 2.0 L 4-cylinder DOHC engine connected to an eddy current dynamometer. Results are presented after a 15 minute learning cycle for only a single (but repeated amplitude throttle transition) which was a change from 5% to 10% of full throttle opening. The test time shown had a duration of 100 minutes which is too long to show the detail and the magnitude of individual air/fuel ratio excursions. The controller does not store past results for use if the sensor fails or the neural network diverges from the optimum solution, and thus needs resetting.

Recurrent Neural Network (RNN) Controller (Offline Trained) A proposal for the use of Recurrent Neural Networks (RNNs) for air/fuel ratio control has been presented by Arsie et al. (2004). The controller was implemented using direct inverse modelling and compensates for wall wetting dynamics to make estimates of air/fuel ratio during throttle transients. The controller was only tested with an engine simulator but nevertheless the results presented show that the air/fuel ratio can be bounded about a target value as shown in Figure 3.18.

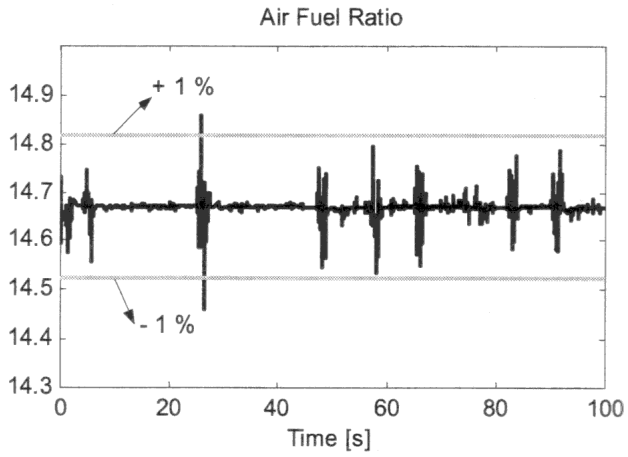

Figure 3.18: Results of direct inverse modelling approach to air/fuel ratio control (Arsie et al., 2004)

The training of the RNN was performed offline and so this implementation was not adaptive and would need extending to allow it to learn online so that it could be made to cope with any parameter change such as fowling of the fuel injectors leading to an altered spray pattern.

3.7.5 Model Predictive Control

Lennox & Montague (2001) have developed a non-linear model predictive control (NMPC) scheme around an ANN model.They explain that the difficulty with NMPC over MPC is that the solution or optimisation of the cost function is not explicitly possible and instead the cost function must be optimised using an iterative non-linear optimisation procedure such as quasi-Newton, genetic algorithms, or successive quadratic programming. These techniques can realise a solution in seconds, which makes them very effective for use in the chemical and process industries. However, they are entirely inappropriate for engine control use which might typically require a solution every 10 ms. The authors rejected the use of ARMAX and NARMAX models in favour of an ANN. They captured the engine process dynamics by incorporating low-pass filters into the neuron processing elements of the ANN. The filter constants are not known *a priori* and therefore need to be determined by the training algorithm in addition to the

network weights during training of the ANN. The authors also decided not to use the *Padé* approximation approach for handling variable delays with ANNs, which they say has been demonstrated previously. Instead they opted to use an external time delay model to time shift the data presented to the ANN due to the large magnitude of delays which can be associated with engine dynamics. They rely on the ANN to accommodate for any small errors in the delay model. For training a Levenberg-Marquardt search direction method was used. The control scheme was tested only against an engine model. The model used was produced by Ricardo and is described in Beaumont et al. (1993), however the reference gives only an overview with very few specifics and presents very limited results. The model was used to generate simulation results which were in turn used to train the ANN. Also a finite impulse response (FIR) model was developed and results shown that compare the fit to the simulation data of the trained ANN model to the FIR model. The ANN output follows the simulation output almost indistinguishably over the range and at the scale presented, but the FIR model diverges substantially during transients. The FIR model was stated to be more accurate than both ARX and ARMAX models which were also attempted, but for which no results are shown. In order to solve the non-linear cost function quickly enough for engine control use, a linear approximation to the non-linear step response, evaluated at each time step was performed and tested under simulation alongside a much more computationally intensive iterative solver for comparison and the two approaches were found to give very similar results which validates the use of the linear approach. As a final step, the controller was tested using a linear approximation to the ANN which is shown to perform poorly against the ANN approach, confirming the requirement for predicting the non-linear process dynamics.

Saerens et al. (2008) present the use of what is described as model predictive control to minimise fuel consumption of an SI engine through dynamic optimisation, using a mean value model of the powertrain and vehicle. The model has two state variables: the pressure in the engine manifold and the engine speed. They use throttle valve angle as a control input to the model which is identified on a dynamometer. Optimum (for minimising fuel consumption) engine speed trajectories are calculated offline, then a trial-and-error tuned PI controller is used to track the static trajectories on-line. Since this approach does use a model for prediction which is ultimately used for control, but does not conform to the conventional understanding of the term model predictive control which implies a controller which uses an on-line optimisation of a receding horizon. The results presented claim that an optimal engine speed trajectory yields a reduction of the fuel consumption of 12% when compared to a linear trajectory.

3.7.6 Direct Inverse Model

Gerasimov et al. (2011) have presented an approach to torque tracking and air/fuel ratio

control using a tandem inverse model and direct model approach. They describe the models as being *grey box* with a fixed structure, the parameters of which are estimated off-line using vehicle data and linear regression. The controller attempts to maintain the gain between the direct and inverse models at unity. A PID controller is used in combination with the feed-forward controller to improve performance and reject external disturbances. Limited results are presented for a V8 engine over speed transients spanning less than 800 rpm.

3.8 Spark Ignition Timing Control

3.8.1 Self-Tuning Optimisation

Scotson & Wellstead (1990) have presented a technique to perform on-line optimisation of the ignition angle map used by engine controllers. Earlier work from Wellstead & Zanker (1981) used similar self-tuning control (or extremum control) techniques for engine speed regulation. For optimisation to take place a feedback signal is required which for ignition angle variation, is usually engine torque, either indicated or measured at the output. For this work the authors used measured variations in flywheel speed to determine changes in torque output. They concede that this produces a torque signal with a poor signal-to-noise ration, but present techniques to cope with noise. The benefit of this approach is that it does not require the addition of a torque sensor, so can be applied to existing engines and the existing ignition map is retained so that the adaptation process can be turned off at any time. A variable amplitude perturbation signal was applied to the normal ignition angle to allow the optimiser to search for new optimums. The authors suggest that the technique could be extended to work on an individual cylinder basis and could also be used for condition monitoring indicated by larger than expected drifts in parameter optimums.

3.8.2 Peak Pressure Position Control for Maximum Brake Torque

Early work by David Powell of Stanford University was presented in Hubbard et al. (1976) for an optimum peak pressure position (PPP) ignition angle controller was developed for a test engine. The experimental results showed a good performance increase over the conventional mechanical distributor advance mechanism with the air/fuel ratio held constant. Powell goes on to show how the humidity of intake air acts as a significant disturbance which is automatically rejected by the PPP controller. The work is continued in Hosey & Powell (1979) where it is shown that the (PPP) controller works in spite of large variations in air/fuel ratio, an the use of the pressure sensor as a knock controller is also introduced. The paper gives further consideration to the various

types of pressure sensor which might be suitable and advocates the adoption of cylinder pressure transducer based PPP control for production engines based on the fact that volume production levels would substantially reduce the per-unit cost of the piezo pressure sensors. Almost 20 years after the initial investigation work began, with cylinder pressure transducer still not fitted to production engines, Powell (1993) revisits the use of cylinder pressure and advocates it's many uses including air/fuel ratio (as detailed in Gilkey & Powell (1985); Gassenfeit & Powell (1989)) and intake charge temperature estimation.

The widespread adoption of lambda sensors combined with piezo knock sensors on production vehicles with SI engines, does now make pressure sensing seem less worthwhile, however controlling an engine against it's knock margin does not necessarily correspond to the maximum brake torque (and hence efficiency) PPP being achieved, particularly in the present of varying humidity and other disturbances.

3.9 Instrumentation for Torque Determination

Torque is a desirable feedback variable for engine control and condition monitoring. It can be used both to maximise power output and to determine/optimise brake specific fuel consumption (BSFC) to a minimum and hence ensure best efficiency is being maintained. As torque is a consequence of cylinder pressure, it can be inferred from pressure signals as well as from direct measurement at the output from the crankshaft. This section looks at some of the various options for measuring engine torque for use as a control feedback variable.

3.9.1 In-cylinder Pressure Transducers

In cylinder pressure measurement is desirable from a control and condition monitoring perspective but historically it has been plagued by problems as transducers that are exposed to the combustion process are subjected to large thermal shock and typically require water cooling. They are also expensive and fragile making them unsuitable for use outside of a laboratory environment.

The lack of a reliable and cost effective cylinder pressure sensor is why they have not having been widely adopted for production engine use. Electronic pressure sensors that are be used for other applications cannot meet the specifications required to be reliable for the $> 10^9$ combustion cycles (approximately 11,000 hours at 3000 rpm) that might be typically required. Piezoelectric sensors have been used for high temperature in-cylinder use in the past. The natural quartz devices can operate up to 350°C without water

cooling but they are prohibitively expensive and have limited durability making them unsuitable for production use. Piezo-ceramics can be cost effective but they are usually used for indirect pressure measurement (e.g. as a spark plug washer) which results in reduced accuracy and large vibration and inertial acceleration errors. Silicon micro-machined sensors made using techniques borrowed from VSLI process technology have been developed but due to the nature of the silicon suffers a temperature limitation of around 150°C and are susceptible to EMI/RFI interference. Higher temperature devices have been developed using silicon on insulator (SOI) and silicon carbide (SiC) designs but the long term reliability of these has yet to be demonstrated, not withstanding the cost and complexity issues that remain.

An optical transducer has been proposed by Ulrich et al. (2001), shown schematically in Figure 3.19 and the two spark plug embedded variants of the sensor are pictured in Figure 3.20. The PSIplug variant of this sensor was purchased for use in this project and is the variant in which the sensor is screwed externally into a port braised onto the side of a modified spark plug. This allows the sensor to be unscrew and used with different spark plugs, but does mean there is a slight pressure propagation delay as the pressure front has migrate up through the hole drilled in the wall of the spark plug before it is sensed. The other variant has the sensor element embedded directly into the wall of the spark plug so that its face is exposed directly to the combustion flame-front. The sensor is novel because it allows the instrumentation to be located some distance away from the sensing diaphragm via a optical fibre so that issue of electrical interference as removed as well as not needing to supply the sensing element with electrical power. It's small size means it can be fitted into a modified spark plug (or diesel glow plug) so that no engine modifications are needed. The sensor does not suffer from the sever thermal shock that similar piezo sensor do and does not need sensitive charge amplifier electronics which can be problematic in the vicinity of high voltage ignition circuitry.

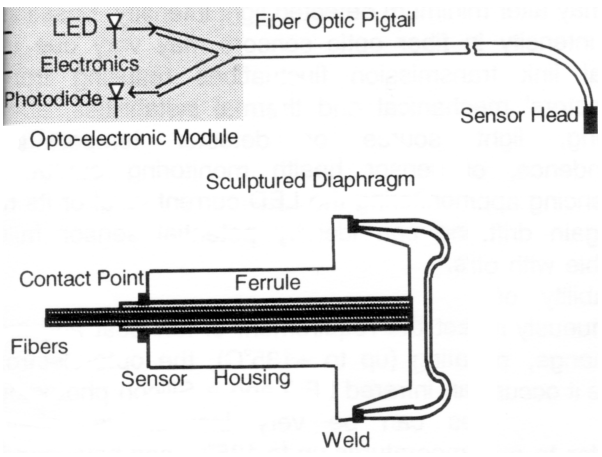

Figure 3.19: Schematic of optical pressure sensor construction (Ulrich et al., 2001)

Figure 3.20: CALplug (left) and PSIplug (right) modified spark plugs using an optical pressure sensor

3.9.2 Spark Plug Ion Sensing

Mecel AB of Sweden produced an engine ECU fitted to some Saab models which made use of spark plug ion current sensing for misfire detection. This lead to a PhD investigation (Eriksson, 1999) into the use of the ion signal from the Mecel ECU to derive a cylinder pressure signal. The ion current does give a good pressure indication, but the signal is very stochastic in nature with a poor signal to noise variation. Eriksson makes some progress with processing of the signal and revisits the peak pressure position control work of David Powell and demonstrates the benefit of the controller for use with water injection. Much interest followed in the literature from this work including many more paper such as Andersson & Eriksson (2001), with studies on the influence of air/fuel ratio and permutations of other variables.

3.9.3 Torque determination via Piezoelectric Spark Plug Load Washer or Engine Mount Strain Measurement

Fleming (1982) describes several techniques for direct measurement of engine torque using strain effects caused directly by torque. The use of a load washer under the spark plug seat is one method considered. This technique is thought to be problematic due to the calibration effort and temperature effects. Also to determine cycle-by-cycle torque requires the evaluation of a pressure integral that requires synchronised volume and pressure signals. This however, might be more easily achieved using a modern engine management system than when the paper was published.

Fleming also considers the use of load-cell force sensors installed in engine and transmission mounts, so that the torque reaction forces of the engine/transmission assembly can measured. The reaction forces are proportional to driveline torque, but to isolate interfering shear forces from the reaction forces, special engine mounts need to be used. The limitations beyond the general use of strain gauges an the associated electrical noise and thermal drift issues, are that inertial brake torque is not measured since the combined driveline torque is being measured. Also anything offering a reaction torque to the engine such as pressurised coolant hoses and electrical cabling will contribute an error to the measurement.

In the text of a US patent, Willner (2006) describes low cost piezo-ceramic devices, such as spark plug washers, as unable to offer high accuracy under all engine operating conditions, and cites that they are also subject to electromagnetic interference and tend to have durability problems related to alloy separation, selective oxidation and diffusion when used in production engines.

3.9.4 Strain Gauge Fitment to a Drive Shaft

A model was developed to help predict the fuel consumption of heavy goods vehicles (Sandberg, 2001), and in order to validate it a series of experiments was performed by instrumenting the propeller shaft on an actual truck as shown in Figure 3.21.

Figure 3.21: Propeller shaft fitted with strain gauges (Sandberg, 2001)

The vehicle was driven at as near as possible constant speed and torque data was recorded for a variety of speeds so that the external forces such as air resistance and rolling resistance could be determined and compared to those predicted by the model.

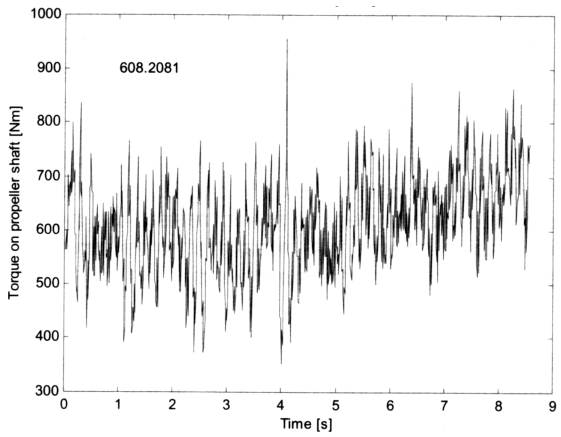

Figure 3.22: Propeller shaft torque (vehicle travelling at 81 km/h sampled at 100 Hz) (Sandberg, 2001)

A sample of the recorded data is shown in Figure 3.22 and the fluctuations caused by the ignitions of the six-cylinder engine can be seen.

3.9.5 Magnetostrictive and Magnetoelastic Torque Measurement

Fleming (1982) presents and introduction to the use of magnetostrictive torque sensing. This technique using the change in magnetic properties in the surface of a material due to strain as mechanically deforms under torsional stress. It is expected that this could be a method to directly measure torque on an engine's crankshaft and benefits from being non-contacting (no wear or electrical contacting issues) and has a fast response time. The limitations are that a clear shaft length of ~22 mm that is required may necessitate engine redesign. Also a sleeve needed to achieve consistent results between sensor installations. The size of the air gap (needed due to the oil-filled/fluidic crankshaft bearings) causes signal-to-noise ratio problems which imposes a frequency response limitation. The sensor is strongly temperature sensitive and if installed on a crankshaft, would be in a location that is exposed to large temperature changes. Fleming (1989) addresses some of the sensor characteristic far more comprehensively with a non-linear model to compensate for real-world magnetic effects such as saturation.

Sobel et al. (1996) describe a similar sensor to Fleming. The instantaneous torque of an internal combustion engine was measured adjacent to the flywheel, using a inductive magnetoelastic torque sensor. They measured torque in a four-cylinder engine and compared data with pressure signals during bench tests. They report excellent correlation with values of the mean effective pressure for each cylinder. Road tests were performed in a car with a five-cylinder engine and automatic transmission. Results are presented with respect to engine control and misfire detection, which show only minor disturbances caused by driving on rough roads.

Uras (2001) describes a permanent-magnet implementation of a magnetostrictive sensor which is constant flux eliminates exhibit temperature an drift problems and the need for a power source. The sensor is only capable of measuring dynamic effects so it's use is limited to misfire detection and stall/crash detection.

Larsson & Schagerberg (2004) describe the use of a commercial implementation of a magnetostrictive/magnetoelastic called *Torductor* produced by ABB, shown in Figure 3.23. It was fitted to the crankshaft of an SI engine and used for research into estimating and controlling the peak-pressure-position (PPP) of combustion in individual cylinders. The sensor requires that copper strips (rather than a sleeve) are attached to the shaft from which torque is to be measured.

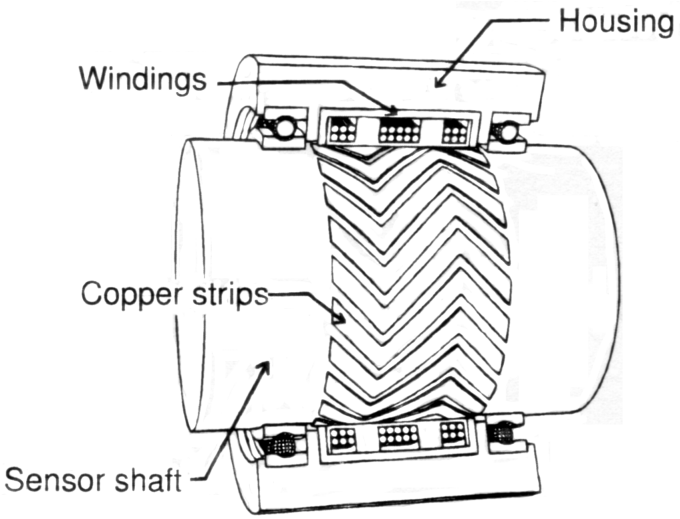

Figure 3.23: ABB's magnetoelastic torque sensor called a *Torductor*

3.9.6 Surface Acoustic Wave Measurement

Surface acoustic wave (SAW) devices are used as frequency dependent strain gauges to measure the change in resonant frequency which is proportional to the strain experienced in the shaft. In SAW devices surface waves are produced by passing an alternating voltage across the terminals of two interleaved comb-shaped arrays which are laid onto one end of a piezoelectric substrate. At the other end of the transducer there is a receiving array to convert the wave signal into an electrical signal.

A company called *Sensor Technology* (Palmer, 2004) has developed a compact commercial unit (Figure 3.24), which contains two *Cygnal* processors. The first processor is used with the SAW sensors and the other is used for outside communications so that a high bandwidth of 5-10 KHz can be achieved. Torque sensitivity ranges from 100 mNm up to around 10,000 Nm are achieved from three case sizes and seven shafts sizes. Various connectivity options are available including RS-232/USB and Bluetooth for wireless interrogation.

Figure 3.24: Commercial non-contact TorqSense RWT310/320 unit

The RWT device uses a non-contact RF (radio frequency) couple for power transmission and signal communication as opposed to the more conventional slip ring approach. The company also supply a NI LabView software component which they have named *TorqView* for use with its sensors.

3.10 Software for Engine Control

3.10.1 Ford's High Level Pascal-F Engine Control Software

Unlike many other automotive manufacturers, Ford has historically, developed all of its EEC range of engine controllers in-house albeit using the services of third party contractors, in all likelihood. Mills (1985) describes how a high-level engine control strategy was written using a specially written real-time version of Pascal (the development of which is detailed in references therein) called *Pascal-F*.

Ford's EEC-IV was introduced from 1982, the code which was originally written in assembly language for the Intel 8061. The 8061 was a custom VLSI version of the popular 8051 8-bit microprocessor, which was developed jointly between Intel and Ford for use in its engine controller units. The assembly language coded program for the EEC-IV was divided into only two blocks of code which were termed foreground and background. The foreground was responsible for performing all of the outputs including spark firing, fuel injector pulsing, and EGR (exhaust gas recirculation) valve control. It also comprised routines which service three interrupts, keeping track of the time between successive cylinder firings, and the time between task-controller executions. The

background code was a piece of round-robin, sequentially and continuously executed code used to perform calculations such as spark angle, fuel injector pulse width and frequency, as well as the amount of EGR to permit. These background tasks are engine state dependent and the code path length will vary. Variable code path lengths can often cause development issues in real-time systems as intermittent and infrequent malfunctions, or *bugs*, can occur which by their nature are difficult to diagnose or debug.

As a direct result of the growing use of engine controllers within Ford, there was a group of engineers who all needed to develop and modify engine control programs. The debugging and modification of the code became a major task for these people. Assembly language programmers with considerable knowledge of processor architecture and low level hardware interfaces were needed. This dependency probably concerned Ford's management enough to commission the team of one assembly language, and two Pascal-F programmers, to develop a robust high-level implementation of the EEC-IV code over a period of six months. A further eight months were used for debugging and testing of the resulting code.

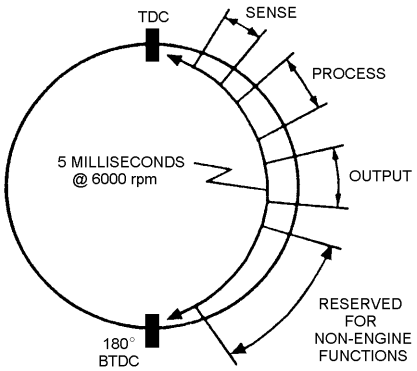

Figure 3.25: Crank angle representation of computer time usage (Mills, 1985)

Various issues had to be tackled to achieve a successful high level implementation. Engine control requires stringent execution time and memory restrictions. Figure 3.25 shows the computational requirements with reference to crank rotation and an example of this is an 8-cylinder engine turning at 5000 rpm has only 3 ms between cylinder firings and in that time the code must handle several interrupts and perform various lookup calculations in order to be ready for the next firing event. The engine controller needs to work with sampled real numbers rather than the integer values that are native to the 8051 architecture and its derivatives. There is no floating point hardware support in the 8061 and software emulation was deemed to be far too slow. Fixed point arithmetic was made part of the Pascal-F language as a compromise to allow real numbers to be used. The original assembly language production version had two code blocks loosely described as one being for real-time tasks, and the other for lower priority tasks. This method was replaced by a small 200 byte kernel, also written in assembly language, which the

complier implicitly makes part of every Pascal-F program. The kernel can support the scheduling of processes and inter-process synchronisation is achieved through hardware interrupts and a software SEND construct. The kernel is able to respond to an interrupt and switch processes within 100 μs. A monitor process allows access to its data from other processes using exported procedures and a signalling mechanism. Each process is assigned a fixed priority by the programmer which can be in the range 0-15 where 0 is the lowest. This priority is used if there is a contention for the processor and it will determine which monitor process is chosen to run.

The replacement EEC-IV program had eight processes in total that are shown in Table3.1.

Process Name	Acronym	Priority (0 = low)	Description
High Speed Output (high time resolution)	HSO	15	2.5μs max output for high accuracy of spark timing and injector pulse width.
Fuel Injector Control	FIC	14	Handles conversion of pulse width into timed on/off signals.
High Speed Input (high time resolution)	HIS	13	Digital signals which need the time of occurrence recording. Also initiates the spark and fuel output events.
Time keeper & low resolution output routines	TMK	12	Keeps track of timers to 5ms resolution. Low resolution outputs (e.g. EGR) also.
Analogue Input Utility (A/D) routines	ADU	10	Control logic for EGR and throttle kicker. Executes synchronously every 10ms from time keeper process.
Engine Synchronous Machine	ESM	7	Controls the conversion of analogue inputs through the A/D converter.
Time Based Machine	TBM	6	Contains code for initialisation and safety fuel pump shutoff. Executes only when no other processes are ready or able to.
Main Program	MAIN	0	Performs engine state determination and calculates the desired spark timing and fuel injector pulse width. It is structured to execute between cylinder firings.

Table 3.1: Engine control program processes (Mills, 1985)

The resulting code was found to be as conformant and reliable as the assembler code which it replaced, but it came at a cost of requiring twice as much memory and 60% longer execution time. For these reasons it was probably never adopted for production release and only used internally to assist with development.

3.10.2 Ford's Automatic Code Generator

A separate attempt was made by another team working within Ford, around the same time as the Pascal-F development, to research an alternative to writing their engine controller code in raw assembly language (Srodawa et al., 1985). Ford Automotive Control Terminology, or FACT, was developed as an abstract high level specification language to meet its own needs for (but not limited to) engine controller programming. It

was based upon FLECS which was a pre-ANSI standard form of what later became FORTRAN 77.

FACT is essentially an automatic code generator and not a complier. It produces assembler code from the abstract control code which is linked with hand written assembler code before compilation. It has data types and attributes that are needed for writing control algorithms. There are scalar, function, table, timer, and I/O channel data types to which several attributes can be assigned to specify items like allocation, size, and engineering units. Unlike the Pascal-F implementation, it has full floating point support which is scaleable up to 32-bit for arithmetic which can preserve significance. By comparison the equivalent production system, to which it was being compared to, could only perform 16-bit arithmetic. FACT uses soft I/O channels to which requests are posted that are later serviced in a periodic manner. FACT also has built-in functions that provide table lookup and interpolation routines.

The hand coded assembler that works with FACT consists of two main parts. The first is an executive operating system (EOS) and the second is the real time interface (RTI). The EOS implements a preemptive multitasking system which has four priority levels. It can support sixteen tasks at each priority level, but the number of tasks and their priority are fixed at compile time. The RTI is the interface between the high level FACT strategy and the low level functions of the EOS. It maps the input channels to FACT input variables, converting any raw values to engineering values, if required. It also maps output variables to calls upon the EOS output primitive and contains interfaces to the function and table interpolation routines. It is itself organised as a set of intermediate-priority tasks.

The resulting engine controller code that was produced was conformance and performance tested against the equivalent production code that was written entirely in assembler. It was found to perform as well, and in some cases out-perform the native code owing to the extra floating point precision that it is able to use which allowed it to operate with greater accuracy. The FACT system used 12.3 kbytes of ROM whereas the production code used only 6.8 kbytes which represents an increase of 81% in this case. This is partially due the overhead of having a generic real time executive scheduler (the EOS and RTI) in addition to the algorithm code. When only the FACT generated code is compared to its production assembler equivalent, only a 47% increase in code size is seen. It would seem that there is an economy of scale advocated by the researchers that; if the FACT code length and complexity was to be increased for another application, then the executive part of the code would remain the same and the resulting binary size would compare more favourably, or even better, than the equivalent assembler only code. With regard to execution speed, the production code background loop time was measured at 10 ms, whereas the FACT produced code took 20-30 ms (engine speed dependent) to do the same. It is speculated by the paper's author that the increased unit

cost created by the extra ROM size requirement will mean that the FACT system would not be used for production use until either the required algorithm complexity increases or the benefits of being able to execute strategies concurrently (such as transmission control) from the same control unit are recognised.

3.10.3 BASEMENT

BASEMENT is a distributed real-time architecture presented by Hansson et al. (1995) and was developed for vehicle internal use in the automotive industry. BASEMENT was inspired by from examination of the MARS (Maintainable Real-Time System) operating system developed at the University of Vienna (Kopetz et al., 1989) during a time of emerging concerns over the performance of safety critical real-time systems (particularly for aerospace applications). A key feature of MARS (and so presumably of BASEMENT) is that all task scheduling is performed offline assuming worst case latencies, and only a single interrupt source used (or allowed) which is timebase for the scheduler. All other hardware events are handled by polling to remove the uncertainty and asynchronous behaviour of hardware interrputs and software latency in processing them. This approach has come to be known as time-triggered. With BASEMENT, they have attemped to take what they describe as a holistic approach to application development across software and hardware boundaries. This is an appropriate approach for modern vehicles as an application often consists of resources spread across different pieces of hardware which are network linked using a combination of LIN-Bus (Local Interconnect Network, UART serial based) and CAN. They key features of BASEMENT are as follows:

- Resource sharing (multiplexing) of processing and communication resources
- A guaranteed real-time service for safety critical applications a best-effort service for non-safety critical applications
- A communication infrastructure providing efficient communication between distributed devices
- A program development methodology allowing resource independent and application oriented development of application software

A later paper (Hansson et al., 1997) considers BASEMENT again with more implementation details and an example application (having implications for safety and reliability) of an intelligent cruise control system which makes use of distributed resources and the syncronisation mechanisms within BASEMENT.

3.10.4 OSEK/VDK

OSEK is a specification (covered in part by ISO 17356) for an automotive embedded operating system, a communications stack, and a network management protocol for vehicular applications. OSEK is a German acronym for *Offene Systeme und deren Schnittstellen fur die Elektronik im Kraftfahzeug* or in English *Open Systems and their Interfaces for the Electronics in Motor Vehicles* and is a standards body founded in 1993 by a consortium of German vehicle manufacturers, suppliers, and the University of Karlsruhe. At around the same time a group of French cars manufacturers, Renault and PSA Peugeot Citroën, had a similar project called VDX (Vehicle Distributed eXecutive) which resulted in them joining the OSEK consortium and the official name became OSEK/VDX.

Some further details of OSEK/VDK are described in a paper by produced by the compiler vendor Wind River Systems and automotive semiconductor manufacture Infineon (Foster & Schwab, 2000). The article showcases their products in the context of OSEK/VDK, but also gives some implementation details. Another overview artical from Motorola (Bannatyne, 2002) gives a concise introduction in the context of microcontroller use in the automotive industry, but lacks any specific details.

An open source implentation of OSEK/VDK has emerged called Trampoline and is described in a paper by Béchennec et al. (2006) which gives a brief survey of OSEK/VDX before describing the details of their Trampoline implementation. Trampoline an acedemic intiative which is not intended to compete with commercial implementations and useful since it supports different architectures (including PIC 18 series, Infineon C167, MPC565 and Linux/x86) to varying levels and allows investigation of a particular application by compiling it as a normal UNIX executable aplication for testing on a PC, obviously forgoing its real-time capabilites.

Chapter 4. Development of a Dynamometer Controller and Automated Engine Test System

This chapter describes the work which has been undertaken to develop an automated engine test facility around an existing water-cooled eddy current dynamometer. The dynamometer was originally purchased in 1956 from the merged Heenan-Froude company which was at the time producing dynamometers under licence from the Dynamatic Company. It is not known how the dynamometer was originally controlled, but it can be reasonably assumed that it incorporated a floor-standing valve-amplifier console unit that was supplied with dynamometers of this type. The dynamometer was subsequently retro-fitted with a newer Froude controller (circa 1980) of an analogue electronic design. At the time of starting the PhD it was assumed that the engine/dynamometer were both fully functional. In the event it quickly became apparent that this was not the case, and the dynamometer controller was unserviceable. After some investigation it was reluctantly concluded that a digital software controllable replacement would have to be designed and implemented, as the project was not fully resourced and funding for a commercial unit was not forthcoming.

The first section of this chapter gives an overview of the hardware developed and installed so that the chronological sequence can be understood, and subsequent sections describe the details of and the challenges met by individual parts of the system. During the early stages of it's development the dynamometer was found to be difficult to control due to its slow overall current response combined with a very rapid initial response to any change in applied voltage. Magnetic analysis of DC machines seems to be currently far less prevalent possibly with the rise in popularity of AC inverter drives. Much of magnetisation theory which remains readily accessible pertains to the AC operation of transformers with some treatment of the effects of DC bias on them. Machines with large solid and unlaminated cores are no longer common place and the explanation of their magnetic properties is no longer readily available. The in-rush effect was modelled as a parallel resitance as this represents the proportionality to applied voltage which it exhibits. However this approach failed to show correspondence with the trainsient nature of the in-rush step and how its effects seemed to be both rate dependent and demininshing with time. After speculating various causes for the magnetisation phenomenon which appear to defy the plausible $\frac{L}{R}$ ratio for the machine, in the interim period, progressively older archives of literature have been catalogued and made searchable elctronically, the cause of both the initial in-rush of current followed by an unexplainably slow response to steady-state was eventually found to be attributed to eddy-current damping. It appears that this phenomenon was a well known and problematic effect at a time when large DC machines where in more widespread use due

to the ability to vary their speed more easily than their AC counterparts. Laminated cores reduce the eddy-current paths so that their effects become far less pronounced. AC operation tends to remain further down the magnetisation curve away from saturation as the alternating field allows less time for saturation to take place and therefore hysterisis and the associated losses is normally of more interest when considering magnetisation effects.

4.1 Overview of Test Bed Work

The development work was undertaken in two phases. In the first phase it was sought to directly upgrade the existing facility which had both the dynamometer controller (Figure 4.1) and operator control panel box (Figure 4.2) hinge mounted together on the outside of the engine test cell enclosure. This existing setup used long cable runs to carry both power and signals to and from the dynamometer.

Figure 4.1: Froude dynamometer controller

Figure 4.2: Froude operator's control panel for the dynamometer

It was decided, due to space restrictions within the cell, to retain some of the control hardware outside the cell. This meant the embedded system for torque and speed control, monitoring and automation, would be located outside the cell in an operator's box (Figure 4.3). A power electronics and signal conditioning enclosure box (Figure 4.4) was mounted inside the cell adjacent to the dynamometer to reduce the cable length. The intention was that this enclosure would contain the solid-state electronics and current controller that would be unlikely to change or require frequent access once configured. Buffered analogue signals (such as torque, speed, and field current set-point) were run to the outside of the cell using individually screen cables to the operator box outside the cell. The existing ad-hoc test bed engine power supply wiring was replaced with suitably rated control cable and routed into conduits leading to an automation plate (Figure 4.5) mounted above the engine which provides fused sold-state relay (SSR) control over the engine and a place to mount a battery isolator, current-trip, the engine ECU, ignition drivers and fuse box. Low current control lines were run from the SSRs to the outside operator's panel so the the engine could be started from outside the cell. Opto-isolated overrides were used to allow software automation of engine starting and testing. Although the intention was to use automated software testing, manual overrides were put in place on the operator box to allow manual control of the dynamometer duty or set-points and to provide a familiar *knob and dial* user interface in case that was preferred by a user. Control and signal wiring was fed through a rectangular aperture made in the cell wall through to the rear of the operator box to allow easy access and modification of cabling. A small enclosure was mounted on the inside of the cell to

cover the aperture and provide a location for connectors to be mounted, allowing cables to be disconnected from the inside of the cell. In the first development phase, the operator control box components and features include the following:

- Mains powered AT PC PSU
- ARM based networked processing unit running embedded Linux
- Bespoke colour LCD touchscreen interface
- Connector panel for serial, CAN, USB and Ethernet network connections
- Operator switches and overrides using bespoke opto-isolated interface
- Engine instrumentation (including battery voltage an oil pressure)
- Bespoke multiplexed thermocouple amplifier
- Bespoke incremental encoder interface

The power electronics enclosure components and features include the following:

- Linear mains powered PSU
- Semikron phase angle controller
- Thyristor bridge module
- Bespoke analogue current controller
- Bespoke frequency-to-voltage tachometer
- Bespoke load-cell strain gauge amplifier
- SSR control of cooling water valve with valve open and water pressure switch indicators

Figure 4.3: Operator control box mounted on the outside of the engine cell

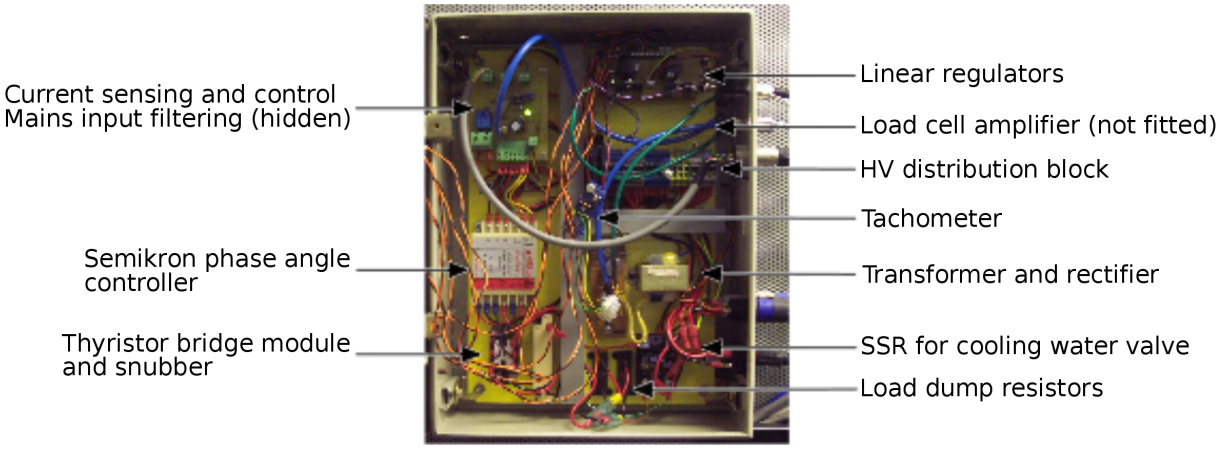

Figure 4.4: Dynamometer power electronics and signal conditioning box

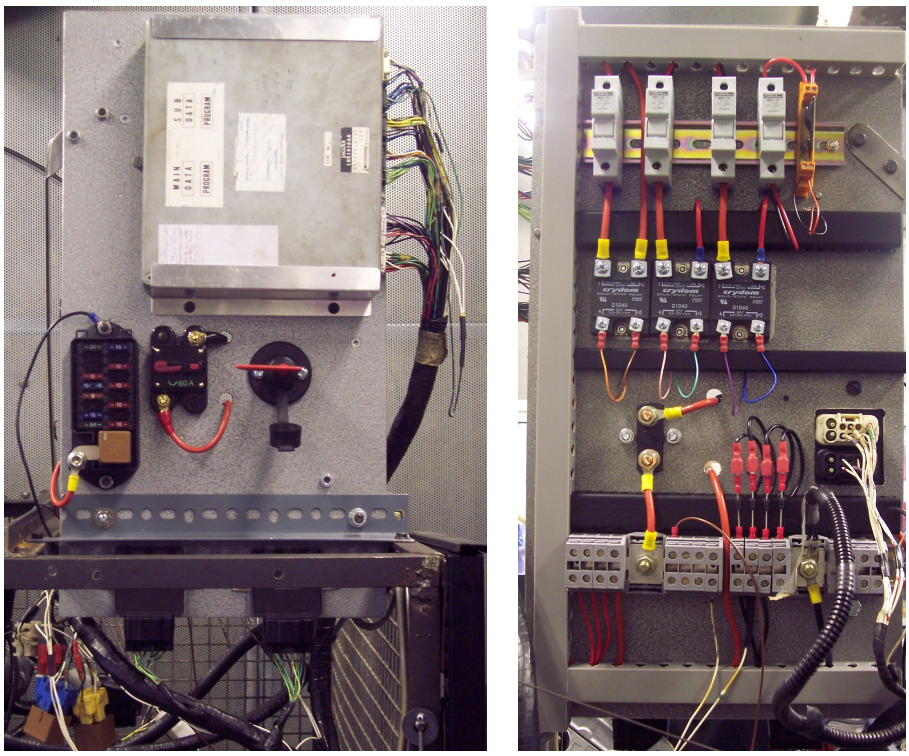

Figure 4.5: ECU and power distribution plate front (left) and rear (right)

During the course of the work it became apparent that the current controller approach adopted was too problematic, and the instrumentation was suffering from electrical noise coming from the mains supply through the earth rail to which the all steel-construction test cell room was connected. Despite precautions been taken to use shielded cable for signals, limit ground currents, and isolate instrumentation using common-mode chokes, the noise spikes were never completely eliminated or their source determined beyond the mains supply being *dirty*. Tests were undertaken to attempt close loop speed control of the dynamometer-engine system. The speed control was performed by varying the dynamometer applied load torque on the engine. This was done as the other control mode, using engine throttle to control speed, was not possible at that stage as a throttle controller had not been developed until a later stage in the project. It was during these tests that it became clear that the current response was far too slow for stable operation and that the system was open-loop unstable for at least some of the operating range. The provision for engine cooling (using a cell-roof mounted radiator and fan) was also found to be inadequate and limited the run-time to the order of minutes under load before overheating occurred resulting pressurisation and loss of coolant. This impacted upon the amount of continuous testing that was possible, but still provided enough data for an initial assessment of the dynamometer's performance.

The second phase of work came about when the existing engine cell was decommissioned and replaced with a much larger facility[1]. The move into the new

[1]Access was prohibited for approximetely 12 months during the rebuilding and recommissioning work.

facility presented an opportunity to improve upon the short comings of the first phase. The first phase of work had revealed problems with long cable runs, signal noise, inadequate processor performance for a software GUI operator display, and that an analogue current controller could not be tuned to be both stable and fast enough in response to allow useful torque and speed regulation. The system layout issue arising from space constraints in the cell and the need for a operator interface to be located remotely from the engine had been removed. The new cell is equipped with a dedicated multi-engine cooling heat exchanger system so that the engine can be run for as long as needed. The cell has an operator's control room, acoustically isolated from the engine enclosure, but with no provision for routing of cabling between the two rooms. There was also no prospect of a cable routing provision being added as it would compromise the air-seal between the two rooms required as part of the fire protection system. This made mounting the operator box in front of the engine the only suitable location. The box was reconfigured as a pendant box (Figure 4.8) so that it had a range of movement to allow access and visibility of the controls from more than one direction. More fundamentally it was decided that it no longer served any purpose to split the dynamometer control between two locations. The ARM embedded system and the power electronics were consolidated into a larger box to contain everything directly relating to the control of the dynamometer. The LCD touchscreen was also moved with the ARM board so that a console monitoring interface could be provided for the dynamometer which is independent of the test bed application software and to assist with the controller development (Figure 4.6). Having everything in one box now meant that much of the external cabling was not needed, and any of the remaining cables leading to sensors were only short in length with more direct routing. The chassis plate mounted components of the previous smaller power electronics box were transferred on the chassis plate to save effort with remounting (Figure 4.7). The analogue current controller was removed and the embedded system was used to control the field current directly under software control. To allow the current to be sampled for control, a current transducer board was developed in place of the analogue current controller. Also a software switchable load-dumping circuit was developed to allow faster removal of torque load under software control. The low-current mains linear supply was replaced with a custom made higher current fan-cool linear supply to meet the increased current demand created by adding additional electronics into the box. A linear supply was constructed to avoid the signal noise problems associated with proximity to switched mode supplies.

An x86 based embedded system with an integrated graphics controller was acquired and used with an off-the-shelf VGA LCD touchscreen (Figure 4.8) to provide a graphical user interface. The operator box has retained the engine related switches and instrumentation and uses a network link to the dynamometer controller to monitor, set and display the operating points of the dynamometer using purpose written application software. The PC processor board was used to acquire engine cylinder pressure data

using the previously developed digital encoder interface. The mains powered PC switched mode PSU replaced by a vehicle switched mode DC-DC 5 V power supply which runs off the 12 V engine battery. 12 V nominal power is supplied directly from the battery where is is required, unregulated, but with some transient protection at the box end. The aperture cut into the box for the AT PC PSU was reused to provide a connector panel for thermocouples and control cables that were previously routed through the back of the box. An industrial cable chain has been used to flexibly transfer the wiring across to the engine test bed.

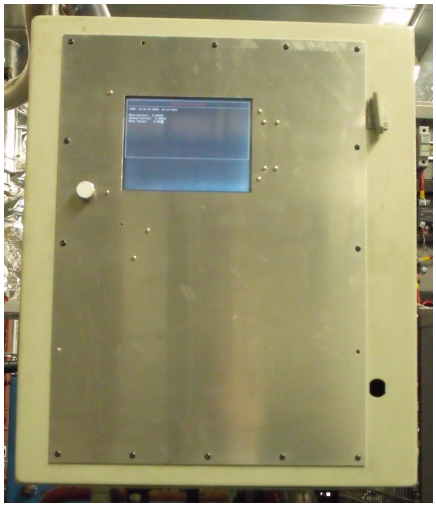

Figure 4.6: Dynamometer power electronics and control box in new engine cell

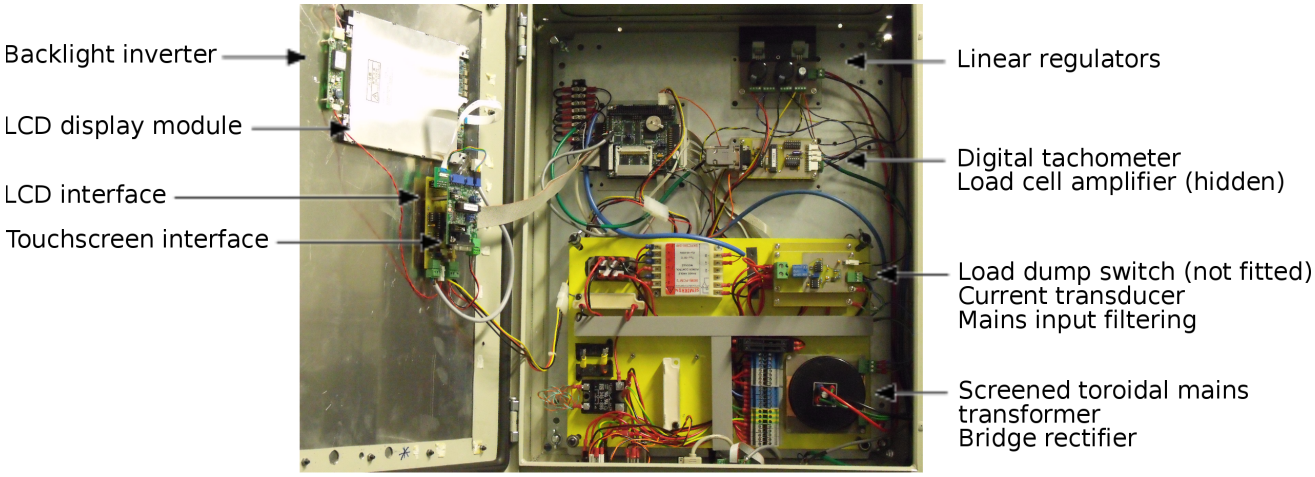

Figure 4.7: Dynamometer power electronics and control box inside view

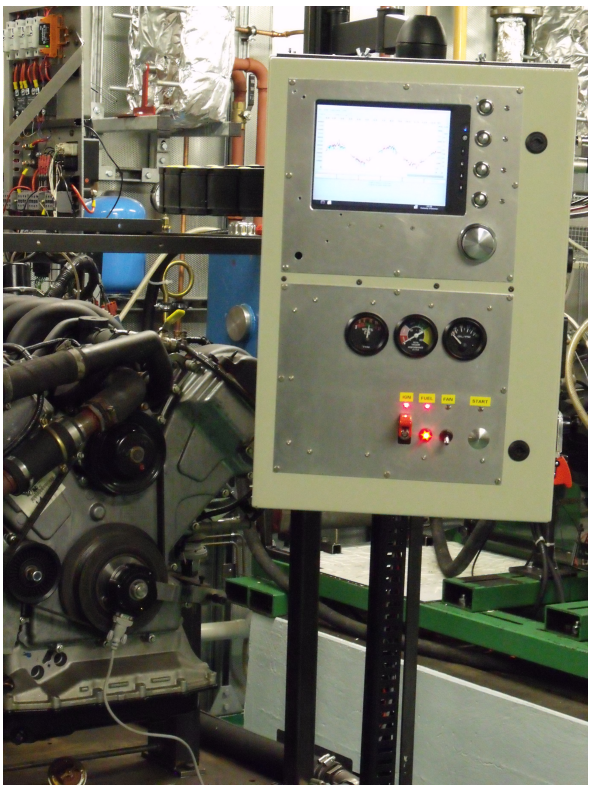

Figure 4.8: Pendant operator's box

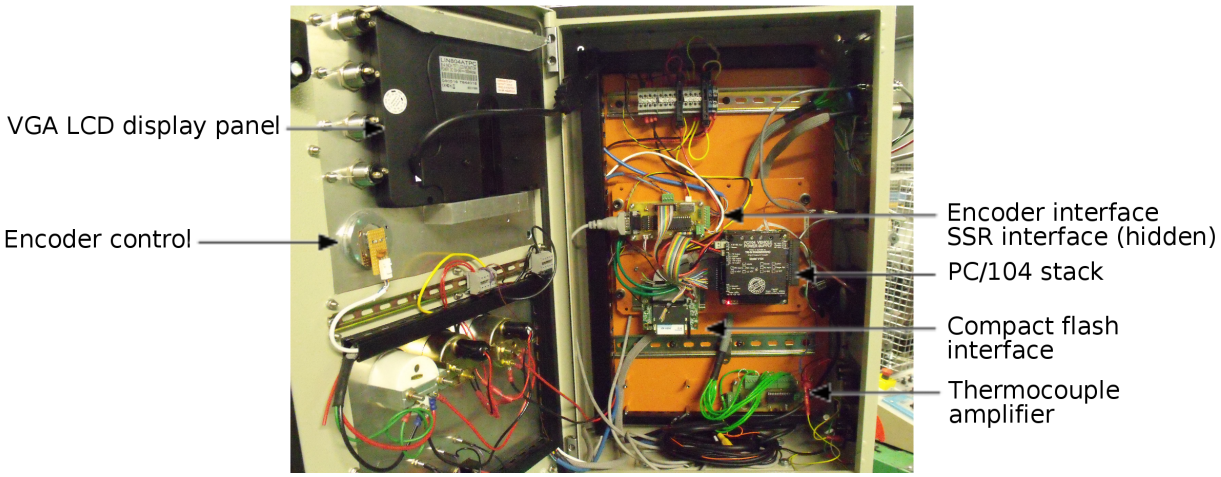

Figure 4.9: Inside the pendant operator's box

4.2 The Dynamatic Dynamometer

There are currently several dynamometer designs in widespread use. Each type has either cost or performance benefits over its counterparts.

- Hysteresis
- DC and AC generator

- Wet and dry internal rotor
- Air-cooled external disc rotor
- Water pump (Heenan Froude)

It is thought that the original Dynamatic company eddy-current dynamometer was manufactured Heenan Froude in the UK under licence from Dynamatic. Historically, there was then a split of the Heenan Froude and Dynamatic companies. For this reason it is now not clear who holds exact specification details of the MK1 machine used for this project, although when contacted, Froude still seem to have some of the original technical drawings. Similar designs are still made and reconditioned by the DyneSystems Inc. in the USA, who took over the original Dynamatic Company's manufacturing plant for the US market. Their company website gives the following details:

> Dynamatic Dynamometer & Engineering Company contracted with Midwest Dynamometer in the 1920's to build its eddy current dynamometers, but in the 1960's, under the ownership of Eaton Corporation, utilised its engineering capabilities to design higher speed, lower inertia dynos. - www.dynesytems.com

Figure 4.10 shows an annotated cross-section of the MK1 design, and Figure 4.11 shows a partial schematic of the retro-fitted Froude controller that was inherited with the dynamometer at the beginning of this project. The design of both the machine and controller is discussed further throughout the rest of this chapter.

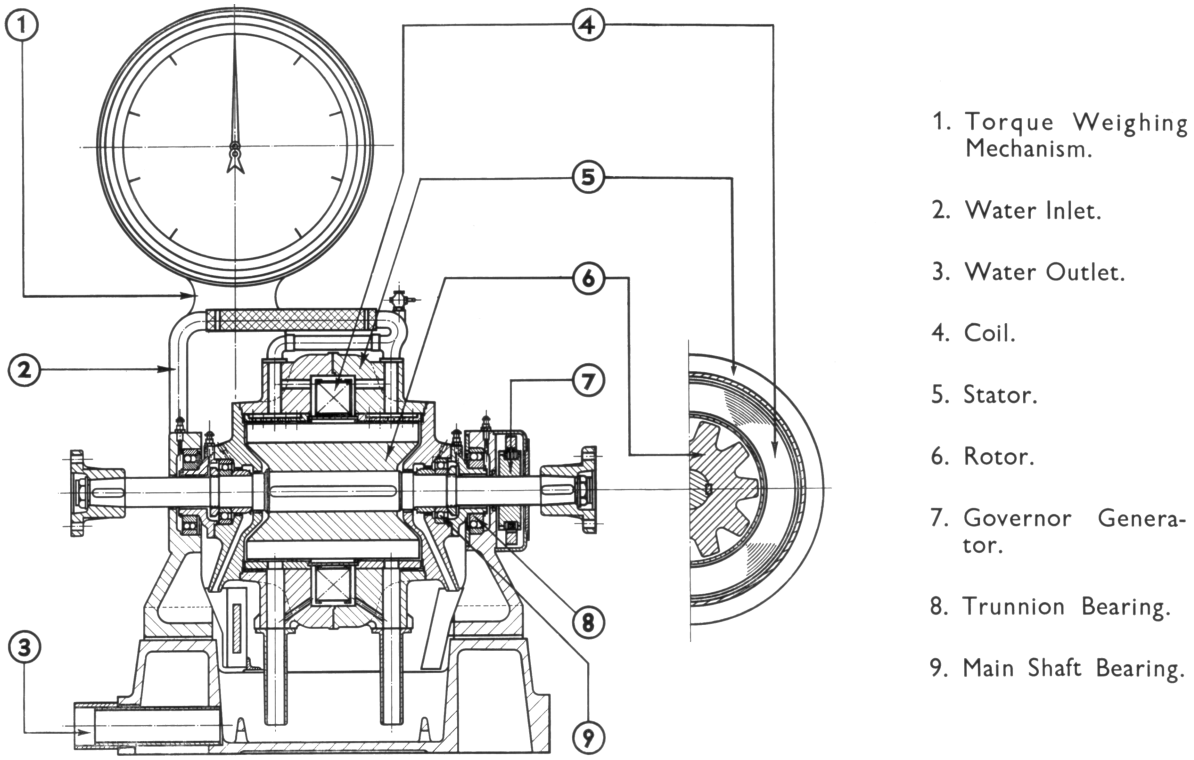

Figure 4.10: Heenan Froude G.V.A.L. Mk1 dynamometer cross-section

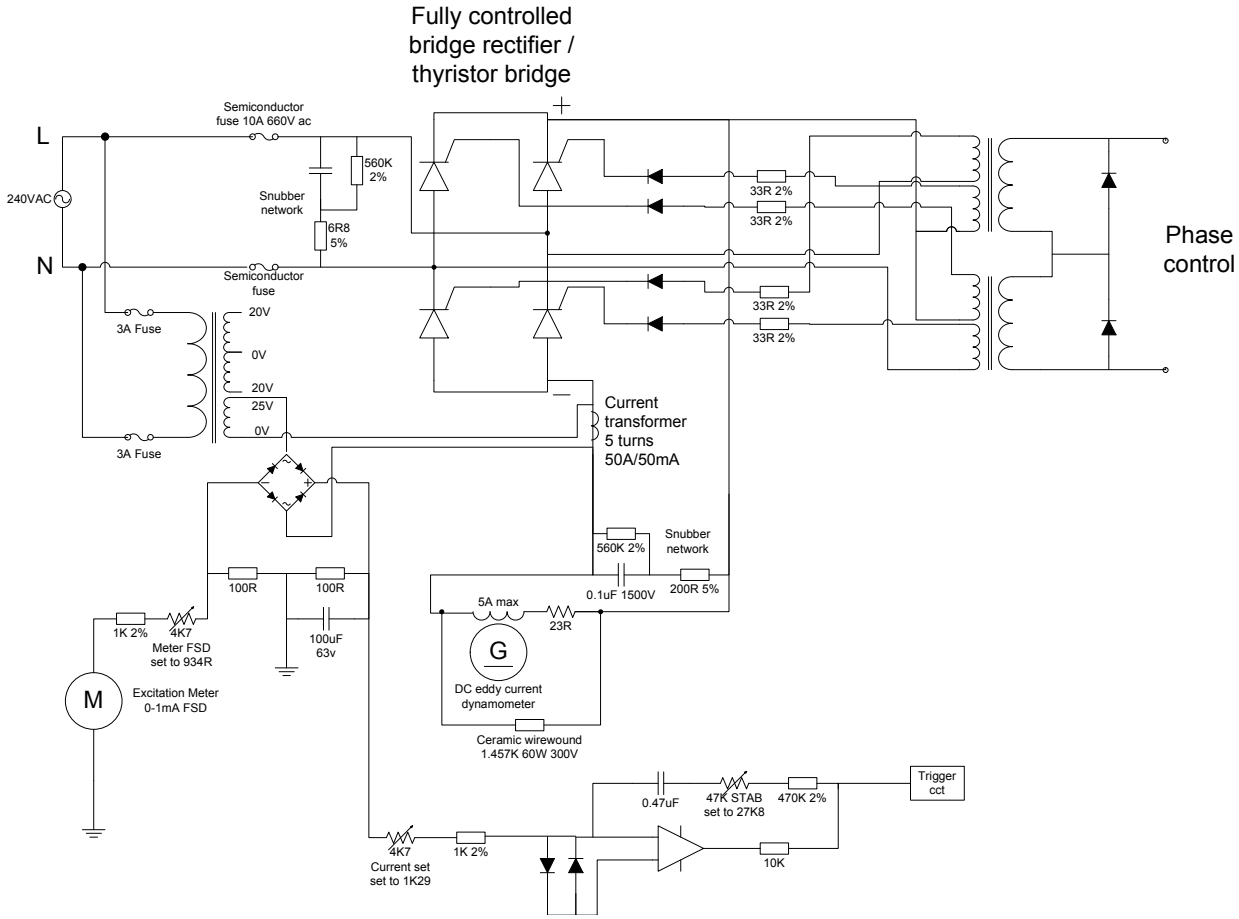

Figure 4.11: Simplified schematic of the original current controller

4.2.1 Assessment of Controllability

It can be seen from Figure 4.12 the G.V.A.L. Mk1 rated torque exceeds that of the 3.2L AJ26 V8 engine for a region between 600 to 3800 rpm. Thereafter, the steady state torque must be reduced parabolically (constant power) to stay within the 150 BHP rating of the machine. This means that for speeds above 3800 rpm the engine cannot be run to its full throttle torque limit for any length of time due to heat dissipation limitations.

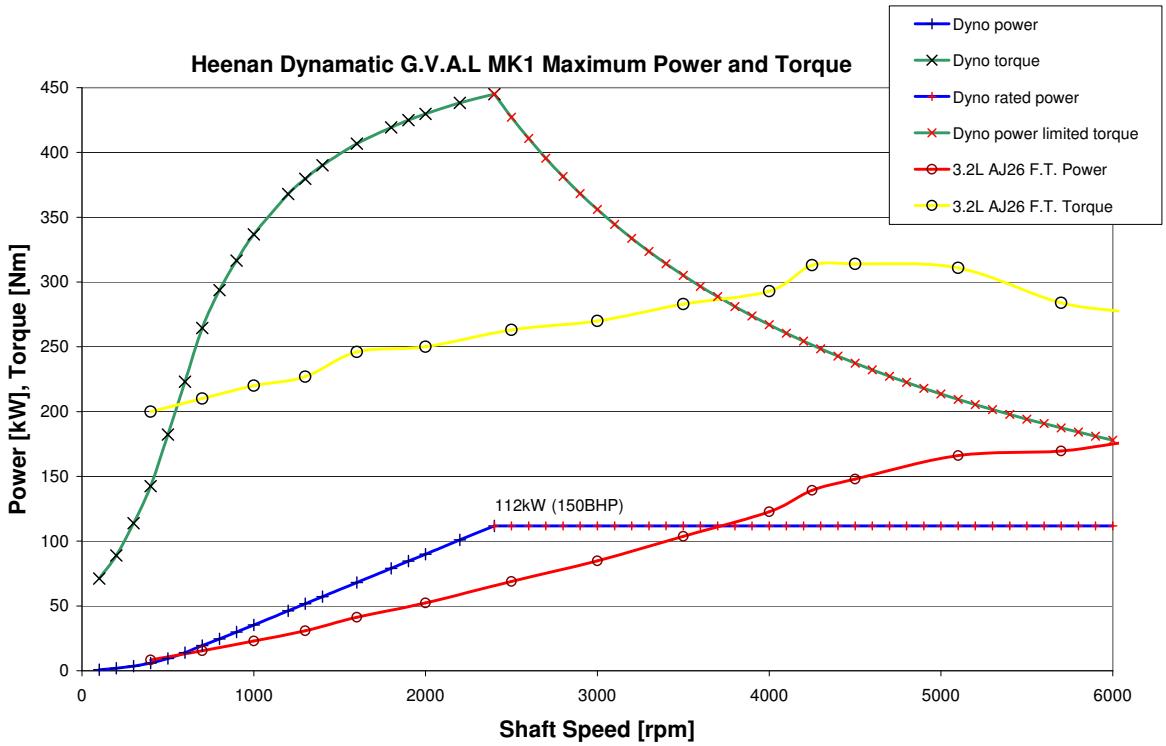

Figure 4.12: Dynamometer and AJ26 V8 engine maximum power and torque curves shown together for comparison

As the dynamometer is now an old piece of equipment, some analysis of the controllability of the combined engine-dynamometer system is beneficial before attempting to realise the controller. It is required to *hold* the engine at a particular load-speed operating point using a combination of engine throttling and dynamometer braking load. Initially, the case of fixed throttle angle is considered. This leaves the brake torque from the dynamometer as the controlled system input and the engine speed as the system output. In physical terms the operator will open the throttle to an arbitrary set-point and the controller will attempt to regulate the engine speed to a predetermined-determined set-point by applying a braking load to the engine. It should be noted that this type of dynamometer can not *motor* an engine. That is, it cannot drive the engine any faster than it is running at a given time by providing additional torque. It can only reduce the braking torque it is applying and thus allow the engine speed to rise of it's own accord. As engine torque output for most of the operating range is proportional to speed, the torque will usually fall with speed and hence also with applied

braking load. Constant speed is achieved when the braking torque equals engine torque, so the engine speed will only rise if the engine torque exceeds the braking torque. The consequence of this is that under fixed throttle conditions the dynamic behaviour of the system will be different when the engine speed is being reduced to when it is being *allowed* to increase. Also if the controller applies too much torque too quickly a run-away condition may occur where the engine torque begins to collapse, which when combined with a slow dynamometer response, may cause the system to enter a limit cycle leading to an engine stall. This is most likely to occur when the torque-speed gradient for the dynamometer at a given excitation current exceeds that of the engine for a given throttle opening, and particularly when at full throttle as it takes time to *dump* the energy stored in the dynamometer's field and the engine is unable to provide more torque.

Both the engine and dynamometer exhibit strongly non-linear characteristics which makes it difficult from the outset to know if a particular engine-dynamometer pairing is controllable from the point of view of the inertias in the system and the response time of the dynamometer. An assessment of whether or not the engine-dynamometer system is speed controllable at a particular operating point using load torque (or equally engine throttle by the same process) follows.

Consider first, the moment of inertia of the combined engine-dynamometer system 4.1.

$$J\frac{d\Delta\omega}{dt} = \Delta T_E - \Delta T_L \tag{4.1}$$

Where J is the moment of inertia of the combined system, ΔN is a small change in engine speed N, engine torque T_E is some surface or unknown function of engine speed N and throttle angle θ:

$T_E = F(N, \theta)$

and dynamometer brake torque or load torque T_L is some surface or unknown function of the field current I and engine speed N:

$T_L = F(I, N)$

A linear approximation can be made using the first term of a Taylor series expansion for a speed set-point, which can be written as:

$$\Delta T_E = \frac{\partial T_E}{\partial N}\Delta N + \frac{\partial T_E}{\partial \theta}\Delta\theta \tag{4.2}$$

$$\Delta T_L = \frac{\partial T_L}{\partial I}\Delta I + \frac{\partial T_L}{\partial N}\Delta N \tag{4.3}$$

where $\Delta\theta$ is a small change in throttle angle. Since a single set-point is being considered the partial gradients can be represented by the unknown constants $\beta_1, \beta_2, \alpha_1, \alpha_2$ valid only at the set-point:

$$\Delta T_E = \beta_1 \Delta N + \beta_2 \Delta\theta \tag{4.4}$$

$$\Delta T_L = \alpha_1 \Delta N + \alpha_2 \Delta I \tag{4.5}$$

Substituting 4.4 and 4.5 back into 4.1 we have 4.6.

$$J\frac{d\Delta N}{dt} = (\beta_1 - \alpha_1)\Delta N + \beta_2 \Delta\theta - \alpha_2 \Delta I \tag{4.6}$$

Let us assume that the effect of speed on load torque is small, thus we can let the gradient tend toward zero, $\alpha_2 \to 0$. To simplify further we can consider a constant throttle angle, allowing only load torque to be varied to control speed, $\Delta\theta \to 0$ and hence β_2 is redundant as well so that 4.6 simplifies to 4.7.

$$J\frac{d\Delta N}{dt} = \beta_1 \Delta N - \alpha_2 \Delta I \tag{4.7}$$

Converting to the Laplace domain:

$$(Js - \beta_1)\Delta N + \alpha_2 \Delta I = 0$$

We will (for now) represent the dynamometer electrically by an inductor with in-series resistance 4.8.

$$\Delta V = L\frac{d\Delta I}{dt} + R\Delta I \tag{4.8}$$

where L is the inductance, R is the in series resistance and V is the voltage applied across the windings. Using the Laplace operator again

$$\Delta I = \frac{\Delta V}{Ls + R} \tag{4.9}$$

By combining 4.7 with 4.9 we arrive at:

$$(Js+\beta_1)\,\Delta N+\frac{\alpha_2\Delta V}{Ls+R}=0 \tag{4.10}$$

We can now add a simple proportional control law with a negative sign to correspond with the braking action 4.11.

$$\Delta V=-K\Delta N \tag{4.11}$$

Adding the control law to 4.10 gives the characteristic equation for the closed loop system 4.12.

$$\{(Js+\beta_1)\,(Ls+R)-\alpha_2K\}\,\Delta N=0 \tag{4.12}$$

Multiply out and collect terms:

$$\left\{JLs^2+(JRs-\beta_1L)\,s-(\beta_1R+\alpha_1K)\right\}\Delta N=0 \tag{4.13}$$

By inspection of 4.13, without resorting to the Rough-Hurwitz method for determining stability, we can observe the necessary conditions $JR>\beta_1L$ and $\beta_1R>\alpha_1K$, which suggests that too little inertia or too much proportional gain may make the system unstable for a given operating point, neither of which is greatly surprising.

It is probably not a safe assumption that the dynamometer's torque sensitivity to speed is much less than the engine's for much of the operating range. A variation of the previous approach requiring less simplification is to put 4.6 and 4.8 in matrix form as 4.14 to allow a state-space analysis.

$$\begin{bmatrix} J & 0\\ 0 & L\end{bmatrix}\begin{bmatrix}\Delta\dot N\\ \Delta\dot I\end{bmatrix}=\begin{bmatrix}\beta_1-\alpha_1 & -\alpha_2\\ 0 & -R\end{bmatrix}\begin{bmatrix}\Delta N\\ \Delta I\end{bmatrix}+\begin{bmatrix}\beta_2 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\Delta\theta\\ \Delta V\end{bmatrix} \tag{4.14}$$

Which can be rearranged into state space form $\dot{\underline{x}}=A\underline{x}+B\underline{u}$ as 4.15.

$$\begin{bmatrix}\dot{\underline{x}}\\ \Delta\dot N\\ \Delta\dot I\end{bmatrix}=\overset{A}{\begin{bmatrix}\frac{(\beta_1-\alpha)}{J} & -\alpha_2\\ 0 & \frac{-R}{L}\end{bmatrix}}\begin{bmatrix}\underline{x}\\ \Delta N\\ \Delta I\end{bmatrix}+\overset{B}{\begin{bmatrix}\frac{\beta_2}{J} & 0\\ 0 & \frac{1}{L}\end{bmatrix}}\begin{bmatrix}\underline{u}\\ \Delta\theta\\ \Delta V\end{bmatrix} \tag{4.15}$$

Consider now a proportional control law, say $\Delta V=K\,(N_{set}-N)$. If also, the throttle position is held constant so that $\Delta\theta=0$, we have:

$$K \equiv \begin{bmatrix} 0 & 0 \\ -k & 0 \end{bmatrix} \tag{4.16}$$

$$\underline{\dot{x}} = A\underline{x} + B \begin{bmatrix} 0 \\ -k\Delta V \end{bmatrix} \tag{4.17}$$

$$\underline{\dot{x}} = A\underline{x} + B \begin{bmatrix} 0 & 0 \\ -k & 0 \end{bmatrix} \begin{bmatrix} \Delta V \\ \Delta I \end{bmatrix} \tag{4.18}$$

$$\underline{\dot{x}} = (A + BK)\,\underline{x} \tag{4.19}$$

From here we can follow the techniques outlined in classical control texts, such as Dorf & Bishop (2001) and Ogata (1997) for assessment of stability and controllability. The system is said to be controllable if an unconstrained (Dorf & Bishop, 2001) external input can move the internal state of a system from any initial state to any other final state in a finite time interval (Ogata, 1997). The system's characteristic equation is formed using the A matrix:

$$det\,(s\mathbf{I} - A) = 0$$

The system is said to be stable if all of the roots of the characteristic equation have negative real parts. If we define the dimension or rank of A to be n, then the controllability matrix is:

$$R = \begin{bmatrix} B & AB & A^2B & \ldots & A^{n-1}B \end{bmatrix}$$

and the system is said to be controllable if R has full rank, i.e. $Rank\,(R) = n$.

These results can only be applied to one set-point at a time, but can be used to search for the boundaries of stability by varying the control gain and the set-point accordingly. Where both stability and controllability cannot be guaranteed (one may contravene the other) it might be indicative of there not being enough inertia in the system, or the dynamometer response being too slow, or a sensitivity to the alignment of the dynamometer-torque-speed and engine-torque-speed gradients.

This analysis has assumed that the engine and dynamometer are joined using a stiff or inflexible shaft. The actual flexibility of the propeller shaft is not known and cannot be ready calculated using hollow shaft torsion theory as the yokes and spider of the universal joints will contribute significantly to the overall flexibility and require more complicated analysis. Having flexibility will change the the transient torque behaviour making the combined inertia assumption a weaker one. The controllability assessment

also assumes that the controlling input is unconstrained i.e. no actuator saturation. In practise the applied voltage is limited to the supply voltage. The other difficulty is that unknown physical parameters such as engine inertia are required. These could be provided by manufacturer or calculated from component dimensions, but requires measurement from either unavailable dimensioned engine schematics or disassembling of the engine and dynamometer. Some dimensions such as the dynamometer rotor could be scaled from drawings, but this kind of estimation might result in a large source of error. *Typical* inertia values quoted for similar engines and dynamometers seem to vary depending upon their source and may prove to be misleading. A throttle-torque-speed mapping for the engine would be needed to asses the required gradients at each operating point considered as well as a steady-state current-torque-speed mapping for the dynamometer. The problem becomes circular in practical terms as these mappings could be experimentally obtained if the system was already operational. Finally the linear time invariant assumption (LTI) may not hold true as the machine's self inductance is vary hard to quantify due to magnetic saturation and other effects such as eddy-current damping.

Nomura et al. (2000) have modelled the engine-dynamometer system as two-masses connected via a flexible shaft. They noticed that the two-mass model can be regarded as a one-mass system when excited by a low frequency, and also as a resonant system in a high frequency domain. This allows the system's unknown parameters to be isolated from each other as the engine inertia can be identified at low frequency (assuming the dynamometer's inertia is already known) and the shaft's spring coefficient can be determined at a high frequency of torque variation. They are able to demonstrate an on-line adaptive identification for engine inertia and the spring coefficient in order to estimate dynamic engine output torque using only the torque measured at the dynamometer. The estimated torque is then fed back to the torque controller which uses the engine's throttle to regulate torque. This reduces torque overshoot due to measurement error and so decouples the engine torque interaction with the speed control loop.

Even when all of the concerns relating to unknown physical parameters are addressed the field current time constant of the dynamometer is a limiting factor to stability an controllability. No account has so far in this discussion been taken for the use of a current controller which can bring down the open-loop current (and hence flux) time constant in inverse proportion to the step change in input. The effect being that small changes to a set-point will be achieved far more quickly than large ones due to the finite voltage which can be applied. This is in itself a non-linearity before the machine's dynamics are considered. The next section describes the work undertaken to design a current controller and is flowed by a section which assess its performance using simulation.

4.3 Development of a Replacement Current Controller

4.3.1 Phase Angle and Current Controller

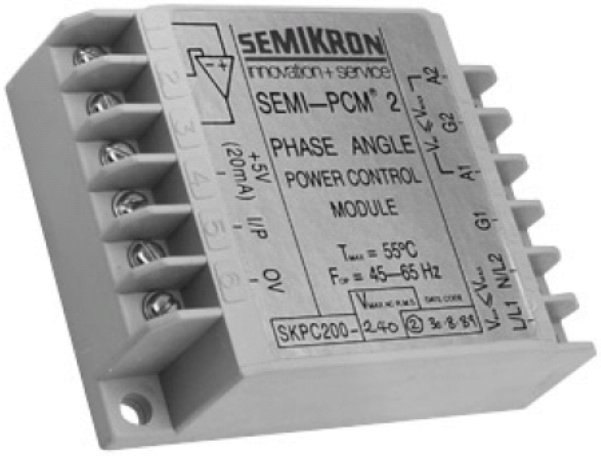

Figure 4.13: Semikron phase angle controller

The Semikron phase angle controller (Figure 4.13), is a sealed unit for isolated control of a resistive or inductive load from a single phase 240 V AC mains supply. The load can be supplied with AC using a triac or DC using one or more thyristors or a group of thyristors mounted in a single package which is called a silicon controlled rectifier (SCR). Open loop control of a load using phase angle control is inherently non-linear as it is achieved by *chopping* a sine wave in linear proportion (in this case the mains supply). Providing a current feedback path to adjust the phase angle set-point effectively linearises the system's input-output relationship. It also removes output drift caused by variations in mains supply RMS voltage and changes in the load's characteristics (e.g. with temperature) to ensure a fixed steady-state input-output relationship. The linearisation achieved by closed-loop control is shown in Figure 4.14.

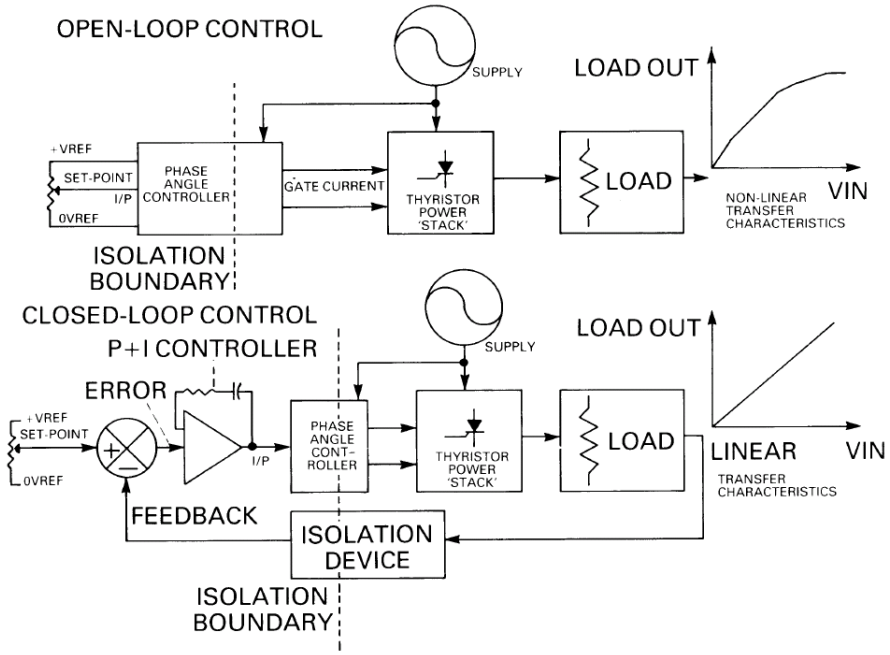

Figure 4.14: Comparison of open and closed loop current control modes

An example control circuit provided with the Semikron unit is illustrated in Figure 4.15. In this circuit the load is supplied with AC from a triac unit. The load current is passed through a current sensing transformer. The output from the transformer is then passed through a bridge rectifier then connected to a burden resistor. The current through the burden resistor produces a voltage across it which is directly proportional to the load current being measured. The choice of *burden* resistor value effectively sets the scaling or feedback gain. Inside the Semikron unit there is an uncommitted op-amp which is used with some external components to produce a PI controller. The controller set point is adjusted using a potentiometer to produce a reference voltage of between 0 and 2 V. The maximum load current that can be set is at 2 V and is determined by the choice of burden resistor.

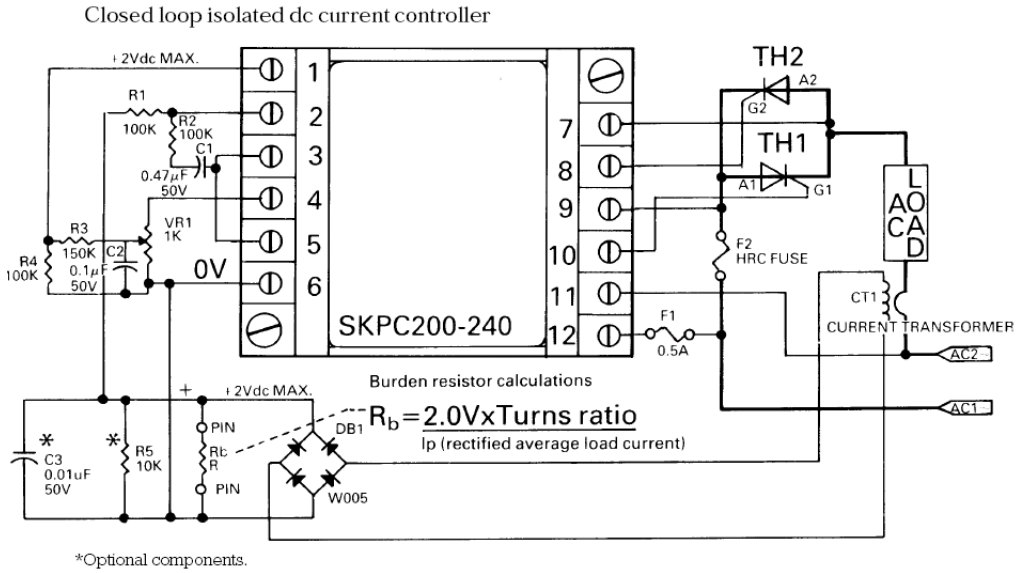

Figure 4.15: Semikron current control circuit for an AC load

The Semikron provided circuit was modified for use with the dynamometer which requires a DC supply. An SCR unit was used in place of the triac to provide a rectified DC supply instead of AC. *Off-the-shelf* phase angle controllers that could be sourced were found to be designed only to be able to trigger half bridge thyristor units. To avoid the development time of designing, assembling, testing a custom phase angle controller, a commercial off-the-shelf controller was required and so a half-bridge SCR (Figure 4.16) was selected to compliment it, and are more readily available than full bridge four-thyristor versions. Half bridge SCRs have a pair of diodes which replace two of the

Figure 4.16: P102W SCR

thyristors (Figure 4.17). The half bridge versions can be shown to regulate the current through a load identically to a full bridge, but at a lower device cost and with reduced triggering complexity. One limitation is that reverse-generation is not possible with the half-bridge configuration.

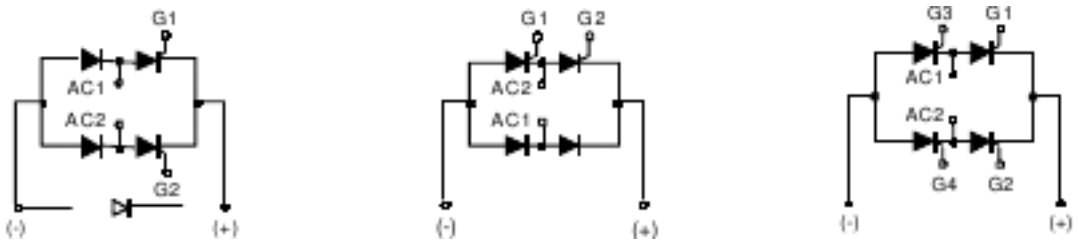

Figure 4.17: P series SCR configurations

A Semikron snubber unit was connected across the SCR to protect it and help prevent false triggering. Snubbers usually form an RC filter, but this device also incorporates a varistor to suppress transient voltage spikes. A pair of ceramic power resistors were also connected across the load to replace a large wound resistive heater element that was used with the original dynamometer controller, mounted in the operator's box.

Figure 4.18: LEM hall effect current sensor

Current sensing transformers tend to be large and are often rated for currents much higher than required for this application. The original Froude controller used a current sensing transformer which was rated for a much higher current (50 A) than the dynamometer (5 A). For the replacement controller, a hall effect current transducer was selected instead (Figure 4.18) as these devices are smaller, can be more cost effective, and are be found in measurement ranges better matched to this application. The device chosen has an output voltage measurement offset of 2.5 V obtained using an internal reference. This is to allow for measurement of current in both directions without the need for a negative supply rail. For this application the current will only flow in one direction so the offset was removed using a c zener reference in combination with an instrumentation amplifier (AD623) to apply the offset whilst also allowing scaling of the feedback signal to a 0-2 V range in the same way as a *burden* resistor is used in the original Semikron circuit. The modified circuit design also has provision for the potentiometer signal to be switch out and replaced by a DAC signal for microprocessor interfacing and also to allow for under/over-speed solid state protection using comparators and potentiometer set references. The DAC signal was used to provide closed-loop speed control and is discussed in the following sections. The circuit is shown in Figure A.1.

