PHD

Dynamic timing of internal combustion engines using non-intrusive sensors

Mobley, Christopher G.

Award date:
2001

Awarding institution:
University of Bath

Dynamic timing of internal combustion engines using non-intrusive sensors

submitted by

Christopher G. Mobley

for the degree of
Ph.D. at the University of Bath
2001

Acknowledgements

I would like to express thanks and gratitude to the following who provided guidance and help throughout the duration of this work.

To Dr R.W. Dunn for supervising this work and providing invaluable assistance and encouragement over the last 8 years; to Dr J.G. Hawley for guidance and wisdom and for providing the test facilities and generally looking after me; to Alan Cox and Tony Elley for helping with the test sensors and results and making life in Wolfson so enjoyable; to Tony Solomonidies for providing time & equipment for me to complete this project; to Dr Rob Williams for reading through endless revisions of the thesis; to Dr Nick Vaughan for all his help in the later stages; and finally to Nicky for her tireless encouragement and understanding.

Abstract

The monitoring of in-cylinder pressure on internal combustion engines is a costly and intrusive process. Current methods generally employ the use of miniature pressure transducers mounted in the cylinder head. These transducers have direct access to cylinder pressure and so give a high degree of accuracy. The disadvantage of using this intrusive method is the cost of the sensors and required modification of the cylinder head. Accordingly, the monitoring of in-cylinder pressure is usually restricted to research engines or large marine engines. However, the information carried in the pressure trace is significant and of benefit in condition monitoring and control. Therefore, the aim of this research was to produce a low-cost, non-intrusive sensor for the monitoring of in-cylinder pressure; this would have the potential to extend the use of pressure information to volume production engines.

The first stage of the research was to develop and validate a novel sensor which could be sited on the outside of the engine and which would have the ability to record variations in load due to the combustion pressure cycle. This was successfully achieved.

The second stage of the research was to develop a signal processing strategy which would enable two key points of the combustion process, namely start of combustion and peak pressure, to be identified. The main problem to be overcome was that the signals produced were contaminated by a number of noise sources. The method selected to overcome this problem entailed the use of wavelets and artificial neural networks. This wavelet-neural network approach enabled the system to detect patterns in the data, resulting in a final output - based on crank angle – of high grade signals showing start of combustion and peak pressure.

This thesis describes the work undertaken to design and validate the non-intrusive sensor and associated signal processing system. The conclusion drawn is that a low-cost system for the measurement of in-cylinder pressure is achievable, and areas for future development work are identified.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Project overview

This project was concerned with the development of non-intrusive sensors and signal processing techniques to enable the low-cost measurement and characterisation of cylinder pressure from internal combustion (IC) engines to take place. The rationale for the project stemmed from the requirement for IC engines in volume production vehicles to meet current and future legislation on the control of toxic emissions.

Recent advances in pressure-based control systems have utilised key measurements of the combustion process (such as location of peak pressure and start of combustion) taken by intrusive sensors. However, the high cost of intrusive sensor systems is prohibitive for their use on volume production vehicles. For these, a system based on the use of non-intrusive sensors would enable a cost effective solution - but the problem with using data from non-intrusive sensors is that the signals generated can be contaminated. Therefore, the research undertaken for this project focused on ways of overcoming this drawback by developing methods of feature extraction and detection.

The project culminated in the development of a low-cost sensor and signalling system which provided two high grade signals indicating the start of combustion and the location of peak pressure. This was achieved by the use of a non-intrusive sensor, wavelet decomposition and artificial neural networks.

1.2 Background

The environmental impact of burning fossil fuels in an IC engine produces a net increase in atmospheric pollutants. With governments around the world under pressure to limit emissions from such engines, controls on exhaust gases have become more stringent. Therefore the engine of the future must aim to have an exhaust virtually free of toxic substances and to have a significantly improved overall fuel economy in order to limit the production of carbon dioxide. Renewable energy sources such as alcohol, hydrogen and fuel cells do not have this drawback, but are not expected to be widely available in the short to medium term.

Standard	**Directive**	**Type of vehicle**	**Date of introduction (for type approval)**
Euro I	91/444/EEC 93/59/EEC 91/542/EEC	passenger cars light commercial vehicles heavy diesels	31 December 1992 1 October 1994 1 October 1993
Euro II	94/12/EC 96/69/EC 91/542/EEC	passenger cars light commercial vehicles heavy diesels	1 January 1997 1 October 1997 1 October 1996
Euro III	98/69/EC Common Position	passenger cars & light commercial vehicles heavy diesels	1 January 2001 1 January 2001
Euro IV	98/69/EC Common Position	Passenger cars & light commercial vehicles heavy diesels	1 January 2006 1 January 2006[1]

Table 1 Emission standards

The European Union (EU) has produced a set of mandatory emission requirements [1], which each member state must implement (see Table 1). The EU directives specify the limit of emissions produced by a vehicle per kilometre [2] and these must be attained before a vehicle is type approved (see Table 2).

Turner [3] noted that in the development of cleaner, more fuel-efficient engines to meet the legislation, electronic control has proved to be a major contributing factor. In his review of current sensor technology for automotive applications, he remarked that electronic control of engine and transmission is the key to optimise economy, emissions and performance. He stated that these new systems rely heavily on sensors and measurement techniques and, as a general rule of thumb, a

[1] Further NOx emission standards will also be introduced from 1 January 2009 as part of the same directive

vehicle manufacturer will normally only tolerate a ‘measurement cost’ of around $10, including the signal conditioning (1999 prices).

Euro Standard	Number of seats	Fuel	Directive	Limit values (g/km)					date
				CO	HC	NOx	HC+NOx	PM	In-use
Euro II	Up to 6	P	94/12/EC	2.2	-		0.5	-	01/01/97
	Up to 6	D	94/12/EC	1.0	-		0.7	0.08	01/01/97
	Up to 6	D(di)	94/12/EC	1.0	-		0.9	0.1	01/01/97
Euro III	up to 9	P	98/69/EC	2.3	0.2	0.15	-		01/01/01
	up to 9	D	98/69/EC	0.64	-	0.5	0.56	0.05	01/01/01
	Note [2]	D	98/69/EC	0.8	-	0.65	0.72	0.07	01/01/02
	Note [3]	D	98/69/EC	0.95	-	0.78	0.86	0.10	01/01/02
Euro IV	up to 9	P	98/69/EC	1.0	0.1	0.08	-	-	01/01/06
	up to 9	D	98/69/EC	0.5	-	0.25	0.30	0.025	01/01/06

Table 2 Emission limit values

Stone [4] had already identified two approaches to the electronic control of engines. The first was to use a memory for storing the optimum set of variables, such as ignition timing and fuel mixture strength, for a discrete set of operating conditions (called mapping). The second approach was to use an adaptive control system to continuously optimise the engine at each operating point (closed loop control). Mapping can be performed on a single engine or group of engines. For mass production, mapping is usually performed on a small sample which is representative of the batch. However, Stone recognised that the mapping technique had one serious drawback, in that tolerances introduced into production engines resulted in

[2] Temporary concession for diesel cars over 2.0 tonnes laden weight which are off road or >6 seats (unladen weight from 1206 to 1660 kg). Concession ceases on 31/12/02.

[3] Temporary concession for diesel cars over 2.0 tonnes laden weight which are off road or >6 seats (unladen weight over 1660 kg). Concession ceases on 31/12/02.

sub-optimal performance for a majority of the batch of engines. He also concluded that, through the life of the engine, the map must be altered to remain optimal. Turner's view was that in an ideal world, closed-loop control of the combustion process would be based on measurements of some meaningful property of the combustion process (as discussed below). However, he stressed the problem that current combustion measurement devices did not meet the requirements of volume production.

1.3 Combustion sensor requirements

Several techniques to measure combustion process parameters have been documented; these include: ionisation sensing [5], optical sensing [6], pressure sensors [7], strain gauge sensors [8] and instantaneous angular speed measurement [9]. The intrusive measurement methods [5],[6],[7] currently require costly sensors and/or extensive modifications to the engine, although lower cost versions are being developed [8]. Also, modern multi-valve engines leave little room in the cylinder head to place an intrusive sensor. Even if a position is found which avoids valves and injectors, the body of the sensor usually protrudes through water jackets, oil ways and inlet/exhaust routes. The cost of adding the sensors and sealing them from the ingress of water, oil and combustion gases is prohibitive on production engines, as is the cost of re-designing the cylinder head with an in-built sensor. If a position can be found which satisfies the location and sealing requirements, then the sensor can be added. However, as the sensor is placed flush with the surface of the combustion chamber it must be designed to withstand high temperatures and pressures. This adds to the cost of the sensor, and reduces its working life.

In contrast, the non-intrusive sensor methods [8], [9] benefit from reduced temperatures and a less harsh environment. However, non-intrusive sensors do not record the combustion process directly, only its effects. Thus the data produced by a non-intrusive measurement method often require some signal processing before they are usable. Various techniques have been employed to try and infer in-cylinder pressure from the outside of the engine. Of these Kirkman's research [10] established that the stretch of the bolts on a single cylinder engine correlated well to cylinder pressure.

1.4 Signal processing strategy

Having established that non-intrusive methods for measuring combustion process provide a method for closed-loop engine control on volume production IC engines, the next stage of the research was to develop a strategy for the required processing of the signals.

The initial stage involved an investigation of similar digital signal processing (DSP) problems. The extraction of a signal buried in noise has been the subject of many documents, and several techniques exist. These standard techniques were applied but ultimately proved unsuccessful (see Chapter 3.6). The value of neural networks [11], [12], [13] and [14] in providing solutions was then investigated. This revealed work by Websper [15] and Chen [16] which showed that the approach of using a single source of data was not immediately applicable.

Accordingly, another method of analysis was examined, namely wavelet decomposition, and this proved to be successful in pointing a way forward (see Chapter 5.3). Wavelets were developed by Haar [17] in 1910 as a mathematical concept. Global bases, such as sines and cosines, are used in Fourier decomposition to approximate a global function from $-\infty$ to $+\infty$. However, these functions do not have compact support[4] and as such cannot isolate in time. Wavelets rely on the use of local bases, which have compact support, to describe a global function. The Haar wavelet, which is a pulse, gives the property of isolation in time.

In the research conducted for this thesis, raw signals were processed by wavelet decomposition, and the Sammon technique [18] was applied to the derived signals.

[4] Functions which have compact support are finite in length. They are zero outside their window of operation. Sines and cosines are infinite in length and have an infinite window of operation.

The results indicated that neural networks could be used on the derived signals to detect relevant combustion features (see Chapters 6 and 7).

1.5 Stages of the research

The research undertaken for this project progressed through the following stages. At Stage 1, a review was undertaken of the past and present technologies which provided guidance on the development work to be undertaken. At Stage 2, the sensors for capturing the relevant signals were designed. At Stage 3, an investigation was conducted into the suitability of standard techniques for analysing the raw sensor data. Because these techniques proved to be unsatisfactory, Stage 4 was concerned with the concepts of wavelets and their application in engine technology. At Stage 5, wavelet decomposition was applied to the raw sensor data and the application of wavelets to the extraction of features was detailed. At Stage 6, methods of feature detection were investigated and two approaches were employed: wavelet reconstruction and artificial neural networks, and some final results were provided. The last stage was to draw conclusions from the work and outline areas for future research.

1.6 Discussion

Volume production cars are being required to meet increasingly stringent conditions relating to the toxic conditions produced by the combustion of fossil fuels. A method of improving fuel efficiency (thereby reducing emissions) is by an engine control system based on measurements of the combustion process. Two key measurements of this process are start of combustion and peak pressure.

There are already closed-loop engine control systems in current use, but because they employ intrusive sensors to generate the required information, they are too costly to be used on volume production vehicles. Therefore, this research was concerned with the development of a sensor and associated signal processing system to enable the less costly non-intrusive measurement of the combustion process to take place.

After reviewing previous investigations in the field of non- intrusive pressure measurement, it was determined that head bolt measurement provided a suitable technique for obtaining signals, and was also one which matched the facilities available for the research[5]. Reviews were also undertaken of the strategies which could be used to process signals from non-intrusive sensors. Here, many problems had to be overcome before a system was finally developed. This resulted in the production of a high grade signal indicating the start of combustion and the timing of peak pressure – thus providing a useful measure of the combustion process which could be employed in an engine control system for volume production vehicles.

[5] This project was undertaken using facilities provided by the University of Bath, including access to their research engines.

2 REVIEW OF CYLINDER PRESSURE MEASUREMENT AND DIGITAL SIGNAL PROCESSING TECHNIQUES

2.1 Introduction

Preliminary investigations (as outlined in Chapter 1) identified that closed-loop engine control based on cylinder pressure is a major contributor to cleaner IC engines. Therefore, the project concentrated on the development of the sensors and signal processing techniques to enable closed-loop engine control technology to be utilised on volume production vehicles. A review of past and current research papers was undertaken in order to identify those techniques and areas of work which would serve as a guide in the development of the systems documented in this thesis. The outcome of this review has been categorised into three main areas: closed-loop engine control; sensors for pressure based closed-loop engine control; and techniques for signal processing. These are summarised below.

2.2 Closed-loop engine control

It was noted by Turner [3] that in an ideal world, closed-loop control of the combustion process would be based on measurements of some meaningful property of that combustion process. He stated that cylinder pressure is the fundamental variable of the combustion process which can be measured to determine the operating state of the engine. Cylinder pressure measurements provide real time feedback on the combustion process, and can be used for cylinder by cylinder control of engine variables to improve performance. His review of closed-loop engine control systems established that in-cylinder pressure measurement is an invaluable tool for modern control systems, and several key technologies were identified.

An article which detailed the use of complete cylinder pressure as the control variable was written by Muller [7], in a report into an artificial neural network based adaptive control system. His paper showed a methodology to control the ignition timing in a feedback loop based on the combustion pressure. This pressure was measured by a piezoelectric sensor, built into the spark plug. The controller was incorporated into an electronic control unit in a vehicle, where it showed good results in extensive tests, and proved that this type of sensor could be incorporated into production engines, given the caveat in Chapter 1, Turner [3]. This article demonstrated the importance of cylinder pressure as the primary variable for control of an engine.

The literature review showed that characteristics of cylinder pressure, such as start of combustion and location of peak pressure, could be used as the main variables in a closed-loop control system. The benefit of moving to a characteristics based closed-loop system, rather than complete cylinder pressure, is that it enables the system developer to use a straight forward approach, based on a physical model. Additionally, the start of combustion and location of peak pressure detection enables the electronic control unit (ECU) to correct for changes in ignition delays, such as fuel cetane number and changes in air, fuel and engine temperatures.

In a system developed by Glavmo [5], ionisation technology was utilised which measured the quantity of free electrons in the combustion chamber. It was found that chemical ionisation occurred during the exothermic reaction, and that this could be measured by an ionisation sensor. In the report, a control system based on the start of combustion was detailed, and tests on a 2.0 litre common rail diesel engine confirmed the ability of the sensor to successfully monitor start of combustion. This paper presented a novel approach to the detection of start of combustion and it has been successfully demonstrated. However, the author identified a potential problem over the long term contamination of the sensor by soot. This would require some maintenance to replace or clean the device at normal service intervals.

Geiser's research [6] demonstrated that the optical characteristics of the combustion process could also be measured. These measurements gave a correlation for light intensity and pressure and provided an accurate indication of the state of the combustion. He detailed the use of an optical fibre technique to record light intensity from a spark ignition (SI) engine. In the paper, a spark plug was modified to incorporate eight sapphire windows with optical fibre pickups. The fibres channelled the light from the sapphire windows to an array of photo-multipliers and a data acquisition system. From the results obtained there appeared to be a good correlation between luminosity and in-cylinder pressure.

Wang [19] reported on a system to control the ignition timing of an SI engine based entirely on cylinder pressure. An intrusive sensor recorded in-cylinder pressure, and the location of peak pressure was extracted by a signal processing system. The processing system used a fuzzy logic control technique, which showed the capability of automatically adjusting the ignition timing in each individual compression stroke without making reference to the engine speed and load.

Gu [9] used instantaneous angular speed of the crankshaft to estimate cylinder pressure. The work detailed the use of a radial basis function neural network as an estimator, and the results produced indicated that this type of network was effective in the estimation of cylinder pressure. The only input to the network was the instantaneous angular speed, thus showing that in certain cases direct measurement variables can be used as the input features for a network. In the conclusions of the work two weakness were highlighted. The first was in using direct measurements as the input features, which resulted in complex networks. These required extensive training data and produced slower recall times. The second weakness was in the dynamic behaviour of the engine under load conditions. This introduced twisting of the crankshaft and caused the geometry of the engine to be variant. Thus the current model could not be applied to an engine operating under such conditions of acceleration and deceleration.

These research papers highlighted ways of measuring in-cylinder pressure by using the combustion process variables, either directly or indirectly. This led to a review (in the following section) of the implications of these methods for sensor design.

2.3 Sensors for pressure based closed-loop engine control

The sensors used for closed-loop engine control, discussed above, were designed to record parameters of the combustion process. The control systems used these parameters, such as location of peak pressure and start of combustion, as the basis for their settings. The different parameters recorded ranged from direct measurement of cylinder pressure to indirect pressure measurement using instantaneous angular speed. In the case of indirect pressure measurement, signal processing was needed to extract the relevant features for the control system. However, direct pressure measurement did not need such extensive signal processing. The research findings on sensors used for closed-loop engine control were reviewed, and details are given below, first for those concerned with direct pressure measurement and secondly for those concerned with indirect measurement techniques. Some of these have been highlighted in the previous section and are included here for completeness.

2.3.1 Direct pressure measurement

Muller [20] designed a sensor for in-cylinder pressure measurement. The sensor was based upon an in-cylinder pressure measurement device, but included a ceramic component for reducing the heat signature seen by the active parts of the device. The device was constructed from a threaded tube, with one end sealed by a diaphragm. The tube was screwed flush with the inside of the cylinder head, and the diaphragm deflected with the change in pressure. The diaphragm was connected to the active recording device by a rod of ceramic material. The ceramic rod did not conduct the heat from the diaphragm to the active device, which could therefore be situated in the airflow above the (relatively) cool cylinder head. This report showed that the sensor performed well in detecting cylinder pressure and there was no measurable degradation in performance over extended periods. However the cost remained prohibitive for mass production vehicles.

Lee [21] reported on an innovative sensor, designed for automotive applications. The sensor used an interferometer to measure pressure-induced deflection of a membrane. As in the design by Muller [20], the membrane was sited at the end of a threaded tube, which was placed flush with the inside of the cylinder head wall. The design also allowed the siting of the active part of the sensor to be placed some distance from the cylinder head wall. This increased the durability and reduced the cost of the sensor. However it still suffered from high manufacturing cost.

The designs of Muller and Lee attempted to remove the active components of the sensors from the cylinder head wall, where they would be subjected to full combustion temperature. They were successful in achieving this. However, the designs still contained an expensive sensing element with precision machined parts, which added to the cost. For mass production, the cost must be significantly reduced.

2.3.2 Accelerometers

Papers were reviewed on the use of accelerometers to detect engine vibration signatures, from which information on the combustion process could be extracted. These papers (see below) showed that peak pressure and other engine parameters could be successfully derived from engine vibration.

DeJong [22] detailed the operation of a 350 horsepower diesel truck engine, which was monitored during laboratory tests using vibration of the block to detect faults in the cylinder pressure and bearings. Faults were introduced into the engine to determine the sensitivity of the vibration measurement in locating a particular fault. It was found that accelerometers placed on the cylinder head bolt were the most sensitive to cylinder pressure changes. This paper highlighted the use of vibration to determine the operating state of the engine. The work showed that it was possible to detect changes in operating state and report faults when they occurred. It also showed that it was possible to determine the location of peak pressure and other combustion parameters from the signals.

Ordubadi [23] conducted work into obtaining cylinder pressure information from the excitation of the cylinder block. The vibration transfer-function was developed and pressure information reproduced. However, in the conclusions of the paper, it was noted that the signals suffered badly from degradation. This was determined to be caused by the excitation of the block from all cylinders, structural resonance within the block and contamination by other noise sources.

Glibbery [24] investigated the onset of engine knock by monitoring engine vibration and summarised much of the earlier work in this field.

2.3.3 Stress and strain sensors

The focus of many research papers was on the use of stress and strain sensors to measure changes in the external structure of the engine. These structural changes in the engine occur as a result of the combustion process. The work reviewed here indicated that measurement of the structural changes correlated well to the combustion process, and had a sufficient signal to noise ratio to be used as signals in cylinder pressure based closed-loop control systems.

Kirkman [10] conducted research into strain gauges placed under single cylinder head bolts of a large diesel engine. The work centred around measuring the compression of a washer placed under the cylinder head bolts. Kirkman showed that the compression of the washer was directly proportional to the load placed on the bolts from the combustion process. In his work, a number of sensors were positioned around the cylinder head and compared with in-cylinder pressure. Initially the work concentrated on fitting a piezoelectric load washer under a cylinder head bolt. However, the clamping arrangement of the head used highly torqued bolts, which precluded the use of commercial piezoelectric load washers.

As a compromise, a small strain gauge washer was made and placed under one of the cylinder head bolts. This was used to provide information on the variation of the clamping forces exerted by the combustion process. The conclusions indicated that, for engines large enough to warrant individual cylinder heads, or large single cylinder engines, cylinder pressure correlated well with the measured bolt strain. This research showed a direct correlation between cylinder pressure and the clamping forces on the cylinder head bolt, with a high degree of accuracy. The problem identified by Kirkman was to miniaturise the sensors for use on multi-cylinder engines.

Miyamoto [25] determined that by measuring the strain of the cylinder head bolts, the horizontal displacement of the crank shaft end, and the vertical displacement of the intake valve stem, cylinder pressure could be reproduced. Pressure diagrams were estimated by applying the pressure-strain diagram obtained from the static pressure test in the cylinder to the strain variation in the cylinder head bolts. This gave a correlation between the static pressure in the cylinder head and the strain on the head bolts. Using this correlation, accurate pressure diagrams were obtained. This work reached similar conclusions to those of Kirkman by showing that the measurement of cylinder head bolt clamping forces was directly related to in-cylinder pressure.

Sellnau [8] successfully showed that non-intrusive sensors could be designed to give accurate recordings of the internal combustion process. In his paper, the detailed design of a non-intrusive sensor based on piezoelectric ceramic was shown. The sensor consisted of a ring of piezoelectric ceramic contained within an outer threaded housing and an inner sleeve. The device was designed to measure compressive forces exerted on the outer sleeve. The sensor was screwed into a threaded boss located in the spark-plug or injector well. The sensor responded to structural changes in the cylinder head caused by the combustion process. In the research, structural analysis was performed on the cylinder head and it was determined that maximum structural displacement occurred at the cylinder head centreline. The operation of the sensor worked on the structural load path formed from the cylinder pressure forces being transferred through the cylinder head centreline to the head bolts. It showed a series of results, which demonstrated a high degree of correlation between the sensor and an in-cylinder pressure transducer. This paper concluded that a non-intrusive sensor, based on piezoelectric ceramic was highly effective in inferring cylinder pressure.

These papers showed that the direct measurement of in-cylinder pressure could be successfully achieved, but that the sensing elements of the process were costly to produce and/or maintain. They also showed that indirect pressure measurements could be obtained and that a non-intrusive sensor could be designed to capture the relevant signals. Accordingly, a review was undertaken of signal processing methods (see 2.4).

2.4 Signal processing

Papers were reviewed on the use of signal processing methods in automotive applications. Methods employed in similar applications were also reviewed for their bearing on the automotive field. In these reviews, two methods emerged as important techniques for the analysis of non-stationary signals produced by IC engines, namely artificial neural networks and wavelets. The research papers demonstrated that neural networks could be used to detect complex patterns in signals, which described an underlying process, and also offered a statistical method of classifying signals which do not have accurate transfer functions.
The wavelets technique was employed for its ability to isolate signals in frequency and time, an ability which was successfully used to remove noise and extract features buried in signals. Further details on the use of these two techniques are given below.

2.4.1 Neural networks

Neural networks are known to be a valuable tool for their ability to approximate non-linear mappings. A mapping is the ability to convert from an input vector space to an output vector space. This has proved useful in a number of areas related to IC engines, and the following papers explored this property.

Shayler [26] described the use of neural networks in two applications traditionally found in ECUs. The first application used a network as a pattern classifier to assist with service-bay engine-fault diagnostics. The overall aim was to produce a system that could distinguish between intermittent and persistent ignition and fuelling faults, and indeed also identify a 'normal' engine operating correctly. The signals used for diagnosis were based on instantaneous crankshaft speed, a measurement which Gu [9] had shown to be effective in determining the operating state of the engine. The results presented in Shayler's paper showed that the network was able to detect 99% of healthy engines and 100% of engines with a persistent misfire. The second application was a steady-state controller, which used a neural network for its non-linear mapping ability. In steady-state, part-load operation, four input signals (air charge temperature, air mass flow rate, engine speed, and normalised air charge), and two output signals (fuel injector pulse width and spark timing) were used. The network was trained on data sets generated from a test engine, and approximated the relationship between input and output space. The paper presented results which showed an 11.8% error in the spark timing and a 1.5% error in the fuel injector pulse width, between conventional and neural network controllers. With these two applications, Shayler demonstrated the ability of neural networks to form complex relationships between input and output vector spaces.

Leonhardt [27] described two methods for the closed-loop control of IC engines. Both methods used the mapping ability of a network to derive the complex relationship between the input and output vector spaces. The first method used parameters derived from an in-cylinder pressure sensor as the input vector for the network. The derived parameters included location of peak pressure and the difference in pressure between motored and firing cycles. The second method used instantaneous angular speed which, as confirmed by Gu [9], contains engine combustion characteristics

Kirkman [28] also used a neural network to monitor the health of a diesel engine. A multi-layer perceptron (MLP) neural network was trained with healthy in-cylinder pressure data. Once a suitably accurate model of the 'healthy' engine was generated, this was used as the reference for fault detection. Cylinder pressures, measured while the engine was running with seeded faults, were compared with the generated 'healthy' model and the differences used to categorise the faults. The results obtained showed that it was possible to create a suitably accurate model of the cylinder pressure data for the 'healthy' condition. In thirteen out of the fifteen seeded faults introduced, the measured data from the load cell washer was sufficiently different from the 'healthy' data to allow the fault to be detected and categorised.

Brace [29] described the development of an integrated control strategy for a passenger vehicle powertrain. He detailed the work undertaken to design and validate a series of engine models for use in the powertrain control project. Various methods of modelling a diesel engine were reviewed, and the chosen approach was to use a neural network to represent the in-cylinder events and predict emissions.

Scaife [30] trained a neural network to detect the onset of faults on a diesel engine by observing seventeen engine parameters and comparing them with previous tests, where known faults had been introduced.

Finally, a paper by Websper [15] on the application of neural networks in the field of electrical transmission systems was reviewed. This was included to highlight an important aspect of neural networks – input feature selection – and to indicate the rigorous mathematical approach he took to the problem of this selection. Websper used neural networks in the solution of adaptive distance protection. This problem was concerned with lightning strikes and other faults on the power lines of the national grid. The work used neural networks to sense the change in voltage and current and to calculate the distance of the fault. Once this had been established, the circuit breaker associated with that section of line could be de-activated until the fault dissipated. The work illustrated how the selection of the input features to the neural network was one of the most important factors in successfully detecting the faults. He showed several techniques for choosing the input features, and concluded that the Sammon algorithm [18] performed best. This algorithm was then used to analyse the suitability of signals as input features to the neural network.