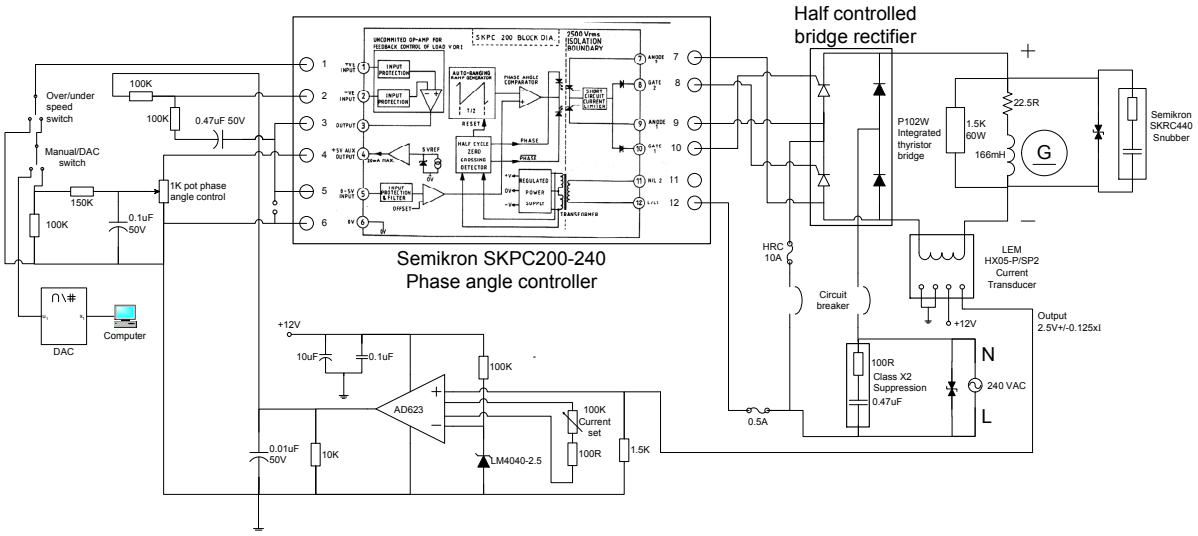

Figure 4.19: Current controller schematic

Initial component values for the PI controller were taken from the Semikron example circuit. The circuit was tested and found to correctly control the current which was monitored independently using a digital ammeter. After testing, further modifications were made to the circuit. It was found using simulation that the time response for the current to settle could be improved so some of the PI related components were changed accordingly with ones that more closely matched those of the iteratively tuned simulation model. An initial estimate for the winding inductance of 40 Henrys was used in the model based on the open-loop response time and the measured winding resistance. This estimate was based on the measured winding resistance and the fact that it took approximately 10 seconds for the current to reach an apparent steady state when measured on an in-series connected ammeter. An integrator wind-down problem was observed when the controller was left at zero demand for a period of time. The phase angle controller's internal op-amp is supplied from dual rail ± 10 V supply thus allowing a negative signal to develop. A slight mismatch between the hall effect sensor's internal reference and the external reference caused a DC offset in the current signal when no current is actually flowing through the load. This is seen by the PI controller as a small steady-state error which the integral action attempts to correct. Additionally, any noise which develops on the potentiometer reference signal or the feedback current signal was found to contribute to the initiation wind-down to the negative rail of the op-amp. A low pass filter was added to the current feedback signal to remove both noise and the 50 Hz mains supply sinusoidal component which is imposed upon it. It was not entirely necessary to remove the 50 Hz component as it was found in simulation to have little effect on the control, but it was removed to allow the current feedback to be observed more easily both in simulation and on the real system. Also the occurrence of negative direction noise spikes, resulting from stray capacitance of the cable joining the SCR to the dynamometer and fast switching of the SCR's thyristors, was significantly reduced, preventing them from being a disturbance to the controller. Leaving a small positive DC

signal on the set-point input, small enough not to trigger the lower threshold of the phase angle controller but sufficient to prevent controller wind-down was used when the set-point voltage was supplied from a DAC rather than from the manual potentiometer. For potentiometer control, a small initial delay in response from a zero set-point due to wind-down recovery was not considered important. The problem of integrator wind-up was reduced or prevented by adding a 5.6 V zener diode across the op-amp feedback path to clamp the PI controller output, rather than allowing it to rise up to the +10 V supply rail. This was necessary as the maximum input signal of the phase angle controller is 5 V for full duty and any further rise in input voltage would not have an increased effect (analogous to actuator saturation), but so long as an error persists, the PI controller would otherwise continue to increase the control signal resulting in excessive overshoot when the set-point is eventually reached.

4.3.2 Current Controller Simulation

Before the controller described in the previous section was actually constructed it was decided to verify the design using a Simulink simulation model and it's PowerSims toolbox. The PowerSims toolbox internally constructs a state-space representation of the components laid out from the graphical building blocks in Simulink. This allows models for complex systems containing power devices to be constructed in a fraction of the time it might otherwise take using basic transfer function building blocks. Figure 4.20 shows an overview of the whole system model. The system takes an input signal of 0-2 V corresponding to a control set-point of 0-5 A. There are two main subsystems which consist of the PI controller and another block which lumps the phase angle triggering unit, the SCR unit, and the dynamometer load. The remaining blocks model the response of the hall effect transducer and a low-pass filter which provide the feedback signal to the PI controller.

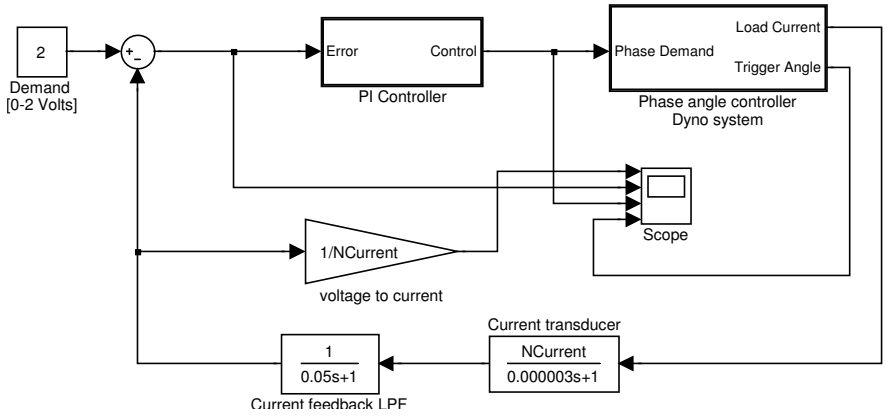

Figure 4.20: Top level of the control system model

Figure 4.21 illustrates the PI controller subsystem. The single transfer function of the op-amp arrangement is shown unconnected for reference, but the terms were separated

out to allow explicit saturation limits to be placed on the integrator. This approach also allows a *Scope* block to be separately attached to the *P* and *I* terms so that their individual contribution to the controller's response can be observed.

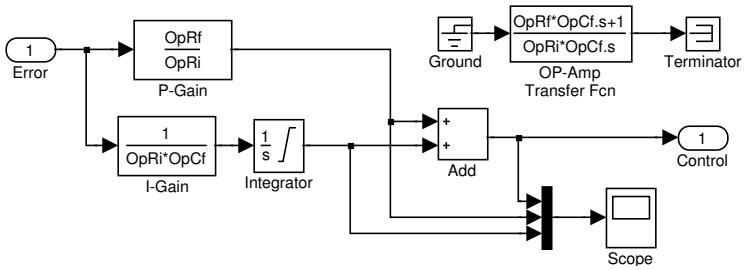

Figure 4.21: PI Controller subsystem

Figure 4.22 shows the main subsystem consisting of the supply, load, SCR bridge, and phase angle trigger system. This might have reasonably been broken down further into separate system blocks, except for the fact that it is constructed partly from PowerSims blocks which cannot be directly mixed with other Simulink blocks. Connection from the PowerSims model to the rest of the system is made through dedicated interfaces on blocks representing the ammeter, voltmeter, and thyristor gates. The model represents the functionality of the Semikron phase angle trigger unit and the SCR bridge with snubber. The supply is represented with some line impedance added to remove unrealistic very high magnitude current spikes from the supply at switching events. The other components such as the ballast power resistor and dynamometer windings are also represented by the model. The figure also shows the inclusion of a bypass resistor internal to the dynamometer which was added at a later stage and discussed in the sections that follow.

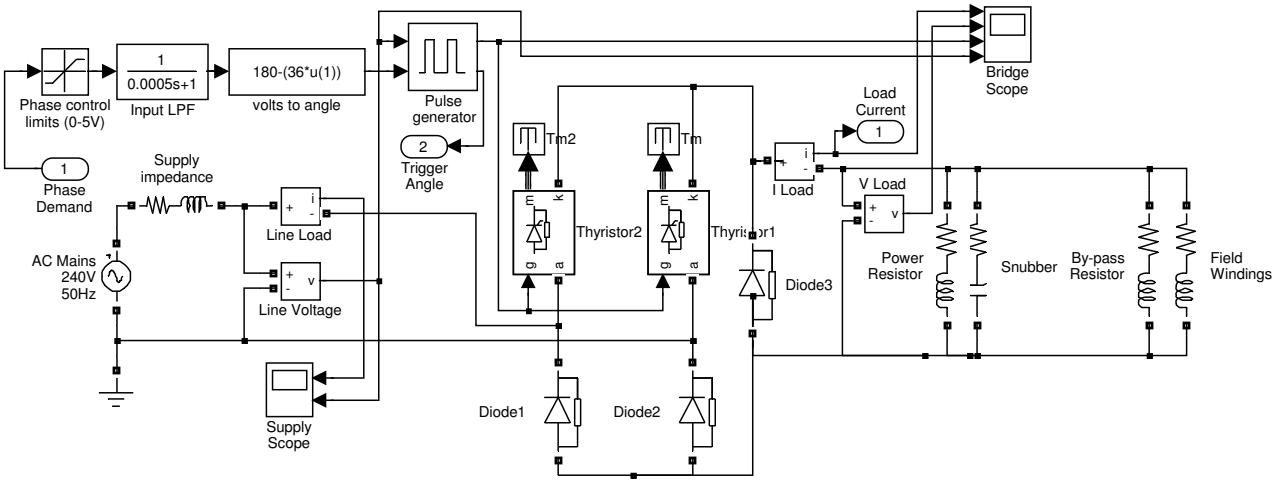

Figure 4.22: SCR bridge, phase angle control, supply and load

There is one further subsystem which consists of the phase detection and gate trigger pulse generation shown in Figure 4.23. This works in a similar way to the Semikron unit, using a ramp function, but takes into account the discrete nature of the simulation system.

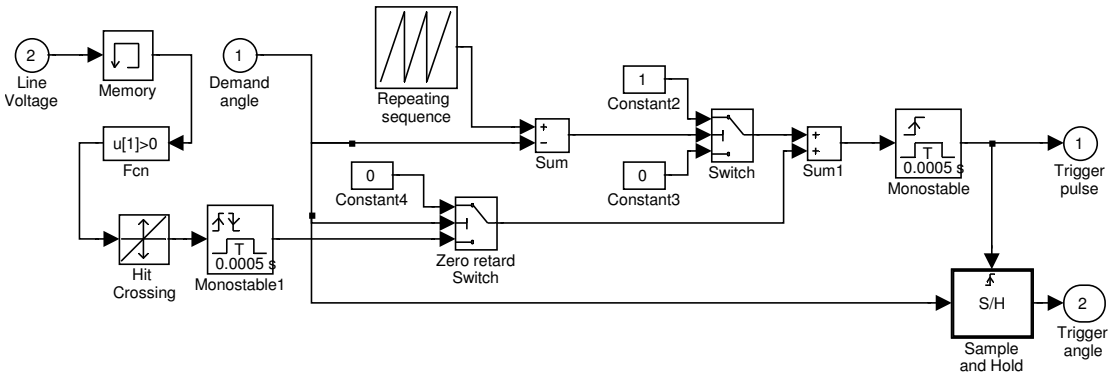

Figure 4.23: Phase detection and gate trigger pulse generation subsystem

The model confirmed that the system behaves in a stable and controllable manner in the presence of supply and load current ripple, and the physical system was constructed accordingly. Once an improved estimate for the winding inductance was obtained, the model was then used to improve the tuning of the actual system as physical component substitution is much more difficult than parameter adjustment in the model.

4.3.3 Controller Transient Behaviour

Although the controller described in the previous section worked well at achieving and maintaining a steady state current, the response time of the system was found to be very slow. This problem is inherent from the properties of the dynamometer. However, given the limitations of the plant, it was considered important to make sure that the controller was achieving a good near optimum response to keep the long rise time to an absolute minimum. As the model shown in the previous section is relatively complex it runs quite slowly and the result of any changes can take significant time to be seen, so a reduced model was made to study the effectiveness of the PI controller. Figure 4.24 shows the simplified model, having no subsystems.

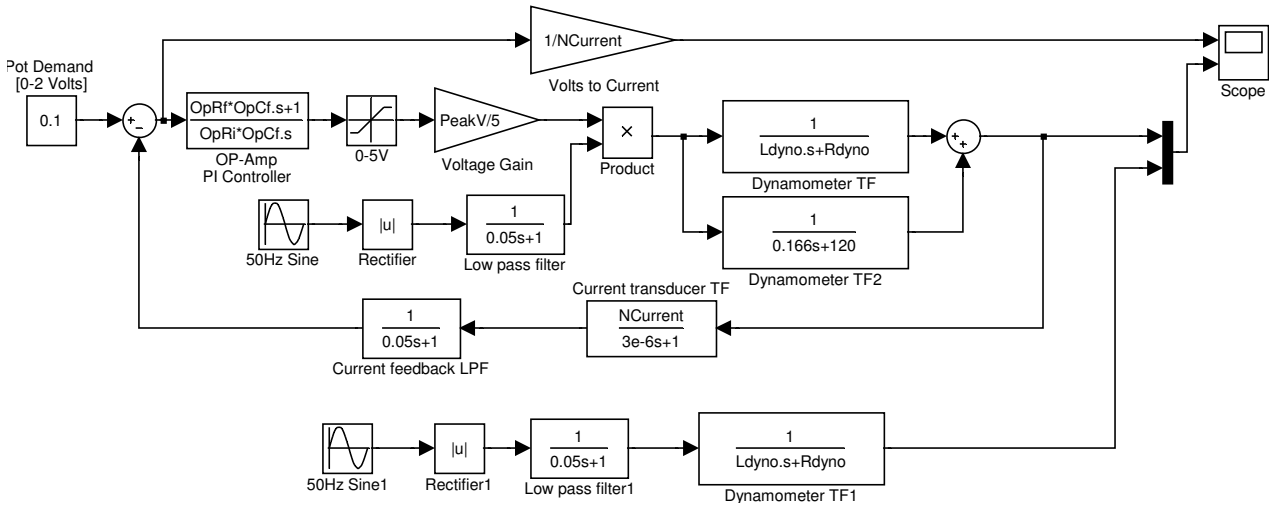

Figure 4.24: Block diagram of simplified current controller model

The controlled system is run in parallel with an identical (open-loop) dynamometer transfer function and which is given the full supply voltage as this forms the constraint for fastest possible response for comparison with the controlled response. The point of interest is when the PI controller departs from this fastest response curve to achieve its set-point. Ideally (as the system is essentially first order), this would occur at the exact instant the set-point was achieved. As the controller has a proportional term, this is not possible as the error will reduce as the set-point is approached and therefore so will the proportional term's response to it. Driving the output into saturation using a high gain may reduce this for a small step change in demand, but for a large scale change a high gain may result in an unsatisfactory overshoot, so in the absence of any derivative action, a compromise has to be reached. Figure 4.25 shows the response to a full scale demand change of 0-5 A. The controller holds the response to the maximum, but exhibits an overshoot of around 8%. It can also be seen that the full response takes around 2.3 seconds and realistically this cannot be improved upon without increasing the supply voltage. Another consequence of this is that the controlled response for smaller step changes will take less time despite the underlying time constant of the machine remaining the same. This will be an important consideration when designing the outer torque/speed control loops which in discrete form may depend on the previous control action having been completed before executing the next. For the outer controller to be fully decoupled it would have to be run at intervals longer than 2.3 seconds without addressing the need that smaller demand changes will require less time to take effect.

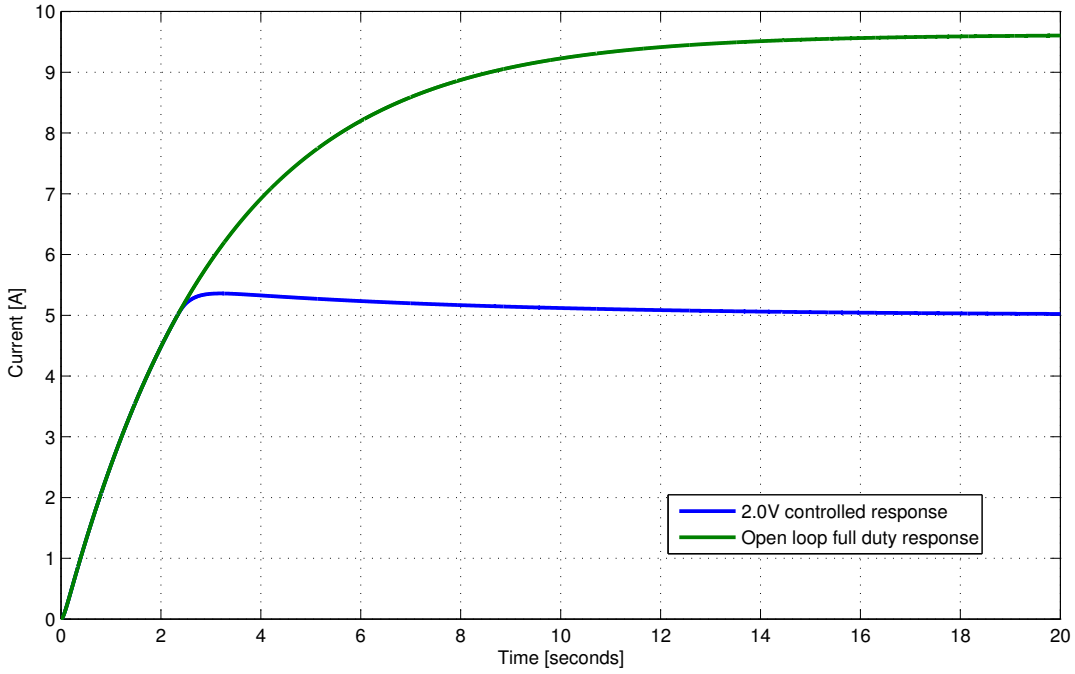

Figure 4.25: Plot of maximum demand controlled response and open loop full duty response

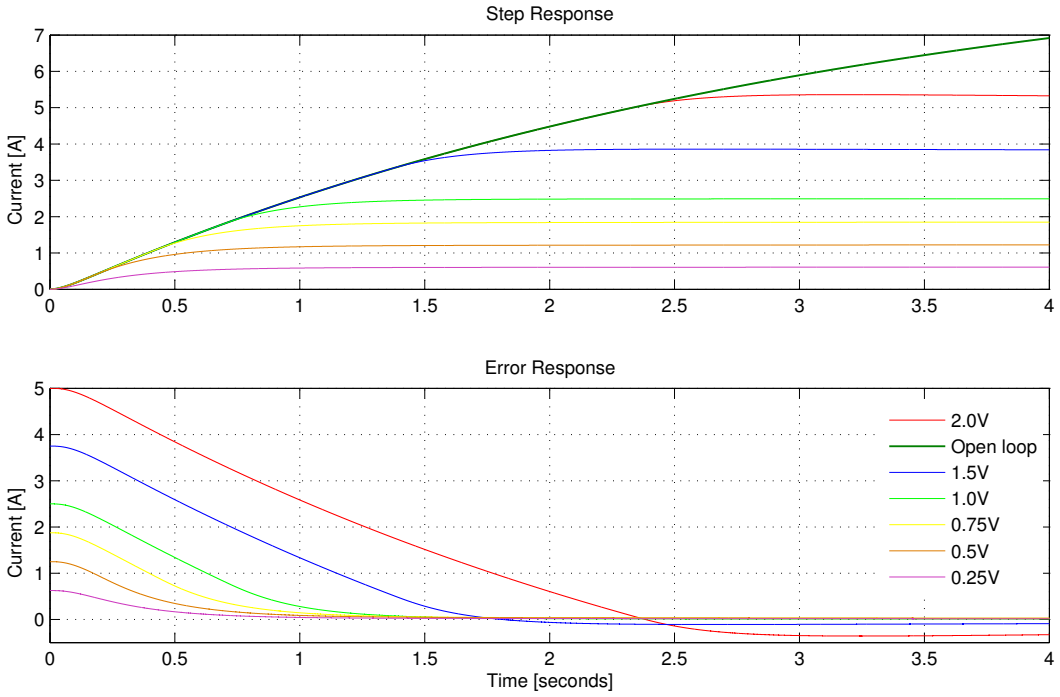

Figure 4.26: Plot of step response to various control set points and corresponding error response

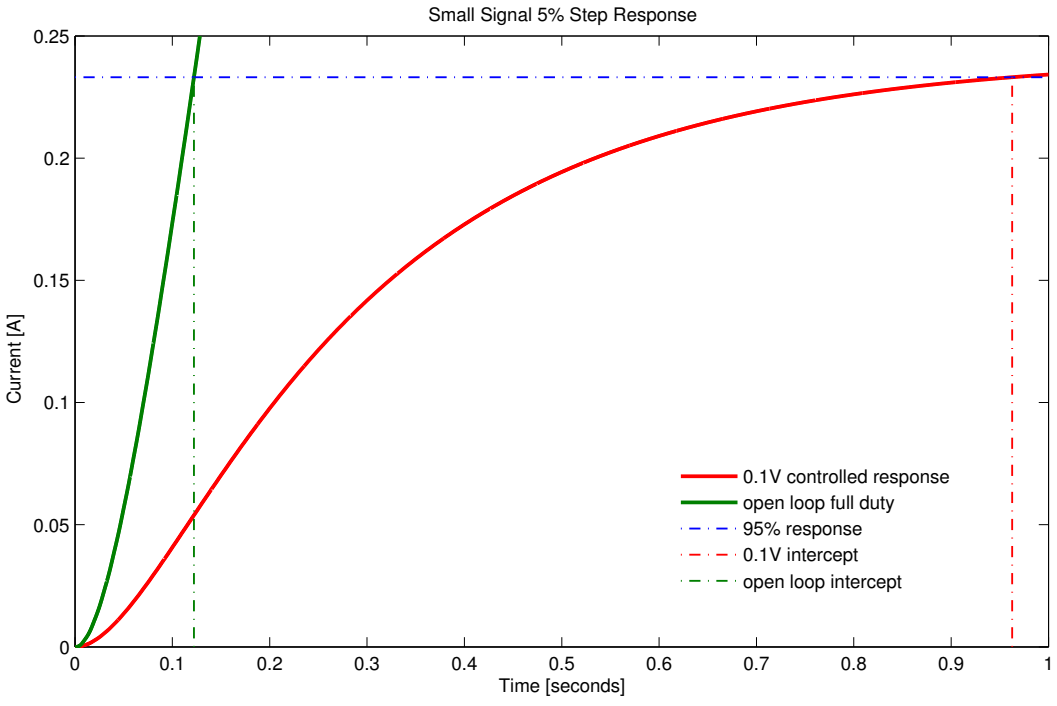

Figure 4.27: Comparison of open loop rise time with a small signal response

Figure 4.26 illustrates that small signal changes achieve their set-point in a fraction of the time of larger ones. The poor response for a small signal (5% of full demand) set-point is shown in Figure 4.27. The green curve again shows response to maximum supply

voltage applied open-loop from which it can be seen that the desired set-point could have been optimally achieved in about 120 ms, but instead took approximately 960 ms, eight times longer. Increasing the gain would improve this, but this would also increase the overshoot for the full-scale demand change to an unacceptable level, possibly exceeding the maximum current rating of the machine for long enough to cause damage.

4.3.4 Speed Control Tests using the Current Controller

The analogue current controller described in the previous section was tested in a digital PID speed control loop using dynamometer applied torque to limit engine speed. The measured torque was recorded but not used as part of the control loop, only speed. A 10 ms control loop time was used to coincide with (but not synchronised to) the mains half cycle frequency. Speed and torque were sampled one hundred times faster (10 kHz per channel) and the mean of each one hundred samples was recorded and provided at each control time step. The PID gains were not normalised to the control period, but very small gains were needed for the system to be at all stable. Tests were performed for a variety of gains, the data for three of which are shown here. The test procedure was to start the engine and allow it to idle. A constraint speed would then be set and the throttle opened manually by increasing arbitrary fixed increments. The controller would be given time to stabilise and achieve a steady state speed before the next increment was applied. It was found that although the response time and overshoot were relatively poor, the system would achieve a steady-state up until a certain throttle opening was reached whereby it would enter a limit cycle and become unstable. Some improvement may have been possible through the use of gain scheduling or some variable gain approach, but the system was found to be fundamentally unstable by a moderate throttle opening regardless of the gains used. This is due to the slow current response time and the current controller's inability to add and remove flux quickly enough. The slow application of torque using low gains is initially acceptable provided the speed overshoot is not and issue. However once torque has become too great the engine's speed drops and so does the torque which results in further speed loss. The controller reacts by removing the applied voltage, but the current continues to recirculate in the dynamometer's windings dissapated only by the winding resistance and the load dump resistor connected in parallel. When a certain throttle opening is reached, the controller is not able to act fast enough and essentially is out of phase with the system as is over-reacts providing too much torque, too late. Figure 4.28 shows a test the speed is controlled to somewhere near the set-point correcting the throttle widening disturbances, albeit with a large overshoot and long settling time of over ten seconds. Figure 4.29 shows a similar test over a longer period using a lower gain which exhibits a slower settling time. The use of a higher gain was initially more effective for wider throttle openings show in a further test plotted inFigure 4.30, but results in the onset of an unstable limit cycle which cannot

be easily recovered even with manual intervention using the the throttle.

The use of a Smith predictor was considered to correct for slow current response by treating it as a delay in the process (a technique popular in the chemical and process industry for dealing with process dead-time), but the idea was later abandoned. After some investigation of literature sources, it was apparent that this would only be appropriate for systems with fairly predictable fixed delays (such as dead-time due to transport delays) rather than slow response delays. Also, if the response delay varies with the process set-point and load conditions, then systems using a Smith predictor tends to be sensitive to errors in the modelled delay and can easily be made to have a worse performance and stability than if the predictor had not been used. Some recent progress has been made in this field, for example Panda et al. (2006) have outlined a modified Smith predictor enhanced PID control scheme which addresses its poor performance in load-change cases and Kwak et al. (1999) have outlined a new structure for the Smith predictor for use with unstable processes. It was felt that this approach would not be suitable for solving this particular problem.

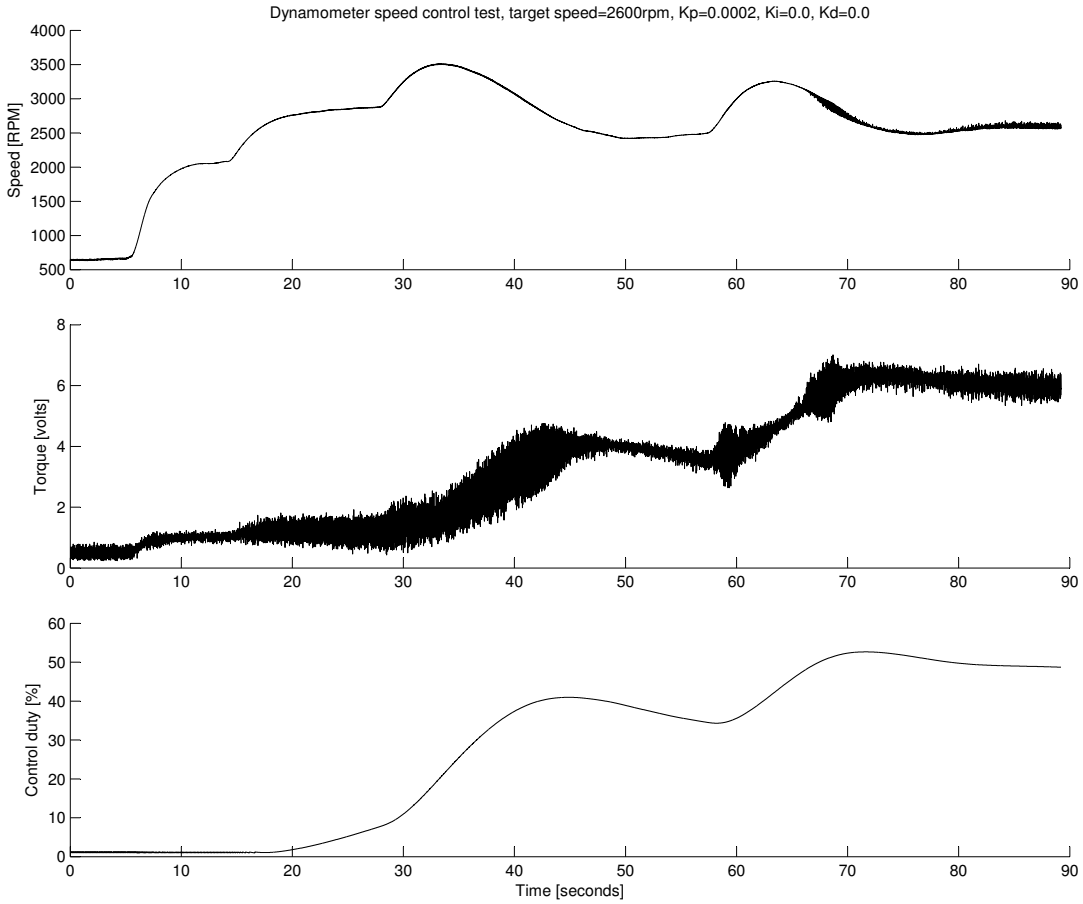

Figure 4.28: Plot of engine speed control test, medium gain

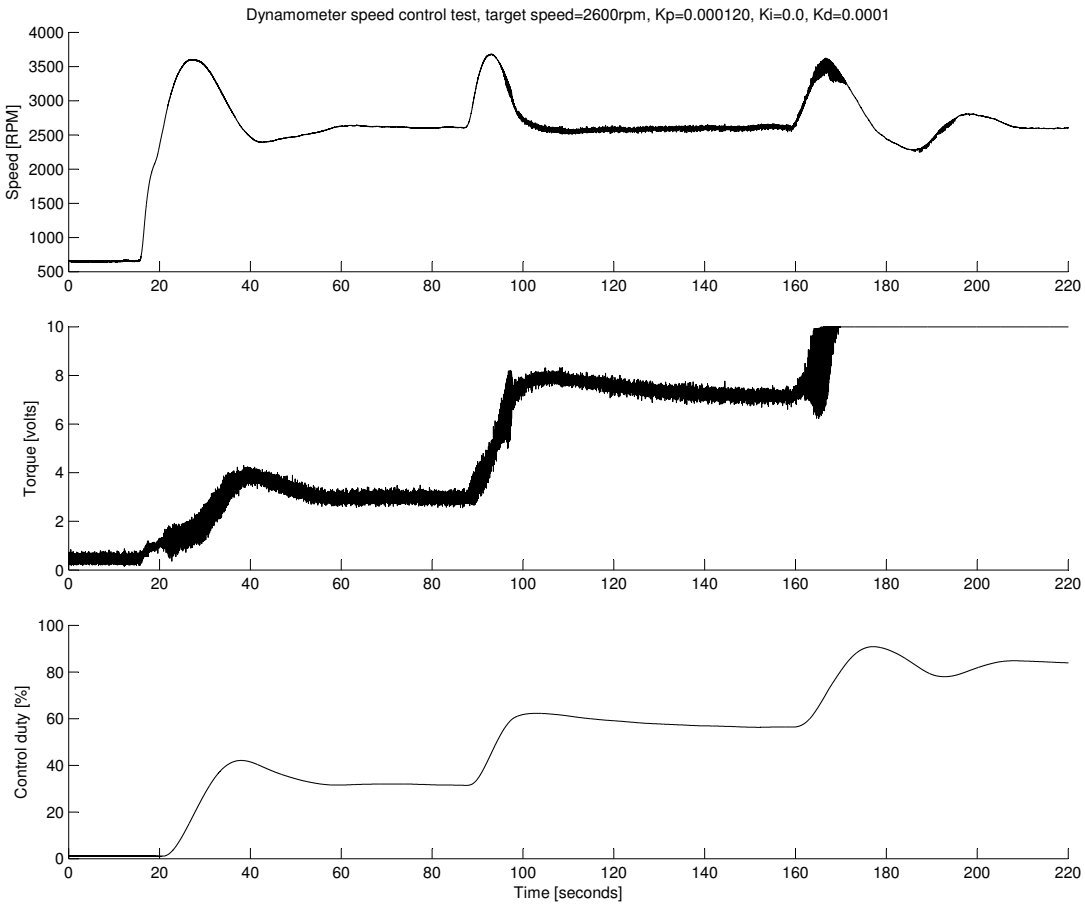

Figure 4.29: Plot of engine speed control test, low gain

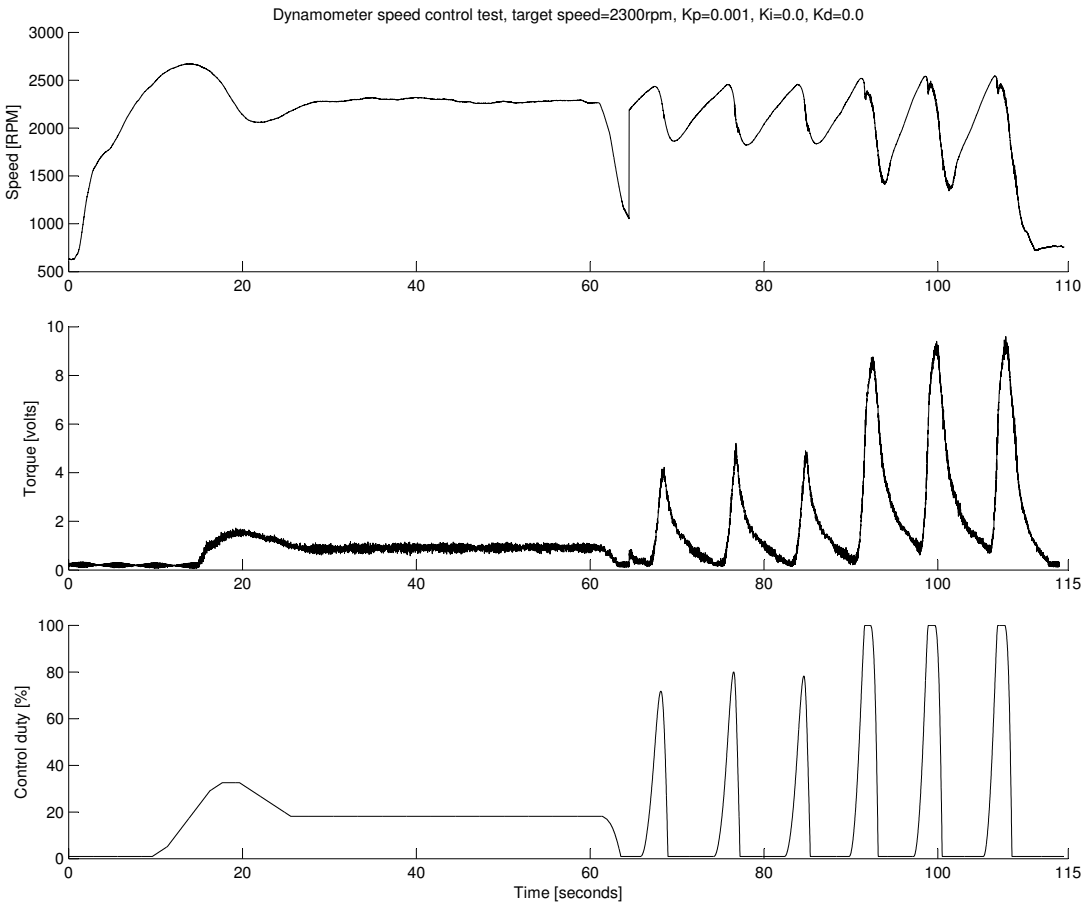

Figure 4.30: Plot of engine speed control test, high gain

It was decided that satisfactory performance could not be achieved through controller tuning or by accounting for the delay in the either the speed or torque control processes and that the current response would need to be improved through better current control. In the next section tests are described which were performed to try to experimentally quantify the current response in order that it's control might be improved.

4.3.5 Dynamometer Response Characterisation

The full current rise time was found to be considerable, taking over 10 seconds for the current to stabilise at 5 A from a zero current initial condition, despite the fact that the current response had been improved by re-tuning the current controller. It was decided to investigate the reason for the long rise time in the hope that a further improvement could be made, as the benefits of low mechanical inertia of this machine type would be negated by the very long current rise time. A step test was performed to help characterise the dynamometer's current response. The test was performed using a 30 V DC linear regulated bench power supply. A digital data storage oscilloscope was used to capture the response and was triggered using the rising edge produced when the mechanical rocker switch was closed. As the load being tested was largely inductive, switch contact bounce

was not expected to be a problem as (unlike a capacitive load) virtually no current should be drawn at the instant the switch contacts close. The result of the step test can be seen in Figure 4.32. The initial observation is the pronounced jump or step in current at the instant which voltage is applied. This initial step is a concern as it forms a disturbance to an outer torque control loop which depends upon a proportional relationship between measured field current and brake torque. The response of the step itself was captured by repeating the test at a higher sampling resolution. From the appearance of the step, it was thought that this might be due to an internal power resistor connected in parallel with the windings to assist with recirulative dissapation of storred engery during unloading and for protection of the insulation (if the windings are disconnected under load resulting in a large voltage as the magnetic field collapses). Alternative causes considered were the effect of the core not being fully magnetised until a certain minimum current flows, sufficient to polarise the magnetic domains, and also the possibility of an inrush current caused by inter-winding and inter-layer capacitance. Inter-winding capacitance is a well known phenomenon which presents itself to high frequency transformer designers, however some simple modelling revealed that the size of the capacitance that would be required to correlate with the measured data would be inplausible. Other causes for the curve shape considered were the effects of saturation and eddy-current damping. In a brief informal telephone conversation with an engineer at Froude Hofmann it was stated that the field coil was wound from a single length of wire an sealed into a tin casing using argon gas welding, a process which they no longer use and the wire was of a type nolonger available to them. This would make unfeasable the rewinding of the field coil or inspection of it to eliminate the possiblility of a partial short. As no further information was available at the time, the simple hypothesis of a parallel resistence (actual or analogous) was pursued for comparsion with the meaured data.

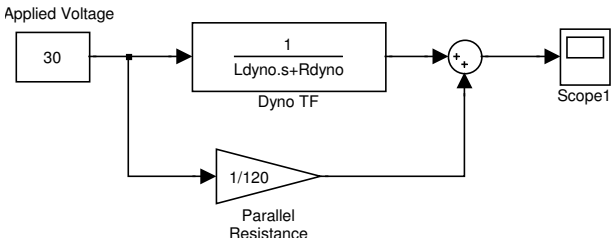

Figure 4.31: Block diagram of the simple DC step response model

The response was modelled with a parallel resistance (Figure 4.31) which results in a similar same apparent initial jump, and responses for different inductances are shown. The 70 Henry curve appears to be the best fit. A slight deviation in curvature from an ideal inductor can be seen appears similar to what can be observed due to skin effect in the windings when a high frequency AC voltage source is applied to a magnetic machine. Skin effect causes a temporary increase in the conductor effective resistivity as progressively less of the conductor's cross-sectional area is used as the frequency of the

driving voltage increases. A temporary or dynamic increase in series resistance due to skin effect in the field windings would result in a faster response as the $\frac{R}{L}$ ratio changes. However as a DC voltage source was used rather than an AC one, and with the machine stationary skin effect can be ruled out and so local saturation of part of the magnetic pathway is likely to be the cause. Partial saturation would also change the $\frac{R}{L}$ ratio, as the effective L would dynamically decrease so that the reponse is hasened slightly as R starts to predominate.

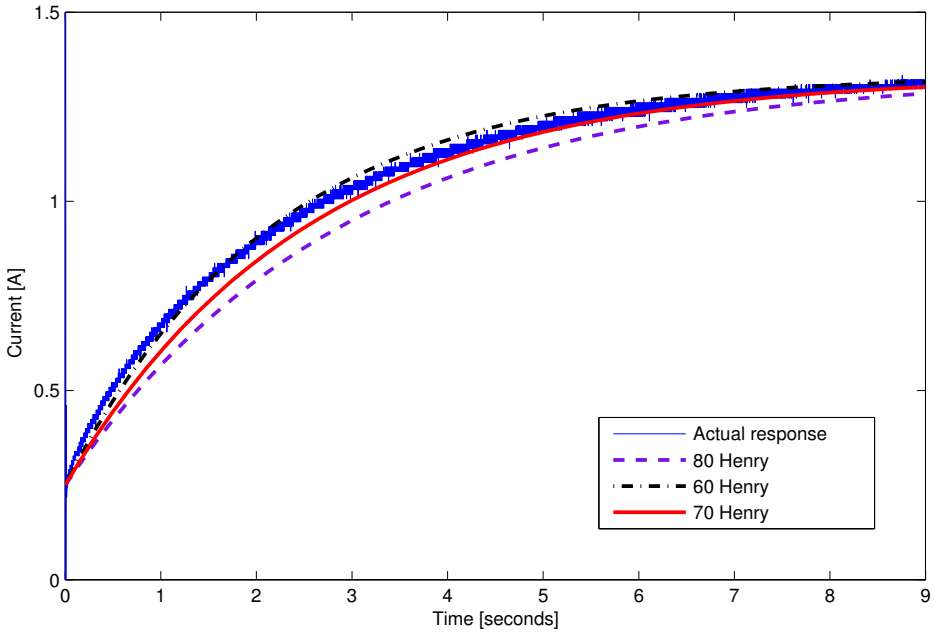

Figure 4.32: Dynamometer winding measured response to 30 V DC step input plotted against different simulated responses

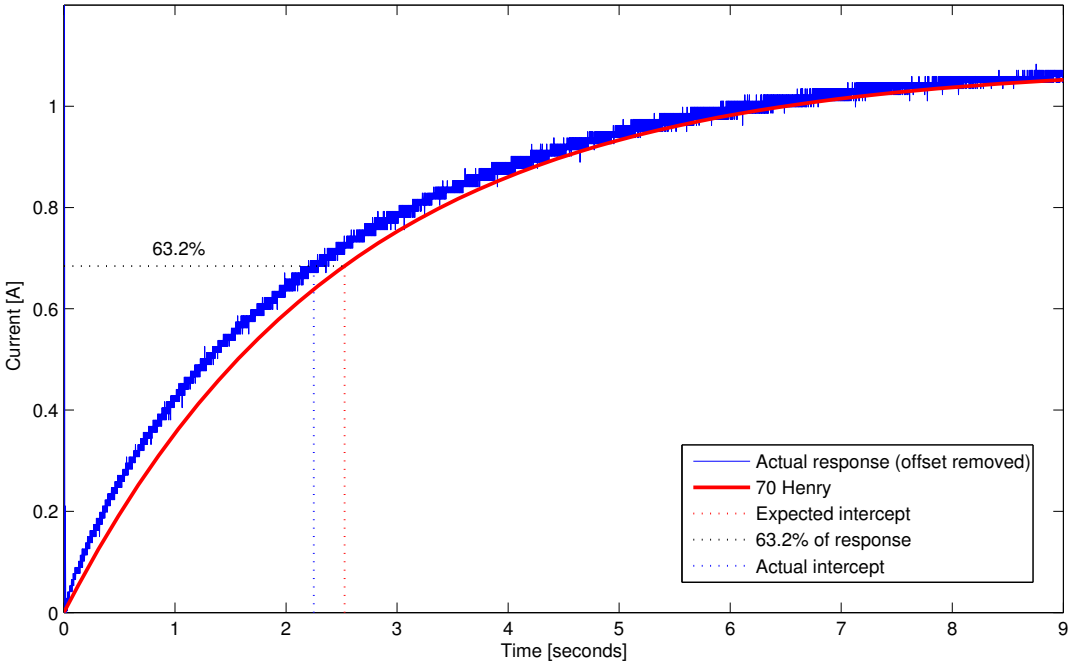

Figure 4.33: Measured 30 V DC step response, plotted with step offset removed to estimate time constant

Figure 4.33 shows the same step response, but with the initial jump removed so that the time constant of the inductive current rise may be estimated. Again the modelled response for a 70 Henry load is shown for comparison. A time constant of about 2.2 seconds can be seen for the actual sampled response and a slightly longer time constant of about 2.5 seconds for the modelled response, a difference in time constant of about 300 ms. The response is shown in Figure 4.34 from which it can clearly be seen that switch contact bounce has obscured the result. A modelled series resistive-inductive response that best fits the sampled curve is shown as well. Also an inductance of 1 Henry seems large for a device or mechanism which is intended to be largely resistive and not inductive. These factors started to suggest that the jump is more likely to be a core magnetisation effect than that of a parallel resistor.

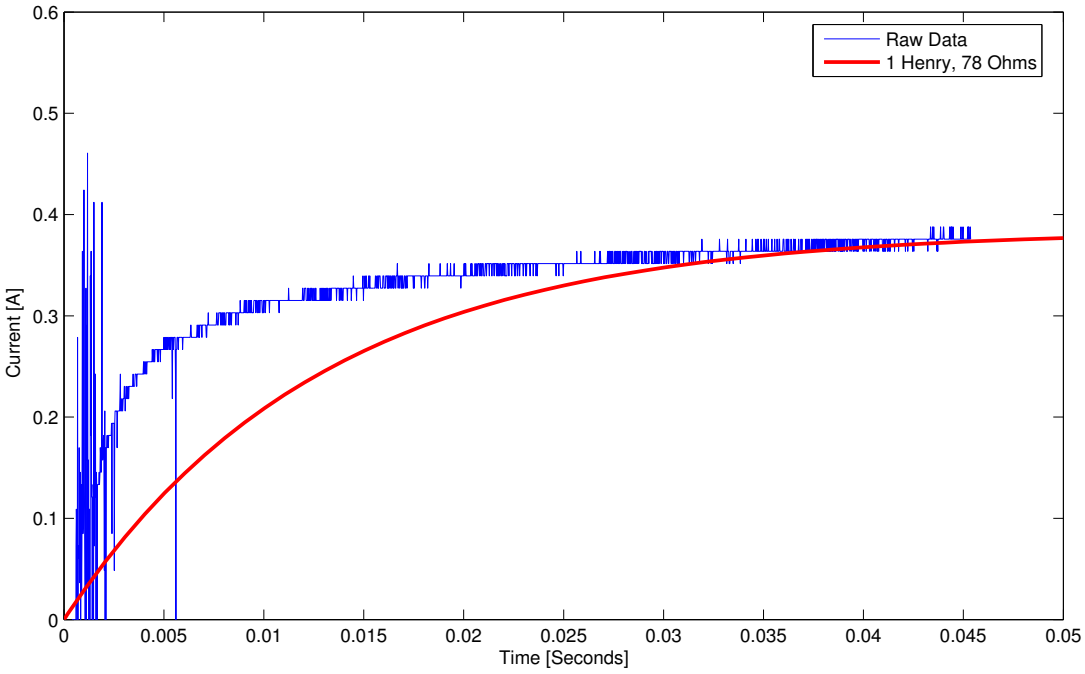

Figure 4.34: Plot of an expansion of the initial step in full response. A parallel resistance with some inductance is shown for comparison.

4.3.6 Improved Response Characterisation

Owing to switch bounce during the step tests, the cause of the initial current jump was still uncertain. From a control perspective, it was considered more important to determine whether the initial current that flows contributes to the brake torque than to understand why it occurs. It was decided that the tests should be repeated using a MOSFET instead of a mechanical switch to capture the initial response more accurately and the resulting test circuit is shown in Figure 4.35.

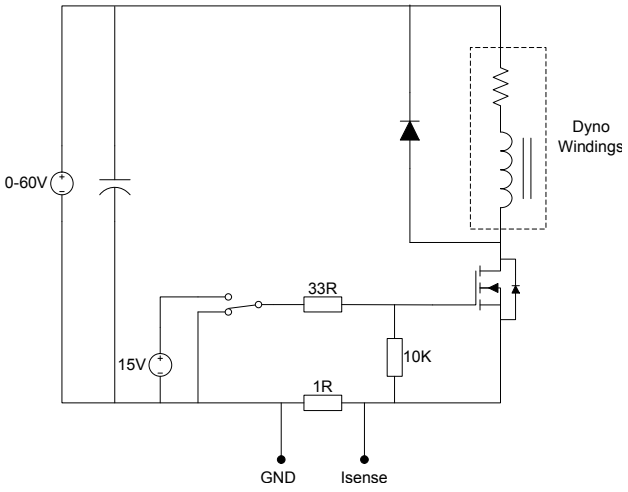

Figure 4.35: FET based step test circuit

A SPDT mechanical toggle switch was used with in-line resistance to reduce gate-ringing and a pull-down resistor to ensure the off-state is maintained. A separate 15 V supply was needed to ensure that the gate voltage (V_{GS}) was sufficient to turn the device fully on given that the current though the sense resistor will cause the source voltage to float above ground. A series of turn-on tests were repeated for a range of applied voltages to establish if there was any voltage dependency on the apparent rise time and the magnitude of the initial step phenomenon.

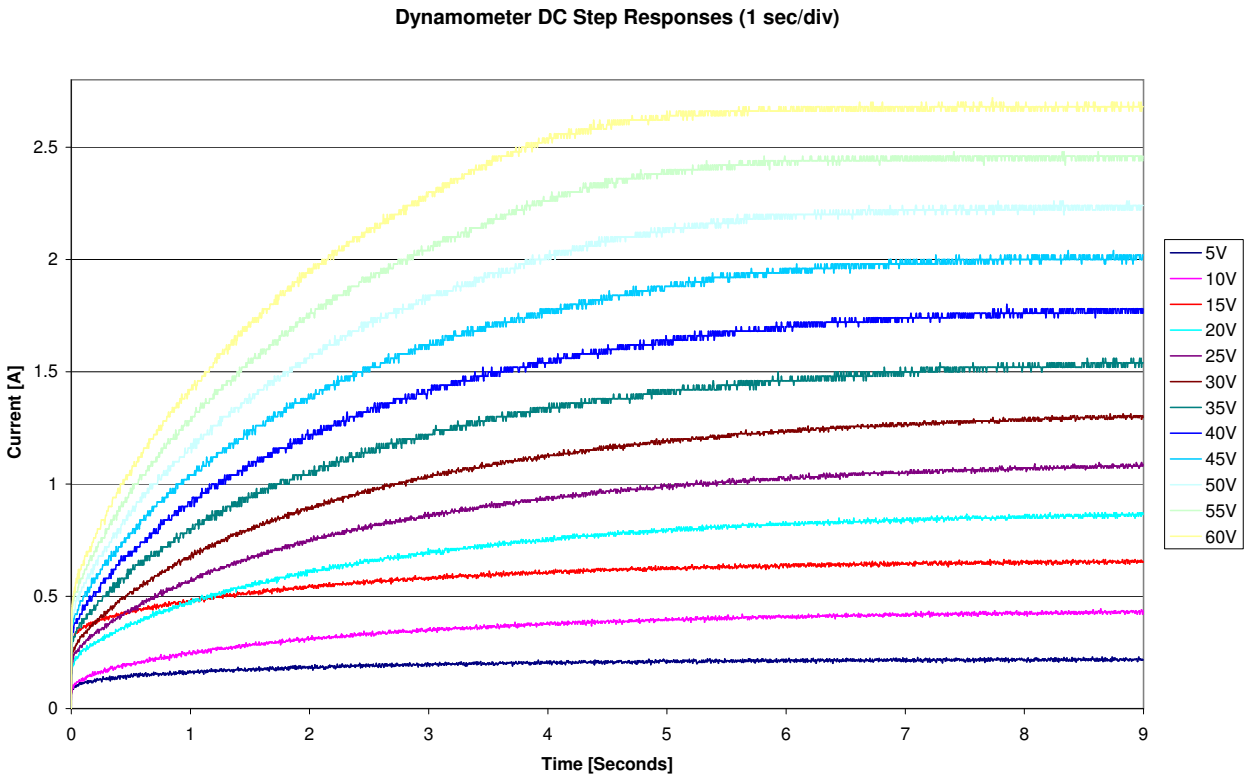

Figure 4.36: Field current step response to steady-state for a range of applied voltages

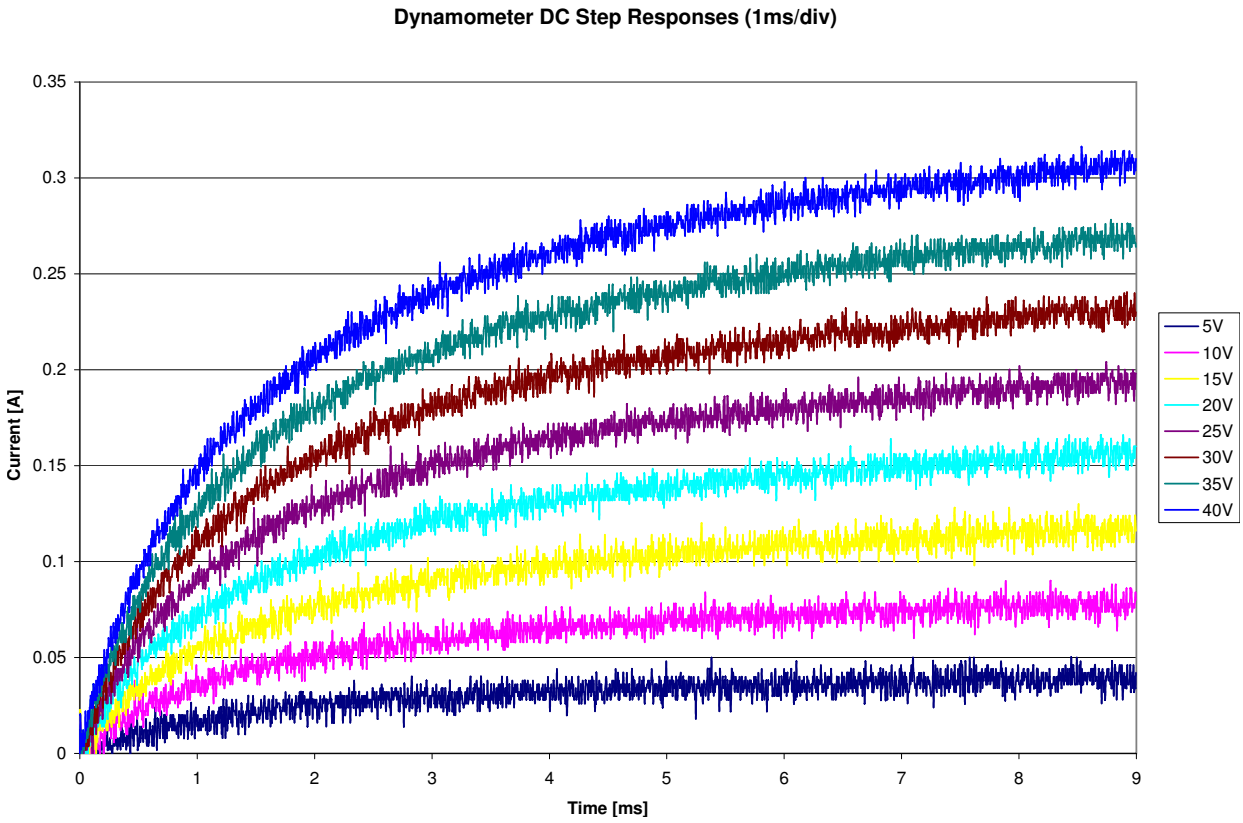

Figure 4.37: Initial rise in field current for a range of applied voltages

Figure 4.36 shows the test results for the set of 9 second duration responses, and also the small scale initial responses over a 9 ms duration are shown in Figure 4.37, both for a range of applied voltages. It can be seen that the large scale response is much the same as the previous test using a mechanical switch. This time the range of voltages tested shows that the initial step appears to be proportional to the applied voltage. The small scale responses, when plotted at a suitable scale, appear to have the same basic shape as the full responses, but without the initial step and over a much shorter time duration. To help resolve this further, a test was needed to establish how the response behaves when the applied voltage is removed. With the existing test circuit the current sense resistor is removed from the current loop when the MOSFET is turned off. This prevents the turn-off response from being logged. When the MOSFET or switch is turned off the freewheel diode forward biases and the current direction remains the same through the windings. The energy stored by the inductance of the machine maintains the current which recirculates through the diode and is dissipated by the resistance of the windings. For this recirculation current to be measured, the sense resistor needs to be moved inside the recirculation loop (4.38). As one side of the resistor still needs to remain grounded to measure the voltage drop across it, then this presents a difficulty in triggering the MOSFET gate with the correct voltage relative to the supply. This might have been resolved by using a p-channel MOSFET and switching the load from the *high* side instead of the *low* side, but instead a mechanical switch was used again to simplify the task and initial fast switched responses had already been captured.

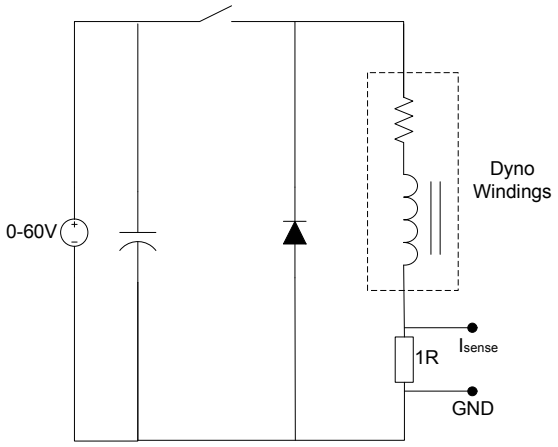

Figure 4.38: Test circuit modified to capture the turn-off response

For the turn-off test a single applied voltage of 24.5 V was used and the test was repeated with the supply being switched off after different amounts of time.

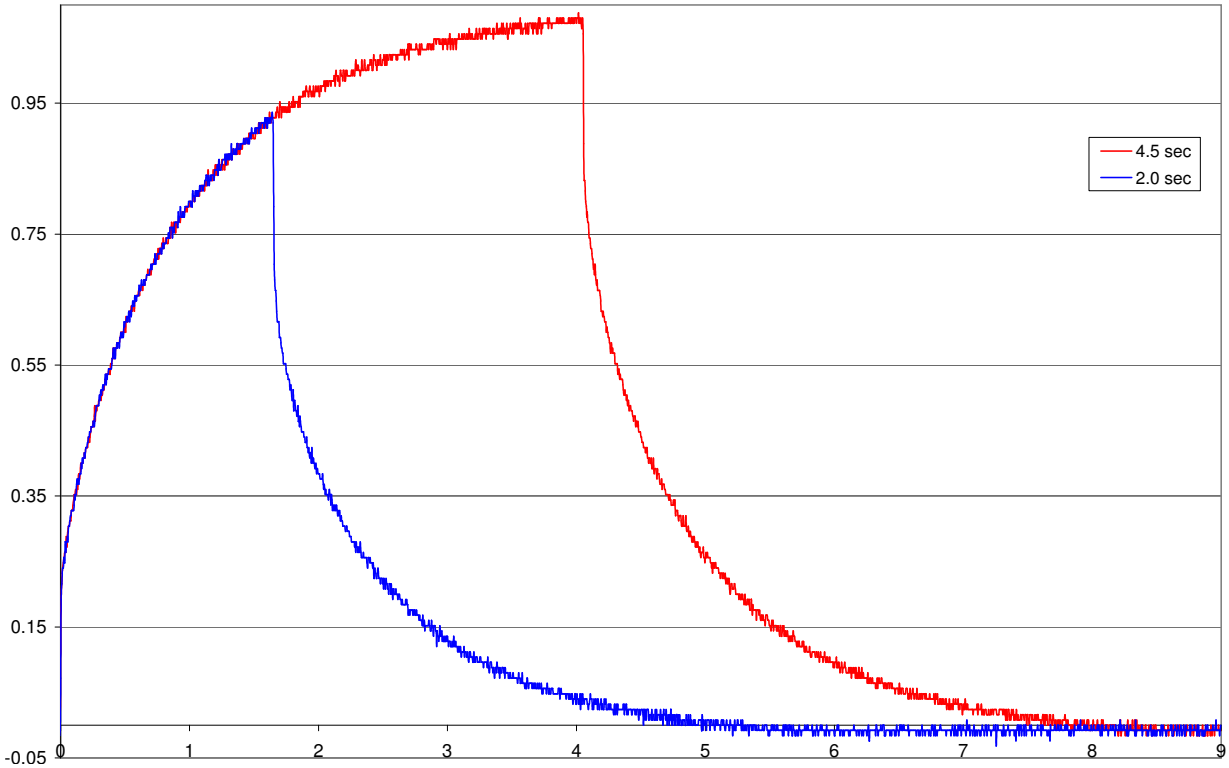

Figure 4.39: Turn-off step tests

Figure 4.39 shows that the turn-off characteristic is remarkably similar to an inverted turn-on response, both being exponential with an initial jump. From the test it is clear that there is evidence of energy storage and it confirms that the inductance is large and that there is a long time constant. The initial drop in the output current immediately after turn-off casts doubt on the effect being an *initial* magnetisation effect (remanence and coercivity on the magnetisation curve) as current is already flowing and the core is energised.

Figure 4.40 shows a selection of the transient responses plotted together in a normalised form to compare how the responses change with applied voltage. All of the responses are similar except for 5V. This may be evidence of a minimum magnetisation energy or core penetration effect. The responses for higher voltages (particularly 55 V and 60 V) appear to deviate more from an exponential shape towards steady-state and this is likley due to the onset of magnetic saturation at a particular flux level. Figure 4.41, Figure 4.42, Figure 4.43, and Figure 4.44 are plots of the natural logarithm of the normalised responses which should be a linear for an ideal exponential response. The plot sections in red have been median filtered to remove outlier noise and also show the data range used for a straight line best fit. The data in the plots shows reasonable adherance to the straight line fit, execpt for the 55 V and 60 V plots which deviate strongly after a certain unknown flux level is reached which is due saturation occuring, possibly localised somewhere in the magnetic pathway. If the coupled pair of inductors model discussed in section 3.1 is assumed, which makes use of a ficticous damper winding, we can deduce from the reciprocal of straight line fit gradient a combined time constant for the field winding and the damper winding, τ_f and τ_e respectively. The size of the initial current step multiplied by the combined time constant $(\tau_f + \tau_e)$ gives τ_f and so τ_e can be found also. Table 4.1 gives the calculated time constants for each of the five plotted responses. It also shows that estimated self inducance varies depending upon the excitation conditions. The peak estimated inductance is around 42 Henrys when the machine is fully excited but less than the point of saturation. This is estimate much lower then the 70-80 Henrys than results from using the current time constant alone. Figure 4.45 shows how the two field and damping time constants vary in proportion to each other over the limited test range. The damper current time constant appears to remain a constant proportion of the field current over about 10 V and the overall combined time constant falls slightly with applied voltage. It should also be noted that these relationships are shown with the machine at rest and may alter significantly with load torque and shaft speed.

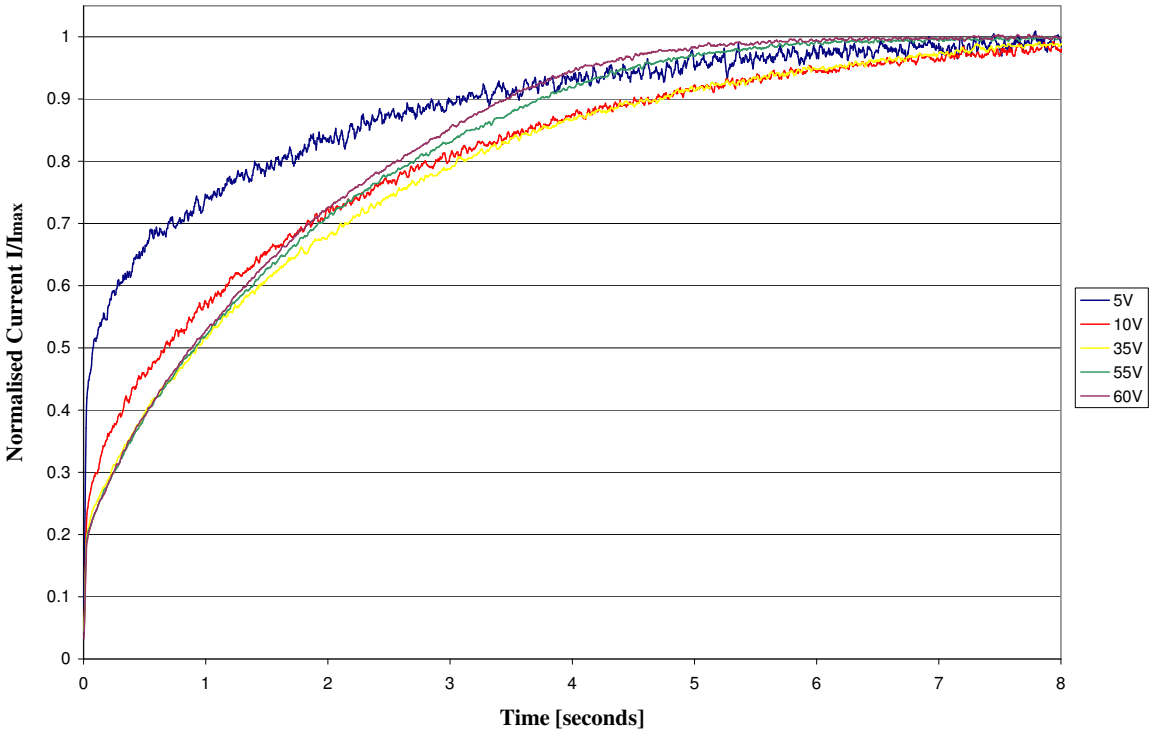

Figure 4.40: Normalised field current transients for several applied voltages

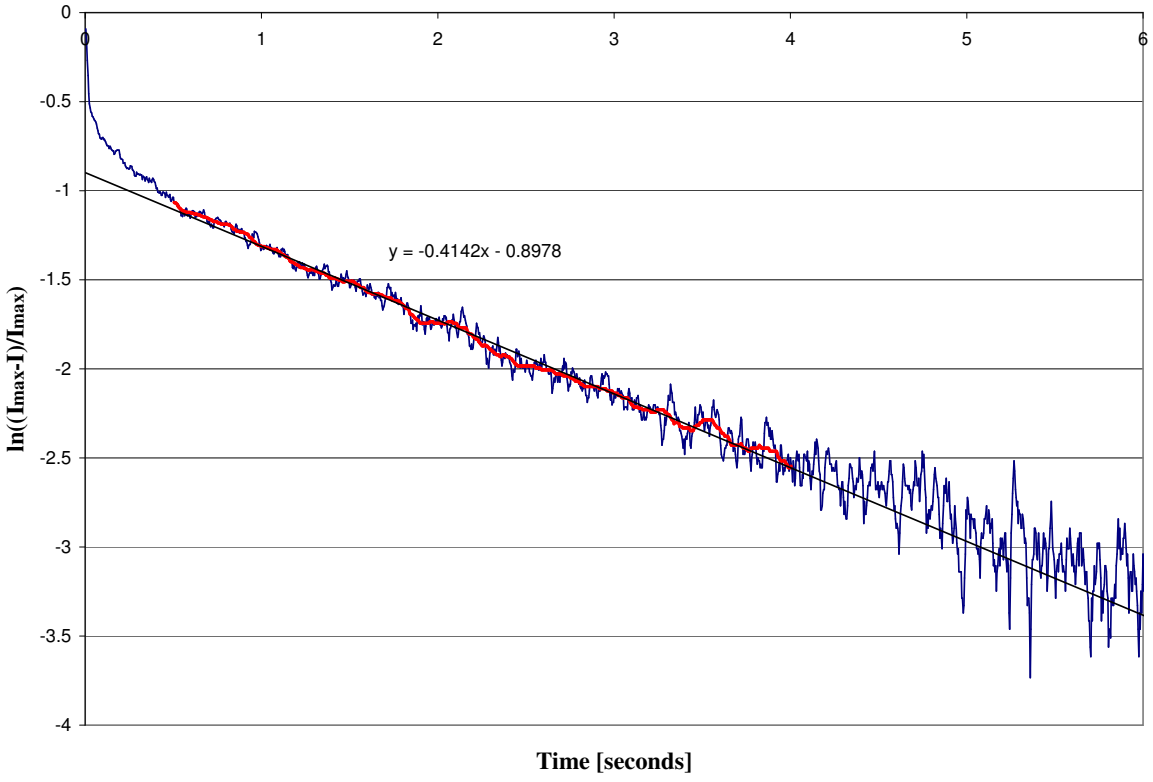

Figure 4.41: Natural logarithm of normalised field current for a 5 V transient

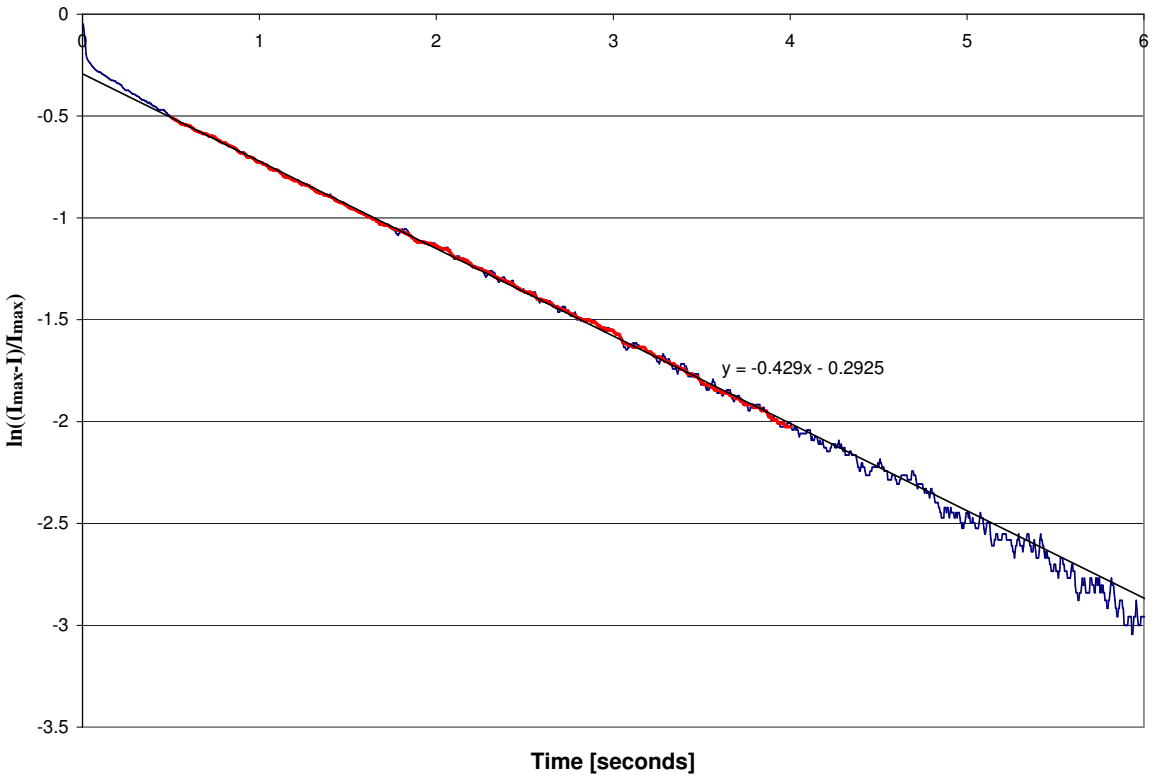

Figure 4.42: Natural logarithm of normalised field current for a 35 V transient

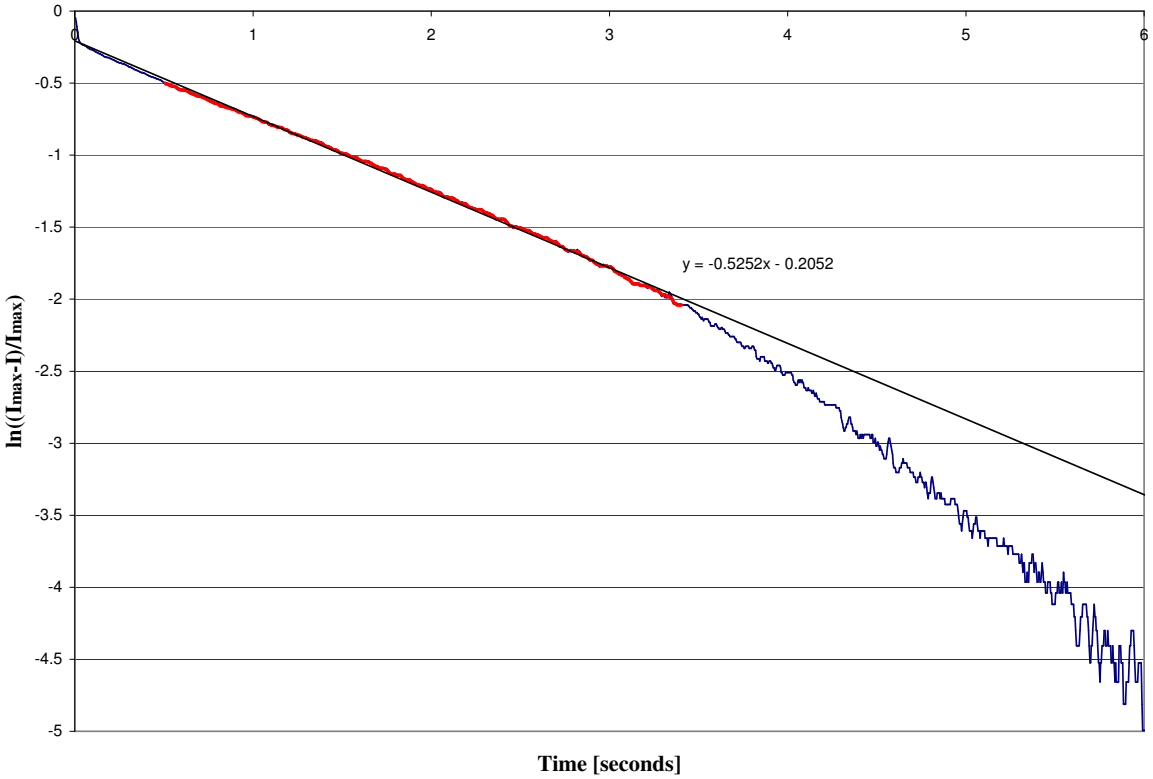

Figure 4.43: Natural logarithm of normalised field current for a 55 V transient

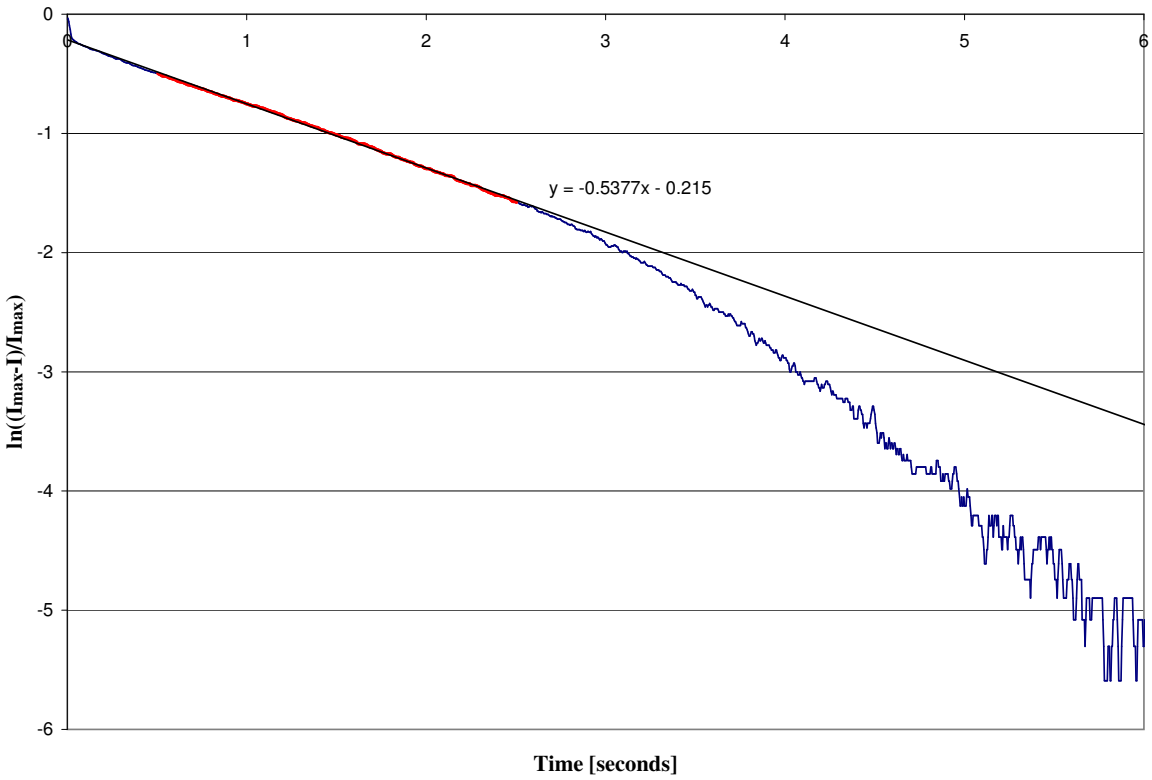

Figure 4.44: Natural logarithm of normalised field current for a 60 V transient

	Applied Voltage [V]				
	5	10	35	55	60
$\tau_f + \tau_e$	2.41	2.47	2.33	1.90	1.86
τ_e	1.33	0.62	0.47	0.38	0.37
τ_f	1.09	1.86	1.86	1.52	1.49
$\frac{\tau_e}{\tau_f}$	1.22	0.33	0.25	0.25	0.25
$L_f = R_f \tau_f$	24.5	41.9	41.9	34.7	33.2

Table 4.1: Estimated time constants and self inductance for a selection of applied voltages

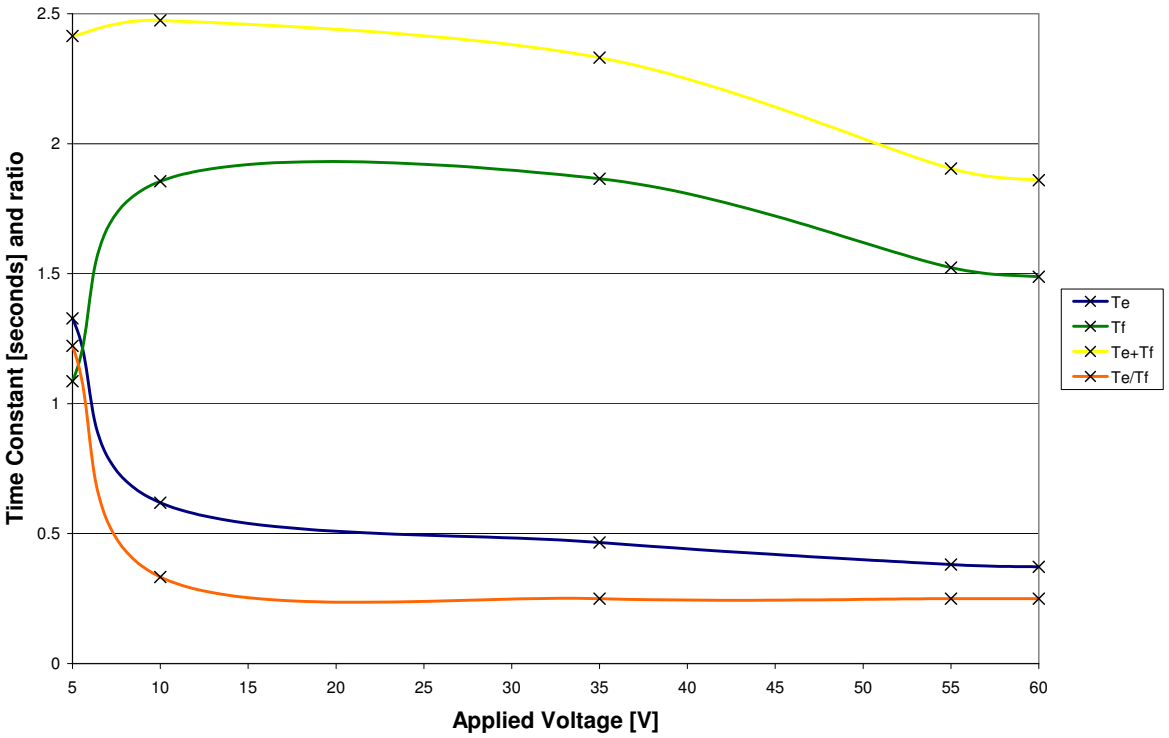

Figure 4.45: Time constant variation for several applied voltages

Figure 4.46 shows a block diagram of the tranfer function models of the field and damper windings. Normalised responses were obtained from this model and compared to the actual measured responses for the two extremes of 5 V (Figure 4.47) and 60 V (Figure 4.48) using the empirically determined time constants shown in Table 4.1. The parameter k determines the level of curvature in the initial step and was determined manually by trial-and-error. The responses show remarkable correspondence to the measured data given the uncertainty in the parameters; the 60 V showing slight diviation due to the onset of saturation. How this saturation onset might be affected by shaft rotation producing an alternating field in the loss drum is not known and would require further tests to determine.

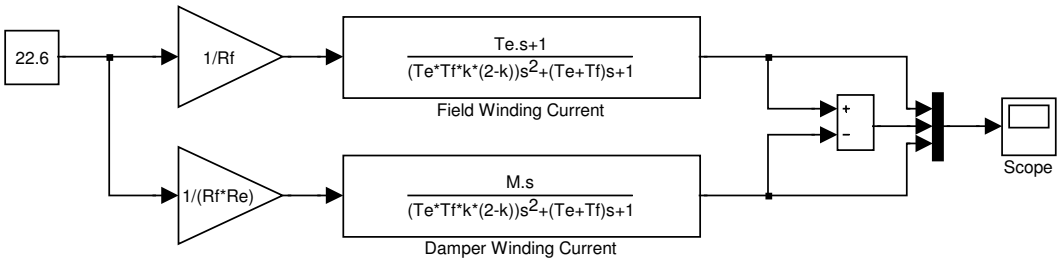

Figure 4.46: Transfer functions for applied voltage to field winding and damper winding current

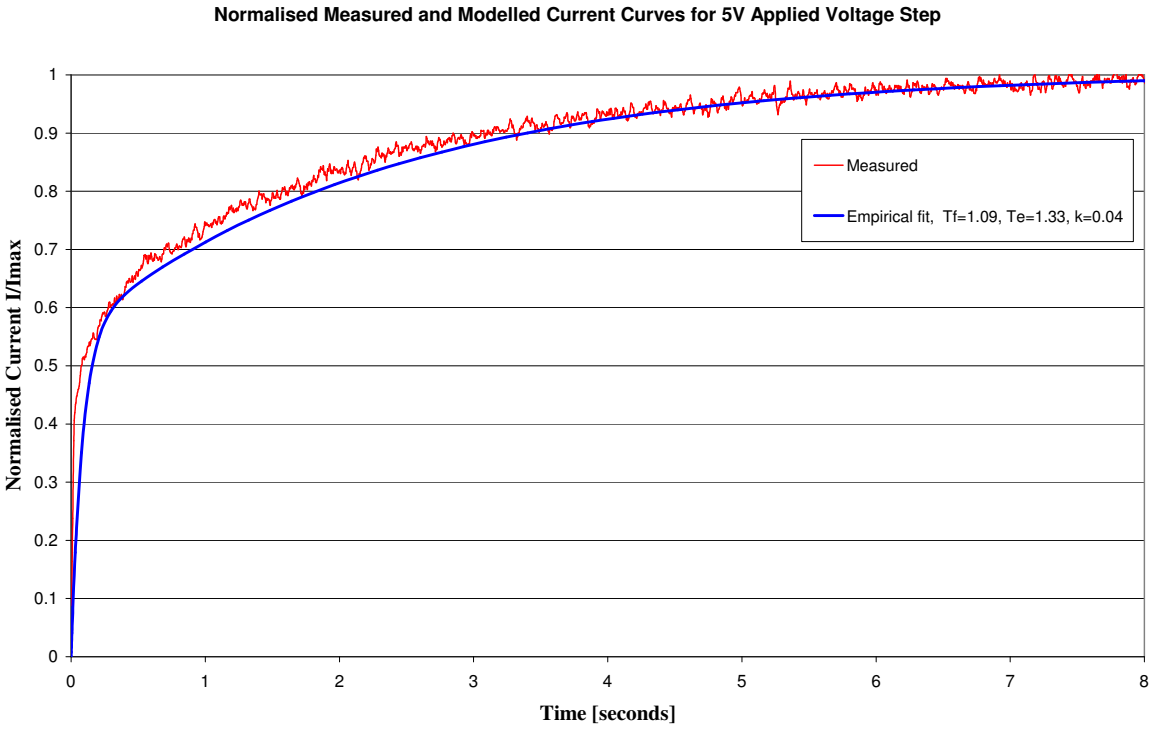

Figure 4.47: Modelled and measured normalised current responses to a 5 V applied voltage

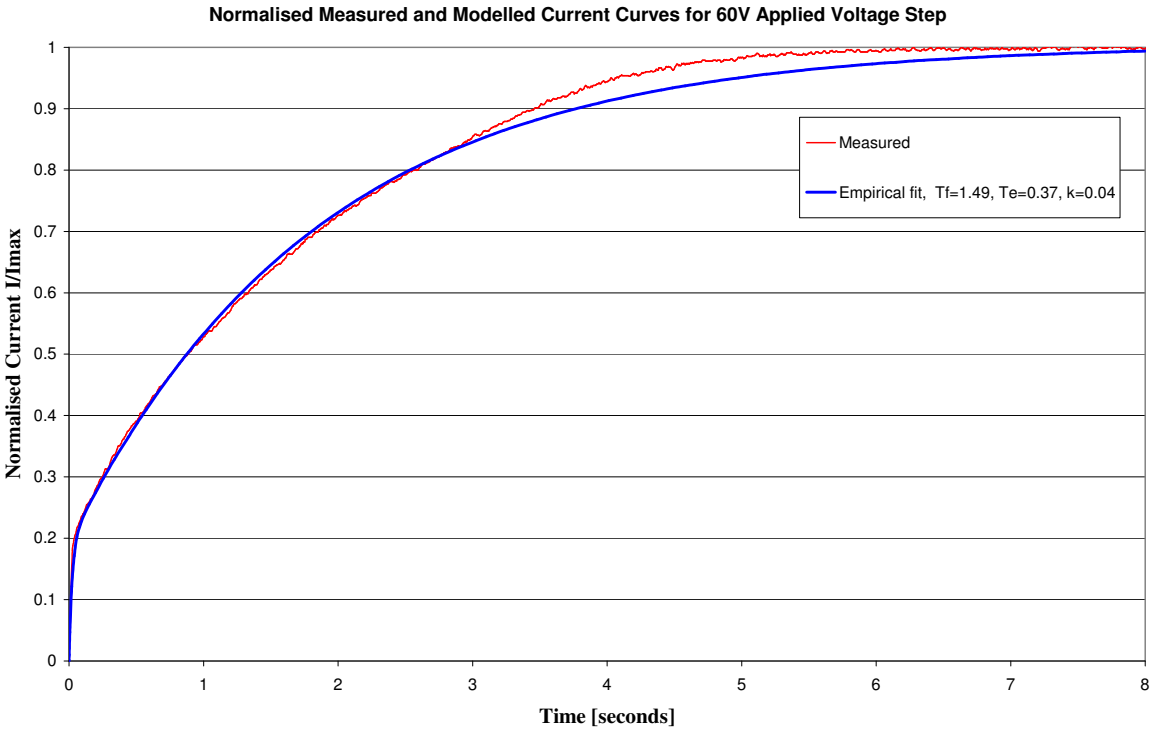

Figure 4.48: Modelled and measured normalised current responses to a 60 V applied voltage

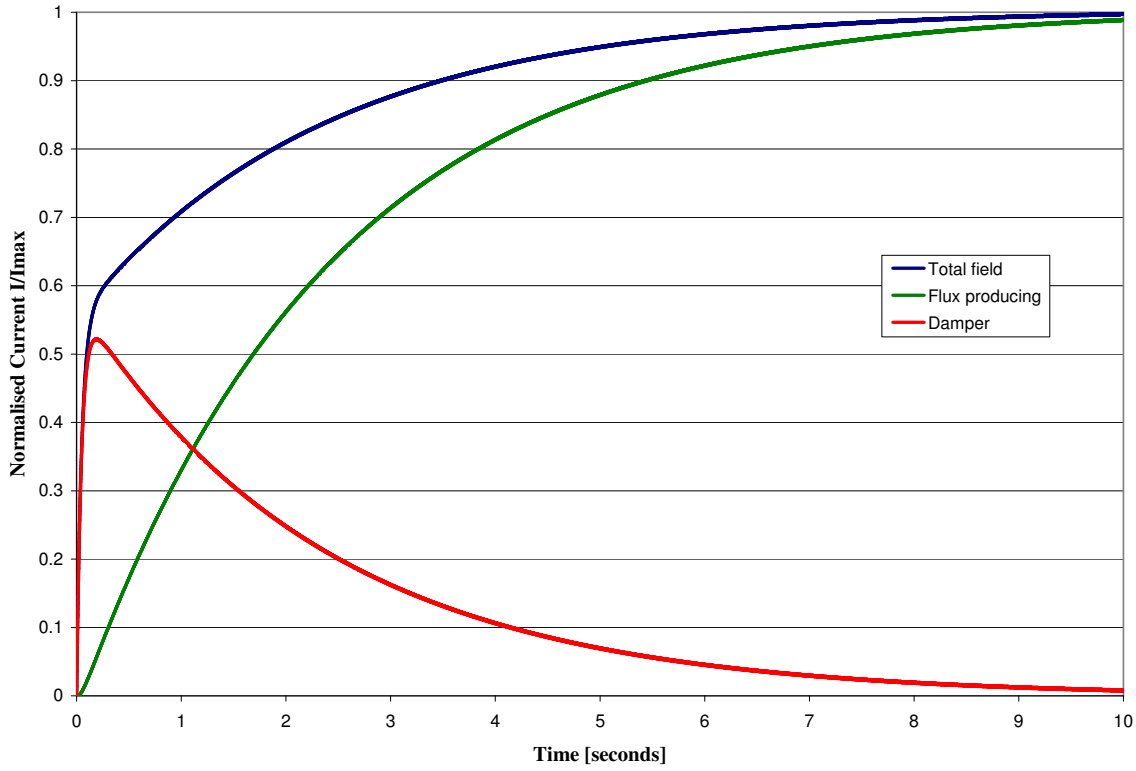

Figure 4.49: Modelled current winding responses and predicted flux current

Figure 4.49 shows the modelled 5 V (chosen as it has a high $\frac{\tau_e}{\tau_f}$ ratio which accentuates the effect) curent responses and allows the difference between them as to be seen as the green curve which can be assumed to be the flux producing current. The value of the coupling parameter M was chosen arbitarily to make the damper response magntiude initially equal to the field current as initially this should be true. It is the green flux curve (which cannot be easily validated) which reveals the nature of the flux response and will be directly proportional to torque for any given shaft speed. Being able to predict this curve would allow a current controller to be used to force the torque response to a demand value in the minimum time. The total field current is all that can be measured externally, and if a controller was to use this for feedback, taking Figure 4.49 as an example, after half a second the feedback current would indicate that around 60%, or one time constant, of the response had been achieved when in fact only around 15% of the steady-state flux rise had occured. The controller might then tend to reduce the applied voltage prematurely increasing the overall time taken to achieve the desired flux asociated with a particular steady-state current value. To obtain an optimal rise time or slew rate in the field windings the applied voltage must be be held at a maximum until the demand current is reached or approached. No controller can achieve this without knowledge of the internal electrical properties of the machine. This leads to a requirement for a model based approach such as a state-space model by which the internal *state* of the field windings can be estimated from a knowledge of the plant model. An impediment to this approach is the parameter uncertainty and variation of parameters with setpoint and operating conditions. The next section demonstrates how a

state-space approach can be adopted to provide a virtual sensor to the current controller.

4.3.7 State-space Estimation and Control of the Field Windings

As it is not possible to just directly measure the flux producing proportion of the current through the field windings, a state-space model can help to observe or estimate the current using the inductance and resistance parameters that have experimentally determined. The estimated state for the part of the field winding current which contributes to the production of flux can then be used as a virtual sensor as feedback to a controller. As there is no flux measurement for feedback the model must be asymtotically convergent with the real flux state so that plant and measurement noise do not cause the estimate to diverge with time. If this were not the case then a Kalman filter (Kalman, 1960) with respresenitive input and output noise models would be required. For the purposes of forming a state-space model the dynamometer is electrically represented here by two inductors connected electrically in parallel with no magnetic coupling. This approach was chosen before eddy-current damping was established to be the most likely cause for the shape of the current response and before the magnetically coupled inductor model shown in the previous section had been validated. However, the general approach is still valid, requiring only modification to the state equations.

Treating the inductor pairs as uncoupled, the current though each inductor element can be expressed as:

$$u(t) = L\frac{di}{dt} + iR$$

By choosing the current i as a state x and rearranging we have:

$$\dot{x} = -\frac{R}{L}x + \frac{1}{L}u(t)$$

Now choosing the first state variable to be the current through the by-pass resistor (denoted x_1) and the second state variable to be the current though the field windings (denoted x_2) expressed in matrix form leads to the following state-space matrices:

$$A = \begin{bmatrix} -\frac{R_1}{L_1} & 0 \\ 0 & -\frac{R_2}{L_2} \end{bmatrix} = \begin{bmatrix} -722.9 & 0 \\ 0 & -0.3957 \end{bmatrix}$$

$$B = \begin{bmatrix} \frac{1}{L_1} \\ \frac{1}{L_2} \end{bmatrix} = \begin{bmatrix} 6.024 \\ 0.01429 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

$$D = 0$$

To form a full order estimator (FOE) the two poles for the two states need to be placed at an arbitrarily fast (relative to the system being estimated) location:

$$P = \begin{bmatrix} -2 & -2 \end{bmatrix}$$

The Matlab function *acker* for Ackerman's method for pole placement can then be used, but the A and C matrices have to be transposed to allow matrix multiplication to take place:

$$G1 = acker(A', C', P)$$

$$G = G1' = \begin{bmatrix} -719.2993 \\ 0.0036 \end{bmatrix}$$

The new A matrix for the FOE, which we will denote F, is formed from the relationship:

$$F = A - GC = \begin{bmatrix} -3.6007 & 719.2993 \\ -0.0036 & -0.3993 \end{bmatrix}$$

With the estimator in place it is now possible to attempt control of the field winding current using state variable feedback (SVF). The vector:

$$K = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

is used to reduce the feedback states to a scaler containing only x_2. The system model which includes SVF acting on x_2 is shown in Figure 4.50. The controller needs to be able to track the set-point rather than act just as a regulator so a simple PI control topology was used to ensure any steady state error is quickly reduced as the set-point changes. A more comprehensive treatment might have included the proportional and integral action in the state-space SVF design, however the topology presented here was found to suffice. The inclusion of a saturation block allows the use of a very high proportional gain which ensures that the maximum applied voltage is used until the set point is very nearly reached, where the response becomes proportional to the error. For simplicity, this simulation model assumes that the supply is a perfectly regulatable 0-240 V DC, with no ripple.

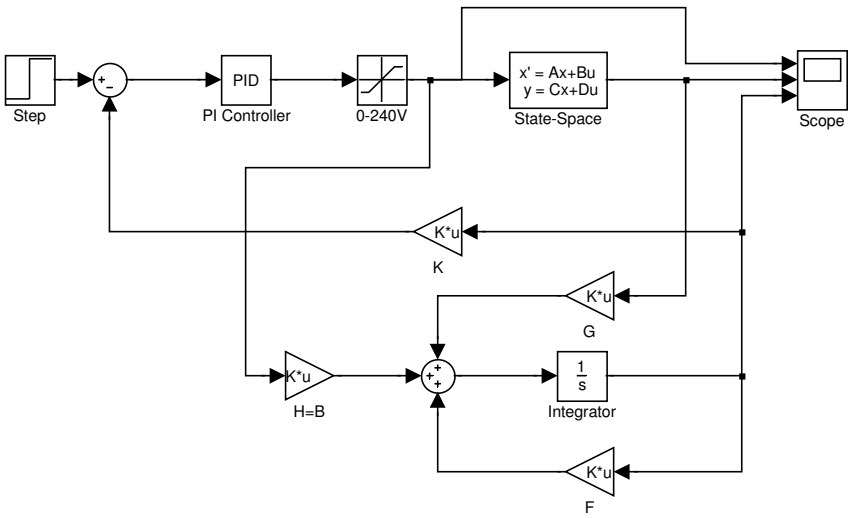

Figure 4.50: State-space model with full order estimator and state variable feedback control

The step block was used to simulate the response to a set-point change from an initial 4 A through the field windings, to a lower value of 1 A after 2 seconds. Figure 4.51 shows that the estimator accurately predicts the two internal states given that there is no additive noise or plant-estimator model mismatch. The total current momentarily exceeds the maximum rated 5 A in achieving the 4 A field winding set-point. The time for the current to fall to the reduced set-point is the order of 3.5 seconds which is considerably longer than the time it took to rise by the same amount. This is because, with the supply removed, there is nowhere for the inductively stored energy to dissipate except due to internal resistance as it recirculates through the windings. The same problem exists for the real system, as once the thyristors have been triggered they cannot be switched off again, until the current falls to zero. With a purely resistive load, this occurs at the next mains zero voltage crossing after the gate drive signal is removed, but with a highly inductive load such as this, it can take considerably longer.

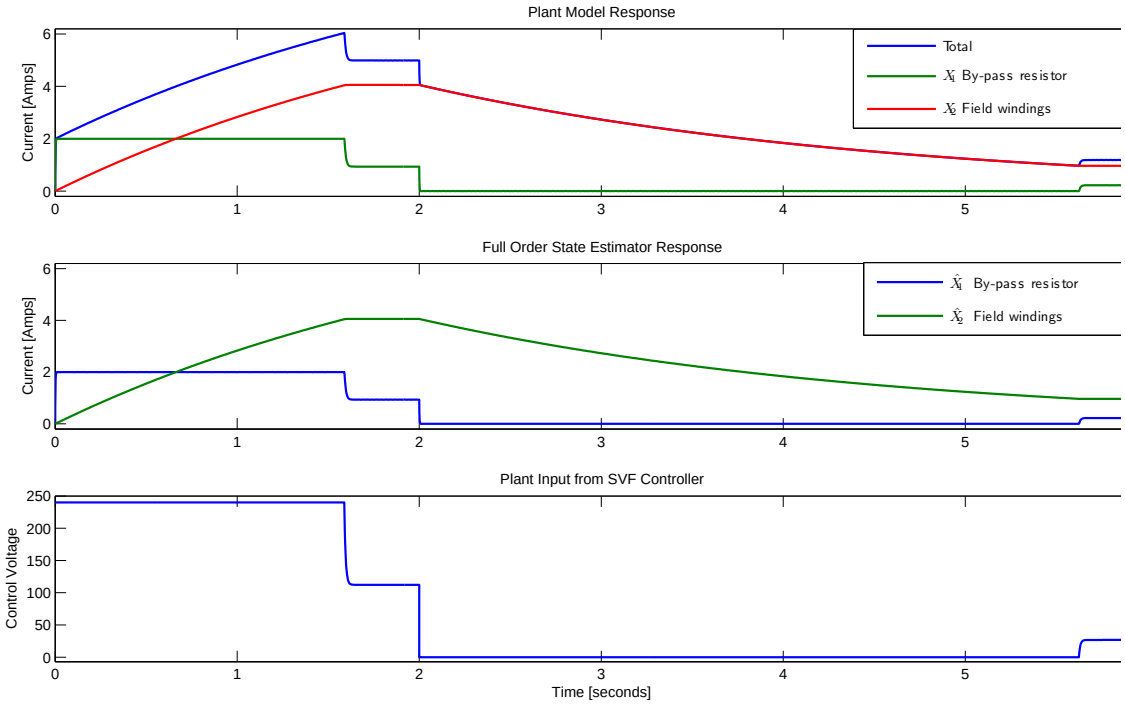

Figure 4.51: Time response plots of the estimated system to step inputs of 4 A and 1 A, at t=2 and t=4 seconds respectively

4.3.8 Discretisation and Software Implementation of Flux Current Estimator

There are two issues arising from the continuous model in the previous section. One is how it can actually be implemented to control the real plant and the other is how to reduce the relatively long plant *wind-down* time. The latter will be deferred for the moment and the former can be addressed by moving the model into the discrete domain so that the control system can eventually be implemented in software. As a first step the plant model is put into discrete form and is verified against the original continuous one. The discrete form of the state-space system can be made from the continuous one, using Matlab's *c2d* function and then the resulting matrices can be exported to the Simulink model:

```
A = [-Rbp/Lbp 0; 0 -Rdyno/Ldyno];
B = [1/Lbp; 1/Ldyno];
C = [1 1]; D = 0;
sys_con = ss(A,B,C,D);
Ts = 0.01;
```

```
sys_dis = c2d(sys_con,Ts);
[F,G,C,D,Ts1] = ssdata(sys_dis);
```

To test that the new discrete model is representative of the continuous one, we can revert back to the state-space model with a DC input to the system as shown in Figure 4.52. The continuous and discrete forms of the model are given the same input and the outputs are compared.

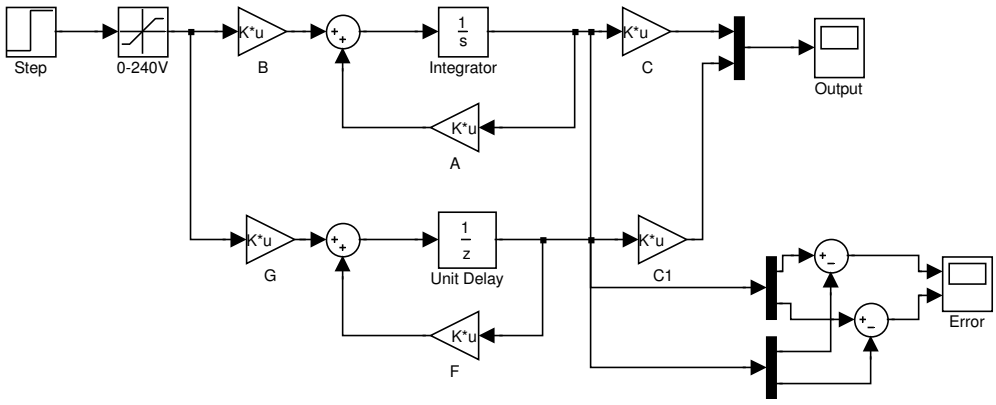

Figure 4.52: Continuous and discrete models for comparison

Figure 4.53 shows that the discrete form gives a reasonable (for control purposes) conformance to its continuous counterpart, exhibiting a 10 ms lag due to the sampling period, as expected.

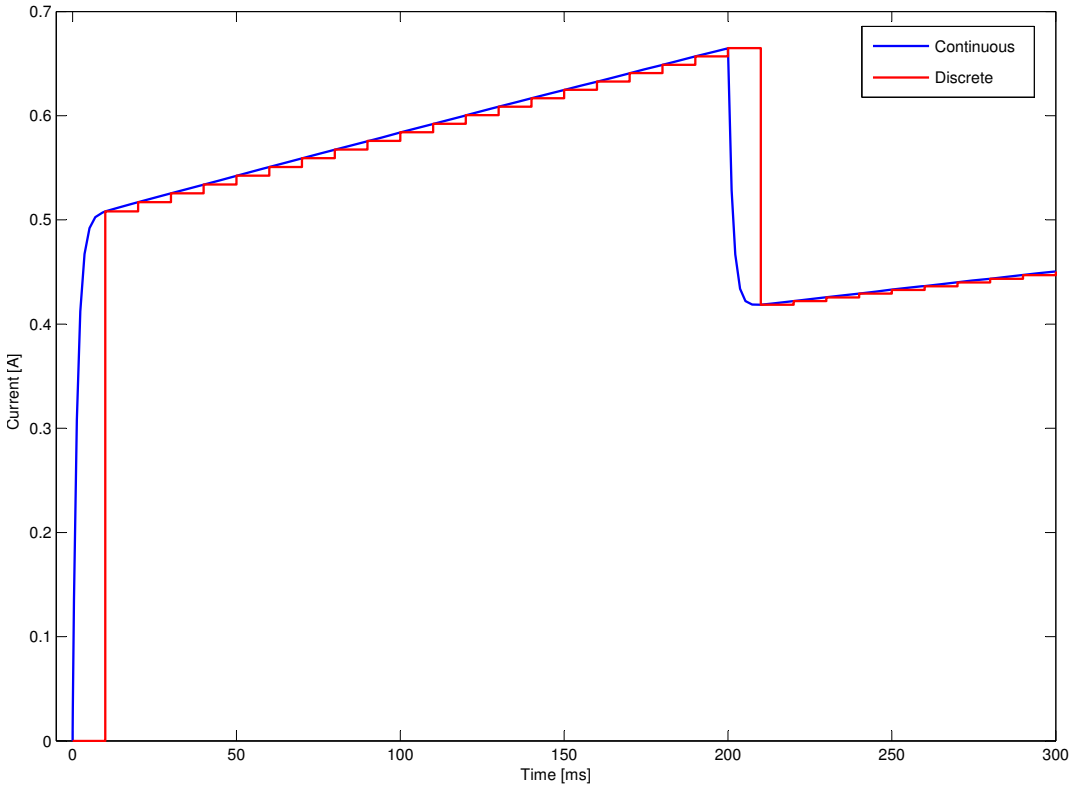

Figure 4.53: Plot showing the correspondence between continuous and discrete models

Since the discrete model gives a reasonable representation, the continuous model can be removed and a discrete Simulink solver can be used instead as there are no longer any continuous states in the model. The next step is to convert the continuous estimator into a discrete form and add this to the previous model and then perform an open loop test of the discrete estimator. It was found that the position of the poles had to be moved closer to the origin to make the estimator faster acting than was necessary for the continuous case to prevent *blow-up*. The following Matlab code was used to construct the discrete form of the estimator:

```
P = [-0.002+0.002j -0.002-0.002j];
Gc2 = acker(F', F'*C', P);
Gcd = Gc2';
Fp = F-Gcd*C;
```

The resulting block diagram is shown in Figure 4.54. Running the model reveals that there is zero prediction error as the system is fully discrete and the is no additive input, output, or measurement noise. For the real system all of these noise sources will exist as well as some inevitable plant-model mismatch. The question which arises is whether or not the estimator will be prone to diverge from the real system. If the estimator is found to diverge, then a Kalman filter approach will be needed, however as the additional effort to construct a Kalman filter with input and output noise models is significant and so is to be avoided here if at all possible. As the hardware required would be the same whether or not a Kalman filter design is employed, we can proceed with the current approach.

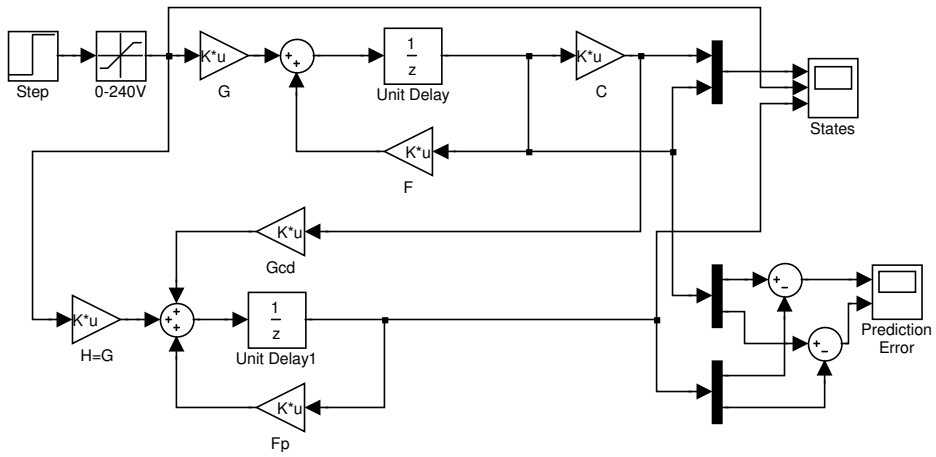

Figure 4.54: Discrete plant and estimator block diagram

With the estimator in discrete form, it is now possible to write it in the form of program code. A Simulink S-function block which can call external C code was used and the estimator was written into a callback function which is called for each time-step of the model. For comparison, the original block form of the estimator was left in place, with a

switch added to manually change between the two. In addition, a discrete PID controller block was added to assess the controlability of the system and provide a wider stimulus for the estimator than the simple step change inputs used previously with the discrete system. The estimated field current is fed back to the controller, as was done with the continuous case. Figure 4.55 shows the resulting block diagram.

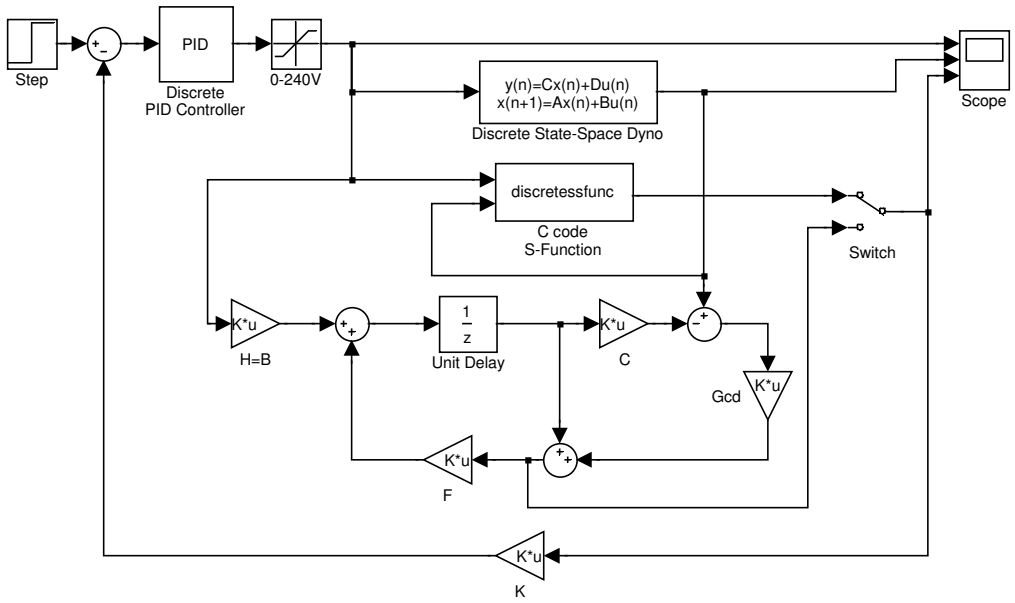

Figure 4.55: Discrete plant and controller with C coded estimator block diagram

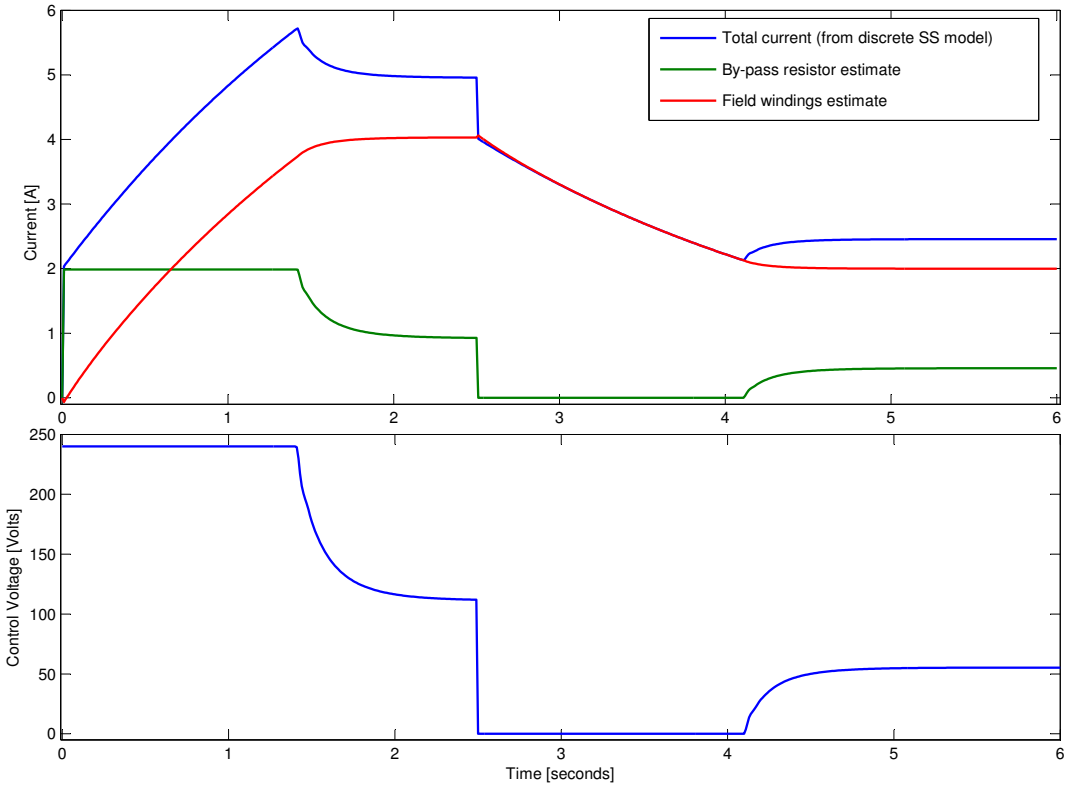

Figure 4.56: Plot of discrete controlled system response to step demand changes of 4 A (t=0 seconds), and 2 A (t=2.5 seconds)

Figure 4.56 shows the response of the system to an initial input demand of 4 A (the maximum allowable on the real system), falling to 2 A after 2.5 seconds. The estimated field current shows agreement with the model-measured current for the period of time where no control voltage is applied, as expected. The system reaches the 4 A set-point in approximately 1.5 seconds as with the continuous case where state feedback was used. The above model is a simplified representation of the real system, and does not include the details of the phase angle control, or the dynamics of the instrumentation. As the discrete estimator and controller models have been shown to perform as well as their continuous model counterparts, it is appropriate at this point to transfer them for testing to the full system model. In addition, a workaround for the slow current fall problem was tested. At the time that the applied voltage is removed from the dynamometer there will be a nominal amount of current flowing through the inductive part of the windings. This current continues to flow when the driving voltage is removed due to the energy stored in the inductor's field. Provided there is still a closed current path, then initially, the same amount of current will flow. As the current recirculates, energy is dissipated due to the resistance in the current path and so the amount of current will fall with time until there is no stored energy left, or the driving voltage is re-instated. The rate at which this current falls will depend upon how much resistance is in the recirculation path. A higher resistance will yield a faster fall time which is desirable from a control perspective an is the objective here. However, including more resistance in the normal current path (in series with it) will serve to impede the current rise when voltage is applied, which is undesirable. If there is no way to reverse the polarity of the driving voltage (as is the case with the thyristor half-bridge topology) and if the same system is to be made to have a fast rise time and a fast fall time, then the effective in-series resistance needs to be changed on-demand. If an external resistance was connected in parallel with the dynamometer (as in the original controller), and if the thyristor bridge was somehow forced to turn off, this resistor would become part of the current recirculation path during the turn-off period and hence reduce the wind-down time. This could be achieved by including a switch into the circuit so that when the control demand is zero the current can be stopped from flowing through the bridge which could be achieved with a gate-turn-off (GTO) thyristor, but not a convention thyristor. The choice of resistor is a compromise between the power it will dissipate under normal conditions when voltage is applied, and the voltage developed across it when the current through the bridge is stopped. In practise the power dissipation is too high for a manageable voltage that does not exceed the rating of the existing components. In simulation very large voltage spikes occur at switching transitions which indicate high-stress conditions for the thyristors and the switch. The exact nature of these spikes will be sensitive to the real properties of the components (thyristors, switch, snubbers) and in simulation will be sensitive to modelling errors.

It is clear that a different topology is needed. By moving the resistor in series (instead of parallel) with the dynamometer, then the current won't have to be stopped from flowing though the thyristor bridge to cause the resistor to be used for recirculation which removes a source of switching stress. This approach leaves the problem of a significantly slower current rise time. If a resistor is only connected during current decay conditions, by employing a switch to by-pass the resistance during normal conditions, then the best of both states can be achieved. Under normal applied voltage conditions with the switch closed (on) the only change to the effective circuit would be the on-state resistance of the switch which can be neglected as it is will be very small compared to that of the windings. When the switch opens (off), current can continue to flow through the thyristor bridge, but it will also have to pass through an external resistor and so the stored energy will be dissipated much faster. The limiting factors are still the maximum power that a given resistor can dissipate, and the voltage developed across it, but the resistor will have a reduced duty (over the parallel approach), dissipating power only under current fall conditions. The properties of the system dictate that the amount of energy that can be dissipated is restricted by the current rise time. For example, if it takes a minimum of 2 seconds for the current to be ramped to the maximum of 4 A and a further 2 seconds for it to fall back to zero, then the sequence can only be repeated once every 4 seconds. By *current fall* we are referring to a *wind-down* condition which the controller demand has fallen to zero (effectively saturated) as a proportional reduction in demand has not reduced the current sufficiently, possibly due to a large change in set-point and not due to the normal perturbations or controller correction about a set point. This will be a function of the controller design and gain. The duty of the resistor would increase for a proportional controller with a high gain as it would be more likely to reach a zero demand than one with a lower gain.

Figure 4.57 shows the block diagram for the modelled dynamometer system with modifications for a discrete controller. A discrete PID block has been added in place of the continuous one. Also the C coded discrete estimator is used. A test for a zero demand condition has been added so that the switch controlling an external shunt resistor can be activated and is included at this level as the signal is also used with the estimator to change the applied voltage which is used for the estimation. When a zero demand condition is detected then the estimated field current is used to calculate the voltage developed across the combined resistance of the external shunt resistor and the by-pass resistor internal to the dynamometer. This voltage is then fed back as the voltage input to the estimator. During this phase there is an increased risk of estimator divergence as no measured data are being used to update the estimate. However, the system is not subject to significant external disturbance when the supply is disconnected so only plant/model mismatch related divergence should occur. If any estimator/plant mismatch has occurred during an open-loop phase, then this should quickly re-converge when the driving voltage is re-applied. A demand voltage to phase angle conversion block has been added

for linearisation as the non-linearity was found to cause a controller gain sensitivity (discussed later). A monitoring subsystem (Figure 4.58) have been added so that the effectiveness of the estimator can be observed without cluttering the top-level model. The phase angle controller and dynamometer subsystem (Figure 4.59) has also been modified to include the shunt resistor. Some additional instrumentation has also been added and directed to a current monitoring subsystem (Figure 4.60).

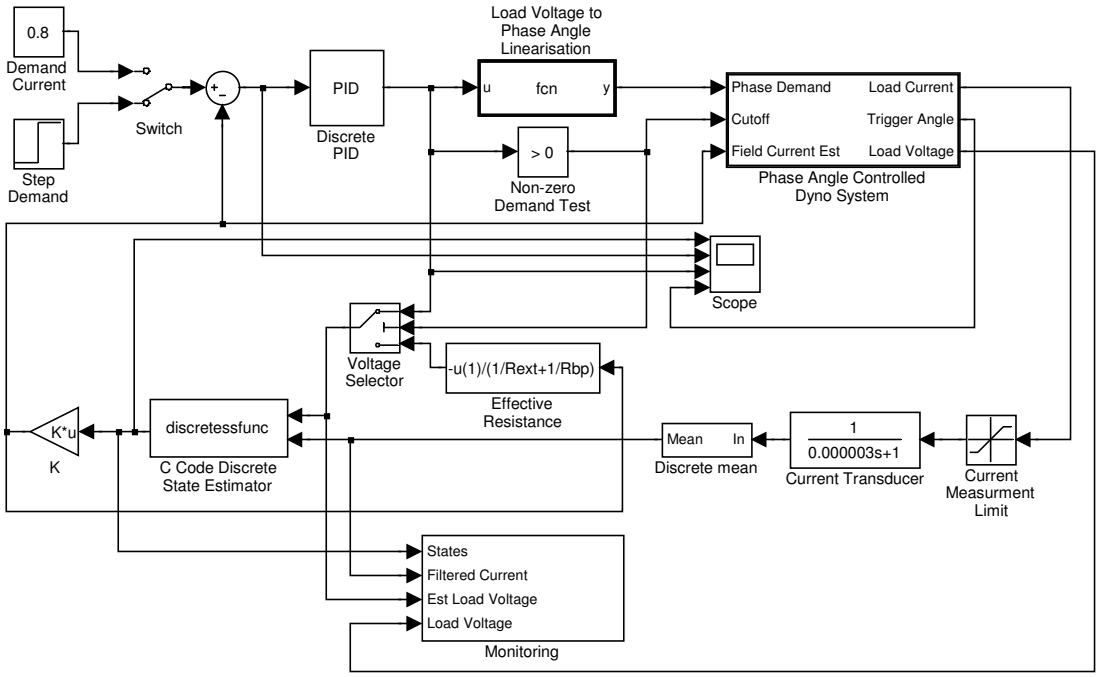

Figure 4.57: Block diagram of the full dynamometer system model with modifications for discrete controller and state estimator

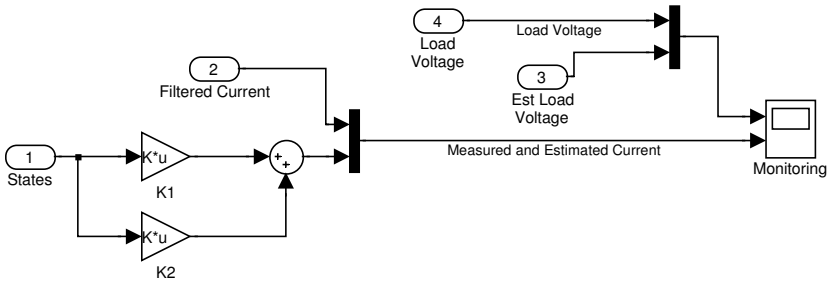

Figure 4.58: Monitoring subsystem block diagram

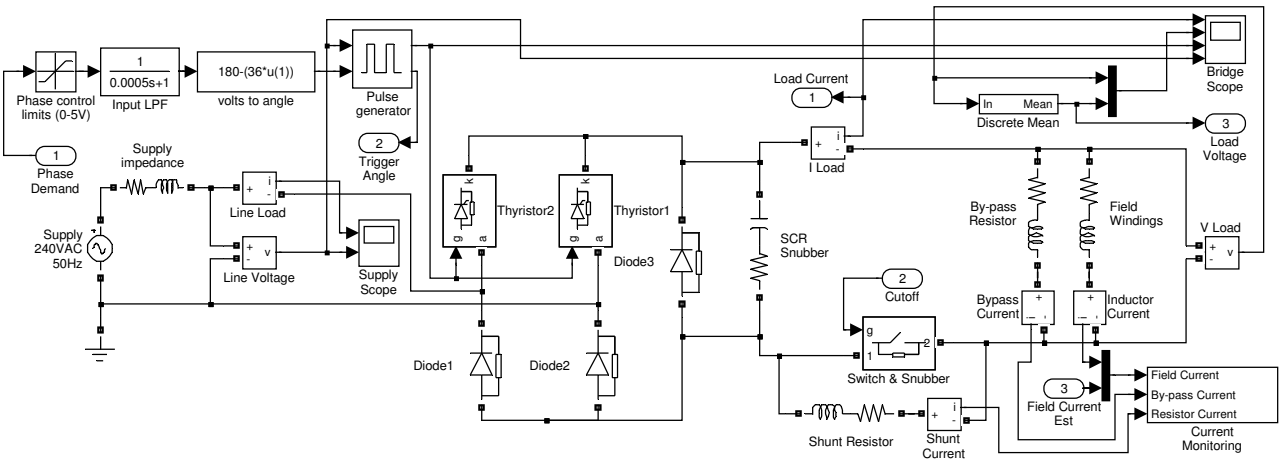

Figure 4.59: Dynamometer and phase angle controller subsystem block diagram with shunt resistor modification

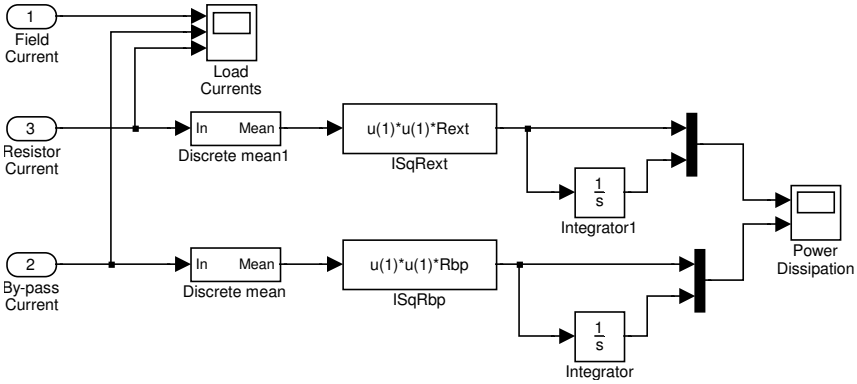

Figure 4.60: Current monitoring subsystem block diagram

Tuning of the system is a compromise between achieving the fastest possible rise time to full current (high gain) and keeping perturbations for low current set-points and small set-point changes to an acceptable level. It became apparent during tuning attempts that the non-linearity caused by using phase angle as a control variable was affecting performance. To remove this affect a block was added to convert a demand voltage directly into a phase angle based upon the integrated area under a sine curve. This is an open loop conversion as the output voltage is not directly measured and the mains waveform will not be perfectly sinusoidal due to harmonic distortions caused by the local affect of the phase angle controlled loading and other remote loads sharing the same supply phase. Figure 4.61 shows the response of the system to step changes in control demand. Initially a target of 4 A is set, reducing to 1 A after 2.5 seconds. The first set-point is achieved in approximately 1.8 seconds. The second is achieved in around 2 seconds with a small steady-state error and slight perturbation about the set-point. The estimator tracks the actual system model well including during the open-loop wind-down phase which occurs after 2.5 seconds when the control duty falls to zero and the shunt resistor is switched in. At this time the voltage across the load inverts to below -200 V which is determined by the combined resistance of the

dynamometer windings and the external shunt resistor, and the current direction reverses through the dynamometer's internal by-pass resistor.

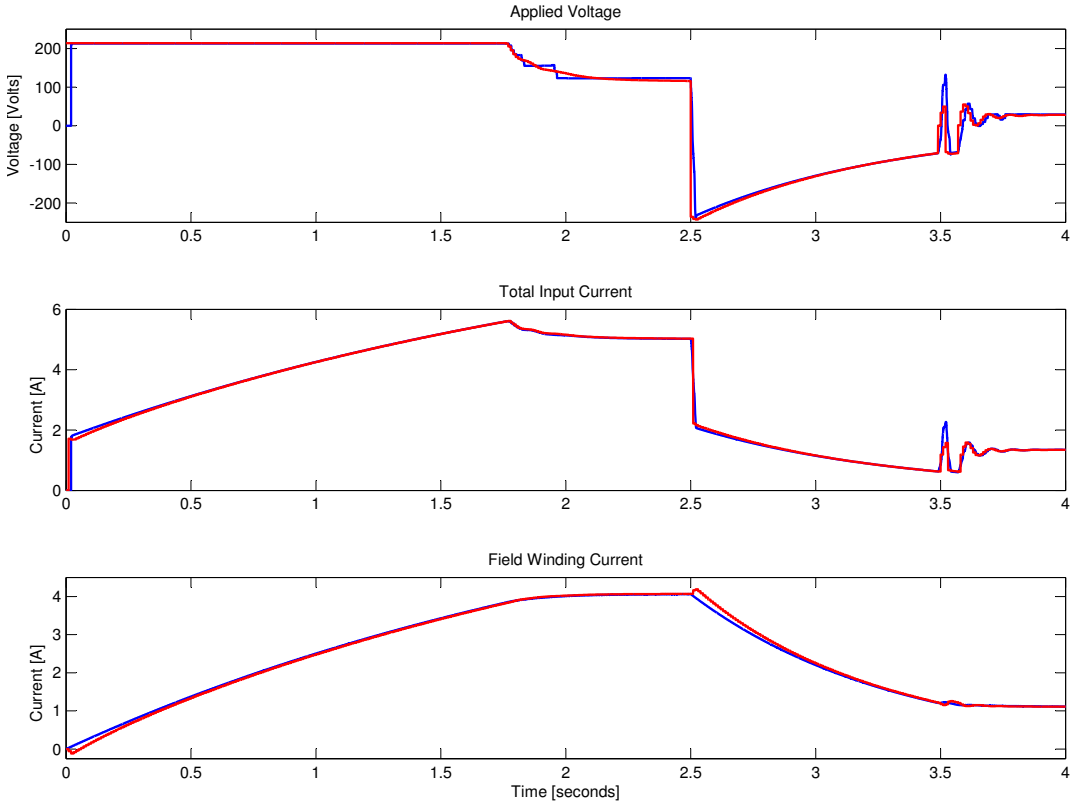

Figure 4.61: Plot of full model and estimator response to control set-point step changes for comparison. (blue represents the model output and red the estimator)

A value of 112 Ω for the external resistor was selected as to limit the peak transient voltage (<600 V) that can occur when switching at the maximum inductor current of 4 A. Figure 4.62 shows that although the voltage developed is around -220 V steady state, it spikes at -550 V during an initial transient which lasts a few milliseconds. The actual voltage developed will be dependent on the switch characteristics which will be subject to modelling errors. An *ideal* switch was used in the model to reduce computational overhead, but a varistor or tranzorb device can be used on the real plant to protect components from over-voltage damage. The chosen value of resistor also provides a current fall time which is comparable to the rise time which should help to construct an outer torque control loop by reducing gain selection sensitivity. The choice of resistor has so far neglected the overall and peak energy dissipation in the resistor. To address this the worst case single event of reducing from 4 A to zero is considered. Figure 4.63 shows a plot (using the current monitoring subsystem block) of the power dissipated when the current falls from 4 to 0 A. After an initial transient peak, the power falls exponentially from 450 Watts to zero over approximately 2.5 seconds. In practise, this condition could not be repeated for another 2 seconds (for the current to rise again), so with adequate heat sinking the temperature of the resistor can be maintained within its design limit. The charge curve is obtained by integrating to find the area under the power

curve and gives an indication of the total energy that could be dissipated in a single full-current *wind-down* event, and is in the order of 170 Joules.

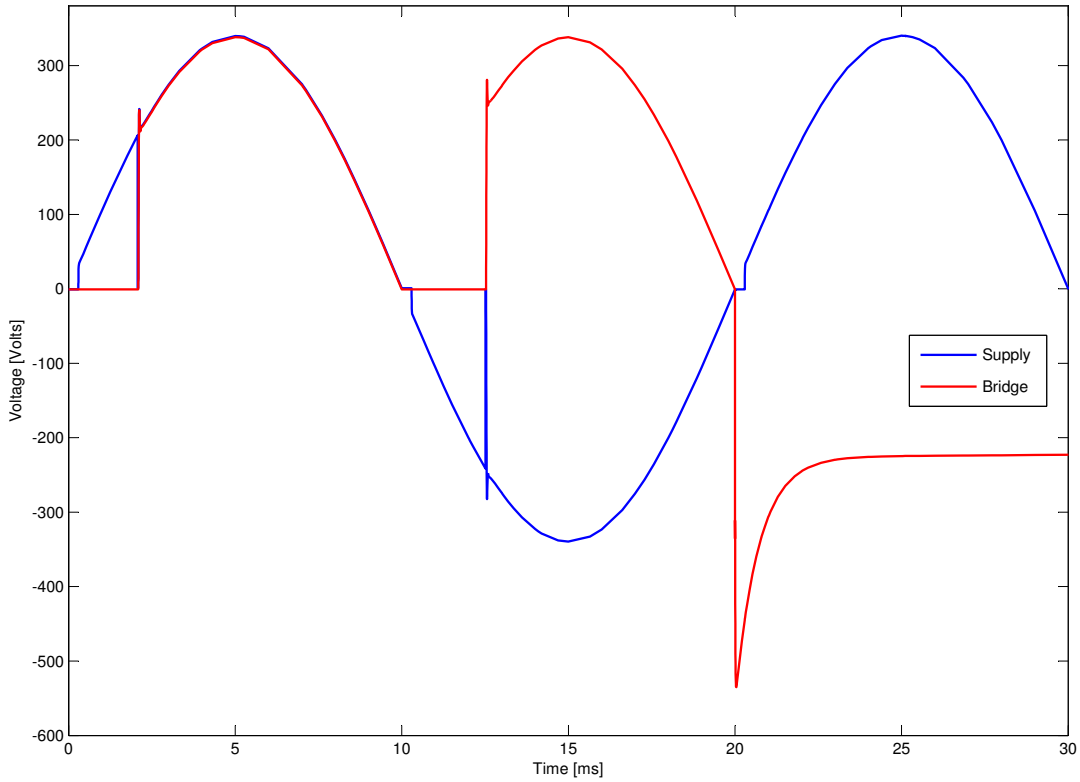

Figure 4.62: SCR bridge voltage spike occurring when a resistor is switched into the circuit at the full current rating

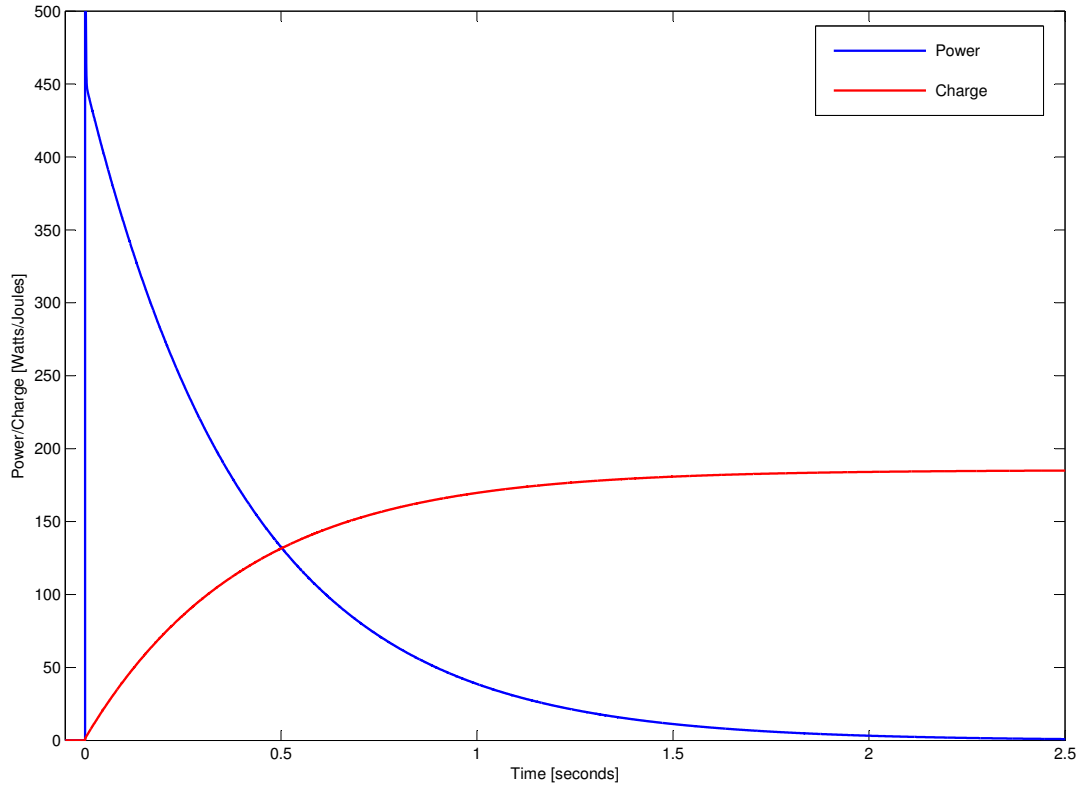

Figure 4.63: Plot of the modelled power dissipation and integrated charge in the external shunt resistor from maximum (4 A) to zero current

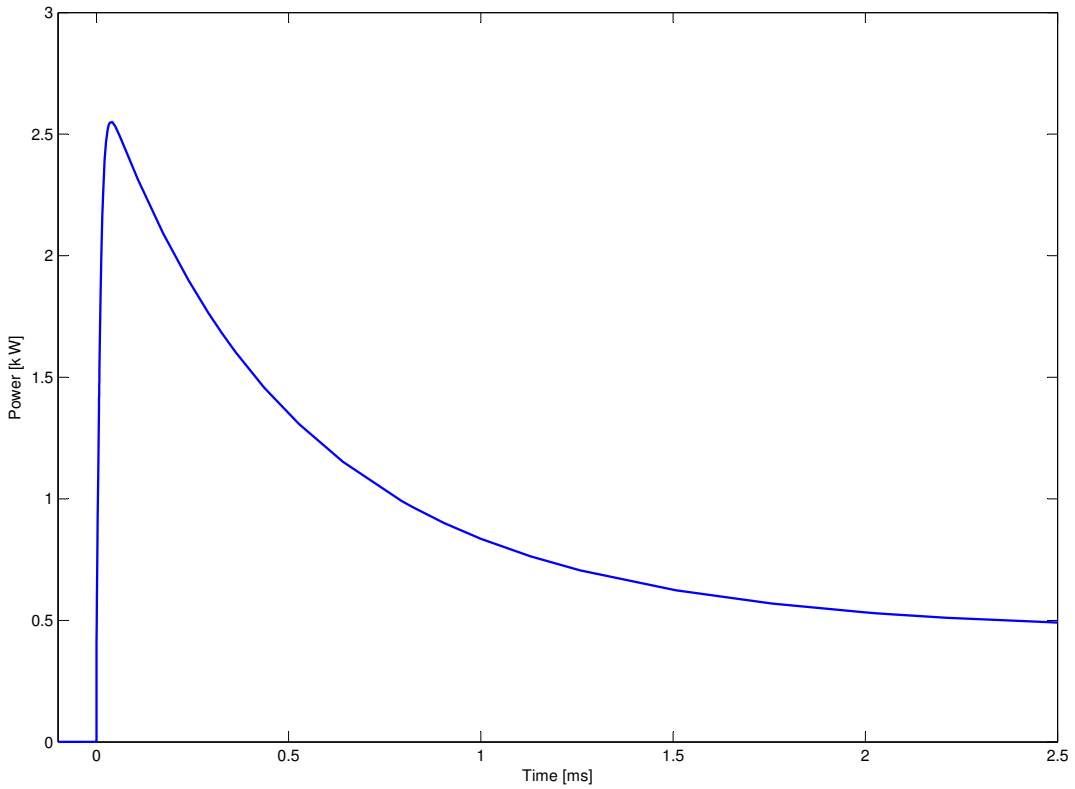

Figure 4.64: Plot of the modelled instantaneous peak power dissipation in the shunt resistor

Regardless of what heat sink is used, the resistor's peak power rating must be able to support the instantaneous peak power. Figure 4.64 shows in more detail the transient power spike of Figure 4.63 which reaches 2.5 kW.

4.3.9 Standstill Testing of the Digital Current Controller

The field current controller was reimplemented using software as part of the second phase of work. A software PID loop was written and run initially at the mains half-cycle frequency of 100 Hz. A key difference from the analogue controller is the linearisation of the control output to the Semikron phase angle controller. This is done by assuming the supply voltage is perfectly sinusoidal and the area under the half-cycle curve from the trigger angle onwards is equivalent to the mean applied voltage. By inverting this relationship the trigger angle for a desired mean voltage can be determined. The trigger angle can then be scaled to a 0-5 V 12-bit DAC output voltage, taking account for any dead-band offset which the Semikron unit does not respond. This step alone was found to improve the current controller performance greatly over the previous analogue design as the gain is not compromised by a varying sensitivity to control output. The controller's steady-state performance was found to improve further when the control period was increased to 20 ms to cover the complete mains cycle as at 10 ms the controller was observed to fluctuate at each half-cycle.

Figure 4.65 shows the current response to a full range step change in demand, taking in the order of 500 ms to reach 5 A. This is a vast improvement over the analogue controller which was observed to take in the order of seconds to stabilise to a constant current. The current response may be close to optimal as the high gain used holds the current rise rate at its maximum until very close to the vicinity of the set point, but the unmeasured flux response will not be optimal since when the demand current is achieved, the flux will still be rising, and the rate at which is does will be reduced when the controller reduces the applied voltage to maintain the current set point. Figure 4.66 shows how the current response behaves for smaller incremental increases in current demand. A 20-40% current overshoot can be seen, although the flux response will not overshoot so it will not be detrimental to using it for torque or speed control. To determine how much the instantaneous current is representative of the flux level, several tests were performed using the the setup. Figure 4.67 shows the result of holding the output at full duty for ten mains cycles or 200 ms, which is the time taken to reach 3 A, then adjusting the trigger angle to a range of lower equivalent voltages. When its voltage was lowered to around 32 V the output stayed much the same at close to 1 A. This shows that although 3 A was being measured, only there was equivalent to less than 1 A steady-state of current producing flux. Figure 4.68 shows the same test in closer detail and demonstrates that the current rise is fairly repeatable as most of the points lie close or on top of each other for the six tests even though the controller output is not syncronised to the mains. The test shown in Figure 4.69 holds the maximum duty until a selected demand current is reached. It then drops the applied voltage to a predetermined voltage which will maintain the same current under steady-state conditions. This gives an indication of how much less the flux producing current is and how much longer it could take to reach its steady-state value under current controlled conditions. Figure 4.70 shows current over and under shoot for a variety of increasing and decreasing step changes in current demand. Figure 4.71 shows that the current decay from slightly over 1 A to zero. This part of the range takes considerably longer than at higher currents due to the fact that as the recirculation current falls so does the power dissipation due to resistive I^2R copper losses in the field windings. It also shows that not only does eddy-current damping impede current rise, it also slows the current fall rate due to rate of change of flux. If the engine is approaching a stall condition, this makes it difficult to quickly remove the torque load. Load dump resistors were fitted for activation under software control, but were not used for these tests. The intention is that they can be switched into the circuit by opening the bypass switch when a sufficiently large drop in demand occurs at the lower end of the current range.

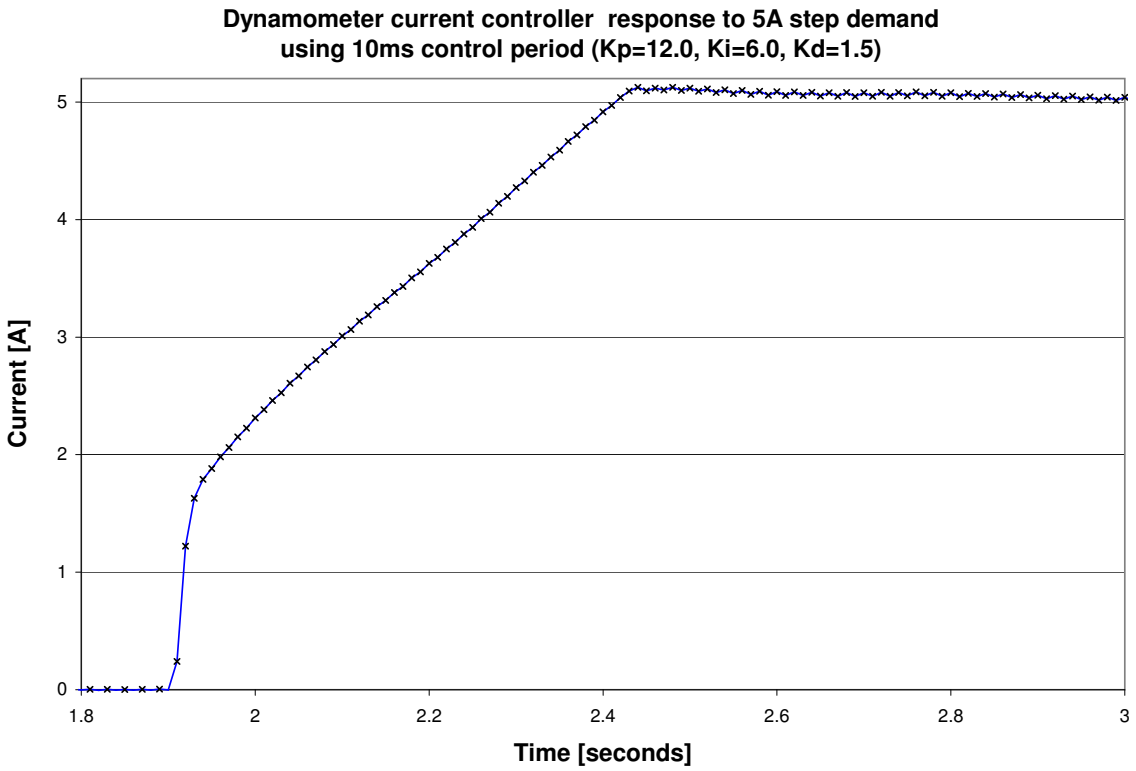

Figure 4.65: Step response to a full scale 5 A demand change

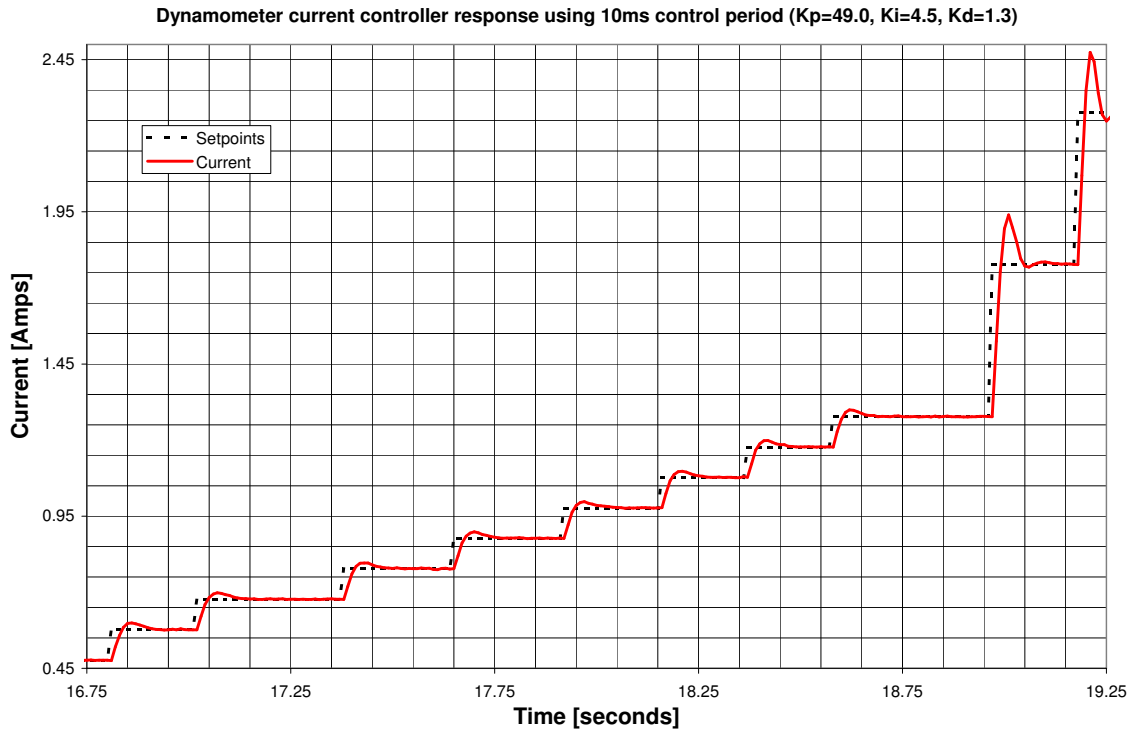

Figure 4.66: Response to incremental step increases in controller demand current

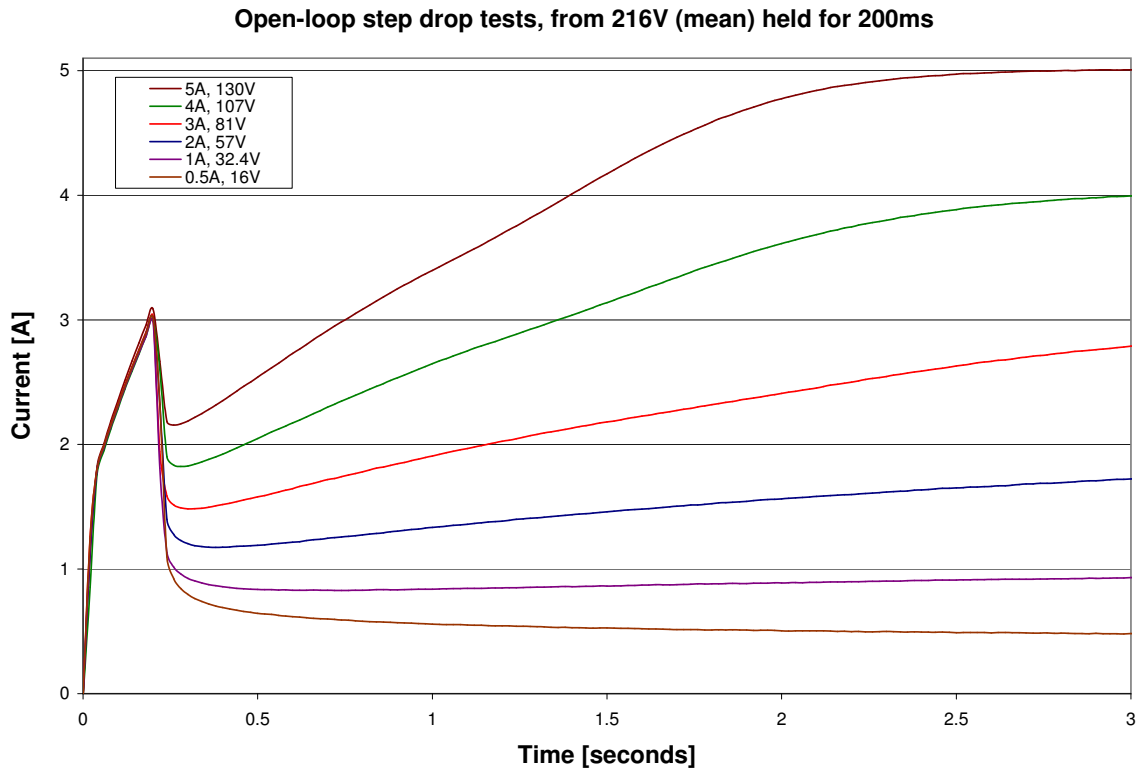

Figure 4.67: Open-loop response step decreases in applied

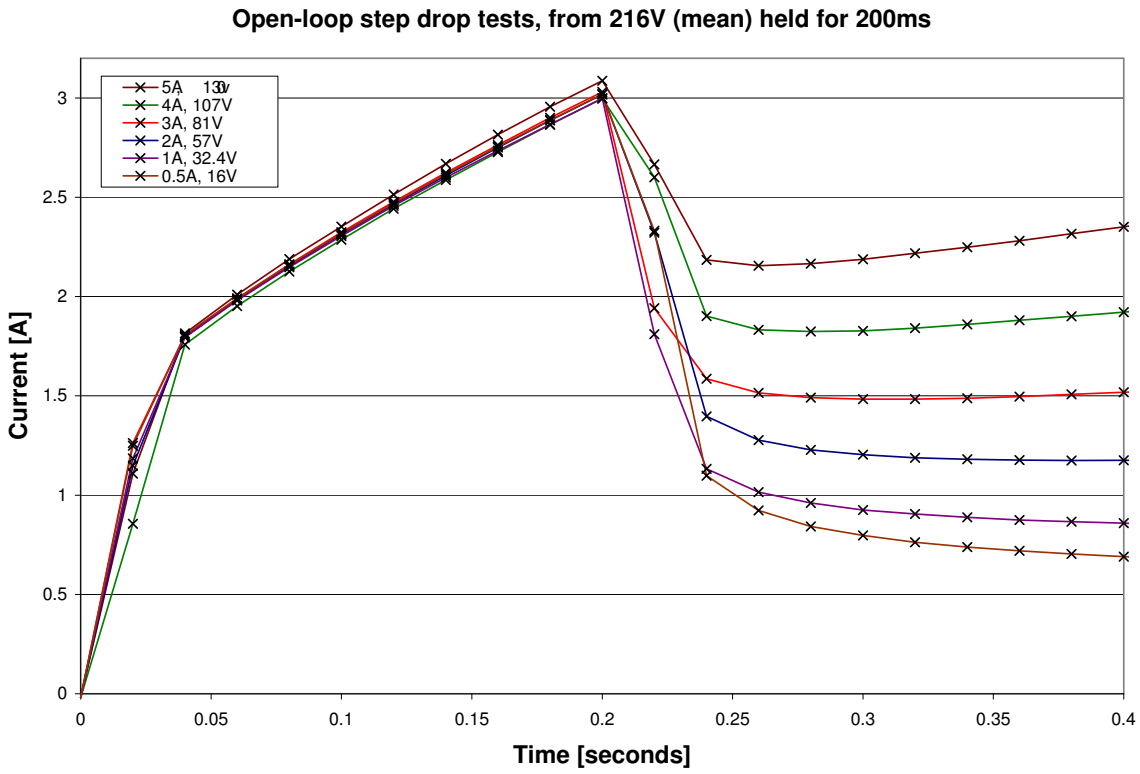

Figure 4.68: Open-loop response step decreases in applied voltage, initial response

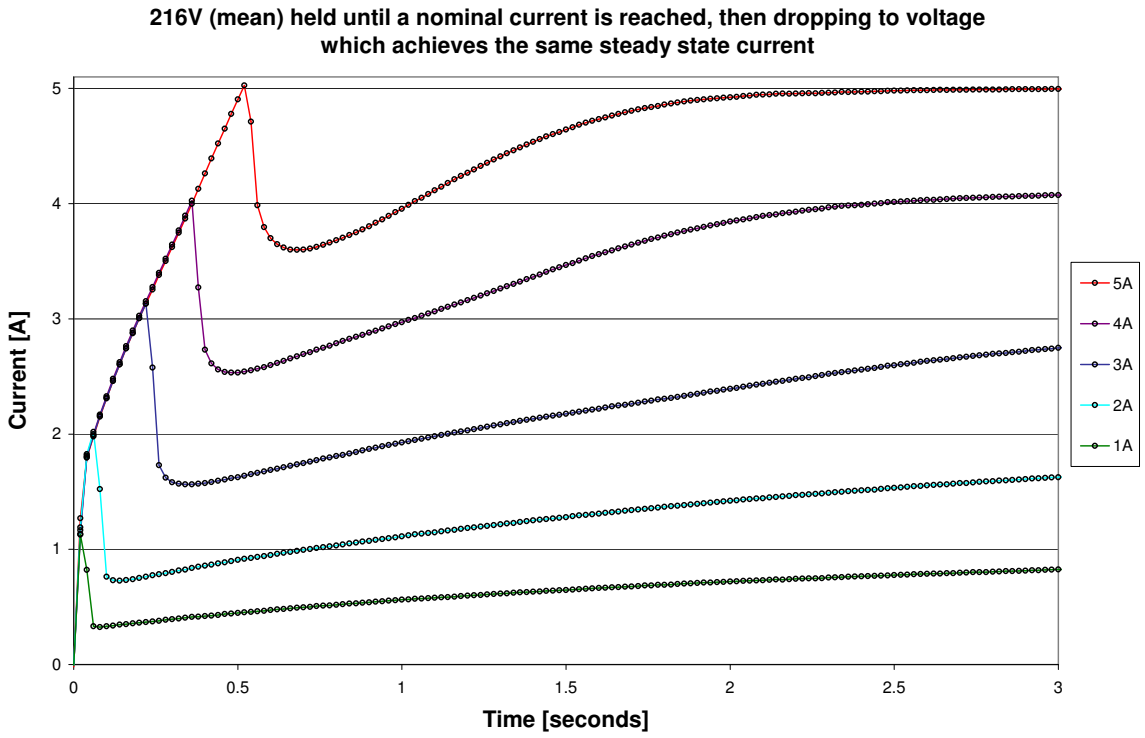

Figure 4.69: Open-loop response step decreases to voltage for equivalent steady-state current

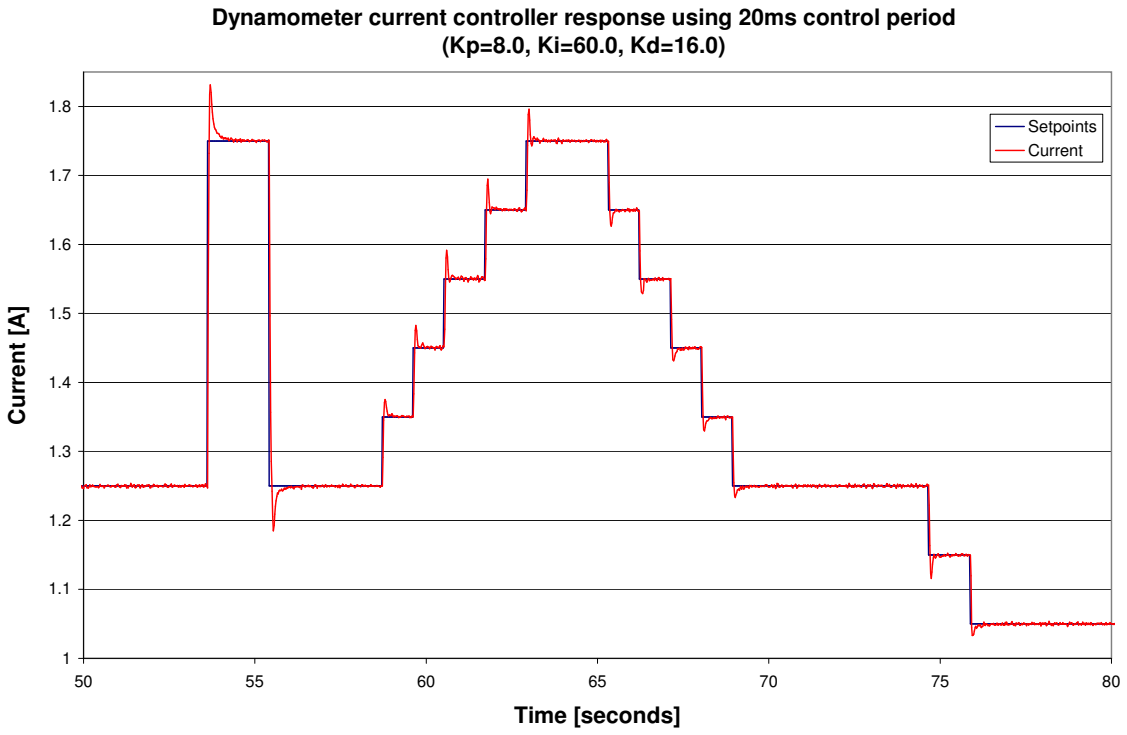

Figure 4.70: Response to incremental step changes in demand

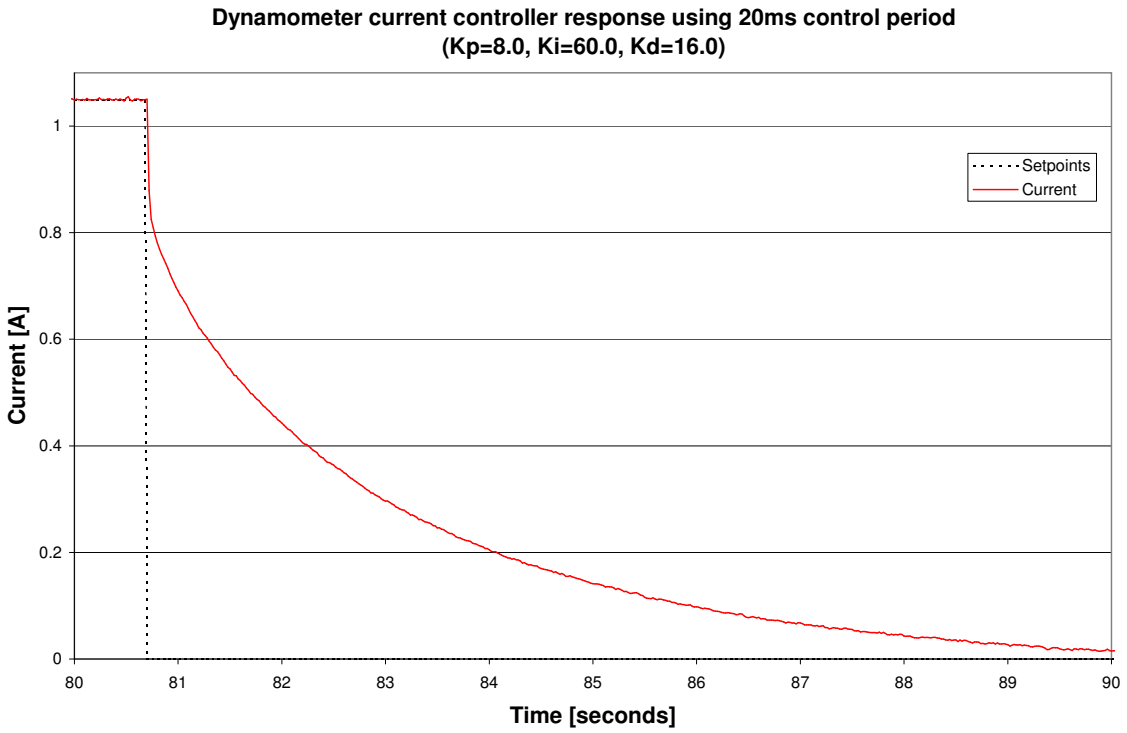

Figure 4.71: Response to step change to zero demand current

4.4 The Control Architecture

4.4.1 Control Hardware Selection

Several conventional proprietary dynamometer controllers feature a self-contained controller with an RS-232 interface to a desktop PC for monitoring and scheduling of the power/torque trajectory. A reduced data-set is provided over the serial line to the PC application software. Another commercial approach uses National Instruments Labview software on a PC running real-time Windows .

With recent developments in microprocessors and microcontrollers intended for embedded devices, there are now processors which can provide a computational performance comparable to that of a PC of only a few years ago, but at fraction of the cost, size, and power requirements. It was proposed that an integrated embedded approach would be investigated as a standalone solution that not only could provide deterministic closed loop control of the dynamometer, but also provide an economic platform for automation and monitoring of the test-cell. It could also provide additional data acquisition for independent verification of engine control strategies. With an embedded controller it would be possible to have direct access to all incoming data as it is acquired, and also provide a user interface as part of a self-contained unit. An LCD touchscreen (for embedded devices) was added to provide a graphical user interface for the test-cell operator directly from the embedded controller. Such a screen reduces the

need for mechanical switches, indicators, and instrumentation to be provided as these can all be implemented in software in a more flexible and configurable fashion. The addition of a network connection to the embedded unit also provides an external connection to a conventional PC, for example to allow permanent storage of any data collected, or remote operation.

The approach described above presented a number of technical challenges to achieve the several different roles from a single piece of hardware with only modest computational performance. To fulfil the design requirements there was a need to integrate several different software and hardware components without any certainty that an effective system would be realisable. Providing the operator with a responsive user interface without compromising the determinism of the control loop is one example of this. A discussion of the development and challenges of this system follows.

4.4.2 Arcom Viper

Recent years have brought an ever increasing number of embedded processor boards. Many of these are so-called *evaluation* boards intended to allow a particular architecture and platform to be assessed for use in a commercial product without need for the prior investment of time an effort to develop a viable board only to find that there are insurmountable shortcomings with the candidate architecture. This approach also allows software and hardware development to take place in parallel as software engineers can use a development board whilst hardware engineers are working on a production design. For the purpose of this project the choice of board was for a one-off design which precludes many of the evaluation boards from being useful. It was decided to narrow the search to PC/104 form-factor boards as these are fully functional and standalone with the option to expand the capabilities through the ISA bus. The majority of PC/104 boards are x86 processor based as the form factor is in essence a repackaging of legacy PC hardware onto a smaller PCB for embedded applications using recent advances in process technology to reproduce older processor variants in smaller packages and run them at lower core voltages so that passive cooling is often sufficient.