2.4.2 Wavelets

Websper [15] identified that the correct selection of the input features is an important factor in the development of a neural network. He established that the primary variables may not provide the optimum selection of input features, and that derived variables can be more suitable. A review was undertaken of papers where techniques for extracting features from the primary variable set were identified. From these, wavelets [31] emerged as a powerful tool for removing noise and extracting secondary features. It was shown that a combination of primary and wavelet derived secondary features enabled networks to converge faster and with lower error rates than primary signals alone.

Chen [16] reported on the application of wavelets in the extraction of features for a neural network. A network was used in the classification of blemishes in manufacturing systems. The report noted that the blemish appeared as a two dimensional structure, which had variance in two dimensions. These variances were removed by using a wavelet transform and the extracted features were applied to a network. It was then shown that a network performed better at classifying blemishes if the derived signals were used as the input features, rather than the primary signals.

Yao [32] investigated the acoustic energy of cutting tools to determine tool wear. In this research, the wavelet packet transform was used to extract important features from the acoustic energy signal that were sensitive to the changes of tool state. A network was then used as a classifier to develop a number of outputs which indicated tool wear state. This work again showed the importance of input feature conditioning to enable an efficient implementation of the network.

The final paper reviewed here was that by Bakhtazad [33]. This detailed work in de-noising signals using wavelets - an important area for applications in the automotive field since many real world signals from IC engines are contaminated by noise. Bakhtazad examined various methods for removing noise from four reference signals. The method which proved most effective was wavelet thresholding. By using this technique, the mean standard error for a signal was significantly reduced over traditional methods.

For further reading in the area of wavelets the reader is directed to the reference work by Daubechies [34] and [35], Mallat [36], [37], [38], [39] and [40], Kovacevic [41], Rioul [42] and Lang [43].

2.5 Discussion

A review of research papers was undertaken to identify the techniques and methods applicable to the closed-loop control of IC engines based on cylinder pressure – in particular those relating to the detection of the combustion parameters required for closed-loop control. This review established that the parameters of start of combustion and location of peak pressure could be used for closed-loop engine control.

Having identified which signals to measure and their relevance to closed-loop engine control, techniques for detecting these signals were reviewed. It was established that intrusive sensors, although accurate, could not be used in volume production vehicles, due to cost. Several techniques for the indirect measurement of the combustion process were then reviewed, and it was shown that many of these were successful. The work by Kirkman was particularly relevant since it matched the preliminary investigations on non-intrusive sensors undertaken for this thesis (see Chapter 3). Therefore, subsequent research in this area drew on his findings.

Several papers reviewed showed that artificial neural networks have a proven track record in pattern classification and signal approximation. Based upon these reviews, artificial neural networks were used as the classification stage in the development of the non-intrusive sensors detailed in this thesis.
Wavelets were investigated as a method of signal conditioning. Several papers showed that wavelets were effective at noise reduction and feature extraction, and based upon this, they were used in this research as the conditioning system for the network classifier.

In summary, this Chapter justified, by example, the methodology used this thesis for the development of the sensors and signal processing strategy which enable pressure-based closed-loop engine control to be used on production vehicles. It established that non-intrusive sensors could be effective in providing detailed analysis of the combustion process by using a wavelet-neural network approach to signal approximation and classification.

3 NON-INTRUSIVE SENSORS

3.1 Introduction

In Chapter 1, it was established that an effective solution to the problem of IC engines meeting future emissions legislation could be provided by cylinder-pressure based control systems. In Chapter 2, the literature reviews indicated that non-intrusive measurement of cylinder pressure could enable a low-cost method of deploying these control systems to be developed. Various types of non-intrusive sensing systems for the measurement of combustion parameters were reviewed from which it was concluded that sensing the load variations experienced by the head bolts would be an effective method for inferring cylinder pressure. From the literature survey into closed-loop engine control based on cylinder pressure, two parameters of cylinder pressure were identified as being fundamental to the control strategy: start of combustion and location of peak pressure.

The next stage of the research (reported below) was concerned with the design of sensors which could be used to detect these two fundamental variables. The first step was to develop a miniature load washer to detect start of combustion and location of peak pressure, for single cylinder engines, drawing on the work conducted by Kirkman [10]. The next step was to undertake multi-cylinder work as outlined in Kirkman's conclusions. The final step in this stage was to demonstrate, by the use of traditional filtering techniques, that the detection of the start of combustion and location of peak pressure is non-trivial. Some initial results were obtained, as indicated in the sections below.

3.2 Background

Piezoelectricity is a property of certain classes of polycrystalline materials. When mechanical pressure is applied to one of these materials, the crystalline structure produces an electrical charge. Conversely, when an electric charge is applied, the crystalline structure changes shape, producing dimensional changes in the material.

Piezoelectric properties occur naturally in some polycrystalline materials and can be induced in others. Many contemporary applications of piezoelectricity use polycrystalline ceramics instead of natural piezoelectric crystals. These piezoelectric ceramics are more versatile as their properties can be tailored to a specific use. Kirkman's initial approach [10] was to employ commercially available load cells which used piezoelectric ceramic elements in their construction. His theory was that a piezoelectric washer placed under the head bolt would be subjected to the load variations caused by the change in cylinder pressure, and this would be reproduced as a change in charge.

When a piezoelectric element is subjected to an external force it generates a charge. This can be thought of as a current source in series with a capacitance, and so standard charge amplifier techniques can be used to amplify this signal. Equation 1 [44] shows this relationship between device size and capacitance

$$C = \frac{\in_r \in_0 \Pi \left(od^2 - id^2\right)}{4 * h}$$

$\in_r$ = Relative permittivity – 1700 at 25 °C for PZT 5A.

$\in_0$ = Permittivity of free space - 8.85×10^{-12} farad / metre.

od = outside dimension of washer.

id = inside dimension of washer.

h = thickness.

Equation 1

3.3 Design of the Mk1 sensor

The reproduction of Kirkman's results guided the design of the Mk 1 sensor and enabled it to be placed under the head bolt of a single cylinder engine. The engine made available for this initial study was a single cylinder Robin diesel engine of $412cm^3$. The engine was an air-cooled unit with exposed cylinder head studs. This gave excellent access to enable the installation of the Mk1 sensor. Figure 1 shows the construction of the Mk1 sensor. The construction of the sensor was based on a washer of piezoelectric ceramic material with a diameter of 25 millimetres and a thickness of 2 millimetres. The mounting hole had a diameter of 15 millimetres. The material had electrical connections on either face and contact washers were placed either side. The contact washers were made from printed circuit board, type FR4. This had a copper deposit on a glass fibre base and enabled solder connections to be made. Steel load washers completed the 'washer stack'.

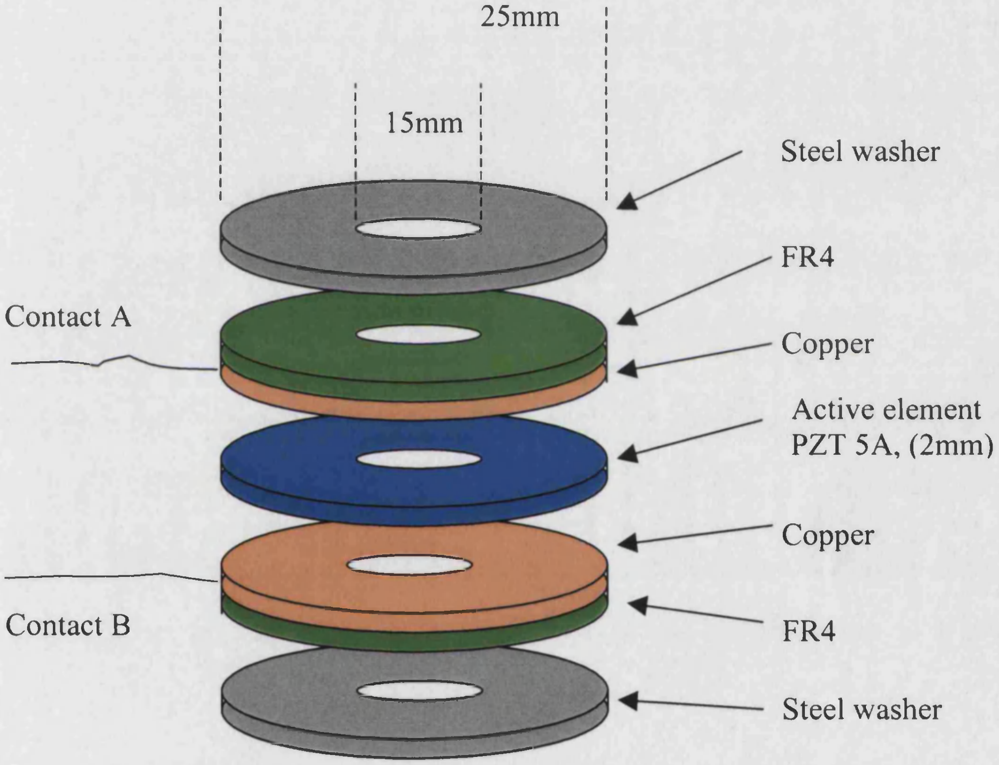


Figure 1 Mk1 sensor construction

The piezoelectric device chosen was Lead Zirconium Titinate (PZT 5A). Table 3 details the range of materials available from Morgan Matroc Ltd. [44]. Navy type is taken from DOD STD 1376A (Ships). Red text indicates a high power 'hard' material. Blue text indicates a high sensitivity 'soft' material. Green indicates custom material. The dielectric constant of the material gives a guide to the sensitivity of the ceramic, (see Figure 2). PZT 5A has a relative permittivity of 1700; this places it close to the middle of the sensitivity range. More sensitive materials were available, but the temperature range of the ceramics affects the performance. The Curie point of the ceramic is the temperature at which it permanently loses its properties. However, as the temperature rises, the long-term stability of the ceramic is reduced, so the ideal is a high Curie point and a low operation temperature. The anticipated operating temperature of the sensor was 100 to 150°C. Therefore PZT 5A was chosen as offering the highest Curie point at 365 °C, with mid-range sensitivity.

The piezoelectric element can be thought of as a capacitance in series with the signal. So to carry out tests using this sensor, a charge amplifier was designed. Using with the dimensions of the washer given in paragraph 3.3 and the dielectric constant given in Table 3, a capacitance of 118 nF was obtained (see Equation 2).

$$Cw = \frac{1700 * 8.85 * 10^{-12} * \Pi * 0.01}{0.004}$$

$$Cw = 118 \; nF$$

Equation 2

The charge amplifier developed is shown in Figure 3. The calculation for the gain of the first stage was based on the reactance of the source, the response time and the required level of amplification. The signal obtained from the sensor was measured using a high impedance oscilloscope probe (10MΩ) at a maximum of 35v. In order to be compatible with the data acquisition system, the output voltage was required to be under 5v, so an overall ÷ 10 was needed.

PZT type	Navy Type	Uses	Curie Temperature (Celsius)	Dielectric Constant ($\in_r$)
4D	I	Sonar & ultrasonic cleaning	320	1350
8	III	Highest power, suitable for ultrasonic welding, sonar & high power probes.	300	1000
5A	II	Hydroplanes, accelerometers & vibration sensors	365	1700
5J	V	High energy & high output voltage e.g. hydroplanes & fuses.	250	2600
5H	VI	Sensitive receivers & fine movement control	195	3400
5R		High acoustic sensitivity – sonar towed arrays.	350	1950
PT 2		High sensitivity – custom products	255	200

Table 3 Piezoelectric ceramic types

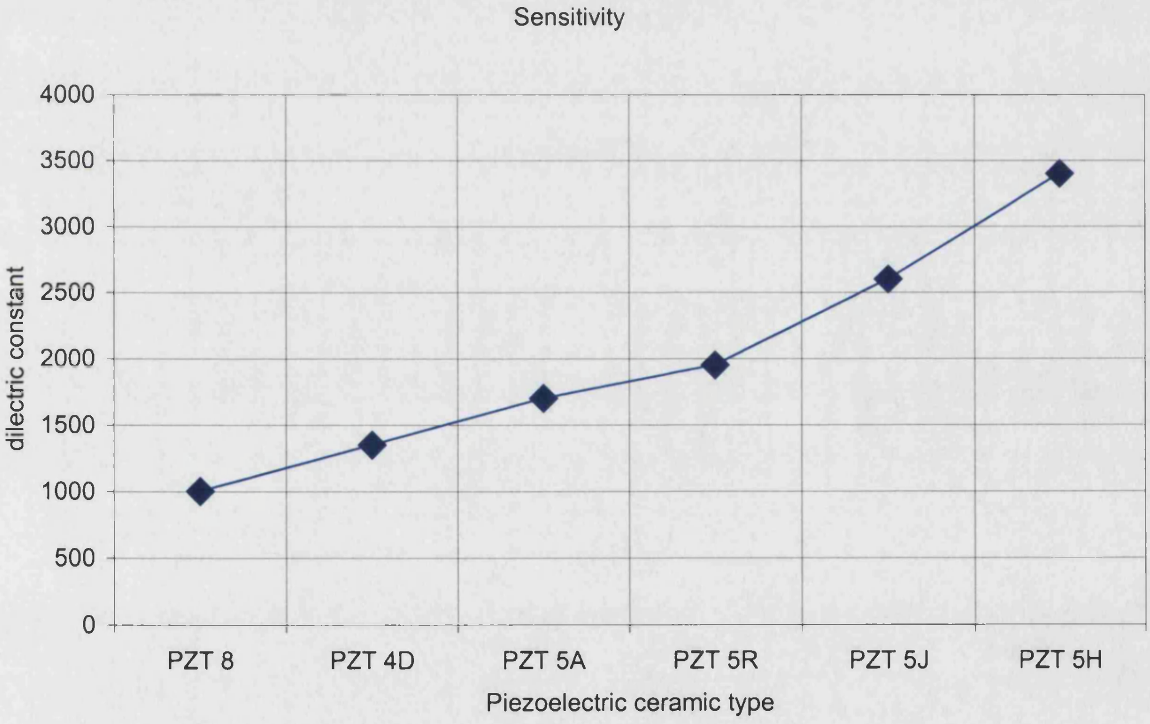


Figure 2 Sensitivity of PZT ceramic

Due to non-availability of high accuracy very small value capacitors, the first stage was limited to ÷100. The second stage then provided x10.

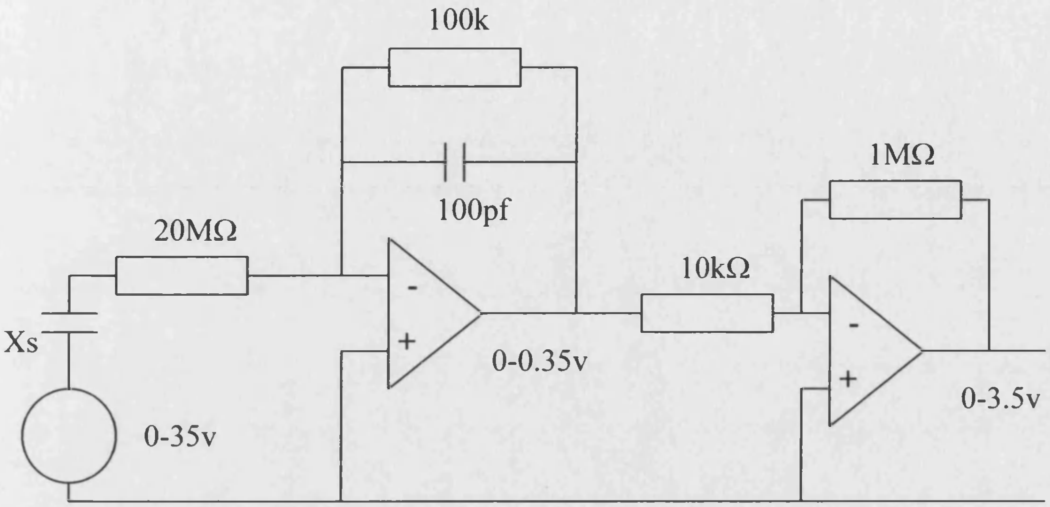


Figure 3 Charge amplifier for non-intrusive sensor

The Mk1 sensor was placed under one of the head studs and clamped in place with a torque of 40NM. Figure 4 shows a comparison of in-cylinder pressure measured by a Kistler 6123 transducer and the Mk1 sensor. By visual inspection, a strong correlation in the compression phase could be seen between the two traces. This enabled the start of combustion and location of peak pressure to be detected. This corresponded with similar results obtained by Kirkman [10].

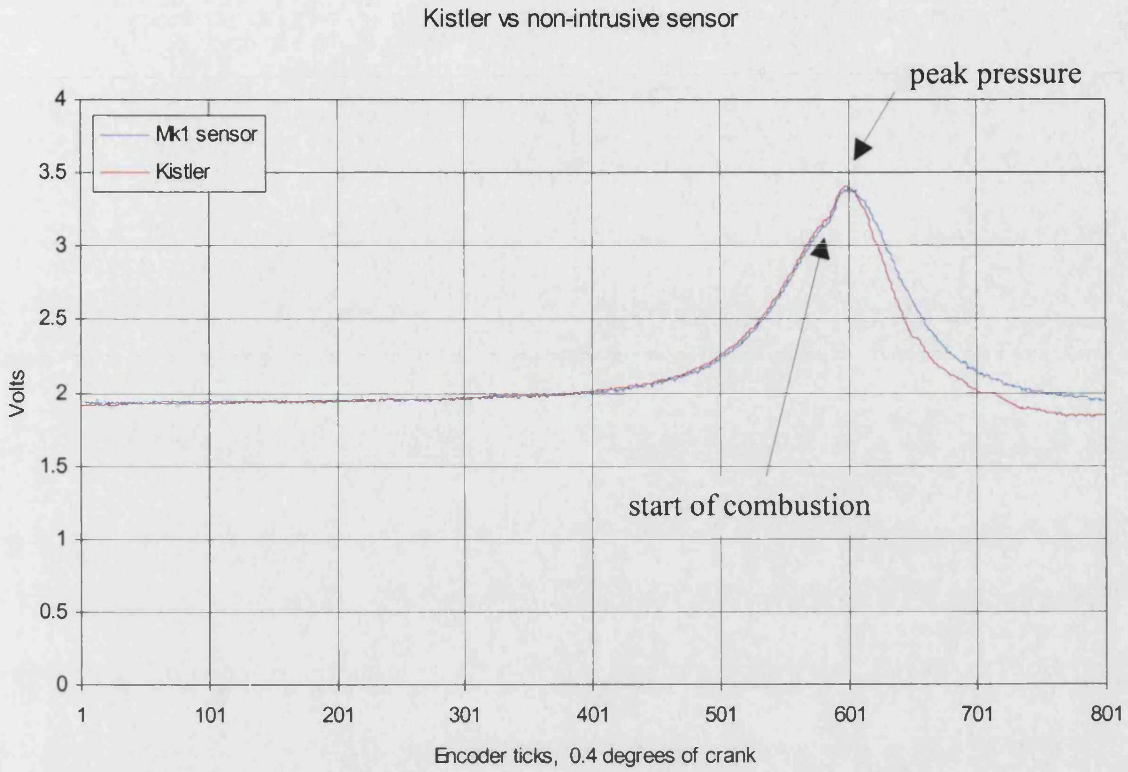


Figure 4 Kistler and Mk1 sensor

3.4 Design of the Mk2 sensor

In the conclusions drawn by Kirkman, multi-cylinder engine work could be possible through the use of miniaturised sensors. Thus the next stage of the project entailed the migration of the sensor to a multi-cylinder engine. The engine made available for this research at the University of Bath was a Ford, 4 cylinder diesel of 1800cm^3 capacity. This was appropriate as being representative of the engines used in medium sized family automobiles, to which emissions legislation and economy are increasingly being applied.

The Mk1 sensor used a washer of PZT as the active element. This gave good results as the variation in bolt compression acted directly on the whole of the PZT. It was realised that a similar arrangement could not be used on the multi-cylinder engine, as a clamping torque of 180NM was required to secure the head bolts. An experiment confirmed that 180NM was sufficient to reduce the PZT washer to powder. Therefore, a new arrangement was required which could withstand this clamping torque. A load sharing device which could take the majority of the torque was designed. This had the main advantage of retaining clamping torque over a period of time, even without the presence of the active element. This was a critical factor in the design of the device for the fitting on the engine. Figure 5 shows the arrangement of the Mk2 sensor and Figure 6 shows a photograph.

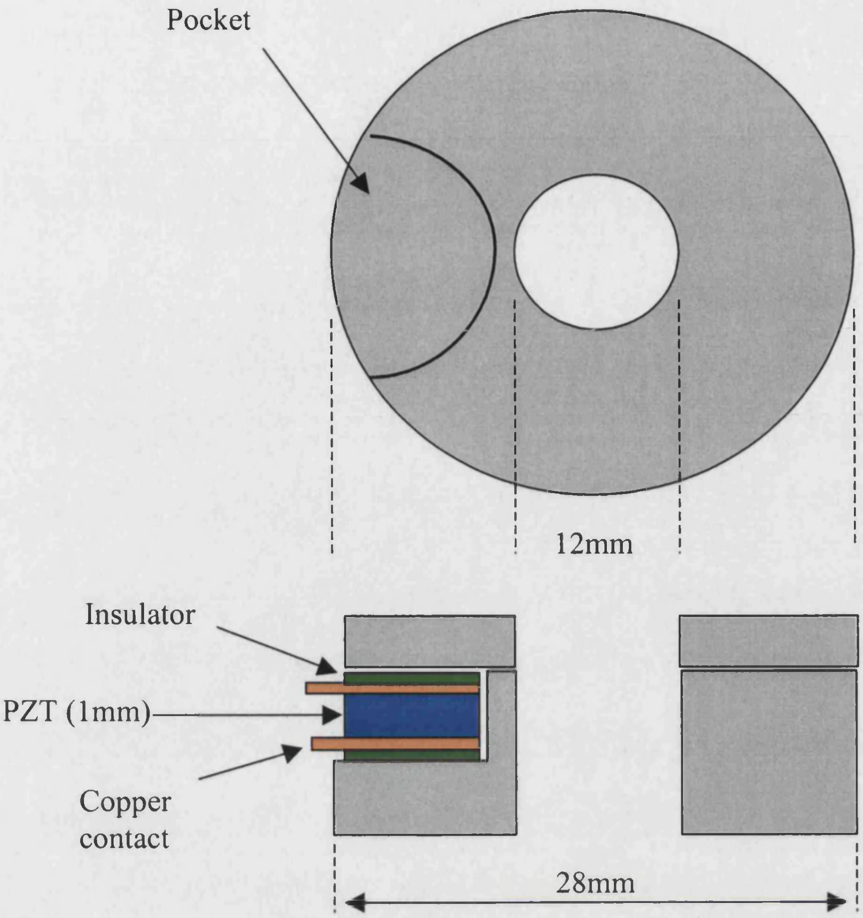


Figure 5 Arrangement of the Mk2 sensor

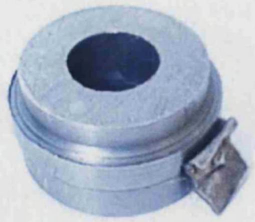


Figure 6 Photograph of Mk2 sensor

The active element was a small device captured in a machined pocket within the structure. When the clamping torque was applied, the majority was passed through the structure and a fraction was applied to the PZT. The PZT now measured the variation in clamping loads but experienced a reduced peak torque.

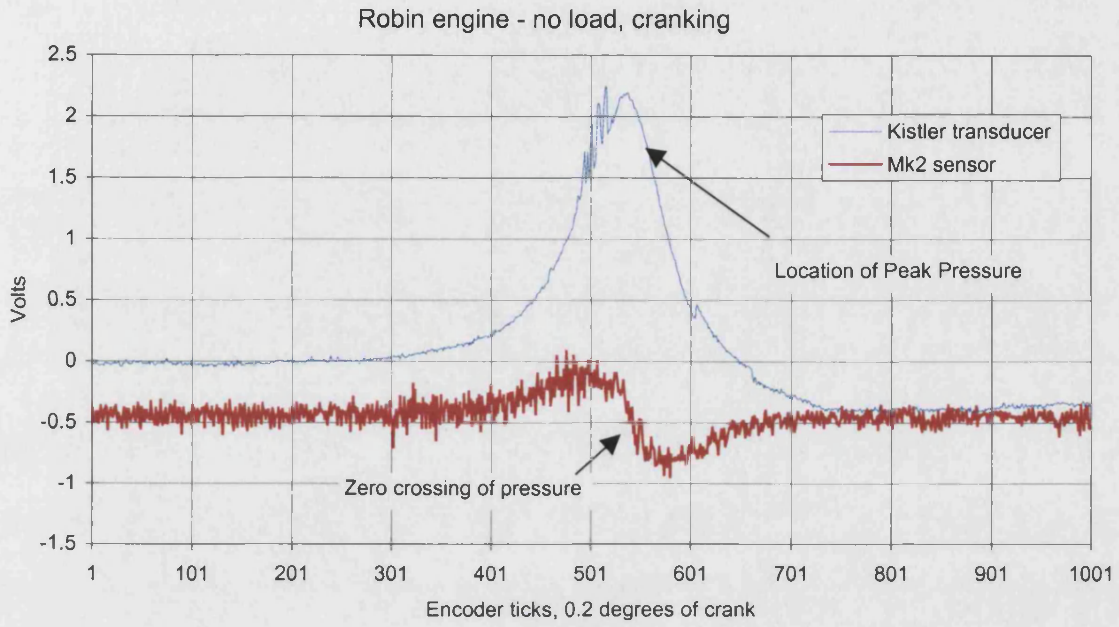


Figure 7 Mk2 sensor and Kistler transducer during cranking

The Mk1 sensor was removed from the single cylinder Robin diesel engine and replaced by the Mk2, and the series of engine tests was re-run. The signals from the Mk2 sensor were disappointing in that they did not correlate well to the in-cylinder Kistler transducer. Figure 7 shows the Mk2 sensor on the Robin engine during a cranking cycle and Figure 8 shows the Mk2 sensor at 1200 rpm. However, the information contained within the waveform showed start of combustion and location of peak pressure, albeit contaminated with noise.

It was interesting to note that the Mk1 sensor reproduced a close approximation to in-cylinder pressure but the Mk2 sensor produced the first derivative. However, the sensor enabled the detection of start of combustion and location of peak pressure. The detection of these two variables was the primary aim of the sensor design.

Thus as the Mk2 sensor was deemed a success no further investigation into the construction of the sensors was undertaken.