The Arcom *Viper* was selected, it is a single board computer (SBC) based around an Intel XScale PXA255 400MHz RISC ARM processor. The board (Figure 4.72) is PC/104 form factor and includes an ISA bus. The XScale architecture is the successor to the StrongARM which was an implementation of the ARM V4 instruction set architecture (ISA) developed by Digital Equipment Corporation (DEC), but later acquired by Intel (Montanaro et al., 1996) and targeted mainly at the personal digital assistant (PDA) market. The XScale is a variant of the ARMv5TE ISA architecture, but with no floating point instructions implemented, and is commonly used for portable devices requiring high performance with low power consumption, low heat dissipation

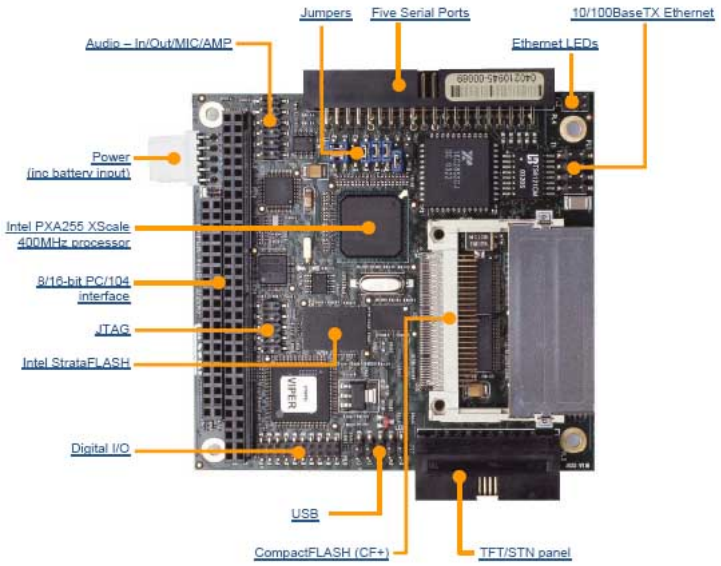

Figure 4.72: Arcom Viper SBC version 1

from a small IC package. XScale processors have a number of on-chip peripherals including multiple high-speed serial channels, PCMCIA host controller, a sound codec, and a integrated graphics frame-buffer for directly driving LCD panels. At the time that the selection was made this specification compared favourably both in terms of performance, specification, and cost with other x86 based PC/104 boards.

The board has suport for embedded Linux and since the author has previous experience with a Intel StrongARM based board also running embedded Linux there was a natural progression to using this. There was also a minor research motivation which was to determine the viability for this type of integrated system for use as a real-time controller and data acquisition system as well as at the same time providing a responsive graphical interface to an operator. The traditional approach is to have a dedicated microcontroller based black box connected to a PC application via some form of communication link, usually a serial or network connection. This integrated approach was found to be both challenging and problematic using the selected hardware and general purpose sotftware such as the X11 graphics server. It might have been possible to write (or purchase) dedicated fully optimised low level graphics software, but this would have been a project in its own right. Recent developements in ARM processors for mobile devices (such as tablets) have improved the situation by including more dedicated hardware support for rendering combined with the incremental increases in processor clock speeds.

4.4.3 Human Machine Interface and Controls

The various functions of the engine testbed can be adjusted from the operator box. The box provides an LCD touchscreen, a rotary encoder and mechanical switches which can be overriden in software. The LCD screen was initially a bespoke setup which was later

moved to the dynamometer control box along with the Arcom Viper board. An off-the-shelf unit replaced the first LCD interface, but provides the same funcionallity. Multiturn rotary potentionmeters were fitted to the side of the box with override swtiches to allow the dynamometer field current (and any other asigned variable) to be set manually. An analogue meter is also present to show the controller field current demand. This faciltiy was removed when the Viper board was transfered to the dynamometer control box, but the potentiometers have been left in place for future use.

LCD Interface Hardware A Panasonic 8-bit passive STN colour LCD screen with factory bonded glass plate 4-wire resistive touchscreen overlay was chosen as an economical display solution that is suitable for direct connection to XScale processor. TFT screens offer better contrast ratios, improved brightness, more colours (colour dithering not needed), and a much better response time. However, in the early stages of the project when the hardware was being selected the price of TFT screens were not cost effective and the interfacing can be considerably more complicated than passive screens. The Panasonic screen is 640x480 (VGA equivalent) resolution. A custom interface board was made to connect the screen to the Viper board, providing the necessary adjustable contrast voltage and regulated 3.3 V power supply which is taken either directly from the Viper or an on board regulator, selectable via a jumper. The interface board provides, as the datasheet for the screen prescribes, a strict signal assertion order as a violation could cause a DC signal on the liquid crystals potentially resulting in damage. This would occur if the interface cable was disconected or pulled out and the LCD was subsequently power on with no data signal. This configuration removes the need for a separate graphics controller as the screen shares the memory pipeline of the processor, but this also has a performance penalty which increases with the size of the display. A 12 V TDK backlight inverter is used with the display and is controlled by the Viper using an IC packaged solid state relay on the interface board. The backlight is autmatically switched on when the display is enabled by the device driver software running on the Viper board. The contrast is controlled either from a panel mounted poteniometer, or via a PWM signal from the Viper, an is jumper selectable.

Arcom TSC Resistive Touchscreen Interface The TSC touchscreen interface is a small Microchip PIC based board for resistive touchscreen overlays and provides 8 and 10-bit resolution and outputs over an RS-232 serial link which is connected to the Viper. Arcom provide a Linux daemon which runs in the background and passes touch co-ordinates to the X server which moves the mouse cursor. Only the 8-bit mode seems to work with the driver. The driver is closed source and the serial protocol is unpublished, but simple enough to deduce. The *tsc1* daemon works with a patched version of the X11 TinyX server. Calibration program is provided which generates a text

format configuration file which can be manually edited if needed.

The XScale PXA processor used on the Viper board is the successor to the StrongARM processor and has direct support for a Phillips UCB1x00 series chipset which privided certain features useful to a mobile handheld device through a dedicated serial codec. One of its functions was a resistive touchscreen interface. When Intel revised the StrongARM to form the XScale they removed this functionality from the architecture which has created the need for an independent touchscreen interface such as the TSC.

4.4.4 Diamond Systems DMM32AT Data Acquisition Card

A Diamond Systems DMM32AT Data Acquisition card was selected for use with the Arcom Viper board to provide analogue and digital I/O to the system. Many of the card's features are useful but cannot be used concurrently. Externally triggered sampling and gating was identified as useful for capaturing cylinder pressure data, but crank position sampling conflicts with control and other monitoring requiring periodic sampling. Also, different sample rates for different channels is not possible without triggering each individual sample or batch of samples from software. Access to the card's FPGA software would have made it possible to sample using a channel pattern scheme, but it was not feasible as it was a proprietary card and to develop a similar card would be a major undertaking and distract from the aim of the project. It is perhaps a design shortcoming that it is not possible to set from software a bitmap of a non-sequential channel sampling pattern.

In writing a driver for the card a significant issue was discovered. Both the Viper and the DMM32AT are specified as being having ISA connectivity (but don't claim compliance), but it turned out that they were not compatible, a high rate of spurious interrupts was occuring. Initially, the driver software under development was thought to be at fault, but after a significant period of investigation it became clear that it was a bus signal issue and not a software one. It was eventually found that the Viper was using a pull-down reistor on the IRQ bus line and the DMM32AT was using a pull-up resistor. The logic level was floating close to the detection threshold and since edge detection was being used, bus noise from data transfers was coupling onto the IRQ line creating bursts of false interrupts. The PC/104 ISA specification (Haris, 2003) outlines the use of pull-up resistors to allow IRQ lines to be shared between cards, and thus the Viper board violates this requirement by using pull-down resistors. The response from Arcom was that the board was not designed for interrupt sharing and that the Viper's bus was not a *real* ISA bus since the bus logic is carried out by a CPLD device and not an ISA chipset. However, it must be reasonable to assume that other card manufacturers would implement their card's logic arrangements in a way outlined by the PC/104 specification to be compatable with other cards even if interrupt sharing is not directly supported.

Arcom were eventually persuaded to exchange the Viper for the second generation of the board which uses weaker pull down resistors. Version 2 of the board was found to work much better with the DMM32AT and only produced spurious interrupts at a rate of several per second. These interrputs can be detected and ignored in software and at that rate do not have a detrimental effect on the system.

4.4.5 CAN-bus interface

An Arcom CAN-104 PC/104 form factor CANbus interface which is based upon Philips SA-11001 controller has been used to provide a CAN interface to the Viper board. This is to allow deterministic data exchange between the engine controller and the dynamometer controller should this be required.

4.5 Dynamometer Instrumentation for Control

4.5.1 Tachometer Circuit

The dynamometer had at one time been retro-fitted with a 60-tooth gear and sheet metal cowling on the rear output shaft shown, as in Figure 4.73. The cowling is used as a mechanical guard and as a means to locate a variable relucatance magnetic pickup sensor in proximity with the gear teeth. The choice of 60 teeth makes the conversion from frequency to revolutions-per-minute trivial as the seconds to minutes conversion cancels with the number of teeth division which makes the tooth frequency directly equal to the shaft RPM.

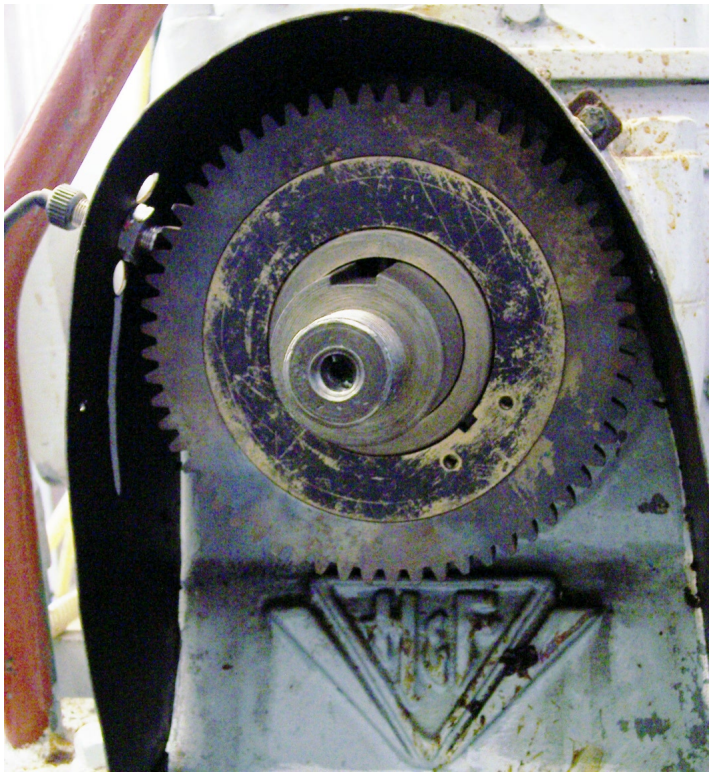

Figure 4.73: Dynamometer 60-tooth shaft-end gear and variable-reluctance sensor

It was required to convert the tooth induced pulse frequency into a form which could be read and used by the digital control system. For this task a frequency-to-voltage converter (LM2907) was chosen. The circuit was constructed in a similar manner to that illustrated in Figure 4.74. This circuit takes the signal directly from the sensor and, using a charge pump, produces a voltage proportional to the input frequency and the supply voltage using a timing capacitor, an output resistor, and a integratinag filter capacitor. The choice of these components forms a compromise between output voltage ripple, linearity, and the maximum measurable intput frequency. In order to supress the effect of supply voltage variations, a voltage reference circuit was used to supply the tachogenerator. The reference was given an improved current sourcing ability using an npn-transistor in the same way as was used to supply the load cell described in 4.5.2. The analogue output from the tachogenerator was the order of 900 rpm/V and was sampled using the ADC card described in Section 4.4.4.

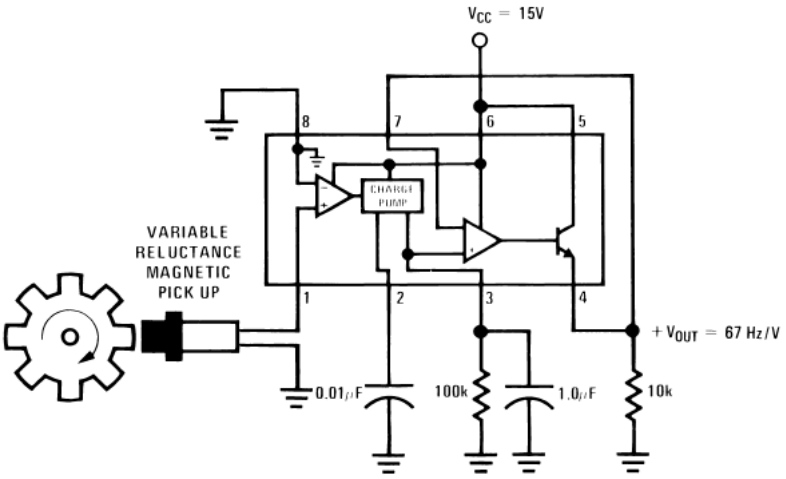

Figure 4.74: Typical tachometer circuit using the LM2907 IC (National-Semiconductor, 2000)

Although the circuit appeared to function well, the LM2907 IC failed in service after several tens of hours of use. It is thought that the 28 V maximum input voltage may have been exceeded at some point during its use which may have contributed to its premature failure.

During of the second phase of work a new digital tachometer circuit was designed and made. This consisted of two major components, the National Semiconductor LM1815 adaptive variable reluctance sensor amplifier and a UFDC-1 universal frequency-to-digital converter produced by SWP Inc. The LM1815 was used for signal conditioning to convert the sensor voltage into a logic level pulse train, and it's opertation is discussed in Section 5.6.3 since it was also used in the design of the engine controller developed for this project. The UFDC-1 has a choice of serial interfaces (logic level RS-232, SPI, I^2C) and references the input pulse train against a 16 MHz quartz crystal clock. The accuracy of the measurement given is in proportion to how rapidly updated values are requested and ranges from 1 to 0.001% relative error, implying that a successive approximation is made which improves with time. The UFDC-1 has a calibration procedure which outputs a 8 MHz square wave signal at exactly half the input crystal's frequency. This signal can be used to cablibrate a correction factor if an higher precision instrument is availible to measure it with. The calibration factor is stored in non-volitile memory and is used for all future calculations unless the procedure is repeated. Little information is available about the electronic design details of the IC as there is only a combined brief specification and application note available which covers only external connections, setup, modes of operation and serial command protocols. It is postulated from the pin-outs that it is constructed from a Microchip PIC or similar low cost 8-bit microcontroller. The circuit was tested and found to work well with the dynamometer and the rpm estimation showed good correspondence with that displayed by the Nippon Denso ECU software. The only limitation for this application found with

the UFDC-1 was that RPM data can be obtained either by polling or via an autonomous mode using external DIP switch settings and outputing data at a periodic rate at a baud rate of 2400 bits/s. The periodic rate was found to be far slower than required for a control loop which makes the polling mode preferable. The limitation is that the IC was (although not documented) found to enter its automous mode several seconds after power-up if no serial commands had been received. Once in autonomous mode the IC ignores any further serial commands and cannot be reset using serial or any logic input to the IC. This is a problem when interfacing to a Linux system as the boot time is much longer than serveral seconds. The solution is to control the power to the IC from software using either an uncommitted GPIO line of serial control line to a small IC packaged solid-state relay. This problem was not anticipated so power control was not included in the board's design and will have to be added externally, or as a temporary solution the board could be manually reset with a switch when the dynamometer control software is about to be started. The UFDC IC is covered by an aritcle in an industry journal (Yurish, 2007) and also its performance for a particular application is more independently evaluated in a conference paper (Pereira et al., 2005).

4.5.2 Load Cell and Amplification Circuit for Torque Measurement

A load cell amplifier was designed and made based upon a AMP04 instrumentation amplfier IC. The board was designed to operate from a single rail linear supply in the dynamometer power electronics box and provides a 10 V precision reference supply for the strain gauge bridge in the load cell. The signal amplfication gain is set using a 25 turn potentiometer mounted on the board. The board has been designed to give maximum supply ripple rejection though the use of a large supply and ground plain area and decoupling as close to the supply pins active devices as possible using both tantalum and surface mount ceramic capacitors to offer a combination of a high reservoir capacity and low in series resistance. Care has also been taken to reduce thermal drift effects by adjusting the null offset as close to zero as possible without causing clipping due to the single rail supply. It is important that the offset pin is driven from a low impedance source as any measurable resistance will degrade the amplifier's common-mode rejection performance as well as its gain accuracy. An OP213 op-amp has been used to buffer the offset null circuit as shown in Figure 4.75.

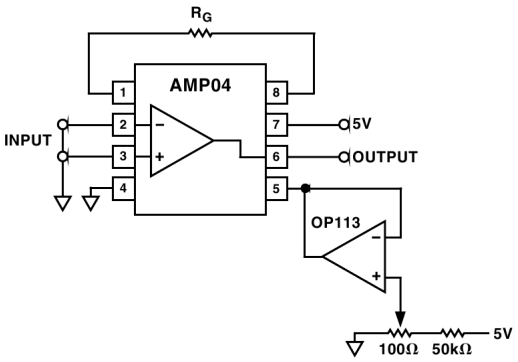

Figure 4.75: AMP04 offset nulling scheme (Analog-Devices, 2000)

The cable shield is often grounded at the analog input common in many applications, however it is claimed (Analog-Devices, 1999b) that improved dynamic noise rejection and a reduction in effective cable capacitance is achieved by driving the shield with a voltage follower at a potential equal to the voltage seen at the input. Figure 4.76 shows an active guard drive suggested for use with AD623 and is configured to improve AC common-mode rejection by *bootstrapping* the capacitances of input cable shields, thus minimising the capacitance mismatch between the inputs. This scheme was employed for both the input and output cables using the uncommited amplifiers from OP213 dual operational amplifiers (the other amplifier being used for either the voltage reference of offset nulling of the AMP04). It was hoped that by actively driving the shields the general noise immunty would increase, however it was found to encourage cross coupling of signals between separately shielded signals in a multicore cable used to supply the RPM voltage signal and torque signal to the dynamometer operator control box. For this reason the output cable shield was simply to a grounded at one end. The full schematic for the board is shown in Figure A.3.

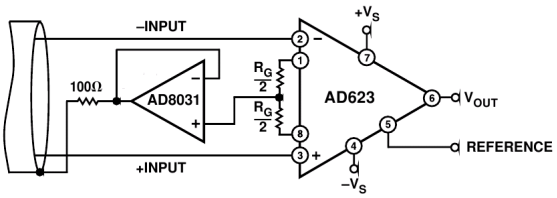

Figure 4.76: Common-mode shield driver (Analog-Devices, 1999b)

4.5.3 Protection Circuit

The dynamometer was fitted with a motorised water valve of a type commonly used in domestic central heating systems, and contains a switch to indicate when the valve has fully opened. This valve is used to switch on the supply of cooling water from the mains supply using a solid state relay in the control box. The dynamometer was originally fitted with a mechanical pressure switch which was used as safety cut-off if the cooling

water flow was insufficient. This device was removed an replaced with a standard adjustable pressure switch. The pressure switch and water valve swith are both supplied with a current limited 5 V signal which can be used to ensure the valve has opened and that water pressure is present. There is facility to perform these check either in software or to allow a zero duty signal to be switched in place of the control signal to the phase angle controller. This is performed using an analogue switch IC. LED indicators were mounted on the control box front panel to indicate the valve and pressure states. These two signals were also made available from the protection circuit at logic level.

4.6 Automation Hardware

4.6.1 SSR Engine Power Management and Opto-Coupled Interface

The battery ignition supply to the engine, its starter motor solenoid, the fuel pump, and a cooling fan are all controlled using high current solid state relays (SSRs). These SSRs can be controlled by the 12 V nominal battery voltage and are connected to operator control box using low current signal wire in a multi-core shielded cable. The SSR cable connects to a power control board which allows conventional mechanical switches mounted on the front panel of the box to activate the SSRs. The board also allows the supervisory embedded system to overide the switch states using a set of opto-couplers. The opto-couplers have been used to provide voltage level shifting from digital logic level to battery voltage, but the grounds are not isolated so there is still electrical coupling between the engine and the control system.

4.6.2 Plint Volumetric Fuel Meter and Digital Retrofit

Figure 4.77: Plint TE12 volumetric fuel meter with digital range selection

A Plint volumetric fuel meter (Figure 4.77) was obtained and fitted to the engine testbed. It consists of a vertically mounted cylinder into which three optical level sensors are fitted. The engine takes and returns its fuel to the cylinder. The cylinder level is controlled by a solenoid valve and is filled under gravity from the main tank. Once filled, the solenoid closes and the engine operates on a closed loop with the cylinder acting as a small reservoir. A digital counter keeps track of how long it takes for the fuel level to drop between two of the level sensors and latches the result. The time/counter is displayed on an LED digit display on the front panel and is also output for external measurement as an analogue voltage proportional to the final latched time using a 12-bit DAC. The particular model of Plint fuel meter was fitted with a mechanical two position rotary switch which is used to select between 100 ml and 50 ml measurement ranges. Later models used a relay so that the range can be remotely selected. It was desired to add this remote selection facility to the existing meter so the rotary switch was replaced with a custom daughter board (A.4) which was socketed into the original location of the switch on the main PCB. In order to retain manual operation, a non-latching press-to-make switch was fitted to the original switch's front panel aperture. Two panel mounted LEDs were used to indicate the selected range. A digital flip-flop circuit was used to latch the selected range which can be toggled using either the front panel switch or by a control line to a microprocessor, allowing the intended remote operation. The daughter board uses an analogue switch IC to replace the mechanical switch so that the original logic levels of the main board are preserved.

4.6.3 Cylinder Pressure Acquisition System

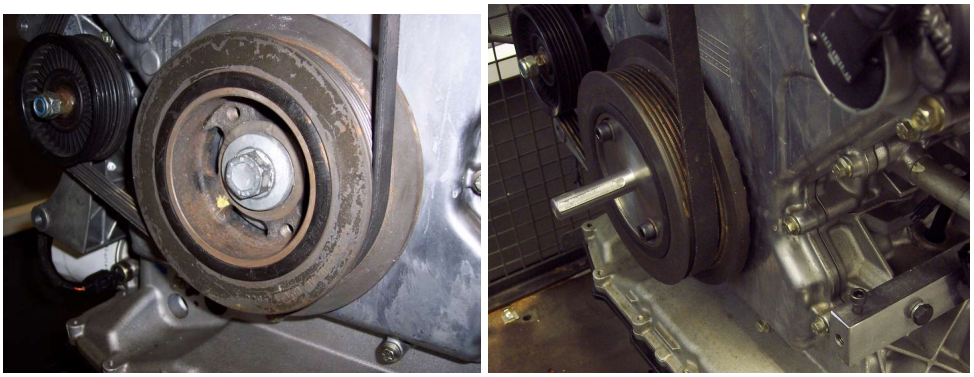

Figure 4.78: Engine crank pulley (left) fitted with adaptor for an encoder shaft (right)

This system is used for cylinder pressure sampling at fixed crank angle intervals. A second Diamond Systems DMM32AT data acquisition card was used with the x86 based embedded system in the operator box. The engine was fitted with a Hohner Series 85 incremental encoder rated up to 3000 rpm with 0.5 degree resolution. As the engine may be operated at up to 6000 rpm (the dynamometer's maximum speed rating). The

manufacturer was contacted to find out what the limiting condition for encoder's maximum speed is. Hohner said that the bearings could be run at 6000 rpm but the optics and trigger circuit may impose a limitation. The encoder was mounted on an electric motor and run up to 6000 rpm to check for distortion or corruption of the output signal and none was found, so it was deemed to be suitable for this application.

HCTL Encoder Interface Circuit The Hohner encoder provides complimentary inverted and non-inverted TTL level logic for the A, B and I channels which can be compared with each other for noise rejection. As there is a relatively long cable run in the presence of electrical noise from the engine ignition system, mains supply, and dynamometer power electronics, the encoder's differential logic signals where used. An RS-422/485 line receiver chip such as the MAX3095 could have been used to decode these signals which would provide some ESD protection, but instead a pair of hex-inverter and AND gate logic chips were used together with a pull-down resistor network to place some current loading on the signals for further noise immunity.

The encoder interface board (Figure A.7) was designed to use a HCTL-2032 IC, which was chosen to decode the encoder quadrature signal and maintain a counter of the encoder ticks, which it is capable of doing for high frequencies, dependent upon the timing crystal fitted to the circuit. The HCTL-2032 is able to separately decode two independent encoders on a multi-axis system. As only one encoder was required for the engine crankshaft encoder, the other spare axis counter was used with a mechanical encoder (of the type commonly used as a digital volume or *jog* control) to provide the operator with a convenient means of setting control parameters such as load torque or speed set point. The HCTL-2032 has two 16-bit counters that are read through an 8-bit digital I/O interface (lower byte first) which was connected to the Diamond DMM32AT PC/104 card. If only the lower 8-bits of counter depth are required then the second byte of the word does not have to be read and can be ignored as the lower byte wraps around.

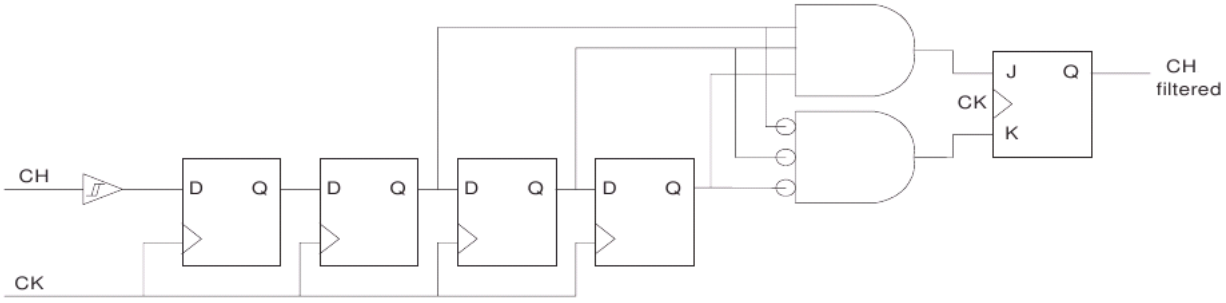

Figure 4.79: HCTL digital channel filter (csewards, 2004)

The HCTL chip has a digital noise filter input section which is responsible for rejecting both common-mode and differential-mode noise on the incoming quadrature signals

which might otherwise cause false counts to be triggered. Figure 4.79 shows the simplified schematic of the input section which is repeated for each of the A, B and I input signals. Two techniques are used to achieve improved noise rejection. The inputs are Schmitt-triggers and a three clock-cycle delay filter is used to reject both low level noise (<1 V) and larger, short duration, noise spikes that typically occur in the presence of switched power electronics (such as is used for the dynamometer). Using a Schmitt-trigger buffer addresses the problem of input signals with slow rise times with low-level noise imposed upon them. The signals are passed onwards to a four-bit shift register delay filter which samples each channel on rising clock edges and constructs a time history of the signals. Any change of the input state is tested for a stable level being present for three consecutive rising clock edges. The filtered output waveforms can only change after an input level has the same value for three consecutive rising clock edges and pulses or noise spikes shorter than two clock periods are rejected. The HCTL chip can be clocked at upto 33 MHz, however at 6000 rpm, this equates to 72 kHz, so a lower frequency clock will suffice. A 10 MHz TTL output oscillator was used and found to work, however the duration of the output pulses where later found to be too short to reliably trigger the DAQ card leading to samples being randomly missed, so the 10 MHz oscillator was replaced with a 4 MHz oscillator and thereafter it functioned correctly.

Figure 4.80: Optrand PSIplug and M3.5 sensor

Optrand Pressure Sensing Spark Plug The test engine has a plug-on-coil type ignition system. This raised some concern about whether the Optrand modified spark plug and sensor (Figure 4.80) could be made to actually fit the engine. The first issue is whether the modified spark plug would still fit into the bore, and the second was whether there is sufficient clearence for the steel braided fibre-optic sensor cable to be routed out

of the bore past the coil pack (Figure 4.81) without violating it's minimum allowable bending radius. The bore and coil channel was measured for diameter and profile and the plug, coil, bore and cylinder head cover channel were parametrically modelled (Figure 4.82) using Autodesk Inventor to establish what clearences are present when the coil is assembled onto the engine. The sectioned drawings were provided to Optrand to serve as requirement for the size constraint of the sensor protrusion on the modified spark plug. When the sensor was received it was found that the insulation boot on the coil needed to be thinned slightly by abrasive linishing to fit between the fibre-optic cable and the spark plug ceramic insulator. This was the only modification to the engine that was required to fit the sensor.

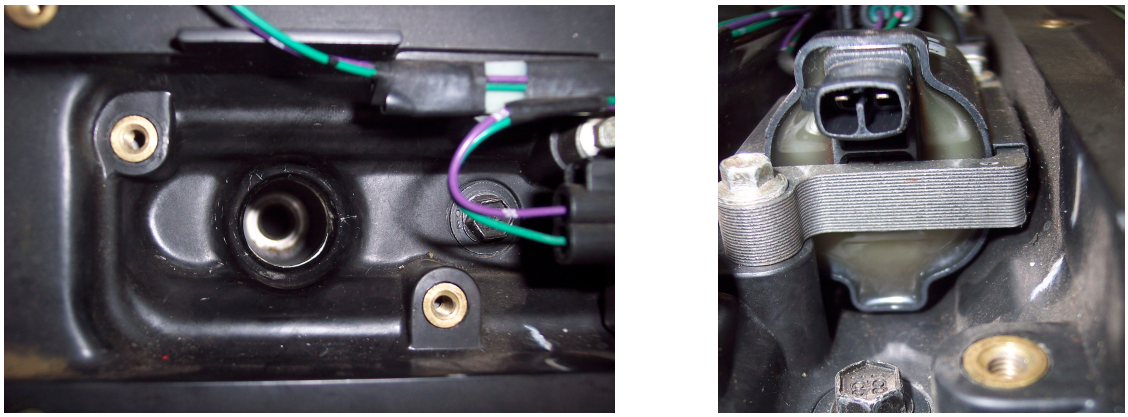

Figure 4.81: Photographs of spark plug bore and coil channel

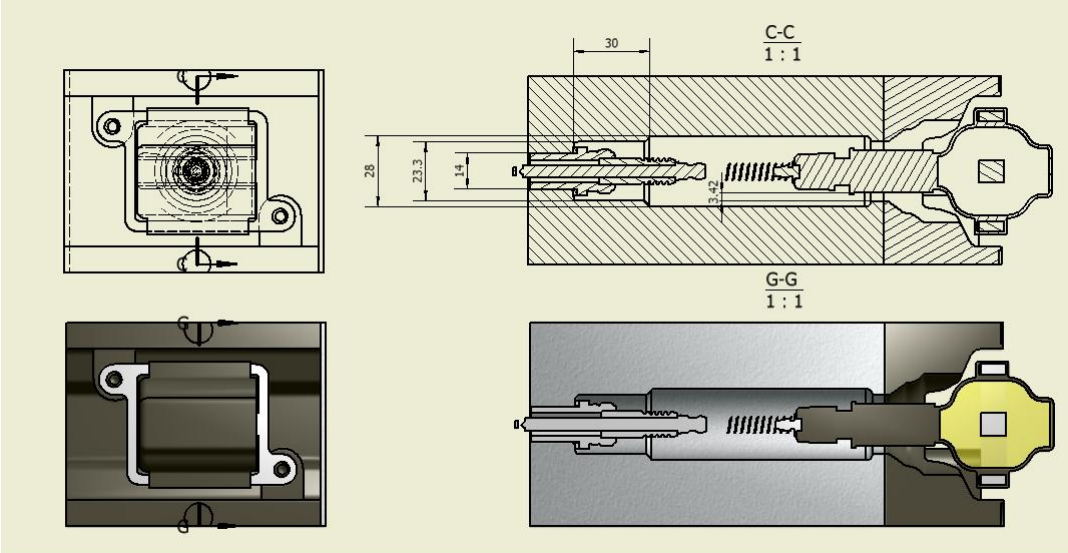

Figure 4.82: Spark plug bore and coil channel 3D parametric model

Determination of the Top-Dead-Centre Position Despite the fact that the engine will normally rotate in one direction when running, all three A, B and I channels where used. This is so that a manual piston top-dead-centre (TDC) search calibration could be performed which requires the engine crank to be turned in both directions so that TDC

position for the instrumented cylinder could be determined relative to the encoder's index mark. With the spark plug removed, a dial test indicator (DTI) gauge was inserted into the bore to rest on the piston crown and measure piston displacement. Knowing the number of counts between the index count and TDC allows data to be *pegged* to actual crank angle values in a repeatable way. The reach of the DTI gauge was extended using an aluminium welding rod with one end threaded and the other probing end rounded to a ball-nose. The AJ-V8 pistons are a flat top design and the spark plug bore is concentrically aligned with the cylinder so that the DTI probe can be introduced orthoganally to the piston crown. The engine was rotated manually using a spanner to search for the TDC position. The trajectory of a slider-crank mechanism means that the piston displacement is least sensitive to crank rotation at the furthest extremes of its travel. This meant that in the vacinity of TDC, there was no appreciable change in DTI gauge reading over a rotation of several degrees. The physical process of rotating the engine by small increments is also difficult to achieve by hand as once enough force is applied to the lever to overcome the friction then engine will tend to rotate more than the desired amount so that the process has to be repeated in the opposite direction and back in the original direction to eliminate backlash effects. After repeating the process several times, an angle central to the measurement dead-zone was recorded. When the engine was motored without any combustion taking place, the peak of the pressure trace was found to be out of alignment with the predicted TDC by about 1.5 degrees so the software offset was adjusted accordingly.

Collection of Cylinder Pressure Data Although the encoder index postion was known relative to TDC, a mechanism for pegging the samples to the encoder index was still needed. The technique devised was to gate the sampling process until the index pulse has been reached for the first time which was achieved using a software resetable latch circuit. The index pulse triggers the latch which enables sampling on the DAQ and the index is then ignored unless the latch is reset at some point in the future by a software controlled digital output. Providing that no samples are ever lost or gained due to false triggering, the angle of every sample is known by its relative position in the sample buffer. The DAQ can buffer up to 500 samples, but was set to dump the buffer every 360 samples by raising an interrupt. The card driver then transfers the samples from the card and buffers them before delivering them in multiples of 1440 to the application software. The application software is able to detect from the measured pressure level which half of the 1440 samples contains the combustion part of the cycle and choose to reject the other half or window the data further to the region of interest. The combustion phase can never be known in advance as the latch circuit can be triggered with the engine in either phase for the instrumented cylinder. Also there is a requirement that the latch should only be reset with the engine stationary as it needs to be done before the DAQ driver is configured so that the sample buffer is empty when the latch is triggered. With the

engine running, the software latancy between reseting the latch and configuring the card means that sampling does not repeatably start from the same angle. However, this is not an unreasonable constraint to perform the setup before starting the engine, as the sampling process can continue continously from engine startup, discarding the data if not required. With this approach, there is a risk that if the DAQ was to be triggered falsely then the angular error would acculumate over time. The aquisition system was run for extended periods of over half an hour and at no time did any noticable angular offset error accumulate which implies that the noise protection measures are adequate. Figure 4.83 shows some pressure data captured during an engine start cycle. The lowest trace occured before any combustion has taken place and is therefore has a peak pressure symmetric with TDC.

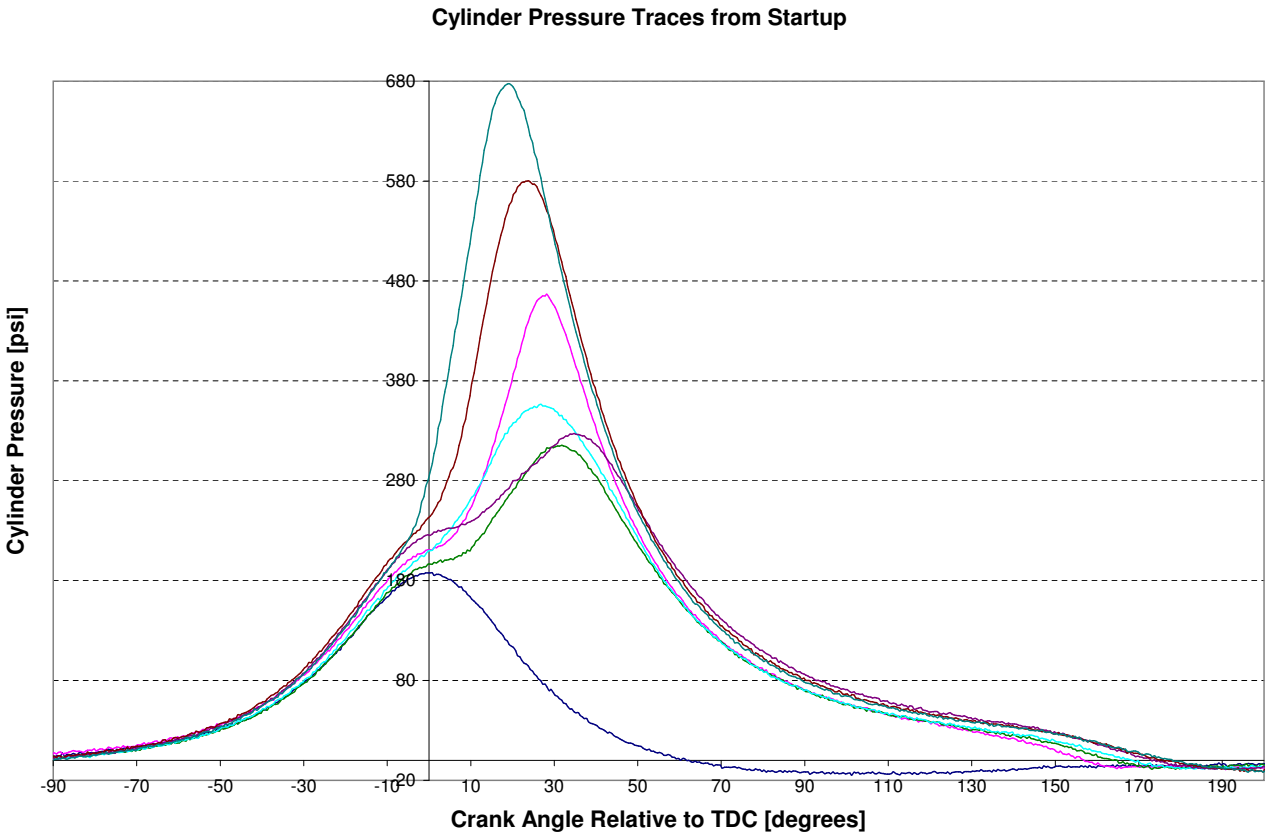

Figure 4.83: Cylinder pressure curves obtained during engine start-up

4.6.4 Thermocouple Interface

A thermocouple instrumentation board (Figure A.5) was developed to allow monitoring of the engine cooling water and exhaust gas temperatures. These measurements are intended for general health monitoring rather than for modelling or control purposes. A Type K Chromel/Alumel thermocouple has a useable measurement range –200°C to +1250°C giving –5.973 to +50.633 mV. The relationship is not linear and polynomial reference calibrations are available in the ITS-90 Thermocouple Database from the (US) National Institute of Standards and Technology (NIST). The AD595 (Analog-Devices,

1999a) IC was used with an ADG507 mutiplexer to amplifiy eight thermocouple channels. The AD595 is an instrumentation amplifier with an integrated thermocouple cold junction compensator on a monolithic chip. It combines an ice point reference with a pre-calibrated amplifier to produce a high level (10 mV/°C) output directly from a thermocouple signal and gives the following input/output relationship:

$$AD595Output = (TypeKVoltage + 11\mu V) \times 247.3$$

The IC's calibration is centred around 20°C but (due to the non-linearity in the actual thermocouple's characteristic) the relationship diverges for colder and warmer temperatures. As the type K thermocouple has greater measurement range above 0°C, then this is more significant for higher temperatures. As an example (from the table provided in Analog-Devices (1999a)) at a thermocouple temperature of 800°C the AD595 will output 8.232 V which corresponds to a measurement error of 23.2°C. For this project temperature measurement is intended for health monitoring and consistency of data collection, then this static error will not be an issue as it will be repeatable and can be compensated for using online software or offline, should a greater level of accuracy be required. The circuit did suffer from a larger source of error created by ground currents between non-electrically inslultated probe tips and the circuit. The multiplexer is an analogue switch and has typically around 280 Ω of impedence between the input and output of the selected channel. As the voltage being measured is very small then it only required a small amount of ground current to induce a large offset error voltage across the multiplexer's impedance. The insulated probes produced only a few degrees of offset error, whereas the non-insulated probes could be out by the order of 40 degrees.

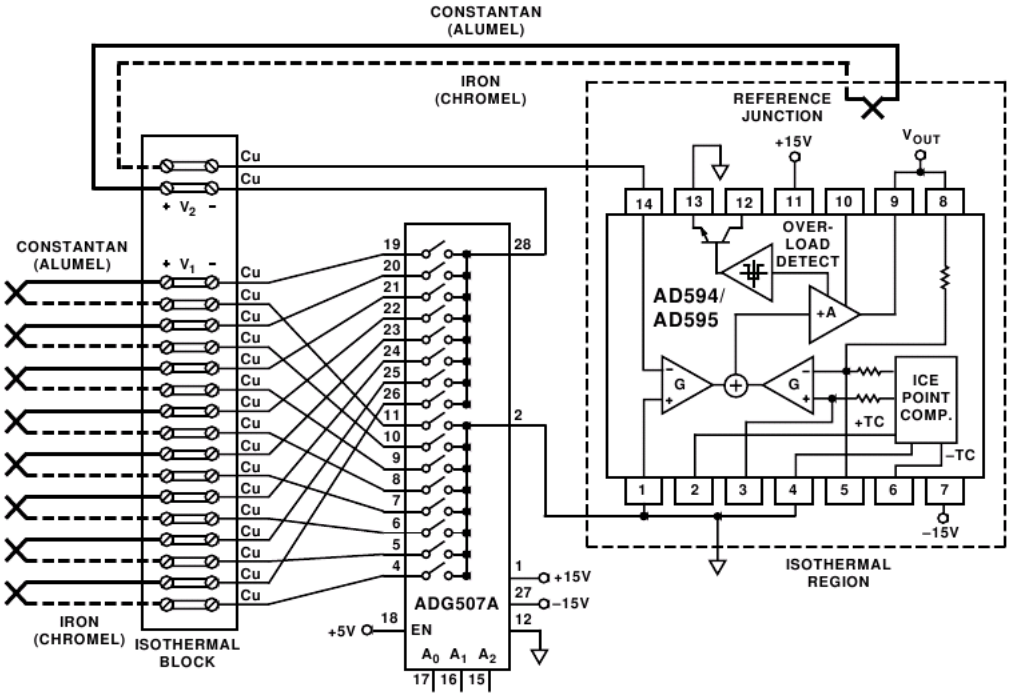

Figure 4.84: Thermocouple multiplexing circuit (Marcin, 1998)

4.7 Software Architecture, Selection, and Development

The Acrom Viper SBC is supplied with Redboot boot-loader which is an eCos application. There is also available a board support Linux package which Arcom call *Arcom Embedded Linux* or AEL, that is loosely based around Debian Linux and makes use of a number of commonly used embedded tools for a reduced memory footprint. A commercial version of FMSlabs RTLinux was also available at considerable cost, although the exact level of integration with the Viper board is not disclosed. As both eCos and the Linux components are all covered under variations of the GPL licence, the development CDROM was obtained from Arcom.

A decision had to be made as to which operating system would be used, balancing the real-time requirements against the up-front porting and setup effort with cost and the potential for rolling out the solution to other applications with real-time requirements within the research group. Any choice would involve a potentially serious commitment of development time which would make it difficult to change at a later stage. After an initial investigation into porting the ARM varient of an open source real-time linux entention called RTAI, it was decided to use AEL Linux as the hardware interrput from the DAQ board could be used to create periodic software event relatively independently of the scheduler's timing mechanisms. The main challenges arrising out of AEL were the age of the packages versions and the limited flash storage space and the host tool's dependence on an obsolete and supported version of Linux Standard Base (LSB).

Appendix B.2 provides further detail on the rationale on how conventional Linux was chosen to implement the embedded controller.

4.8 Control Software

The control software for the dynamometer was based on the ARM based Arcom Viper board and a Diamond Systems data aquisition card (DAQ) connected to the Viper through an ISA bus. The software written was cross compiled in a sandbox environment called Scratchbox on a Linux host PC. The control loop uses the DAQ card to generate an interrupt based periodic event when a batch of samples has been acquired. The control code was initially written as a standalone C program with the intention of merging it into a C++ based GUI monitoring application being developed in parallel for the same hardware. When difficulties were encountered with the GUI software, the control program was migrated to a standalone C++ program with a console display. Further information on the Diamond DMM32AT card and API compatibilty library issues are provided in Appendix B.3.

4.9 Application Software

This section describes the work carried out to provide an application for a user operator interface and for monitoring and data recording on the engine testbed. Application software was initially written to run on the ARM based Arcom Viper board for display on an LCD touchscreen. The application software was intended to provide a front end for the dynamometer control software. After much difficulty with rendering issues and general performance, the software was moved to an embedded x86 platform (Kontron MOPS) which had nominally similar processor performance, but owing to dedicated graphics hardware, the application is able to run at an acceptable rate on a larger VGA signal compliant LCD screen. The control software was retained on the Viper board and a colour *ncurses* console was used instead of a rendered window environment, as this was within the capability of the display. The MOPS board was used with Crunchbang Linux 9, derived from Ubuntu, but simplified to use Openbox window manager and other minimalistic software making it an appropriate starting point for an embedded PC. Crunchbang was installed onto a 2 GB high speed CompactFlash card and was used with a CompactFlash to ATA-IDE adaptor so that the MOPS board could be used without a hard disk. Any software packages missing from the default Crunchbang install (such as wxWidgets) could be installed via a network connection to the board from the normal Ubuntu package repositories. Appendix B.4 provides more discussion on the chosen Windows Server TinyX used on the Viper board (SmallX as it is now commonly called).

4.9.1 The mseDyno Application

The torque signal can be optionally filtered using the filter button below the plot area. The filter is an expontentially weighted moving average filter of the form:

$$Y_t = \alpha X_t + (1-\alpha) Y_{t-1}$$

where α is a tuneable coefficient that can be related to the time constant of an R-C filter for a fixed sample period of ΔT:

$$RC = \Delta T \frac{(1-\alpha)}{\alpha}$$

This filter was used for its low computational overhead due to its recursive nature and is a simple infinite impluse response (IIR) as it depends on both previous inputs and outputs and has an impluse response similar to a physical R-C filter. As such it is said to be *causal* as it cannot see ahead into the future and so introduces some phase lag seen as a time shift delay between the filtered and unfiltered signals. Since the data is being process in time-triggered batches, the lag could possibly be removed using a *non-causal* filter that require data ahead of time, but it was not considered to be of high importance to do so for a monitoring application. Such a filter would also prevent the most recent sample data points from being rendered at the time they are received as the ahead of time data (relative to those points) would not be available until the next batch arrives. The full torque signal deviation for each decimated point that is redendered at a particular zoom level can be selectively shown using the Min-Max button. The rendering approach for plotting incomming data in near-realtime (as it is received, subject to varying transfer delays) could adopt one of three approaches.

1. Continuously scroll the plot from a fixed point. This would require buffering the data blocks in a FIFO and create a data stream or delaying rendering by one or more blocks of data so that there is always at least one block of data *in-hand*. This approach results in a high rendering overhead as the whole area has to be replotted each time the data window is advanced. The continuous rendering would normally result in either a visible flicker (if the plot area is cleared after each rendering) or tearing (if the plot area is incrementally overwritten and background blanked) that would require double buffering to remove. Double buffering is the process of rendering to a *hidden* plot area then swapping the completed scene into view. The previously viewable area then becomes hidden and can be used to render the next frame. wxWidgets supports double buffering, but the actual implementation depends on lower software layers not accessible to the user API (other than to

enable double buffering and controlling when the frame is to be switched), so there may be significant additional overhead caused by the block image transfer (BLIT) depending upon how much assistance the graphics hardware is able to provide and how that hardware support has been utilised by the various software layers that lie between the framebuffer(s) and wxWidgets.

2. Continuously wrap-around and over draw data. This technique can often be observed on digital electrocardiogram (ECG) displays in hospitals and also digital oscilloscopes when set to a slow timebase. The data stream is plotted from left to right and starts again from the left side over drawing the previous trace. This results in a visible tearing or seam where the old and new data meet. In this case the tearing is not a problem as it marks the current point in time and can be marked by a moving vertical cursor line if desired. This technique results in a lower rendering overhead as the whole data window does not have to be redrawn at each time increment, only new data added. Also, the overhead is consistent so there is a reduced likelihood of glitching. A possible drawback is that once data has left the plotted window, then it is gone for good as it does not make sense (from a user interface perspective) to attempt to scroll a data window that has been wrapped around.

3. Plot data as it arrives and scroll the plot area by fixed increments. This is a compromise between continuously scrolling and continuously plotting (and wrapping around) the data. New data is appended to the existing plot either on a time interval basis or in blocks as it arrives. This prevents having to redraw the entire visible data window onto the plot area for each update, but also allows the plotted view to be shifted instead of wrapped. By shifting the data it is possible to scroll backwards through time or pause and resume (catchup) data plotting. The view can be scrolled by the same amount and interval as the drawing of new data blocks, or it can be scrolled ahead at longer intervals leaving some blank space for drawing so lowering the overall rendering overhead. The downside to this approach is an overall reduction in the smoothness which the plot progresses and a less consistent overhead which could result in glitching if there is a delay in the extra computational load being met at each scroll event.

The third approach was adopted for mseDyno due to its lower rendering overhead and ability to preserve the time history of the data collected. There is a menu selectable auto-scroll option (which is enabled by default) which when selected caused the display to advance by $2/3$ of the display area so that newly plotted data can always be seen as a continuation of previous data.

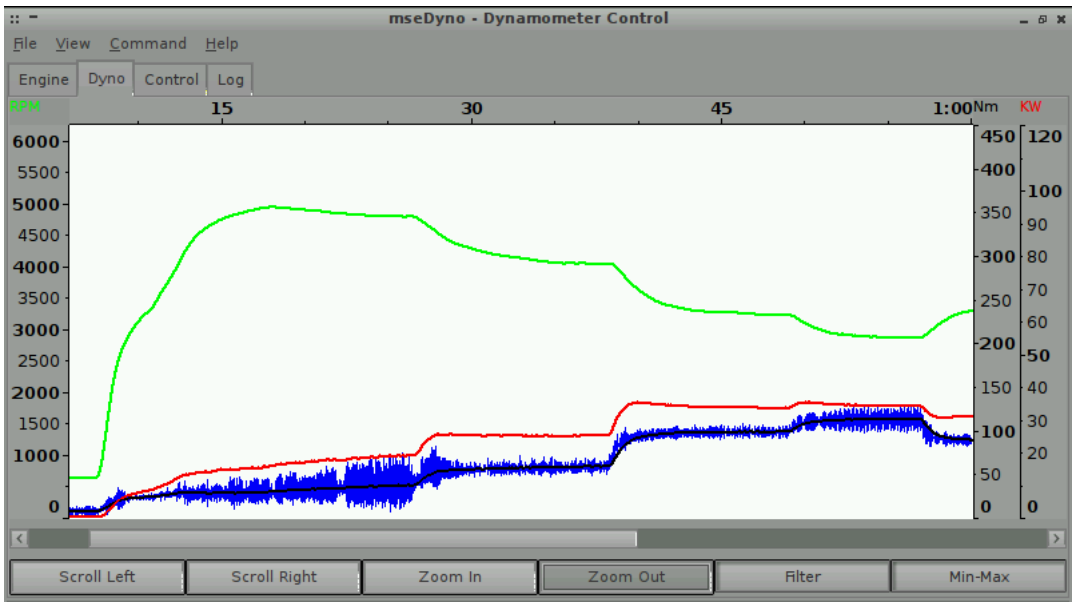

Figure 4.85: The mseDyno front end GUI showing the data plotter panel

The cylinder pressure panel can display cylinder pressure as it is acquired until the frame rate is exceeded by the ignition rate. It uses an embedded OpenGL/Mesa panel to render pressure traces which can be rendered in either 2D or 3D. The 2D view (Figure 4.86) marks the peak-pressure-position for the current and previous seven traces using the sample which has the largest pressure. This could be improved using a gradient search or a mass-burned-fraction technique (Eriksson, 1999), but it was considered better to keep the processing overhead as low as possible for online display. The 3D view (Figure 4.87) shows the current and previous nine pressure traces in an receding horizon waterfall so that brief trends in pressure can be observed by the operator. The axes can be removed in both views if desired. Data is transferred from a dedicated server program called mseEncoderMon. This program takes care of setting up the encoder interface and reading sample from the DAQ card, discarding non-combustion phase data, then packing the remaining data into a binary stream for transmission to mseDyno or other listening application using TCP/IP. This allows mseDyno or other custom data logging software to be located on a different computer/device away from the hardware if required. Data packing and serialisation is done using the GNU implementation of External Data Representation (XDR) library which standardises how data types such as floating point are transmitted between different kind of computer systems. This helps to prevent issues of endianess between the various embedded boards in use on the engine testbed.

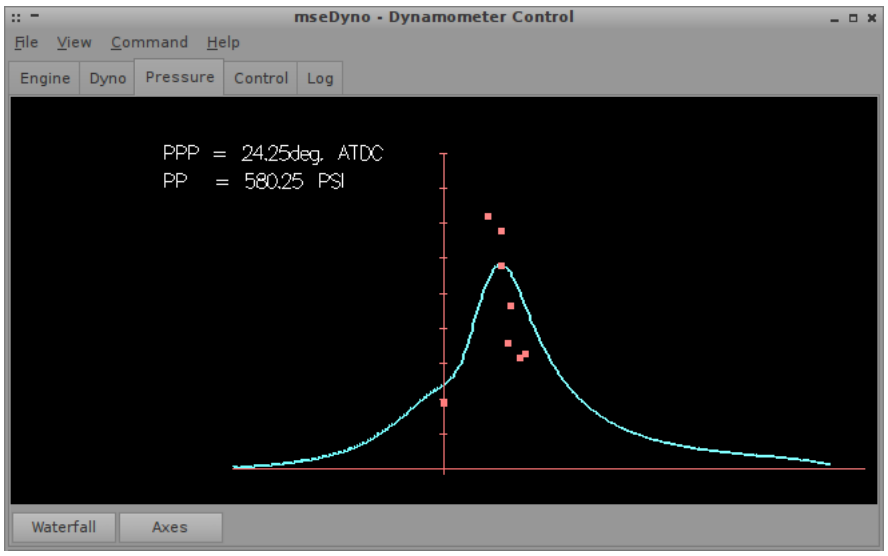

Figure 4.86: The mseDyno front end GUI showing the pressure plotter panel

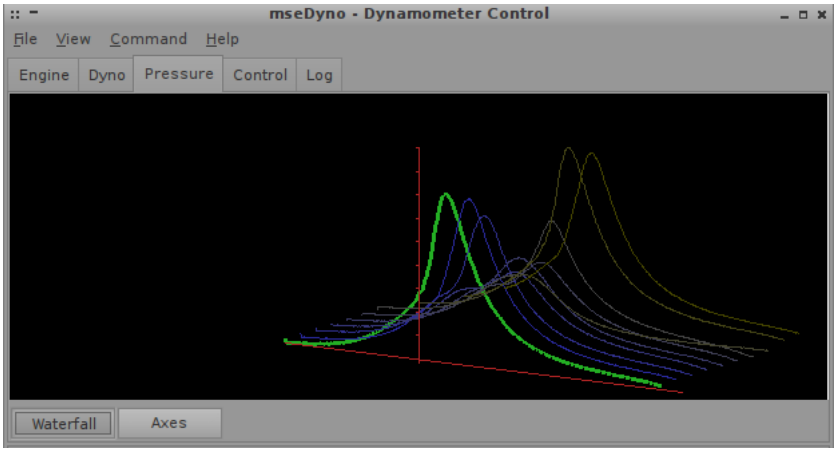

Figure 4.87: The mseDyno front end GUI showing the pressure plotter panel

Chapter 5. Development of an Engine Controller

This chapter describes the work undertaken to develop an engine controller for research purposes. The motivation for this work has been due to the absence of an existing controller which can allow the direct real-time access to inputs and outputs, high bandwidth data acquisition for model identification and the direct implementation of computationally intensive model based control algorithms.

For a university based engine research project to take place there is often a dependency on developing a relationship with a vehicle manufacturer to obtain an development ECU and engine to carry out a piece of research which is of mutual interest which may mean it must have some inherent short term commercial value to be viable. Aftermarket ECUs may at first seem to be an alternative, but these are usually aimed at enthusiasts or race applications and are supplied with static control software which can be calibrated using a proprietary piece of software to provide the desired fuelling and ignition timing, but have no mechanism to be dynamically overridden by an external source. An industry *de facto* standard method of ECU software prototyping is often to use the generic controller units from the dSPACE product range combined with Mathworks Simulink model based autocoding, and this approach has been cited for use in university engine related projects. However since these units (and others) are aimed towards so-called *Tier 1* customers and suppliers thereof, the cost is prohibitive and even if the hardware is obtained there is still an overhead to interfacing with an actual engine and producing the basic level of software to realise a running engine. Commercially available ECUs that are supplied ready for connect to an engine without additional interface hardware (except for wiring and connectors) and which allow end-user software to be written or Simulink model generated and deployed, examples are Ricardo's rCube or it successor Morfeus, and certain models from the Pi-Shurlok OpenECU range. However, the cost of these units and that of the supporting software required to achieve a running engine is considerable and outside that of research budgets. The requirements for meeting safety integrity levels (SIL) makes development versions of production ECU hardware expensive also, then the cost of a software calibration tool licence has to be met, and a Mathworks Simulink Embedded Coder licence if a model based approach is adopted.

It was decided to search for a PC/104 form-factor board containing an MPC555 processor commonly used by the automotive industry, to use as the basis for a research ECU. The original intention for this was to allow the I/O and communication capability to be extended beyond that of the MPC555 should that be required as the project's research aims developed. However, it quickly became clear that a greater benefit would be to allow the MPC555 to be coupled to an x86 board for data transfer and expansion of

a computationally intensive modelling and control technique on to a much faster (but less deterministic) x86 environment. Two candidate boards emerged from a search, the first was a PC/104 form-factor PPC/104-555 board made by Parvus Corporation, illustrated in Figure 5.1. This board hosts an ISA bus which allows peripheral boards to be added, but prevents it being directly connected to a x86 processor board since it would also be a bus host. The PPC/104-555 board was no longer available from stock or in active production for low volume purchases when an enquiry was made to Parvus.

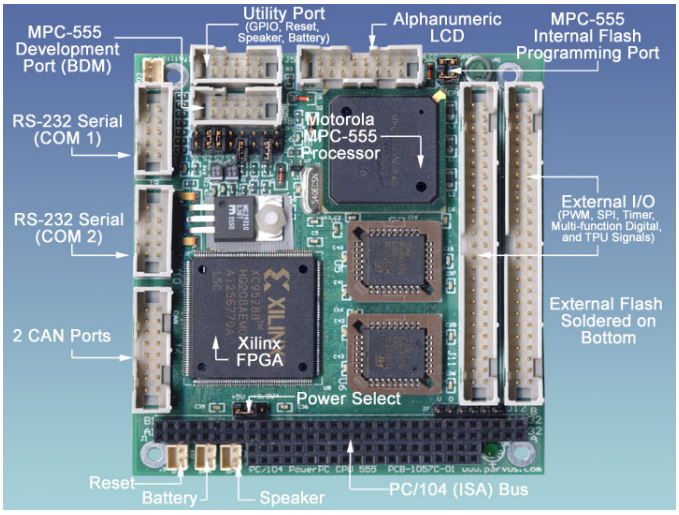

Figure 5.1: Parvus PPC/104-555 board

The second candidate board is the MPL PATI shown in Figure 5.2. This board is PC/104+ form-factor and is a PCI target (peripheral card) that can also be used standalone. It has an ISA Bus pass-through but there is no data connection to the Bus. It is able to access the PCI Bus for data exchange with a PCI host, normally an x86 based processor board. The PATI was selected for use with this project and used initially as standalone, but later with a Kontron MOPS x86 PC/104+ processor board which was a arbitrary choice of cost effective board with sufficient resources to support a PC variant of Linux. The PATI is supplied as a bare board with no software support (as none existed), so a board port (the process of writing/porting board specific supporting code) of an open source operating system called eCos was made as a basis for further software development.

Figure 5.2: MPL PATI PC/104+ board

To support the PATI board in it's role as an engine controller, a number of other interface circuit boards were designed and produced to allow it to interface to the engine. These boards were also made to comply with the PC/104 mechanical dimensions specification so that the controller could be assembled to form a PC/104 stack if desired. Since the controller hardware itself was under development and testing, the boards were mounted onto a chassis plate in an electrical enclosure for protection (Figure 5.3). The existing engine wiring loom was spliced with multi-pole connectors so that either the prototype ECU or the existing OEM ECU could be quickly reconnected to the engine at any time. The interface boards consist of two injector driver boards (for driving eight injectors), a digital I/O buffer board, and an analogue signal conditioning board (Figure 5.4).

The peak-and-hold injector driver boards are able to current control four fuel injectors. The digital I/O buffer board with a row of LEDs to give a visual indication outputs (to visually validate firing order for example), and an analogue signal board which provides active filtered signal conditioning for two Lambda sensors, two RTD sensors (thermistors), two hall effect sensors, scaled battery voltage, and general filtered signals inputs for other sensors such as the engine's MAF sensor. In addition to the interface boards, an independent throttle controller board with an opto-isolated CAN transceiver board was produced so that the electronic throttle unit fitted to the engine could be controlled independently from the main controller, accepting input commands from either the mechanical throttle cable, the engine controller (for idle control), or the dynamometer controller (for torque/speed control).

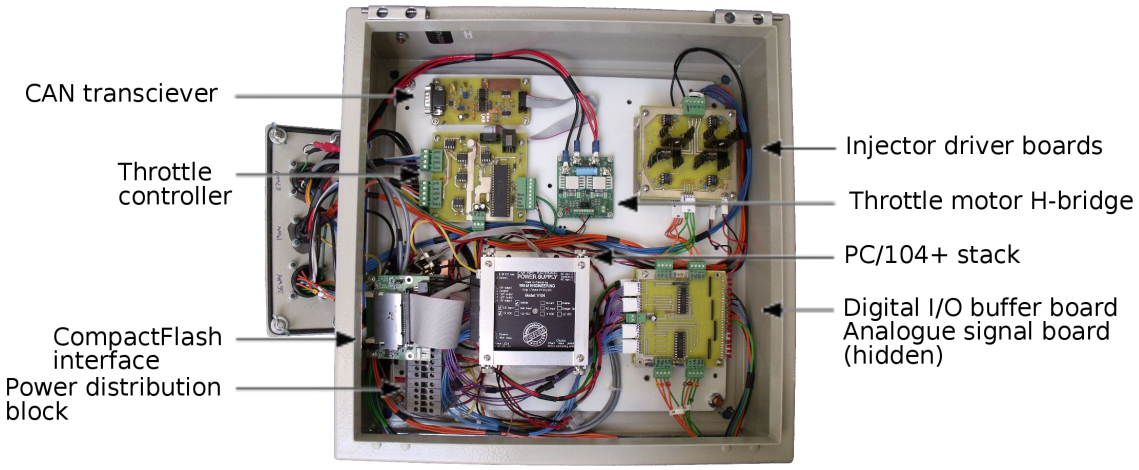

Figure 5.3: Inside the research ECU box

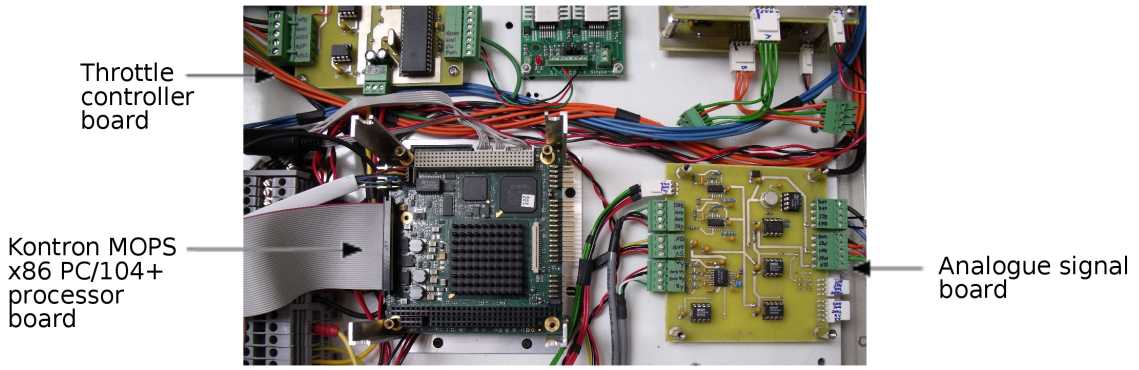

Figure 5.4: Inside the research ECU box showing the MOPS embedded PC and analogue signal board

The engine controller hardware was tested on the engine and each of the subsystems was found to function as intended, from throttle control, through to actuating the injectors and reading analogue signals. Unfortunately, time constraints and difficulty with the Motorola TPU microcode prevented the software from being developed to the point where the engine could be tested whilst running. The rest of this chapter describes the challenges in developing the software and hardware which has resulted in the prototype research ECU, beginning in the next section by looking at how the original OEM ECU for the engine is constructed.

5.1 Nippon Denso ECU

An AJ26-V8 engine was supplied by Jaguar with a development ECU (refered to in Jaguar terminology as an engine control module (ECM)) and software for use with an

undergraduate project that had taken place a number of years prior to the commencement of this one. The history of the engine and ECU are not known, but from an examination of the ECU it appears to have been a prototype for the unit which was eventually fitted to production vehicles. The arrangement of the ECU is slightly unusual as it has three PCBs and appears to be an adaptation of a two PCB design. The lower main PCB can be seen in Figure 5.5 and contains the main power input, electronics to drive actuators. The throttle H-bridge can be seen mounted on the side of the case and eight transistors mounted (without heatsinks) in the middle region of the board are evidently for driving the fuel injectors. There are also low ohm solid resistors for current sensing/monitoring. The lower board joints to a much smaller intermediate board via a set of three short board-to-board ribbon cables. This board contains what appears to be two locations for surface mount microcontrollers to mount and a Intel 82527 CAN controller IC. In place of the expected processors there are mounted instead socket breakout boards. Four hand spliced ribbon cables are then attached to these adaptor board and carry signals up to the final upper board. The top level board can be seen in Figure 5.6 and contains processors and EPROMS. Other than the EPROMS, the major components are not identifiable by part number also, but their roles can be assumed. The following devices have been identified:

- Two Motorola XC370600FU CPUs marked as a samples.
- Two Hitachi HN27C1024HGJ-90 1-Mbit (64-kword × 16-bit) EPROMs.
- Two 4.0 MHz clocked large devices possibly FPGAs or other programmable logic devices
- Two Xicor X28HC256 256 kbit 32 kB x 8-bit EEPROMs
- A Sound Design Technologies device marked DSP 7133 LA90GB, possibly an application specific integrated circuits (ASIC) .

No information about the Motorola CPU is available from the part number, but it is highly likely that it is a microcontroller variant of the 68000 series 16-bit CPU. Each CPU is accompanied by a large high pin count device which has their own pair of 8-bit EEPROMs. It is possible that these are FPGAs programmed to handle the low level timings tasks of the engine controller. The last device of interest has a manufacturer marking of SDT where the *S* formed using the integrand symbol. This can be assumed to be a Sound Design Technologies device for knock signal processing, either a custom ASIC or a Digital Signal Processor (DSP), although from it age, the former is more likely. There is also a 3-pin header which connects to a DE-9 connector mounted on the case for RS-232 serial development port that interfaces to the calibration software running on a PC.

Figure 5.5: Nippon Denso ECM, middle and lower PCBs visible

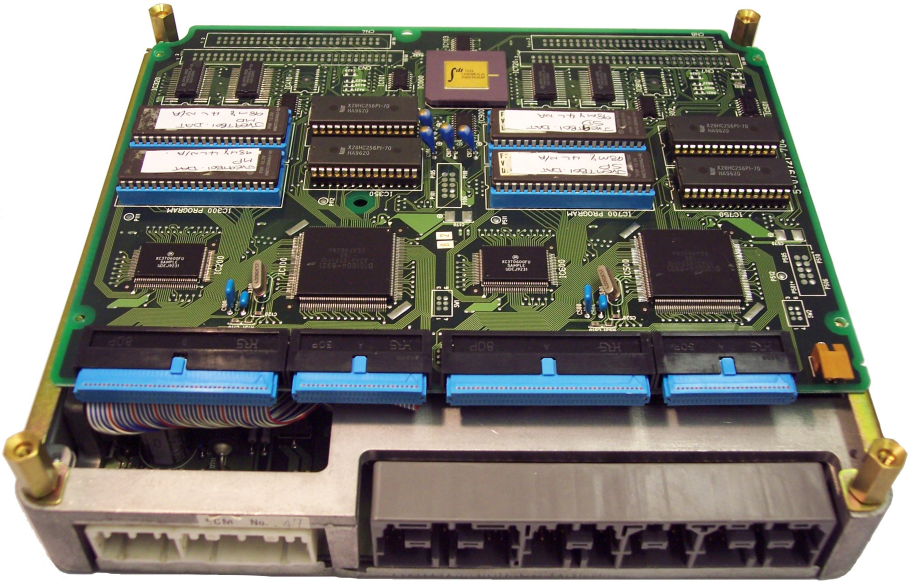

Figure 5.6: Nippon Denso ECM, upper PCB visible

The calibration software runs on MS-DOS and connects the the ECU via a serial cable or CAN, although it is not known which CAN hardware is supported. It allows 16 variables to be monitored and logged to a data file. It allows the many calibration tables to be viewed and edited either on the running engine or offline. The ECU firmware for the two CPUs is stored in Motorola s-record files an can be uploaded to the ECU. Figure 5.7

shows the software running on FreeDOS using the QEMU processor emulator under Linux. The DOS serial COM ports can be mapped to a USB to serial device by the emulator allowing the software to run on recent hardware. The screen shown is a fuel map of mass air flow normalised per engine revolution against fuel injector opening duration which ranges from around 450 µs to 10 ms.

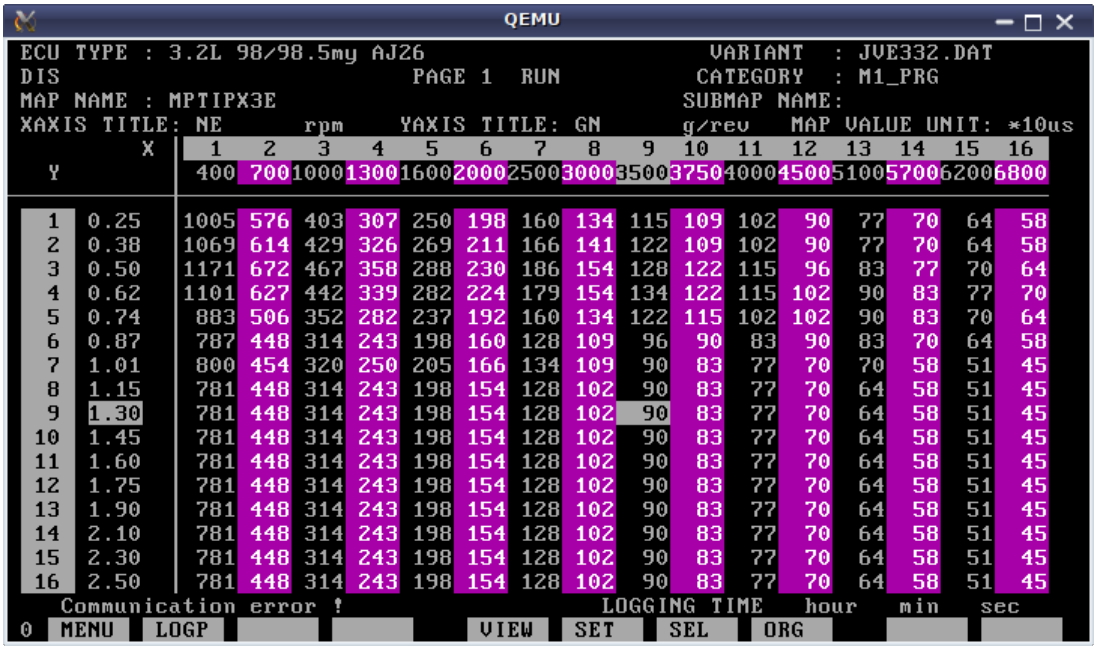

Figure 5.7: Nippon Denso calibration software for MS-DOS

The details of the communication protocol between the ECU and the PC software are not known. The pairing of ECU calibration software with the ECU of a particular manufacturer is something which has been addressed by a German interest group formed from Car manufacturers originally calling itself ASAP (A German acronym for Working Group For Standardising Calibration Systems). This group adopted a standard called CAN Calibration Protocol (CCP) which forms the subset ASAP1a of the standard called ASAP1. The standards work carried out by the group has since been renamed in December 1998 to form a new organisation called ASAM which is an acronym for Association for Standardisation of Automation and Measuring Systems. The association was founded as an initiative of German Car manufacturers. Despite its name, CCP does not mandate the use of a particular physical layer or bitrate so it could also be applied to a serial link. It is possible that Nippon Denso have used CCP for either the CAN and/or serial communication links to the ECU, or that a proprietary protocol has been used. The ASAM standards are not freely available, but Vector (2004) gives a comprehensive overview of the ASAP/ASAM history and the protocol level details of CCP. CCP is a relatively simple and effective protocol that can be used for uploading and flashing binary code to an ECU, setting/modifying calibrations and on-line monitoring of data variables and calibrations. It is not normally used alone. Application software which

uses CCP normally requires a database (known as an ASAP2 file) which contains a list of data variables and calibration tables along with information about the data items, as a minimum the name and memory address on the ECU where they are to be found, a brief description of their purpose and the offset, scale factor, and engineering units which the stored value represents. The ASAP2 file is normally generated by post processing the binary image output from from the compiler for symbolic information and comparing it with a data-dictionary which contains a formatted list of the data items with their corresponding descriptions and other information. Without the ASAP2 file, CCP is of little use since the significance of data stored at a particular memory address is not known. Figure 5.8 shows how the ASAP interfaces relate to each other. At the centre is the PC software based calibration tool. As as result of the adoption of the ASAP standards, the particular tool used can be chosen by the ECU calibrator rather than the ECU manufacturer which has seen the emergence of independent ECU calibration tools such as Vector CANape, ETAS INCA, and ATI Vision. With the ability to read ECU variables on-line as the control program is running, it is possible to perform processor-in-the-loop (PIL) tests by setting overrides for measured variables to contrive external conditions, or hardware-in-the-loop (HIL) tests using a dedicated test hardware (often an industrial PC with specialist I/O timing interface cards) which can connect to the application PC using the APAP3 RS-232 protocol or otherwise. Chen & Carpenter (2010) describe the use of feedback with HIL test hardware and ASAP3 to manipulate the internal state if the ECU software that cannot be directly reached by statically mimicking inputs to the ECU.

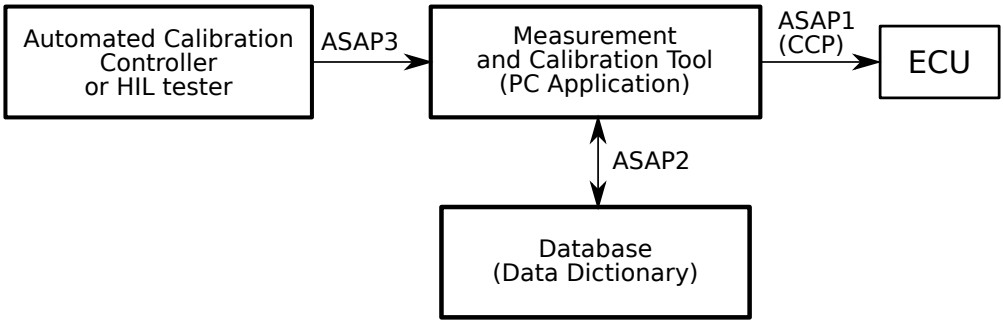

Figure 5.8: The relationship between the ASAP interfaces

5.2 The MPC555 Microcontroller

In brief the MPC555 microcontroller has the following features:

- Made from over 6.7 million transistors using 0.35 μm manufacturing process
- RISC PowerPC architecture with FPU @ 40 MHz, 52.7 kMIPS (Dhrystone 2.1)
- 8 differential 10-bit ADC channels, 25 kHz each

- 32 timing channels from two Time Processor Units (TPU3)
- Two Serial RS-232 interfaces
- Two CAN 2.0b (TouCAN) interfaces
- 448 kB Motorola one transistor (MoneT) CMF Flash, one cycle access (limited to 100 erase/write cycles)
- 26 kB SRAM, one clock cycle access for data
- 5 V general purpose I/O (GPIO)

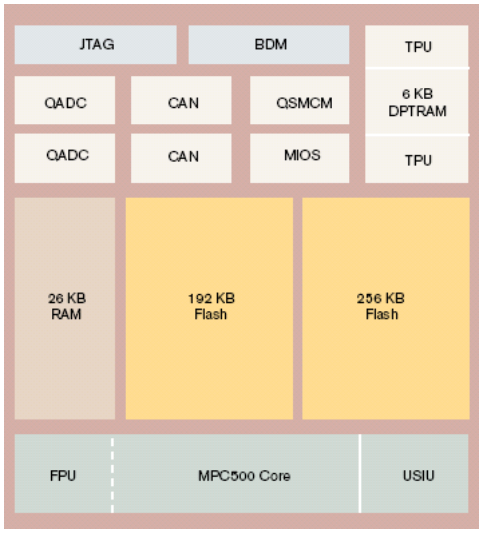

Figure 5.9: Block diagram of MPC555 internal features

Figure 5.9 shows the key components of the processor's architecture. The System Interface Unit (SIU) supports full burst mode for glueless (no arbitration logic required) connection external memories (such as SRAM and Flash). It has a 32-bit data bus and a 24-bit address bus (16 MB) with four chip select regions, each with extensive timing options. A 64-bit timebase is used and there is a built in real-time clock (RTC), watchdog and bus monitor protection circuits. Notably there is no memory management unit (MMU) and no cache. This arises from the intended application of the processor being for time critical applications. An MMU is required for general purpose applications such as a desktop PC where the user can attempt to run arbitrary code such as a word-processor, whereas *in the field* the MPC555 is likely to run only one well tested deterministic application which is treated as firmware. The need for code to run under memory protection becomes less of a issue in this case and the addition of an MMU would only serve to increase the cost and complexity of the chip unnecessarily. The processor does however implement a supervisor mode and 36 of its special-function registers (SFRs) can be protected from casual non-supervisor accesses. The lack of cache is due to the fact that the processor is not clocked fast enough to prevent direct

access to contemporary memory for program execution, and in general the addition of cache can add a variable latency which may be unacceptable for time critical control and instrumentation applications. There is also a small amount of (26 kB) of on-chip SRAM which is accessible in a single clock cycle if fast access memory is required from program data variable storage.

There are two dual queued analog-to-digital converter modules (QADC) which together multiplexed 32-channels with 10-bit resolution with an achievable 5 µs conversion time. Since the two ADC modules are independent the processor is capable of simultaneously sampling two analog inputs.

The Modular Input/Output Subsystem (MIOS) unit is provided for general purpose digital I/O. It features programmable counters, eight PWM sub-modules, and a 16-bit parallel port. A limitation of the PATI is that none of the MIOS channels are exported to external connectors and so unfortunately are not available for use.

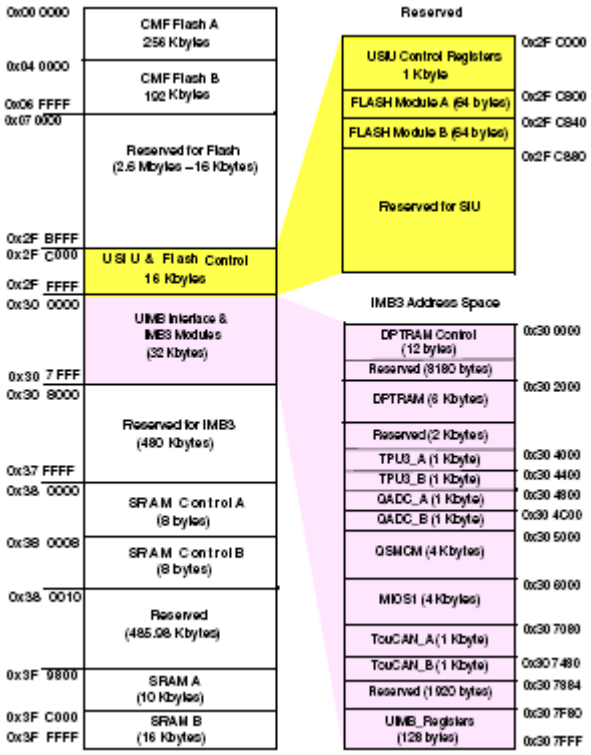

Figure 5.10: IMMR layout detail

All of the special function registers, internal memory, and other resources of the MPC555 are accessable from a 4MB region within its address space shown in Figure 5.10.

5.3 The Time Processor Unit

The time processor unit (TPU) is a semi-autonomous micro-controller designed for timing control. It operates simultaneously with the CPU, processing micro-instructions

to schedule and process real-time hardware events. It can access shared data without CPU intervention. This effectively decouples it from the main CPU and allows it to achieve very deterministic real-time performance. There is still a scheduler latency, but as it is a hardware scheduler this will always be the same and a worst case latency can be calculated for each channel. The deterministic nature means that it can be used for tasks which might conventionally only be practical to implement using dedicated hardware external to the CPU, or using dedicated processors that are unable to perform other tasks without inducing unwanted latency. The TPU can replace general CPU hardware IRQ events normally requiring ISRs. It essentially is able to perform or process a series of events that are either a *match* or a *capture*.

The TPU consists of two 16-bit time bases, 16 independent timer channels, a task scheduler, a micro-engine and a host interface for CPU access. Each of the 16 channels corresponds to a single dedicated digital I/O line that can be programmed as either an input or an output. The TPU channels are general and independent. No channel is specialised for a particular task so that any channel is capable of running any time function. Upon initialisation, the CPU can assign a time function and a priority to each channel. The priority gives more frequent service to high-priority channels and less to low-priority channels.