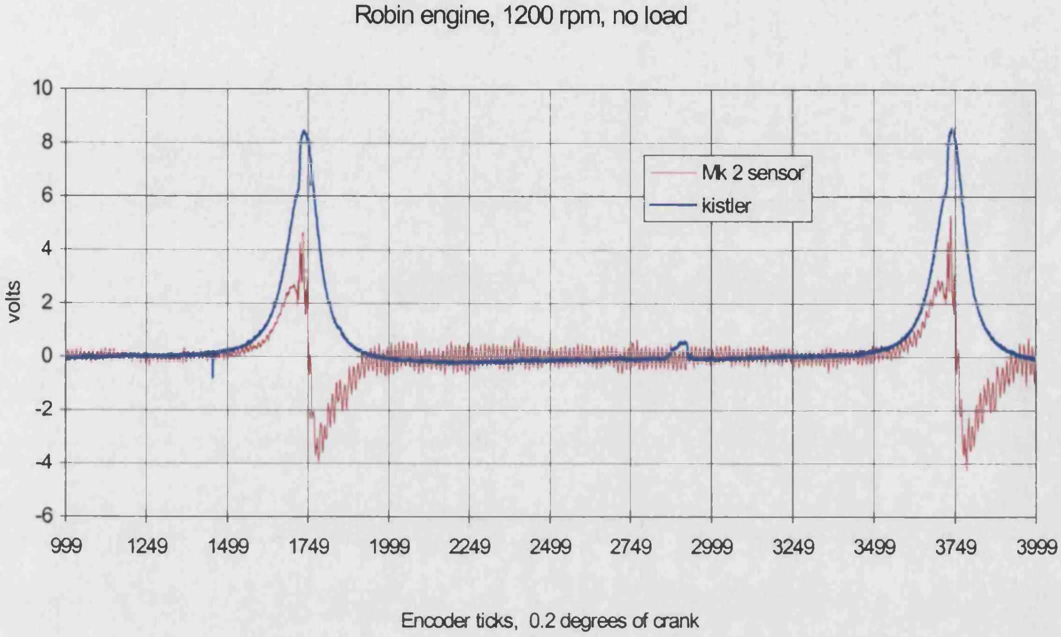


Figure 8 Mk2 sensor and Kistler transducer at 1200 rpm, no load.

The initial work on the two sensors was carried out on the single cylinder Robin engine. The next stage was to apply the sensors to a multi-cylinder engine with a common cylinder head.

3.5 Mk 2 sensor on a multi cylinder engine

The Ford 1800cm^3 engine has exposed cylinder head studs on one side, and on the other, they are contained within the camshaft cover. Two Mk2 sensors were placed on the engine as shown in Figure 9. Sensor A was positioned under the camshaft-head nut close to cylinder 1 and Sensor B was placed under the head nut close to cylinder 4. These positions were chosen as the least likely to cause delays to the separate engine research programme continuing on this engine. The ideal positions would have seen the ‘in-cylinder’ transducer mounted in number 2 or 3 cylinder, and four Mk2 sensors positioned around the cylinder. This would have enabled readings of head bolt load to be taken from the minimum distance around the cylinder.

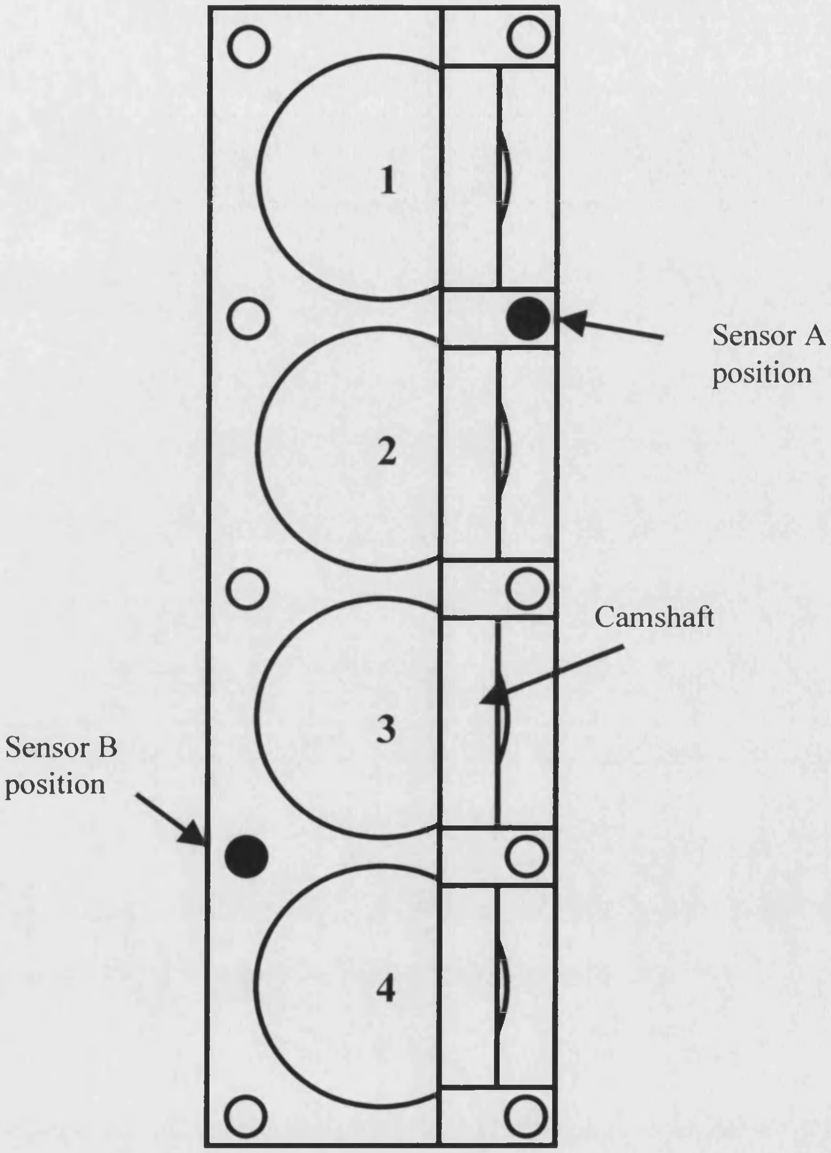


Figure 9 Position of the Mk2 sensors on the multi-cylinder engine

3.5.1 Operating conditions of the engine

The operating condition of the engine must be taken into consideration during the interpretation of the results. All the tests were performed with heavily retarded ignition, in order to place no load on the engine. Thus there were two locations of peak pressure. The first was motored peak pressure (MPP) which was developed as the pressure cycle of a non-firing engine, and the second was combustion peak pressure (CPP) which occurs after start of combustion (SoC). In an engine under normal operating conditions the SoC occurs before MPP, and thus MPP cannot be determined. Thus in this research MPP was detected as the location of peak pressure.

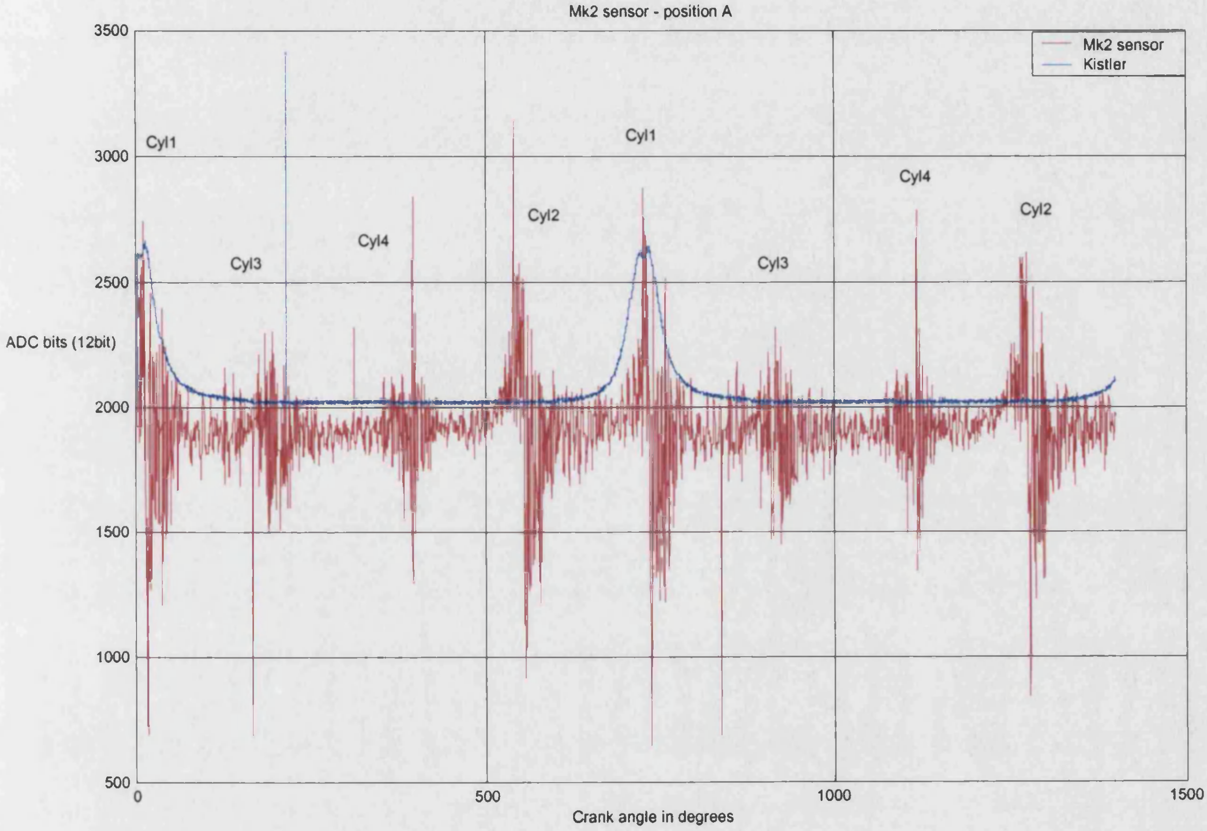


Figure 10 Raw signal from the Mk2 sensor in position A

3.5.2 Initial analysis of engine test data

The raw signals from the Mk2 sensors are shown in Figure 10 and Figure 11 along with the in-cylinder pressure transducer. The signals were taken with the engine running at 1750rpm, 4 degrees of timing ATDC, and 43 NM of load. At first glance there was a marked contamination of the signals from noise.

The signal traces clearly showed the excitation of the sensors by the action of all four cylinders. The signals differed in their composition due to the placing of the sensors. Sensor A was placed under the head bolt between cylinders 1 and 2, and sensor B was placed under the head bolt between cylinders 3 and 4. Additionally the head bolt for Sensor A was used to retain the camshaft bearing. Thus sensor A was subject to excitation from the camshaft as well as cylinder pressure.

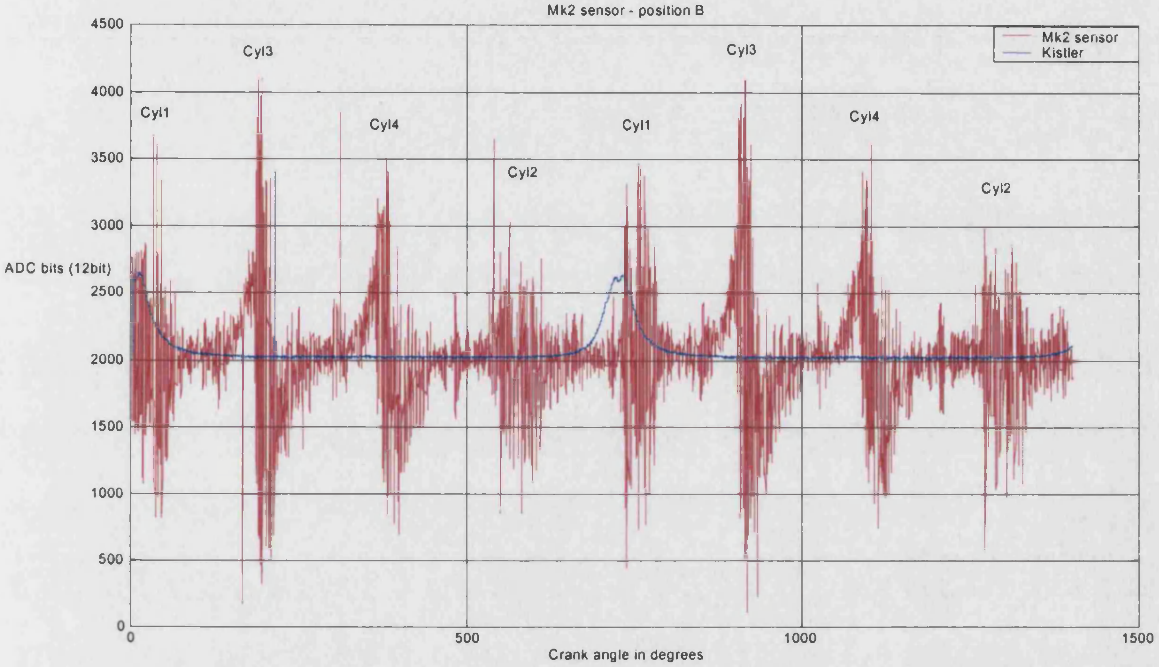


Figure 11 Raw signal from the Mk2 sensor in position B

As was expected, the sensors detected the excitation from all four cylinders but the cylinders closest had greater information due to the proximity of the sensors to the source of excitation.

An analysis of the power spectrum of the signals revealed that the sensor in position A had signal energy spread over much of the spectrum (see Figure 12), whereas the signal from the sensor in position B showed a degree of grouping (see Figure 13). For clarity, the running speed of 29.1 Hz at 1750 rpm has not been shown.

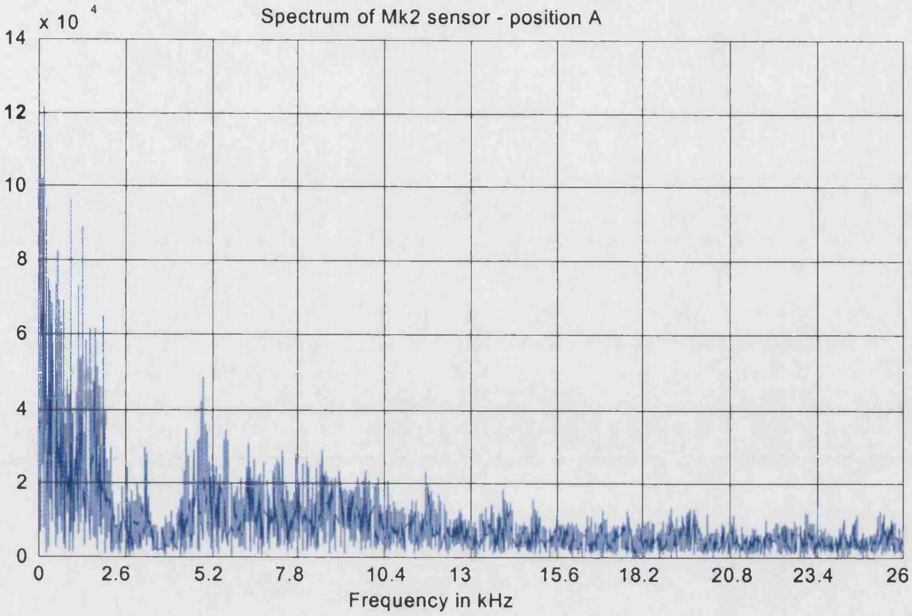


Figure 12 Spectral content of raw signal from Mk2 sensor in position A

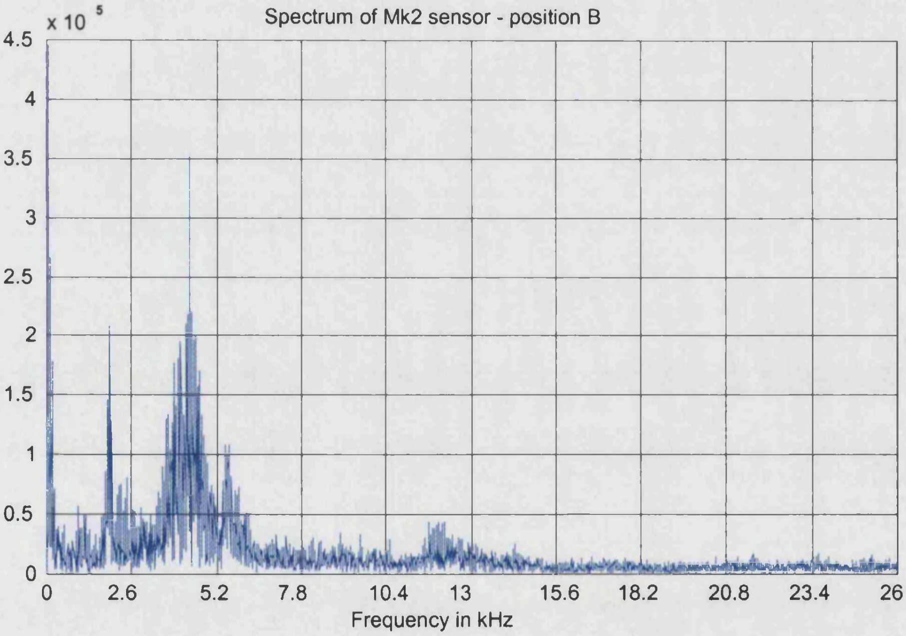


Figure 13 Spectral content of raw signal from Mk2 sensor in position B

An analysis of the Mk2 sensor data is shown in Figure 14. This is a graph of the Mk2 sensor in position B, with TDC for cylinder 3 centred at 0 degrees crank angle. From the detail (see Figure 15 and Figure 16 overleaf) there was a dramatic

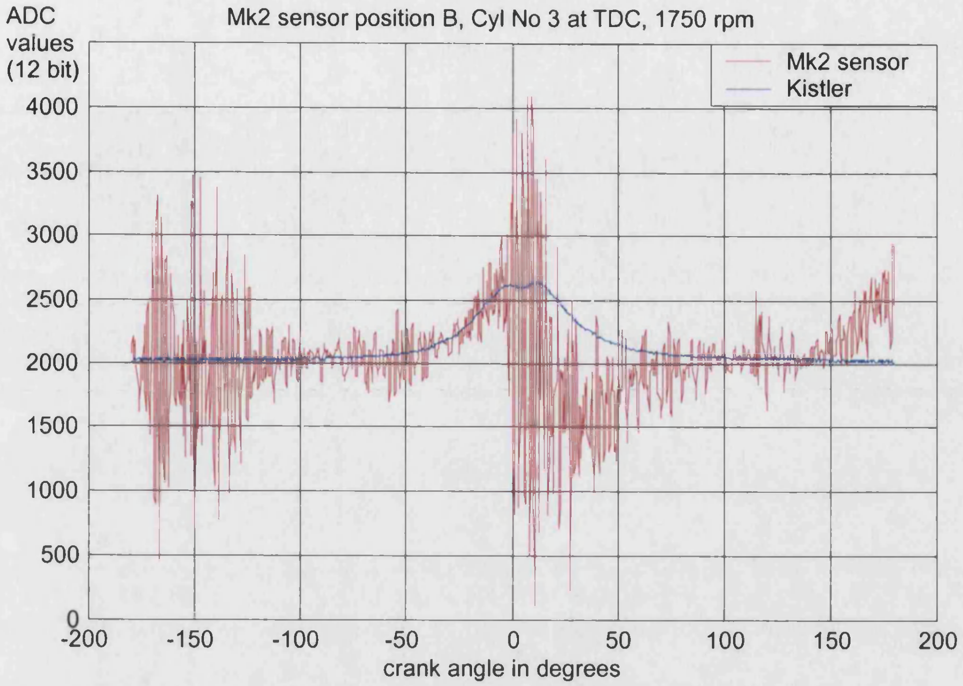


Figure 14 Mk 2 sensor position B, Cylinder No. 3 at TDC

increase in energy around location of MPP and SoC. It was these turning points which needed to be determined from the signal. A further close examination revealed some significant changes in the underlying energy of the signal. During the rising edge of the compression stroke, the low frequency energy of the Mk2 sensor matched the in-cylinder sensor. An abrupt change in the energy content then occurred at MPP. The oscillations produced at MPP were then subject to interference at SoC. The oscillations then continued in a diminished state for a further 90 degrees. The examination of the Mk2 sensor output at 2000rpm, 40NM load, revealed that the large oscillations, which occurred at MPP and SoC, were not fully independent of engine speed. However, the abrupt change in the underlying energy of the signal indicated the location of MPP and this was independent of RPM.

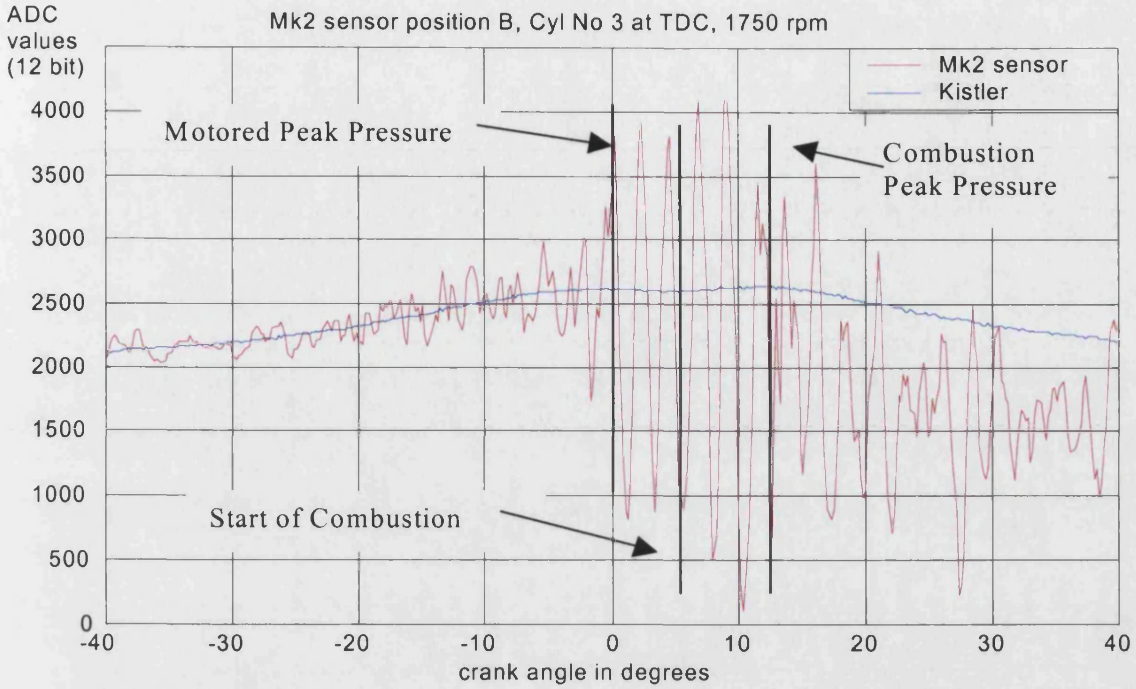


Figure 15 Close-up of data from Mk 2 sensor, position B, cylinder No. 3 at MPP

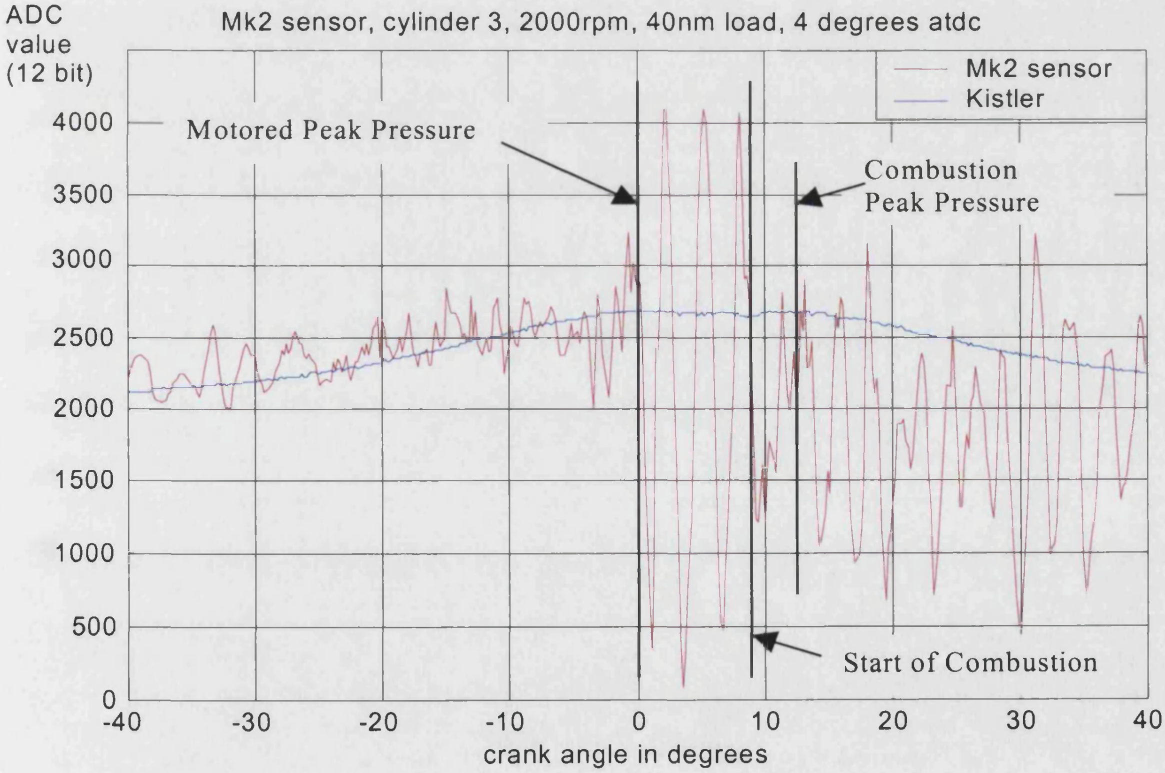


Figure 16 Mk 2 sensor, position 2, cylinder No. 3 at TDC, 2000 rpm

3.6 Preliminary analysis of signals

An analysis of the Mk2 sensor data using standard techniques was undertaken. The primary motivation was to establish the set of routines with which to build the analysis algorithm. The primary goal of the algorithm was to extract MPP, SoC and CPP from the sensor data, in real time. In Figure 15, MPP was characterised by an abrupt change in the energy of the signal in the 4.5kHz band. SoC was less well defined but was characterised by a high frequency component superimposed on the 4.5kHz signal at SoC.