The MPC555 has two TPU3, third revision units which in total provides 32 channels. The TPU3 is identical to the TPU2 and TPU and it has some additional features including those added to the TPU2. The enhancements include up to four 2 kB for micro-code storage, eight words of parameter RAM, and an improved comparator. On the PATI all of the 32 channels are exported to an external connector and are therefore available for use as a part of the ECU.

A detailed description of what the TPU is and how it works and is programmed is given in Dyson & Bannoura (1999). Motorola previously supplied information on programming in TPU microcode, in the Time Processor Unit Reference Manual TPURM/AD (Motorola, 1990). Unfortunately, in later editions (labelled TPURM/AD-xx), the microcode information was removed, as it caused some confusion to application developers who just wanted to use the standard functions supplied by Motorola, making that process seem much more complex than it actually is. The updated simplified version of this document is now provided by Freescale (1996). A summary of the key information from these references is provided in Appendix B.5.

5.4 The PATI Platform

PATI is an acronym for **P**owerPC controlled **A**nalog and **T**imer-I/O **I**ntelligence board (Figure 5.11) and is made by MPL AG Electronik of Switzerland (Bea, 2004). It is a

PC/104+ form factor card implemented as a PCI slave device. It has no ISA connectivity, although a pass-through connector is provided. As it's name implies, it is intended to function as an active, or intelligent I/O and data acquisition card to a PCI master device such as an x86 processor, but despite this it is capable of functioning as a standalone board. It is based upon the Motorola/Freescale MPC555 processor which is has been commonly used in the automotive industry, and is also suitable for robotics applications. Unlike more conventional data acquisition cards, nearly all of the board's functionality comes directly from the MPC555 itself. In addition to the processor there are 16 MB of SDRAM, 4 MB of programmable flash memory, a PCI-bridge chip and an erasable programmable logic device (EPLD) to facilitate the glue logic and power-up cold boot configuration of the processor.

The card was designed to exploit the extensive peripherals that surround the MPC555 processor's PowerPC core. The card was selected for this project, not for use as an acquisition card in the strictest sense, but to make direct use of the processor for engine control with the potential for future expansion through the PCI-bus. The choice of a card which has an MPC555 processor was a natural one given that this processor is used in some production ECUs. This makes it easier to have prior confidence that the platform is both capable of, and suitable for, controlling an engine. It will also permit a reasonable assessment to be make of any *new* control algorithm's suitability for implementation on existing ECU hardware which has only modest computational speed when compared to a desktop PC for example. There is also a certain amount of commercial support for development on this architecture for control applications such as Mathworks Embedded Coder (formerly Real-Time Workshop for Simulink). Further detailed information on the MPC555 Peripherals is provided in Appendix B.5.

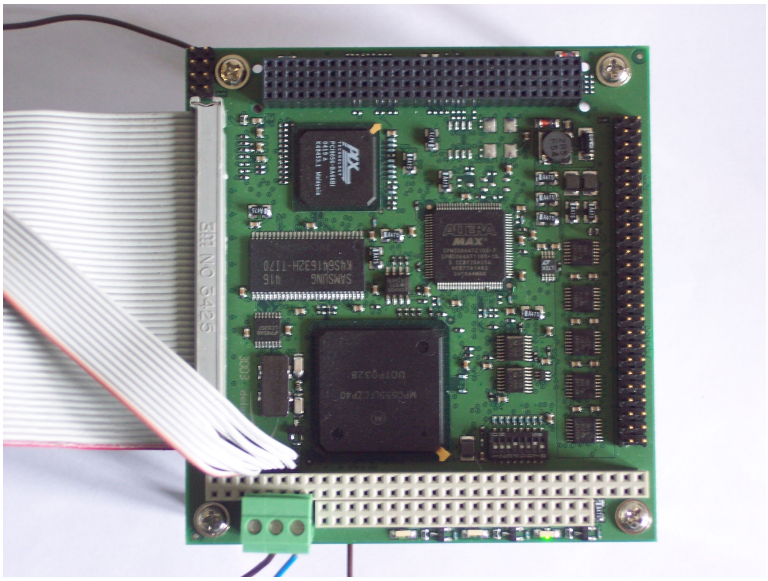

Figure 5.11: MPL AG PATI PC/104+ card

5.5 Porting eCos to PATI

The Embedded Configurable Operating System (eCos) was chosen as the operating system to run on the PATI board. The reasons for this are that it is real-time capable and completely open source with a small memory footprint. It is also highly configurable. An MPC555 varient was already in existence and early startup code specific to the PATI was needed to assign memory mappings and configure the external SDRAM refresh. The equivalent U-Boot code was used as a reference, but since this memory setup operation has to be performed before a C stack has been created, this code had to be written in assembly language rather than C as used for the U-Boot code. Additionally, board specific reset code was added and the two existing serial drivers for other MPC555 based boards were consolidated into a single generic MPC555 serial driver. A board specific driver was written for the external Intel Flash, which required only slight addition to the generic Intel Flash driver to enable the write/erase voltage to the flash by writing to a specific register in the EPLD. These changes were then committed back to the main eCos CVS repository as a board port. Further information on eCos and real-time issues are provided in Appendix B.6.

5.6 ECU Hardware Development

5.6.1 Fuel Injector Peak-and-Hold *Driver Circuitry*

The current required to overcome the kinetic and constriction force to open a solenoid is several times greater than the current needed to hold it open. The LM1949 IC can be used to control the current through the injector using an external NPN Darlington transistor, as shown in Figure 5.12. By directly measuring the current through the injector it can initially saturate the driver until the peak current reaches four times the holding current to guarantee that the injector opens. The current is then reduced to a sufficient level to keep the injector open for the duration of the input pulse. This technique reduces the amount of power dissipated in the system as well as improving the correspondence of input pulse duration to fuel quantity by reducing the closing time of the injector as the current only has to fall from a reduced holding level. Fuel injectors can be modelled by a simple *RL* circuit (National-Semiconductor, 1995). In operation the value of inductance L will change whether the solenoid is open or closed. The peak current is determined by the value of the sense resistor R_s. When the voltage drop across the sense resistor reaches the peak threshold level (nominally 385 mV) the IC is tripped from the peak state into the hold state. The IC will then attempt to reduce the injector current to one quarter by maintaining 94mV across the sense resistor in a closed loop.

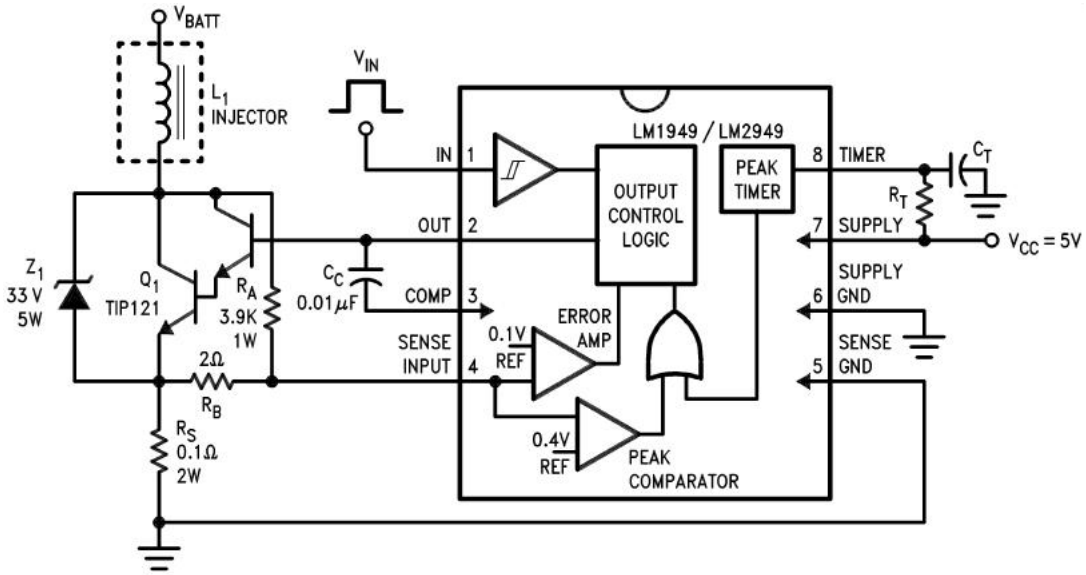

Figure 5.12: Injector drive controller typical application circuit (National-Semiconductor, 1995)

Since the injector is an inductive load, a change in current controlled by Q_1 will result in a voltage spike. This occurs at the peak-to-hold transition and at the end of each input pulse. A fly-back zener diode is needed to provide a current path which prevents the voltage level exceeding the breakdown voltage of the Darlington driver. A zener is preferred to a clamp diode as automotive systems are prone to voltage transients on the battery line and a zener can provide protection to the Darlington during short over voltage conditions.

A timer function constrains the time that the current is allowed to rise to the peak level. For example, if the battery voltage is too low, the peak current may never be achieved, but a slightly lower current may persist whilst the circuit waits for the peak level to be reached. The timer function provides a timeout so that if the peak current is not reached in this period, then the current will be reduced to the hold level regardless. Without the provision of the timer circuit, an injector current of just less than the peak level may be held for the duration of the input pulse which may overheat the injector. The time constant for the timer is set by choosing the values of R_T and C_T such that $\tau = R_T C_T$. A time constant in the order of 4 ms is typical, but should be chosen to suit the particular injector to ensure that it is given sufficient time to open.

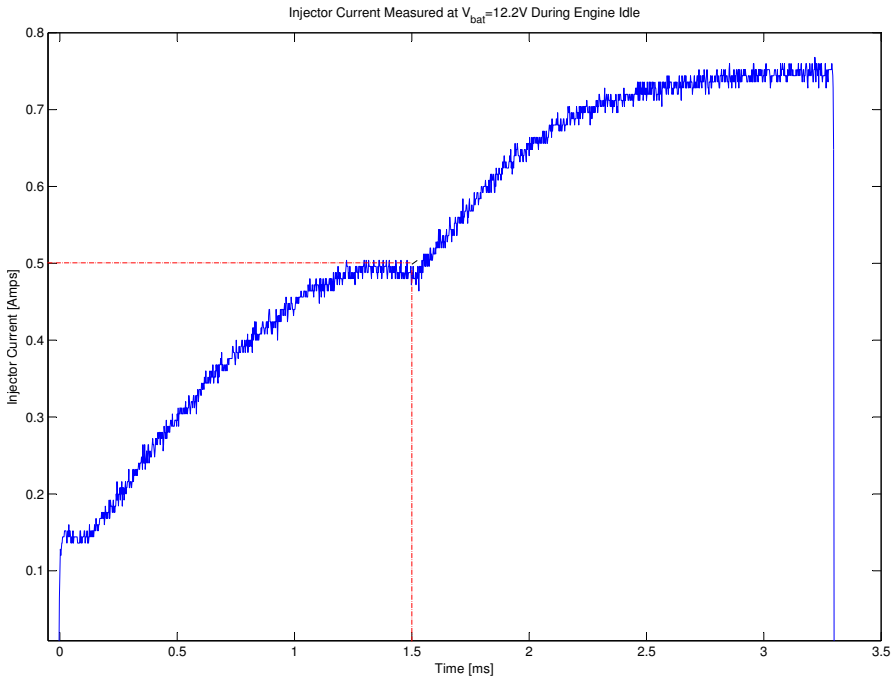

Figure 5.13: Fuel injector current response showing a *knee* point as the solenoid opens

Another design consideration is power dissipation in the driver, zener, and sense resistor. Generalisation possible using a PWM and external comparator technique with additional TPU channel to detect when the peak current has been reached.

5.6.2 Ignition Driver Circuitry

The AJ-V8 uses a pair of self-contained ignition driver modules, of the type shown in Figure 5.14. This removes the need to design a set ignition drivers since they are not integral to the existing ECU unit. However some consideration of their design is still necessary.

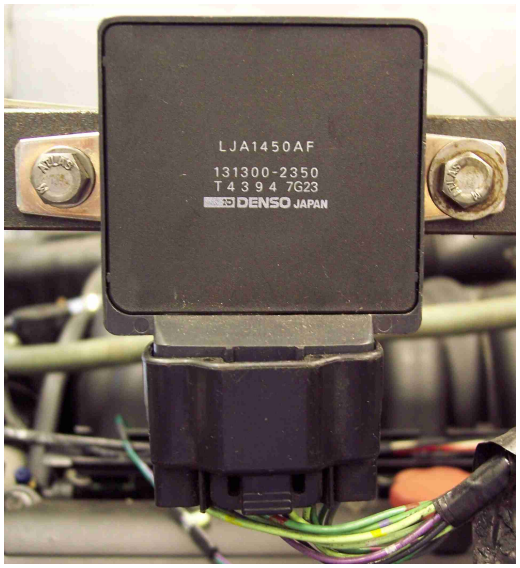

Figure 5.14: Denso ignition driver unit

Generally, ignition driver stages feature either multi-stage Darlington power transistors or IGBTs which control the flow of primary current through the coil, replacing contact breaker points found on earlier ignition systems. The ignition output stage also has to limit both the primary current and the primary voltage. The primary voltage is restricted to prevent excessive increases in the supply of the secondary voltage which could damage components in the high-tension circuit. Restrictions on primary current hold the ignition system's energy output to a specified level. Power output stages can be internal or external to an ECU.

When the firing point arrives, the ignition driver stage interrupts the flow of current. The flux in the magnetic field induces a voltage in the coil's secondary winding. The potential of the secondary voltage depends upon a number of factors which include the amount of energy stored in the ignition system, the capacity of the windings and the coil's turns-ratio as well as secondary load and the restrictions on primary voltage are determined by the ignition system's driver stage. There are essentially three types of ignition system that have been in common use. The first uses a single oil-filled ignition coil and directs the spark potential using a mechanical rotor arm and distributor. The second type removes the need for any moving parts by using (for a four-cylinder engine) two coils which each have a spark plug attached to either end. This type is sometimes called *wasted-spark* ignition as the spark potential is applied to two spark plugs at once. For this type, cylinders are chosen in pairs so that when one requires ignition, the other will be in a late exhaust phase. This places some restriction on the permissible spark timing so as to not unintentionally reignite products of combustion or freshly induced mixture. Some sources also state that a spark will only occur at one of the two plugs due to the higher impedance presented by lower pressure in one cylinder than the higher pressure and *wetter* more conductive conditions in the other. The third type is the so-called *coil-on-plug* type which has emerged from better driver control circuitry and epoxy enclosed *dry* insulation which together make it possible to reduce the unit size enough to make them practical. This third type is the only one which requires a knowledge of the engine's valve phase (using a camshaft sensor) to know which cylinder to fire. With the first type this is taken care of by the mechanical distributor, and the second it is unimportant as two plugs are always fired simultaneously. If spark timing is to be controlled on an individual cylinder basis (such as for knock prevention) then a camshaft sensor is still required.

When the primary current is switched, an undesired voltage of about 1-2 KV is induced in the secondary winding (termed switch on voltage) the polarity of which is the opposite of that of the intended high-tension spark-producing voltage. It is essential that arcing at the spark plug (termed switch-on spark) is avoided as this could ignite the mixture at completely the wrong stage in the cycle causing backfire or mechanical damage. Systems with a rotating spark distribution system use a distributor spark discharge gap

for effective suppression of this phenomenon. On systems with single-spark coils a special diode (in reality a stack of diodes to achieve the required voltage rating) is incorporated in the high-voltage circuit to perform the same function. With stationary and two spark coil systems the high voltage needed to create arcing when two spark plugs are connected in series suppresses the switch-on spark, without additional measures being necessary.

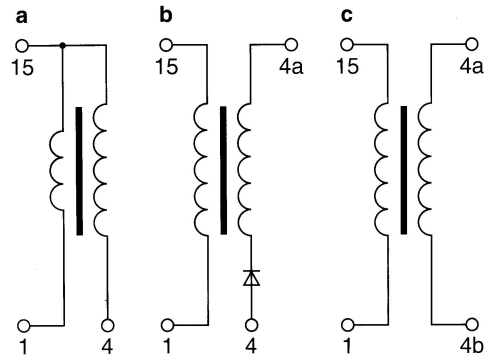

Figure 5.15: The three main types of ignition coil (Steinbrenner et al., 1994). a) Conventional single spark coil used with a distributor. b) single spark coil (coil-on-plug) c) dual spark coil (wasted spark)

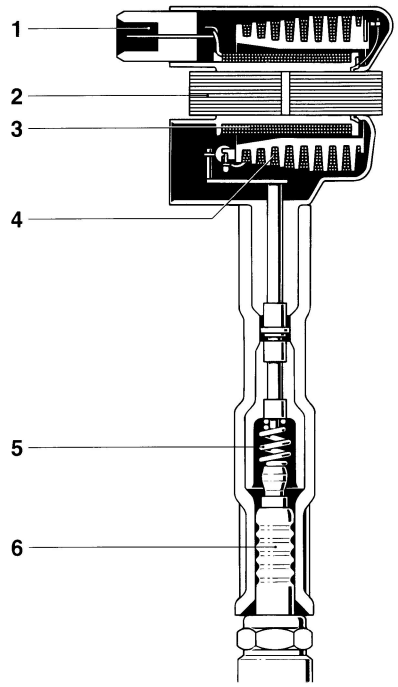

Figure 5.16: Coil-on-plug assembly (Steinbrenner et al., 1994), 1) LV terminal 2) Multiplate iron core 3) Primary winding 4) Secondary winding 5) HV terminal 6) Spark plug

Various ignition coil driver IGBTs targeted specifically at coil-on-plug designs have emerged on the market which are able to limit the stress applied to the ignition coil and provide over-voltage protection using an active voltage clamp between the collector and

the gate (Figure 5.17). The voltage clamping is termed self-clamped inductive switching (SCIS), and is a way of controlling the spark energy for consistent ignition by clamping to typically 400V for a total discharge of around 200-500 mJ.

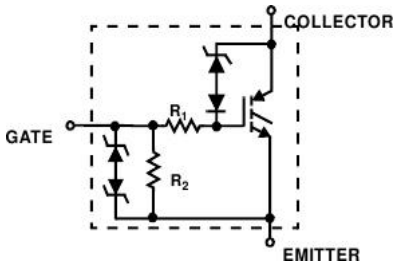

Figure 5.17: N-Channel ignition IGBT (Fairchild-Semiconductor, 2005)

5.6.3 Engine Speed and Phase Sensors

The AJ-V8 engine is fitted with two variable reluctance sensors to detect its angular position and valve phase. The engine's position is detected using a timing disc that is embedded into the flywheel. The relative change in this position is used by the ECU to get a direct estimate of the instantaneous speed. Using a disc on the flywheel is an alternative to using either a dedicated toothed disc or suitably modified crankshaft pulley with teeth at the other end of the crankshaft that is often used on other engines. Slots in the flywheel (Figure 5.18) give 36 timing positions from the webs left in between adjacent slots. Two of the webs are left out to provide an index which is detected when the gap in the generated pulse-train is twice as large as the previous and subsequent periods.

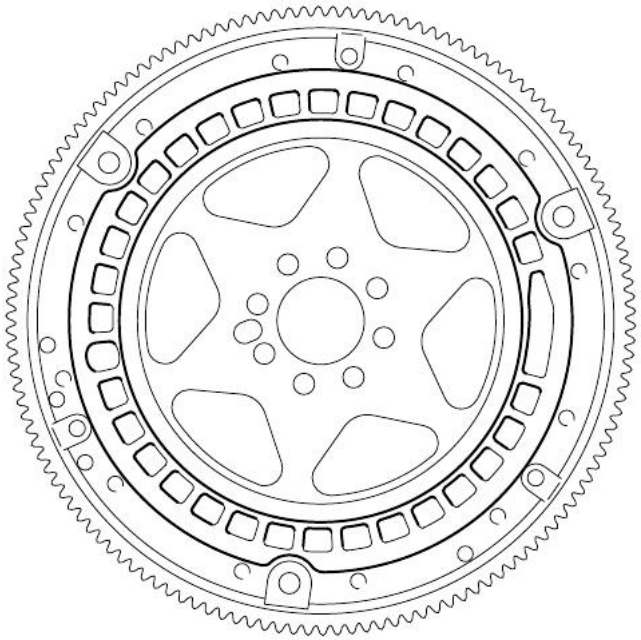

Figure 5.18: AJ-V8 flywheel embedded timing disc (Jaguar, 1996)

One of the four camshafts has a timing ring attached (Figure 5.19) with a key which triggers the second variable reluctance sensor once per camshaft revolution (or every two crankshaft revolutions). An effect of this is that during starting of the engine, it may need to be cranked for at least two revolutions before the ECU can determine enough information about the engine's position and phase to be able to trigger the ignition.

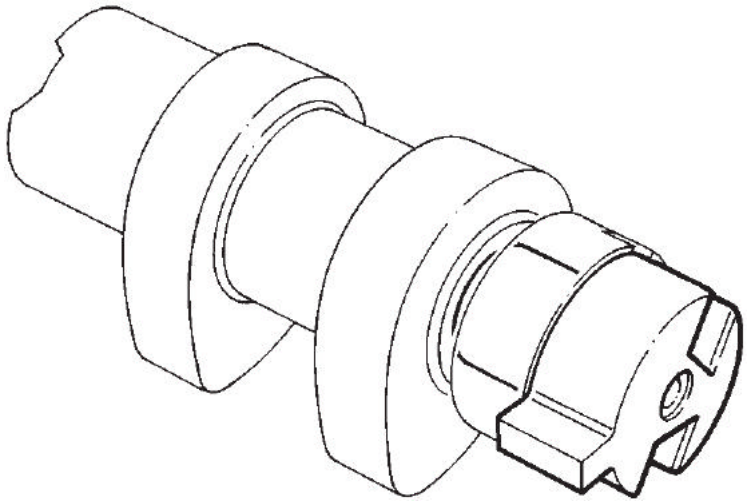

Figure 5.19: Camshaft timing ring (Jaguar, 1996)

The engine speed sensor has an output signal which is sinusoidal in appearance. Figure 5.20 shows the signal close-up and also at a resolution which shows a complete engine revolution. The 34 cycles with a gap of two cycles can be seen.

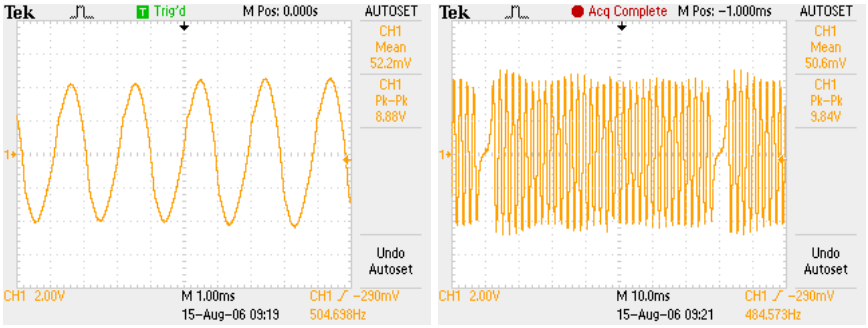

Figure 5.20: Engine speed sensor outputs captured using an oscilloscope

The output from the camshaft sensor is similar, as can be observed in Figure 5.21.

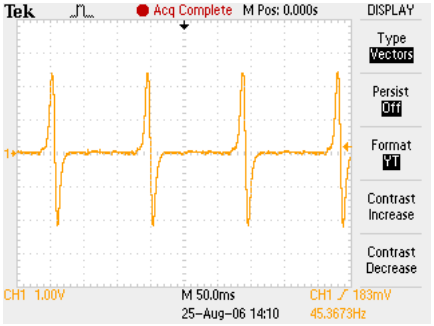

Figure 5.21: Camshaft variable reluctance sensor output

The varying voltage amplitude pulses produced by the variable reluctance sensors need to be conditioned into discrete logic level waveforms before they can be used by the engine management system. For this an adaptive variable reluctance sense amplifier IC LM1815 (National Semiconductor) has been used been used for the research ECU. The basic application circuit is shown in Figure 5.22.

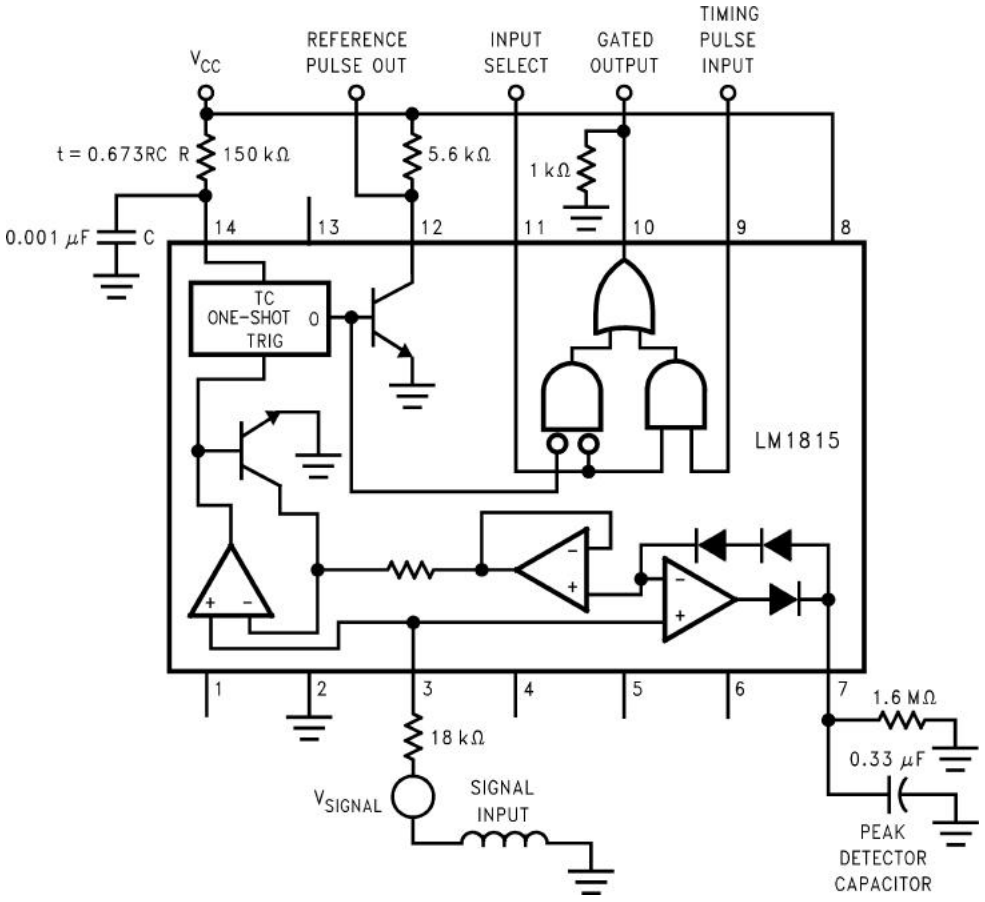

Figure 5.22: Adaptive sense amplifier typical application circuit (National-Semiconductor, 2005)

The LM1851 IC is more sophisticated than a rudimentary comparator wave-shaping circuit than might be considered for this signal conditioning application. The circuit is able to sense in situations where there is high frequency noise is greater than the low

frequency signal amplitude. This is achieved using input hysteresis which varies with input signal amplitude. A positive-going threshold is derived by peak detecting the incoming signal and dividing this down. The sense amplifier gives a one-shot pulse output whose leading edge coincides with the negative-going zero crossing of the input signal as shown in Figure 5.23. The LM1815 acts as a zero-crossing detector which cannot be triggered until the input signal has crossed an arming threshold on the positive-going part of the waveform. Subsequent zero-crossings are ignored until the arming threshold is exceeded again. This makes the circuit far more robust to false triggering than a simple comparator based design.

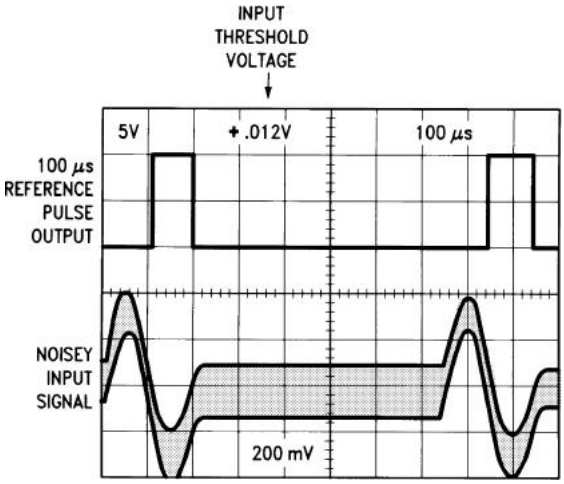

Figure 5.23: LM1815 output triggered from a low amplitude input signal

There are some design considerations when using the LM1815. The input signal voltage is internally clamped. An external resistor has to be chosen to limit the current to ±3 mA based upon the maximum peak voltage that is expected to be generated by the variable reluctance sensor. The other consideration is the duration of the one-shot pulse which will limit the maximum input frequency for reliable operation. The pulse needs to be wide enough to be noticed by the ECU, but narrow enough that consecutive pulses do no merge. The pulse duration t is determined by the values of R and C in Figure 5.22 which is given by the relationship:

$$t = 0.673RC$$

where the maximum usable input frequency is:

$$F_{in}\,(max) = \frac{1}{1.346RC}$$

The TPU input detection threshold can be programmed in terms of the number of clock cycles a signal transition must be held before it is considered a valid logic state. Since

this allows for some flexibility in the design, then the datasheet typical values (shown in Figure 5.22) of 150 kΩ and 1 nF were used giving a pulse of about 101μs duration.

5.6.4 Lambda Sensor Signal Conditioning Circuitry

The LM9040 IC made by National Semiconductor has been employed to condition the signals from the lambda oxygen sensors. Figure 5.24 shows the typical connection to the sensors. For use with the AJ-V8, the main implementation exception is that a dedicated signal ground path is not available as part of the standard wiring loom and so a common battery ground has to be used instead. The signal itself is supplied through a screened single cable which is grounded at the ECU end, rather than the twisted pair cable grounded to the sensor body and again at the signal conditioning end.

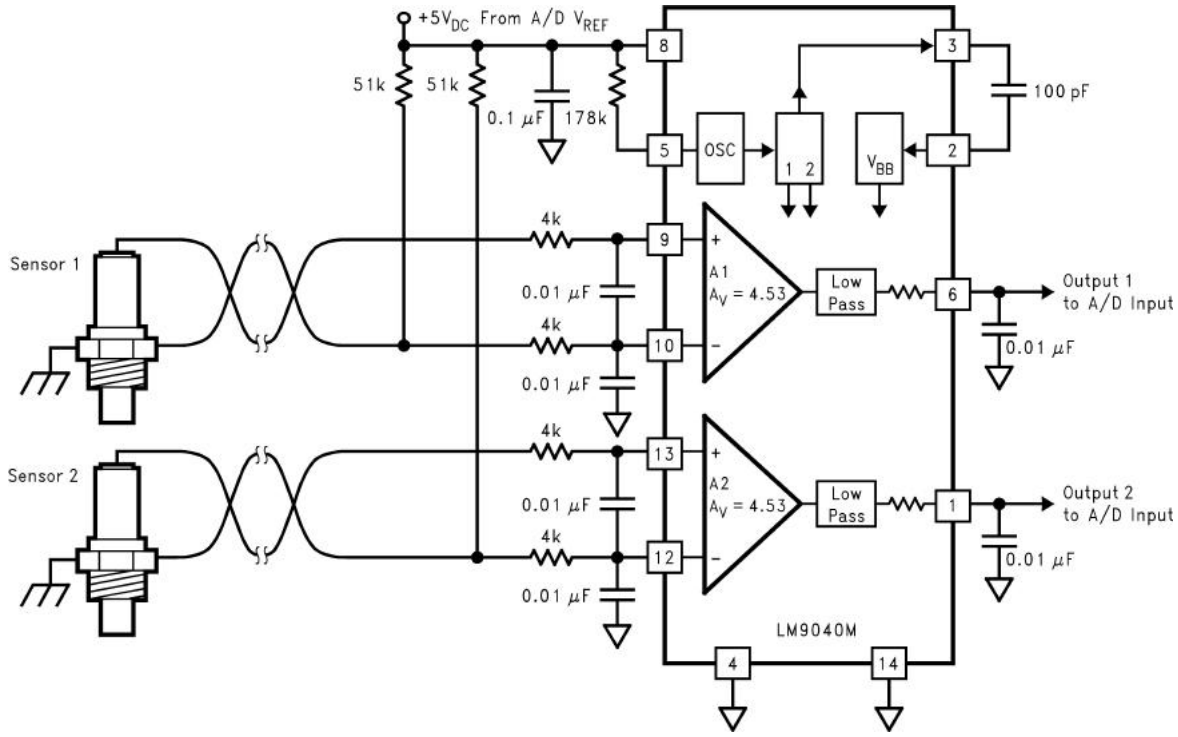

Figure 5.24: Lambda sensor interface amplifier typical application circuit (National-Semiconductor, 2001)

5.6.5 Knock Detection Circuitry

Combustion *knock* is a phenomenon which occurs when freshly induced mixture pre-ignites in spontaneous combustion before being reached by the expanding flame front. It often produces a characteristic audible *knocking* or *pinging* noise caused by local pressure increases which is how it has become known as knock. Shock waves with velocities in excess of 2000 m/s can occur (compared to 30 m/s for normal combustion)

(Steinbrenner et al., 1994) which can lead to mechanical damage and therefore knock is to be avoided.

Optimum spark timing is knock-limited for much of an SI engine's operating range. Historically conservative ignition timing was used to prevent knock, allowing for worse case conditions and factors related to engine ageing such as carbon buildup (which produces a slight increase in compression ratio and the creation of local hot-spots) that can reduce knock margins. Modern engines (and older turbocharged engines) employ a knock detection system that allows a near-to-optimum timing at all times. A piezo sensor is used to detect structure borne acoustic noise generated by knock and is usually measured on either the cylinder block or cylinder head. The optimum number and location of sensors is somewhat engine dependent. The general rule is that if it is desired to determine which cylinder caused the last knock event then a single sensor is needed for four cylinder engines but at least two are needed for *V* configuration engines due to mechanical separation and eight cylinder (and over) engines which also need two or more due to over-lapping combustion timing regardless of whether they are in an in-line or *V* configuration. By continuously running an engine on the verge of knock, the best efficiency is maintained in the presence of varying fuel properties (grade and quality) and other influences such as mixture and engine temperature variations. For example, engines are more prone to knock when they are fully warmed up (to their designed normal operating temperature) or are hot. When a hot engine (due to prolonged load conditions such as towing or hard acceleration) is combined with a hot weather day then there may be a propensity for knock to occur. Using knock detection, knock can be avoided under these adverse conditions without compromising efficiency and performance during more favourable conditions, which may exist for the majority of the time.

Bosch produce a knock sensor signal conditioning ASIC that is likely to have been used in their *Motronic* range of OEM ECUs.

F Frequency divider, **G** Rectifier, **L** Filter, I Integrator, **O** Oscillator, **P** Multiplexer, R Reference signals, **S** Sensor inputs, **T** Test-pulse divider, **V** Amplifier, **OUT** Output.

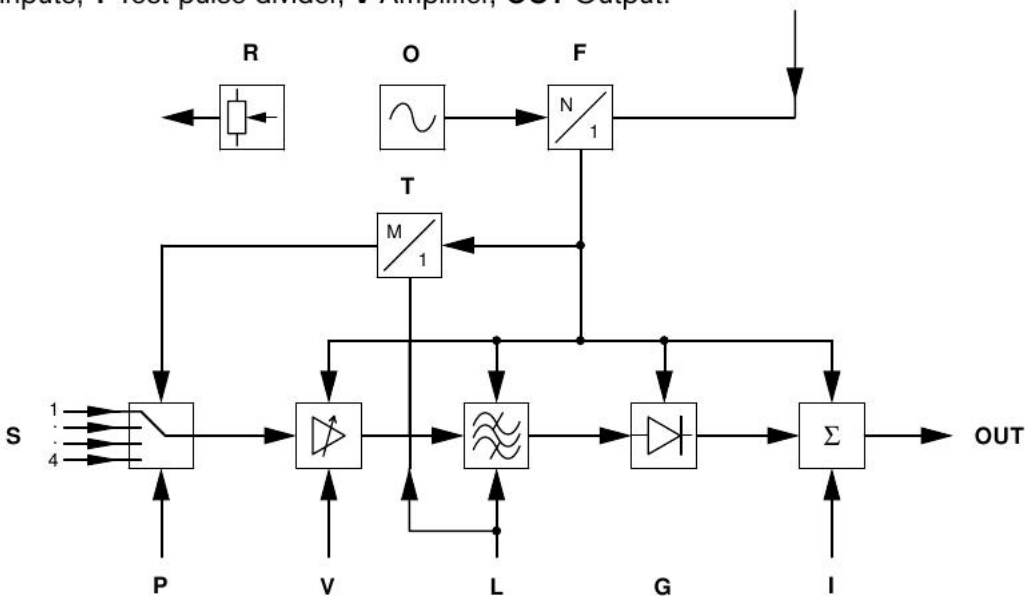

Figure 5.25: Bosch CC195 knock signal conditioner internal schematic

K Signal-integral output, **P** From microcomputer port driver, **S** Sensors, **T** Quartz clock, C_1/C_2 Capacitors as near as possible to housing pins.

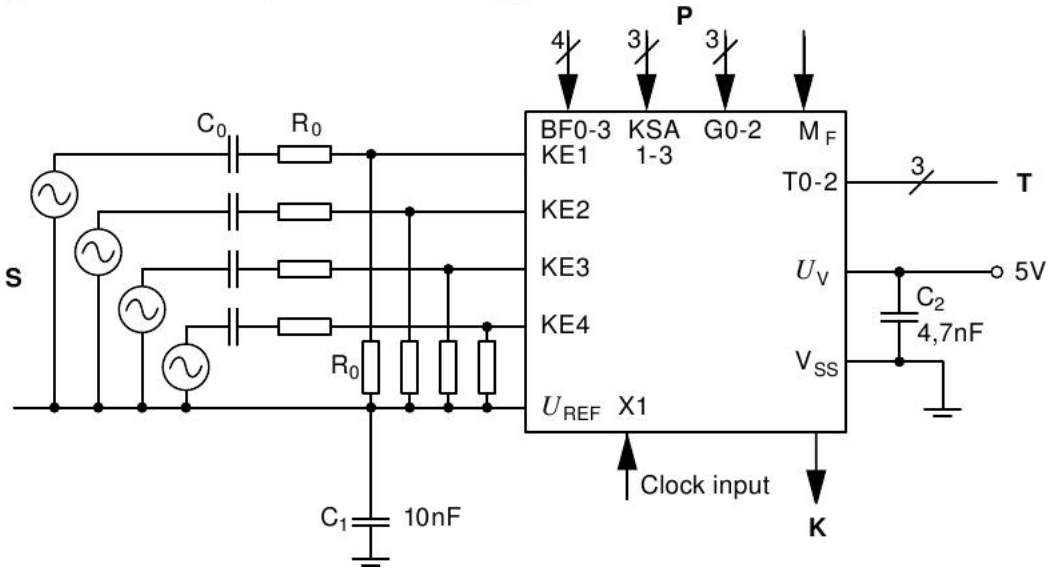

Figure 5.26: Bosch CC195 knock signal conditioner internal schematic

A US patent filed by a General Motors (Hernandez, 2007) describes how the functionality of ASIC based knock detection can be performed by a digital signal processor (DSP) microprocessor. Freescale have integrated features into the MPC5500 family of microprocessors (which have now superseded the MPC500 series) to help reduce the need for an external signal conditioning IC (Freescale, 2007), such as the CC195. There is a differential interface to an on-chip ADC. The ADC can sample at 800 kHz so that it has sufficient bandwidth to sample from one or more knock sensors at an equivalent rate of 150 - 200 kHz and still have time to sample the other analogue

inputs of the system. The time processor unit (TPU) can be setup to automatically start and stop (gate) the ADC to acquire samples at engine crankshaft angle intervals at which knock is likely to occur. There is also a memory controller which can autonomously move the ADC results into RAM and sort the data from different sensors into different buffers. One buffer can be filled whist another is being processed by the microprocessor. A DSP engine is included capable of 200 million multiplies per second which using multiply-accumulate (MAC) instructions can implement a finite impulse response (FIR) or infinite impulse response (IIR) for bandpass filtering and fast Fourier transform (FFT) for frequency spectrum analysis. Using the microcontroller to directly acquire and process knock signals keeps the component count down conserving board area, but more importantly than the slight cost saving, allows greater flexibility in the knock detection strategy over a fixed hardware design. Handling knock detection in software rather than hardware may make more complex approaches such as that described by Borg et al. (2005) more practically realisable to improve the detection quality over the conventional technique which uses bandpass filtering combined with amplitude threshold level measurement of the nominally 15 kHz knock signature. A university final year project by Rajagopalan (2006) describes the use of Motorola's ProSAK IC (which is another knock detection ASIC), but extends the application to a software statistical characterisation approach which lead to a US patent (Naber & Rajagopalan, 2008) being granted.

The two types of digital filter that are used for knock detection are the finite impulse response (FIR) and the infinite impulse response (IIR), and are briefly summarised as follows:

FIR

- Works without feedback (previous outputs)
- Inherently stable
- Requires data ahead of time so induces a time delay
- Can be designed directly from waveform data (bird song, knock signature etc)
- Design can produce many coefficients (>100 typically)
- Can be updated dynamically
- Inherently stable
- Can be made to have no phase distortion

IIR

- Uses feedback
- Requires only a few coefficients (typically 3 input and 3 output)
- Can be unstable (due to feedback) particularly if coefficients suffer from rounding errors
- Can model conventional analogue filter topologies well
- Cannot be designed directly from the frequency domain, needs a trial-and-error or brute force approach
- Cannot be updated dynamically due to iterative design method

IIR is suited to relatively simple filter requirements such as high or low pass, whereas FIR can be used for more complicated designs. Figure 5.27 shows an example two *tap* IIR band-pass filter which could be used for knock detection.

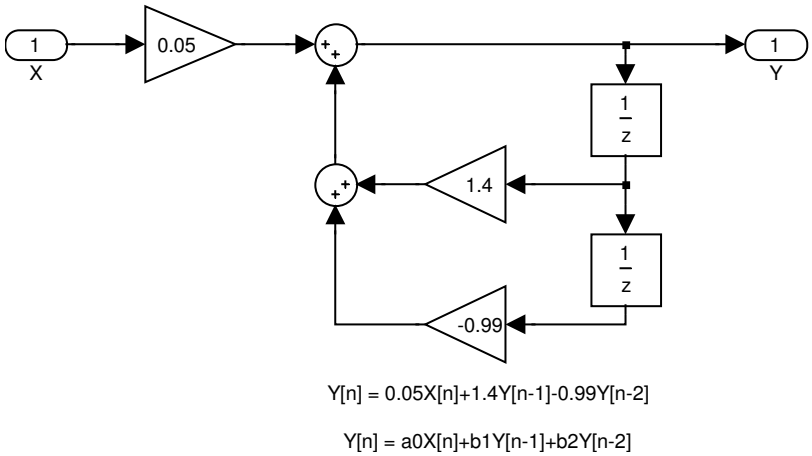

Figure 5.27: IIR Band-pass Simulink representation

The filter can be placed before a knock detection integrator and a threshold used to indicate a knock condition. Figure 5.28 shows how a knock detection algorithm might be formulated in hardware and software. The knock signal is represented by a chirp passing through the frequency at which knock is expected with some additive white noise. The signal is rectified before sampling, then once in the digital domain, is passed through the IIR filter. A dead-zone is used to provide a threshold to ignore the effects of noise and slight knock. When the magnitude of the filtered signal exceeds the threshold value, it starts to integrate. Figure 5.29 shows the filter's response the the chirp signal and the output from the integrator. By setting a threshold value (of say 1) for the integrator then knock can be deemed to have occurred.

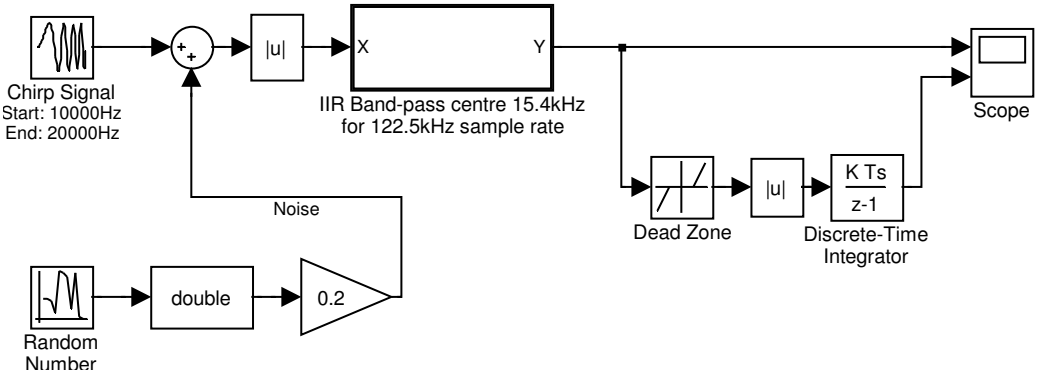

Figure 5.28: Knock detection example model

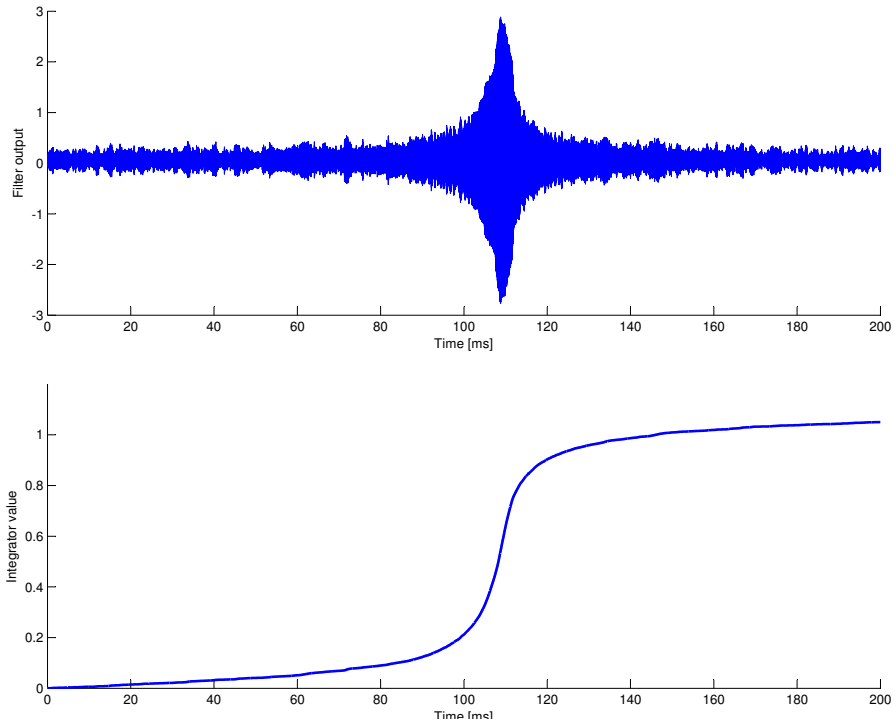

Figure 5.29: Plot of filtered chirp signal and model detection output

5.6.6 Electronic Throttle Unit and Controller

The AJ-V8 engine is fitted with a Denso electronic throttle unit. The electronic throttle enables the ECU to perform the following functions:

- Engine air flow control
- Engine idle speed control
- Vehicle cruise control
- Torque reduction for stability/traction control (in response to ABS commands)
- reverse gear torque reduction

- Engine power limiting
- Engine and vehicle speed limiting
- Mapping of throttle response to engine torque allowing for torque based control and compensation for changes in vehicle altitude and other climatic conditions
- Compensation for torque loss due to catalyst light-off spark retard strategies
- Tip-in bump elimination on manual transmissions

The interested reader can refer to McKay et al. (2000) for further detail about how electronic throttle control can be usefully employed. In earlier mechanical throttle designs for fuel injection systems an idle air by-pass valve is used to allow fine control of the intake air flow when the throttle is fully closed and the engine is idling. When electronic throttles are fitted, idle control strategies can use a combination of throttle actuation and ignition advance control to vary torque output and regulate engine idle speed. The throttle position movement provides for a slow coarse adjustment whilst controlling ignition advance provides a faster means to provide a finer adjustment to sooth the idle speed. The exact proportion to which each is used will depend upon various constraints such as requirements upon noise vibration and harshness (NVH) and exhaust emissions. Cruise control requires an electronic throttle, but so-called *drive-by-wire* also makes possible the decoupling of the accelerator pedal from the actual throttle *butterfly* which is advantageous as it can be used to prevent the driver applying very fast throttle transients leading to A/F ratio excursions which cannot be corrected and have a detrimental effect on emissions. Also throttle response characteristics can be programmed to alter the throttle response feel as perceived by the driver to create sports modes, compensate for altitude torque loss, or create a driver torque-demand based control provided the engine's torque response is known for all conditions.

The unit is of a somewhat larger and more complicated design than those which have been more recently used. The main distinction in the design is that it has mechanical redundancy/limitation from a convention throttle cable whereas newer designs are entirely electronic and have no physical connection to the vehicle's accelerator pedal. When the unit is unpowered the throttle plate can be moved using the accelerator input shaft due to the spring force acting between the throttle butterfly and the input shaft mechanical guard. When a vacuum is applied, the guard is moved and the throttle butterfly moves to its fully open position. The throttle plate can be closed under the action of the throttle motor. The torque required from the motor is therefore more to close the plate than to open it. During normal driving ECU attempts to track the accelerator position with the throttle valve to within a minimum offset distance so that the engine's power output is directly determined by driver demand. The time for the throttle to fully open from the idle position is a maximum of around 120 ms, and from

fully open to idle a maximum of around 140 ms (Jaguar, 1996). From inspection of the Denso ECU, observation of throttle operation and measurement of the drive signal, the throttle plate appears to be driven in both directions from a half-bridge driver stage using a signed anti-phase whereby a 50% PWM duty is required to hold the motor stationary. A PWM frequency of around 240 Hz is used by the Denso controller. It is not known why this frequency was used (seemingly low for a PWM motor drive) but it was possibly just high enough to be filtered out from the analogue input signal filters (forming a low end constraint) whilst being kept as low as possible to reduce EMI related problems and/or keep switching losses low (the frequency at which they occur) allowing for a lower slew rate also to minimise potential radio frequency noise problems (forming a high end constraint). Using a low frequency results in the power conversion causing an audible whine in the motor, but the noise is not of great concern as it is low in comparison to the noise of a running engine and the vehicle occupants are insulated from it from within the cabin. The control strategy appears to display some level of adaptivity in the gains used as it was possible to *trick* the controller by sudden application or removal of vacuum to the throttle unit which resulted in some temporary instability each time until a short period of adjustment has elapsed (in the order of a couple of seconds).

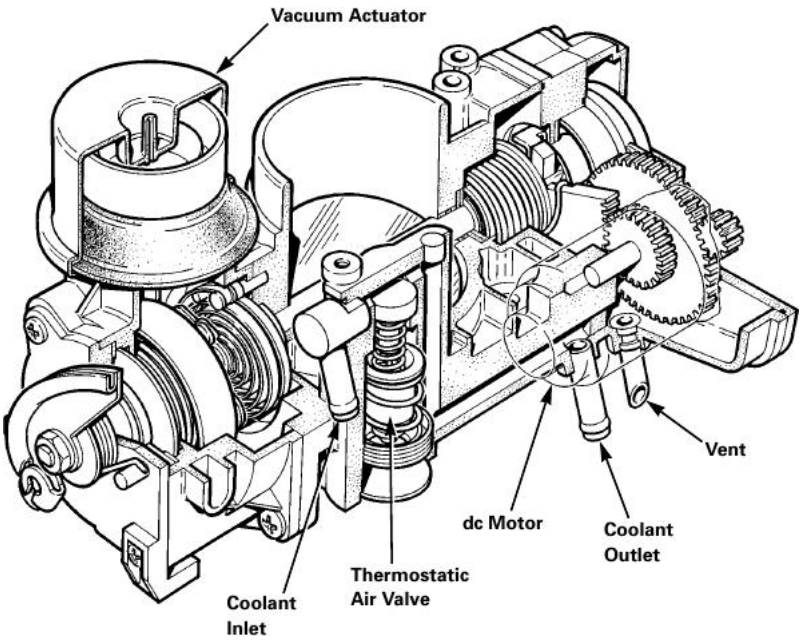

Figure 5.30: Electronic throttle unit cross-section (Jaguar, 1996)

The throttle unit is fitted with three position sensors (Figure 5.31). There are two potentiometers configured as separate tracks in a single unit shown in Figure 5.33, one (dual track) for measuring the accelerator pedal position, and the other (single track) to measure the position of the mechanical guard. The two are normally coincident, but during cruise control the guard is moved out of the way using vacuum, so its position has to be monitored separately to the accelerator pedal. The third sensor is a rotary dual hall

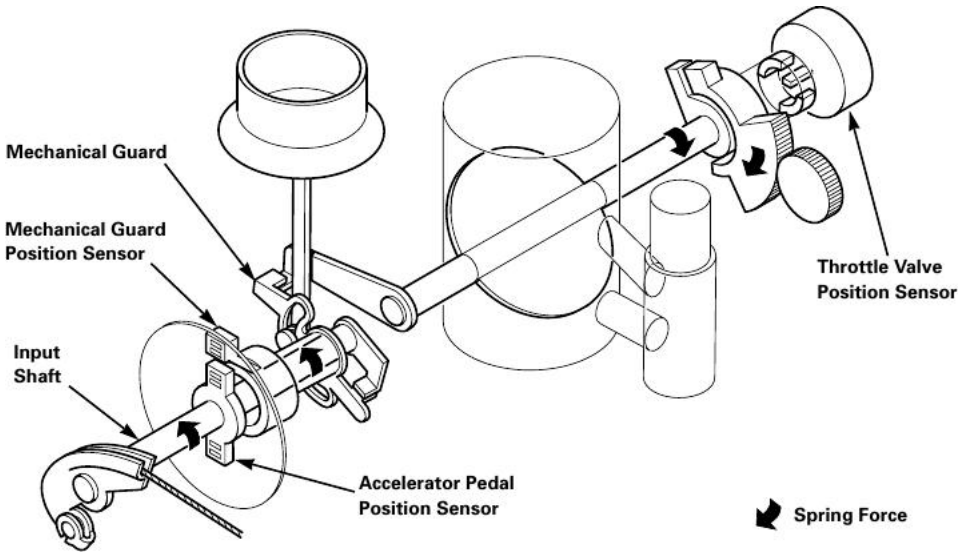

Figure 5.31: Simplified diagram of the electronic throttle unit (Jaguar, 1996)

effect sensor at the motor end of the electronic throttle, which measures the position of the throttle plate shaft. Figure 5.32 is taken from a datasheet (P3-America, 2005) for rotary hall effect sensors and shows that the characteristic is approximately linear over a 90° range which adequately covers the range of motion for a throttle butterfly. The butterfly is servo-actuated and forms a control loop actuator during idle and under cruise control conditions which could contribute to significant wear of a conventional potentiometer. As hall effect sensors are non-contacting, there is no track wear.

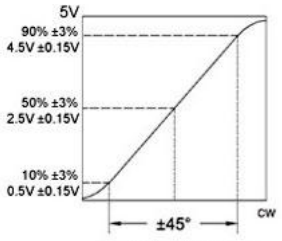

Figure 5.32: Rotary hall effect potentiometer characteristic (P3-America, 2005)

During engine air flow control when the accelerator pedal is above idle, the ECU positions the throttle valve in response to the inputs from the accelerator pedal position sensor. The valve follows or leads at a minimum distance to the mechanical guard as the guard is moved by the accelerator pedal. Thus engine power output is directly related to driver demand.

A solenoid valve is used to apply a vacuum stored in a small reservoir tank to lift the mechanical guard out of the way to allow full throttle actuation without driver intervention. When cruise control is cancelled the vacuum is released and the guard returns to the position determined by the accelerator pedal. Fail-safe measures are in place (such as a redundant vacuum release solenoid and brake pedal switches) to ensure

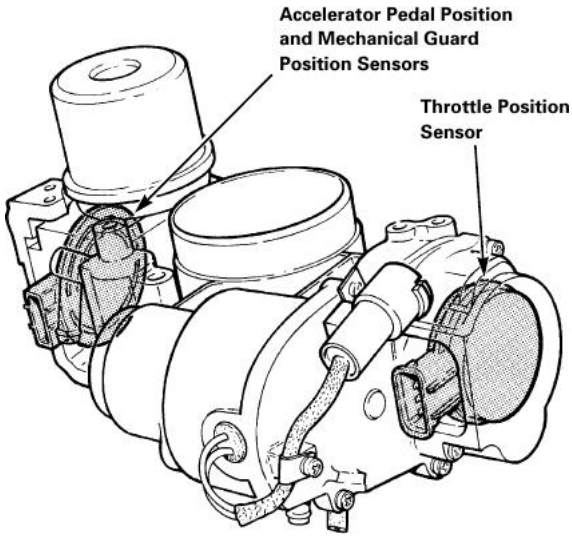

Figure 5.33: Electronic Throttle Sensors (Jaguar, 1996)

that cruise control can always be cancelled so that the protection provided by the mechanical guard is not defeated.

It was initially thought the throttle controller design would be design an be based around a dsPIC which could be used both for the dual purpose of engine knock detection (making use of the DSP instruction set for IIR or FIR band-pass filtering) and throttle control this is separate to the main ECU processor so that the throttle position can be independently set by the dynamometer control system and used without the main development ECU needing to be in an operational state. This might permit the engine to be run under dynamometer load using the original ECU off-idle. When used alongside the development ECU, the throttle controller could be commanded directly from the ECU via SPI for idle control then overridden or monitored remotely using a CAN interface and a suitable protocol such as SAE J1939. However, time constrains meant that the knock detection circuit would never be realised within the scope of this project, so it was decided instead to reduce the throttle controller to a simpler design based upon a 8-bit microcontroller which the author has previous experience, a PIC18F458, and commanded through a CAN interface only, leaving SPI as a future exercise. A board was designed and constructed to control the throttle unit. To ensure independent operation from the ECU, the board has its own 5 V sensor output to supply derived from a precision reference which is buffered to increased current sourcing capability sufficiently to supply all of the throttle sensors with some reserve capacity. The board is supplied with 5 V for the digital components and 12 V nominal battery voltage needed for the analogue signal conditioning components and 5 V reference. Both supply inputs are Tranzorb/TVS diode protected to help improve immunity supply over-voltage transients. The 5V sensor supply output is also Tranzorb clamped to help protect the sensors from over-voltage should the reference circuit fail or the sensor wiring be mis-connected to the battery positive (in such an event the Tranzorb would likely fail short-circuit which

should result in a supply fuse blowing). The throttle unit has an output for each of the two accelerator position tracks and the mechanical guard track. Each of the three tracks are supplied from the 5 V source and the wiper voltages are fed back to the board. The throttle position sensor (TPS) is hall effect and has two current controlled outputs which are supplied from the same 5 V reference, but through 2.2 kΩ resistors (as per the original ECU) so that the current can be measured as a voltage. None of the sensor outputs cover the full 0-5 V span and head room is left at both ends of the range. This allows for fault diagnostics to be performed as a reading of either 0 V or 5 V may indicate a short to ground, short the vehicle supply, open circuit in the wiring or a faulty sensor. Wiring diagnostics are less of a concern for a research and development environment as faults can be investigated directly and more easily on a test bed than an unsupervised in a vehicle. This combined with use of the relatively low precision 10-bit ADC built into the PIC16F458 meant that an increase in effective resolution could be obtained by removing some of the offset and scaling the signals which have no absolute calibration to obscure. Offset voltages are generated on the board using fixed resistance potential divider resistor pairs which are fed through unity gain voltage followers (to reduce the source impedance) into the two ADC voltage reference pins on the PIC18F458 device.

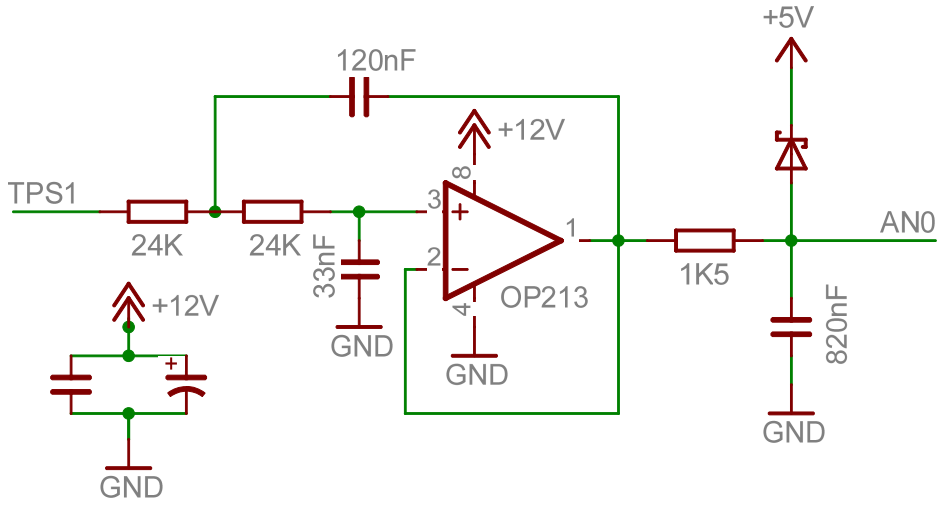

Figure 5.34: Active input filter schematic diagram

An active 3^{rd} order Butterworth low-pass filter in a Sallen-Key topology (Figure 5.34) was designed to be used with each of the input signals to provide anti-aliasing, noise reduction, and to provide a low impedance source to the ADC to minimise the effects of signal distortion caused by the ADC sample-and-hold capacitor loading an input signal each time a sample is taken from it. The filter consists of a 2^{nd} order unity gain active filter with a cutoff of around 117 Hz at -3 dB.

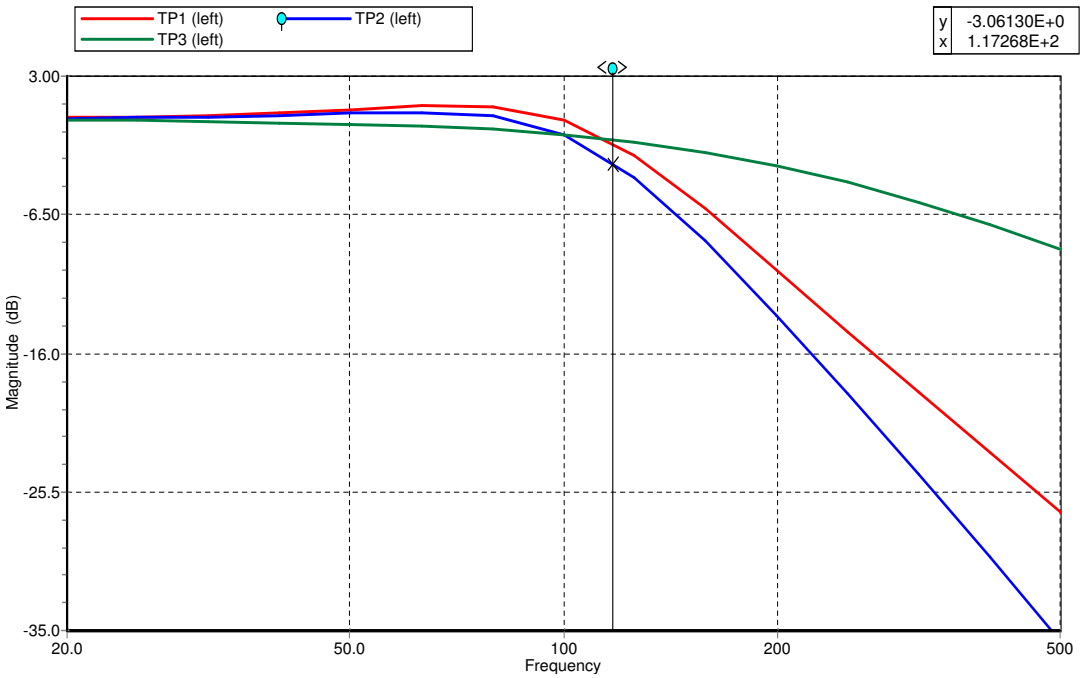

Figure 5.35: Simulated frequency response plot for input filters used on the throttle motor controller. The Green line shows the 1^{st} order filter response, red the 2^{nd} order, and blue shows the combined 3^{rd} order response. The cursor marks the -3dB point for the combined filter stages.

This frequency was chosen to allow a fast throttle response without introducing too much phase lag for control purposes whilst also keeping noise input to a minimum. The operational amplifiers that have been used are supplied from the 12 V rail so if they were to have no input connected at the outputs whilst powered, their outputs could potentially float up to that 12 V rail. This would result in a low impedance connection of 12 V or more to the 5 V maximum rated ADC input. To prevent this from happening, for each filter circuit, a current limiting resistor was placed on the output and then a Schottky signal diode to the 5 V supply to clamp the output to 5V. Since there was now to be a resistor on the output, the opportunity to put a capacitor after it to form an additional 1^{st} order low-pass filter was taken. The 2^{nd} order active filter was designed to have a slightly raised frequency response close to the cut-off frequency to compensate for the additional roll-off created by the addition of the 1^{st} order passive filter. The result is a 3^{rd} order overall filter with a nearly flat frequency response and a sharp fall-off at the cut-off frequency. Figure 5.35 shows the separate and combined simulated frequency response of the filter sections produced using spice software called *5Spice*. It was not possible to achieve a completely flat response before roll-off using standard series resistor and capacitor combinations, but the slope of amplitude decay and the sharpness around the cut-off frequency is much improved over using a simple 1^{st} order RC filter alone. The filters are decoupled with through-hole mounting tantalum and surface mount ceramic chip capacitors in parallel which are mounted as close as possible to the supply pins. This combination provides both a large reservoir capacity and very low equivalent series

resistance (ESR) to help reduce the effects of supply variation and noise as well as sudden loading of the filter output by the ADC sample and hold capacitor. Partial ground and power plains (due to the 2-layer design) are used to deliver the supply to the op-amps. Most of the ICs were chosen as the through-hole type so that they can be socketed and removed if required (since the design is a prototype). The surface mount passive components have been used for the filters to increase the component density enough that the 2-layer design could be routed onto the area of a PC/104 form-factor board.

Additionally, the board provides connection outputs for two status LEDs to be mounted remotely if required, a PWM logic output, and a direction logic output. An open drain output to permit low-side switching of the throttle motor relay has also been provided. The relay switching is performed using an Infineon BSP318S MOSFET which is rated for automotive applications and is logic level switchable, and avalanche capable (without damage) which make it suitable for switching slightly inductive loads such as a relay. Despite this a protection diode was also added between relay connection and the battery supply rail. Additional protection of the gate was added in the form of an in-series resistor (1 kΩ) then a resistor to ground (47 kΩ) to prevent gate ringing and ensure fast turn-off (discharging of the gate capacitance). A 5 V zener diode was also connected between the gate and ground to reduce the likelihood of gate damage due to electrostatic discharge or gate over-voltage. A header is provided for an off-board CAN transceiver as well as an RJ11 connector for in-circuit programming/debugging of the PIC18F458. An I^2C serial EEPROM was added to the board design to provide for non-volatile storage of fixed or adaptive parameters such as the controller gains and the fully open/closed sensor positions. It was decided to keep the PWM motor drive frequency the same as used by the Denso ECU. A 4 MHz crystal was used for the PIC18F458 as this is the fastest clock frequency that could be used to generate a 244 Hz PWM frequency limited by the maximum divisor ratio for that device.

A Simple-H board (Figure 5.36) was selected to drive the DC throttle motor. This board is based on a pair BTS7960B automotive ICs which consist of logic level controllable half-bridge MOSFET-pair drivers. There is *shoot-through* protection by virtue of the IC's design. Only one switch can be enabled at a time as there is only one logic input which toggles between either the high-side or low-side being closed depending upon its high or low state and there is built-in dead time which is dependent upon the slew-rate, externally programmable via a set resistor. The ICs can be paired together to form a full H-Bridge if required. The Simple-H board bridge configuration can be set via jumpers so that the bridge-halves can be *ganged* together to increase the current rating, or used individually to form a two output full-bridge. There is also provision for supply and mounting of a cooling fan and the on-chip load current sensing is exported to the connector as well as the logic connections.

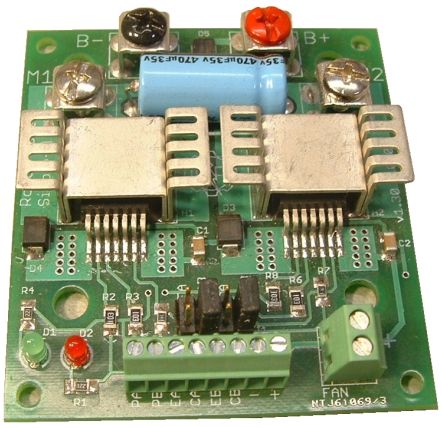

Figure 5.36: Simple-H driver board

The controller board was configure to use the Simple-H board as a half-bridge to drive the motor in one direction only. This was chosen to minimise the development time of the control software. The use of signed anti-phase mode could potentially cause the throttle to slap the end of its travel in the fully closed position during the setup of the 50% PWM duty needed to hold the motor stationary. This could of course have been mitigated by not enabling the h-bridge until a safe duty is asserted, but it dependent upon the correct operation of the software right from the beginning of the development cycle. By using the uni-directional half-bridge mode, a zero start-up duty results in no motor action and the PID control influence can bought into play progressively by increasing the gains incrementally until satisfactory performance is obtained. Half-bridge mode also avoid the need for managing direction changes coherently such as the duty cycle inversion that is required. Full-bridge mode has the potential to be problematic if a oscillatory disturbance was placed on the throttle when it is being held at a constant opening by the controller. This scenario could require rapid changes in the direction of torque application from the motor. The situation may have be improved by the fact that the torque needed to hold throttle plate at constant opening is biased in one direction by the spring which is attempting to hold it against the mechanical guard. The spring force might have been enough to mitigate the need for a motor direction change when rejecting oscillatory disturbances, but without testing this is an unknown.

The throttle unit was removed from the engine to allow it to be bench tested and the new controller software to be developed. The test setup can be see in Figure 5.37. The throttle motor was powered from a pair of 6 V sealed lead acid batteries connected in series, whilst the controller board was power from a bench PSU. A vacuum pump was connected to the vacuum actuator to hold the mechanical guard out of the way of the throttle motor. With the guard raised the throttle plate is decoupled from the accelerator pedal lever so that only the motor can move it.

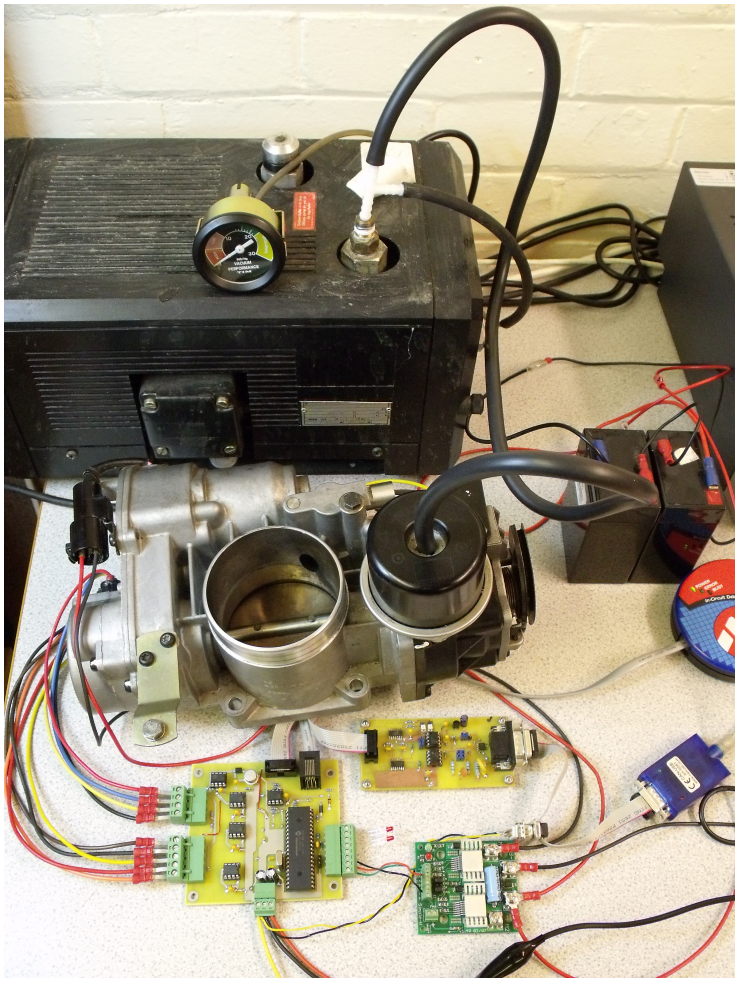

Figure 5.37: Photograph of the throttle unit test setup including the controller board, CAN interface, vacuum pump, and batteries

The control software was written in Microchip C for the 18 series PIC and is structured in a semi-time triggered manner. A fully time-triggered system would use only a single periodic timer interrupt to control program flow through a loop and all other resources would be polled. For this configuration a timer is used to in this way, but also a separate CAN interrupt is used and there is also main program loop to perform low priority non-control related asynchronous tasks like processing CAN messages. Timer 0 is used for the high priority interrupt and the low priority interrupt is used for servicing the CAN buffers. Timer 0 is used to action the main control loop and is set to 16-bit mode with a 1:1 prescale. An interrupt occurs whenever the timer overflows and the timer is incremented by each processor clock cycle. The timer is set to give around a 814 Hz interrupt rate by writing to the timer register at the end of each ISR. The value written is compensated with the value currently in the timer register which is the time the ISR has taken to execute in timer ticks, since it is has wrapped around to zero when the ISR is initially triggered and continues to count up thereafter. This technique ensures the interrupts are exactly periodic and occur independently of how long each ISR takes to reach it's end through varying code paths. The ADC is software triggered to sample from one of five channels during the interrupt service routine (ISR) and the channel

number is incremented each time this ISR is run. The ISR does not wait for the ADC conversion to take place, but instead collects the result the next time it is run before initiating the next conversion. When four samples have been taken from each channel, after twenty times through the ISR, the mean sample value is calculated for each channel and a PID routine is run to update the PWM output. This gives a PWM update rate of six PWM periods corresponding to 24.6 ms or a 41 Hz rate. The main program loop (within the C `main()` entry point) is used to process incoming CAN messages as can be preempted by both the timer ISR and the CAN ISR, except where data is transferred atomically, as interrupts are briefly disabled. The controller was setup to receive throttle demand positions via CAN messages and through the use of the accelerator pedal lever. Whichever of the two demand the widest opening takes precedence. CAN messaging was setup for testing to be protocol-less, accepting any identifier and using only the first data byte as the demanded position, ignoring subsequent data bytes if present in the message. For the purposes of collecting transient data, an area of RAM was set aside in the linker script for data storage. Due to the limited RAM available on this low-cost microcontroller, there was only sufficient space to store in the order of a few seconds of data. Once the RAM region was full, the data would then be automatically output over CAN where it could be logged on a host PC.

With the motor unpowered the throttle plate opens fully under the action of a spring. The main issue with using the half-bridge mode is that the fastest opening time response is determined by the spring. Figure 5.38 shows the throttle plate opening response from fully open to fully closed. The controller saturates to zero output as the motor is unable to drive the plate open in this configuration. It can be seen to take around 2 seconds to open under the action of the spring with a rate of opening of around 60° per second for much of the range. This time would be unacceptable to the driver of a car and severely affect drivability, however for use with a dynamometer controller and for idle control it is adequate. The dynamometer controller will only be required to manipulate the position by small increments to obtain closed loop engine speed/torque control and can follow a relatively slow trajectory to arrive in the vicinity of the set-point. At idle the throttle plate is close to being fully closed where the spring force is at its highest and therefore a reasonable response to small demand changes should be obtainable.

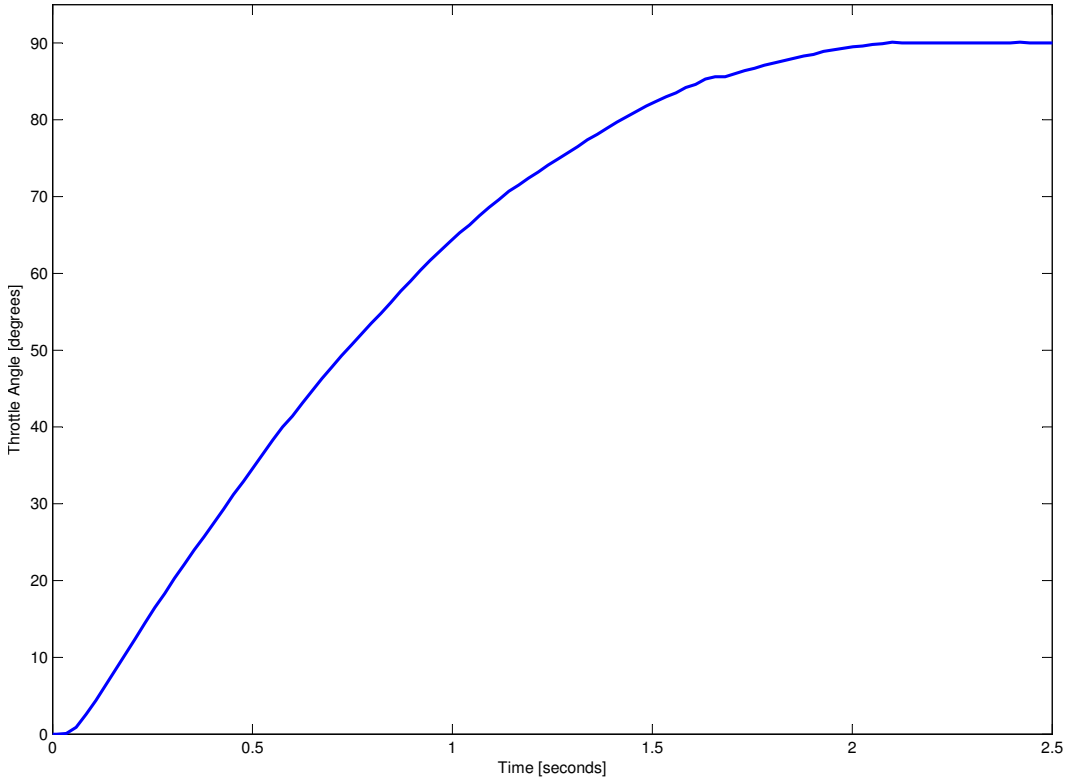

Figure 5.38: Throttle plate opening under spring action alone

A constant motor duty offset was used to overcome the spring force needed to hold the throttle plate stationary. This was found to reduce the reliance upon the integrator term of the PID controller to act to hold the plate when the position error is zero, and so acts as a simple feed-forward constant term. The effect is that when the throttle is opening and the demand position is approached the motor starts to act before the sign of the error has changed which reduces overshoot. Ideally this offset duty would be a function of the either a predetermined function of spring force (which doesn't appear to conform to Hooke's law due to the opposing action of different springs in the system) or be learnt adaptively. For simplicity (and therefore expediency) a constant value was used an found to be adequate. No velocity trajectory control was used, but the maximum velocity was constrained to an upper limit minimise overshoot. Figure 5.39 shows the response to a step changes in position demand issued over CAN. The test was repeated with and without velocity constraining to show the severity of overshoot occurring in the unconstrained case caused by the throttle *bouncing* against the torsional force of the spring as the control action is withdrawn past the set-point. The throttle position returns toward the set-point at a slightly faster rate than when opening under the free action of the spring due to this momentum bounce effect. Constraining the velocity removes the overshoot whilst reaching the set-point in the same time as the overshoot return.

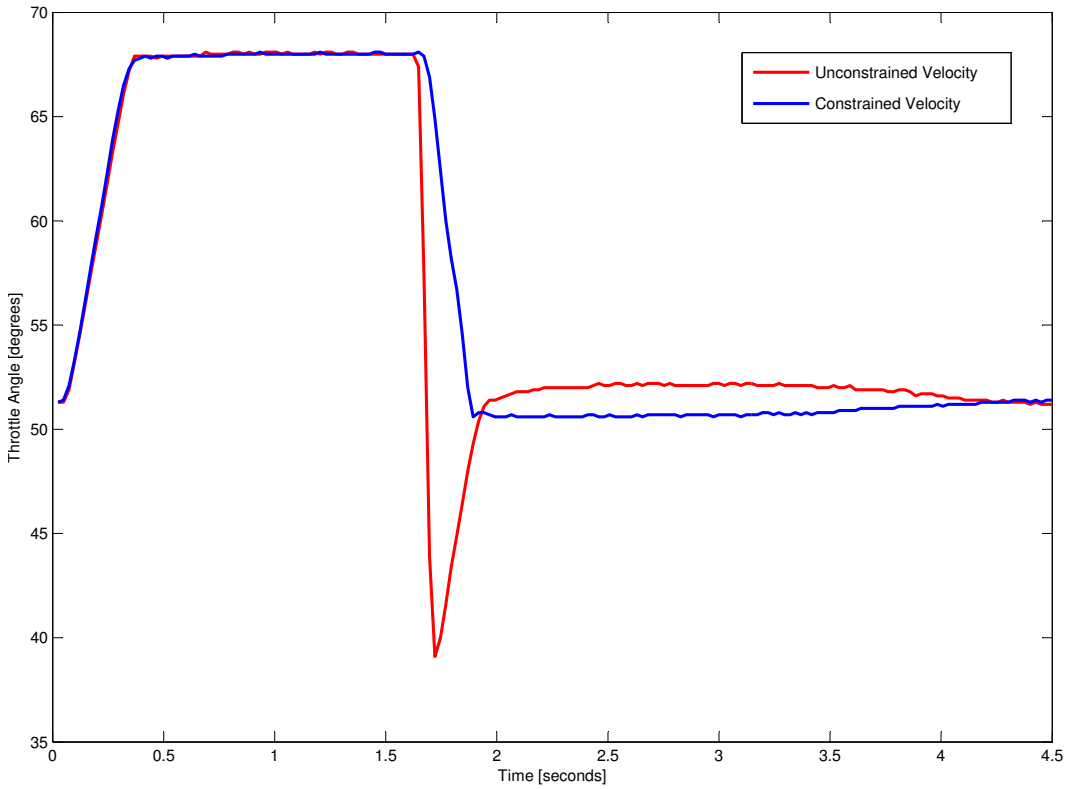

Figure 5.39: Throttle controller transients in response to CAN demanded step changes

Figure 5.40 shows the system's response to a brisk opening and closing of the pedal lever by hand. The controller achieves reasonable tracking performance for most of the range, but struggles close to wide-open-throttle due to its reliance on the spring force. The controller constrains the demand position close to fully open and closed so that the motor is not powered against the end of the mechanism's travel which could result in either an impact if it is already moving, or excessive current being drawn if stationary, but the demand is increased beyond the end of travel.