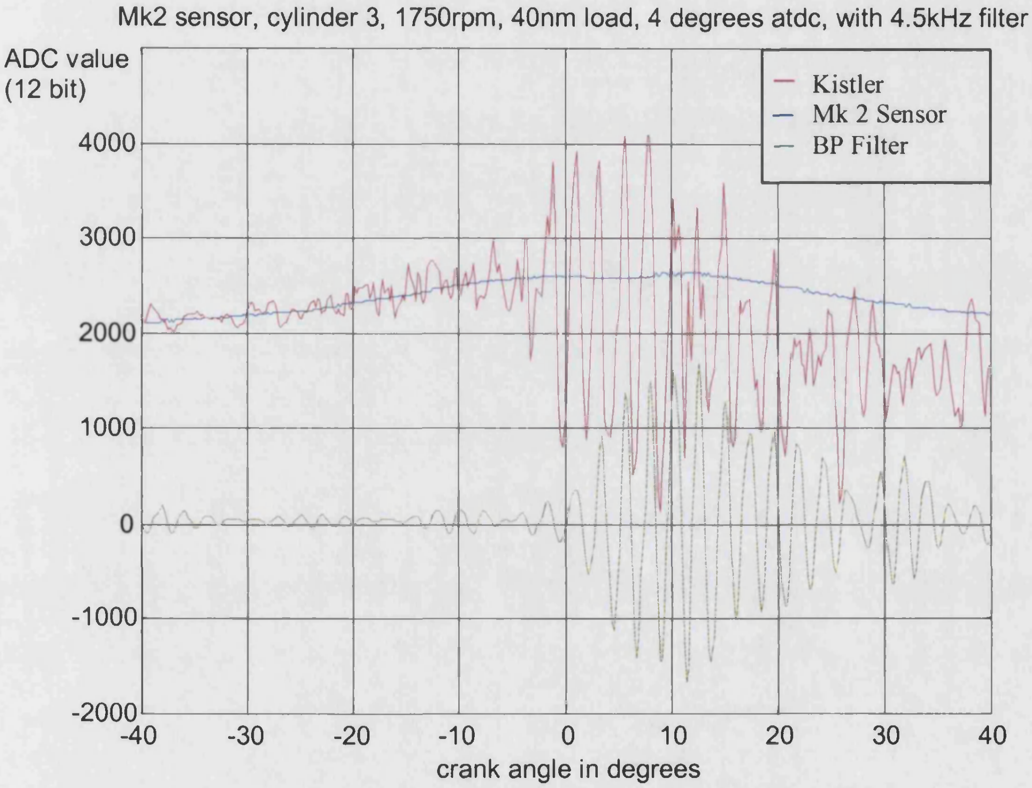


Figure 17 Mk2 sensor data with 4.5kHz band pass filter

In Figure 17 the Mk 2 sensor data has been band pass filtered (using the filter with characteristics shown in Figure 18 and Figure 19) to try and extract the MPP signal. Although the band pass filter had detected the feature, the point at which it started was not clear. All that could be drawn from the filtered signal was that combustion had occurred sometime after MPP. The design of the filter followed standard FIR filter design methods and was developed in MATLAB.

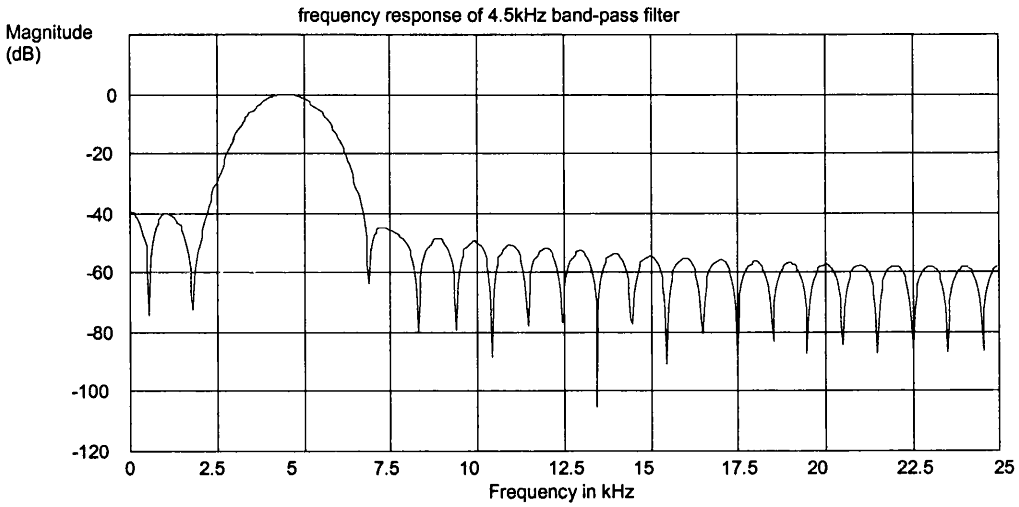


Figure 18 Frequency response of 4.5kHz band-pass filter

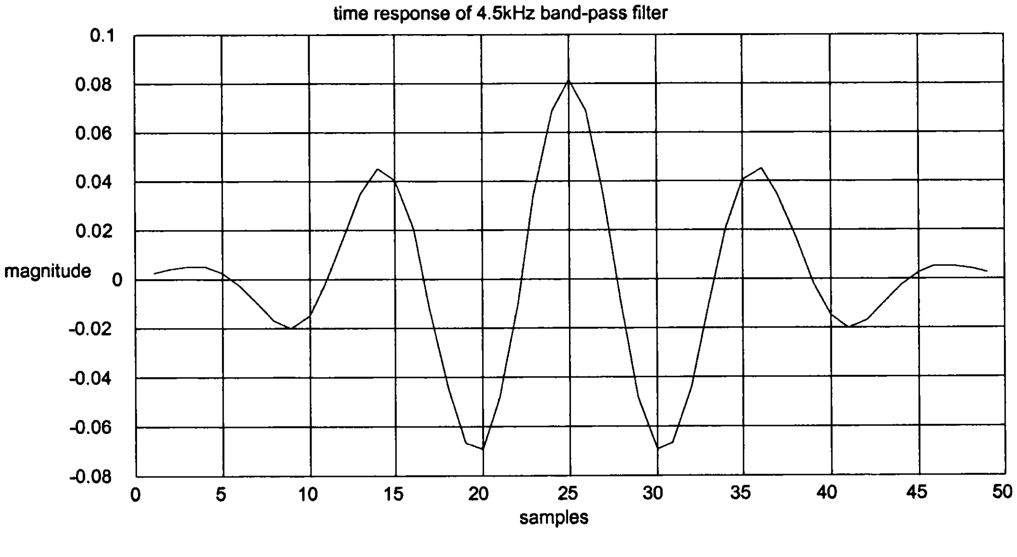


Figure 19 Time response of 4.5kHz band-pass filter

The main reason for the lack of precise detection of MPP was due to the large number of terms in the time response of the filter (see Figure 19). The main problem was that as the selectivity of the filter in frequency was increased, so the time response of the filter increased. This affected the transient response of the filter, and longer 'wind-up' periods were required. The 'wind-up' on the filtered signal was due to the finite time response of the filter. What was required was a frequency filter with a low number of terms in the time series, to reduce the response time and decrease the 'wind-up'. This requirement is developed further in Chapter 5.

3.7 Discussion

This Chapter was concerned with the design of the sensors used to detect two fundamental variables of the combustion process - start of combustion and location of peak pressure. Firstly, it outlined the design of the sensors for single cylinder engines, drawing upon the work conducted by Kirkman, and developed a miniature load washer to detect SoC and LPP. Secondly, it detailed the development of a further sensor – the Mk2 – which overcame the head bolt clamping torque problems (identified by Kirkman) on multi-cylinder engines. Finally it detailed the development of a finite impulse response (FIR) filter for use in the detection of SoC and LPP and it demonstrated that traditional FIR filters do not perform well at detecting features based in time. Work on detecting features of the combustion process which are time and frequency dependent will be investigated in Chapter 5.

4 DIGITAL SIGNAL PROCESSING SYSTEM

4.1 Introduction

The analysis of non-intrusive data was performed in two phases. The first phase involved 'off-line' data acquisition and storage. The second phase was 'real time' acquisition and processing of the signals. The term 'real time' described the acquisition of data from the engine and processing in one engine cycle.

To accomplish these two phases, two systems were used. The first was a data acquisition card with the off line analysis performed in Matlab.

The second used a custom built digital signal processing system.

4.2 Off-line acquisition and processing

The off-line system performed two tasks. The first enabled engine data to be captured and stored during a test. The second used a signal processing package to process the stored data.

4.2.1 EZ-LAB SHARC development board

The off-line data acquisition was performed using an EZ-LAB development board by Analog Devices. This contained an Analog Devices ADSP21062 processor, PC AT interface circuitry and a number of communication links. An analog to digital subsystem was built, which had 4 channels, 1.2MSPS at 12bit per channel, and communicated to the ADSP21062 using two of the links.

The choice of acquisition board was determined by the design of the real time system. The real time system used the ADSP21062 processor for reasons discussed later. Thus the EZ-LAB development board was used as the off-line acquisition system, and as the development board for the real time system.

4.2.2 Application code for the EZ-LAB development board

To enable the capture of data from the ADC subsystem, an embedded program for the EZ-LAB board was developed. This had the task of waiting for activity on the two link ports, and then transferring the data to internal SHARC memory. The program would then issue an interrupt to the PC to indicate that data was ready for collection. The program also had the ability to perform digital filtering on the incoming signal. Appendix B shows the listing of the C program for the EZ-LAB board.

4.2.3 ADC interface

The ADC interface worked in the following way. The SHARC processor has 6 general purpose bi-directional links. However, only two links are available on the EZ-LAB board. These links can operate at up to 2x the frequency of the processor, thus with a 40MHz clock a maximum 80MHz transfer rate can be achieved. Each link is 4 bits wide with clock and acknowledge (6 wires). The SHARC assigns one DMA channel per link, and can DMA 48bits in one clock cycle. The link ports automatically pack or unpack 4 bit data into 48, 32 or 16 bit values (user selectable). The ADC board packs two, 12 bit ADC channels onto one link port (the remaining 12 bits are not used), at a frequency which is driven by the external clock. The maximum throughput of the link is 40MSPS aggregate, thus the maximum frequency of the converters is 20MHz. The converters used were Linear Technology LTC1410CSW, which have a maximum frequency of 1.2MHz. By using 2 converter boards, 4 channel, 12 bit, 4.8MSPS total throughput was achieved. The engine used an 1800ppr encoder (0.2 crank angle), and had a limit of 6000rpm. This gave a maximum of 180kHz per channel, well within the specification of the ADC system.

The logic for the converter boards was derived using boolean logic and state machines. A Lattice 'in-circuit programmable' EPLD was used as the logic device and was programmed using the Lattice programmable development system software. Appendix B shows the circuit diagram, board layouts and lattice EPLD files for the ADC subsystem. Figure 20 shows a picture of the completed circuit board.

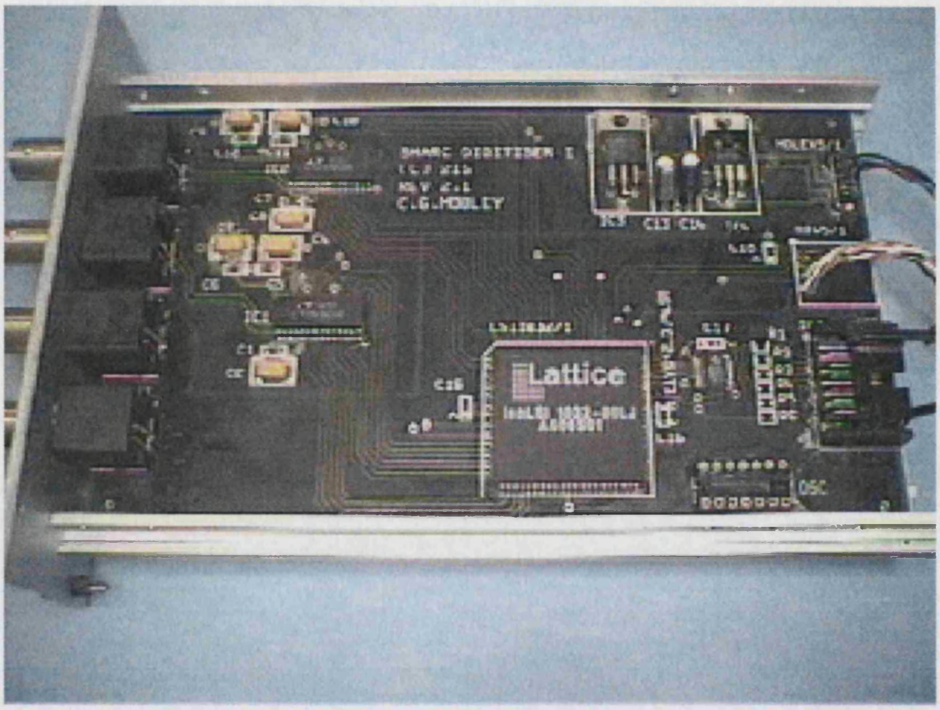


Figure 20 Circuit board for ADC subsystem

4.2.4 cmSnapshot

A program was needed to display and store the captured data. Standard spreadsheets were available, but proved to be difficult to view large data sets. The EZ-LAB system could store 2 revolutions of the crank, which contained 3600 points per trace, and 4 traces per run. It was decided to develop an ‘oscilloscope’ program which could display the traces. The program was designed to run in the Windows 3.11/95 environment and was built using C++ and object oriented techniques. The basis of the program was a foundation class library called wWindows. This provided all the advanced graphics functions required and was designed to interface directly with the Windows 95 API.

The software was designed to view data from various sources. Files with the .snp extension could be opened directly, and also ASCII files could be imported. Data from the EZ-LAB board could be uploaded, displayed and then logged to disk. Matlab was then used to process the data files.

4.2.5 Matlab

Matlab was used to analyse the captured data. This was chosen as it offered rapid analysis of the data and had standard signal processing functions available. The use of the verified functions within Matlab reduced the chance for error in the analysis of the data.

4.3 Real time processing system – 'Compass'

The requirements for a real-time processing system were examined and the following chosen as the salient features:

- Floating point performance in 32 bit.
- Cost.
- Ease of development and construction.
- Availability of assembler, C compiler, and DSP libraries.

There are many processors available, which could be used, but few could match all the above requirements.

The following sections detail the choice of processor, the outline of the system and the some of the detailed issues arising from developing and building the system.

4.3.1 Review of digital signal processors

4.3.1.1 Requirements

In the above paragraph the floating point performance of the processor is of prime importance when choosing the processor. However, it was necessary to quantify the amount of performance required. This can be done by analysing the type of algorithms which are going to be run on the system. For the non-intrusive sensor project, several key functions would need to be performed, these are:

- FIR Filtering (convolution)
- FFT and IFFT
- Correlation (auto and cross)
- Matrix manipulation

These functions would need to be calculated within 1 revolution of the engine, so assuming an upper limit of 10,000 rpm and V12, there is 2ms between successive TDCs. Any signal processing needs to be completed within the 2ms window.

4.3.1.2 Digital signal processor vs. conventional processor

A digital signal processor is a real-time microprocessor optimised for computation-intensive signal processing applications. It differs from normal microprocessors in several key areas.

a) Digital signal processors have hardware to **directly implement basic DSP operations**. For example, most are designed to directly perform multiply and accumulate in 1 cycle.
b) DSPs have support for **circular buffers**, which are an important feature of many signal processing algorithms. A circular buffer is a region of memory that wraps around. The buffer uses a pointer that automatically resets to the beginning of the buffer if the pointer is advanced beyond the last location in the buffer. Buffers are used in the routines to store filter coefficients and implement a delay line of input samples. Performance suffers if the DSP has

to perform pointer calculations for the circular buffer as well as the calculations for the routine.

c) **Data registers**. The number of general-purpose data registers available can impact the code performance. Fewer registers require intermediate results to be stored in memory, decreasing performance and increasing the load on the memory bus. The DSP should have enough registers to store all the intermediate results of a radix-4 FFT kernel.

4.3.1.3 DSP performance

Three digital signal processor families were considered:

- Analog devices ADSP210x0 family.
- Texas Instruments TMS320Cxx family
- Motorola DSP96002 (24 bit processor)

Table 4 shows the capabilities of the processors considered.

	ADSP-21060	TMS320C40	DSP96002
Instruction Cycle Time	25ns	40ns	50ns
Arithmetic Instruction Execution	25ns	40ns	50ns
Memory Move to Register	25ns	40ns	50ns
Interrupt Response	3 Cycles	4 Cycles	3 Cycles
DMA bandwidth	240MB/sec	100MB/sec	80MB/sec
Divide	6 Cycles	9 Cycles	7 Cycles
Inverse	6 Cycles	8 Cycles	7 Cycles
Square Root	10 Cycles	12 Cycles	12 Cycles
FIR Filter (96 Tap)	2.5us	4.05us	N/A
Radix-2 Complex FFT	0.51us	1.54ms	1.05ms
Radix-2 Real FFT	0.26us	1.01ms	0.64ms

Table 4 DSP comparisons

Table 5 gives the 1024 - point complex FFT performed on several 32-Bit floating point DSPs.

DSP Processor	Instruction Rate	Instruction Cycle Time	Number of Cycles	Total FFT time
TMS320C30	20MHz	50ns	60,800	3.04ms
TMS320C40	40MHz	25ns	38,945	0.97ms
TMS320C6701	167MHz	6ns	19,875	0.12ms
ADSP-21065L	60MHz	16.67ns	18,221	0.31ms
ADSP-21160	100MHz	10ns	9,111	0.09ms

Table 5 FFT performance of various processors

4.3.1.4 The effect of memory size on performance

The choice of processor is also greatly affected by code size. All of the above tests were performed using a small code size FFT routine. The routine was of sufficient size to reside within the memory of the devices tested. However, in reality there would be many parts to a signal processing system, which would require the use of more memory. If the code size is larger than the available memory within the device a significant penalty in processing speed is incurred.

The problem stems from the arrangement of the buses within the processor. The most efficient implementation of the load/store architecture is to have 3 separate buses for the 3 arguments required in performing the mathematical operation. One bus is used to load the instruction and the other two are used to load the arguments.

A good example of this is in the filter routine, where the input data is multiplied to the filter coefficients and the result added to a running total. To achieve this in one clock cycle, the instruction and data items need to be fetched from memory simultaneously, and the result is added to a temporary accumulator.

4.3.1.5 Analog devices ADSP21060

From the previous section the decision was made to use the ADSP-21060 processor. It has a number of architecture features which make it excel at DSP computations, and these are highlighted below.

4.3.1.6 Super Harvard architecture (SHARC)

The ADSP-21060 (SHARC) uses the idea of the super Harvard architecture. This concept uses three separate buses, one for instructions and two for data, to perform DSP tasks. The three buses used internally come from 3 sources: the instruction cache, the memory bank A and the memory bank B.

To write assembly code for the SHARC, the programmer must assign the filter coefficients and the instruction code to one memory bank and the input data stream to another bank. The loop which performs the filter operations must be within the size of the cache memory, otherwise a 'cache miss' will be suffered. The first time the processor runs the filter code the cache will be empty, and the processor must perform two cycles to fetch the instructions and two data operands. On the second pass through the filter code instructions will arrive from the cache, and the two data operands will arrive from the data buses, thus only a single cycle is needed.

For example, the SHARC assembler instruction

```
R7 = R6*R5,     DM(I0,M1)=R6,        PM(I1,M2)=R5
```

would compute register 6 * register 5 and place the answer in register 7; register 6 would be loaded from data memory using index register I0; and register 5 would be loaded from program memory using index register I1, in one clock cycle.

The SHARC enables one of the internal buses to be connected to external memory. This enables single cycle operation of program code from external memory with the following restrictions. To achieve single cycle operation the external memory cannot contain the instructions, and both data operands. The arrangement must be, either, instructions and operands 'A' in internal memory and operands 'B' in external memory, or instructions and operands 'A' in external memory and operands 'B' in internal memory.

In the case of the ADSP21060, which has 4Mbit of on board memory, the maximum code size is 80k words of 48 bit instructions. The maximum data size is 128k of 32 bit floating point data. The arrangement of the algorithm in the system depends upon the size of the code and data. If the code and filter coefficients are small in size (i.e. <80k words) then this can reside on internal memory, and the data can reside in external memory. If however the code and coefficients are large (i.e. >80k words) then this must occupy external memory and the data internal memory.

4.3.1.7 Multi-processing

The SHARC processor can form part of a larger system and can be connected in two ways. The first is via shared memory, and the second is via link ports.
The SHARC has in-built support for shared memory cluster programming. Up to 6 processors can be connected directly together via their external data and address buses. Processor interlock signals provide a hardware method of communication and the address space of the 6 processors forms a continuous space which can be accessed by and of the processors. A version of the SHARC is available which contains 4 processors on one BGA.

The second method for multi-processing is through the use of the link ports. The link ports behave in a similar fashion to the transputer links. The SHARC has 6 link ports, with each port 4 bits wide. Each link is clocked at twice the frequency of the core processor, thus can transfer 8 bits per clock cycle. At 40MHz the SHARC has a transfer rate 240Mbytes per second over the link ports.

4.3.1.8 Direct memory access

To move data in and out of the processor 10 DMA channels are available. These can be used to perform reads or writes of data from one part of the processor to another without affecting the core processor activity. The DMA channels have direct access to internal memory and can read & write without stalling the main processor. The DMA channels can transfer up to 240 Mbytes/s across the link ports, external memory and internal memory.

4.3.2 System design – 'Compass'

See Appendix C for the schematics, printed circuit board layouts, EPLD files and C & assembler programs.

4.3.2.1 Overview

The main aim of 'Compass' was to provide a real-time processing system for analysing non-intrusive pressure data. However, the design was affected by the requirement to be compatible with existing automotive equipment. Existing equipment used a controller area network (CAN) interface as the communication format between electronic control units and the desktop PCs.

The University used Matlab / Simulink to develop and simulate control strategies and Matlab / Real Time Workshop to embed the strategy into the ECU controlling the equipment.

An important requirement of 'Compass' was to be able to integrate into this development cycle, so that the non-intrusive sensor work could be developed into the overall engine control strategy. However, initially the coding for 'Compass' was done in 'C' using standard math libraries from Ixthos.

4.3.2.2 Separation of model and communication processes

In the design of 'Compass' the main digital signal processing functions of the system took highest priority. However, higher functions, which could not be interrupted, such as data acquisition and communications, could not be allowed to impinge on the performance of the DSP functions. This led to a dual processor system, where the SHARC would perform the DSP functions – called the 'model' and the other processor would perform data capture, communications and other system functions.

By separating the two functions of model and communications the software for the model could be developed independently of the system.

The model processor accessed all its external data via a shared memory block. This enabled the SHARC code to refer to the incoming (raw) and outgoing (computed) data stream by pointers, which were re-locatable. The shared memory block could be accessed by the communications processor, which would input or output the data accordingly.

4.3.2.3 Communications processor

The communications processor was responsible for capturing data from the sensors on the engine and passing these to the shared memory. It was also responsible for collecting processed results and sending them to the desktop PC via the CAN and/or serial interface.

The communications processor used was the Motorola 68376, which is based upon the ubiquitous MC68000. The reason for this choice was ease of use and familiarity with the architecture. The 68376 is a 32 bit embedded microcontroller, with on board CAN and a suite of integrated peripherals. It is available in a surface mount package at an operating frequency of 20MHz, and in an extended temperature range. The device is widely used in the automotive industry, and can provide the required levels of performance at low cost.

4.3.2.4 Communication standards ASAM – MCD1a

The communications between the desktop PC and 'Compass' needed to be compatible with existing and future automotive equipment. There are many standards in use in the automotive field with several manufacturers developing similar but incompatible systems. One standard adopted by the German automotive industry is the set of ASAM standards. The ASAM standard defines the communications protocol between ECUs and is called CAN calibration protocol (CCP). It is this standard which the University has adopted and was implemented on 'Compass'.

4.3.2.5 Operation of Compass

A DSP model was written and compiled using the SHARC compiler and Ixthos DSP libraries. This was then downloaded into 'Compass' via the CAN link and the CCP protocol. Once the model was installed into the SHARC memory, the system could be started. The communications processor controlled the operation of the SHARC and issued commands for reset, start, stop etc.
The communications processor received commands via the CAN link from the desktop PC using the CCP protocol. The PC received calculated data from 'Compass' via the CCP layer stack.

Once a DSP model had been debugged and produced the desired results, it could be stored in the FLASH memory of 'Compass', for immediate use without the need for the desktop PC. 'Compass' was then connected into the engine and transmission control strategy via the CAN link.

4.4 Discussion

In this Chapter, the digital signal processing equipment was outlined. It started with an outline of why the system was needed, and reviewed the current crop of high performance micro-processors available. The needs of the project fell into two areas: data acquisition and analysis and real time DSP. The two areas were examined and the equipment designed and constructed for each task was detailed. The equipment formed the heart of the project and enabled the algorithms developed in subsequent Chapters to be validated.

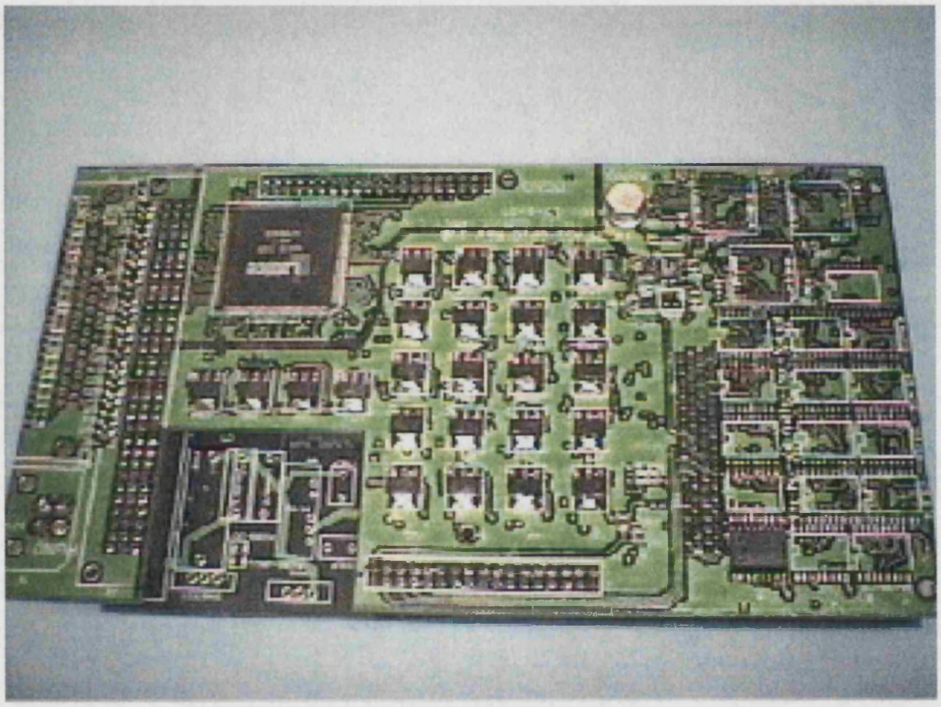


Figure 21 Main board for compass

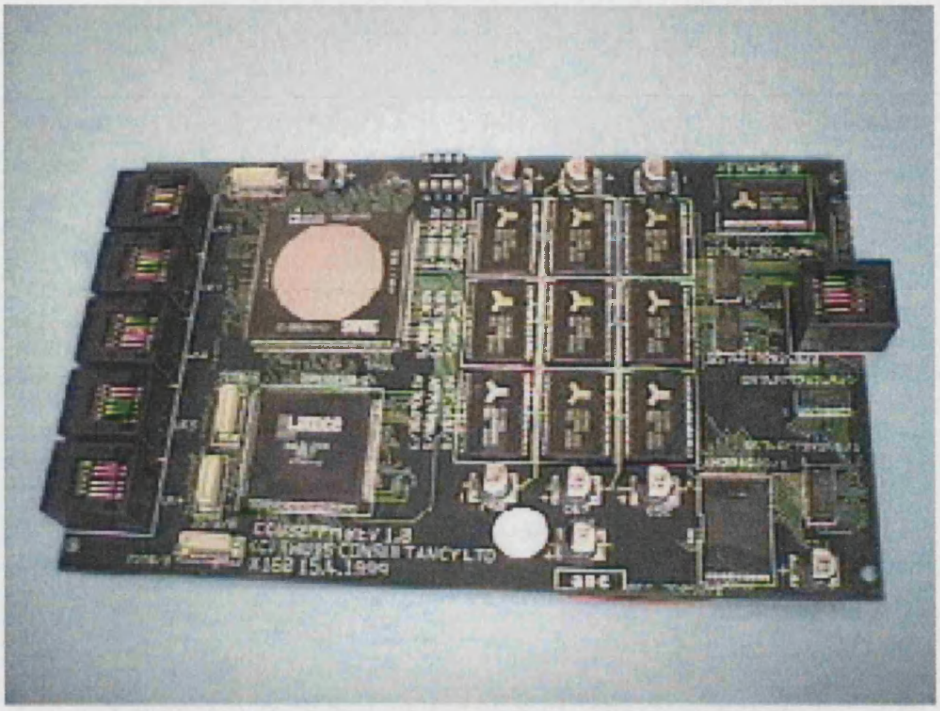


Figure 22 Sharc processing board for Compass

5 *DESIGN OF FEATURE EXTRACTION SYSTEM*

5.1 Introduction

Chapter 3 gave details of how the sensors were developed which enabled the non-intrusive capture of the combustion process. However, the parameters of interest within the combustion process were non-deterministic and were masked by other signals from the engine, collectively described as noise. In section 3.6, the difficulty of using traditional techniques on these non-stationary signals was demonstrated.