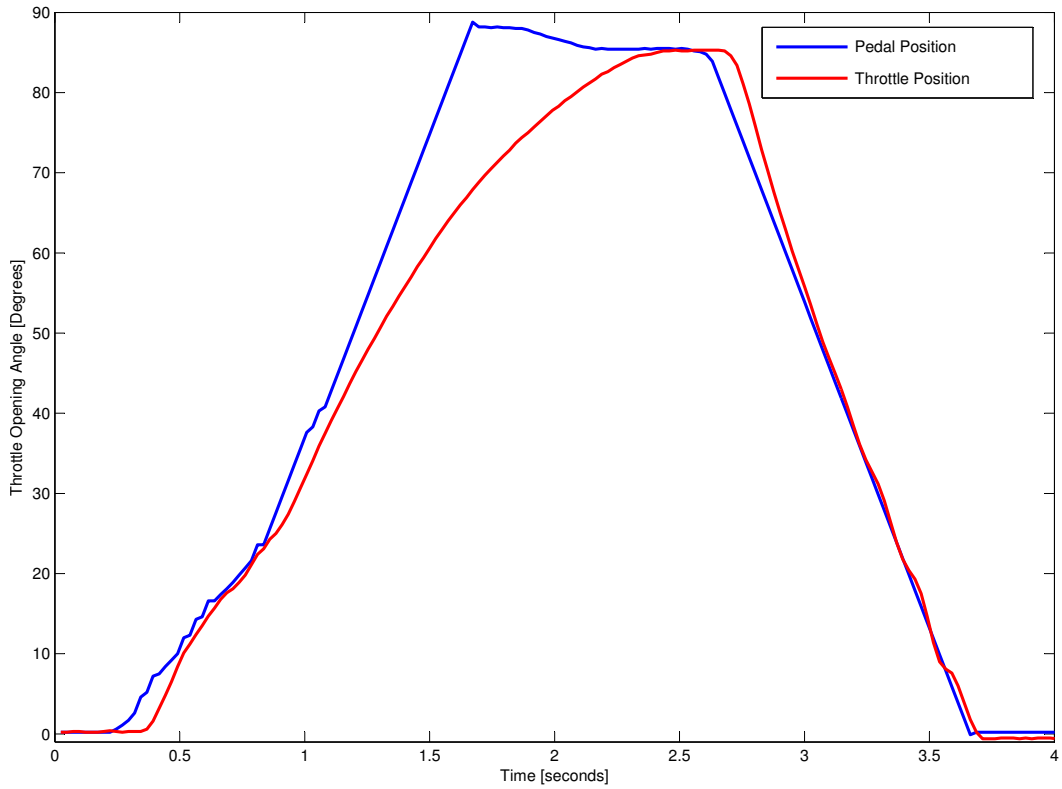

Figure 5.40: Throttle tracking of accelerator pedal lever demand position

5.6.7 Temperature Measurement

The are two temperature sensors associated with the engine which are both resistive thermal devices (RTD). The engine coolant temperature (ECT) sensor which is a negative temperature coefficient (NTC) thermistor and mass air flow sensor (MAFS) unit which has an integrated NTC thermistor to measure the intake air temperature. Figure 5.41 shows the characteristic for a similar NTC thermistor made by Bosch for automotive applications which cover the coolant temperature range of -30 to +120°C.

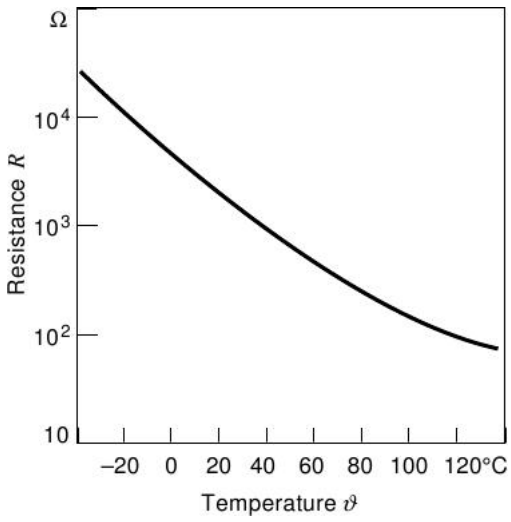

Figure 5.41: Typical automotive NTC thermistor characteristic

The temperature is determined by forming a potential divider with the thermistor and another fixed value resistor. The potential divider is supplied from a constant voltage source (usually 5 V) and the voltage drop across the fixed resistor is measured as the current varies with temperature. The fixed value resistor is chosen to produce a measurable voltage over the required temperature range. The thermistor characteristic stored within the ECU was interrogated by connecting fixed value resistors in place of the thermistor and recording the voltages and corresponding temperatures reported by the ECU using the Denso software. It was established that a 2K7 resistor has been used in series with the thermistor to produce a measurable voltage in the 0-5 V ADC range. The values are quantised to the 10-bit ADC resolution.

Extended Steinhart-Hart equation:

$$\frac{1}{T} = A + B\ln(R) + C\ln(R)^2 + D\ln(R)^3$$

A	1.9131×10^{-3}
B	1.6683×10^{-5}
C	3.1637×10^{-5}
D	-1.1696×10^{-6}

Table 5.1: Steinhart-Hart curve fitted coefficients for ECU NTC characteristic

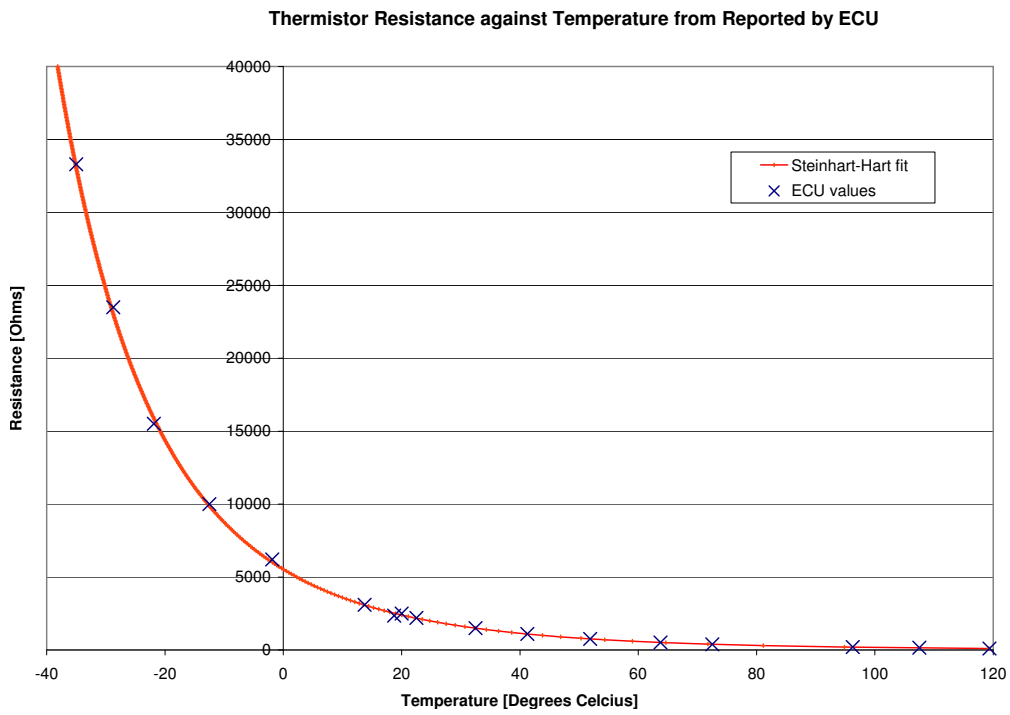

Figure 5.42: NTC thermistor characteristic measured from the ECU

5.6.8 *Intake Air Mass Flow Meter*

There are two methods commonly used to detect the engine load, or the mass of air which has entered the cylinder so that the corresponding amount of fuel can be injected. One method is to measure the intake manifold pressure, the other is to measure the mass flow rate of the air moving through the intake system using either a volumetric air flow meter or a mass air flow meter. The AJ-V8 is fitted with a hot-wire mass airflow sensor (MAFS).

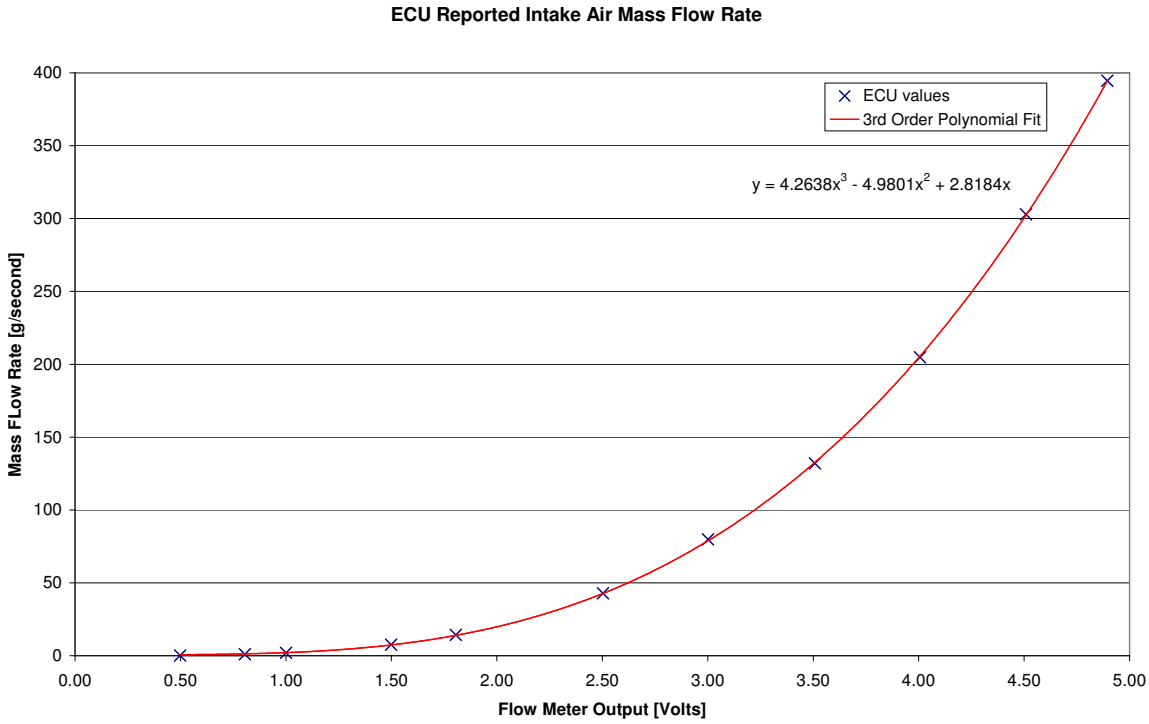

Figure 5.43: MAFS characteristic measured from the ECU

5.6.9 *Establishing the Ignition, Fuelling Sequence and Base Calibration*

Jaguar (1996) provides some limited information about the ignition *firing* sequence for the AJ-V8 engine, but nothing about the absolute meaning of the flywheel and camshaft datums other than their existence and that they are used by the engine management system to determine the engine's angular position and phase. The firing order is 1A,1B,4A,2A,2B,3A,3B,4B where A and B signify the left and right cylinder banks respectively when looking at the engine from the front and the cylinders of each bank are numbered 1 to 4 from front to back. This firing order is far less intuitive than a more conventionally understood four cylinder in-line engine until the geometry of the V8 is considered. The AJ-V8 engine has a cross-plane crank which is conventional for modern V8 engines intended for road vehicles as it has much improved mechanical balancing characteristic to the alternative flat-plane design (which is more commonly used for race engines as it incurs a weight saving by not requiring large crankshaft balance weights).

The flat-plane crankshaft has its crank pins aligned to the same plane so that there is either 0 or 180° offset between them as in a conventional four cylinder in-line engine, the difference being that each pin is shared by two connection rods so that is displaces eight rather than four pistons. A V8 cross-plane engine has its crank pins aligned to two planes that are 90° apart. As each pin is shared by two pistons, this places one of the four pins at each of the 12, 3, 6, and 9 o'clock positions when the crank shaft is viewed from one end. When fitted to a 90° V bank angle engine such as the AJ-V8, this places two pistons (connected to different crank pins) simultaneously at TDC every 90° of crank shaft rotation. An unavoidable drawback with the cross-plane design is that it is not possible to avoid at least one pair adjacent cylinder firing consecutively at some point in the sequence (3B and 4B for the AJ-V8) and this may lead to uneven heat distribution that can result in one or two cylinders being hotter than the others.

Since no information about the relative phase of the camshaft key pulse to the flywheel missing tooth was available, or either the missing tooth and the camshaft key to the absolution position of any piston in its cycle, some investigation was required. It is essentially a mechanical relationship which can be established by inspection of the design, for example by looking at the camshaft lobe positions, but it was desirable not to have to dismantle the engine if it could be determined by other means. The TDC position of the piston in cylinder 1B was previously determined relative to a crankshaft mounted incremental encoder. The position of the first tooth after the missing tooth gap could then be found relative to an absolute angle of crankshaft rotation by reading the angle between TDC of cylinder 1B to the position where the first flywheel tooth after the missing tooth gap is aligned with flywheel sensor. The alignment was found by turning the crankshaft by hand and using an inspection hole present in the under side of the engine casing to feel for the missing tooth in the timing ring on the inward side of the flywheel. The sensor was then removed to allow the first tooth after the missing tooth gap to be centred in the orifice in which the sensor is mounted.

Once the crank angle to flywheel datum relationship had been established, the engine camshaft phase needed to be determined. In a wasted spark ignition system, the phase can be neglected as the spark energy is naturally directed to the cylinder spark plug which has the lowest spark gap impedance. This is the cylinder under compression (the one with compressed wet vapour). However, with a plug-on-coil system (as fitted to the AJ-V8), the engine's phase needs to be known so that the correct plug is triggered at the correct time in the engine cycle. For this to be known, two pieces of information are required. The first is the (crankshaft) angle at which the camshaft key pulse occurs, and the second is the significance of the key pulse for a particular cylinder, whether or not that cylinder is in its compression or exhaust phase. To determine the camshaft key pulse position relative to the flywheel pulse sequence, the engine needed to be run at a relatively constant speed to allow an oscilloscope to trigger consistently from the

flywheel pulse train. This was achieved by removing all eight spark plugs and *motoring* the engine with the starter motor to achieve a steady pulse train. Figure 5.44 shows the resulting sensor output and the LM1815 conditioned pulses from both the flywheel and camshaft sensors. The first visible flywheel pulse is the first tooth after the missing tooth gap, and it can be clearly seen that the camshaft pulse occurs almost coincident with the sixth flywheel pulse.

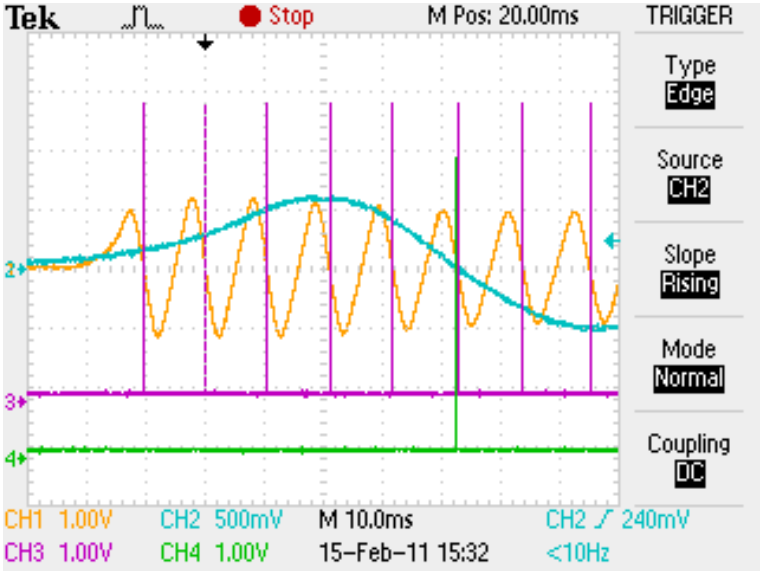

Figure 5.44: Camshaft to crankshaft phase shown from dual LM1815 outputs. CH1 from is the flywheel sensor and CH3 the zero crossing detection pulse, CH2 is from the camshaft sensor and CH4 shows the corresponding detection pulse.

The camshaft key position relative to the engine phase was then determined by connecting the original ECU to the engine and running the engine and searching for a cylinder who's ignition trigger pulse from the ECU was located temporally near to the camshaft key pulse. Since it had already been determined that cylinders 4B and 2A are at TDC at the midpoint between flywheel tooth 4 and 5, and that the camshaft key is in the vicinity of tooth 6, then the search was reduced to the ignition point of these two cylinders. The exact ignition point is unimportant and will vary under control, but its proximity to a particular cylinder's TDC unambiguously reveals the engine's phase. Figure 5.45 shows the occurrence of camshaft key pulse relative to the ignition trigger logic pulse output from the ECU for cylinder 4A, that confirms the key pulse signifies the compression phase for cylinder 4A. With this information (combined with a knowledge of the firing order) the firing point of each cylinder can be expressed relative to flywheel and camshaft pulses and so the TPU can be configured to fire each spark plug at an absolute angle and in the correct engine cycle phase.

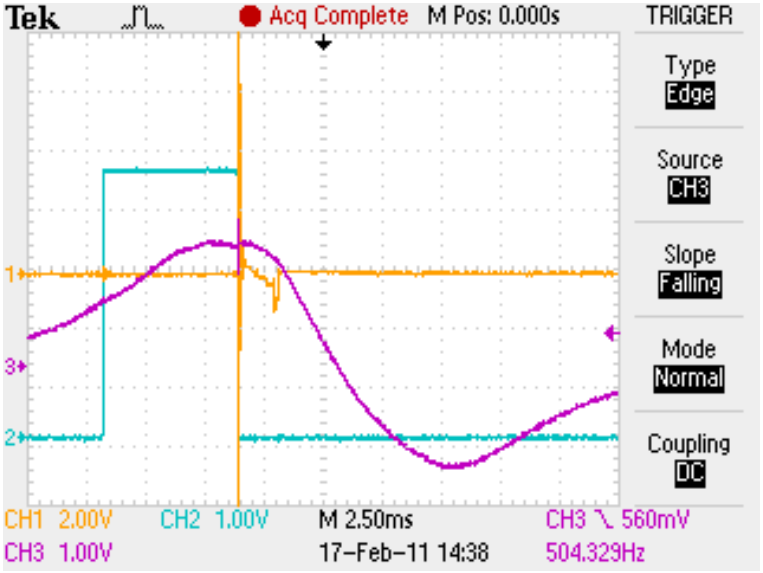

Figure 5.45: Camshaft key pulse to ignition phase. CH1 is the injector voltage, CH2 is the ignition trigger pulse for cylinder 4A from the ECU, and CH3 is from the camshaft sensor.

Chapter 6. Conclusions

This work has been undertaken to establish if the output torque of a spark ignition IC engine can be beneficially used to optimise it's performance using a blackbox model approach coupled with online system identification or other adaptive compensation technique. To achieve this aim a large amount of time and effort was placed on providing the supporting architecture upon which the necessary identification work can be performed. This chapter summarises the contributions and achievements, and suggests how they can provide a useful platform for future work in this area.

6.1 Conclusions

The aim of this project was to establish if a directly measured or inferred engine torque signal could be used to optimise the operation of an SI engine, starting from either no initial calibration or a suboptimal calibration, in the face of parameter drift due to varying steady-state conditions such as environmental changes, ageing effects, or changing fuel composition. The intention was to consider the engine essentially as a black box upon which there would be an initial structure detection process of either a forward or direct inverse plant model. If a forward model was used, then this could then be placed into a non-linear model predictive controller or other model based control scheme. Alternatively, a direct inverse model could be used to compensate a conventional proportional control scheme in place of the mapped look-up calibration tables that are currently commonplace for engine control. An existing experimental facility which comprised a Jaguar AJ-V8 SI engine coupled to a Heenan-Froude Dynamatic GVAL (Mk 1) dynamometer was adopted for this work, in the expectation that it would provide a flexible and reconfigurable test-bed on which the engine controller could be implemented and evaluated. In the event this decision proved to be problematic insomuch that it required considerable effort to re-commission the engine, after which it was established that the eddy-current brake controller was no longer serviceable and required a complete redesign; which subsequently proved to be a major distraction from the original aims of the work. Nevertheless, a flexible distributed engine test system consisting of a combined user interface and cylinder pressure monitoring system, a dynamometer controller, and a modular engine controller which is close coupled to an embedded PC has been created. The considerable challenges involved in developing this system, have meant that the key objective of this project has not been met, i.e. establishing a torque feedback based engine control scheme. Notwithstanding the departure from the projects initial aims, this has led onto some interesting aspects of eddy-current dynamometer control, in particular current control, which is an area that

can be assumed to be increasingly left to a commercial domain (as indicated by the lack of recent published work) both in terms of an understanding of the control issues and in being able to fund the purchase of the necessary hardware.

6.1.1 Dynamometer Control

This work has encountered interesting issues of dynamometer control, a machine that is still in widespread use, and it is suggested that by putting it into the more contemporary context of embedded digital control, it may encourage future work in this area, making the domain of engine testing in general more accessible to university based research. The dynamic effects of eddy-current damping and magnetic saturation were found to perturb current control loop considerably and obscure the current to flux (and hence torque) relationship. A possible outcome from using a model based current control approach is that the need for a high voltage supply (above that of 240V mains) for improved time response to demand input changes is reduced as a more optimal response time can be achieved using the simple, robust and off-the-shelf available hardware consisting of a phase angle controlled thyristor bridge. Phase angle control of dynamometer has received some criticism in other published work due to the slow response time caused by switching opportunities only occurring once every mains half cycle. However, linearisation of the thyristor trigger angle to output voltage relationship in software was found to give improved performance and when combined with a model based scheme the need for a faster response to cancel the dynamic effects of eddy-current damping diminishes. A model predicting the current response at stand-still using a fictitious damper winding was found to be valid up to the point where significant saturation starts to occur. This model required parameter fitting to each region of operation, and more work is needed to quantify the change in parameter values over the operating range then assess the quality and repeatability of a piecewise linear or non-linear model which covers the entire envelope of operation for control. An XScale processor was used to implement a combined controller and graphical user interface for the dynamometer. The performance of this device was found to be inadequate to provide a responsive and usable graphical interface, but performed well as a controller with a simpler terminal user interface. An embedded PC board became available during the work and this was used to provide a graphical operator interface. Recent developments in ARM core processors for hand-held devices may mean that once these processors have become available in the form of embedded processor cards for industrial applications, there may be an improved prospect for creating a combined controller and graphical operator environment run on the same device.

6.1.2 Engine Controller

An open architecture software and hardware engine controller and independent throttle controller aimed at research use has been created. The novelty of this system is that it allows close co-operation between the low-level engine control application running on a microcontroller and a higher performance attached processor device such as an embedded PC. This approach will provide higher bandwidth than a CAN Calibration Protocol connection, and makes real-time data transfer possible without degrading the control determinism. There is provision for the dynamometer controller to take control of the throttle position to remove the need for a separate throttle cable actuator for closed loop torque/speed control. The hardware layout is flexible since it is modular and scalable so that there is scope for adaptation for use with different engines by adding/replacing individual module boards. The boards can be arranged on a panel for initial setup work then re-configured into a PC/104 stack if space is restricted or easier portability is required. The Motorola TPU microcode functions for engine control, particularly the PMM function for crank angle detection, did not work effectively, and unfortunately because of severe time constraints towards the end of the project there was insufficient time remaining to adopt an alternative approach. An optical cylinder pressure transducer and half-degree crank angle synchronous sampling system with real-time graphical rendering of pressure traces was developed to assist the validation of the ignition timing on-line optimisation process so that it can be demonstrated when a controller is able to maintain the peak cylinder pressure position close to that which maximum brake torque is achieved. The pressure monitoring system also allows an independent means to monitor of the onset of combustion knock which may be encountered if a torque feedback learning algorithm is allowed to run free since under certain operating conditions the maximum brake torque is not achievable without the onset of knock and is therefore said to be knock limited. There is also the possibility to use the indicated cylinder pressure as a means to infer output torque and so it could be used in place of a torque sensor as part of the engine control loop.

6.2 Recommendations for Future Work

From the work on dynamometer control there are issues which remain that offer the opportunity for further investigation. The dynamometer has voltage-current, current-flux, and flux-torque relationships which change both transiently and in steady-state with both the speed and torque operating points. To control the output torque in a timely fashion using applied voltage, it may be sufficient to manipulate the magnetic flux level using an estimate determined from measured current and a current-flux model. To establish the current-flux and flux-torque relationships for a range

of operating points a high power servo-motor could be used and magnetometers (or other flux measuring device) placed onto one the rotor's teeth to determine air-gap flux. A telemetry system might be required to acquire the data from the rotating magnetometers if fitted to the rotor and the instrumentation would require sealing from exposure to the cooling water. If a high-powered servo motor was used, then this could be used to drive the dynamometer for characterisation. A servo-motor would give reduced torque and vibration noise over that of an IC engine and provides an independent means to allow data to be collected from a range of operating points, removing the challenge of controlling the dynamometer to reach each operating point whilst at the same time attempting to excite as required for the characterisation or identification process. More work is required to investigate suitable signal conditioning for the torque signal that provides the least phase shift or time delay in the measurement despite the large amplitude wide-band noise created by vibration from the engine and coupling to the dynamometer. A suitably designed filter could improve the phase-gain margins of the system and give improved transient disturbance rejection where torque is used for control feedback. Further improvements in transient response could be achieve by considering the torsional stiffness of the coupling between the engine and dynamometer as well as the separate intertias of the two so that engine torque can be estimated from the measured dynamometer torque. These two torques are equal under steady-state conditions but differ momentarily during speed transients. Initially, an XScale based Arcom Viper board was used to provide both real-time control of the dynamometer and an interactive graphical user interface. The resources of this platform were found to be insufficient to perform the rendering of the interface with an appropriate level of responsiveness. During the course of this project, newer processors have emerged which use ARMv7 Cortex architectures (such as OMAP) which offer greatly increased performance over the XScale based on ARMv5 since they allow faster clock speeds, and can have multiple cores with floating point and DSP instructions. This now makes it more feasible to construct the combined embedded user interface and controller which was oringinally attempted. If this approach was successful, then further investigation into the use of real-time Linux and the interaction between the controller code and the user interface code without loss of determinism would be worthwhile. To complete the engine controller which has been constructed the issue of TPU microcode for crank angle measurement needs to be resolved. Assuming that commercially available microcode would not be acquired due to high cost, the challenge that remains is to write custom microcode equivalent to the Motorola PMM and PSP functions, or investigate new and improved eTPU microcode functions from Freescale which exist for more recent microcontrollers such as the MPC55xx series which has superseded the older Motorola MPC5xx series. The change to a new eTPU processor platform would provide greater longevity and allow use of a microcode compiler that has a C like programming interface. Pseudo serial device drivers were written as a data transfer mechanism through

a PCI bus connecting the PATI board and an embedded PC. Although this mechanism works, it became apparent that a CAN driver, upon which the CAN Calibration Protocol (CCP) could be implemented, would be a better solution both for the underlying hardware and to make use of the existing CCP protocol that can support data acquisition, observing variables and modifying constants (calibratibles) on a running system. Both eCos and Linux have CAN device driver APIs which could be used to reduce the development effort of buffering data and delivering it to the applications running on both sides of the bus. The use of DMA could also be investigated as an alternative for transfers of large blocks of data if that feature was needed for a particular purpose. The ECU hardware was tested with the AJ-V8 engine, but could be generalised for use with other engines. One way to achieve this is through the use of programmable peak-and-hold injector current to allow different fuel injectors to be used. This could be done using a TPU channel configured as a PWM output which would then be fed through a filter into a comparator to set the reference peak injector current which in turn triggers a TPU input capture channel to adjust the PWM signal for the hold current. This could be done in a host service routine using existing TPU functions, or autonomously if custom microcode was written. It would also allow the peak and hold currents to be separately programmable whereas the injector drive IC used for this project only supports a hold current which is a fixed ratio of the peak current. There is also scope to add TPU triggered analogue sampling, at fixed crank angle intervals, for example.

References

Åkesson, H., & Sällberg, B. (2003). *Identification and Analysis of Nonlinear Systems*. Masters, Department of Telecommunication and Signal Processing, Blekinge Institute of Technology.

Analog-Devices (1999a). AD594/AD595 monolithic thermocouple amplifiers with cold junction compensation. Electronic PDF datasheet.

Analog-Devices (1999b). AD623 single supply, rail-to-rail, low cost instrumentation amplifier. Electonic PDF. Rev. C.

Analog-Devices (2000). AMP04 precision single supply instrumentation amplifier. Rev. B.

Andersson, I., & Eriksson, L. (2001). Ion sensing for combustion stability control of a spark ignited direct injected engine. (pp. 609–614).

Aquino, C. (1981). Transient A/F control characteristics of the 5 liter central fuel injection engine. *SAE Journal Transactions*, (pp. 1819–1833).

Arsie, I., Pianese, C., & Sorrentino, M. (2004). Nonlinear recurrent neural networks for air fuel ratio control in SI engines. *SAE Journal Special Publication*, *SP-1822*, 359–368.

Bai, L., & Coca, D. (2008). Nonlinear predictive control based on narmax models. In *Proc. 11th International Conference on Optimization of Electrical and Electronic Equipment OPTIM 2008*, (pp. 3–10).

Ball, J. K., Bowe, M. J., Stone, C. R., & Mcfadden, P. D. (2000). Torque estimation and misfire detection using block angular acceleration. *Modeling of SI Engines (SP-1511)*, (2000-01-0560).

Bannatyne, R. (2002). Microcontrollers for the automobile. Web page article. Micro Control Journal.
URL http://www.mcjournal.com/articles/arc105/arc105.htm

Barreto, G., & Araujo, A. (2004). Identification and control of dynamical systems using the self-organizing map. *Neural Networks, IEEE Transactions on*, *15*(5), 1244 – 1259.

Bea (2004). ATI product overview datasheet. Electronic PDF datasheet.

Beaumont, A., Beauchamp, S., & Noble, A. (1993). Air-fuel ratio control technology for ultra-low emissions vehicles. In *ISATA Proceedings from the 26th international symposium on automotive technology and automation 1993*, 93EN032, (pp. 399–406).

Beaumont, A., Noble, A., & Mercer, A. (1988). Predictive control of transient engine testbeds. In *International Conference on Control*.

Béchennec, J.-L., Briday, M., Faucou, S., & Trinquet, Y. (2006). Trampoline an open source implementation of the OSEK/VDX RTOS specification. In *Proc. IEEE Conf. Emerging Technologies and Factory Automation ETFA '06*, (pp. 62–69).

Billings, S., Chen, S., & Backhouse, R. (1989). The identification of linear and non-linear models of a turbocharged automotive diesel engine. *Mechanical Systems and Signal Processing*, *3*(2), 123 – 142.
URL
http://www.sciencedirect.com/science/article/pii/0888327089900125

Billings, S., Fadzil, M. B., Sulley, J. L., & Johnson, P. M. (1988). Identification of a non-linear difference equation model of an industrial diesel generator. *Mechanical Systems and Signal Processing*, *2*(1), 59–76.

Borg, J., Cheok, K., Saikalis, G., & Oho, S. (2005). Wavelet-based knock detection with fuzzy logic. In *Proc. CIMSA Computational Intelligence for Measurement Systems and Applications 2005 IEEE International Conference on*, (pp. 26–31).

Box, G. E. P., & Draper, N. R. (1987). *Empirical Model Building and Response Surfaces*. John Wiley & Sons.

Brunt, M. F. J., Huang, C. Q., Rai, H. S., & Cole, A. C. (2000). An improved approach to saving cylinder pressure data from steady-state dynamometer. *SI Combustion*, *SP-1517*.

Camacho, E., & Bordons, C. (2003). *Model Predictive Control*. Springer-Verlag.

Chamaillard, Y., & Perrier, C. (2001). Air-fuel ratio control by fuzzy logic, preliminary investigation. In G. Kiencke, U. Gissinger (Ed.) *Advances in Automotive Control 2001. Proceedings of the 3rd IFAC Workshop*, (pp. 211–216). Pergamon. Accesion no: 00824366.

Chang, C.-F., Fekete, N., Amstuz, A., & Powell, D. (1995). Air-fuel ratio control in spark-ignition engines using estimation theory. *IEEE Transactions on Control Systems Technology*, *3*(1), 22–31.

Chang, C.-F., Fekete, N., & Powell, D. (1993). Engine air-fuel ratio control using an event-based observer. *SAE Journal Transactions*, *102*(3), 1002–1017.

Chen, W.-H., Ballance, D., & O'Reilly, J. (2000). Model predictive control of nonlinear systems: computational burden and stability. *Control Theory and Applications, IEE Proceedings-*, *147*(4), 387 – 394.

Chen, Y., & Carpenter, R. (2010). Use of feedback control to implement ECU system function testing. *SAE Journal Transactions*, (2010-01-0663).

Chevalier, A., & Hendricks, E. (2000). Predicting the port air mass flow of si engines in air/fuel ratio control applications. *SAE Journal Transactions*, *109*(3), 183–209.

Chevalier, A., MÃ¼ller, M., & Hendricks, E. (2000). On the validity of mean value engine models during transient operation. *SAE Journal Transactions*, *109*(3), 1571–1592.

Chikkula, Y., Lee, J., & Ogunnaike, B. (1995). Robust model predictive control of nonlinear systems using input-output models. *American Control Conference, 1995. Proceedings of the*, *3*, 2205 – 2209.

Cho, D., & Hedrick, J. K. (1988). A nonlinear controller design method for fuel-injected automotive engines. *Transactions of the ASME; Journal of Gas Turbines and Power*, *110*, 313–320.

Cho, D.-I. D., & Oh, H.-K. (1993). Variable structure control method for fuel-injected systems. *Transactions of the ASME; Journal of Dynamic Systems, Measurement, and Control*, *115*, 475–481.

Choi, S. B., & Hedrick, J. K. (1998). An observer-based controller design method for improving air/fuel characteristics of spark ignition engines. *IEEE Transactions on Control Systems Technology*, *6*(3), 325–334.

Clarke, D., Mohtadi, C., & Tuffs, P. (1987a). Generalized predictive control–part i. the basic algorithm. *Automatica*, *23*(2), 137–148.
URL
http://www.sciencedirect.com/science/article/pii/0005109887900872

Clarke, D., Mohtadi, C., & Tuffs, P. (1987b). Generalized predictive control–part ii extensions and interpretations. *Automatica*, *23*(2), 149–160.
URL
http://www.sciencedirect.com/science/article/pii/0005109887900884

csewards (2004). Agilent HCTL-2032 quadrature decoder/counter interface ICs datasheet. Electronic PDF datasheet.

Darley, S. (1997a). Period measurement with missing transition detection (PMM) TPU function. Electronic PDF.

Darley, S. (1997b). Position-synchronized pulse generator (PSP) TPU function. Electronic PDF.

Davies, E. J. (1963). An experimental and theoretical study of eddy-current couplings and brakes. *82*(67), 401–419.

Davies, E. J. (1966). General theory of eddy-current couplings and brakes. *Proceedings of the Institution of Electrical Engineers*, *113*(5), 825–837.

Demerdash, N., & Nehl, T. (1979). Use of numerical analysis of nonlinear eddy current problems by finite elements in the determination of parameters of electrical machines with solid iron rotors. *IEEE Transactions on Magnetics*, *15*(6), 1482–1484.

Dobner, D. J. (1980). A mathematical engine model for development of dynamic engine control. *SAE Journal Transactions*, *89*(1), 373–381.

Dobner, D. J., & Fruechte, R. D. (1983). An engine model for dynamic engine control development. In *Proc. American Control Conf*, (pp. 73–78).

Dorf, R. C., & Bishop, R. H. (2001). *Modern Control Systems*. Prentice Hall.

Dutton, K., Thompson, S., & Barraclough, B. (1998). *The Art of Control Engineering*. Addison Wesley Longman.

Dyson, A., & Bannoura, M. (1999). *TPU Microcoding for Beginners*. AMT Publishing, Austin, Texas, USA, 3 ed.
URL http://www.amtpublishing.com

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, *7*(1), 1–26. The 1977 Rietz Lecture.

Eriksson, L. (1999). *Spark Advance Modeling and Control*. Ph.D. thesis, Division of Vehicular Systems, Department of Electrical Engineering, Linköping University.

Fairchild-Semiconductor (2005). ISL9V2540S3ST EcoSPARK N-Channel Ignition IGBT. Electronic PDF.

Fegley, K. A. (1956). Determining the parameters of a short-circuited winding that represents eddy-current paths. *Part III Power Apparatus and Systems Transactions of the American Institute of Electrical Engineers*, *75*(3), 143–147.

Fleming, W. J. (1982). Automotive torque measurement: A summary of seven different methods. *31*(3), 117–124.

Fleming, W. J. (1989). Magnetostrictive torque sensor performance - nonlinear analysis. *IEEE Transactions on Vehicular Technology*, *38*(3), 159–167. TY - JOUR.

Foster, N., & Schwab, M. (2000). Real-time 32-bit microcontroller with OSEK/VDX operating system support. *SAE Journal Transactions*, *109*(3), 1470–1476.

Freescale (1996). *TPU Time Processor Unit Reference Manual (including the TPU2)*. Freescale Semiconductor Inc.

Freescale (2007). Knock detection: Combining efficient architecture and advanced peripherals for quality solutions. Email newsletter.

Gan, Q., & Rosales, E. (2003). CMAC with linear functional weights. In *Proceedings of the 13th IFAC Symposium on System Identification (SYSID '03)*, (pp. 1838–1843). Rotterdam, The Netherlands: Elsevier.

Gassenfeit, E., & Powell, D. (1989). Algorithms for air-fuel ratio estimation using internal combustion engine cylinder pressure. *SAE Journal Transactions*, *98*(3), 351–356.

Gerasimov, D. N., Javaherian, H., & Nikiforov, V. O. (2011). Data driven inverse-model control of si engines. In *Proc. American Control Conf. (ACC)*, (pp. 426–431).

Gilkey, J., & Powell, D. (1985). Fuel-air ratio determination from cylinder pressure time histories. *Transactions of the ASME; Journal of Dynamic Systems, Measurement, and Control*, *107*.

Glass, J. W., & Franchek, M. A. (1999). NARMAX modelling and robust control of internal combustion engines. *International Journal of Control*, *72*(4), 289–304.

Hansson, H., Lawson, H., Bridal, O., Eriksson, C., Larsson, S., Lon, H., & Stromberg, M. (1997). BASEMENT: an architecture and methodology for distributed automotive real-time systems. *IEEE Transactions on Computers*, *46*(9), 1016–1027.

Hansson, H. A., Lawson, H. W., Stromberg, M., & Larsson, S. (1995). BASEMENT: a distributed real-time architecture for vehicle applications. In *Proc. Real-Time Technology and Applications Symp*, (pp. 220–229).

Haris, P. (2003). PC/104 specification version 2.5. Electronic PDF.

Hendricks, E. (1997). Engine modelling for control applications: A critical survey. *Meccanica*, *32*(5), 387–396. TY - JOUR.

Hendricks, E. (2000). A generic mean value engine model for spark ignition engines. In *41st Simulation Conference, SIMS 2000*. DTU.

Hendricks, E., Chevalier, A., Jensen, M., Sorenson, S. C., Trumpy, D., & Asik, J. (1996). *Modeling of the Intake Manifold Filling Dynamics*, vol. SP-1149, chap. 960037. Society of Automotive Engineers.

Hendricks, E., & Sorenson, S. C. (1990). Mean value modelling of spark ignition engines. *SAE Journal Transactions*, *99*(3), 1359–1373.

Hendricks, E., Vesterholm, T., & Sorenson, S. C. (1992). Nonlinear, closed loop, si engine control observers. *SAE Journal Transactions*, (3), 326–343.

Hernandez, C. A. (2007). DSP-based engine knock detection including knock sensor and circuit diagnostics.

Hochschwarzer, H., Kriegler, W., & Schon, M. (1992). Fully automatic determination and optimization of engine control characteristics. *SAE Journal Transactions*, *101*(920255), 380–391. SAE 920255.

Hosey, J., & Powell, D. (1979). Closed loop, knock adaptive spark timing control based on cylinder pressure. *Transactions of the ASME; Journal of Dynamic Systems, Measurement, and Control*, *101*, 64–70.

Hubbard, M., Dobson, P., & Powell, D. (1976). Closed loop control of spark advance using a cylinder pressure sensor. *Transactions of the ASME; Journal of Dynamic Systems, Measurement, and Control*, *98*, 414–420.

Jaguar (1996). *AJ-V8 Engine and 5HP24 Transmission Introduction*. Service Communications, Jaguar Cars Ltd.

Jamieson, R. A. (1968). Eddy-current effects in solid, unslotted iron rotors. *Proceedings of the Institution of Electrical Engineers*, *115*(6), 813–820.

Jayasuriya, S., & Franchek, M. A. (1991). Frequency domain design for maximal rejection of persistent bounded disturbances. *Transactions of the ASME; Journal of Dynamic Systems, Measurement, and Control*, *113*, 195–205.

Kaidantzis, P., Rasmussen, P., Jensen, M., Vesterholm, T., & Hendricks, E. (1993). Advanced nonlinear observer control of SI engines. *SAE Journal Transactions*, (pp. 1029–1037).

Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME; Journal of Basic Engineering*, *82*, 35–45.

King, P., Burnham, K. J., D.J.G., J., Norton, J., & Sharpe, S. R. (1991). Implementation of a self-tuning controller to the dynamometer torque loop of an engine test cell. In *International Conference on Control 1991*, vol. 1, (pp. 110–114).

Kopetz, H., Damm, A., Koza, C., Mulazzani, M., Schwabl, W., Senft, C., & Zainlinger, R. (1989). Distributed fault-tolerant real-time systems: the Mars approach. *IEEE Micro*, *9*(1), 25–40.

Koustas, J. (1984). *Optimal Control of Engine Test-beds by Microcomputer Networks*. Ph.D. thesis, Thermal Power Section, Department of Mechanical Engineering, Imperial College of Science and Technology.

Koustas, J., & Watson, N. (1984). A transient diesel test bed with direct digital control. In *SAE Journal Transactions*. SAE 840347.

Krishnaswami, V., & Rizzoni, G. (1995). Nonlinear parity equation based residual generation for diagnosis of automotive engine faults. *Control Engineering Practice*, *3*(10), 1385–1392.

Kulreja, S. L., Galiana, H., & Kearney, R. (1999). Structure detection of narmax models using bootstrap method. In *Proceedings of the 38th Conference on Decision and Control*, vol. 1, (pp. 1071–1076). Phoenix, AZ, USA.

Kwak, H. J., Sung, S. W., Lee, I.-B., & Park, J. Y. (1999). A modified smith predictor with a new structure for unstable processes. *Industrial & Engineering Chemistry Research*, *38*(2), 405–411.
URL http://pubs.acs.org/doi/abs/10.1021/ie980515n

Larsson, S. (2003). *SI-Engine spark advance control using torque sensors*. Thesis for the degree of licentiate of engineering, Control and Automation Laboritory, Department of Signals and Systems, Chalmers University of Technology, University of Gothenborg, Gothenburg, Sweden.

Larsson, S., & Schagerberg, S. (2004). SI engine cylinder pressure estimation using torque sensors. *Electronic Engine Controls 2004*, *SP-1822*, 401–410. Features ABB torque sensor.

Lee, S., Howlett, R., & Walters, S. (2003). Fuzzy air-fuel ratio control of a small gasoline engine.

Lelic, M., & Gajic, Z. (2002). *Intelligent Vehicle Technologies*, chap. 9, Adaptive Control System Techniques, (pp. 259–287). Butterworth Heinemann. ISBN 0-7506-5093-1.

Lennox, B., & Montague, G. (2001). *Nonlinear predictive control: theory and practice*, chap. Chapter 12: Neural network control of a gasoline engine with rapid sampling, (pp. 245–255). The Institution of Electrical Engineers.

Leontaritis, I., & Billings, S. (1985a). Input-output parametric models for non-linear systems. part i: Deterministic non-linear systems. *International Journal of Control*, *41*(2), 303–328.

Leontaritis, I., & Billings, S. (1985b). Input-output parametric models for non-linear systems part ii: Stochastic non-linear systems. *International Journal of Control*, *41*(2), 329–344.

Leontaritis, I., & Billings, S. (1987). Experimental design and identifiability for non-linear systems. *International Journal of Systems Science*, *18*(1), 189–202. Liverpool Harold Cohen has holdings.

Lida, K., Katsuo, A., & Kido, K. (1990). Imep estimation from instantaneous crankshaft torque variation. *SAE Journal Transactions*, (pp. 1374–1385).

Lumsden, G., Browett, C., Taylor, J., & Kennedy, G. (2004). Mapping complex engines. In *Direct Injection SI Engine Technology 2004*, vol. 1, (pp. 107–118).

Majors, M., Stori, J., & Cho, D.-i. (1994). Neural network control of automotive fuel-injection systems. *IEEE Control Systems*, (pp. 31–36).

Manzie, C., Palaniswami, M., Daniel Ralph, D., Watson, H., & Yi, X. (2002). Model predictive control of a fuel injection system with a radial basis function network observer. *Transactions of the ASME; Journal of Dynamic Systems, Measurement, and Control*, *124*(2000-01-1248), 648–658.

Marcin, J. (1998). Thermocouple signal conditioning using the ad594/ad595. Electronic application note.

McKay, D., Nichols, G., & Schreurs, B. (2000). Delphi electronic throttle control systems for model year 2000; driver features, system security, and OEM benefits. ETC for the mass market. In *Electronic Engine Controls 2000: Controls (SP-1500)*, 2000-01-0556. Detroit, Michigan: SAE Technical Paper Series; SP-1500.

Melgaard, H., Hendricks, E., & Madsen, H. (1990). Continuous identification of a four-stroke si engine. In *Proc. American Control Conf*, (pp. 1876–1881).

Mills, R. A. (1985). A high-level language implementation of an engine control strategy. *IEEE Transactions on Industrial Electronics*, *32*(5), 313–317.

MISRA (1994). Development guidelines for vehicle based software. URL http://www.misra.org.uk

Montanaro, J., Lee, T. H., Witek, R. T., Lin, P. C. M., Anne, K., Madden, L., Black, A. J., Murray, D., Cooper, E. M., Pearce, M. H., Dobberpuhl, D. W., Santhanam, S., Donahue, P. M., Snyder, K. J., Eno, J., Hoeppner, R. S. G. W., Thierauf, S. C., & Kruckemyer, D. (1996). A 160-MHz, 32b, 0.5-w CMOS RISC microprocessor. *IEEE Journal of Solid-State Circuits*, *31*(11), 1703–1714. Also reprinted in Digital Technical Journal, Volume 9, Number 1, 1997. pp. 49-62.

Motorola (1990). *Time Processor Unit Reference Manual TPURM/AD*. Motorola, first ed.
URL http://www.eslave.net/

MPL-AG (2004). *PATI Preliminary User Manual*. MPL AG Switzerland.

Mrabet, M., Fnaiech, F., Chaari, A., & Al-Haddad, K. (2002). Nonlinear predictive control based on narx models with structure identification. *IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the]*, *3*, 1757–1762.

Naber, J. D., & Rajagopalan, S. R. (2008). Combustion knock detection and control through statistical characterization of knock levels.
URL http://www.patentgenius.com/patent/7415347.html

National-Semiconductor (1995). LM1949 injector drive controller. Electronic PDF.

National-Semiconductor (2000). LM2907/LM2917 frequency to voltage converter. Electronic PDF datasheet.

National-Semiconductor (2001). LM9040 dual lambda sensor interface amplifier. Electronic PDF.

National-Semiconductor (2005). LM1815 adaptive variable reluctance sensor amplifier. Electronic PDF.

Nayfeh, A. H., & Mook, D. T. (1995). *Nonlinear Oscillations*. Wiley Classics Library. John Wiley & Sons Inc.

Nielsen, L., & Eriksson, L. (1998). An ion-sense engine fine-tuner. *IEEE Control Systems*, (pp. 43–52).

Noble, A. D., Beaumont, A. J., & Mercer, A. S. (1988). Predictive control applied to transient engine testbeds. *SAE Journal Transactions*, *97*, 6.798–6.803. SAE 880487.

Nomura, M., Suzuki, M., Hori, M., & Terashima, M. (2000). Decoupling torque control system for automotive engine tester. *36*(2), 467–474.

Nuno, M. C. d. O., & Biegler, L. T. (1994). Constraint handing and stability properties of model-predictive control. *AIChE Journal*, *40*(7), 1138–1155.

Ogata, K. (1997). *Modern Control Engineering*. Prentice Hall. ISBN 0-13-227307-1.

P3-America (2005). Potentiometers: Hall effect precision potentiometer - series mp1613. Electronic PDF, San Diego, California.
URL http://www.p3america.com/pp/pdfs/mp1613.pdf

Palmer, D. (2004). Torqueing sense to the outside world. *Eureka on campus*, *2004*(Summer), 10–11.

Panda, R. C., Hung, S.-B., & Yu, C.-C. (2006). An integrated modified smith predictor with pid controller for integrator plus deadtime processes. *Industrial & Engineering Chemistry Research*, *45*(4), 1397–1407.
URL http://pubs.acs.org/doi/abs/10.1021/ie0580194

Park, S., Yoon, M., & Sunwoo, M. (2003). Feedbackerror learning neural networks for air-to-fuel ratio control in si engines. *SAE Journal Transactions*, (pp. 522–526).

Passaquay, D., Boverie, S., Heredia, G., Ollero, A., Titli, A., & Aracil, J. (2001). Fuzzy modelling, control, and stability analysis of an automotive engine. In *Advances in Automotive Control 2001. Proceedings of the 3rd IFAC Workshop*, (pp. 223–237).

Pereira, J. M. D., Postolache, O., & Girao, P. S. (2005). A self-adaptable method to optimize the performance of frequency-to-code conversion based measurement systems. In *Proc. IEEE Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications IDAACS 2005*, (pp. 295–298).

PLX-Technology-Inc (2002). PCI 9056 product overview datasheet 9056-sil-pb-p2-1.0. Electronic PDF.

Pohl, R. (1949). Rise of flux due to impact excitation: retardation by eddy currents in solid parts. *Proceedings of the IEE -Part II: Power Engineering*, *96*(49), 57–65.

Powell, D. (1993). Engine control using cylinder pressure: Past, present, and future. *Transactions of the ASME; Journal of Dynamic Systems, Measurement, and Control*, *115*, 343–350.

Powell, D., Fekete, N., & Chang, C.-F. (1998). Observer-based air-fuel ratio control. *IEEE Control Systems*, (pp. 72–83).

Rajagopalan, S. R. (2006). *Experimental Measure and Analysis for Determination of Combustion Knock Intensity in a Spark Ignition Engine*. Master's thesis, Mechanical Engineering, Michigan Technological University, Houghton, MI.

Rizzoni, G. (1989). Estimate of indicated torque from crankshaft speed fluctuations: A model for the dynamics of the ic engine. *IEEE Transactions on Vehicular Technology*, *38*(3), 168–179. TY - JOUR.

Saerens, B., Diehl, M., Swevers, J., & Van den Bulck, E. (2008). Model predictive control of automotive powertrains - first experimental results. In *Proc. 47th IEEE Conf. Decision and Control CDC 2008*, (pp. 5692–5697).

Sandberg, T. (2001). Modeling and validation of traveling resistance for heavy trucks. *Electronic engine controls 2001; modeling, controls, OBD and neural networks*, *SP-1585*, 127–133.

Scotson, P., & Wellstead, P. (1990). Self-tuning optimization of spark ignition automotive engines. *IEEE Control Systems Magazine*, *10*(3), 94–101.

Shafai, E., Bianchi, M., & Geering, H. P. (2003). detectNARMAX: A graphical user interface for structure detection of narmax models using the bootstrap method. In *Proceedings of the 13th IFAC Symposium on System Identification*, (pp. 1013–1018).

Shiraishi, H., Ipri, S., & Cho, D.-i. (1995). CMAC neural network controller for fuel-injection systems. *IEEE Transactions on Control Systems Technology*, *3*(1), 32–38.

Silviero, C., Scattolini, R., Gelmetti, A., Poggio, L., & Serra, G. (1995). Analysis & validation of mean value models for si ic-engines. In *Advances in Automotive Control: Postprint Volume from the IFAC Workshop, Ascona, Switzerland, 13-17 March 1995*. Pergamon.

Smart, J., Hock, K., & Csomor, S. (2005). *Cross-Platform GUI Programming with wxWidgets*. Prentice Hall.

Sobel, J. R., Jeremiasson, J., & Wallin, C. (1996). Instantaneous crankshaft torque measurement in cars. *Electronic Engine Controls 1996 (SP-1149)*, *SP-1149*.

Srodawa, R. J., Gach, E., Rodger, & Glicker, A. (1985). Preliminary experience with the automatic generation of production-quality code for the ford/intel 8061 microprocessor. *IEEE Transactions on Industrial Electronics*, *32*(4), 318–326.

Steinbrenner, U., Barho, H., Bottcher, K., Gandert, V., Gollin, W., Haming, W., Joos, K., Mezger, M., Peter, B., & Wild, E. (1994). *Motronic Engine Management*. Stuttgart: Robert Bosch Gmbh, 3rd ed.

Tough, J. (2002). Pi Technology Limited, unpublished internal document.

Ulrich, O., Wlodarczyk, R., & Wlodarczyk, M. T. (2001). High-accuracy low-cost cylinder pressure sensor for advanced engine controls. *SAE Journal Special Publication*, *SP-1586*, 61–68.

Uras, M. H. (2001). Magnetostrictive dynamic strain sensor. *Sensor and Actuators 2001 (SP-1609)*, (2001-01-0617).

Vector (2004). Introduction to the can calibration protocol. Tech. Rep. AN-AMC-1-102, Vector CANtech Inc.

Wang, Y.-Y., Krishnaswami, V., & Rizzoni, G. (1997). Event-based estimation of indicated torque for ic engines using sliding-mode observers. *Control Engineering Practice*, *5*(8), 1123–1129. TY - JOUR.

Ward, M., Brace, C., Vaughan, N., Ceen, R., Hale, T., & Kennedy, G. (2002). Investigation of 'sweep' mapping approach on engine testbed. *SAE Journal Transactions*, (pp. 1130–1135).

Wellstead, P. E., & Zanker, P. (1981). *Application of self-tuning to engine control*, chap. 12, (pp. 282–295). IEE Control engineering series 15. Institution of Engineering and Technology, Peter Peregrinus Ltd.

Wendeker, M., & Czarnigowski, J. (2000). Hybrid air/fuel ratio control using the adaptive estimation and neural network. *SAE Journal Transactions*, *109*(3), 1477–1484.

Williams, C. (2002). Linux schedular latency. Electronic pdf, Redhat Inc. URL www.linuxfordevices.com/files/article027/rh-rtpaper.pdf

Willner, C. A. (2006). Fiber optic cylinder pressure measurement system for a combustion engine.

Won, M., Choi, S. B., & Hedrick, J. K. (1998). Air-to-fuel ratio control of spark ignition engines using gaussian network sliding control. *IEEE Transactions on Control Systems Technology*, *6*(5), 678–687.

Wright, M. T. (1972). *Steady-state and Transient Performance of Eddy-current Couplings*. Ph.D. thesis, The University of Aston in Birmingham.

Yodaiken, V. J. (1999). Adding real-time support to general purpose operating systems.

Yurish, S. (2007). High-speed universal frequency-to-digital converter for quasi-digital sensors and transducers. *Sensors & Transducers*, *80*(6), 1225–1229.