In this Chapter, the techniques needed to approximate the non-deterministic signals produced by the sensors were explored. The approximation was developed from a series of deterministic global and local functions. This deterministic approximation could then be used by the control system in place of the non-deterministic signals. The system of approximation chosen depended on the nature of the signal. If the signal was said to be 'stationary', i.e. it was cyclic in nature, for example a square wave, then it could be approximated by a global basis function. If it was non-stationary then it could be approximated using a local basis function. This Chapter begins with a review of global basis functions and Fourier analysis. It then introduces local basis functions and wavelets. Finite impulse response filters and multi resolution analysis are discussed and finally several wavelets families are examined and the choice is made of one of these for use in this research.

5.2 Global basis functions and the Fourier transform

5.2.1 Time and frequency

A signal has two fundamental representations in time and frequency. Each is the dual of the other and the signal exists in both domains simultaneously. At any given instance in time, the signal is made up of an infinite number of frequency components and it is not possible to know both its frequency and time period. The spectrum of a signal is a measure of the power of these components. The problem with calculating the spectra of signals is that it is assumed in the calculation that the signals are stationary, i.e. that they do not change over time. However, real world signals are usually non-stationary, and this causes problems in calculating their spectra. If the spectrum of a non-stationary signal is to be obtained, then the calculation must include all the signal from – to + infinity. However, if the signal is time windowed i.e. not all the signal is used, then a band limited spectrum results. This leads to an interesting problem in that the spectrum of a signal cannot be produced at an instance in time unless the frequency spectrum is truncated. As the time window for the instance in time is reduced, so the truncation in frequency is larger.

In summary, for non-stationary waves, it is not possible to know what the spectrum of a signal is at a single instance in time; it is only possible to estimate the spectrum over a short time period.

5.2.2 Fourier transform

One standard technique by which the approximation of the signal can be determined is the Fourier transform. This method uses sets of sines and cosines as the global basis functions to approximate the signal. The approximation then has the two representations in time and frequency. The time representation is given by the summation of the sines and cosines in time, and the frequency representation is given by the sum of the sines and cosines in frequency. The global basis functions used in the Fourier transform assume infinite width, in time and frequency. However, as discussed in 5.2.1, if the truncation of the global basis functions or the signal occurs, then errors are introduced into the approximation. A truncation in time results in errors in frequency and a truncation in frequency results in errors in time.

5.2.3 Window functions

The majority of errors introduced by truncation of the signal occur at the discontinuities at the start and end of the signal. A jump discontinuity in time produces infinite ripples in frequency, which distort the spectrum of the signal and introduce errors into the approximation. One method of reducing the effect of truncation is the to use windowing functions. These are mathematical functions - such as the Gaussian window – which have been designed to reduce the frequency components of the truncation to a minimum. They are applied to the signal and have the effect of smoothing the transition from no signal to full signal.

5.2.4 Short term Fourier transform (STFT) and local bases

In Fourier analysis, the bases used to analyse the signal are global, i.e. they have infinite duration in time and frequency. However, by using windowing functions the global bases can be thought of a local bases, which have finite duration. Thus the series of sines and cosines used with a window function can be used to approximate a signal with finite duration.

In the last step to develop the STFT, a window size is chosen which is smaller than the duration of the signal. The window is then shifted along the signal in discrete steps. Thus, using the STFT, time localisation and frequency localisation can be developed, and a frequency spectrum is developed for each shift. The STFT produces a frequency – shift diagram, where the shifts represent steps in time. The sines and cosines used in the windowing function are called local bases since they have finite duration.

It is interesting to note that the window size of the local basis can also be altered in relation to the signal, and a new series of shifts computed. This, in theory, would produce frequency spectra at different time shifts for different window sizes. Each size of window would emphasise aspects of the frequency domain. Small windows would highlight high frequency components, and large windows would show the low frequency components. However, due to the action of the windowing function on the underlying sinusoidal waveforms, this is difficult to realise.

5.3 Local bases functions - from STFT to Haar

In the definition of the STFT, the concept of local bases was introduced. In this case, global bases were altered by a windowing function to form local bases.

However, there exist many local bases which are not derived from sinusoidal functions. The simplest of these is the Haar basis [17] in which the approximation waveform is a pulse. The pulse has a finite window, over which it has the value of 1, and is 0 at all other instances. By adjusting the height, size and shifting of the local basis an approximation to a global function can be achieved. In the STFT, the window size is fixed, due to complications with the global nature of the sinusoidal functions. However, the Haar basis does not have such complications and can make use of the different window sizes. The result is a series of waveforms which show the approximation of the signal at a different scale. The local basis is said to have 'decomposed' the signal into time based waveforms, each representing a different scale (frequency content) of the original signal.

The main problem with the Haar local basis is that there is a jump discontinuity at the start and end of each basis. This produces harmonics in the frequency domain, which distort the analysing signal and remove the ability to recreate the original from the approximations. However, several families of local bases do exist which offer perfect decomposition and reconstruction. The main families are discussed in section 5.7.

5.4 Frequency filters and local bases

The local bases introduced in section 5.3 produce a series of time based waveforms, each at a different scale (frequency). This output is similar to the frequency filter. These are also used to separate a signal into various spectrum bands. These devices have stop bands, where the signal is attenuated and a pass band where the signal progresses unchanged. The filter is selective in frequency and will only allow components of the incoming signal to progress unchanged if they have the desired frequency content. All other frequency components of the signal are attenuated.

In the digital domain, the finite impulse response (FIR) filter is commonly used. The main problem in using these filters is that they are designed using sinusoidal global bases – Fourier analysis, and thus they suffer from a lack of local support in time. The result of this lack of local support was demonstrated in Figure 17 where the filter suffered from a period of 'wind-up'.

5.5 Multi-rate systems and filter banks

Multi rate systems are produced when the sample rate of the signal is altered during the processing within the system. They are used in digital signal processing as a method of achieving a reduction in computation time and to adjust the sample rate to the most appropriate value for the bandwidth of the signal being analysed. To reduce the computation required, the aim is to operate, at each point of the system, at the lowest sampling rate, without introducing aliasing effects.

Variable sample-rate signal processing is widely used in the discrete Fourier transform (DFT). As the bandwidth of each of the *N* parallel channels in a DFT filter-bank is only 1/*N*th of the total input signal bandwidth, the output sample rate in each channel can be reduced by a factor of *N* – without incurring any loss of information.

One application of multi-rate systems is in the design of trans-multiplexers. These are devices which convert between frequency division multiplexing (FDM) and time division multiplexing (TDM). There are many other applications of multi-rate systems, but one is of particular interest, the filter-bank.

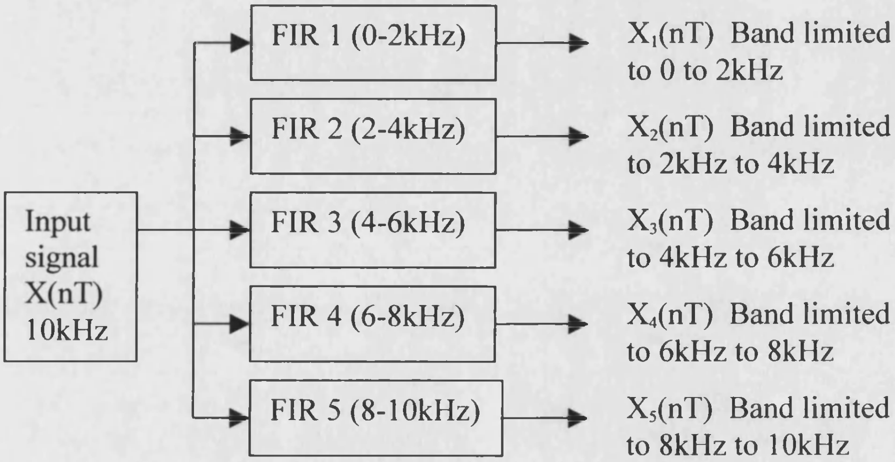

Input signal X(nT) 10kHz
FIR 1 (0-2kHz)
FIR 2 (2-4kHz)
FIR 3 (4-6kHz)
FIR 4 (6-8kHz)
FIR 5 (8-10kHz)
X₁(nT) Band limited to 0 to 2kHz
X₂(nT) Band limited to 2kHz to 4kHz
X₃(nT) Band limited to 4kHz to 6kHz
X₄(nT) Band limited to 6kHz to 8kHz
X₅(nT) Band limited to 8kHz to 10kHz

Figure 23 parallel filter-bank using multiple FIR filters

5.5.1 Filter-banks

Filter-banks are widely used in the implementation of the DFT. In this signal processing system, the filters are arranged in parallel (see Figure 23). In a parallel filter-bank, the designer has complete control over the centre frequency and pass band of each filter. However, each filter must be designed with different filter coefficients, and this leads to inefficient signal processing systems.

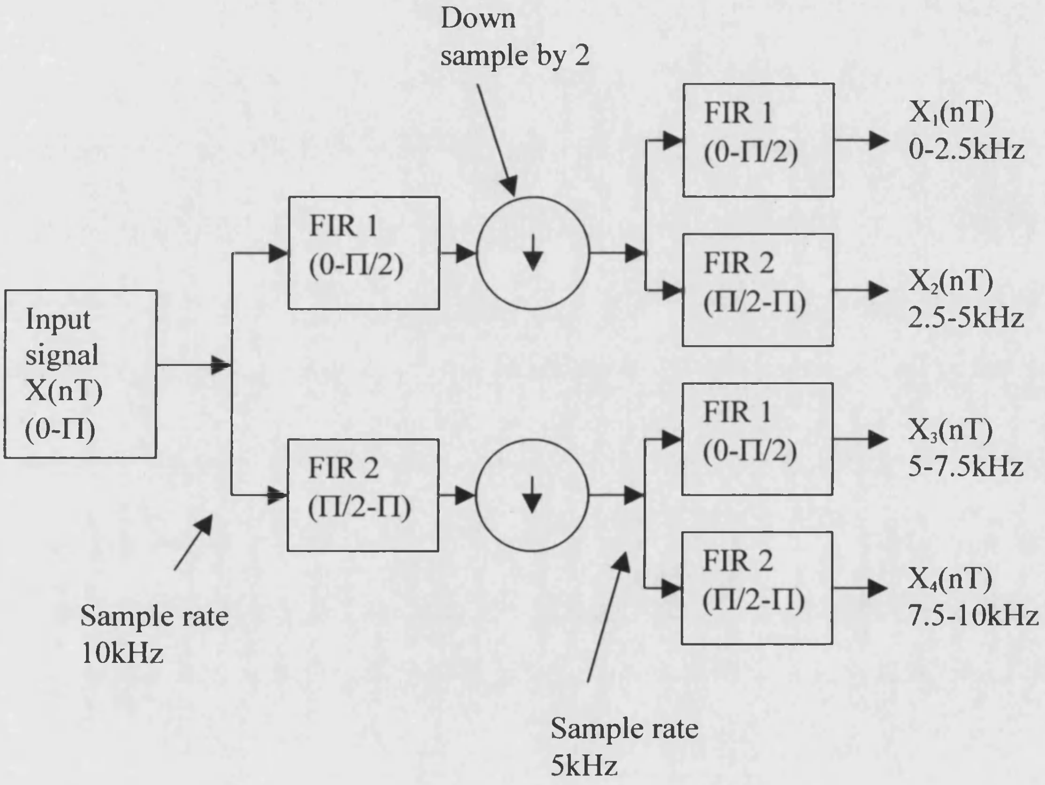


Figure 24 Multi-rate filter-bank using dual FIR Filters

The structure can be rearranged in to a tree configuration (see Figure 24). In this arrangement, a very efficient filter-bank can be realised when the filters are all of equal bandwidth and the number of filters is a power of two.

The filters are of FIR type and at the first or root stage the filter pair splits the signal into upper and lower bands. As the signal progresses down the tree, the sample rate decreases and this effect increases the frequency selectivity of the filters. The important part is that all the filters are an exact copy of the original filter pair at the root of the tree; it is only the sample rate which alters.

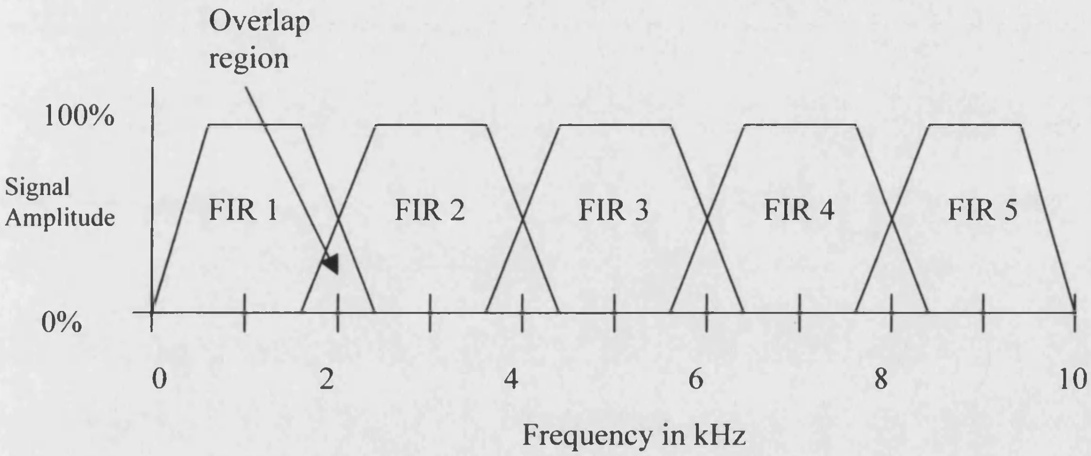


Figure 25 Frequency response of filter-bank using FIR filters

The design of the filter-bank must done with care, since perfect frequency filters are not available. The problem is that the frequency response of imperfect filters causes over-lapping and distortion of the signal. This occurs in the cut-off region between the filter frequency responses. Figure 26 shows a perfect filter-bank

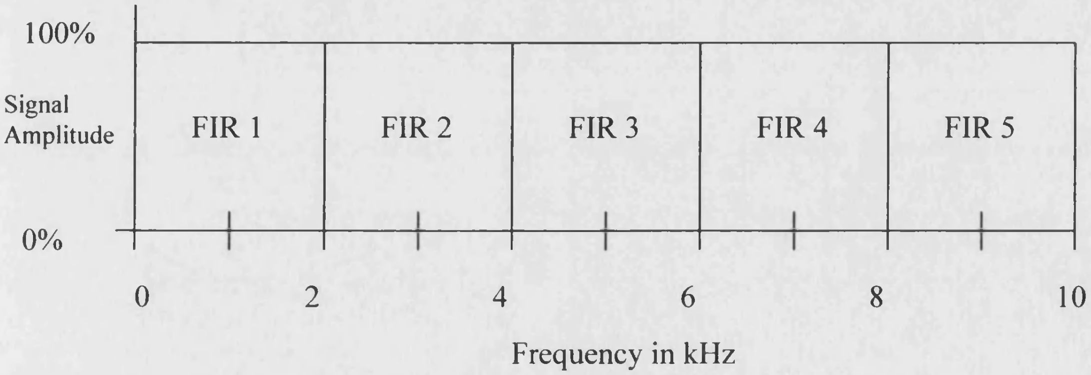


Figure 26 Frequency response of perfect filter-bank

frequency response, and Figure 25 shows the frequency response using FIR filters. If the cut-off rate of the frequency response of the FIR filters is increased, to try and reduce the distortion, then the number of terms in the convolution sum increases.

5.6 Multi-resolution analysis (MRA)

When using a multi-rate filter-bank to decompose a signal it is only necessary to use one side of the decomposition tree. Figure 27 shows the decomposition tree of a multi-rate filter bank.

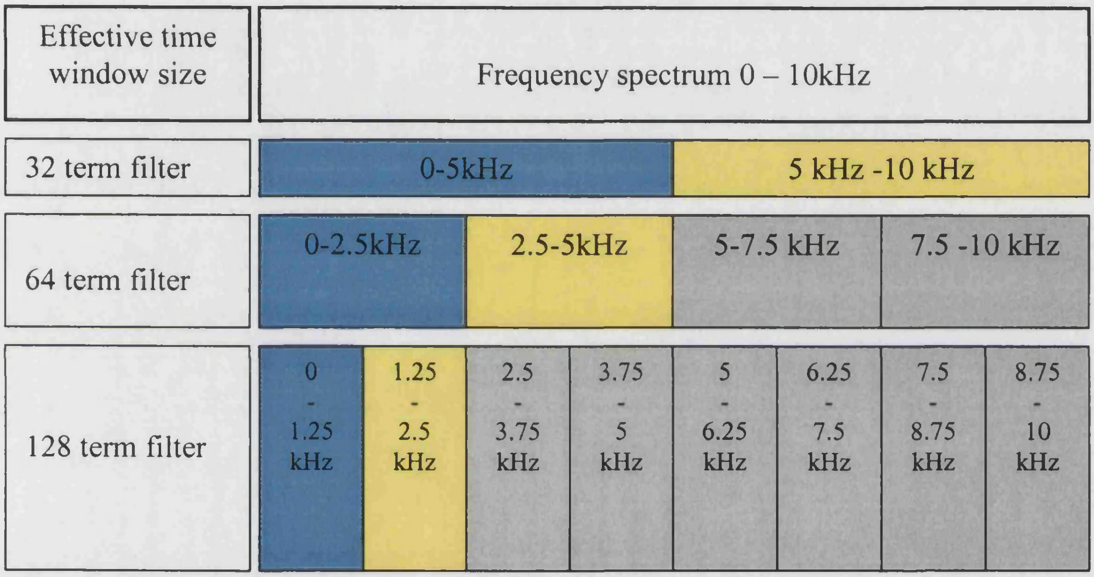


Figure 27 Multi-rate filter-bank showing redundant decomposition.

The initial signal has frequencies up to 10kHz, and the filter used has 32 terms. Note that at each decomposition the number of filter terms is not increased but the sample rate is decreased. After the signal is passed through the 32 term filter and is down-sampled, the maximum frequency content is now 5kHz, so performing another filter operation on the high frequency part of the signal will produce a redundant result. As the signal progresses down through the levels of the tree, so the size of the time window increases and the selectivity of the filter increases. The yellow boxes contain the results of the filter at each stage of the decomposition.

Note that the last stage has a yellow and blue box. The blue box contains the low frequency components. The boxes in grey are redundant as they contain no useful information.

This approach is called multi resolution analysis. MRA introduces the notion of time and scale and that the time and frequency windows are inversely related. As the frequency window narrows, so the time window increases. In Figure 27 the boxes are of equal area, showing the inverse time-scale relationship of the signal. MRA is useful in modelling real world signals, as low frequency signals tend to last for longer time periods than high frequency ones. A single cycle of 240V AC UK mains lasts for 20ms, whereas a single cycle of bird song may last for only 0.2ms.

The problem with implementing MRA with traditional filters is that the signal is subjected to losses and inaccuracies. Traditional FIR filters are implemented by a global basis - an infinite series of sine waves. However, in a digital computer with a limited memory and computational resource, the series must be truncated. The size of the truncated series depends on the level of error which can be tolerated, the size of the memory of the computer and the computational power available. As the truncated series is applied to the incoming waveform, information is lost through the approximations of the filter. This leads to two main problems:

- Any reconstruction of the waveform is not possible, without loss of signal.
- If the filter is truncated to produce a narrow time window then large errors are introduced, and if the errors are minimised then the time window is too large for the filter-bank to operate efficiently.

5.7 Wavelet families

In this Chapter, the need for approximating non-stationary waves has been discussed and the problems in using traditional global bases for this have been highlighted. The STFT was discussed as a potential development for using global bases to approximate non-stationary signals. However, it was noted that the STFT is limited by its fixed window size. FIR and filter-banks were discussed as methods to approximate the signal. However, as these are traditionally developed using sinusoidal global bases they suffer similar problems to the STFT. Finally multi resolution analysis was discussed as a method for reducing the complexity of filter banks, and the notion of time-scale analysis was introduced.

Wavelets are now introduced as a method for approximating signals in time using local bases.

In an ideal world the wavelets of choice would offer:

- Compact support in time and frequency.
- Perfect decomposition and reconstruction.
- Realisation by FIR filter structures.
- Compatibility with multi resolution analysis.

From the above requirements, the local basis used for the approximation must have some general attributes. The first is compact support in time and frequency. This is achieved by using a local basis. A local basis is a signal with finite duration (compact support). Many functions can be used to provide compact support, but only certain functions can be used as a local basis. The local basis chosen must have compact support in frequency (i.e. behave like a frequency filter).

The second attribute is perfect decomposition and reconstruction. The idea of a perfect filter-bank is that a signal can be decomposed into many approximations of differing frequency, and then reconstructed from those approximations. In a filter-bank based upon global basis functions, such as the quadrature mirror filter (QMF) used in telecommunications, the decomposition introduces errors into the approximations. These errors make the perfect reconstruction of the signal impossible. However, using a filter-bank based on a local basis function enables perfect decomposition and reconstruction to take place. This attribute is not strictly necessary if the approximated waveform is not to be reconstructed.

The third attribute is a practical one, in that the wavelets must be realisable by simple filter structures such as FIR filters.

The fourth attribute enables a time-scale view of a signal to be taken. In the STFT, the window (the window offering compact support in time) is of a fixed size. Thus it offers time based frequency analysis, but only at one scale. This has the effect of limiting the ability of the approximation to pick out either the fine details of the waveform or the trends, but not both. By using MRA with a true local basis the window is not fixed and can be adjusted. This allows 'close-up' views of the signal for detecting fine details, and global views of the signal for viewing trends. For wavelets to be compatible with MRA, two wavelet filters are needed, low pass (called a scaling function) and high pass (called the wavelet function). These provide the frequency filter function needed by the MRA process.

There are many wavelets which offer some of the above requirements. However, only a few offer all of them. A brief outline of wavelets and their properties follows.

5.7.1 The Harr wavelet

The Harr wavelet was discussed in section 5.3. The Harr wavelet provides isolation in time by the use of a pulse, but this produces an infinite frequency distribution. However, this distribution tends to 0 at infinite frequency, so isolation in frequency can be achieved. The poor support for isolation in frequency leads to the main drawback of the Harr wavelet.

5.7.2 The Shannon wavelet

The Shannon wavelet [36] is the reverse of the Haar wavelet, as it is a simple pulse in frequency which is an ideal frequency filter. This provides excellent localisation in frequency, but poor time localisation, as the ripples in time extend to infinity. The distribution in time is the main drawback of the Shannon wavelet, which cannot perform perfect decomposition and reconstruction.

5.7.3 The Meyer wavelet

The method employed by Meyer [35] is to apply a smoothing function to the edges of the Shannon wavelet i.e. an ideal frequency filter. The smoothing functions belong to the family of piecewise polynomials, and B – splines are often used. The ideal frequency filter has infinitely sharp cut-off; this produces infinite ripples in time. As the sharpness of the cut-off is reduced so the ripples in time diminish, and the time window decreases. In the limit, the square wave of the ideal frequency filter becomes the Gaussian function. The problem with this design is that compact support cannot be fully achieved. There is always a small time ripple effect in place. Thus Meyer wavelets suffer from the problem of imperfect decomposition and reconstruction.

5.7.4 The Daubechies wavelet

Daubechies wavelets [31] have perfect frequency and time localisation. They also possess the ability to be used in MRA, where the scaling and wavelet functions are needed. The Daubechies wavelets offer the ability to decompose a signal into many time streams, each spanning a different frequency domain. The QMF filters used in the decomposition and reconstruction are implemented with simple FIR filters with between 4 and 64 taps. This gives a low MIPS[6] requirement for the filters. Perfect decomposition can be achieved with these filters, and the sum of the wavelet coefficients reconstructs the original signal perfectly.

The Daubechies wavelets are said to be 'minimum wavelets' [45]. This indicates that they use the minimum number of parameters needed for the specified degree of regularity of interest. This regularity implies that the wavelets possess the maximum number of zero crossings for the minimum number of coefficients. This property makes the Daubechies wavelets ideally suited to efficiently analyse polynomial behaviour [33]. Figure 28 shows the Daubechies scaling function in time and frequency.

[6] MIPS – Million of instructions per second

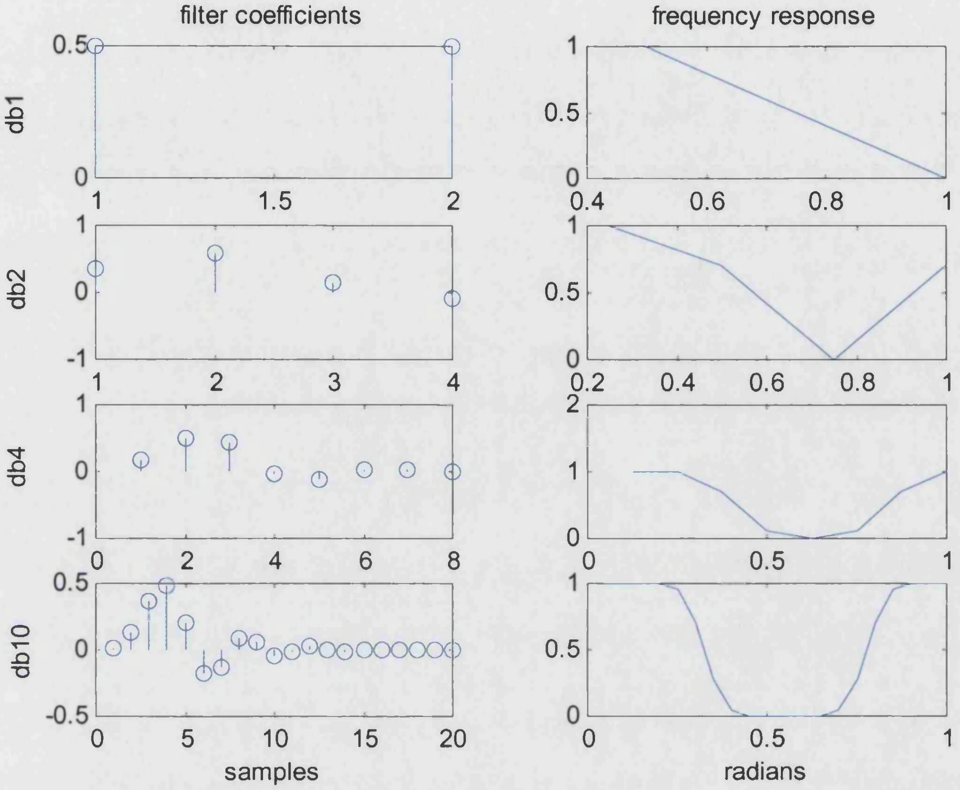


Figure 28 Daubechies scaling function - time and frequency response.