Appendix A. Circuit Schematic Diagrams

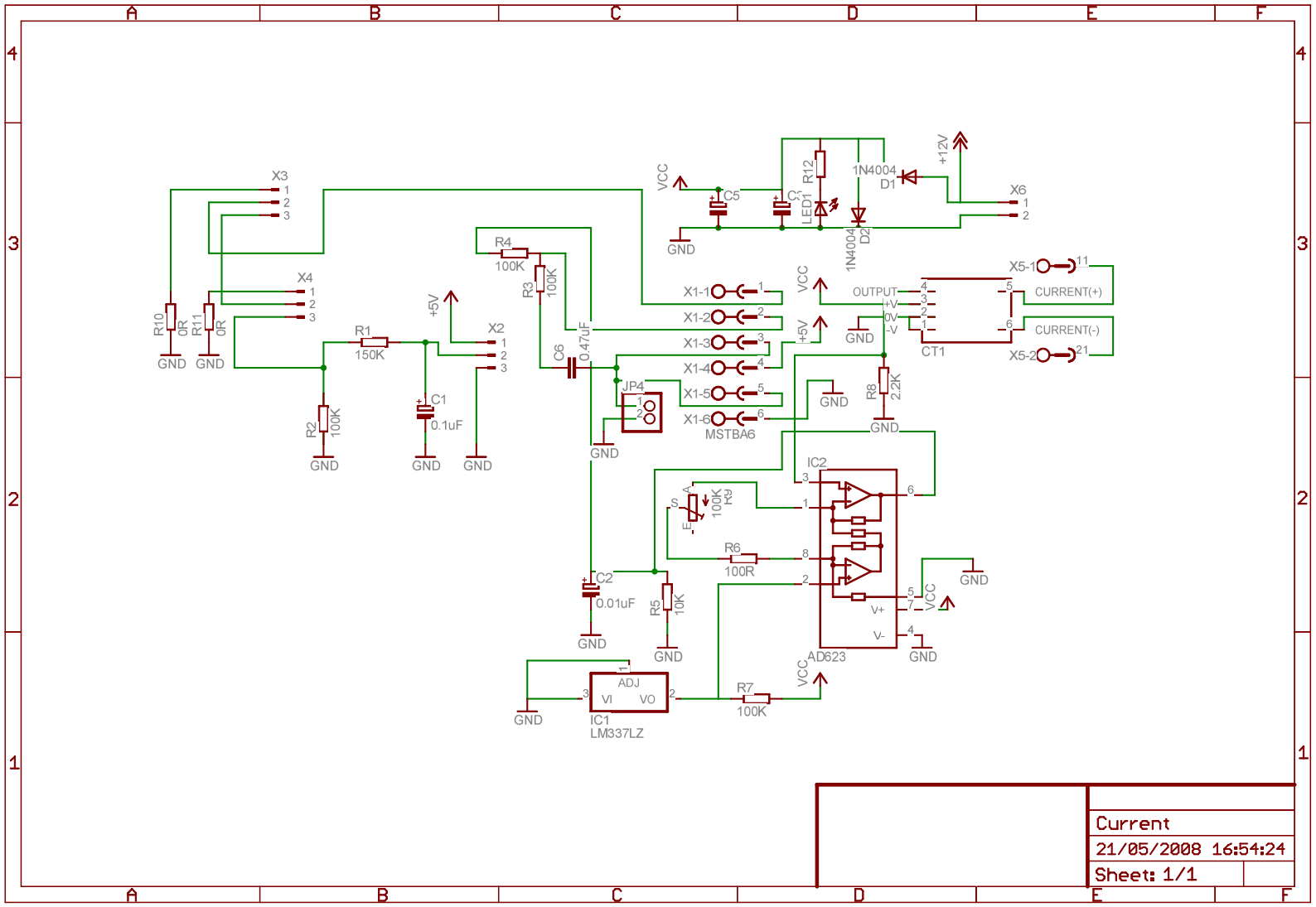

Figure A.1: Current controller board schematic

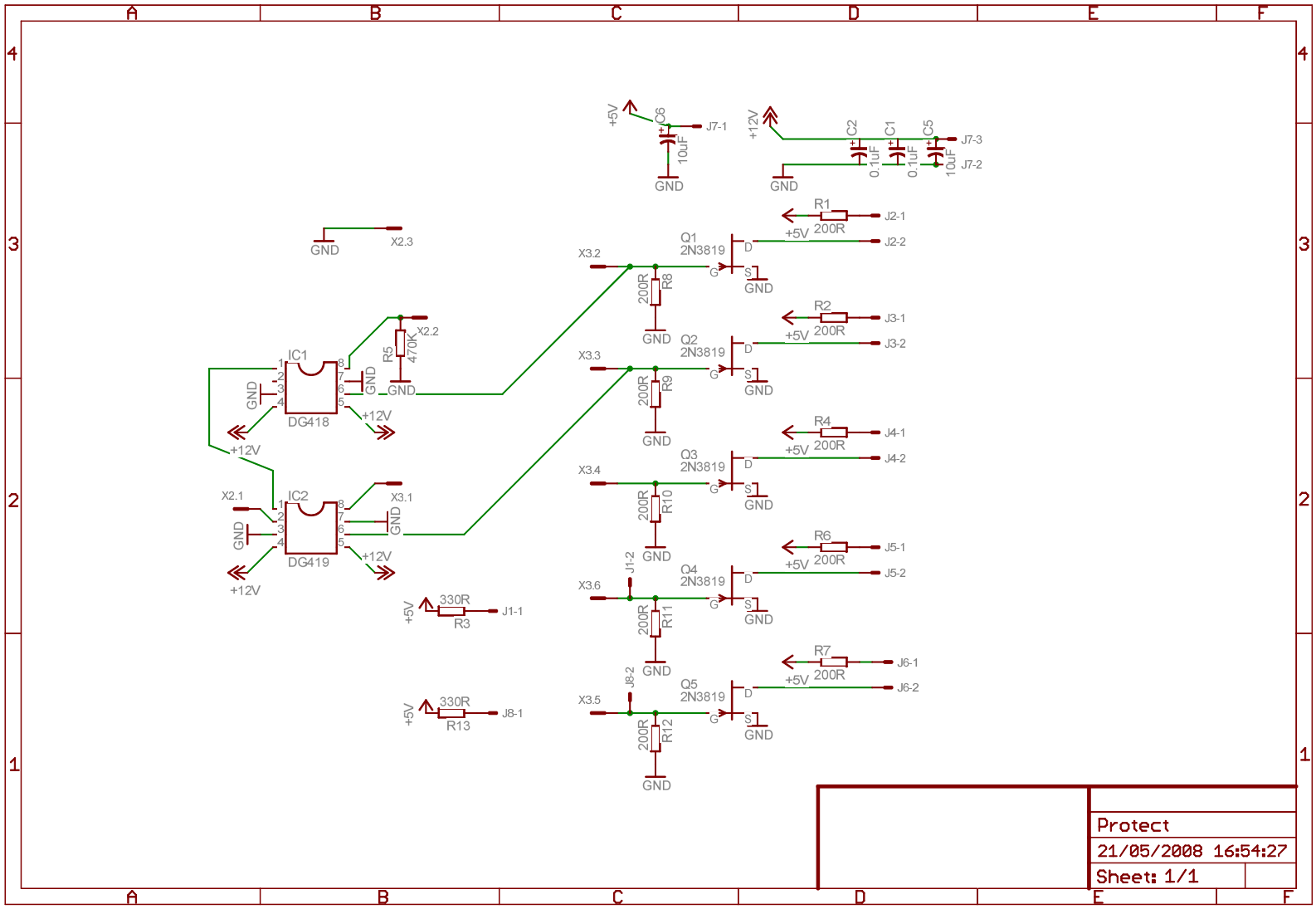

Figure A.2: Protection circuit schematic

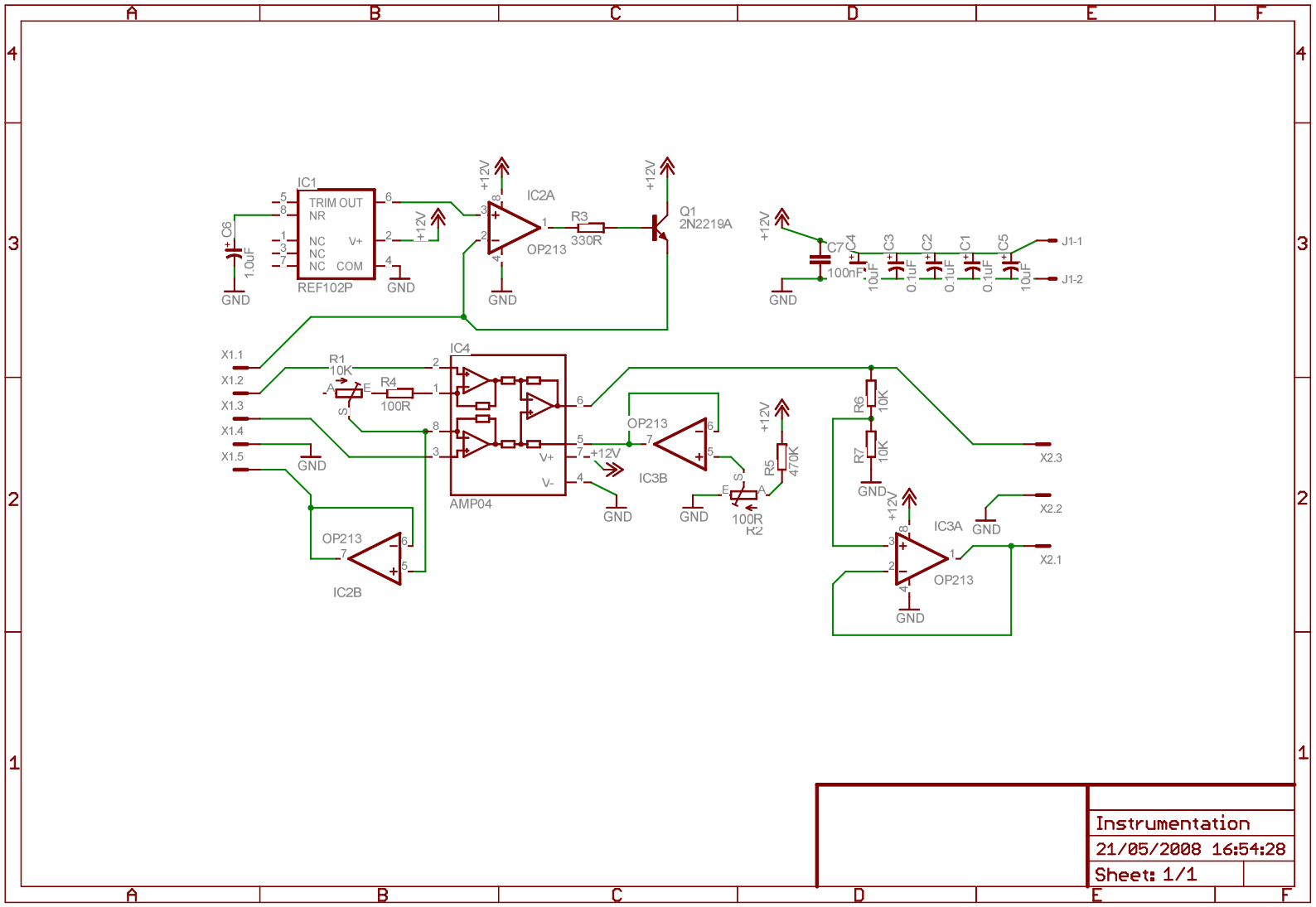

Figure A.3: Load cell amplification circuit schematic

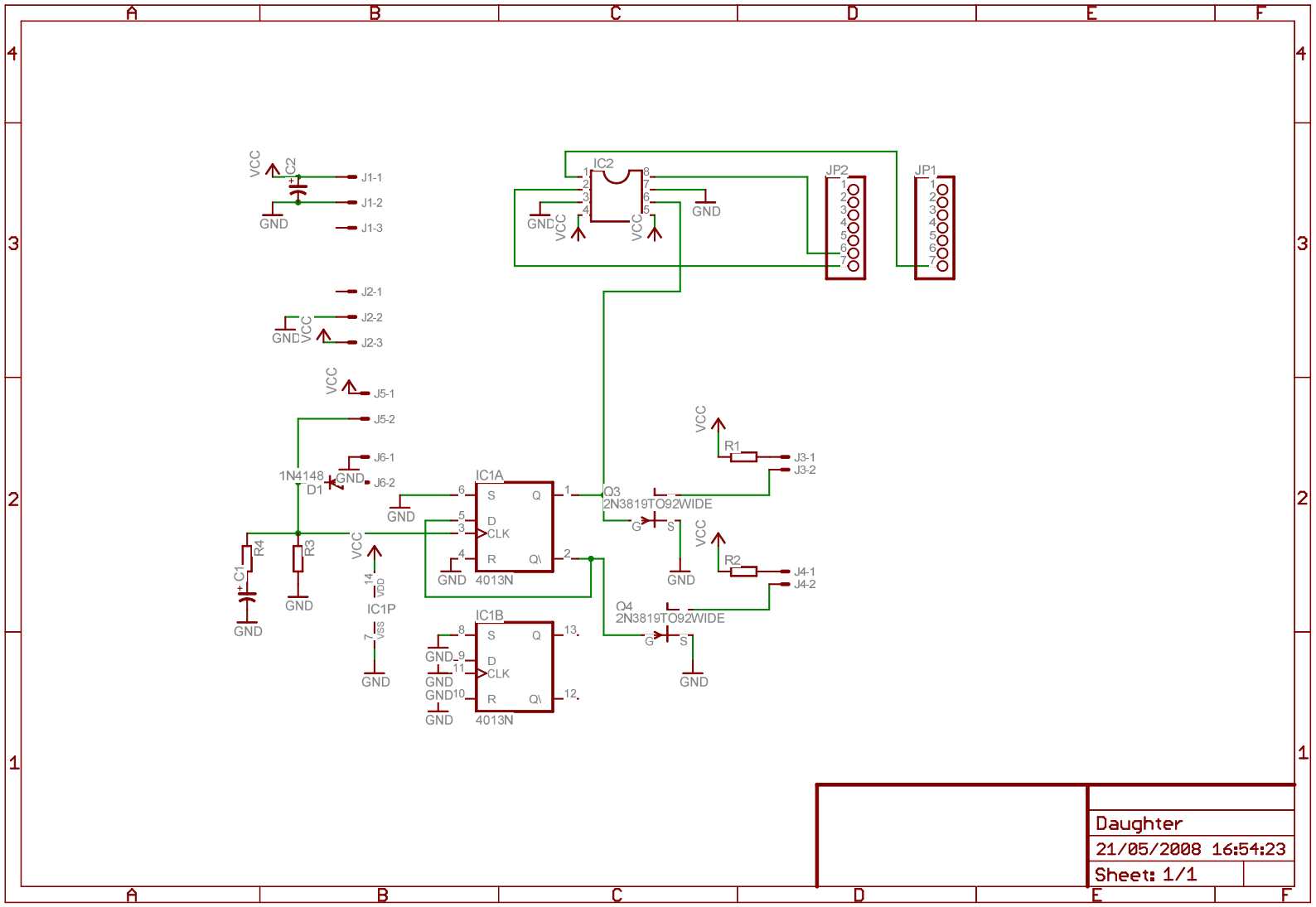

Figure A.4: Plint meter daughter board schematic

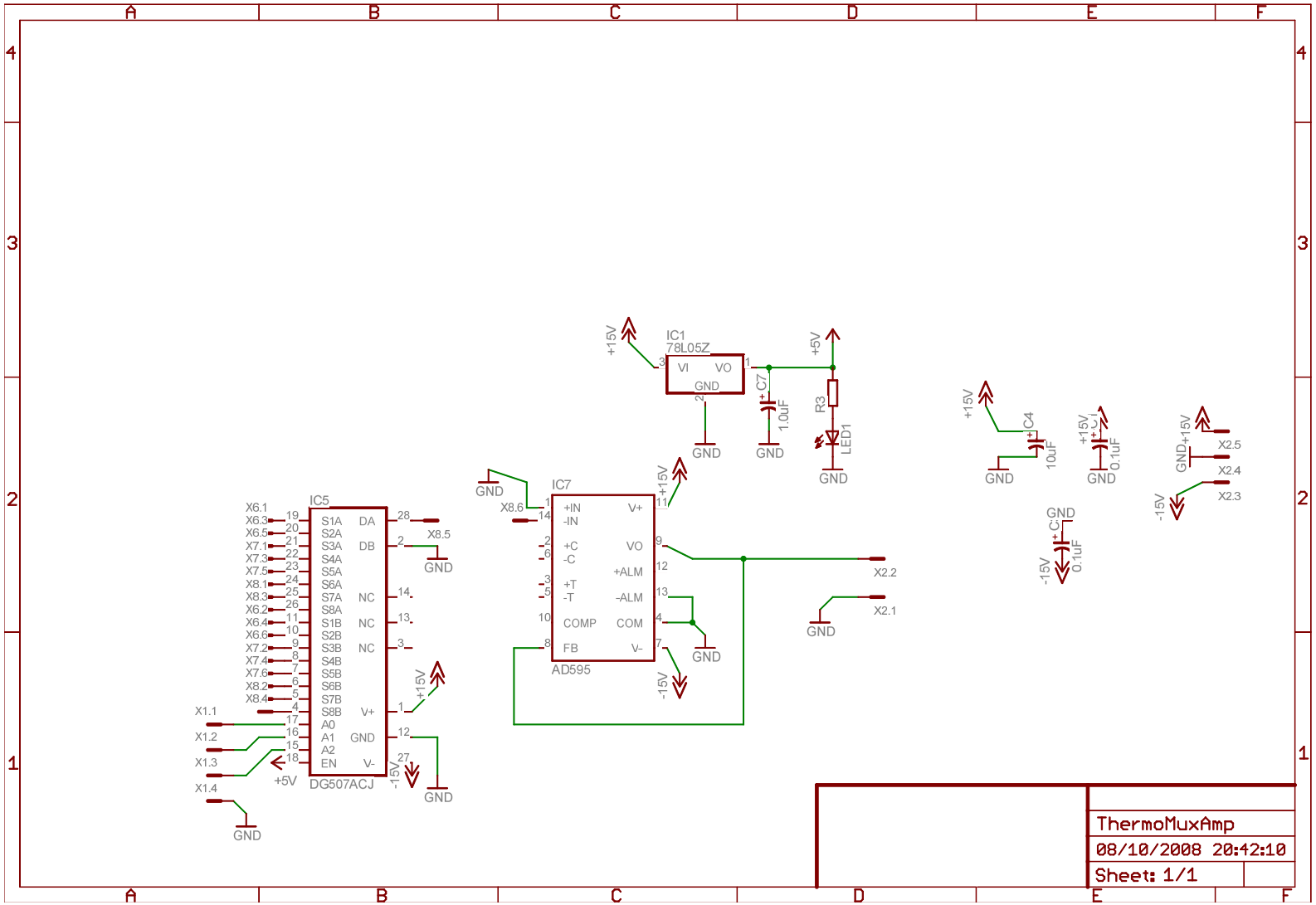

Figure A.5: Thermocouple amplification board schematic

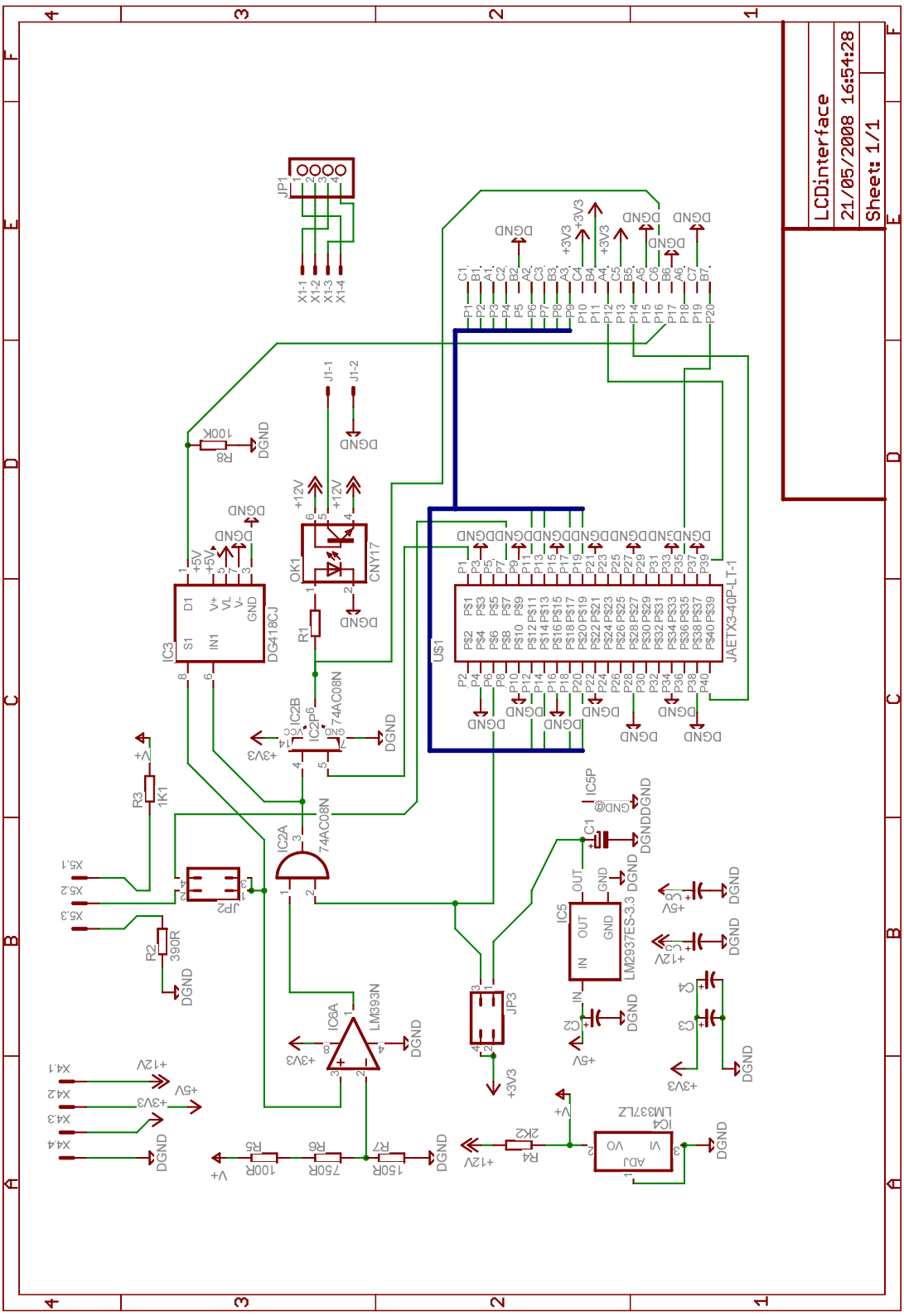

Figure A.6: LCD interface board schematic

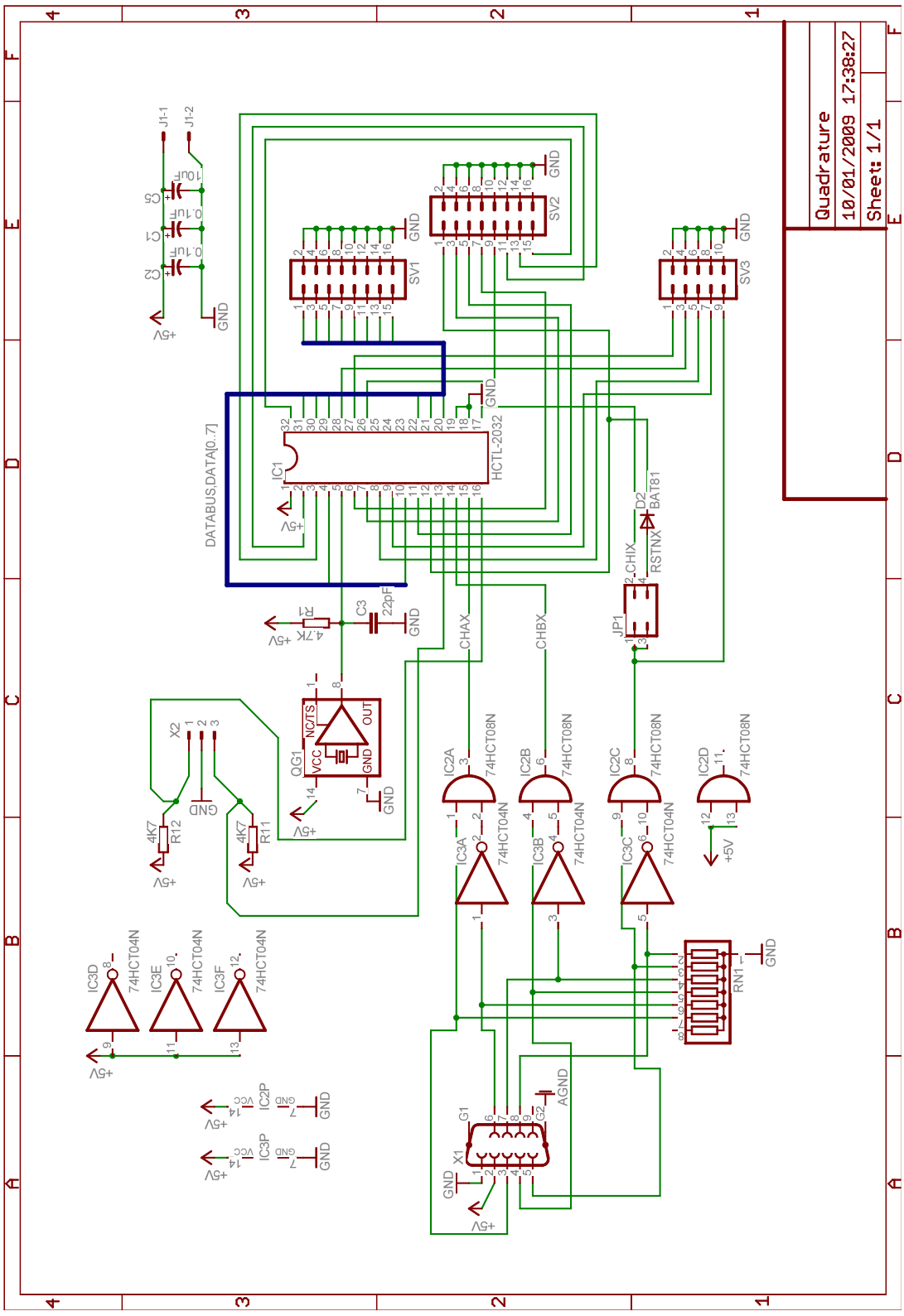

Figure A.7: Encoder quadrature interface circuit schematic

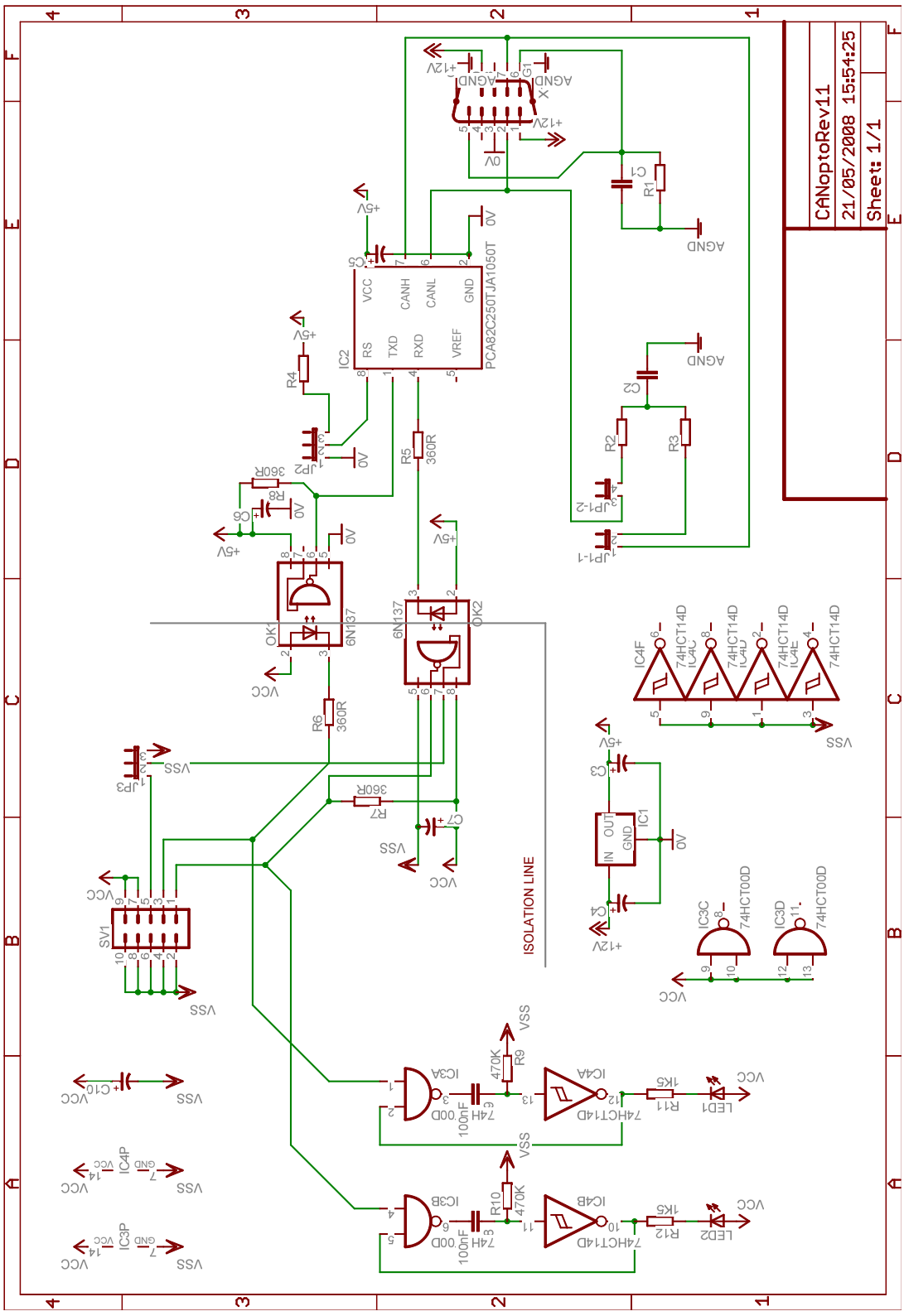

Figure A.8: CAN bus transceiver board circuit schematic

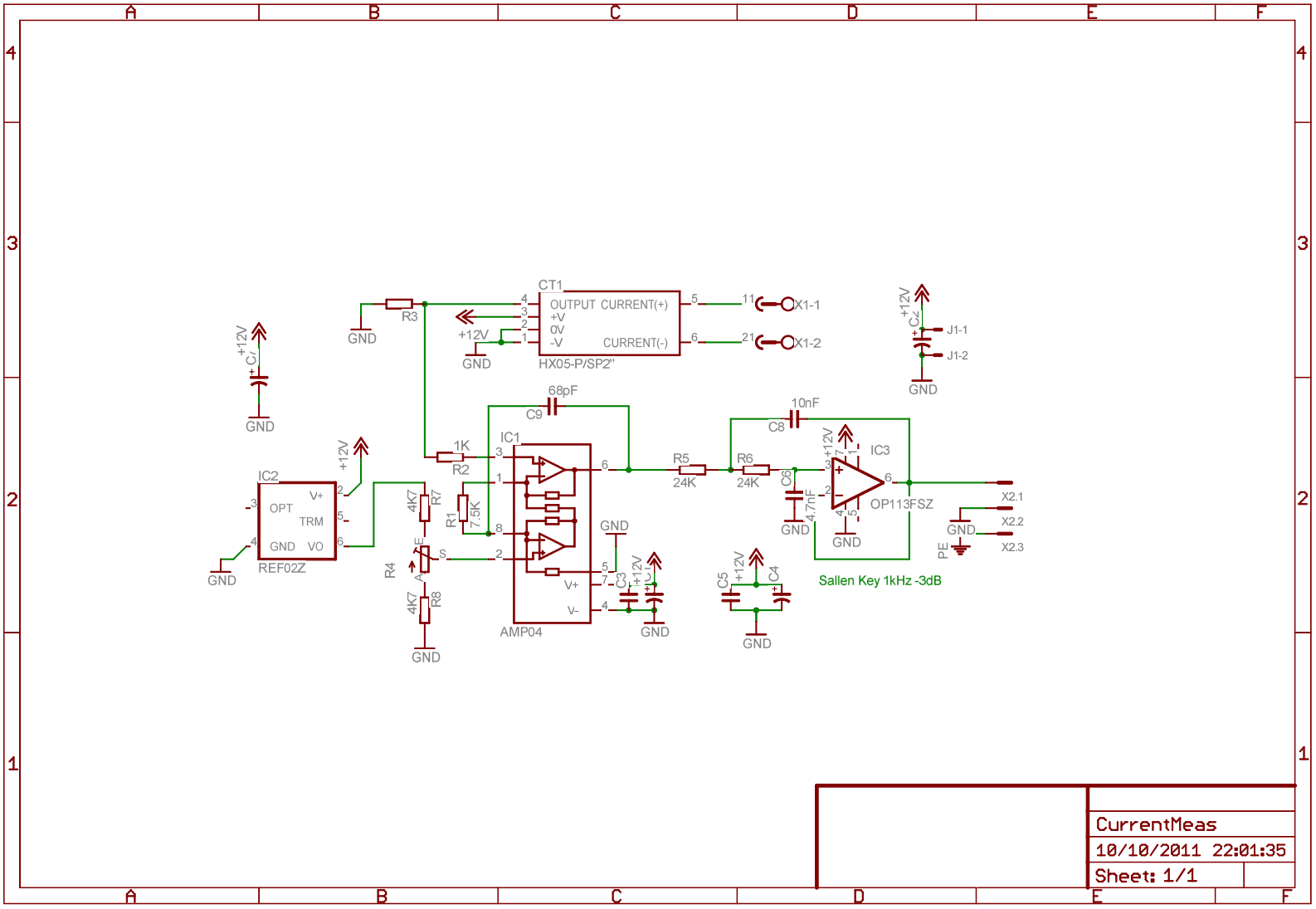

Figure A.9: Current transducer board circuit schematic

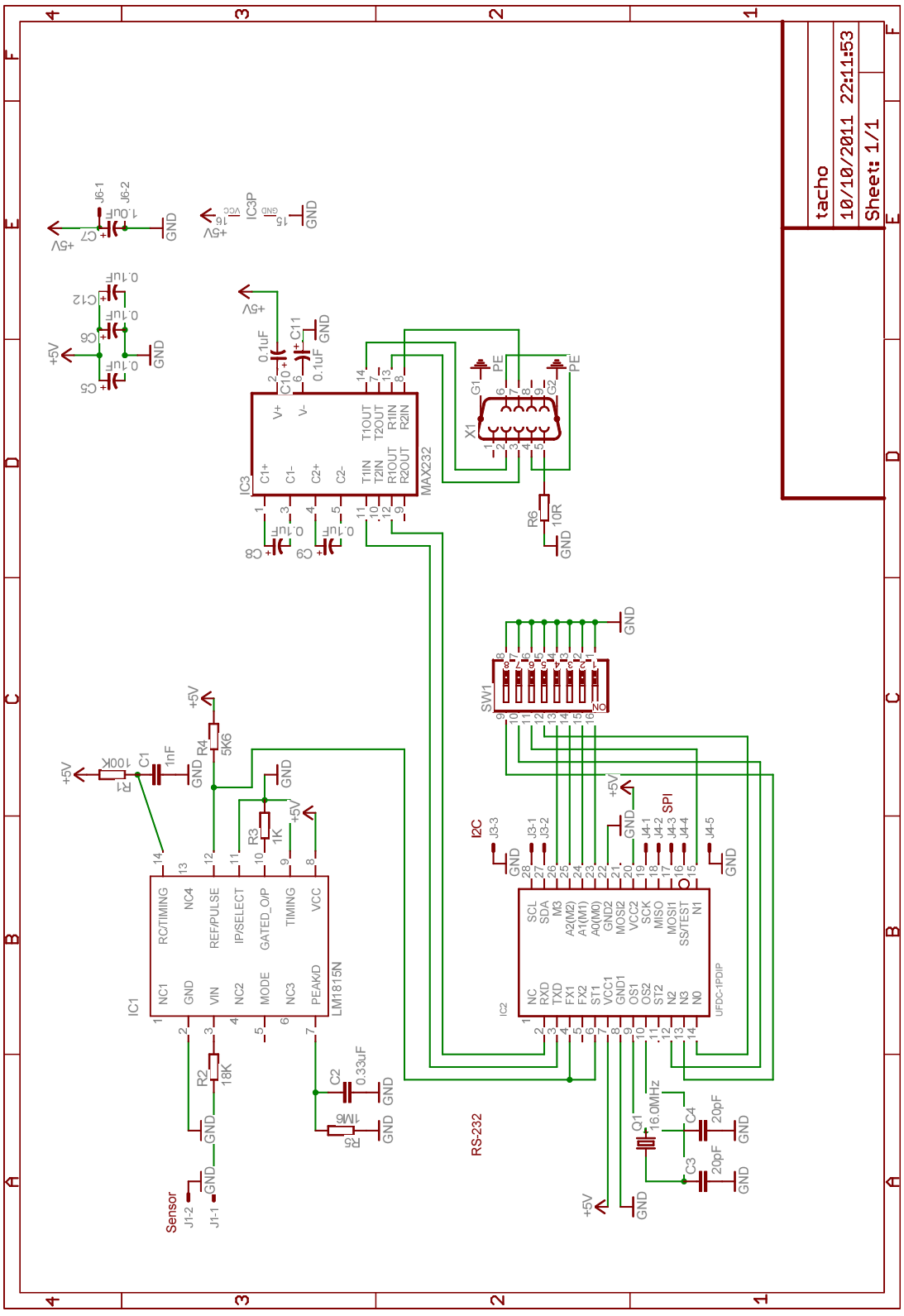

Figure A.10: Digital tachometer board circuit schematic

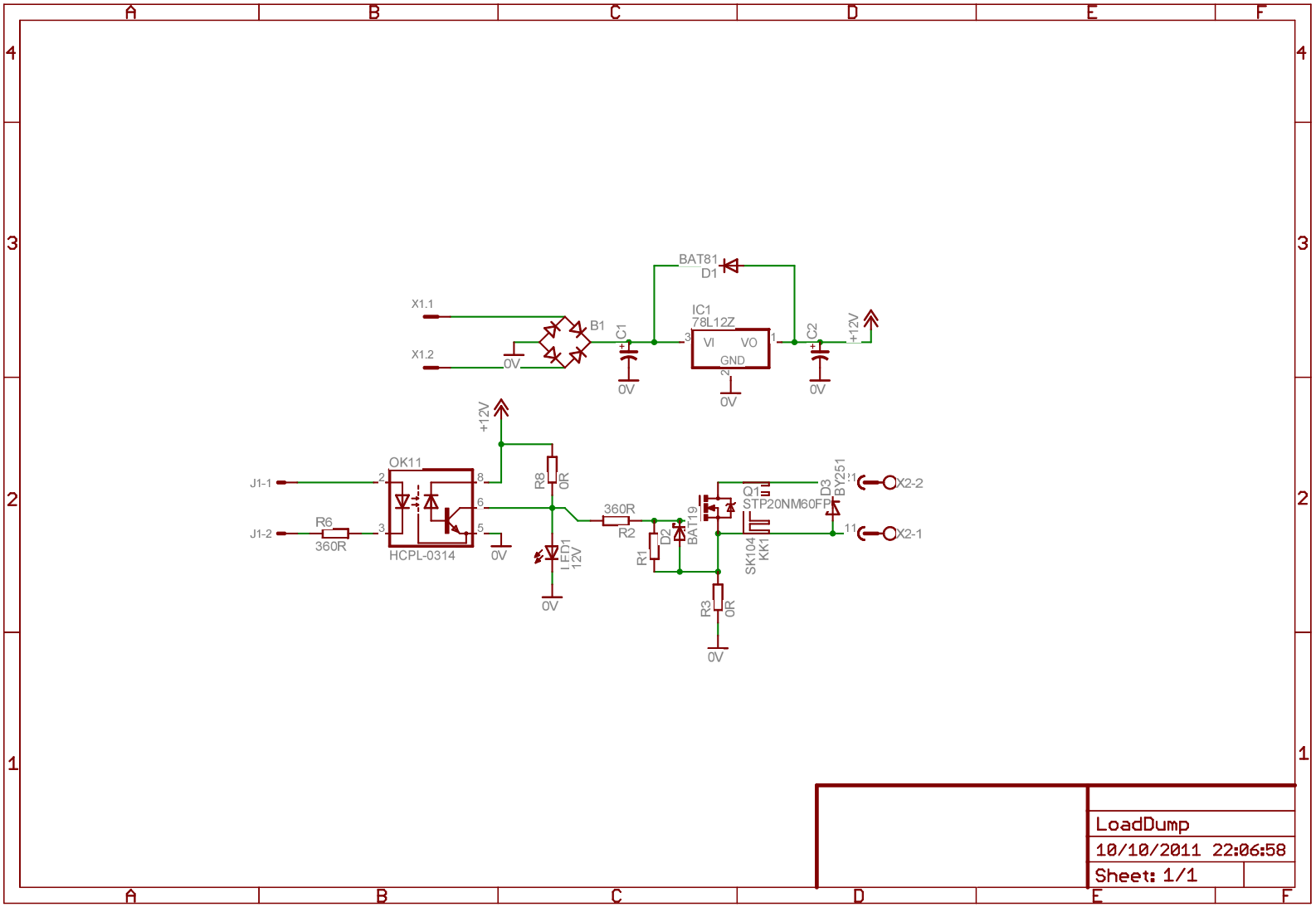

Figure A.11: Load dump board circuit schematic

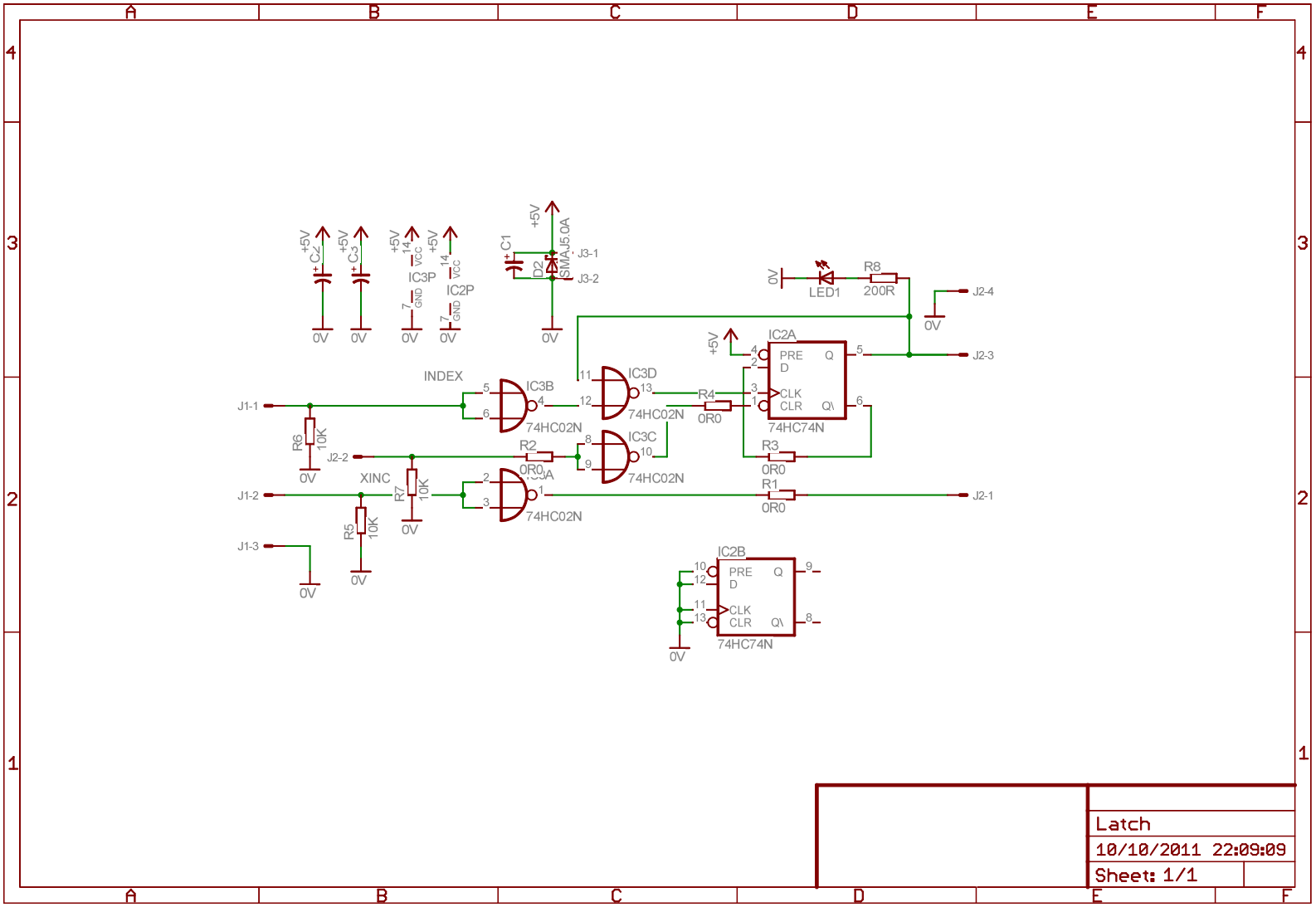

Figure A.12: Encoder index latch board circuit schematic

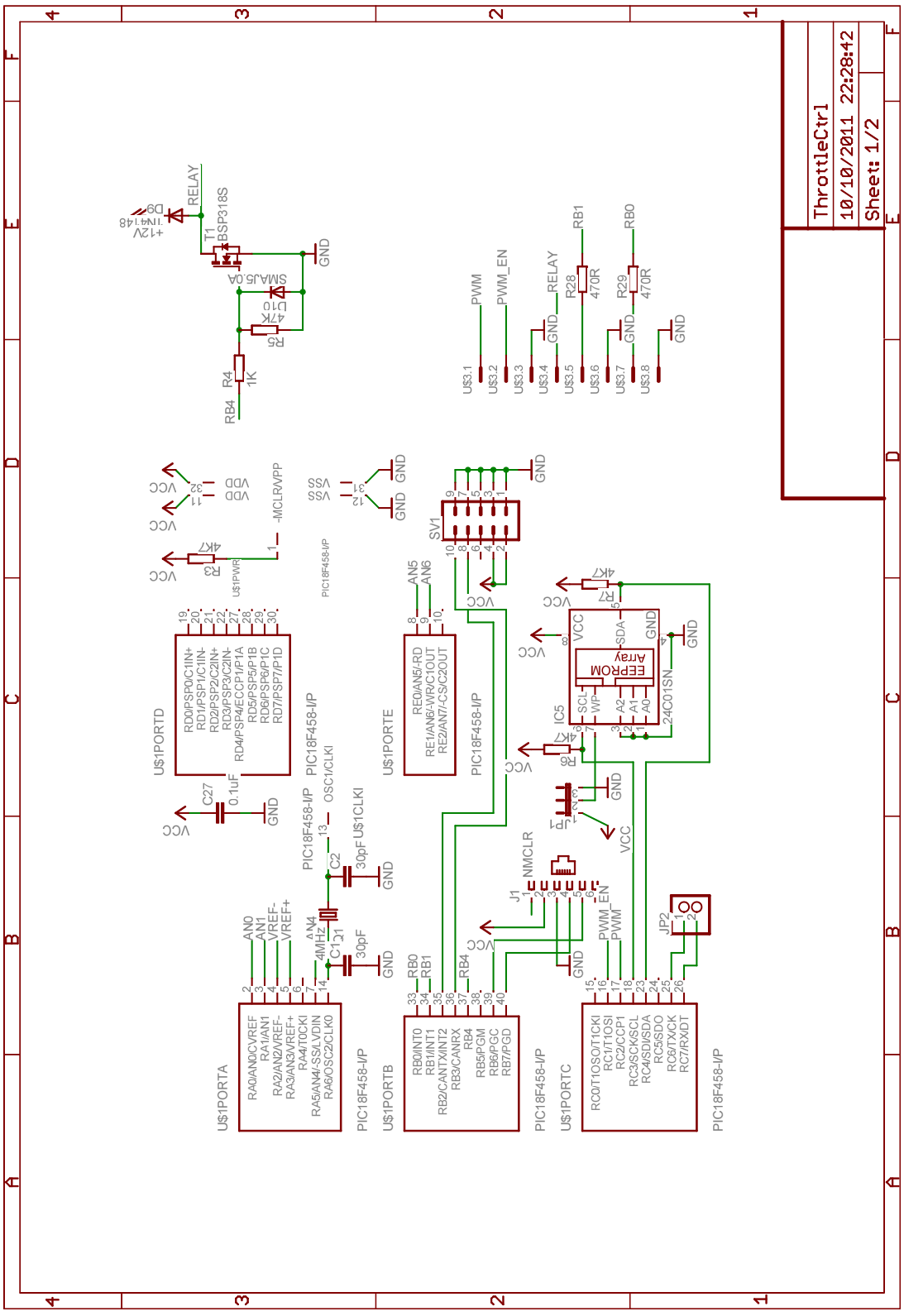

Figure A.13: Throttle controller board circuit schematic 1

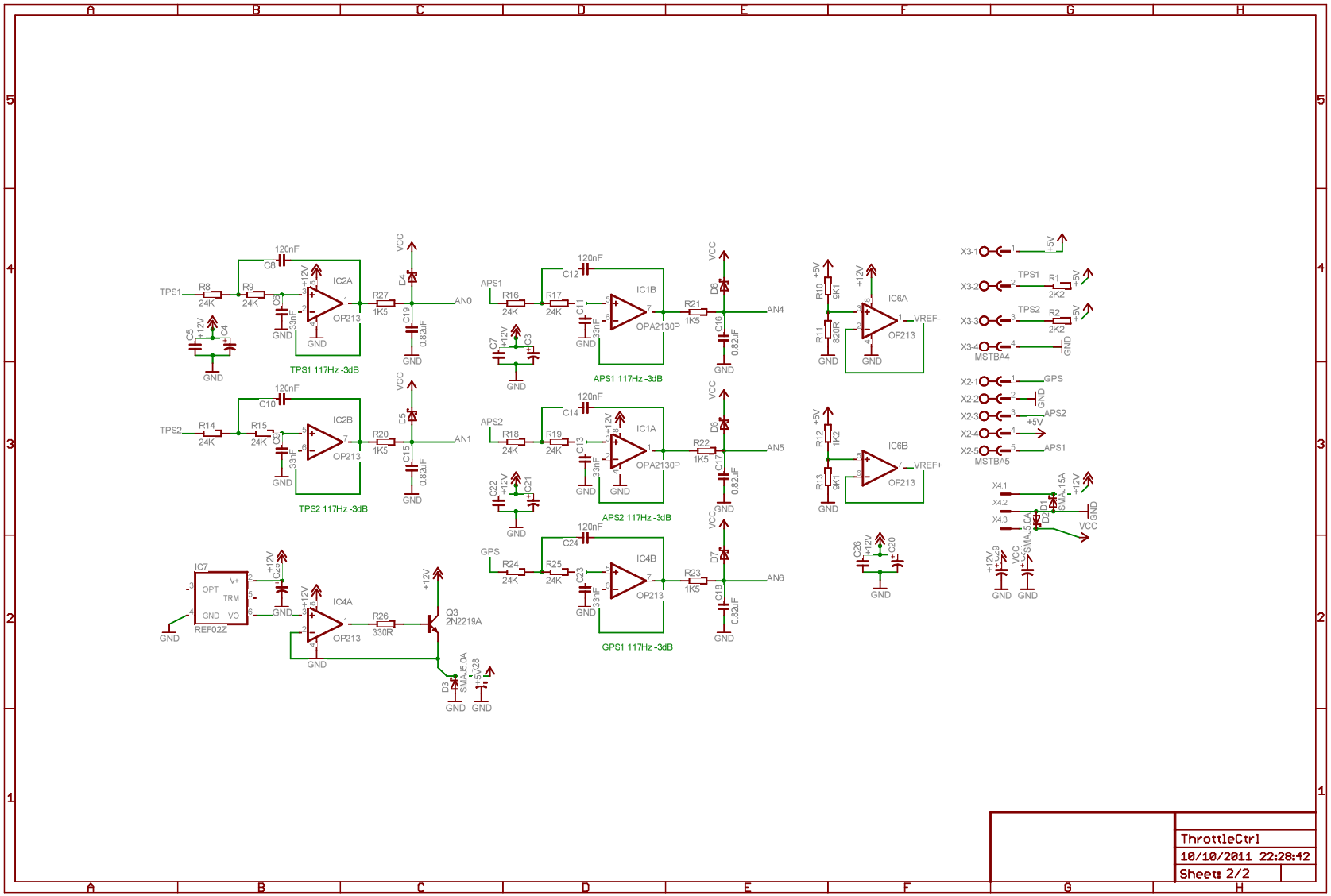

Figure A.14: Throttle controller board circuit schematic 2

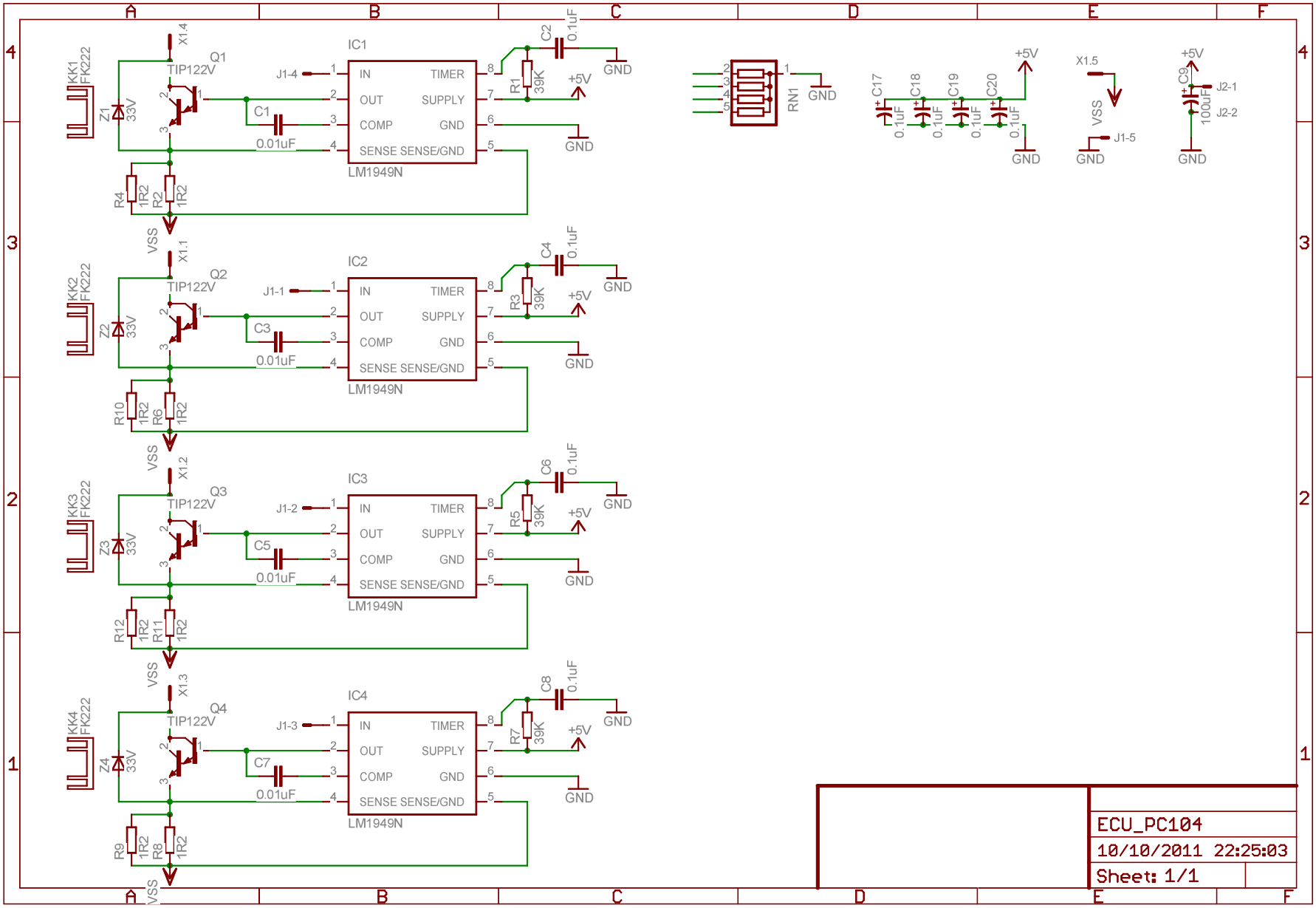

Figure A.15: ECU fuel injector driver board circuit schematic

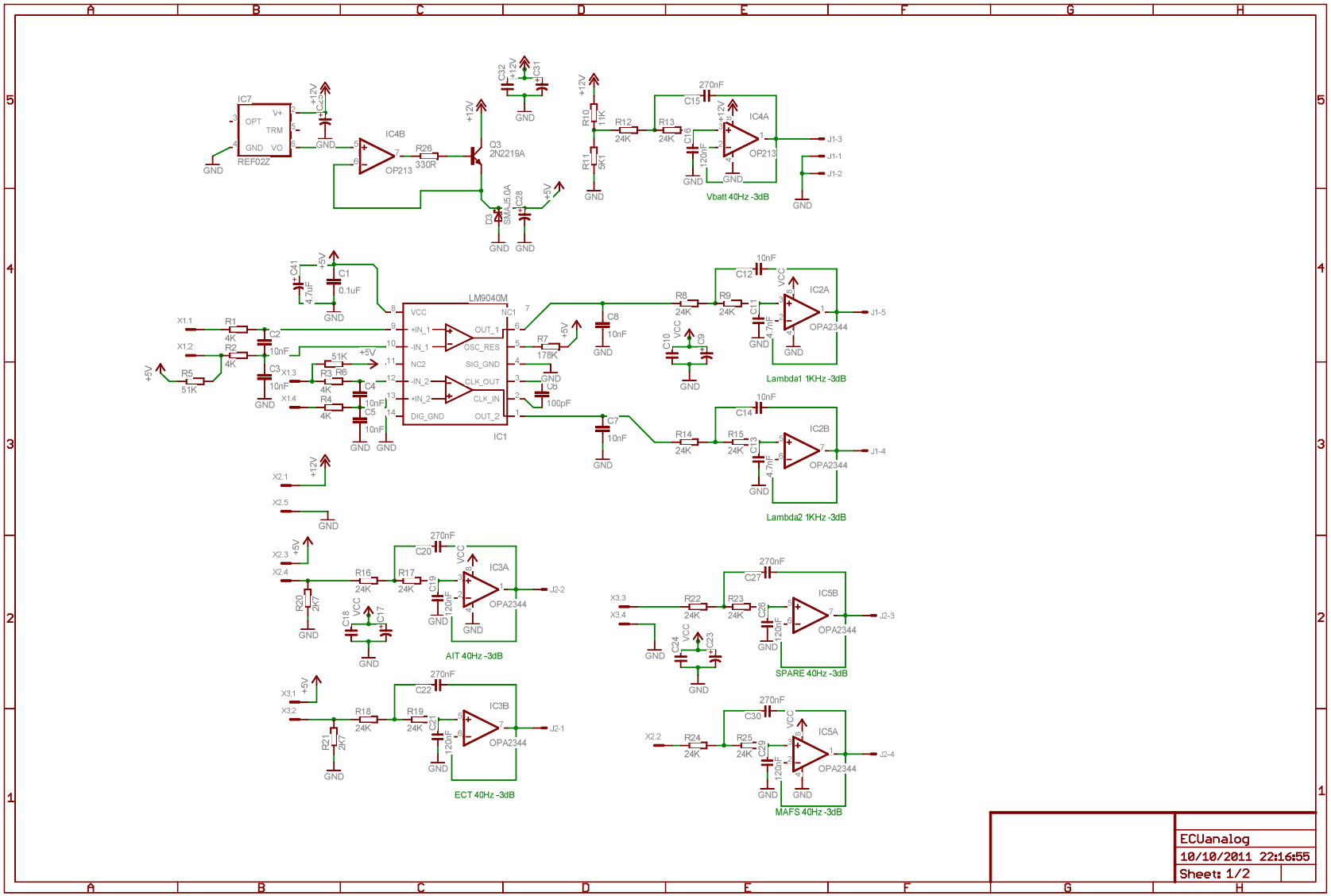

Figure A.16: ECU analogue signal board circuit schematic 1

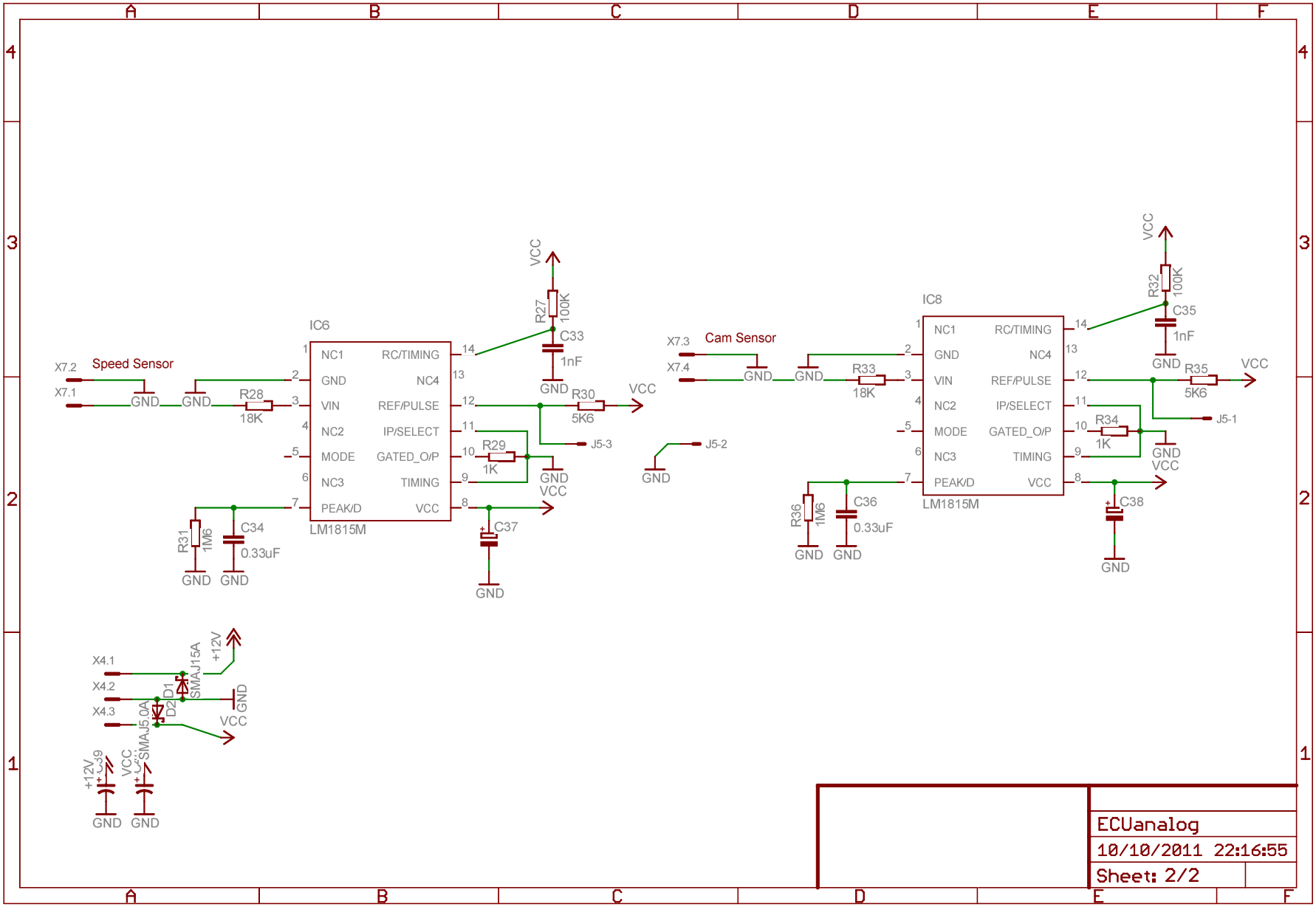

Figure A.17: ECU analogue signal board circuit schematic 2

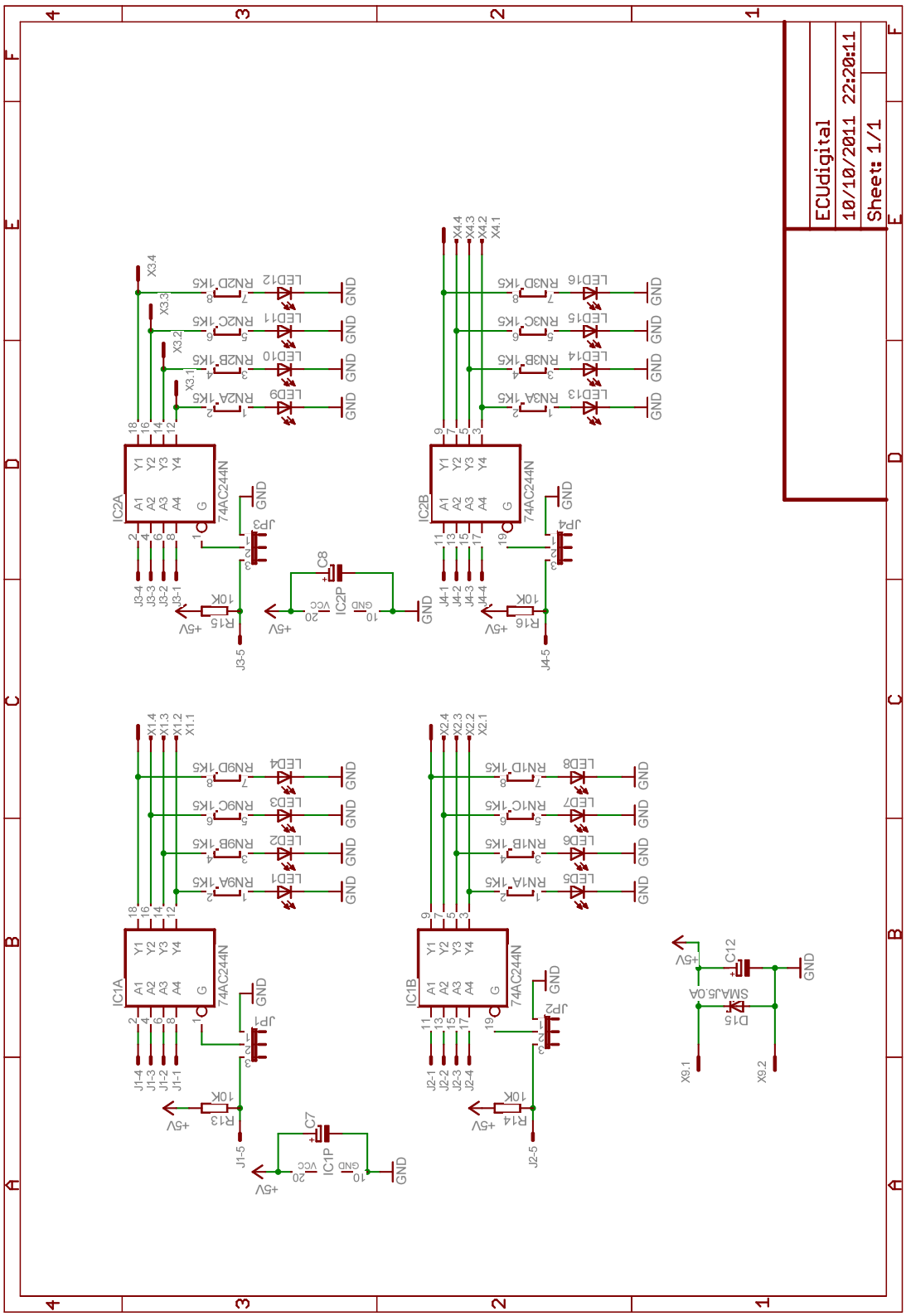

Figure A.18: ECU digital signal buffer board circuit schematic

Appendix B. Embedded Control

B.1 TPU Mask A

The following is a code except which shows the automotive TPU Mask A converted from s-records to a 2kB C char array with the empty function entry points padded. The embedded comments show where each region of the codespace resides.

```
const unsigned char tpu_mask[] = {
/* Microcode block, 4 x 32-bit instructions per row */
  0x3F,0xFF,0xFF,0xFE,0x7F,0xFF,0xFE,0xFE,0x23,0x1F,0xFF,0xFF,0xE3,0x1E,0x21,0xFF,
  0xE3,0x1E,0x41,0xFF,0xE3,0x1E,0x61,0xFF,0xE3,0x1E,0x81,0xFF,0xE3,0x1E,0xA1,0xFF,
  0xE3,0x1E,0xC1,0xFF,0xE3,0x1E,0xE1,0xFF,0x60,0xF9,0xFE,0xD7,0xBF,0xFF,0xFF,0xC8,
  0x3E,0x7F,0xF8,0x0E,0xD2,0x12,0xFF,0xFF,0x3C,0x7F,0xF8,0x0B,0xAE,0x12,0xFE,0xFF,
  0x3C,0x7F,0xF8,0x0B,0xBE,0x14,0xF2,0xCF,0x10,0x1F,0xF0,0x13,0x42,0x5C,0xF5,0xC7,
  0x16,0x7E,0x40,0x0F,0x30,0x7C,0xF8,0x0F,0xA6,0x19,0xFE,0xFF,0x1F,0xFF,0xF0,0x03,
  0xA2,0x28,0xFE,0xFF,0x03,0x5F,0xF0,0x07,0x90,0x29,0xFF,0xFF,0x20,0xDF,0xD0,0x07,
  0x7F,0xF9,0xFF,0xFF,0x22,0xDC,0x0F,0xFF,0x84,0x00,0xFE,0xFF,0x3E,0xDF,0xF0,0x07,
  0x3E,0xFF,0xF0,0x0F,0x9A,0x00,0xFF,0xFF,0x7F,0xFF,0xFF,0xD3,0x3F,0xFF,0xF8,0x17,
  0xB0,0x26,0xFF,0xFF,0x7F,0xFF,0xFF,0xD6,0xC4,0x02,0xF0,0x00,0x3F,0xFF,0xF8,0x16,
  0x90,0x2F,0xFE,0xC7,0x1E,0x7F,0xF0,0x13,0x60,0xDD,0xDF,0xC7,0xD0,0x24,0x70,0x13,
  0xA0,0x2F,0xFE,0xFF,0x3C,0x7F,0xF8,0x0B,0x42,0x58,0xF5,0xC6,0x7F,0xFD,0xFF,0xFE,
  0xBF,0xFF,0xFF,0xD8,0x3C,0x7F,0xF8,0x17,0x7F,0xF9,0xFE,0xCA,0x7F,0xFF,0xFF,0xFA,
  0x38,0x7F,0xFD,0xDB,0x3A,0x7F,0xFD,0xDF,0xAC,0x33,0xFF,0xFF,0xBF,0xFF,0xFF,0xC0,
  0xCF,0xFF,0x50,0x07,0x1F,0xFF,0xF8,0x0B,0x1F,0xFF,0xFA,0x03,0x56,0x5C,0x3F,0xFF,
  0x3C,0xFF,0xF0,0x12,0x16,0xFF,0xFA,0x03,0x02,0x3F,0xF0,0x03,0x9A,0x44,0xFE,0xD3,
  0x1F,0xFF,0xF0,0x0F,0x1F,0xFF,0xF8,0x0F,0x1F,0xFF,0xFA,0x03,0x56,0x5D,0xFF,0xFF,
  0x3F,0xFF,0xF8,0x13,0x22,0x7F,0xFF,0xFF,0x1E,0x7F,0xF2,0x03,0xFF,0x5E,0x0F,0xFF,
  0x36,0x6A,0x3F,0xFD,0xDF,0xFF,0xE8,0x07,0x16,0x1D,0xE0,0x13,0x80,0x4D,0xF6,0xFF,
  0xC0,0x30,0xF0,0x03,0x30,0xFE,0xF0,0x13,0x50,0x5D,0xFF,0xD2,0xBF,0xFF,0xFF,0xF9,
  0x1F,0xFF,0xF0,0x07,0x3F,0xFF,0xF0,0x03,0x7F,0xF9,0xFE,0xD2,0x98,0x8D,0xFF,0xFC,
  0x78,0x5F,0xFF,0xDF,0x1F,0xFF,0xF8,0x0B,0x16,0x1C,0x00,0x13,0x94,0x52,0xFF,0xFF,
  0x84,0x5C,0xFE,0xC7,0x03,0x5F,0xD0,0x0B,0xC2,0x5E,0xF8,0x0F,0x70,0x3F,0xDF,0xCF,
  0x03,0x5F,0xE8,0x0B,0x16,0x3F,0xE0,0x0F,0x10,0x1F,0xF0,0x17,0x22,0xFE,0x7F,0xFF,
  0x17,0xFC,0x00,0x17,0x84,0x67,0xFF,0xFF,0x02,0x7F,0xE0,0x13,0x00,0x9C,0xC8,0x13,
  0x84,0x6E,0xFE,0xFF,0xD2,0x6E,0xFF,0xFF,0x20,0xDF,0xDF,0xFF,0x1F,0xFF,0xF0,0x13,
  0x20,0x9D,0xEF,0xFF,0x84,0x6D,0xFE,0xFF,0x17,0xFC,0x48,0x13,0x84,0x6E,0xFE,0xFF,
  0xC0,0x55,0xF0,0x0F,0x00,0xDF,0xE8,0x13,0x24,0x9F,0xF0,0x13,0x37,0x5F,0xFF,0xFF,
  0x34,0x7F,0xF8,0x0B,0x22,0x3F,0xFF,0xFF,0xE6,0x7E,0x2B,0xFF,0x16,0x7F,0xD2,0x03,
  0x1E,0x1F,0xFA,0x03,0x32,0x1E,0x3F,0xFD,0x36,0xFE,0x4F,0xFF,0x1C,0x1E,0x30,0x17,
  0x20,0xC6,0x3F,0xFF,0x20,0xC6,0x3F,0xFF,0x9E,0x7D,0xFE,0xFF,0x3F,0x5F,0xFF,0xFF,
  0x3F,0x5F,0xDF,0xFF,0x05,0x5D,0xF8,0x03,0x94,0x81,0xF5,0xFF,0xD2,0x84,0xFF,0xFF,
  0x36,0x75,0xFF,0xFF,0x36,0x65,0xFF,0xFF,0x90,0x84,0xFE,0xFF,0x36,0x7F,0xDF,0xFF,
  0x8C,0x86,0xFE,0xFF,0x7F,0xFF,0xF3,0xFF,0x35,0xFD,0xFF,0xFF,0x94,0x89,0xFE,0xFF,
  0x1F,0xFF,0xF8,0x07,0x26,0x9C,0x08,0x03,0x52,0x59,0xFE,0xFF,0xE7,0x3E,0x21,0xFF,
  0x94,0x7D,0xFE,0xFF,0x7F,0xFF,0xFE,0xFE,0x7F,0xFF,0xFF,0xFB,0x1F,0xFF,0xF8,0x17,
  0x1F,0xFF,0xF0,0x13,0x7A,0x19,0xFE,0xDF,0x21,0xFC,0x48,0x07,0x8C,0xA5,0x50,0x84,
  0xB0,0x96,0xFF,0xFF,0x36,0xFF,0xFF,0xFF,0x40,0x5D,0xCF,0xCE,0xB0,0x99,0x50,0xC7,
  0x98,0x8E,0xFE,0xFF,0x1A,0x1F,0xD8,0x17,0x00,0x3F,0xF0,0x13,0x3F,0xFF,0xF8,0x07,
  0x34,0x7F,0xF0,0x0F,0x98,0xA5,0xFF,0xFF,0x7F,0xFF,0xFF,0xFB,0x21,0xFC,0xCF,0xFF,
  0x84,0x95,0xFF,0xDF,0x33,0xFC,0xCF,0xFF,0x90,0xA5,0xFE,0xC7,0x8E,0xA6,0xFF,0xFF,
  0x3A,0x5F,0xF0,0x0F,0x20,0x5F,0xF0,0x0F,0x5F,0xFD,0xFF,0xD6,0x98,0xAF,0xF4,0xCF,
  0xD0,0xAB,0xFF,0xFF,0x7F,0xFF,0xF3,0xFF,0x98,0xAD,0xFE,0xCF,0x29,0xFF,0xFF,0xFF,
  0x5C,0x5D,0xCF,0xD6,0xB0,0xAF,0xFF,0xFF,0x3C,0x7F,0xF8,0x0B,0xD4,0xDF,0xFF,0xFF,
```

```
0x31,0xE5,0xCF,0xFF,0x84,0xB4,0x4F,0xD7,0x7C,0x7E,0x3F,0xC7,0x56,0x5D,0xC1,0xDE,
0x5C,0x5C,0x3F,0xCE,0xB0,0xB9,0x54,0xC7,0x69,0xFF,0xFF,0xFB,0x3C,0x7F,0xF8,0x0B,
0xC0,0xB0,0xF0,0x07,0x9A,0xA5,0xF1,0xFF,0x00,0x7F,0xF0,0x07,0x7A,0x1F,0xDF,0xFB,
0x21,0xFC,0xC0,0x0F,0x94,0x9F,0xFE,0xFF,0xD4,0xDF,0xFF,0xFF,0xD0,0xCC,0xFF,0xFF,
0x1A,0x1F,0xF0,0x13,0x01,0xFC,0x68,0x17,0xAC,0xC4,0xFE,0xFF,0x3F,0xFF,0xF8,0x06,
0x8C,0x00,0xFF,0xFF,0xBE,0xA5,0x50,0xC7,0x3F,0xFF,0xF8,0x07,0xAC,0x00,0xFE,0xFF,
0xB0,0xD6,0xFE,0xFF,0x3F,0xFF,0xF0,0x07,0xA4,0xCC,0xFE,0xFF,0x9C,0xB5,0xFF,0xFF,
0x18,0x1F,0xF8,0x0B,0x32,0x3E,0xCF,0xFF,0x35,0xFC,0x0F,0xFF,0x90,0xD1,0x4F,0xD7,
0x58,0x5D,0xF5,0xCE,0x36,0x3E,0x3F,0xFF,0x33,0xFC,0x8F,0xFF,0x8C,0xD5,0xFE,0xC7,
0x54,0x5D,0xF5,0xCE,0x54,0x5D,0xC1,0xDE,0x1A,0x1F,0xF0,0x13,0x00,0x7F,0xF0,0x17,
0x21,0xFC,0x40,0x07,0x8C,0xA5,0x50,0xC7,0x21,0xFC,0xC0,0x0F,0x94,0xBE,0xFF,0xD7,
0x8C,0x95,0xFF,0xDF,0x5A,0x5D,0xFF,0xD6,0xBC,0x00,0xFE,0xF9,0x02,0x3F,0xF0,0x03,
0x22,0x7F,0xFF,0xFF,0x1E,0x1F,0xFA,0x03,0xFF,0x5E,0x0F,0xFF,0x32,0x0A,0xFF,0xFD,
0xDB,0xFF,0xFF,0xFF,0x32,0xE7,0xFF,0xFF,0x3E,0x67,0xD8,0x0B,0x3E,0x1F,0xCF,0xFF,
0xBF,0x14,0xFE,0xF8,0x7E,0xF9,0xEE,0xFF,0xA4,0xF4,0xFE,0xFF,0xE1,0xFC,0x03,0xFF,
0x94,0xEE,0xFF,0xFF,0x20,0xDF,0xD0,0x0B,0xA0,0xF4,0xFF,0xFF,0x9A,0xF2,0xFE,0xFF,
0x20,0x0D,0xFF,0xFF,0x32,0x0D,0xFF,0xFF,0x95,0x0B,0xFE,0xFF,0x5C,0x5D,0xCF,0xFE,
0x00,0x1F,0xF8,0x03,0x3E,0xDF,0xF0,0x0B,0x3C,0xFF,0xF0,0x03,0x7C,0x22,0xCF,0xFF,
0x98,0xFA,0xFE,0xFF,0x32,0x09,0xFF,0xFF,0xF2,0x08,0x01,0xD7,0x32,0x7F,0xF8,0x13,
0x34,0x7F,0xF8,0x17,0x1F,0xFF,0xF0,0x0F,0x1A,0xDC,0x05,0xFB,0x20,0xDF,0xD5,0xFB,
0x87,0x11,0xFE,0xFF,0x33,0xFD,0xFF,0xFF,0x95,0x0B,0xFE,0xFF,0x1E,0x1F,0xF0,0x0F,
0xFF,0x5E,0x0F,0xFF,0x22,0x3F,0xFF,0xFF,0x32,0x0A,0xFF,0xFD,0x32,0x25,0xFF,0xFF,
0x8D,0x0B,0xFF,0xFF,0x91,0x0B,0xFF,0xFF,0x5C,0x5C,0xBF,0xCE,0x1E,0x1F,0xC5,0xFB,
0x31,0xFD,0xFF,0xFF,0x8D,0x13,0xFE,0xFF,0xA0,0xF3,0xFE,0xFF,0xD1,0x13,0xFF,0xFF,
0x3C,0x7F,0xF8,0x03,0x3E,0x1F,0xFF,0xFF,0x3E,0x7F,0xF8,0x0B,0x5C,0x5D,0xCF,0xFF,
0xFE,0x3F,0xFF,0xFF,0x72,0x17,0xDF,0xD7,0x20,0xFE,0x75,0xFB,0x73,0xBE,0xBF,0xC2,
0xAD,0x1C,0xFF,0xC7,0xA5,0x1E,0xFE,0xFF,0xA1,0x33,0xFF,0xFF,0x56,0x5D,0xCF,0xFE,
0xA5,0x33,0xFE,0xFF,0xA1,0x22,0xFE,0xFF,0x1F,0xFF,0xF5,0xFB,0x20,0xDF,0xD5,0xFB,
0xAF,0x10,0xFF,0xFF,0x7F,0xFB,0xFF,0xFF,0x1F,0xFF,0xF0,0x07,0x3A,0xDC,0x0F,0xFF,
0x95,0x27,0xFF,0xFF,0xC3,0x11,0xF5,0xFB,0x36,0x5F,0xFF,0xFF,0x02,0x3F,0xE0,0x0B,
0xB3,0x2C,0xFE,0xFF,0x23,0xFD,0xFF,0xFF,0x22,0x9F,0xD0,0x0B,0x23,0xFC,0x8F,0xFF,
0x85,0x30,0xFF,0xFF,0x3E,0x9F,0xF0,0x0B,0x3E,0xDF,0xE5,0xFB,0x3F,0xBF,0xEF,0xFF,
0xAD,0x32,0xFE,0xFF,0x7F,0xFF,0xFF,0xDF,0x56,0x5D,0xCF,0xFA,0x3D,0xFC,0xCF,0xFF,
0xC2,0xF4,0xF0,0x0B,0x8C,0xEB,0xFF,0xFF,0x03,0x5F,0xF8,0x0F,0x36,0x7F,0xD8,0x0F,
0x1F,0xFF,0xF8,0x0F,0x16,0x3F,0xF8,0x0B,0x37,0xFC,0x8F,0xFF,0x85,0x3E,0xFF,0xFF,
0x3C,0x7F,0xF8,0x17,0x7F,0xFB,0xFF,0xFE,0xAF,0x41,0xFE,0xFF,0x3C,0x7F,0xF8,0x13,
0xD4,0x02,0xFF,0xFC,0x20,0x7F,0xFF,0xFF,0x1F,0xFF,0xF2,0x03,0x22,0x9F,0xD2,0x03,
0xB3,0x49,0xFE,0xFF,0x7F,0xFF,0xF1,0xFB,0x1F,0xFF,0xF0,0x03,0x3E,0x7F,0xF8,0x0F,
0xBF,0xFF,0xFF,0xF8,0x7F,0xF9,0xFE,0xFE,0xE1,0xE4,0x01,0xC7,0x8F,0x4E,0xFE,0xF8,
0x78,0x59,0xFE,0xFF,0x7A,0x59,0xFE,0xFF,0x3C,0x7F,0xF8,0x07,0xD5,0x51,0xFF,0xFF,
0x52,0x5C,0xB5,0xFA,0x16,0x3F,0xF0,0x0B,0x10,0x1D,0xF8,0x0F,0x87,0x58,0xFF,0xFF,
0x36,0xFE,0xB0,0x13,0x37,0xFC,0x4F,0xFF,0x85,0x59,0xFF,0xFF,0xD9,0xFF,0x1F,0xFF,
0x54,0x5C,0xF1,0x8A,0x54,0x5C,0xF1,0x4A,0xA5,0x4E,0xFF,0xFF,0xD1,0x4F,0xFF,0xFF,
0xA5,0x5F,0xFE,0xFF,0x9D,0x5F,0xFE,0xFF,0x3C,0x7F,0xF8,0x07,0xD5,0x51,0xFF,0xFF,
0x54,0x5C,0xF3,0xFA,0x50,0x5D,0xF3,0xFE,0xAD,0x69,0xFF,0xFF,0xBF,0xFF,0x0F,0xFC,
0x58,0x5F,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xB1,0x69,0xFE,0xC3,0xCF,0xFF,0x30,0x0B,
0x5C,0x5E,0x31,0xFF,0xB7,0x6D,0xFF,0xFF,0x1F,0xFF,0xF0,0x07,0x70,0xE9,0xFE,0xFB,
0x3F,0xFF,0xF0,0x06,0x70,0xE9,0xFE,0xFB,0x30,0xFF,0xC0,0x06,0xBD,0x6B,0x40,0xBF,
0xBD,0x6D,0x40,0x7F,0xE1,0xE4,0x01,0xC7,0x8F,0x75,0xFE,0xD0,0x78,0x59,0xFF,0xFF,
0x3A,0x5F,0xFF,0xFF,0xB3,0x87,0xFE,0xFF,0x1F,0xFF,0xF0,0x0F,0xCF,0xFF,0x30,0x13,
0x3C,0x7F,0xF8,0x03,0x30,0x7F,0xFF,0xFF,0x02,0x7F,0xF2,0x03,0x10,0x1F,0xFA,0x03,
0x3F,0xFF,0xF8,0x17,0x16,0x3F,0xF0,0x0B,0x12,0x1E,0x30,0x0F,0x52,0x5C,0x34,0xCB,
/* End of normal microcode region */
0x34,0x7E,0x38,0x06,0x9C,0x00,0xFE,0xFF,0xCF,0xFF,0xE8,0x17,0xB6,0x00,0xFF,0xFF,
0x36,0xFE,0x30,0x07,0x50,0x5D,0xF3,0xD6,0x7F,0xFF,0xFE,0xDF,0x3C,0x7F,0xF8,0x03,
0x10,0x7E,0xF0,0x0B,0x5C,0x5C,0x35,0xC3,0xAD,0x8E,0xFE,0xFF,0x1F,0xFF,0xF0,0x13,
0x23,0x5F,0xFF,0xFF,0xD4,0x02,0xFF,0xFC,0x3F,0xFF,0xF8,0x06,0x7F,0xFF,0xFE,0xDE,
0x99,0x96,0xFE,0xFF,0x1F,0xFF,0xF0,0x13,0x00,0x7F,0xF0,0x17,0xCF,0xFF,0xEA,0x03,
0x16,0x7E,0x30,0x0F,0x36,0x7E,0x38,0x07,0x56,0x5D,0xF3,0xD6,0xBF,0xFF,0x07,0xFC,
0x67,0x39,0xFE,0xFF,0x3E,0xFF,0xC0,0x0F,0xB5,0x9C,0xFF,0xFF,0x3E,0xFF,0xF0,0x0F,
0x38,0x7F,0xF8,0x0A,0x67,0x39,0xFF,0xFF,0x10,0x3D,0xF8,0x13,0xB7,0xA2,0xFE,0xFF,
```

```
  0x10,0xFF,0xCA,0x03,0x35,0xFD,0xCF,0xFF,0x8C,0x01,0xFE,0xFF,0x30,0x3C,0xF0,0x0F,
  0x1C,0xFF,0xF8,0x17,0xB3,0xA8,0xFE,0xFF,0x36,0x7F,0xD2,0x03,0x35,0xFD,0xCF,0xFF,
  0x8F,0xAB,0xFE,0xFF,0x1F,0xFF,0xF2,0x03,0x30,0xFF,0xD2,0x02,0x30,0xFF,0xE2,0x02,
/* Actual Microcode end, using 5 1/2 unused function entry points for code space */
/* Padding bytes for the last half of unused function 5 entry point */
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
/* End padding bytes */
/* Entry point 6 - QDEC */
  0xF8,0x01,0xF8,0x01,0xE9,0x9C,0xE9,0x97,0x61,0x9D,0x61,0x9D,0x61,0x9D,0x61,0x9D,
  0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,
/* Entry point 7 - SPWM */
  0xF8,0x00,0xF8,0x00,0x01,0x71,0x61,0x81,0x39,0x90,0x39,0x96,0x71,0x87,0x91,0x78,
  0x91,0x8F,0x61,0x82,0x91,0x8F,0x61,0x82,0x91,0x92,0x61,0x82,0x71,0x86,0x91,0x78,
/* Entry point 8 - DIO */
  0x31,0x70,0x31,0x70,0x31,0x6F,0x11,0x62,0x31,0x6B,0x41,0x68,0x31,0x6D,0x41,0x68,
  0x10,0x01,0x10,0x01,0x10,0x01,0x10,0x01,0x31,0x6B,0x41,0x68,0x31,0x6D,0x41,0x68,
/* Entry point 9 - PWM */
  0x29,0x5C,0x29,0x5A,0x01,0x4A,0xF8,0x00,0x91,0x61,0xF9,0x5E,0x31,0x4E,0x31,0x4E,
  0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,
/* Entry point A - ITC */
  0x11,0x47,0x11,0x47,0xF8,0x00,0xF8,0x00,0x31,0x36,0x31,0x36,0x31,0x36,0x31,0x36,
  0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,0x31,0x36,0x31,0x36,0x31,0x36,0x31,0x36,
/* Entry point B - PMA/PMM */
  0x10,0xE6,0x10,0xE6,0xF8,0x00,0xF8,0x00,0x50,0xEA,0x19,0x18,0x50,0xEA,0x19,0x18,
  0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,
/* Entry point C - PSP */
  0x80,0xC0,0xA0,0xC7,0xE8,0x8F,0x10,0xDE,0x70,0xA9,0x70,0x97,0x70,0xA7,0x70,0xB5,
  0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,0xF8,0x01,
/* Entry point D - SM */
  0x10,0x8D,0x10,0x8D,0x10,0x4F,0x60,0x53,0x90,0x5C,0x90,0x59,0x90,0x5C,0x90,0x59,
  0x10,0x8D,0x10,0x8D,0x10,0x8D,0x10,0x8D,0x10,0x8D,0x10,0x8D,0x10,0x8D,0x10,0x8D,
/* Entry point E - OC */
  0x00,0x34,0x00,0x34,0x10,0x00,0x10,0x30,0x10,0x31,0x38,0x4A,0x10,0x31,0x38,0x4A,
  0x10,0x01,0x48,0x3D,0x10,0x01,0x48,0x3D,0x10,0x01,0x48,0x3D,0x10,0x01,0x48,0x3D,
/* Entry point F  - PPWA */
  0x10,0x00,0x10,0x00,0x10,0x0A,0x10,0x00,0x40,0x0F,0x80,0x2C,0x40,0x0D,0x80,0x2C,
  0x10,0x01,0x10,0x01,0x10,0x01,0x10,0x01,0x40,0x0F,0x80,0x2C,0x40,0x0D,0x80,0x2C
};
```

B.2 Linux as an Embedded Operating System

The eCos Operating System For control purposes eCos would be a natural choice as it is a low-level real-time OS with a variety of scheduler policies to choose from. However the board support that was developed by Arcom was only sufficient enough to allow the Redboot boot-loader (a minimal schedalarless eCos application in itself) to function correctly. Also under the eCos single process model, providing a graphical framebuffer GUI is possible, but would not be trivial in terms of development effort. There is some open source support under eCos for Microwindows (renamed to Nano-X Window system due to Microsoft tradmark infringement), Bejing based FMSoft's MiniGUI orginally developed for a Linux based CNC machine tool user interface, and there has previously been commercial support for a framebuffer graphical environment called posC, although this seem to have ceased to exist since the time that this work was carried

out. The amount of work to make any of these solutions work with the PXA processor hardware and eCos was not known and would have required further investigation.

Conventional Linux Linux is a general purpose operating system which provides good overall work throughput for use on a desktop machine and also for dedicated servers. In its standard form it does not allow any one process to monopolise the processor's time which maintains a responsive user interface. It has a multi-threaded non-preemptive kernel which can be the cause of occasional but significant scheduler latencies for user mode applications. These factors make Linux much less of an appropriate choice for real-time control applications. Despite this Linux has gained increasing popularity in recent years for embedded devices and consumer electronics. The open source nature and ports to a very wide range of architectures has assisted with this as well as the availability of a wide range of compatible open source software. The very fact that software can be obtained as source code allows it to be cross-compiled to different processor architectures without necessarily requiring the consent or support of its original author. In fact, several distributions including Debian are supplied pre-compiled for non-x86 plactforms suchas ARM and PowerPC. This means if the dependencies are present, a package can be simply downloaded and copied or installed directly onto the embedded target. For AEL this ment bringing some of the base level packages into line with a current version of Debian by manually installing their contents into situe so that other packages could be used without having to cross-compile all of the source and dependencies. This process was assisted by the use of a cross-compling sandbox environment called Scratchbox. It is possible to install all of the base dependecies into a sandbox and cross-compile against them. Then by using Scratchbox remote shell (which in turn uses *chroot*) is is possible to run the newly compiled program from a network share of the sandbox on the PC, on the target hardware without having to install it or any dependencies onto the hardware. This speeds up the development process and allows different packages to be tested without having to install them. It also removes the need to use the outdated AEL provided cross-compiler an use source which knows how to be configured for cross-compilation. Much of the open source software has never been tested for cross-compilation or written with that in mind, but Scratchbox allow the configuration scripts to believe they are being natively compiled for the target hardware which removes the source of many problems.

RTLinux RTLinux offers a compromise to conventional Linux by allowing existing native Linux software to be run whilst offering a real-time framework at the same time. It does this using a patented dual OS model (Yodaiken, 1999) whereby a general purpose OS is added as a task to a real-time one as depicted by Figure B.1. There is a POSIX compliant real-time OS under which Linux is run as a self-contained application. Any

real-time control code must be run as a process on the underlying POSIX OS and communication which any Linux applications is performed through a dedicated IPC mechanism.

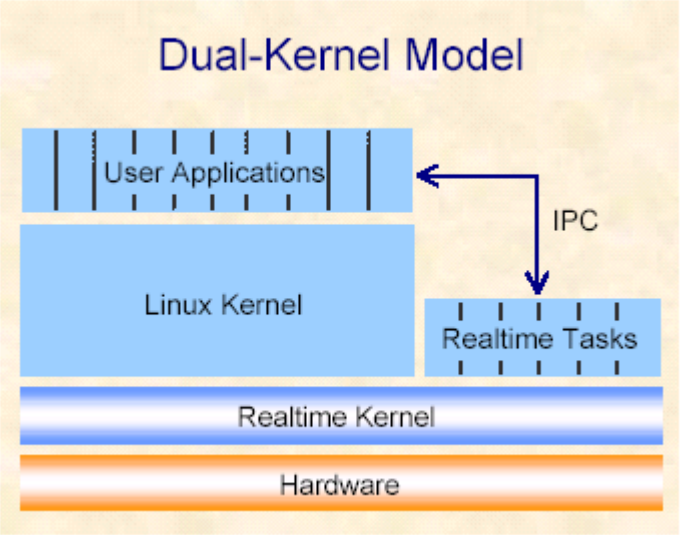

Figure B.1: RTLinux dual kernel model

A quote was obtained for RTLinux, but the licensing requirements were found to be very restrictive allowing only a single architecture to be chosen per licence. Also it was not clear what level of Viper specific hardware support would be provided. Any additional hardware device drivers such as CAN or acquisition data cards that would take advantage of the real-time support would require reimplementing for the POSIX kernel as native Linux drivers are not supported.

Linux Kernel Real-Time Patches The option exists to modify the standard or *vanilla* Linux kernel to bring down the worst case scheduler latencies. Williams (2002) performed a test of the unmodifed vanilla kernel running on a 700 MHz Pentium class x86 machine to estabish the magnitude frequency of occurrence of scheduler latencies whilst running software to stress the kernel over a 41 minute period. Figure B.2 show the results of the test from which it can be seen that (noting the logarithmic scales) the vast majority of samples fall within 100 µs, but a significant minority take a much greater time of up to 110 ms. This is caused by the kernel entering long or non-preemptable code sections or loops. There existed a set of patches to reduce this preemption and low-latency patches which were two slighly different approaches to solve the same problem. The preemption patch modifies the kernel to allow preemption to take place more often and reduce the amount of time it take from a call the the scedular to the point where it runs. The low latency approach is to intersect long or slow code sections with explict preemption points where it is safe to perform a context switch.

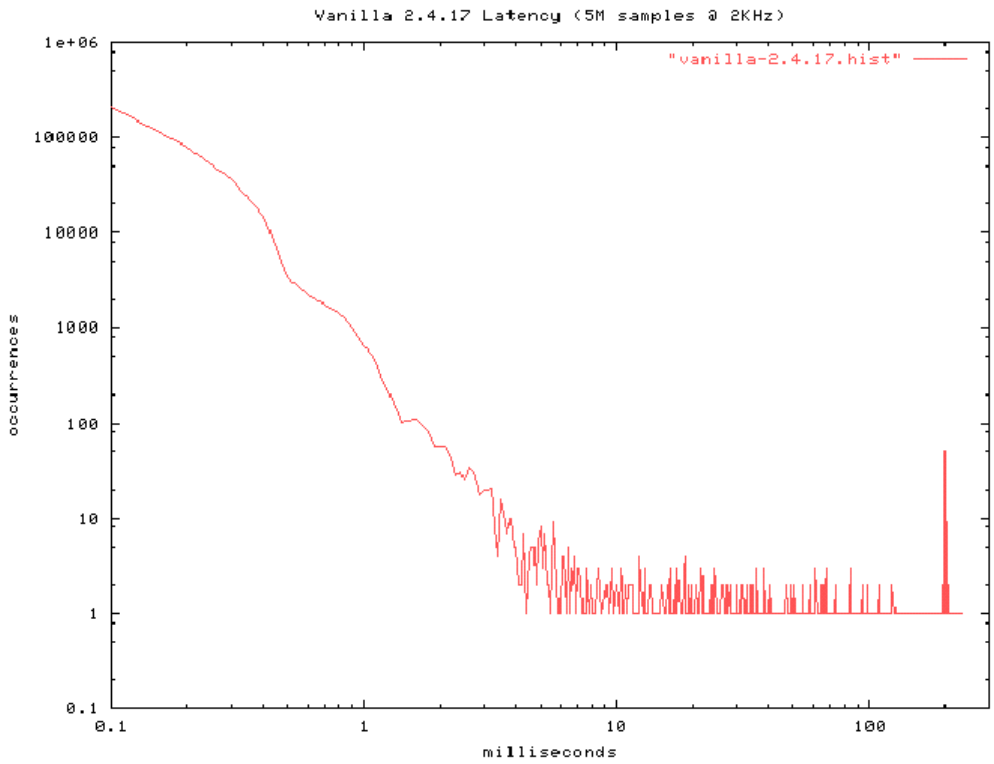

Figure B.2: Linux kernel latency (Williams, 2002)

RTAI Real-Time Application Interface (RTAI) is a project that was created to add real-time capability to Linux and circumvent the RT-Linux patent by adding that support directly into the operating system, rather than layering it on top of a separate deterministic operating system. RTAI uses a modification to the kernel which they call Adaptive Domain Environment for Operating Systems (Adeos) which is a nanokernel hardware abstraction layer (HAL) that resides between the kernel device drivers and the hardware. What is does in essence is to form a thin virtualisation layer between the kernel and the hardware so that it can capture interupts and hardware access then decide how, when and where to displach them, thus preventing the Linux kernel from interfering with real-time processes. RTAI then form the framework for these real-time processes.

To investigate the suitablity of RTAI for this work, a version of the host development distrubution's kernel supplied with Mandriva Linux was patched to include RTAI support in order to evaluate RTAI using a desktop PC. This required hand merging the changes made to the standard version 2.6.13 kernel source by the Mandriva team with those made by RTAI as both patch sets can only be applied to the standard kernel source. The resulting kernel source was successfully compiled and run, and an example real-time trivial application run, but some kernel instabiliy was incountered and time constraints prevented any further investigation of this framework. It was anticipated that a significant amount of extra work would be required to get RTAI to work with the AEL version of the Arm Linux 2.4.26 version of the kernel, particularly if an RTAI patch doesn't exist for this exact kernel version. AEL uses an unmodified standard or 'Vanilla'

kernel, but despite this it provides additional device drivers and configuration options that might conflict with the RTAI modifications. At the time of investigation, the 2.6 branch of the kernel was becoming the standard revision in use and support of version 2.4 (still used by the AEL version in use at that time) was starting to decline as it became a legacy kernel. Also a significant deciding factor was that the ARM platform had much less support than x86 from RTAI developers as the low level interrupt catching mechanism differs from that of the x86 processor where a larger user base and interest level existed. On this basis it was decided to wait until the exact nature of the real-time requirements had been established before committing more time to development of RTAI for AEL Linux which might have been problematic. Debugging of kernel code is often far more difficult than conventional code without expensive hardware tools as the conventional software mechanisms (such as the GDB debugger) are not able to work. There are a number of techniques which can be employed to assist with kernel development but many of them require further modifications to the kernel source to allow for instrumentation. This could further exacerbate the problem by making yet more changes to the kernel. Since this work was carried out, there have emerged two deguggers, KGDB and KDB, the first being a remote serial debugger allowing a second machine to be used to debug the first, and the second a local debugger for the kernel. Both require patches to the kernel and have only been introduced from around release version 2.6.26. Also, the Adeos abstraction layer does itself permit patchless debugging of the kernel, but how effective this would be when it is required to assist with setting up of RTAI is not immediately clear.

Xenomai During the course of the development work, another project for real-time Linux emerged called Xenomai. This project was motivated and bourne out of certain frustrations surrounding RTAI, and general lack of support for non-x86 architectures having previously merged with the RTAI project as RTAI/fusion, it forked again and has different emphasis and goals, particularly being more readly usable rather than providing the absolute mininum latencies as a point of techical achivement. Xenomai has a lot in common with RTAI, but has diverged from RTAI and was not intended to remain compatible with it. Xenomai has been an interesting development, but its re-emergence from RTAI occured at a stage which was too late to adopt it for this project.

B.2.1 Booting Embedded Linux from Flash

A problem encountered using the Arcom Viper was that the flash disk space was found to be restrictive for development purposes and also file access to the JFFS2 filesystem images is only possible using the running operating system on the board. This make upgrading libraries and important system files on the running system precarious as it

could render the system unstable or inoperable. An inoperable system that requires the entire filesystem image is replaced using the Redboot bootloader. The process is quite time consuming and also mandates (to be worthwhile) that regular backups of the system partition images are taken which is in itself a time consuming manual process. Image files can be transfered over an Ethernet network using trival file transfer protocol (TFTP) which requires configuring a TFTP server on a host PC, or alternatively a serial line connection to a terminal program can be used which is even more time consuming and temperamental due to the large file sizes. Creation and modification of JFFS2 images on the host PC is possible but requires memory technology device (MTD) support to be compiled into the kernel which is often not for PC based distributions as raw flash access is seldom required due to the fact that the vast majority of flash devices (including CompactFlash, USB *Pen* drives) work as block devices that function like disks at the hardware interface level. The host Mandriva kernel was recompiled from source to allow JFFS2 image and *loopback* mounting of the Viper image, but this approach is neither convenient or maintainable as updated kernels are releases at regular intervals for security fixes and other improvements.

A solution that was investigated was the use of CompactFlash to add significant additional removable file storage space. Device drivers were already provided and installed, so mounting a filesystem on CompactFlash is relatively trivial. The Linux *chroot* command can be used to swap the location of the root filesystem to another location for all processes executed after the command is run. This offers an intermediate solution if all of the system files are replicated on the external CompactFlash device. It was felt that a better solution would be to boot directly from the compact flash itself, allowing it to be removed and the root filesystem to be modified from a host PC, and even duplicates kept on separate flash devices to support different configurations under test. It also would make the backup process as simple as backing up any other removable storage device. With embedded Linux, the kernel image is normally stored separately to it's root filesystem. In the case of the Viper, the kernel image is stored in a raw flash partition and is loaded into memory by Redboot and executed in place. The bootloader is able to pass a command line of arguments to the kernel before it starts and the kernel uses this to determine where to look for its root filesystem. The kernel has the JFFS2 filesystem drivers it needs to read the disk image contents from the flash device which is mapped as memory, so no hardware drivers are required. This mechanism is in contrast to the boot procedure used by PC based Linux distributions which make use of a temporary file system in RAM called an initial ramdisk (initrd image). The PC boot process make use of the BIOS, a disk based bootloader, then the initrd image which allows the generic kernel to gain access to device driver modules and other critical files which it may need to continue the boot process. This allow a kernel image to be built which doesn't need to include every possible hardware driver it may need to boot on any PC. For the embedded case the kernel is usually compiled for a specific board or

hardware configuration. Booting directly from compact flash, by loading the kernel image off either the external flash or the on-board flash and allowing it to use the external flash for its root filesystem is technically possible from a hardware standpoint. It does also require software co-operation. The Viper kernel was recompiled to have the necessary drivers built into the kernel to read the compact flash thought its PCMCIA interface and to be able to read an ext2 or FAT filesystem on the card. It was found that despite the kernel having the drivers which were previously compiled as modules, built into its image, there were more file dependencies that could not be resolved to get it to work. The problem was circular in that it required files from the filesystem in order to read the file system. This could have been overcome by creating a filesystem in the on-board flash which contains the necessary files, but it was felt that the time consumed was too great for the benefit that would be obtained.

B.3 Control Software

B.3.1 Diamond Device Driver

The Diamond DMM32AT card was purchased in the belief that it is provided with open-source driver code. It was assumed that the necessary changes could be made to this source code to allow it to be used with the Arcom Viper SBC. Unfortunately it was subsequently found that on closer inspection of the provided code all that is supplied is an open source kernel module and pre-compiled i386-Linux library which an application can be linked against. The driver is essentially what is known as a user mode driver. The kernel module handles interrupts and notifies the library code residing in user space of the confirmed interrupt and supplies it with any data resulting from an ADC conversion. The configuration of the card is performed in the closed source library. This was obviously unsuitable for use with ARM Linux as the library is not binary compatible, so the UK supplier was contacted to find out if the source code could be obtained. In spite of the fact that at least one other customer was reported to be interested in an ARM port, they were not prepared to contact the manufacturer about releasing the library source, or putting a time scale on work to port the library to ARM Linux. Their library has been coded in such a way so that it can be compiled for several operating systems (such as Linux, QNX, Windows, and DOS) whilst maintaining the same API, which may be a significant factor in their unwillingness to release source code due to the loss of intellectual property in their approach to creating a multi-operating driver system API. It was subsequently realised that the supplier (Diamond Point International) is only a distributor for Diamond Systems Corporation and although bears a similar name, does not necessarily represent them, and that also the source code is available directly from Diamond System for an undisclosed fee.

Without the library, there was a limited amount of information available from the accompanying documentation about how to use the card from a low-level of register access. Actual examples given related to either simple one shot software sampling, or using their API and library to achieve higher rate interrupt based sampling. Although all of the registers are documented, it was not possible to ascertain how several advanced features of the card functioned such as performing auto-calibration, loading calibration data, and setting the programmable range of the DAC to something other than the 5 V default.