5.8 Wavelet techniques for detection of signals

5.8.1 Detection of discontinuities

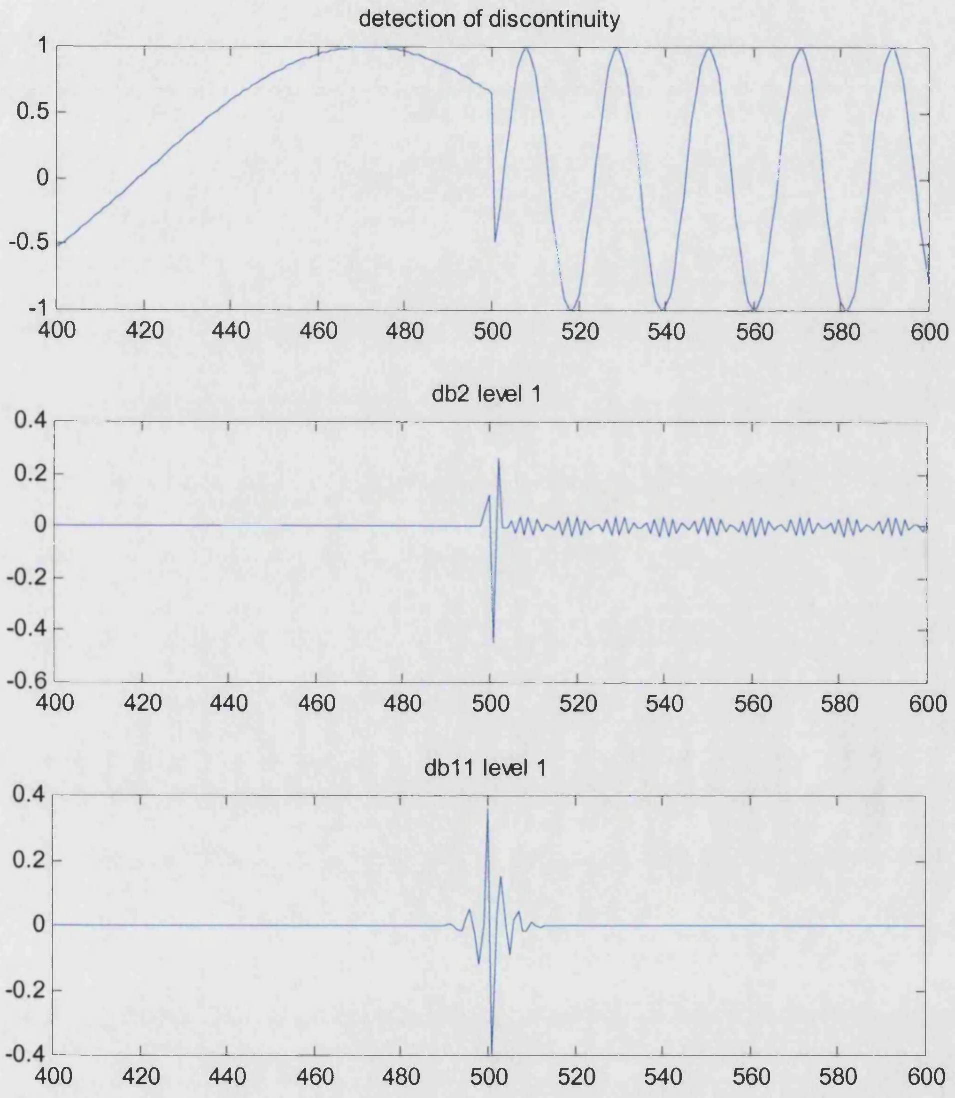


Figure 29 Detection of discontinuity

Wavelets can be used to detect the changes within a waveform. An abrupt change, such as a discontinuity, contains localised, high frequency components. These high frequency components can be detected by using a narrow time window high pass filter such as the db2 wavelet. At level 1 decomposition, the frequency band is a quarter of the sample rate (0.25 radians), and the discontinuity will exist in this upper frequency band.

As the size of the time window is increased (either by choosing a higher wavelet or by using a lower level of decomposition) the frequency range is lowered. This can be used to 'fine tune' the detection of the discontinuity.

Figure 29 shows two sine waves joined in time with a discontinuity at the join. The wavelet analysis was performed with two wavelets, db2 and db11, to show the effect on the detection of using different wavelets. Figure 30 shows detection of a step change.

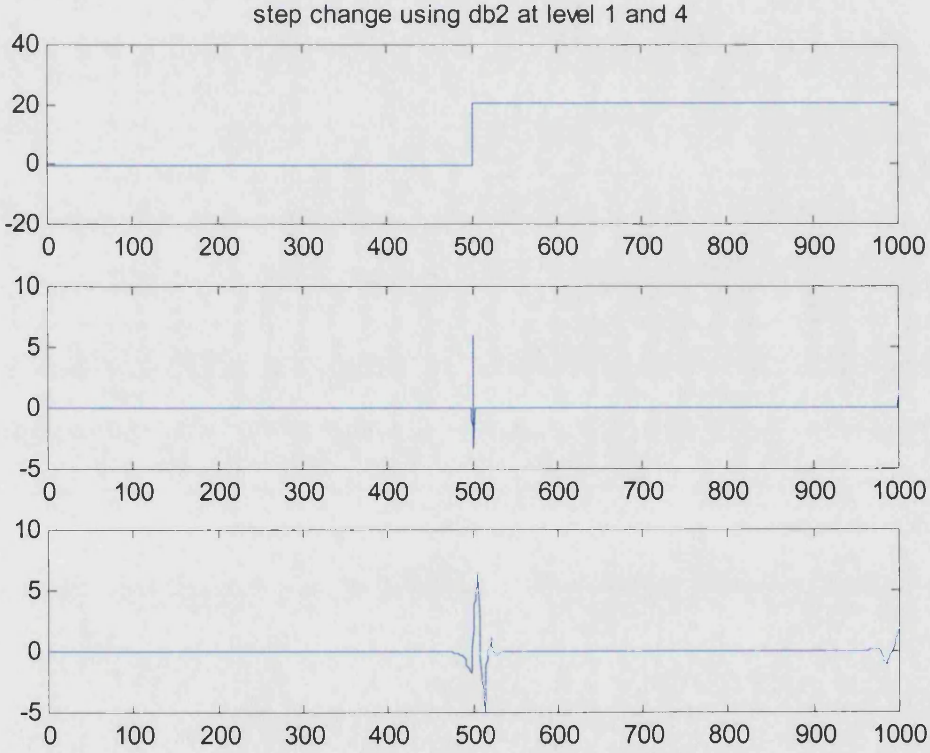


Figure 30 Detection of step change

5.9 Discussion

The aims of this Chapter were to explore the techniques needed to approximate the non-deterministic signals produced by the Mk2 sensors. The information was organised so that the techniques for examining deterministic, stationary signals could be modified to suit the requirements for non-deterministic, non-stationary signals. This involved 4 stages.

- Stage 1 examined the use of global basis functions as a method for approximation. It concluded that the standard methods using global basis functions, Fourier analysis and the short time Fourier transform, were inadequate for approximating non-stationary signals.
- Stage 2 introduced the concept of local bases, which enable the possibility of approximating non-stationary signals.
- Stage 3 detailed the requirements of the signal processing system being developed, and showed that filters arranged in banks using multi resolution analysis provide an ideal method for viewing the non-deterministic, non-stationary signals in the time-frequency domain.
- Stage 4 introduced the concepts of wavelets. These are used as the analysis and synthesis filters in the MRA filter banks. The different types of wavelets were discussed and their properties highlighted. An application of using wavelets in the detection of discontinuities was shown.

The conclusion from this Chapter was that local basis functions can provide a method of analysing and approximating non-deterministic, non-stationary signals.

6 FEATURE EXTRACTION FROM ENGINE DATA

6.1 Introduction

This Chapter presents a series of results taken from the application of a set of wavelets to the Mk 2 sensor design, described in section 3.5. The purpose of this is to show how a series of coefficients can be developed, which display the time – frequency characteristic needed to analyse the non-stationary signals produced by the Mk 2 sensor. The fundamental reason for using the time-frequency domain was that it allowed individual characteristics of the signal to be analysed and approximated. This was achieved through:- (1) the selection of the local basis – the wavelet series, (2) the starting wavelet and (3) the decomposition level.

- The choice of local basis was based upon the characteristics of the original signal. Section 5.7.4 established that the Daubechies wavelets had a maximum number of zero crossings for the minimum number of coefficients, and this offered good performance in analysing signals which exhibited polynomial behaviour. The signals from the Mk 2 sensor showed a high degree of oscillatory behaviour and were well matched to the Daubechies wavelets.
- The starting wavelet – sometimes called the 'mother' wavelet - provided the minimum time window for the analysis. This time window was the number of terms in the FIR realisation of the wavelet. There are a number of Daubechies wavelets db1 to db N. However, the first 10 proved to be the most useful. It must be noted that as the number of terms in the FIR filter increases, from 2 for db1 to 20 for db10, the frequency window narrows accordingly. Thus a db2 wavelet has a narrow time window but a wide frequency window, and a db10 has a wide time window and a narrow frequency window.
- The final stage in the selection of the wavelet was the decomposition level. At a level of 1, 2x sub-sampling of the signal has taken place. Thus the application of the wavelet filter has split the original into two frequency bands – high and

With these three parameters the signal can be decomposed into multiple time-frequency spaces, and the characteristics of the signal which may be obscured in one view can be extracted from another.

This Chapter has been organised to follow the decomposition of one set of data from MK 2 sensor by a series of Daubechies, high pass wavelets. The wavelets start with db2 and end with db5, and for each wavelet, 4 levels of decomposition are shown. Thus there are 16 graphs which show the effect of decomposing the signal from the Mk 2 sensor. Table 6, shows the Figure reference and the page location of the results based upon the wavelet number and the level of decomposition. It also shows the frequency and window size for the signal sampled at 52.5kHz (19.04 μS)

	db2 (4 samples)	db3 (6 samples)	db4 (8 samples)	db5 (10 samples)
Level 1	Figure 32, pp82 **13-26kHz,** ***76 μS***	Figure 36, pp86 **13-26kHz,** ***114 μS***	Figure 40, pp90 **13-26kHz,** ***152 μS***	Figure 44, pp94 **13-26kHz,** ***190 μS***
Level 2 (Divide by 2)	Figure 33, pp83 **7.5-13kHz,** ***152 μS***	Figure 37, pp87 **7.5-13kHz,** ***228 μS***	Figure 41, pp91 **7.5-13kHz,** ***304 μS***	Figure 45, pp95 **7.5-13kHz,** ***380 μS***
Level 3 (Divide by 4)	Figure 34, pp84 **3.75-7.5kHz,** ***304 μS***	Figure 38, pp88 **3.75-7.5kHz,** ***457 μS***	Figure 42, pp92 **3.75-7.5kHz,** ***609 μS***	Figure 46, pp96 **3.75-7.5kHz,** ***761 μS***
Level 4 (Divide by 8)	Figure 35, pp85 **1.87-3.75kHz,** ***609 μS***	Figure 39, pp89 **1.87-3.75kHz,** ***914 μS***	Figure 43, pp93 **1.87-3.75kHz,** ***1219 μS***	Figure 47, pp97 **1.87-3.75kHz,** ***1523 μS***

Table 6 Arrangement of results

In sections 6.2 to 6.5 the Mk2 sensor data is decomposed into a series of frequency bands using different Daubechies wavelets. Each wavelet chosen has two components – the details and the approximations, which come from the action of the high pass and low pass MRA filters (see section 5.6). In sections 6.2 to 6.5 the decompositions shown are the results from the high pass MRA filters. This is because the low pass filter is used for the next level of decomposition. However, if there is no more decomposition then there is a result from the approximation wavelet, which can be viewed. This low pass approximation contains the components of the signal from 0 Hz to the lowest frequency in the decomposition. Figure 31 shows a decomposition tree for Daubechies Db2 wavelet, which illustrates this relationship. The mother wavelet is shown at the top and each branch is a decomposition level.

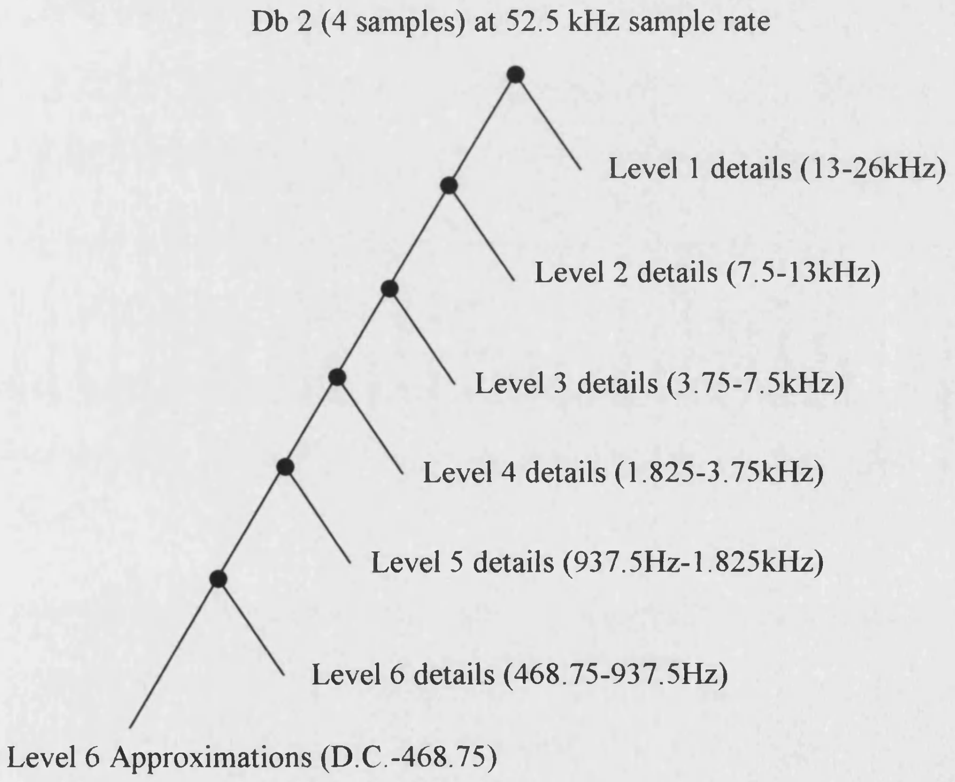

Db 2 (4 samples) at 52.5 kHz sample rate
Level 1 details (13-26kHz)
Level 2 details (7.5-13kHz)
Level 3 details (3.75-7.5kHz)
Level 4 details (1.825-3.75kHz)
Level 5 details (937.5Hz-1.825kHz)
Level 6 details (468.75-937.5Hz)
Level 6 Approximations (D.C.-468.75)

Figure 31 Decomposition tree for Db2 wavelet

6.2 Analysis of data using Db2 detail (high pass) wavelets

6.2.1 Db 2 wavelets at level 1 decomposition

In Figure 32, the db2 wavelet filter at level 1 produces a short time window of ***76 μS*** (4 samples), and a large frequency window from **13kHz to 26kHz**, i.e. the upper half of the spectrum of the sampled waveform. The output from the filter has identified some high frequency components, which mainly occur at around 12 degrees.

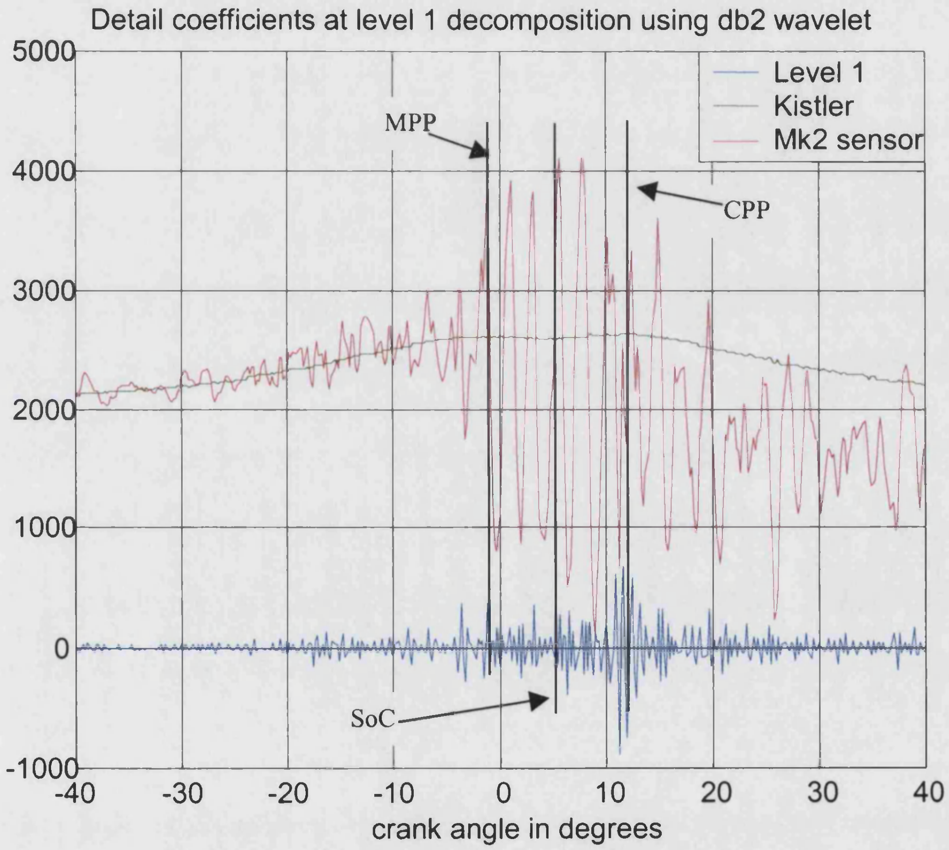


Figure 32 Detail coefficients at level 1, using db2 wavelet

6.2.2 Db 2 wavelets at level 2 decomposition

In Figure 33, the db2 wavelet filter at level 2 produces a time window of ***152 μS*** (8 samples) and a frequency window of **7.5-13kHz**. This corresponds to the next frequency band down from level 1 and a time window of double the size. The output clearly shows the change in signal content at MPP. This corresponds to an increase in signal strength for this frequency band at MPP. The energy of the signal remains high whilst the oscillations die down.

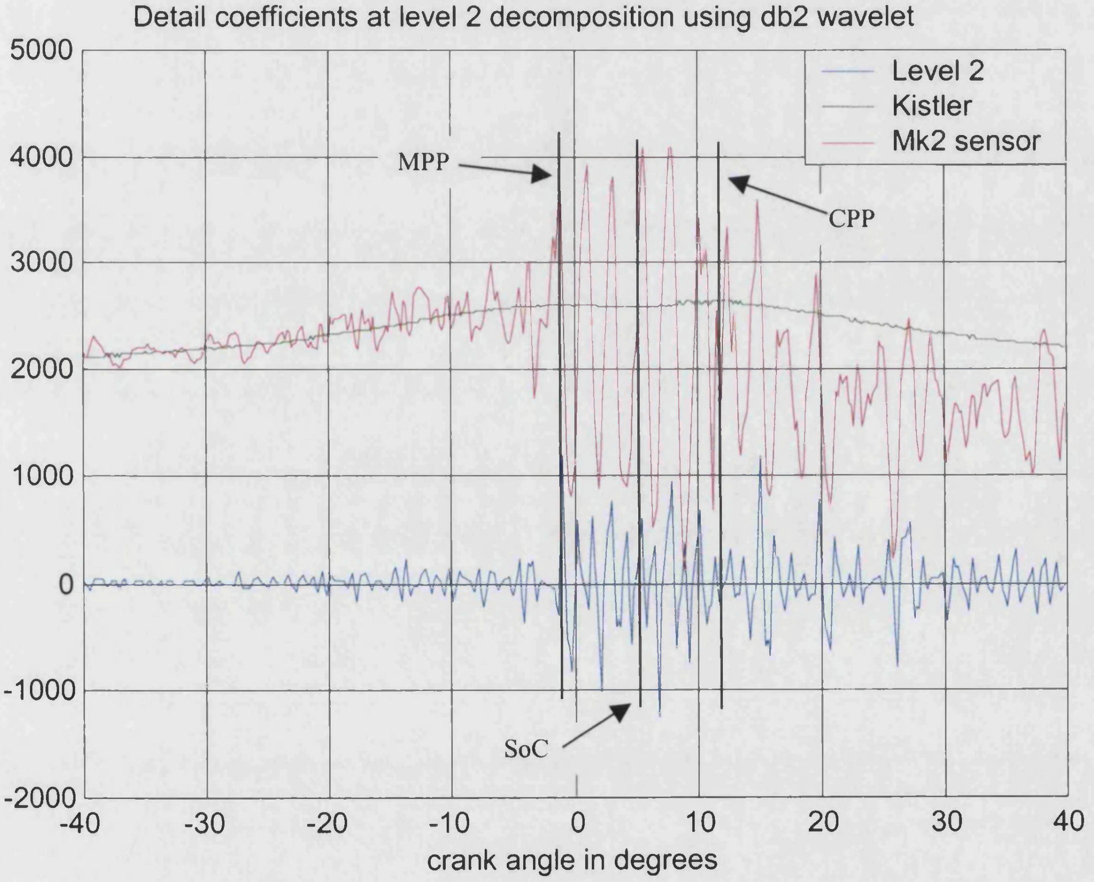


Figure 33 Detail coefficients at level 2, using db2 wavelet

6.2.3 Db 2 wavelets at level 3 decomposition

In Figure 34, the db2 wavelet filter at level 3 produces a time window of ***304 μS*** (12 samples) and a frequency window of **3.75-7.5kHz**. This corresponds to the next frequency band down from level 2 and a time window of double the size. Again the output clearly shows the change in signal content at TDC. This corresponds to an increase in signal strength for this frequency band at TDC.

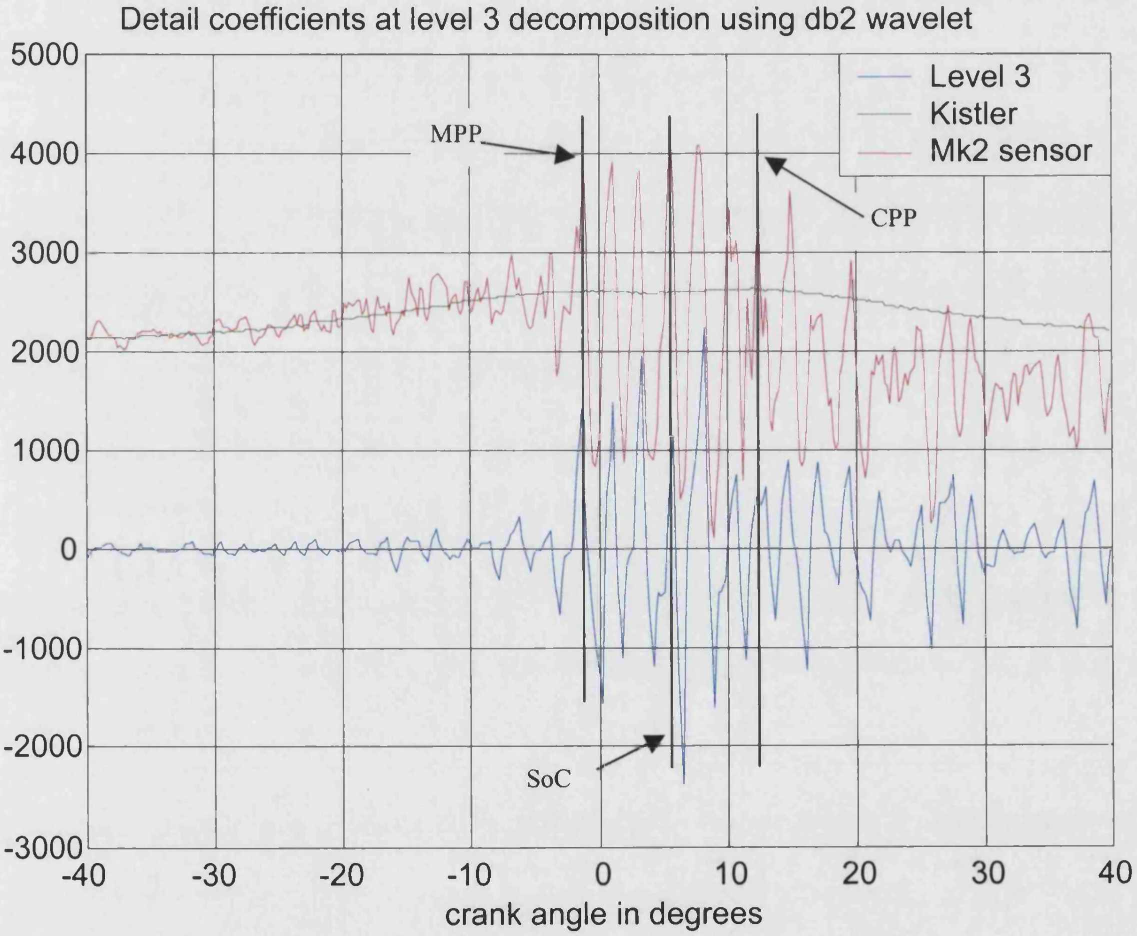


Figure 34 Detail coefficients at level 3, using db2 wavelet

6.2.4 Db 2 wavelets at level 4 decomposition

Figure 35 uses a level 4 decomposition of the db2 wavelet, which corresponds to a time window of ***609 μS*** (16 samples) and a frequency window of **1.87-3.75kHz**. Again there is a change in the energy content at MPP but the change is not so precise.

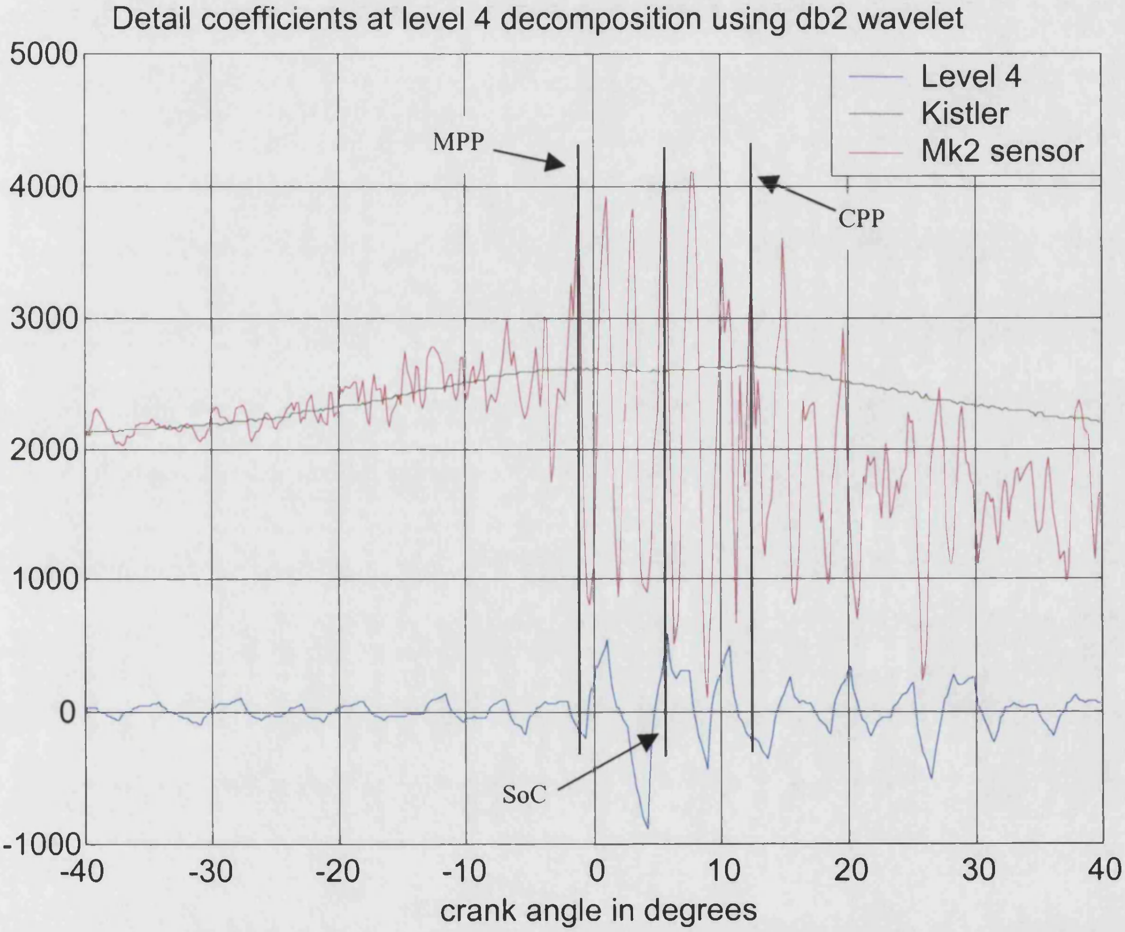


Figure 35 Detail coefficients at level 4, using db2 wavelet

6.3 Analysis of data using Db3 detail (high pass) wavelets

Using the db3 wavelet, the initial time window is ***114 µS*** (6 samples). Thus in the level 1 decomposition, although the spectrum of the filter spans the same frequency region as the db2 wavelet (**13-26kHz**), the time window differs.

6.3.1 Db 3 wavelets at level 1 decomposition

In Figure 36, the features are similar to the output of the db2 wavelet filters at level 1. This is as expected since the same frequency range is covered (**13-26kHz**). However, the output waveform does not contain as many high frequency peaks. This is due to the longer time window affecting features of 6 samples and under.

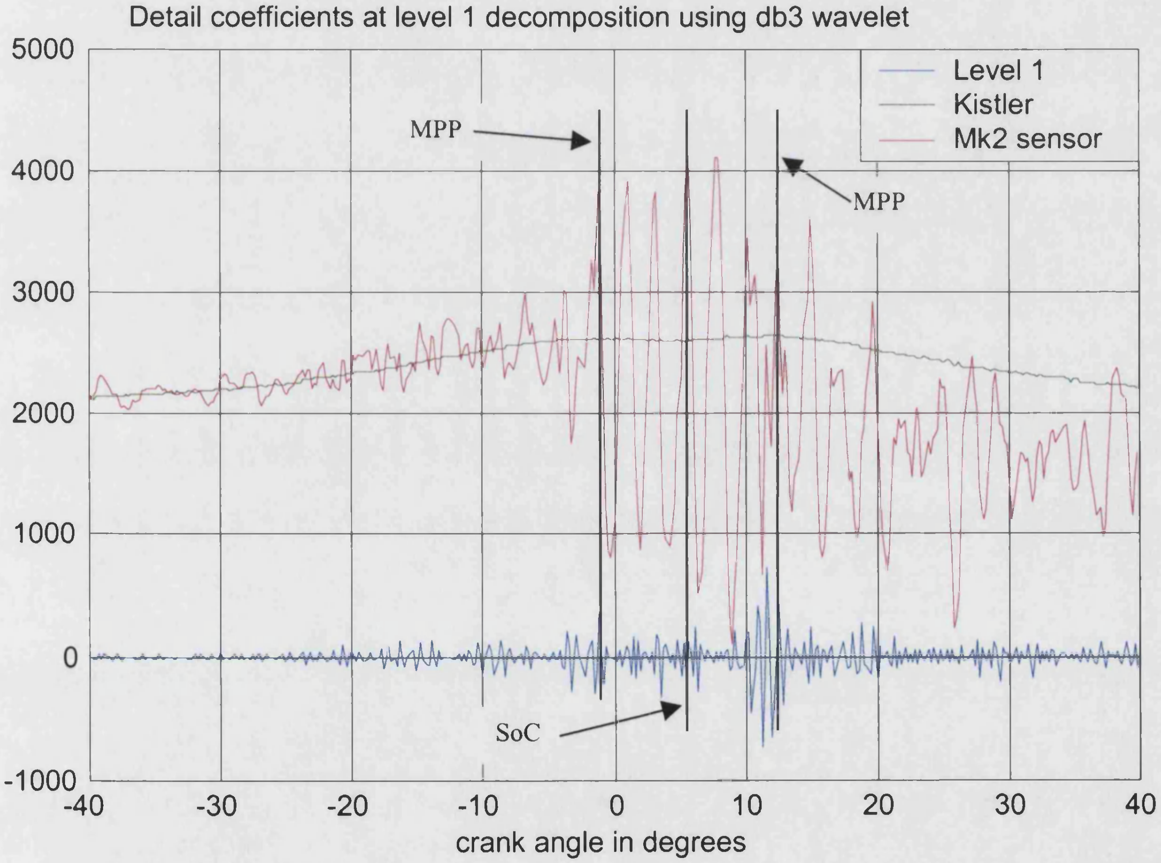


Figure 36 Detail coefficients at level 1, using db3 wavelet

6.3.2 Db 3 wavelets at level 2 decomposition

Figure 37, shows the decomposition performed at level 2 using the db3 filter, again using a frequency range of **7.5-13kHz**, which is the same as level 2, db2. However, the size of the time window is now ***228 μS***, which is the same as the level 3, db2 filter. This produces an output which is confusing and does not enable MPP to be identified.

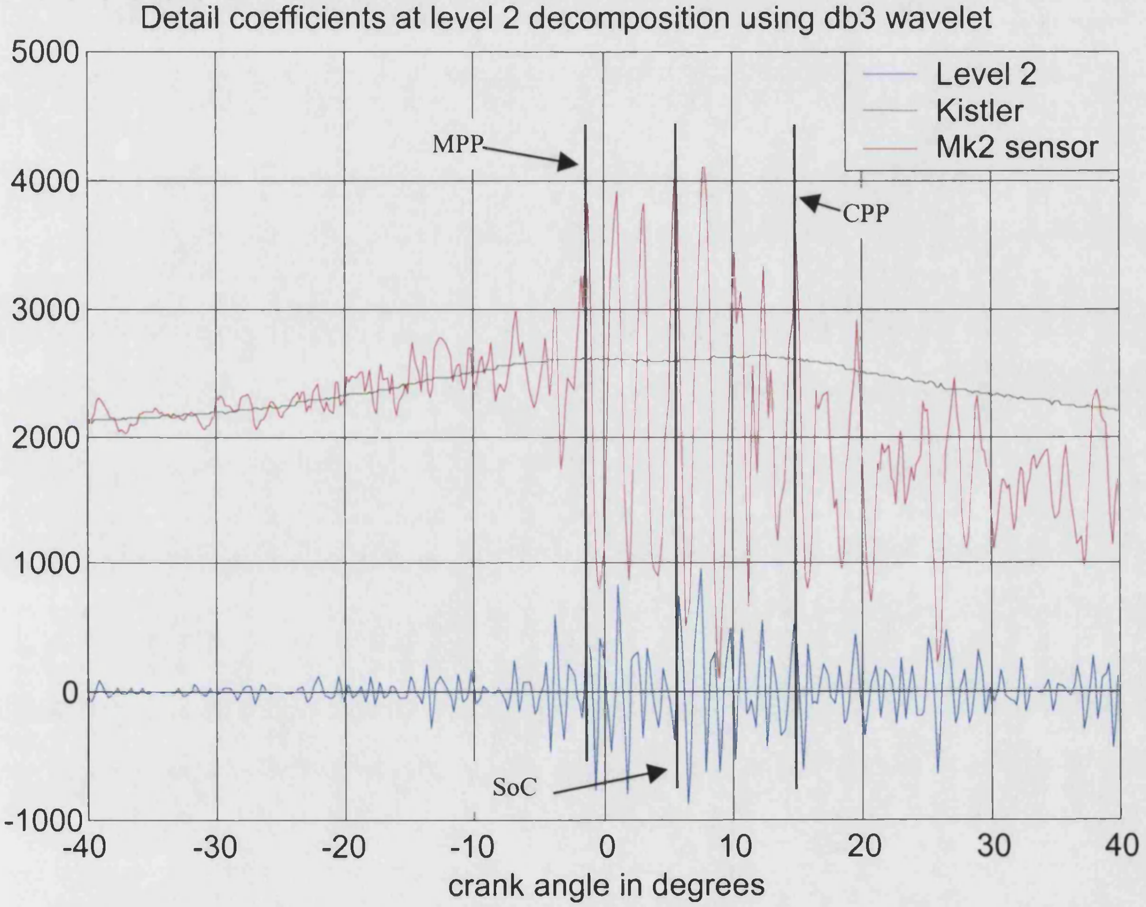


Figure 37 Detail coefficients at level 2, using db3 wavelet

6.3.3 Db 3 wavelets at level 3 decomposition

In Figure 38, the time window is now ***457 μS*** (18 samples) and the frequency window **3.75-7.5kHz** is the same as level 3, db2. The output of the filter clearly shows that there is significant energy in this frequency band, and that the energy changes significantly at MPP.

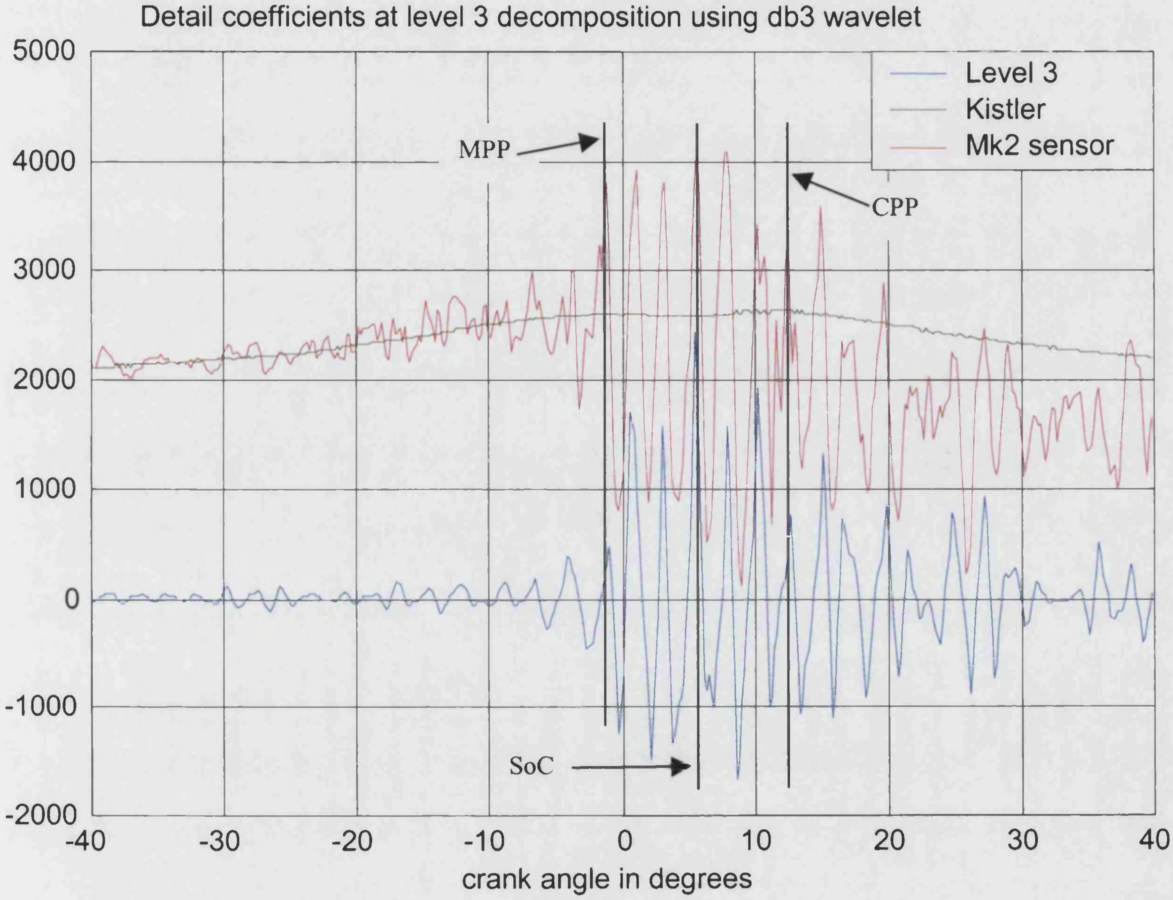


Figure 38 Detail coefficients at level 3, using db3 wavelet

6.3.4 Db 3 wavelets at level 4 decomposition

In Figure 39, the time window is now ***914 μS*** (24 samples) and the frequency window is **1.87-3.75kHz**. The change in energy is not so significant in this output, although it has identified MPP.

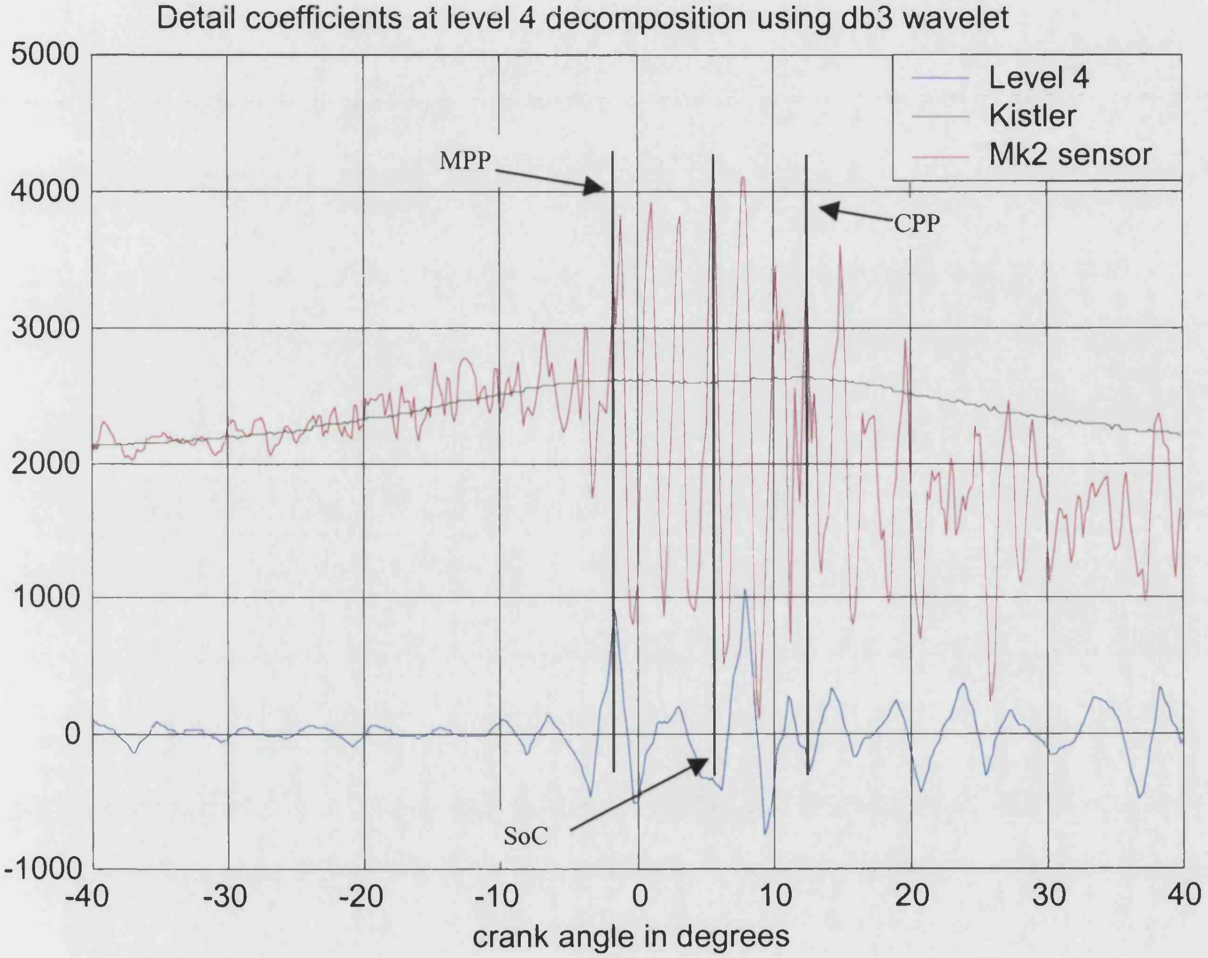


Figure 39 Detail coefficients at level 4, using db3 wavelet

6.4 Analysis of data using Db4 detail (high pass) wavelets

In theory, the db 4 wavelet should offer a similar result to the db2 wavelet, since the time windows of the db4 is twice as big as the db2. However, although the time window is a multiple of db2, the frequency window is the same as db2.

6.4.1 Db 4 wavelets at level 1 decomposition

In Figure 40, the time window for the filter is ***152 μS*** (8 samples) and the frequency window is **13-26kHz.** Again the high frequency content of the waveform does not contain significant energy.

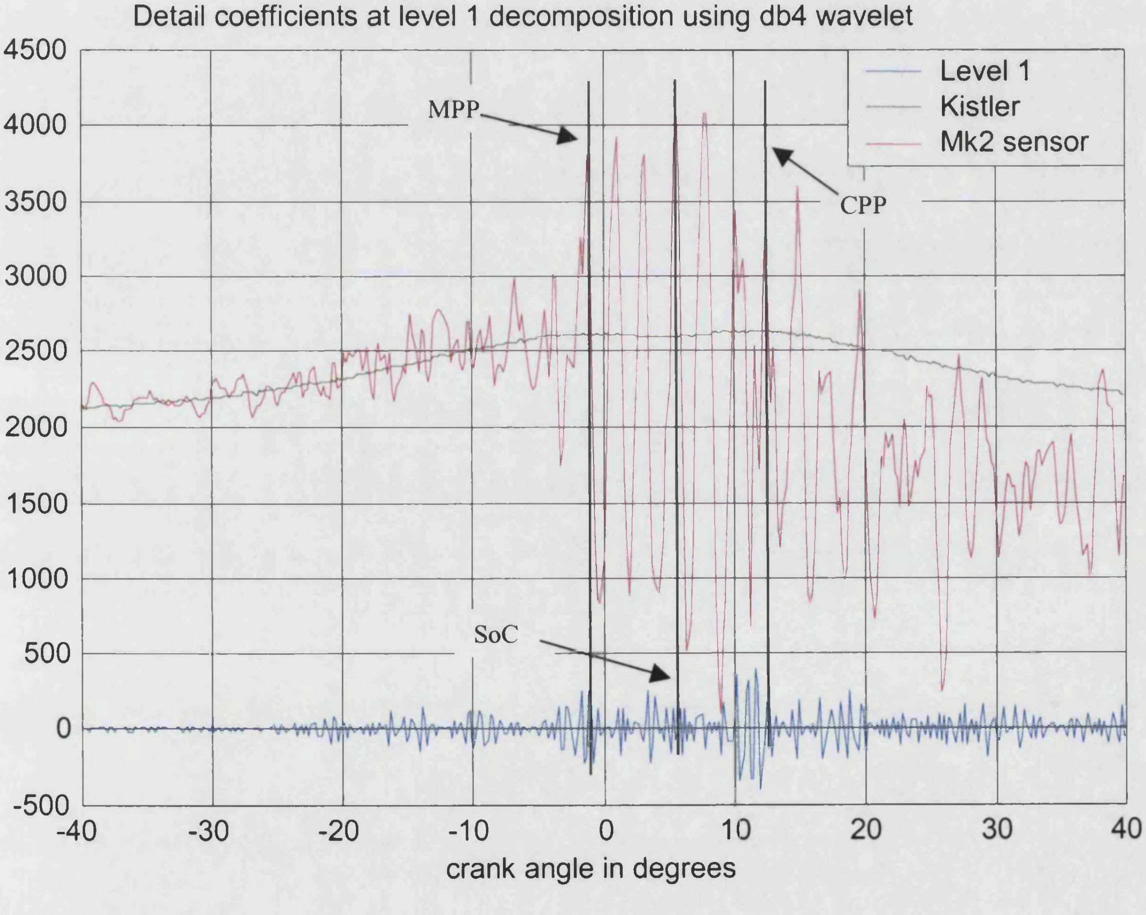


Figure 40 Detail coefficients at level 1, using db4 wavelet

6.4.2 Db 4 wavelets at level 2 decomposition

In Figure 41, the time window is ***304 µS*** (16 samples) and the frequency window is **7.5-13kHz**. Again there is significant energy in this frequency band. However, the distinction between no-combustion (before MPP) and combustion (after MPP) is not so clear.

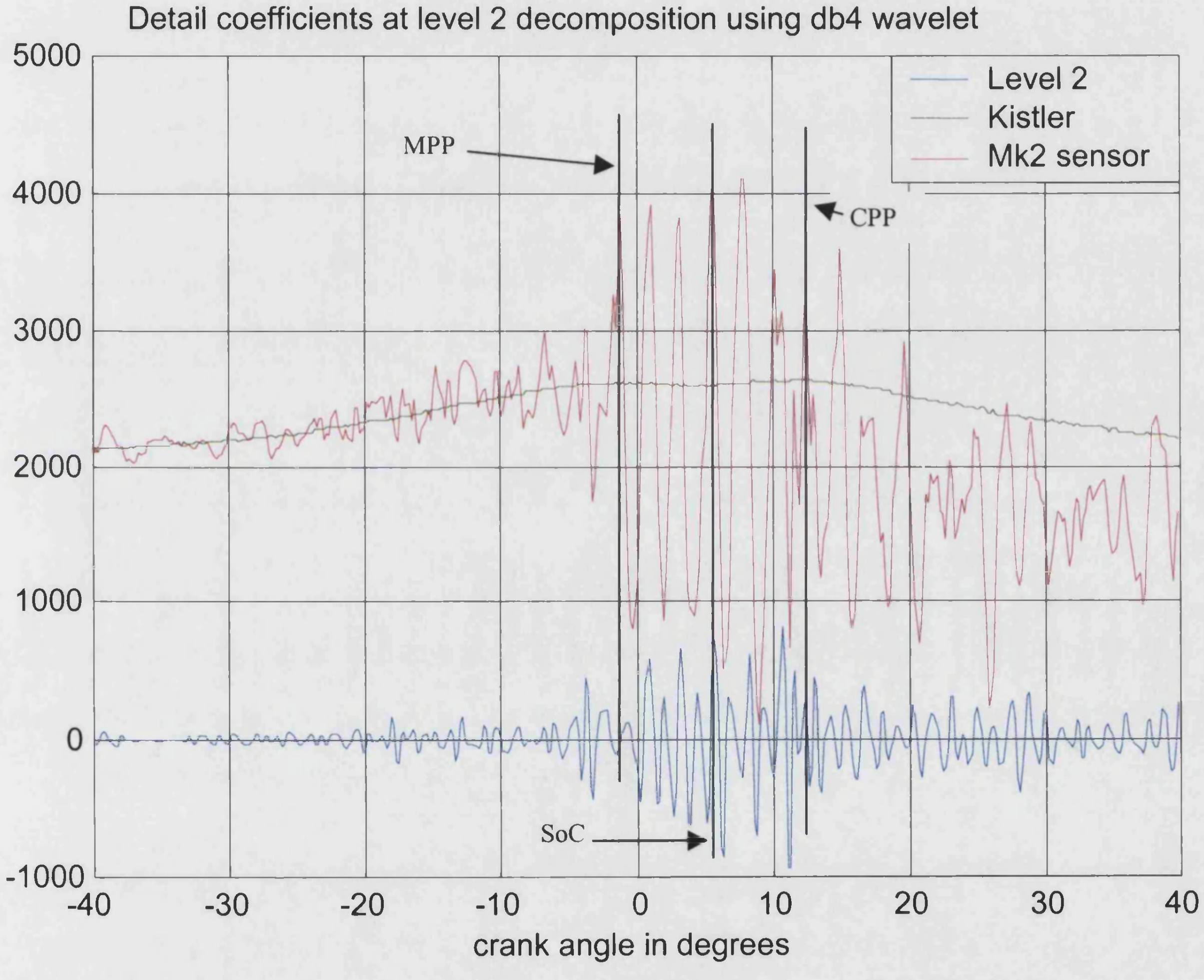


Figure 41 Detail coefficients at level 2, using db4 wavelet

6.4.3 Db 4 wavelets at level 3 decomposition

In Figure 42, with a time window of ***609 μS*** (24 samples) and a frequency window of **3.75-7.5kHz,** the change in energy at TDC is clearly shown. The smoothing effect of the longer time window can also been seen.