To preserve the general structure and maintain API level compatibility with the library provided by Diamond (in case an ARM variant might be release in the future), it was decided to disassemble their binary library, and as best as possible, reconstruct it as C source so that it could be re-compiled for any Linux architecture. Support for other operating systems was not required which simplified the task somewhat. From the unstrapped binaries contained within the static library archive it was possible to 'stub-out' the general structure used from the function names and their dependencies upon each other and on function calls to system libraries such as the POSIX *pthread* library. It was then possible to reconstruct the basic functionality required to run the card by rewriting the source using the register documentation and a certain amount of trial and error using the actual card to test the code. It was also possible to achieve some efficiency improvements as the existing library supports much of Diamond's product range and checks which card type is in use every during every function call. There was also evidence of a kind of state machine which appeared to be producing unnecessary overhead.

Integrating a C Callback with a C++ Application One problem facing C++ programmers is finding a method to connect to the outside world, when this is often done at the C level by registering a function pointer to an operating system call. C++ function pointers are usually incompatible with the C function pointer and there can be compiler specific implementation differences. A solution to this problem was required to write a class to handle the Diamond DMM32AT card. It was desired that a single instance of a class could be instantiated and methods of the class called to initialise the card and deliver sample results. The Diamond supplied library for use with the card uses the typical C callback technique were a user C function can be called when sample results are available. No state information is retained by the function and global variables have to be used to gain access to the card status and sampled data once inside the callback function. Retaining all of the state information within a class and having a class method called when data is available is a preferable and more contemporary approach, although opinions as to the *correctness* any approach in software often vary greatly depending upon the background of the individuals concerned. It is normal practise for C callback

interfaces to allow a `void*` data item to be set and then passed to the callback function when it is called. The called function understands the significance of the pointer and can cast it to another data type or simply ignore it as appropriate. This allows the programmer to pass current contextual information into the function or provide a pointer to a data storage location which is only known at run-time. The Diamond library provides this interface (they state in the documentation) to maintain calling compatibility with Microsoft Windows. However the pointer is not retained and passed to the callback when set. Since an API compatible library was written for this project to allow cross-compilation to ARM Linux, this functionality could be easily added. A C++ class member which has been declared `static` can be used as a helper function and set as a C callback function and a reference to the class instance (`this` pointer) can be set as the `void*` data item. The helper function is compatible with the C compiler calling interface, but is also a C++ class member so it can cast the `void*` data item back to a class instance an call the appropriate class method. In this case the method was made a *pure virtual* so that the class could be derived from and the derived function would be called instead. The derived class was a controller class and the callback event is used for the controller update period using the sampled feedback data. This keeps the details of accessing the DAQ card separate from the control code which remains synchronised to the sampling hardware and so it's period is determined by the hardware rather than by software timers. The same callback helper function technique was used with eCos for the engine controller described in the next chapter to provide thread and timer class abstractions from the eCos C interfaces. There may be some irony in the fact that wrapper functions are required to create a C++ API for eCos when it is actually written in C++ and the internal thread objects are represented by classes, but it only publishes C level interfaces. Since eCos is open source, it is possible to use the internal C++ infrastructure directly, but its use is not documented or recommended and there is no guarantee that it may not change significantly at the internal C++ level at some time in the future.

B.3.2 Phillips SJA-11001 CAN4Linux Driver

An Arcom AIM104-CAN PC/104 form factor Controller Area Network (CAN) bus interface which is based upon Philips SA-11001 controller has been used to provide a CAN interface to the Viper board. This is to allow deterministic data exchange between the engine controller and the dynamometer controller should this be required. A modified version of the CAN4Linux driver was cross-compiled for the Viper. The driver required modification to the ISA bus address range from 16 to 32-bit data storage type due to the address range allocated to the ISA bus by the Viper. Also, the opto-coupled interface on the AIM104-CAN required inverting the logic as a configuration parameter. Once these two issues had been resolved, the driver was found to function well even though it had been written for x86 hardware with a *real* ISA bus.

B.4 Application Software

The Viper board was supplied with a cut down version of the X11R6 X Window System referred to at the time as TinyX. A very limited amount of information or documentation could be found about this, and only by examining the X11R6 source it was found to be possible to configure the X11 build process to generate the TinyX server. TinyX (or SmallX as it has more recently become known) is a highly stripped down version of the X server for embedded systems, sometimes referred to as a thin X server. Exactly what functionality has been removed and how this impacts on compatibility is not immediately clear.

A possible implementation (or reimplementation) of TinyX is called KDrive named after it's author Keith Packard. The terms KDrive and TinyX were being used interchangeably at the time this work was carried out, but it is not clear if they amounted to the same thing at some point in time. Keith Packard has been a long time contributor to the X Window System until a schism occurred in the XFree86 project which lead to Keith Packard forming the X.org project which has largely superseded the XFree86 project due to the high level of adoption and support it has received. Keith Packard is a well known member of the open source community, but despite having a personal web-page and public profile, does not seem to be directly contactable and there is otherwise very little information or support for either the TinyX or KDrive servers. This meant that upgrading the X server or 8-bit pseudo colour palette and dithering issues encountered were not resolved. A variety of window managers were tested including Matchbox, Blackbox and some of it's derivatives such as Fluxbox and Openbox. Also, a selection of applications which provide a *taskbar* or panel interface were tested. The most suitable combination was found to be fbpanel and Openbox 3 window manager, which at version 3 was completely rewritten and no longer contains any source code in common with Blackbox. Both of these programs could be configured into a minimalistic state that worked with the colour palette restrictions and the touchscreen touches which are equivalent the left click of a mouse.

The next section describes how a widget set was chosen and tested for use in the resource constrained TinyX embedded environment, so that an application could be developed for monitoring of the engine test bed which is discussed in the subsequent section.

B.4.1 Selection of a Widget Toolkit

Widget toolkits (or GUI toolkits) are sets of basic building blocks for graphical user interfaces. They are often implemented as a library, or application framework. Most make extensive use of an object-orientated framework not least as the components of a

typical GUI such as windows and menus are intrinsically objective in nature. Also these components lend themselves well to the concepts of inheritance and building upon abstract bases classes. Others which are intended for use with non-object orientated languages such as C are implemented in procedural code which may have some level of modularity.

Making a Choice for an Embedded Application The toolkit that was chosen for development of an application was wxWidgets (formerly wxWindows derived from the name *Windows and X widgets*, but changed due to Microsoft trademark infringement). It was originally created in 1992 at the AI Applications institute in the University of Edinburgh (Smart et al., 2005). This is an extensive toolkit which can also be described as a middleware as it provides many non-GUI components for various things such as strings, regular expressions, and array container classes for handling numeric data, all of which are useful for loading, handling, and saving data from files. The toolkit provided the type of codebase for the author which a team of software developers would normally have written to have at their disposal and might otherwise have taken considerably longer to develop and test. There was also the added benefit that wxWidgets provides a similar API framework to MFC (although it predates MFC) which the author has previous industrial experience with, somewhat lessening the learning curve. The cross platform and open source nature of the toolkit means that any code written could be potentially moved and cross compiled to another platform should the need arise and there is also a possibility it could be reused in other areas of work. For example a PCB drilling application was initially written to and used extensively to CNC drill the majority of printed circuit boards designed for this project. The code for this application was recycled to form the initial basis for the dynamometer application.

As a reasonable alternative to wxWidgets, there is another cross-platform toolkit called Qt (pronounced *cute*). It is was produced by a Norwegian company which has changed names many times (Quasar Technologies, Trolltech, Qt Software, and Qt Development Frameworks) and currently owned by Finnish telecommunications company Nokia who have used it for many of their handheld mobile products and have recently sold the Qt licencing rights on to another company called Digia PLC. The toolkit is open source and is free to use for non-commercial purposes, but a licence is normally required for commercial use. A embedded variant is also available. Qt based applications that were run on a PC but re-targeted to display on the Viper's LCD were found to render well, using dithering to display within the limited colour environment of the 8-bit display. An initial attempt to cross-compile the embedded version of Qt was made so that it could be compared with wxWidgets, but significant problems in the build process were encountered that could not be immediately resolved. Since time and effort had already been invested in using wxWidgets, Qt was not explored any further.

Unlike Qt, wxWidgets attempts to maintain the *look and feel* of the native environment which it is run on. For this to be possible (and to reduce the development overhead) wxWidgets makes use of other widget sets from the native environment which the library is configured then compiled for. For Windows this is the native Windows widget set, but for Unix based X Window environments, a choice of intermediate widget sets can be used which have a C API (as opposed to C++ used for wxWidgets and Qt). It allows GTK+ (GIMP toolkit), Motif, and its own universal X Windows widget set to be used, as shown in Figure B.3. Where X Windows is used directly the actual widget set is provided by wxWidgets itself in a layer called wxUniversal, but is only a partial implementation with not all features working or provided. For use on the Viper this means that unless the X Windows variant is used then the toolkit libraries upon which is relies needs to also be cross-compiled and made available on the board. The convenience of a developed application framework means that there is significant additional layering which adds to the processing overhead between making a function call within the application to the framebuffer display being modified. This overhead is insignificant for most PC based applications, but was found to impact noticeably for embedded ones. Operations requiring dynamic memory allocation or rendering were found to be particularly slow. This is most noticeable during application start up when both memory allocation and rendering are taking place. The usability was pushed towards a boundary of being too slow to be of reasonable use. For this reason all of the available widget sets which wxWidgets supports were cross-compiled and tested on the Viper.

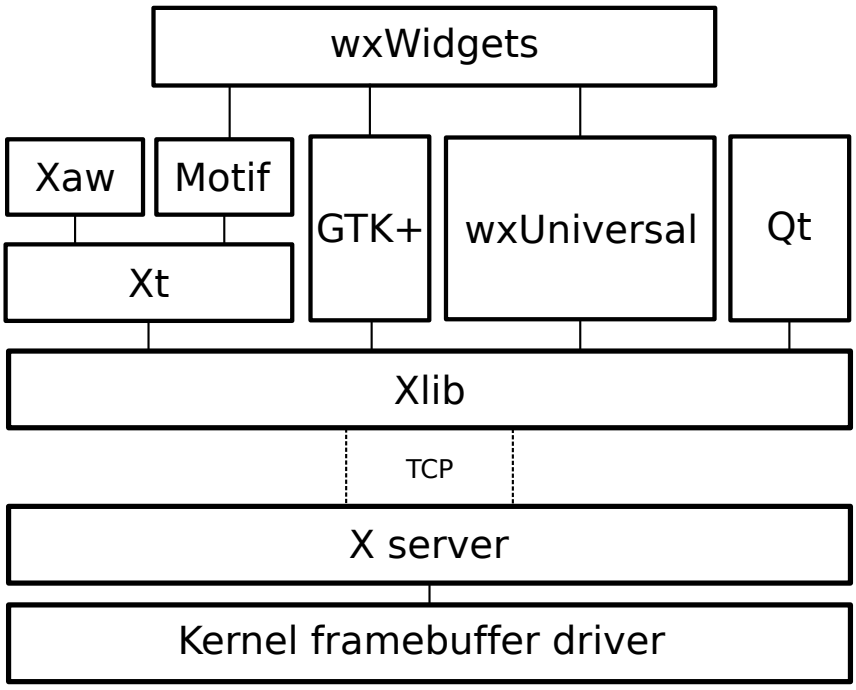

Figure B.3: wxWidgets and X Window System and widget toolkit stack

Motif Motif is a widget toolkit for building graphical user interfaces under the X Window System on Unix and other POSIX-compliant systems that emerged in the 1980s. It's 3D *chiseled* style has a somewhat dated appearance by current standards, but it's relative simplicity makes it still acceptable for industrial user interface applications where more complicated shading effects might be considered unhelpful. The Motif API is covered by the standard IEEE 1295. Both the use and distribution of the original release of Motif requires the payment of royalties, but since the year 2000 the source code has also been released under a more permissive licence as Open Motif. An independently developed open source implementation called LessTiff has also been written which provides an alternative to Open Motif whilst still conforming to the IEEE 1295 API. Both variants cross-compiled and installed alongside a wxMotif and tested on the Viper.

GTK+, The GIMP Toolkit GTK+ is a C based toolkit that was originally created (as GTK) for the GNU Image Manipulation Program (known as GIMP), a raster graphics editor, in 1997 by Kimball, Mattis, and MacDonald of the University of California, Berkeley, to provide an alternative to Motif. GTK+ does not use the X Toolkit Intrinsics (Xt) library which allows GTK+ to be more portable, using GIMP Drawing Kit (GDK) to interface to the low level Xlib. A possible disadvantage of this approach is that it doesn't have access to the X resource database, which is the traditional way for customising X11 applications. For the embedded software developer the disadvantage of GTK+ is that it has many sprawling cross-dependencies and it increasingly depends on other libraries such as ATK (accessibility), pango (international text rendering), cairo (2D vector rendering) which have their own dependencies and version sensitivities and run-time overhead. Getting all of these dependencies cross compiled and installed into the limited flash disk space is both time consuming and often challenging.

The wxUniversal native X11 interface was found to be the fastest and most responsive, but lacked a full enough implementation to be useful as some control did not render correctly or at all. wxMotif was found to be slower than wxUniversal and still had rendering deficiencies and incomplete control implementations. Both Open Motif and LessTif were tested and found to be similar in performance, although there were some subtle differences with the rendering and functioning of widgets, neither could be said to better overall. GTK+ had a fully functional widget implementation, but was found to be too slow to be useful and there was colour palette issues where each time the application (or any other application) was run, colours in the palette were modified which often lead to the display appearing corrupted or unviewable when similar colours which are intended to contrast are placed adjacent to each other. When the source code for both the wxUniversal and wxMotif layers of wxWidgets was examined, it was apparent that they had not be updated for many years during which time wxWidgets had undergone many

feature changes and major releases. This undoubtedly due to their lack of demand when compared to the mainstream wxWin32, wxGTK and wxMAC layers.

A solution considered was to use either GtkFB (GTK+ built in support for the Linux framebuffer using a single process model) or the work of the GTK on DirectFB project (GTK-DFB) to completely remove the window manager and protocol overhead of the X Window System using a windowless environment to render applications directly to the framebuffer using a version GDK which interfaces to DirectFB instead of Xlib. This in turn could in theory be used by wxWidgets without requiring any modification. Figure B.4 shows how these software layers would stack on top of each other. DirectFB stands for Direct Frame Buffer and is a self contained abstraction layer library for Linux framebuffer drivers. It is also possible to use Qt embedded with DirectFB. The GTK-DFB approach would probably give a substantial performance increase, but was not pursued due to there being too many uncertainties associated with getting all of the dependencies cross-compiled and installed on the Viper and if it would even work with the particular framebuffer driver version found on the Viper which is closely coupled to the kernel and is not merely upgradable. At the time that this was investigated, GTKFB and the GTK-DFB project were considered experimental and appeared to not be actively supported for the current GTK+ version.

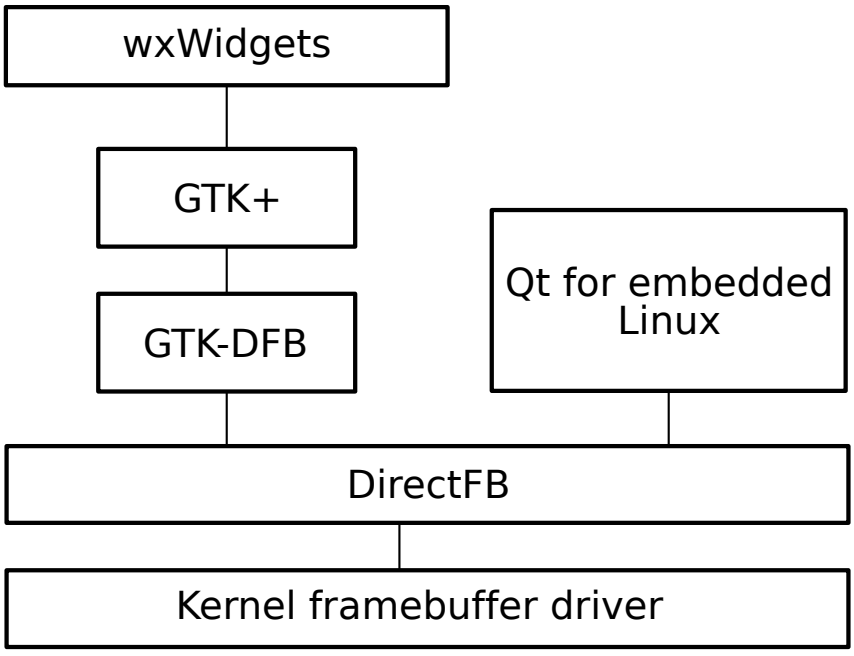

Figure B.4: wxWidgets on top of a DirectFB stack

Limitations of Legacy Pseudo Colour Support and an 8-Bit Passive LCD Display
The display used was a 8-bit Super-twisted Nematic (STN) LCD display which at the hardware level consists of an 8-bit data bus with control lines for a pixel clock, a line clock, and a frame clock. Since the bus is 8-bit the screen is only able up to 256 different

colour shades. The framebuffer device must limit itself to a palette of 256 colours whilst a dithering patterns are used to create the illusion of more colours. Qt based applications that were redirected from a desktop PC to use the TinyX server display over a network connection were able to use dithering to render reasonably well even when attempting to display a high number colours such as a web page with photographs. GTK applications, whether run on the board, or redirected from another PC, were found to display poorly, and did not attempt to dither, but instead modified the palette, displacing colours already in use and often making the display unusable through colour cycling. The disrupted colour allocation persists even after a reboot. For this reason an updated version of GTK+ was persued to allow other dependent software to be tested (such as Matchbox and fbpanel). The upgrade path was obstructed by the fact that more recent versions of GTK+ had adopted the use of a 2D rendering library called Cairo. After much experimentation with different versions a freedesktop.org bug number 4945, with the description "Cairo doesn't support 8-bit pseudocolor visuals", was found to have been reported by other users in November 2005. This had been reported in the context of GTK+ applications that had previously worked, but had started to crash when run on 8-bit X server displays. Since the bug had been reported around 2 years prior to this work and nothing had been done to fix it in the interim period, it looked as if support for 8-bit displays had effectively been abandoned. If nothing more, then it was clear that Cairo and GTK+ releases were not being tested on 8-bit displays prior to release as it was left to end-users to discover the lost functionallity. This issue was eventually confirmed to have been fixed in version 1.8.6 of Cairo by end-users who posted their testmonials in Febrary 2009.

B.5 The MPC555 and Time Processor Unit

B.5.1 Time Slicing

The TPU executes time functions on its channels. Each time function is made up of so-called *states* which are simply a group of micro-instructions. Despite the fact that there is only one execution unit per TPU, all time functions are able to run concurrently. This is achieved by the fact that the execution unit uses *time slicing* to service a channel's state, before moving on to the state of another channel. The execution unit is not able to decide which channel to service next as this is handled by the *scheduler*. The scheduler looks at all the channels which are requesting service and decides (based upon channel priority) which channel number to pass to the execution unit. Unlike a software operating system, the scheduler and execution unit are autonomous parts of the hardware architecture which allow the TPU to achieve a high degree of real-time performance with small but predictable latencies. Despite the fact that all channels can run time functions

concurrently, the more channels that are running, the longer a particular channel may have to wait for service. The latency for a channel is determined by the number of channels running, the longest state in each function, and details of how priorities are assigned. The worst case latency of a particular channel can therefore be calculated.

B.5.2 Channel Priority Levels

The TPU employs a primary and a secondary priority scheme that are both used together. The primary scheme prioritises requesting channels that have different priority levels. The user assigns a primary priority level of high, medium, or low to each channel. The secondary scheme prioritises requesting channels that have the same priority level. If no channel of the priority assigned to the current time slot is requesting service, the TPU scheduler can pass priority to other levels. Granting service to a different level channel is called *priority passing*. Priority passing is implemented in hardware and does not contribute to worst case latency.

Inevitably, channels having the same priority level will request service simultaneously, because channels can randomly request service. The secondary scheme prioritises these requests. The scheduler services channels on each of the three priority levels, beginning with the lowest numbered channel on that level. It services all of the requesting channels of a particular level, before clearing any of them for new service. This means that the choice of pin for a particular function may affect how often it is serviced which must be borne in mind when designing and allocating pins to external hardware. The two priority schemes are designed to ensure that high demand functions are serviced frequently and that there is a minimum time allocation to all channels requesting service, regardless of their priority level.

B.5.3 Code Development for the TPU

Development of custom code for the TPU might be considered a daunting task for someone with no previous experience as it requires the use of a rather obscure micro-instruction assembly language. Unlike the assembly language of a conventional processor, there are tight constraints on the exact order that instructions are relevant and permissible. There is no multiply instruction, but there is hardware support to make multiply routines easier using sucessive additions. The TPU has only ten user registers shown in Figure B.5. There are seven 16-bit registers: two timer counter registers (*tcr1*, *tcr2*); the accumulator (*a*); the data input/output buffer (*diob*); the shift register (*sr*); the event register temporary (*ert*) and the preload register (*p*). The two timer counter registers are free-running counters and are usually only read from and not written to.

Register *p* can also be used as two separate 8-bit registers *p_high* and *p_low*. Additionally there are three 4-bit registers, two of which (*chanf_reg* and *dec*) can be used as a combined 8-bit register *chan_dec*. The TPU uses dual-ported RAM to exchange parameters with the host CPU which can also be used to pass parameters between channels, and it is this mechanism which allows the state of one channel to affect the state of another such as for an angular position (input) triggered output. Each channel has eight 16-bit parameter words, the first of which is used for configuration of the channel, and the rest can be used for whatever purpose the channel's assigned function requires. The word sized parameters are often interpreted in two 8-bit halves or smaller proportions to increase the number of parameters that can be read in one access of the RAM. Each word must be accessed coeherently as a whole word (not an 8-bit read/write), so if a CPU resident program needs to write to one part of the parameter it must first read all of it, modify the subsection of the parameter then write the result back to the parameter RAM.

The TPU has five instruction formats, each of which can have one of four types of subcommands, designated by *ram* for parameter RAM, *chan* for channel, au for arithmetic unit, and finally the sequencing subcommand has with no designator. The subcommands for each instruction are separted by a semicomlon. The last subcommand in an intruction statement is terminated by a period. Each intruction takes place in two system clocks

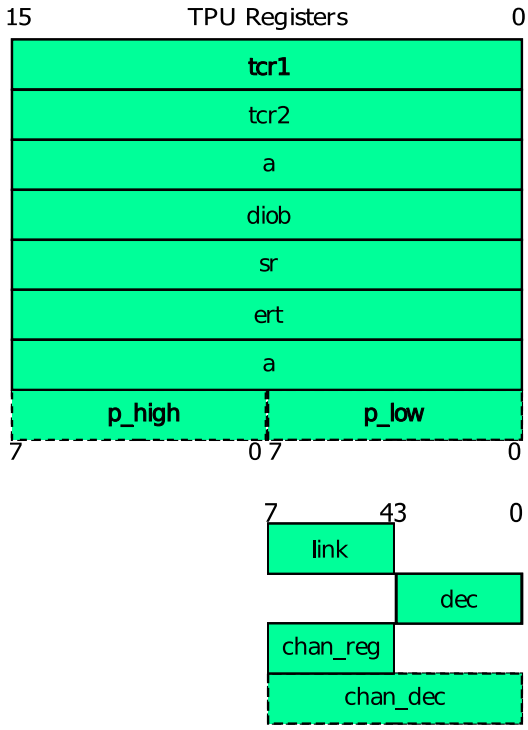

Figure B.5: Layout of the ten TPU User Registers

Examples of code for a single micro-instruction of code look like:

```
chan    PAC:=low_high, (* wait for a low to high transition *)
        neg_tdl,       (* negate transition detect latch *)
        cir.           (* interrupt CPU *)
```

also,

```
%macro MY_PARAM 'prm1'.  (* Create an alias for the second RAM location *)
au      ert:=tcr1+p;     (* add tcr1 and p then assign to ert *)
ram     diob->@MY_PARAM. (* copy diob contents to parameter RAM prm1 *)
```

and,

```
goto    LABEL. (* Unconditional branch to code beginning with LABEL *)
```

A TPU channel corresponds to a single digital I/O line. There are sixteen channels per TPU execution unit. A function has to be provided for each of the channels to be used and the same function can be assigned to multiple channels if required. All of the microcode instructions are 32-bits wide and execute in two CPU clock cycles. All of the sixteen channels share the same set of registers, but each channel has its own private RAM area which must be used to store its data due to the fact that the registers are shared.

Entry Points An *entry point* tells the TPU execution unit where to go to start executing a state. An entry point is an address that points to a time state. Each function is allowed sixteen time states, which when multiplied by the maximum of sixteen channels gives a maximum of 256 entry points. Entry points are stored in an *entry table*.

B.5.4 The Standard Masks

Historically, processors which contained a TPU were supplied with one of two so-called *standard masks* which are pre-programmed into the TPU's ROM. The standard mask contained the microcode for several commonly used TPU functions. The original TPU automotive standard mask is known as *Mask A*, and has functions shown in Table B.1.

#	Short name	Full name	Entry point
1	QDEC	Quadrature decode	6
2	SPWM	Synchronised pulse-width modulation	7
3	DIO	Discrete input/output	8
4	PWM	Pulse-width modulation	9
5	ITC	Input capture/input transition counter	10
6	PMA/PMM	Period measurement with additional/missing transition detect	11
7	PSP	Position-synchronised pulse generator	12
8	SM	Stepper motor	13
9	OC	Output compare	14
10	PPWA	Period/pulse-width accumulator	15

Table B.1: Mask A functions

Mask G, is known as the motor and motion control standard mask supports DC brushed and brushless motors, AC induction motors, stepper motors, switched reluctance motors, and multi-axis control using the appropriate functions shown in Table B.2.

#	Short name	Full name	Entry Point
1	FQD	Fast quadrature decode	6
2	MCPWM	Multi channel PWM	7
3	HALLD	Hall effect decode	8
4	COMM	Multi phase motor commutation	9
5	NITC	New input capture/transition counter	10
6	UART	Asynchronous receiver/transmitter	11
7	FQM	Frequency measurement	12
8	TSM	Table stepper motor	13
9	QOM	Queue output match	14
10	PTA	Programmable time accumulator	15

Table B.2: Mask G revision C functions

The automotive mask was the default mask supplied in ROM during the first generation of TPU production, but the control mask could be requested as well. From TPU2 onwards the microcode memory (either ROM or RAM) is arranged into pages called *banks*. As the microcode memory is shared between TPUs (on processors which have more than one), the ability to select different banks means that a wider range of functions can be assigned between TPUs. A TPU can use only one bank at a time and once a bank has been selected during initialisation it is not possible to change banks without resetting the TPU. The MPC555 contains a pair of third generation TPU3 units which are supplied with a mask made up from Mask G with the addition of some of the more generic functions from Mask A (FQD, DIO, ITC, SPWM). The full list of functions are shown in Table B.3. Most of the functions are repeated over two banks, with the execption that two new utility functions (ID and RWTPIN) have been provided in the second bank (bank 1). The first of the new functions returns an identification number for the mask, and the second allows the channel pin state to be read or written to as well as the current value of the two timers tcr1 and tcr2. When the second bank is used the functions SPWM, PPWA, and MCPWM are not available as they reside in bank 0 only. Applications which check the mask version and also use one of the three functions only present in bank 0 can do so by enabling bank 1 to perform the version check, then resetting the TPU and use bank 0 thereafter.

#	Short name	Full name	Banks	Entry point
1	SIOP	Serial input/output port	0/1	0
2	SPWM	Synchronised pulse-width modulation	0	1
3	RWTPIN	Read write timers and pin	1	1
4	DIO	Discrete input/output	0/1	2
5	PWM	Pulse-width modulation	0/1	3
6	OC	Output compare	0/1	4
7	PPWA	Period/pulse-width accumulator	0	5
8	ID	Mask identification	1	5
9	FQD	Fast quadrature decode	0/1	6
10	MCPWM	Multi channel PWM	0	7
11	HALLD	Hall effect decode	0/1	8
12	COMM	Multi phase motor commutation	0/1	9
13	NITC	(New) Input capture/input transition counter	0/1	10
14	UART	Asynchronous receiver/transmitter	0/1	11
15	FQM	Frequency measurement	0/1	12
16	TSM	Table stepper motor	0/1	13
17	QOM	Queue output match	0/1	14
18	PTA	Programmable time accumulator	0/1	15

Table B.3: Mask TPU3

If neither of the standard masks are suitable for an intended purpose then all is not lost as there is an *emulation mode* which the TPU can operate in. The CPU and TPU share an area of dual ported memory that the TPU can execute from once it is suitably initialised with the required code by the CPU. There is no execution performance penalty for using emulation mode and the TPU functions identically to the case where the code has been permanently stored in its ROM. The source code for the individual functions (freely available but copyrighted) can be used to construct a custom mask in whatever combinations are needed, or custom functions can be written and combined with the existing ones as required. The two functions from Mask A which are of greatest importance to an engine control application (PMM and PSP) have not been supplied with the TPU3 mask on the MPC555, so either Mask A needs to be used in emulation mode or a custom mask created which contains these two functions (or equivalent alternatives) to use in emulation mode. The masks are compiled into byte code from source using a Motorola compiler called TPUMASM. The compiler reads a '*.asc*' file which is created for the mask to instruct it which functions are to be included and what bank location and function number they should have. The code itself is contained in a separate '*.uc*' file for each function. The compiler outputs a Motorola *s-record* or *SREC* format file which contains S19 records. The SREC file contains an ASCII encoding of the binary. A short example mask is as follows:

```
S11F00003FFFFFFE7FFFFEFEE1E401C78E06FEF87859FEFF7A59FEFF3C7FF807BC
```

```
S11F001CD409FFFF525CB5FA163FF00B101DF80F8610FFFF36FEB01337FC4FFFFC
S11F00388411FFFFD9FF1FFF545CF18A545CF14AA406FFFFD007FFFFA417FEFFD4
S11F00549C17FEFF3C7FF807D409FFFF545CF3FA505DF3FEAC21FFFFBFFF0FFC78
S11F0070585FFFFFFFFFFFFFB021FEC3CFFF300B5C5E31FFB625FFFF1FFFF0074C
S11F008C70E9FEFB3FFFF00670E9FEFB30FFC006BC2340BFBC25407FBFFF477C88
S10B00A858583EFE5C583EFE70
S11F0620E001E001E0010029002B002B002B002BE001E001E001E001002B002B68
S11F063C002B002B302830283027101A3023402030254020100110011001100 16B
S11F0658302340203025402028142812 0002F8009019F81630063006F801F80195
S10F0674F801F801F801F801F801F801A0
S9030000FC
```

There is one record per line and each record starts with a start code which is the *S* character. Imediately following the start code is a record type code then the remaining hexadecimal byte sequence determined by the record type. The S19 SREC file format output by the compiler is so-called as it contains S1 records which are data sequences and a final S9 end-of-block record. The file is made up of S1 records with the exception of the final S9 record used to terminate the file. In an S1 record characters use hexadecimal notation. The next two characters after the record type code are used for the byte count which is the number of bytes remaining in the record. Following that there is a four character address field, then the data itself. Each record is finished with a one byte checksum. Table B.4 shows an example decomposition using the 7^{th} record from the example mask above. Once each record has been decomposed the data field can be extracted and used to build a *C char* array of the mask to copy to DPTRAM at runtime. Normally the offset address for most records will follow on from the end of the previous record, but care should be exercised as each mask will usually contain a *hole* in the address range where the microcode finishes and the function entry points start. By convention the function entry points are filled in reverse order as any unsed entry points can be used for microcode. Allocating them in reverse allows the microcode remain as a contiugous block if it is large enough to spill over into the function entry point area. The hole must be padded with bytes to ensure the correct code alignment in memory. Any unused entry points will also result in holes so it is best to assign consecutive numbers to functions to prevent fragmentation leaving only one place to pad. The hole can often be noticed by a record which is much shorter than its neighbours, as can be seen by the record marked red in the example mask above. Failure to observe this fact caused the author considerable lost time in tracing the reason for emulation mode code not working at all due to mis-aligned entry points. For example, Mask A allocates functions from 15 to 6 and the microcode occupies the space that is used for 0 to 5, but only half of number 5's entry point space is used by the microcode which leaves a gap of 16 bytes before the start of function 6's entry points. If the C array containing the mask is to be copied to

DPTRAM as a contiguous block, then these 16 padding bytes need to be added to the array in the correct place for the rest of the entry points to be at the correct offset. The full Mask A converted to a C char array is shown in Appendix B.

Record Type	Byte Count	Address Offset	Data	Checksum
S1	0x0B (11)	0x00A8	0x58 0x58 0x3E 0xFE 0x5C 0x58 0x3E 0xFE	0x70

Table B.4: Example s-record decomposition for the record S10B00A858583EFE5C583EFE70

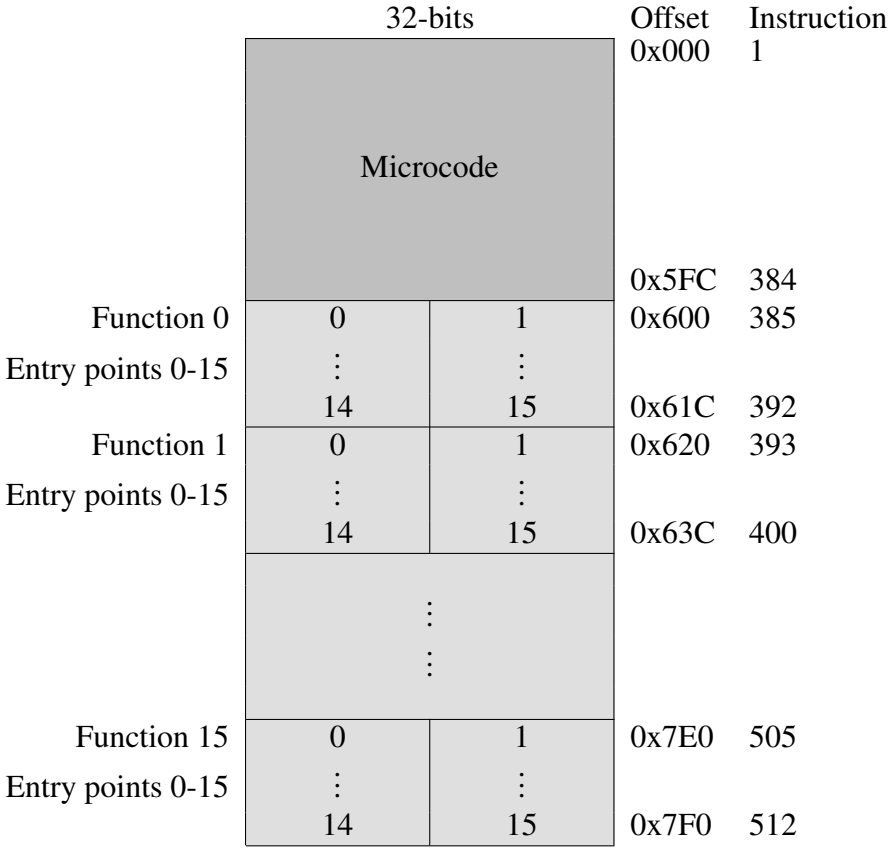

Figure B.6: Microcode and function entry point 2kB address region

B.5.5 TPU Emulation Mode

As discussed in earlier in this section, the functionallity of the TPU can be extended beyond the built in microcode supplied on ROM by uploading alternative microcode to a special area of dual-ported RAM called DPTRAM. This is required if the MPC555 is to be used for angular based engine control as the built in ROM mask does not contain any suitable functions.

B.5.6 PMM and PSP TPU Functions

The two TPU functions which are most pertinent to engine control are Period measurement with additional/missing transition detection (PAM/PMM) and Position-synchronised pulse generator (PSP). The two functions work together to autonomously determine the instantaneous angular position of the engine, triggering the fuel injectors and ignition at the correct angles in the engine cycle. Once configured, these functions are able to in-effect, run an engine with minimal intervention from the host CPU. This relieves the CPU from the critical timing aspects of the control that would otherwise only be achieved by writing carefully timed assembly language code which would monopolise the CPU in order that these timings are preserved. By adopting a combination of time-triggered and angular triggered (via TPU generated events) approach to the control software, complex control strategies are possible without compromising the responsiveness to hard real-time deadlines. The functions make use of both timer counter TPU registers. *tcr2* is set to external trigger and is connected to a position pulse sensor such as the signal conditioned output of a hall effect sensor commonly fitted to engines with electronic engine management. *tcr1* is free running at a software programmed rate derived off the processor clock and is used as a high resolution time reference to compare against *tcr2*. It is the interaction between channels which allows TPU functions to have an automous connection between input and output channels. This is done using the channel parameter RAM (registers). In the case of the PSP function using the PMM function's angular position estimate to decide when to trigger an output, the angular estimate is stored in a location in the PMM function parameter RAM. Once the channel allocation is known, the address of the angle estimate in the PMM function parameter RAM can be set by the application into the PSP parameter RAM. This is used in an analogous way to a *pointer* in C. Table B.5 shows how the parameter address byte is determined for a particular channel.

Channel Number	Parameter Address							
	0	1	2	3	4	5	6	7
0	00	02	04	06	08	0A	**0C**	**0E**
1	10	12	14	16	18	1A	**1C**	**1E**
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
14	E0	E2	E4	E6	E8	EA	EC	EE
15	F0	F2	F4	F6	F8	FA	FC	FE

Table B.5: Parameter RAM address MAP. Locations 6 and 7 in for channels 0-13 shown in bold are only available from TPU2 onwards.

Period Measurement with Missing Transition Detection The PMM function is described in detail by Darley (1997a) and is used with an angular timing disc mounted on the crankshaft and detects missing transitions embedded in a series of input pulses by

measuring each pulse period to a 23-bit resolution using *tcr1* which is set to increment at a fixed rate derived from the processor clock. A missing transition is detected when the current period (measured in *tcr1* ticks) is greater than the previous period multiplied by a programmable ratio. The ratio must be set to discriminate between the largest difference in consecutive pulses that may result from angular acceleration/deceleration of the engine and that which is caused by the so-called missing tooth. The PMM function is usually used in conjunction with the PSP function which generates an output pulse in relation to the missing tooth. It can also be used with the Input Transition Counter (ITC) function to detect the phase of the engine using a camshaft mounted once-per-revolution sensor pulse. When used in this way the function is put into Bank Mode, otherwise it is in Count Mode. Bank Mode requires more CPU intervention than Count Mode as the current tooth count has to be reset by the CPU after each missing transition as the function uses the analogy that there are twice as many teeth with two missing transitions, one in each phase of the engine. In order to discriminate between valid and false detections the function performs the following two sanity checks:

1. The total number of transitions does not exceed TCR2_MAX_VALUE
2. The number of normal transitions (between missing transitions) is equal to NUM_OF_TEETH

The first tooth is numbered zero so that the parameter NUM_OF_TEETH should be set to one less than the actual number of physical teeth. For each transition the function increments the *tcr2* register. When a missing transition occurs then the value stored in *tcr2* should equal the expected number of teeth. If this is not true then an error condition is indicated to the CPU is interrupted due to the missing transition and then on every transition thereafter until the error is cleared and it is left to the software application to decide what measures should be taken. In addition the following points should be adhered to:

- PMM and PSP channels must have the same priority level.
- PMM must be allocated to a lower channel number than PSP.
- PMM must have a high enough priority to ensure that it is serviced before the next normal transition.

The last condition can be applied more widely to the CPU as it must always be able to service the missing transition event before the next normal transition. This is so higher priority interrupts must not occur frequently enough or take so long to process that the interrupt latency is not too long for the shortest (worst case) period period between

missing and normal transitions that occurs at high engine RPM. This also assumes that the function serviced directly from an interrupt service routine (ISR), not by a routine that must be scheduled after the interrupt has been processed, such as an eCos DSR which might have to compete with queued handlers for other hardware events and be preempted by further ISR processing for other interrupt sources of any priority.

The parameter MAX_MISSING is used to set the number of missing teeth per revolution and must be set to at least one. The function can accommodate more than one missing tooth. This is so that an asymmetrical missing tooth pattern can be used to allow synchronisation to occur faster. It is not clear from the documentation if the function is able to correctly handle two consecutive missing teeth as used by Jaguar for the AJ26-V8 engine, although it appears that at least one normal transition is expected between and missing transitions. A PIC microcontroller was programmed to generate the equivalent pulse train that would be seen from the AJ26-V8, but the the PMM function could not be made to synchronise with the sequence in Count Mode using MAX_MISSING values of either one or two. The source code file for the PMM function has the creation date of “Pre 89” implying that it has existed in some form prior to the year 1989. Additional comments refer only to clean-up work and syntactical changes to keep it in-line with changes to the compiler.

An independent investigation (Tough, 2002) into the viability of the Motorola engine control functions for actual deployment revealed a chain of issues and limitations, the most serious of which being:

- Resolution is limited to 16-bits of usable precision (and not the claimed 23-bits) after which is raises an error. It continues to count up to 23-bits but the extra precision is of no further use.
- Poor crank ’syncing’ ability taking many revolutions to synchronise and the count can be out by one tooth either way after the missing tooth.
- Suffers from interference from other functions running on other channels during the missing tooth transition period which can lead to loss of synchronisation.

The issue of 16-bits (or even 23-bits) resolution is that for slow cranking speeds this can easily saturate unless the *tcr1* clock rate is reduced which compromises the measurement resolution at much higher speeds. The usable range for the 8-bit RATIO parameter is only 1 to 1.99 spread over 7-bits of precision, but more importantly the maximum ratio of ~2 does not permit particularly high rates of engine acceleration without potentially causing loss of synchronisation. These factors place significant doubt on whether the PMM function is actually suitable to run an engine without significant modification or reimplementation.

Position-Synchronised Pulse Generator The PSP function is used to generate an output transition that is referenced to a time that has been previously determined by another channel (Darley, 1997b), typically the PMM function. The PSP function can be used to provide an angular ignition dwell and trigger signal or a fuel injector drive signal pulse. The PSP function has two operating modes; angle-angle and angle-time. In angle-angle mode both the rising and falling edges of the generated pulse are determined from angles referenced to the PMM function. The angle-time mode uses an initial reference from the PMM function and the falling edge is determined by a fixed time relative to the initial rising edge. Once synchronised, the PSP function continuously generates output pulses based upon angle and ratio parameters which can be updated by the CPU.

Angle-angle mode uses the channel parameters ANGLE1, ANGLE2, RATIO1, and RATIO2 to determine the start and finish angles of the pulse. When the tooth count stored in ANGLE1 matches *tcr2* (the tooth count) then the rising edge is produced after a time determined by RATIO1 multiplied by the period pointed to by PERIOD_ADDRESS which is normally the most recent inter-tooth period calculated by the PMM function. Since the angle between the teeth is known then the ratio multiplied by a time gives a fractional angle between teeth. The pulse level is maintained high until ANGLE2 matched *tcr2* and a further time of RATIO2 multiplied by the period pointed to by PERIOD_ADDRESS has elapsed. It is therefore possible to specify the pulse start and end times in relation to the fractional angles between teeth. This can be used to control the ignition coil dwell and trigger angles in a similar manner to that which was performed by a mechanical ignition distributor and contact breaker points. If external circuitry which takes care of the coil charging is present then angle-time mode may be more appropriate as a short pulse of fixed duration at the required ignition angle may be all that is required.

In angle-time mode the parameters ANGLE1, RATIO1, and HIGH_TIME are used to determine the rising and falling edges of the pulse. As with angle-angle mode, when the tooth count stored in *tcr2* matches ANGLE1 then the rising edge is produced after a time determined by RATIO1 multiplied by the period pointed to by PERIOD_ADDRESS. The pulse is held high for a time of HIGH_TIME specified in *tcr1* clock ticks relative to the rising edge, thus it is possible to specify an injector opening angle and pulse width.

The PSP function has a number of issues documented by Motorola (Darley, 1997b) which must be worked around in software to avoid abnormal behaviour:

1. In angle-time mode, if the falling edge is set to occur before the rising edge of ANGLE1+1, (i.e. within a single tooth period) then a stream of short pulses will occur as *tcr2* continues to match ANGLE1. This can be avoided by setting

ANGLE1 to the previous tooth and using a larger RATIO1 value to bring the start angle to the same point.

2. In angle-angle mode, if ANGLE2 = ANGLE1+1 (i.e. consecutive teeth are used for the start and end reference points) and RATIO1 is greater than 128 (0x80), then the pulse will last as much as twice as long as expected. This condition can be avoided by using ANGLE1=ANGLE2 with a larger RATIO2 value.

3. If the high time of a pulse is long enough that the falling edge occurs after the next scheduled rising edge, then the next pulse fails to occur at all. Ensuring a small time gap between the expected end and scheduled start of the next pulse is the only prevention for this condition. In practise this may not be easily guaranteed with angle-time mode if the engine is under going a rapid speed change.

In addition to the limitations highlighted above, the investigation documented by Tough (2002), also raises the issue that inter-tooth angles are timed and limited to a resolution of $1/64$ of the inter-tooth spacing (0.156 degrees for a 36 toothed timing disc). It lacks the facility to cope with transient conditions which may require that the pulse duration is modified whilst it is in progress or the facility to specify an end angle or time-angle mode to prevent open valve fuelling. Also, it has been found that it can become stuck when issuing a host service request (HSR).

B.5.7 The PCI 9056 Interface

A unique and powerful feature that the PATI augments the MPC555 processor is a Peripheral Component Interconnect (PCI) bus which allows high-bandwidth interfacing to another processor. Close-coupling of the MPC555 through a PCI bus to a higher performance processor is of particular advantage for a research ECU as data collection, processing, and alorigthm implementation can be passed on to a much higher performance (but less determinstic) processor, leaving the able MPC555 to meeting the strict timing requirements of engine control. The PC/104+ specification provides for a PCI bus via a stack-through connector. A PLX Technology PCI 9056 bridge chip used on the PATI which is intended for use in I/O intensive embedded designs such as network switches/routers and industrial equipment using a hot swap/pluggable CompactPCI backplane (PLX-Technology-Inc, 2002). A PCI host is the same as other devices connected on the bus except that it is responsible for configuring the PCI bus and it must also provide an *arbiter* to provide arbitration for devices wishing to master the bus. The directions of the reset and interrupt signals are reversed for the host. The PCI 9056 can achieve this via a strapping option. The PCI 9056 is capable of performing the host role on a PCI bus as it has an arbiter, but it has been configured for peripheral use only on the PATI .

The combination of the EPLD and PCI chipset allows a sliding memory window to be used to access all of the processors internal resources and also the external memory provided on the board. This is an unusual feature that is made possible through the processor's design which allows up to eight of them to connected in parallel on a shared memory bus, with one acting as a master and the rest as slaves exposing all of their internal resources to the master. This parallel architecture is potentially very powerful as it makes a lot of intelligent digital I/O and analogue channels available on a single board. A possible limitation to this is the relatively small address space of the processor which would have to be shared between all of the processors. The PATI has only one processor on it, but if it is started in slave mode, the role of the master processor (not to be confused with a PCI bus master) can be taken on by the PCI 9056 PCI chipset, allowing access to the MPC555 internal registers from the PCI bus. This facility has the potential to allow direct hardware source debugging through the PCI bus thus eliminating the need for a separate BDM debugger device, although there is no pre-existing software for this. The PATI can be configured so that MPC555 processor's execution can be halted from power-up (MPL-AG, 2004). The PCI bus can then be used to write executable program data directly to the SDRAM on the PATI, then commence execution from SDRAM, after pre-initialising the MPC555 to boot from there.

To realise the potential of the PATI it is necessary to perform data exchange between the MPC555 and the close-coupled processor. The PCI 9056 (as configured on the PATI) provides a number of data transfer mechanisms. The IC has internal mailbox and doorbell registers for indirect parameter passing and also can perform direct data transfer between the local and host processors using direct master, direct slave, and direct memory access (DMA) transfers. It also has an implementation of a messaging queue called I_2O.

There are eight 32-bit mailbox registers that are readable and writable from both the PCI and local buses, and can be typically used to pass command and status information directly between the PCI and Local Bus devices. A local interrupt is asserted if at least one of the first four mailbox registers are written to by the PCI host. The interrupt is cleared once each of the affected registers have been read by a device on the local bus. The result of this mechanism is that synchronous transfers using just the mailbox and it's associated interrupt can only be initiated by the PCI host. Transfers in the other direction would require polling by the PCI host in anticipation or signalling by the local device using another interrupt mechanism such as the host doorbell register.

There are two 32-bit doorbell registers, one assigned to the local bus (PCI-to-local), and the other to host (local-to-PCI). The registers are bit sensitive so that if any single bit is set then it will raise a doorbell interrupt which can only be de-asserted when the set bits are cleared by writing zero to them.

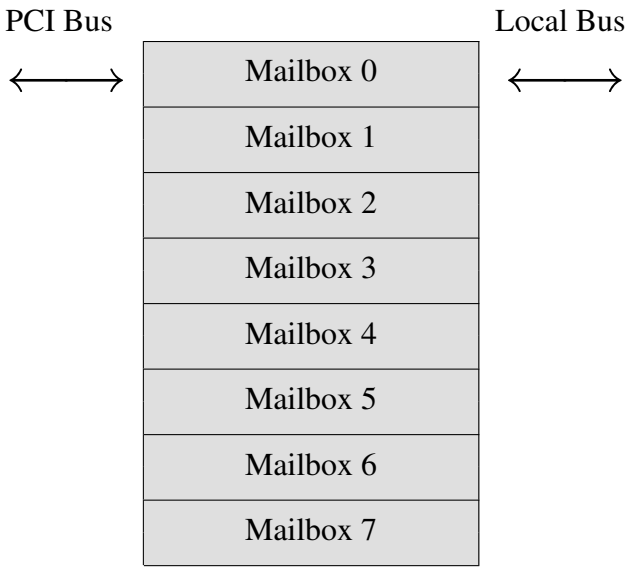

Figure B.7: PLX PCI 9056 32-bit mailbox registers

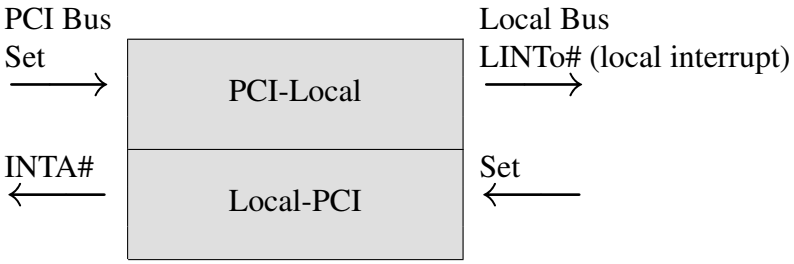

Figure B.8: PLX PCI 9056 32-bit doorbell registers

Direct Master, Direct Slave reads/writes and DMA The PCI 9056 supports Direct Master and Direct Slave reads and writes. In a Direct Master write, the Local processor (Master) writes data to the PCI Bus (Slave). In a Direct Master read, the Local processor (Master) reads data from the PCI Bus (Slave). Separate FIFOs are used to allow the processor to perform direct reads and writes indepentently of the PCI Bus. The local bus data exchanges can be performed using a burst mode memory transaction, although this is not supported by the MPC555. Direct Slave mode is similar to Direct Master, but the PCI Bus is considered to be the Master and the PCI 9056 can map transactions directly to local bus address space which saves CPU intervention.

Direct memory access (DMA) mode transfers allow a Master on either the local or PCI Bus to place a description of the entire data transfer that is required into the PCI 9056 I/O accelerator registers and the whole transaction is performed by the PCI 9056. This allows whole blocks of memory to be automatically copied directly to or from the host's memory to the target's memory. DMA transfers can be preformed simultaneously with Direct Master/Slave transfers which take a higher priority to DMA.

Device Drivers for PCI Data Transfers The PCI 9056 offers a variety of data transfer mechanisms of varying complexity. The fastest or most efficient methods require the most setup effort, but least processor intervention once the transaction is initiated. For this reason they are only suitable for infrequent transfers of large amounts of data, such as the copying entire memory regions. A graphics framebuffer would be one example of this where each frame of pixel data from a video stream may require copying from the application to graphics card memory.

It was decided to use the doorbell and mailboxes to create a syncronous mechanism for data transfer as the requirements would intially be to transfer control parameters and possibly logged data from a region of memory. This would leave the more advanced mechanism as an excercise for the future if larger or higher bandwidth transfers become required. It was decided to attempt a simple pseudo serial port driver. Since there are eight mailbox registers, and two doorbell reigisters (one for each direction) it seemed reasonable to divide the mailboxes equally between the sending and recieving directions so that full-duplex communication can be performed without interference between the two. All the registers are 32-bit double word length. One byte from the doorbell registers was allocated for indicating the number of data bytes in the transaction and the other three where made available for data. Additionally, 16 bytes from the four mailbox registers can be used to transfer a total of up to nineteen data bytes for each ring of the doorbell. Using one byte for the data byte count entures that at least one bit is set in the doorbell so that in interrupt is raised on the recipient of the data.

As eCos was to be used for the target software on the PATI, it's serial device driver API was used which is layered to allow different serial harware device drivers to use a common mechanism to provide or recieve communication data to/from an application through a software buffer. Many of the serial configuration parameters such as baud rate do not apply, which simplifies the level of lower level driver implementation which is required. On the host side, a Linux kernel character driver was written which takes up to nineteen bytes of data at a time from a connected application, and fills the mailbox then dorrbell registers throught the PCI Bus. The target eCos driver must then clear any set doorbell bits to indicate that the data has been received and copied. The target then rings the host's doorbell to indicate that the transaction is complete and that zero or more data bytes have been loaded into the other mailbox registers, and so the process continues. Due to the PCI Bus speed of 33 MHz and the parallel data exchange, transfer speeds are near instanteanous when compared to RS-232 serial or CAN which must send each bit one at time at a much slower rate. The syncronous mechanism used to transfer data requires a lot more CPU overhead per byte transferred on both sides than then Direct Master/Slave or DMA methods, but is still more efficient than either serial or CAN as up to nineteen bytes can be processed in one transaction. The driver pair was tested using a simple test applications running on the PATI and the MOPS x86 board running

CrunchBang Linux 9. Data was sucessfully transfered in both directions. The only significant isssue was resolving a PCI interrupt allocation issue on the MOPS board as some hardware such as USB appears to not work with interrupt sharing. The PATI is fitted with DIP switches which allows selection of one of the four (named A, B, C, and D) PCI interrupts it uses. It is a matter of trial-an-error to discover which of the four interrupts works without conflict for a particular host. Interrupts are normally assigned by the host PCI chipset and may depend upon the card slots ocupied on a backplane. For a PC/104+ formfactor there is no backplane so *changing the slot* is not an option to resolve conflicts. The PATI's DIP switches allow another means to c*hange the slot.* When an interrupt occurs on the host side, the Linux kernel calls each driver which has claimed ownership of a device which shares the interrupt in turn. Each driver must determine if it's device was the source of the interrupt (without causing further interrupts to occur) and inform the kernel. If no driver claims the interrupt then it is treated as spurious by the kernel and intially ignored then later masked if it continues to occur and not be claimed. If two devices generate an interrupt at the same time then the interrupt will persist after the first device has been processed (and cleared it's interrupt source) and the interrupt will retrigger so that the second device's driver will be given the chance to process it's interrupt source. Due to this mechanism, an interrupt handler may be called much more frequentely than anticipated and must endeavour to determine if it's device is a source for the interrupt as quickly as possible to minimise the overhead to the rest of the system. For the PATI PCI driver, this means reading the mailbox registers and checking for set bits.

There is more work required to make the pseudo serial driver useful. As with a real serial link, there needs to be a protocol in place to allow meaningful data exchanges to take place and application code to process the protocol and provide and interpret the data passed. One such protocol already exists for CAN based communication of data and variables, between an ECU and a piece of calibration software, and is called CAN Calibration Protocol (CCP) (Vector, 2004). CCP is simple and flexible enough that it could be directly implemented, except it then raises the question of why a pseudo serial port has been implemented and not pseudo CAN. In fact since CCP only makes use of CAN's 8 data bytes and two fixed message identifiers, then either the existing pseudo serial (with slight modification to identify blocks of 8 bytes), or a pseudo CAN driver would work. It actually makes some sense to use a pseudo CAN link since only the data bytes subset of CAN would be required (the message identifier would be redundant as the message recipient is implicitly known) and the CAN4Linux infrastructure could be used on the host side and the eCos CAN driver API layer on the target side. It could also provide a level of indirection so that an actual CAN link could be used instead with minimal additional effort (assuming MCP555 TouCAN drivers had been already written).

B.5.8 The U-Boot Bootloader

The name *U-Boot* was inspired by the German WW2 film Das Boot, and is perhaps a play on words between the English term for a *boot* and the German for *boat* or submarine even though the German word boot is pronounced as the English boat.

U-Boot is an open source project to provide boot-strap support in the form of a boot-loader to a large variety of hardware platforms. It is largely based on the Linux kernel sources and is intended to allow very early platform specific start-up code to be removed from the kernel image and placed into the boot-loader itself. This makes kernel images more portable and generic within a given architecture and should greatly reduce the need to compile a kernel image for every single platform variation of a given architecture.

The PATI board is supplied with U-Boot programmed into the external flash. PATI is not capable of running a standard Linux kernel as the MPC555 has no MMU which is a basic requirement for Linux. Although highly modified versions of Linux have been developed for basic processors with small memory footprints and no MMU, Linux in its native form is not real-time and therefore not suitable for use as an engine controller. Even though real-time patches and other schemes such as RT-Linux and RTAI exist it is still unwieldy and unsuitable for use as an engine controller particularly as no direct user interaction is required which makes large parts of its functionally somewhat redundant.

U-Boot is not an operating system in its own right, but can run specially linked code with some *libc* services. For very deterministic procedural code that has no task, thread, or device driver abstraction, this may be all that is needed. However it is not an adequate solution for anything other than procedural code or simple co-operative multitasking, i.e. no device drivers so therefore it would not be much use as the basis of an engine controller. Powering up the PATI results in the following serial console output:

```
U-Boot 1.1.4 (Jul 17 2006 - 13:21:28)
(c) 2003 by MPL AG Switzerland, MEV-10084-001 released
CPU: MPC555/556 Version 2 at 40 MHz:
Board: PATI-1 Rev A SN: 107
DRAM: 16 MB
FLASH: 4 MB
In: serial
Out: serial
Err: serial
pati=>
```

The serial console command prompt is the only way for a user to interact with a program running on the board as there are no USB, Ethernet, or PS2 connections to provide the usual familiar interfaces.

B.5.9 The MPC555 Address Map Problem

All of the registers, internal memory, and other resources of the MPC555 are accessible from a 4 Mb region within its address space. This region does not have to be fixed in any one location as it can be placed in one of eight locations shown in Figure B.9. U-Boot requires that normal memory, or RAM is available from the address 0x0000000 which in the case of the PATI corresponds to the external SDRAM. Given that the default MPC555 memory layout (that used by convention, not by necessity) is to place the IMMR in the 0x0 slot, this presents a problem.

The MPC555 has many of the resources of a typical embedded system located on-chip which should make much of the existing code written for it relatively portable between platform variants as the absolute addresses of those resources need not change.

After examination of the source code, it appeared that some existing code for the MPC555 (examples are Codewarrior example platform code, Matlab Real-Time Workshop bootloader, and eCos) seemed to have been written with the assumption that the IMMR is located at 0x0. From the comments in the code and its structure, the authors of much of this code appear to have recognised the possibility that the IMMR might be relocated from its default location and have made some use of compiler macros or relative addresses to define the offsets of the various system registers. However the actual implementations appear to be incomplete (and therefore untested) in that respect as there are always some dependencies on hard-coded address locations meaning that further work and testing would be required to get them working correctly with the IMMR anywhere other than in the 0x0 position.

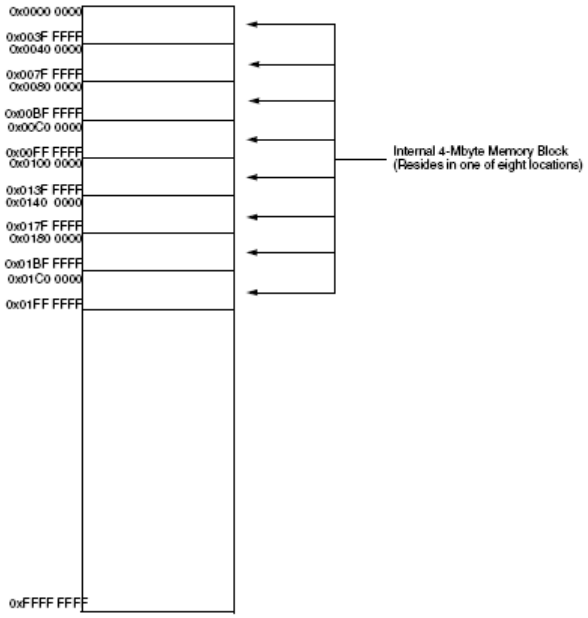

Figure B.9: Diagram of the eight possible IMMR locations within the full address space

The solution is to replace U-Boot and in order to remove the 0x01C00000 dependency to allow more standard MPC555 code to be compiled and run provided the hardware has been initialised correctly by some board specific start-up code. However, this is far from straightforward. Because of the unusual hardware configuration of the PATI board (having resource access through a PCI Bus), U-Boot initialises an external memory controller (EPLD) and the SDRAM which must be brought online and for correct refreshing and allowed to settle before memory can be accessed reliably. The SDRAM is needed for storage of the application image as the processor only has 26 Kb of SRAM which is sufficient to provide a stack for the boot-loader, but not much else.

The EPLD initialises the processor's start-up state on power-up and after a hard reset. The EPLD is factory programmed to force the IMMR to 0x01C00000 using the processor's Hardware Configuration Word (HCW). The EPLD asserts the HCW on the data bus to initialise the processor which stores it in a register, and HCW is set to place the IMMR into its furthest slot away from 0x0 (0x01C00000) to allow for 16 Mb of SDRAM to start at 0x0. Paradoxically the IMMR location is set by a register within the IMMR itself which moves with it to the new location when it is changed. The HCW can be partially controlled using board mounted DIP switches to select certain operation modes. Once the processor is running the HCW register can be changed in software and after any subsequent resets the changes are applied. It should be possible to get U-Boot to perform this operation and reset itself, or to halt execution using a hardware debug tool and to directly modify the HCW and perform a reset.

To permanently change the HCW used on power-up, so that U-Boot is no longer required, the value stored for the HCW must be reprogrammed in the EPLD. This can

only be done using the PCI chipset EPROM channel which requires the co-operation of the PCI 9056 chip. MPL have provided the necessary function calls in a special version of U-Boot which can be called from the command prompt. However any of the commands which reference the PCI chipset seem to fail, perhaps because no host is present the PCI 9056 is unable or unwilling to co-operate. As there is no PCI host, any memory accesses to the PCI region were found to cause machine check exceptions, i.e. a crash. This makes reprogramming the EPLD without a host (or direct reprogramming of the EPLD using a JTAG tool and factory firmware) impossible.

The only remaining solution is to make the MPC555 use its internal *shadow* flash to get the HCW at power-on rather than to load it from the external memory bus. This also requires the co-operation of the EPLD which is done using the on-board DIP switches. The shadow flash is a small memory region (Table B.6) that is page selectable in place of the normal internal flash starting at the same memory location, at the beginning of the IMMR, usually at 0x0.

Offset Range	Description
0x00 0x03	Reset configuration word
0x04 0x0F	Reserved for future applications
0x10 0xFF	General use MoneT shadow information

Table B.6: Table of shadow flash layout

The shadow flash region shown in Table B.6 can be programmed using a suitable hardware debugging/programmer tool and the appropriate software.

B.6 Porting eCos to PATI

With the provision of a board port for eCos, the Redboot bootloader (itself a minimal eCos application with no kernel included) was compiled and flashed into the MPC555 internal flash, a process which is quite convoluted, but only needs to be performed once per board. It was done by compiling a modified version of U-Boot which was changed to enable the internal flash programming voltage by writing to a register in the EPLD. U-Boot is able to perform a self-update. The cross-compiled U-Boot binary image is converted into the correct format and uploaded over serial to the PATI using U-Boot. U-Boot is then able to determine that the binary is a valid U-Boot image and then update the external flash with the new image. Once the board has been reset, the modified version of U-Boot runs and enables the internal flash programming voltage. At this point

a hardware debugger is used with the BDM port to take control of the processor, set the IMMR region back to 0x0, then perform a soft reset of the processor (leaving the programming voltage enabled). The harware configuration word in shadow flash can then be written as described in the previous section. It was found that this must be done before writing to the main flash area as the erase operation for the shadow region also clears the main region. The internal flash can then be programmed with the Redboot image using Codewarrior. Codewarrior uploads a small program into the MPC555 on-chip SRAM using BDM and excutes it. The program receives the image to be flashed over BDM and performs the flash writing operation in stages. With the internal flash written, the PATI can be powered down and the DIP switch setting is changed to use internal flash to provide the harware configuration word (HCW) instead of the EPLD. Once powered back on, the PATI will then be able to boot Redboot from internal flash from reset and provides the following output over serial:

```
+ RedBoot(tm) bootstrap and debug environment [ROM]
Non-certified release, version PATI - NCL - built 12:30:01, Jul 16 2008
Copyright (C) 2000, 2001, 2002, 2003, 2004 Red Hat, Inc.
Copyright (C) 2003, 2004, 2005, 2006 eCosCentric Limited
RAM: 0x00400000-0x013fffff, [0x004088c0-0x00fe0fff] available
FLASH: 0xffc00000 - 0x0, 64 blocks of 0x00010000 bytes each.
RedBoot>
```

The external flash was configured to have the following contents:

```
RedBoot> fis list
Name              FLASH addr  Mem addr    Length      Entry point
FIS directory     0xFFC00000  0xFFC00000  0x0000F000  0x00000000
RedBoot config    0xFFC0F000  0xFFC0F000  0x00001000  0x00000000
jffs2             0xFFC10000  0xFFC10000  0x00050000  0xFFFFFFFF
nano              0xFFC60000  0x00410000  0x00050000  0x00410000
enginecontrol     0xFFCB0000  0x00410000  0x00080000  0x00410000
```

The external flash is formatted to have a FIS partition which allows it to be used like a disk, having a partition table. The first partition is the partition table itself which describes the rest of the flash contents. The next partition contains Redboot configuration parameters such as the default baud rate and whether to use a startup script to automatically load and execute an application for example. A JFFS2 file system was created in the next partition to provide a small filesystem for the application to use for parameter storage or configuration data. The JFFS2 partition contains only one XML formatted configuration file, but could contain more if required. The minimal console

based text editior called *nano* was cross-compiled as an eCos application using a port of *pdcurses* (public domain curses, a light weight re-implementation of *ncurses*) and takes the next partition place. The editor can be used to load the XML file from the JFFS2 partition, edit and save it, requiring nothing more sophisticated than a terminal program to edit application parameters (such as controller gains) in place on the board. The last partition defined is for the engine control application and is positioned last to allow it space to expand as it is developed and functionallity is added. Applications are run by loading them from flash into the external SDRAM and excuting them from there rather than directly from flash. The external SDRAM was used since Redboot is already using part of internal SRAM for it's stack so this was the most straight forward approach.

B.6.1 Real-time Performance

Being real-time capable is an obvious requirement for a low-level engine controller, however there are different levels of real-time. As far as eCos is concerned, its real-time capabilities are relatively good by virtue of the kernel's design, although there are no absolutes, as the exact performance depends upon the precise configuration and target hardware. It has to be accepted that the determinism will always be less than the equivalent procedural code and there is an overhead associated with context switching. The kernel can optionally be instrumented to assess various latencies and the effect of the actual instrumentation is taken into account. It must also be conceded that any dynamically scheduled software system which is subject to potentially random external hardware interrupts will suffer reduced determinism, thus reducing its predictability. eCos reduces this affect by keeping interrupt service routine (ISR) durations to a minimum. As a device driver implementer you are advised, but not compelled to adhere to this bare minimum ISR approach. The body of work that has to be performed in response to an interrupt is done in a deferred service routine (DSR). These DSR's run in their own context once an interrupt has been acknowledged. The occurrence of another interrupt from another source will cause the execution of a DSR to be suspended until the relevant ISR has been processed. Application code is blocked until all the DSR's have finished or yielded. This behaviour allows good overall performance with low latencies, but does make application code less deterministic as even the highest priority thread has to wait for DSR's to complete. The exact interrupt behaviour is configurable as to whether interrupts use their own stack or whether nested interrupts can occur and if any arbitration methods are used.

B.6.2 Open Source Architecture

The availability of the source code is an important consideration for the requirements of this project. The PATI has a unique hardware configuration and as a consequence of this

there was no pre-existing software support for it. It would have been possible to run a closed source proprietary OS on it, but when inevitable difficulties arise with the particular hardware it would unlikely have been any assistance would be available without an expensive support contract, and it might not have been possible to resolve those difficulties otherwise. The fact that eCos is open source makes it transparent and the full execution path, from application code, through kernel code can be stepped though and debugged. It also demystifies the workings of the kernel and makes porting to new hardware possible. It should be stressed though, that *possible* does not necessarily mean it is easy or time and cost effective. It is always a trade-off between time, effort, and cost and therefore open source does not equal free.

B.6.3 Configurability and Scalability

eCos is made highly configurable by the extensive use of compiler directives and conditional compilation. It is intended for firmware applications in highly embedded systems that have fixed hardware and have very little need to change configuration at run-time. This makes it possible to compile only the code that is absolutely required for a particular application on a particular piece of hardware. It is generally considered bad programming practise to litter source code with conditional statements of compiler directives as it can lead to maintenance issues, effect code readability and introduce bugs into code. eCos embraces and manages this through the use of a TCL derivative script language called CDL. A set of utility programs are used to manage the inclusion of packages and the configuration of all of the various options. These programs are able to parse the CDL options and generate a build tree with all of the required header files to provide the necessary compiler pre-processor defines for a particular configuration. To further reduce the size of the resulting application binary, the eCos developers had a special extension implemented in the gcc linker to force it to remove any unreferenced (unused) code sections. A typical package might include an optional device driver, or a software component such as a TCP/IP stack. As a result of this lean approach, a typical self-contained eCos application raw binary image, which includes all of the kernel code and the actual application, might be less than 100 kB in size and boot within one second. When compared to Embedded Linux, whose kernel alone is over 6 MB and can take several minutes to boot from flash.

B.6.4 Serial Interrupt Delay Problem

The use of the hardware serial device drivers with the transmit interrupt enabled revealed a further issue. This was at first thought to be a bug or limitation of the driver code, but after investigation proved to be an unfortunate limitation of the hardware, which is a

Algorithm B.1 Serial Transmission

1. Application calls cyg_io_write() with char array
2. IO layer driver code unmasks Tx IRQ
3. Tx interrupt imediately fires as buffer is already empty
4. ISR masks Tx IRQ and posts request for serial DSR
5. DSR calles generic serial.c code which calls device specific code
6. Device code copies a single character the returns success
7. Shift register copies character to empty hardware for immediate dispatch
8. serial.c attempts to copy next character, calls device code again
9. Device code copies a single character to now empty buffer returns sauces
10. serial.c attempts to copy next character, calls device code again
11. Device code copies find buffer not ready, returns failure
12. serial.c code returns to DSR, DSR unmasks TX IRQ
13. Repeat from 4. until all characters are sent

good lesson in real-time performance. The MPC555 has two RS-232 capable serial ports as part of its Serial Communication Interface (SCI). Both of these serial channels have only a single byte transmit buffer which uses a shift register to load the character to be transmitted into the actual hardware. In addition one of the two serial channels has an optional 16-byte queue which can be used like a circular buffer. The original driver code only implements this using the single byte buffer for both serial channels so as to keep the code non-specific to either channel as a slight abstraction. There are two types of transmit interrupt which can be used with the MPC555 SCI interface. The first is 'transmit buffer empty' and the second is 'transmission of the last character has completed'. The driver makes use of the former interrupt to load the next byte in a sequence into the buffer. The software transmit process is shown in Algorithm B.1.

A consequence of this was found to be that an application thread set to loop at a 10 ms interval, using a delay timer, was completely blocked from running during the transmission of a 64-byte serial packet at 57600 baud. This is because the DSR code takes precedence over application code and at that baud rate there was insufficient time for the kernel to context switch to the application thread. To determine that this was the case and that there wasn't an interrupt problem, certain parts of the code were instrumented. Two counters where placed into the low-level driver code to maintain a count of bytes successfully copied to the transmit buffer and unsuccessful attempts when the buffer was still full with the previous byte. A hardware debugger was used to

interrupt the execution of the code at non-specific intervals. A ratio of 2.06 successfully copied bytes to unsuccessful was always observed. I/O lines were also used to allow oscilloscope measurements to be taken of the thread execution and the serial transmit time for each packet so that the interference of the threads execution could be observed. By varying the baud rate it was found that 9600 baud caused only very slight, barely noticeable, jitter on the thread's timer event. Increasing the baud rate to 19200 baud introduced pronounced noticeable amount of jitter. A baud rate 36400 bits/s caused sever jitter in the occurrence of the timer event. 57600 bits/s again totally prevented the occurrence of the timer event. The driver uses the size of the software buffer (a CDL option) to determine if interrupts are to be used. With a buffer size of zero interrupts are not used, but the same code is called directly from the thread context. As the code is not called from a DSR it is subject to the same priority as the calling thread and may be preempted by the scheduler if a higher or equal priority (if time-slicing is enabled) thread is ready to run. This means that without interrupts being used a higher priority thread is not perturbed by a serial transmission. The downside of this is that the transmit code has no way of knowing when the buffer is ready to receive a byte and has to keep testing until it becomes empty.

There is an important lesson to be learnt from this which is even when using a minimalistic and deterministic real-time operating system such as eCos, the real-time performance of application threads cannot be guaranteed in the same way as it can be with procedural code. In this case the problem was also in part to the overhead incurred though having a kernel perform context switches and in part due to the fact that the serial processing is done in the context of a hardware event (interrupt) which is given precedence over running application code. The only way to resolve this, so that even another interrupt source could not potentially cause the problem to manifest again, would be to move the periodic application code into a device driver and have it activated by an external clock driven interrupt or the system RTC. Interrupts can also be given priorities and the serial interrupts could be set to a lower priority. However this is undesirable and if possible it would be preferable to make the application code (perhaps a control loop) more robust to timer jitter.

In the case of the MPC555 based engine controller, the severely time critical aspects are taken care of by the two TPU's leaving much less critical (but still hard real-time control code) to run as application code.

B.6.5 Creation of a TPU Device Driver for eCos

The approach normally adopted for eCos drivers is that of the hardware abstraction layer (HAL). The eCos HAL approach is to create a generalised I/O layer which the application uses to connect to the hardware. The I/O layer in turn interfaces with a low

level driver layer. This approach allows the same I/O layer to be used with different physical hardware without the application needing to have any hardware specific functionality. It also allows the hardware to be changed without modification of the application as only the low level driver layer needs to be exchanged. CDL is used to configure which I/O layer a driver depends upon and which hardware is present on a particular processor board implementation.

When implementing the TPU driver, it could have been approached from two directions. The first is to consider the class of device being controlled and make that the I/O layer abstraction, hiding the TPU implementation details within the driver. An example of this might be to have an incremental encoder driver. The device at the I/O layer would be an encoder and would be concerned with things like counts per revolution, total revolutions or angular distance moved, and current RPM. The low level driver would then take care of configuring the TPU and assigning the fast quadrature decode (FQD) function the the relevant channel. This approach would allow the I/O layer to be preserved if the hardware was changed to not use a TPU, maybe instead a dedicated IC. If different TPU functions are used together, each having different device class abstractions, then this approach quickly becomes difficult to manage as each function, although consider separate, still requires configuration and use of the TPU.

The approach adopted for this work was the converse to the one described above. Instead, a TPU driver was created which allows a functions to be allocated to each channel at the CDL level. This method still configures the channel allocation at compile time rather than run-time, but is more flexible when it comes to allocating functions to channels. Figure B.10 shows the configuration tree options for TPUA (TPUB has the same repeated options) using the eCos configuration tool to enumerate the CDL option tree. The setup of the module configuration registers (MCR) is done on a per TPU basis. This could be done at run-time, but the since configurations are for fixed hardware, it is possibly more robust to determine the required settings offline an allow the parameters to be set at compile time from CDL. Using CDL offline simplifies configuration process for the TPU and demystifies it as the resulting TCR frequencies are shown in CDL and do not need to be calculated at run-time. Many Cryptic hard-coded *constants* are removed from the code improving ease of maintenance and reducing continued reliance on datasheets and application notes. There is an option to load a TPU mask from a C function containing the mask encoded as C char array. This defaults to the Motorola Mask A. This mask must be shared by both TPUs if they are set to use emulation mode, as determined by the hardware.

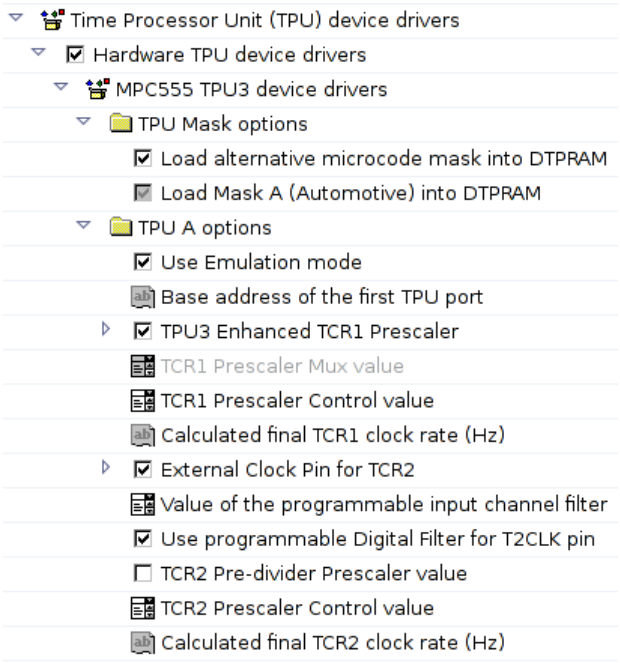

Figure B.10: TPU Tree TPU Options

Figure B.11 shows the next level in the CDL hierarchy where individual channels can be assigned a specific microcode function and generates the driver mount point based on the function name and the channel number for the application to use to access, such as "/dev/itcA0" if ITC was assigned to the first channel of TPU A. The limited complexity of CDL mean it is not possible to completely restrict a set of selections to valid ones, for example it does enforce that the PMM function is assigned to a lower channel number than PSP, so doesn't totally remove the need for some understanding of how the functions work, but caveats and limitation can be documented in the description text for each setting if required. Figure B.12 shows that specific function drivers can be enabled for each TPU. Enabling a particular function driver specifies whether the code for that function is compiled into the driver. The TPU is configured when the first channel is opened from the application and *looked-up* by the kernel code. When the setup code has been run the TPU is flagged as configured so that subsequent calls to configure other channels do not repeat the configuration process. Drivers were created for ITC, FQD, PWM, DIO, PMM and PSP functions. All of the functions apart from PMM and PSP have been successfully used as part of another project (mobile robotics DC motor control) that the author has been involved with. The remaining two functions have not been fully tested initially due to problems with uploading the code in emulation mode, then due to time constraints preventing the work from being complete. However, it is suspected that both the PMM and PSP functions would require microcode modifications to work correctly for this project and is discussed further in Section B.5.6.

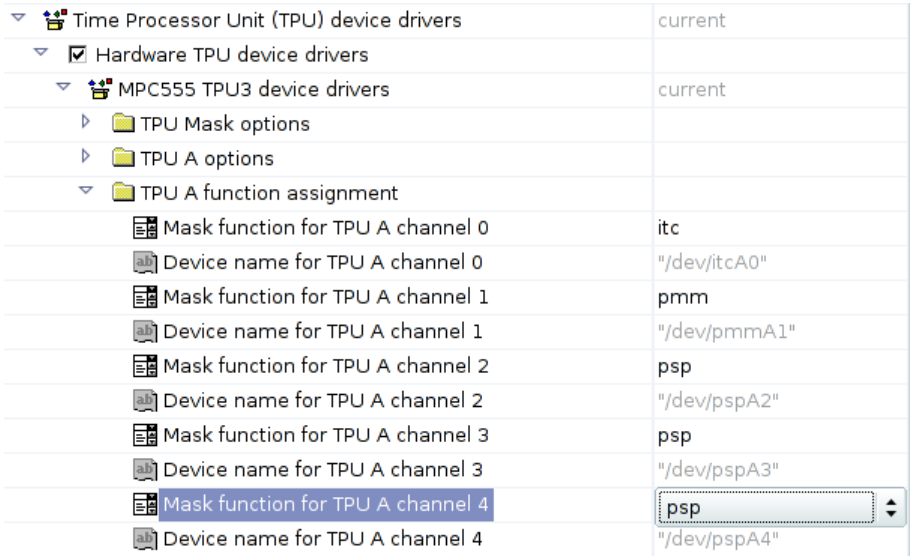

Figure B.11: TPU Tree function options

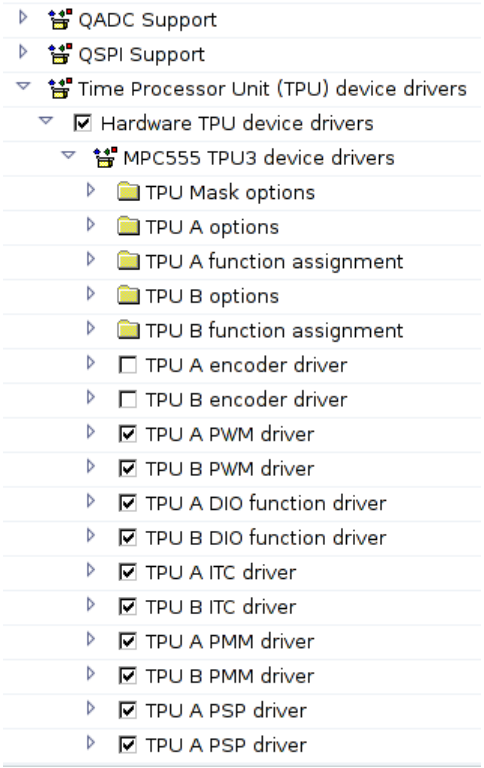

Figure B.12: TPU Tree CDL enumeration