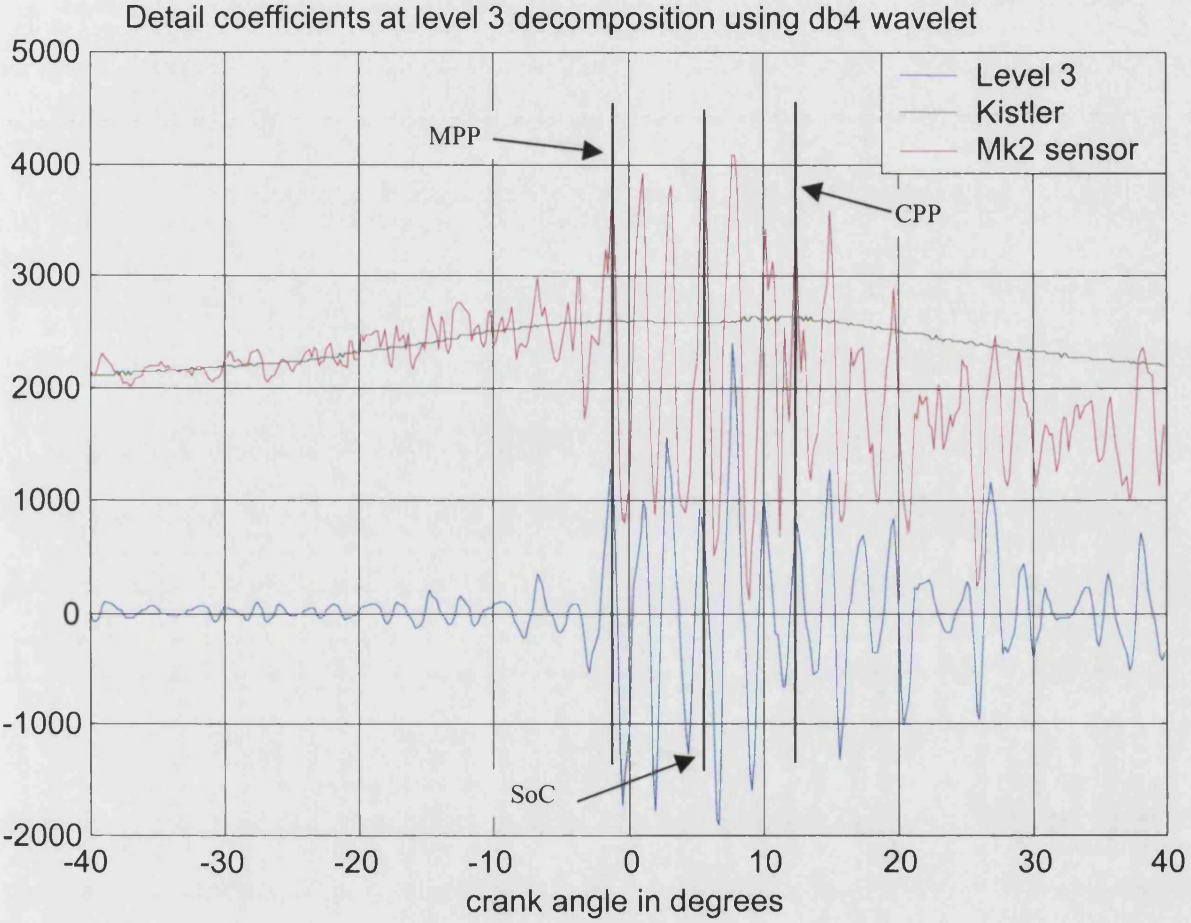


Figure 42 Detail coefficients at level 3, using db4 wavelet

6.4.4 Db 4 wavelets at level 4 decomposition

The filtered signal in Figure 43 shows that very little energy is produced at the frequency of **1.87-3.75kHz**. However, the time window of the filter is quite long at ***1219 µS*** (32 terms), and so features of 32 samples and under will be affected by smoothing.

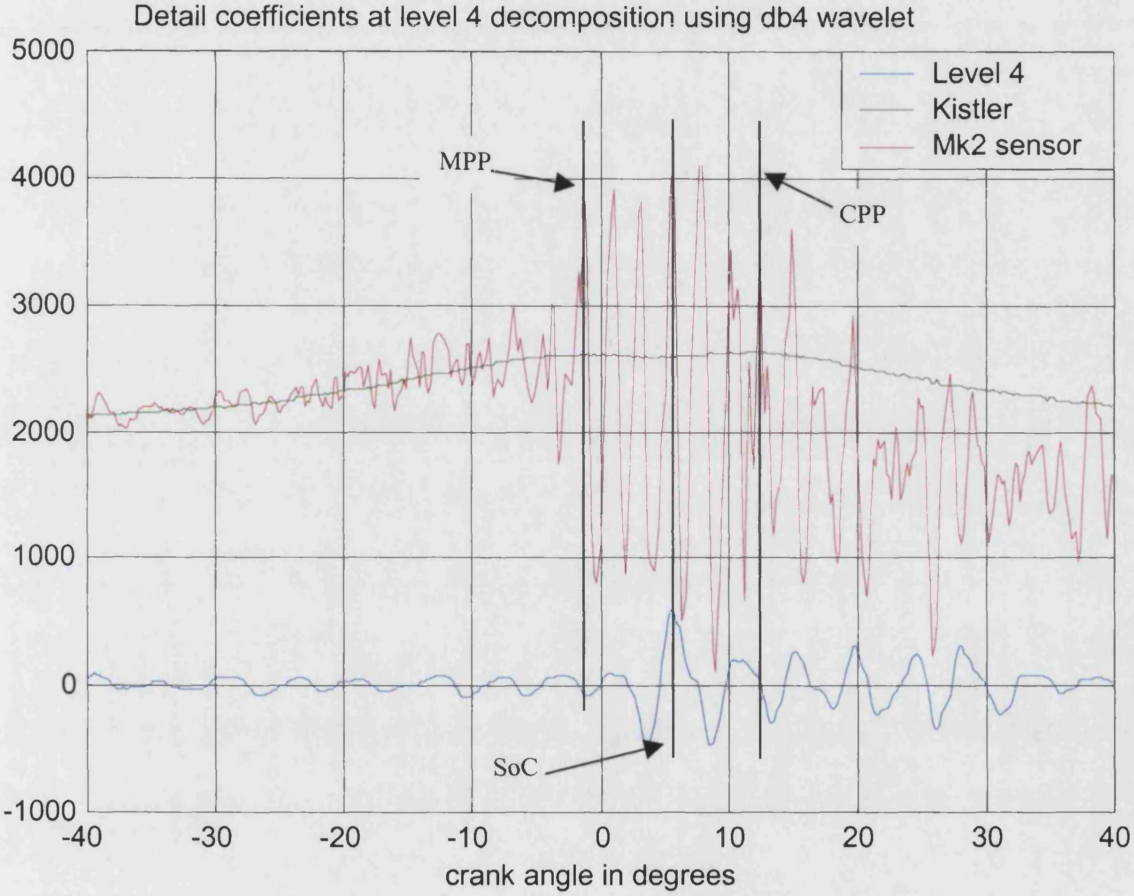


Figure 43 Detail coefficients at level 4, using db4 wavelet

6.5 Analysis of data using Db5 detail (high pass) wavelets

Db5 wavelets start with a time window of ***109 μS*** (10 samples). With this size of window the filter will not be able separate any energy from the original signal at the higher frequency bands. Signals with feature sizes lower than the 10 samples, will be adversely affected by the time window of the filter.

6.5.1 Db 5 wavelets at level 1 decomposition

In Figure 44, no significant energy exists in this frequency band **13-26kHz**, using the db5 wavelet filter and a time window of ***109 μS*** (10 samples).

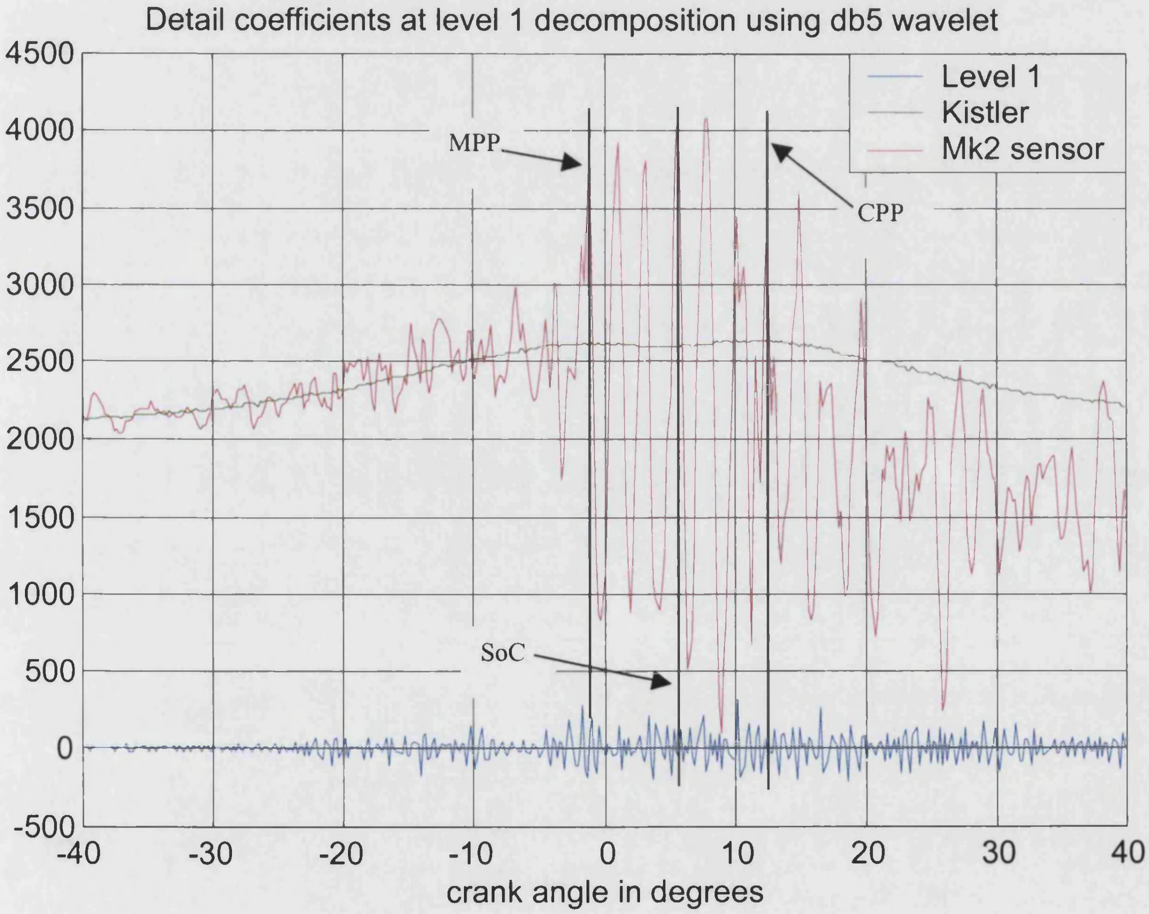


Figure 44 Detail coefficients at level 1, using db5 wavelet

6.5.2 Db 5 wavelets at level 2 decomposition

Figure 45, shows the wavelet coefficients at a time window of ***380 µS*** (20 samples) and a frequency of **7.5-13kHz**.

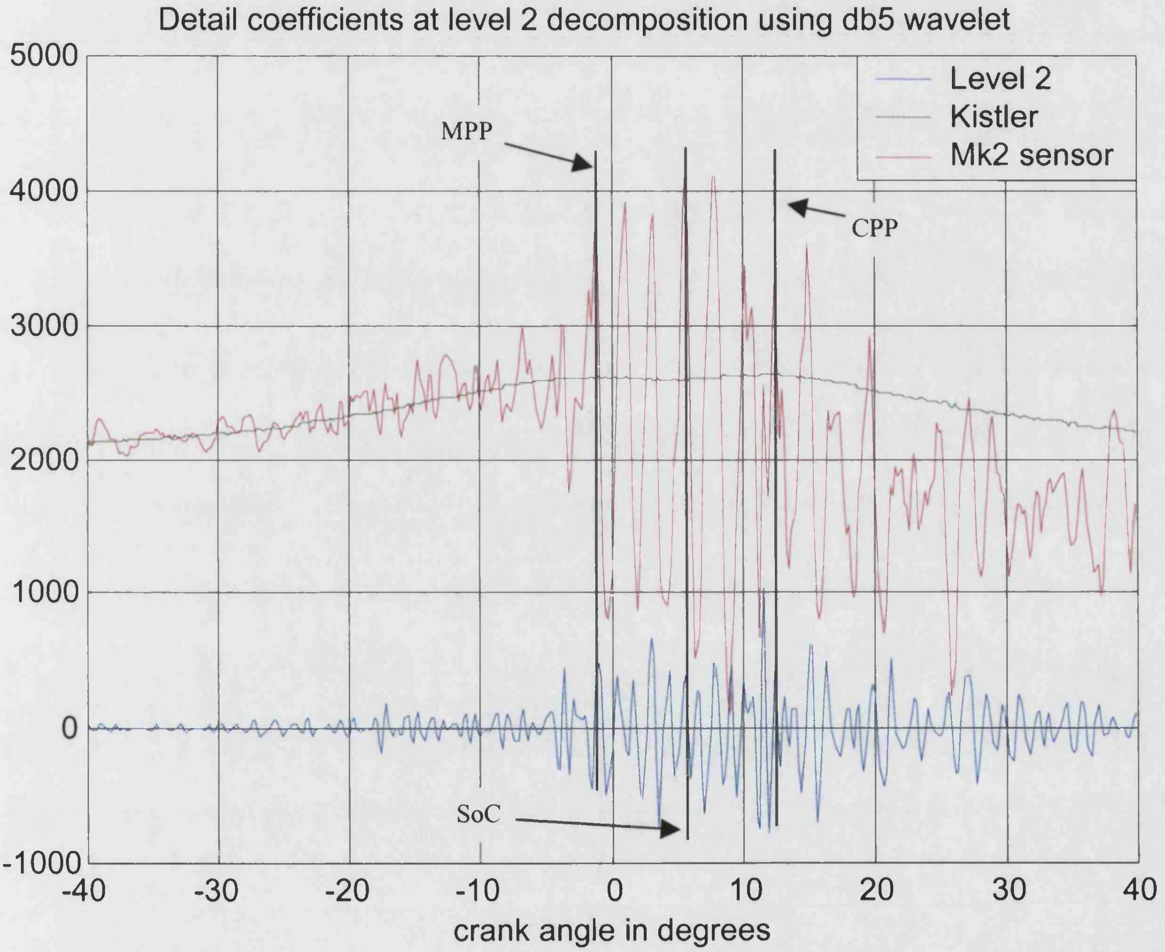


Figure 45 Detail coefficients at level 2, using db5 wavelet

6.5.3 Db 5 wavelets at level 3 decomposition

Figure 46, shows the wavelet coefficients at a time window of ***761 μS*** (30 samples) and a frequency of **1.87-7.5kHz**.

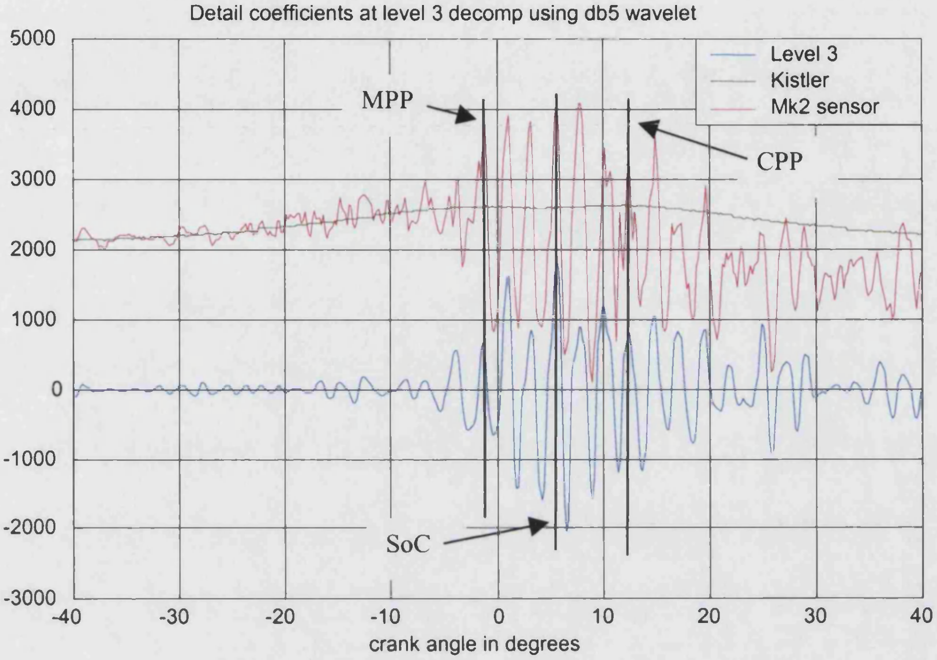


Figure 46 Detail coefficients at level 3, using db5 wavelet

6.5.4 Db 5 wavelets at level 4 decomposition

Figure 47, shows the wavelet coefficients at a time window of ***1523 μS*** (40 samples) and a frequency of **1.87-3.75kHz**.

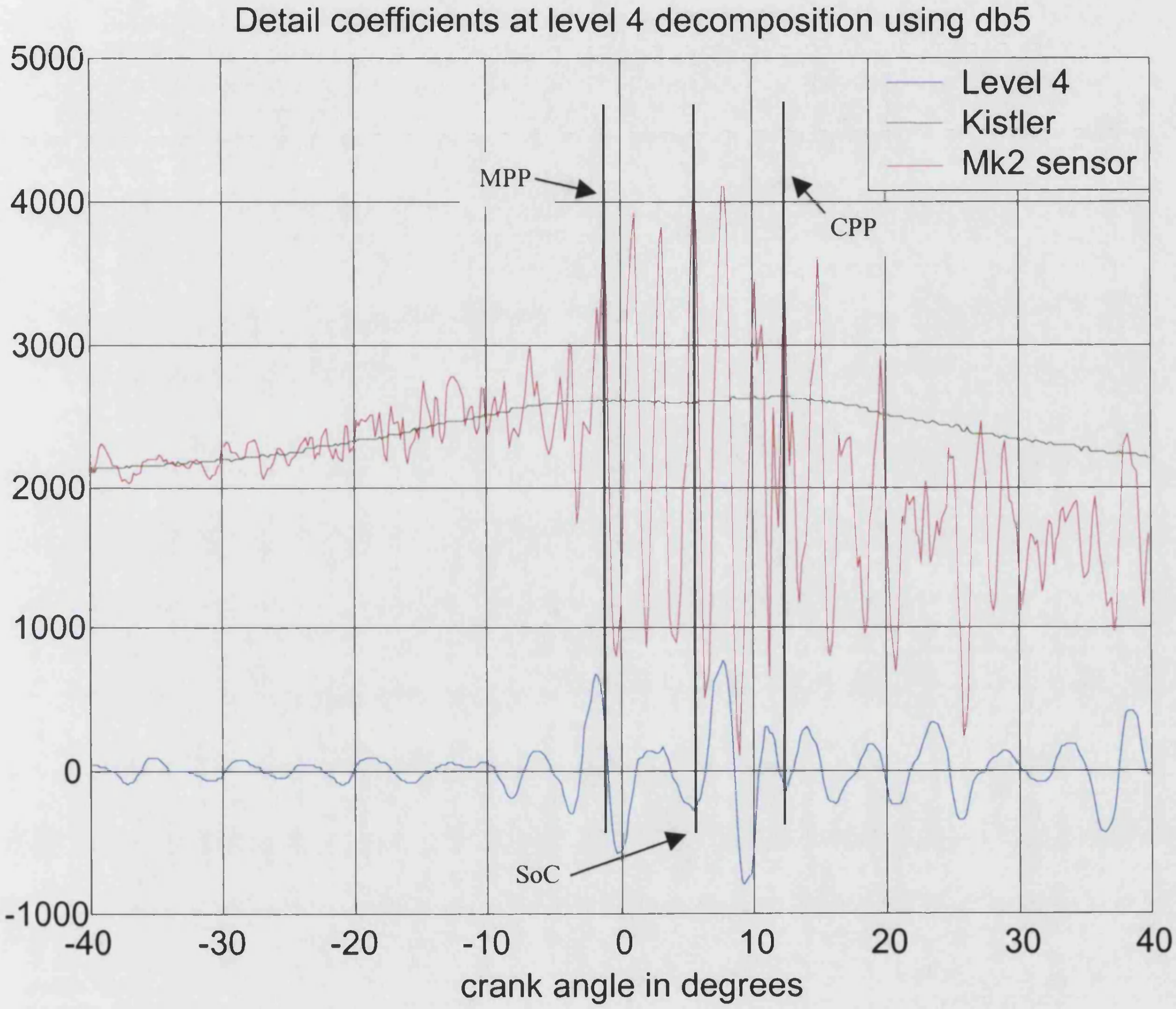


Figure 47 Detail coefficients at level 4, using db5 wavelet

6.6 Analysis of data using Db2 and Db6 analysing wavelets

In Figure 48, the engine is running at 2000rpm; this gives a 60kHz data rate. Thus the approximation coefficients are produced from a time window of 400 μS and a frequency window of D.C. to 468.75Hz

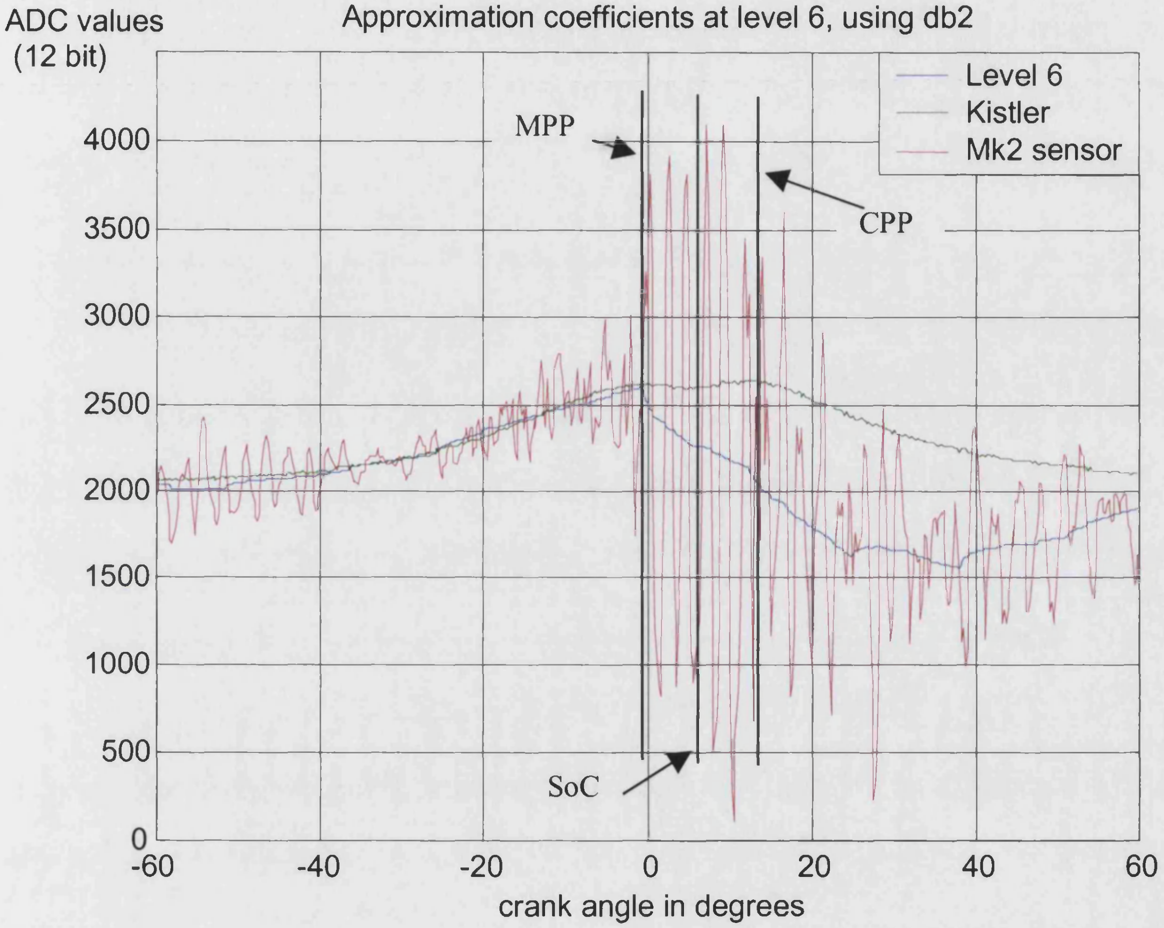


Figure 48 Approximation coefficients at level 6, using db2 wavelet

In Figure 49, the approximation coefficients are produced from the Db 6 wavelet at a level 6 decomposition. This gives a time window of 1200 μS and a frequency window of D.C. to 468.75Hz.

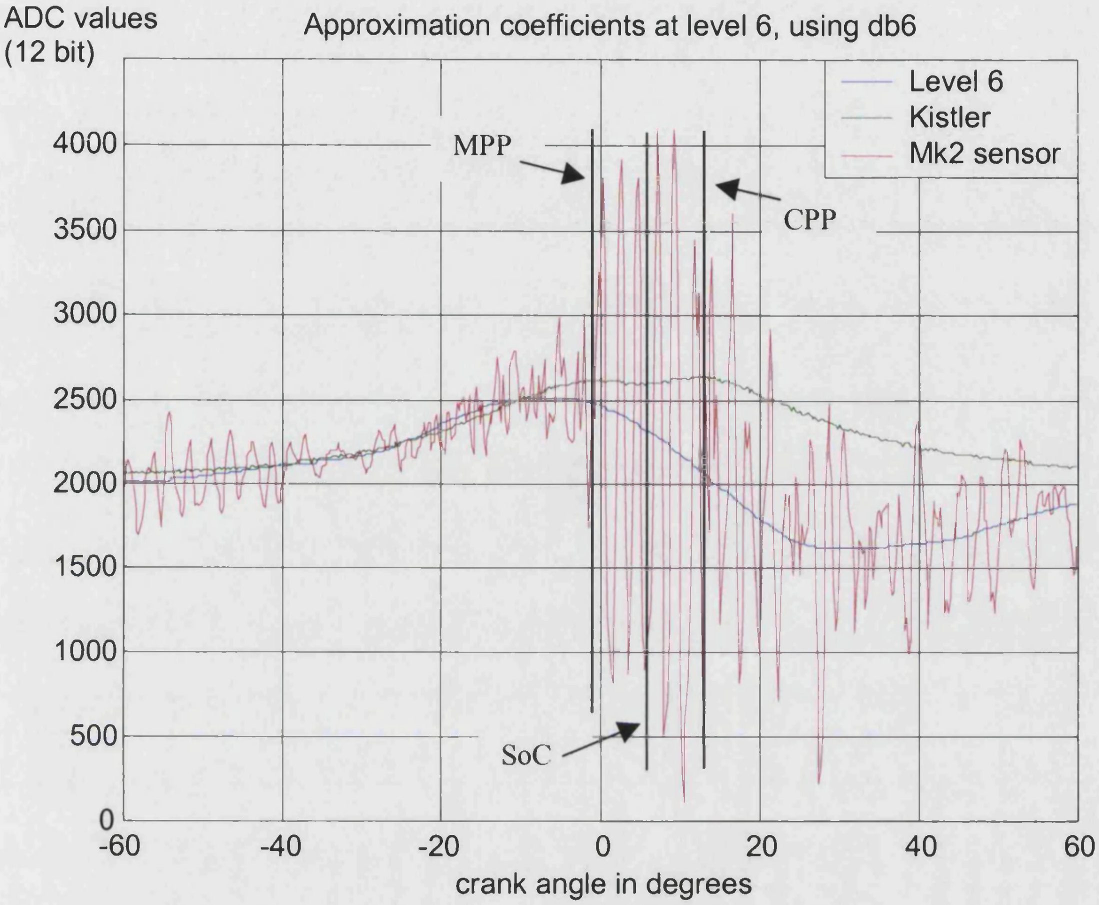


Figure 49 Approximation coefficients at level 6, using db6 wavelet

The conclusions from this is that the approximations are best analysed with a short filter length wavelet, such as Db2 decomposed the to 6th level.

A comparison between a traditional low pass filter such as a 5th order Butterworth filter and the wavelet low pass filter is shown in Figure 50. This shows how the short time window of the wavelet filter enables the trend of the data to be visible without distorting the signal through 'smoothing'. The low pass Butterworth filter is a 5th order, with a 3dB cutoff point at 234 Hz . This corresponds to the level 6 wavelet filter which is set at 0 to 468 Hz.

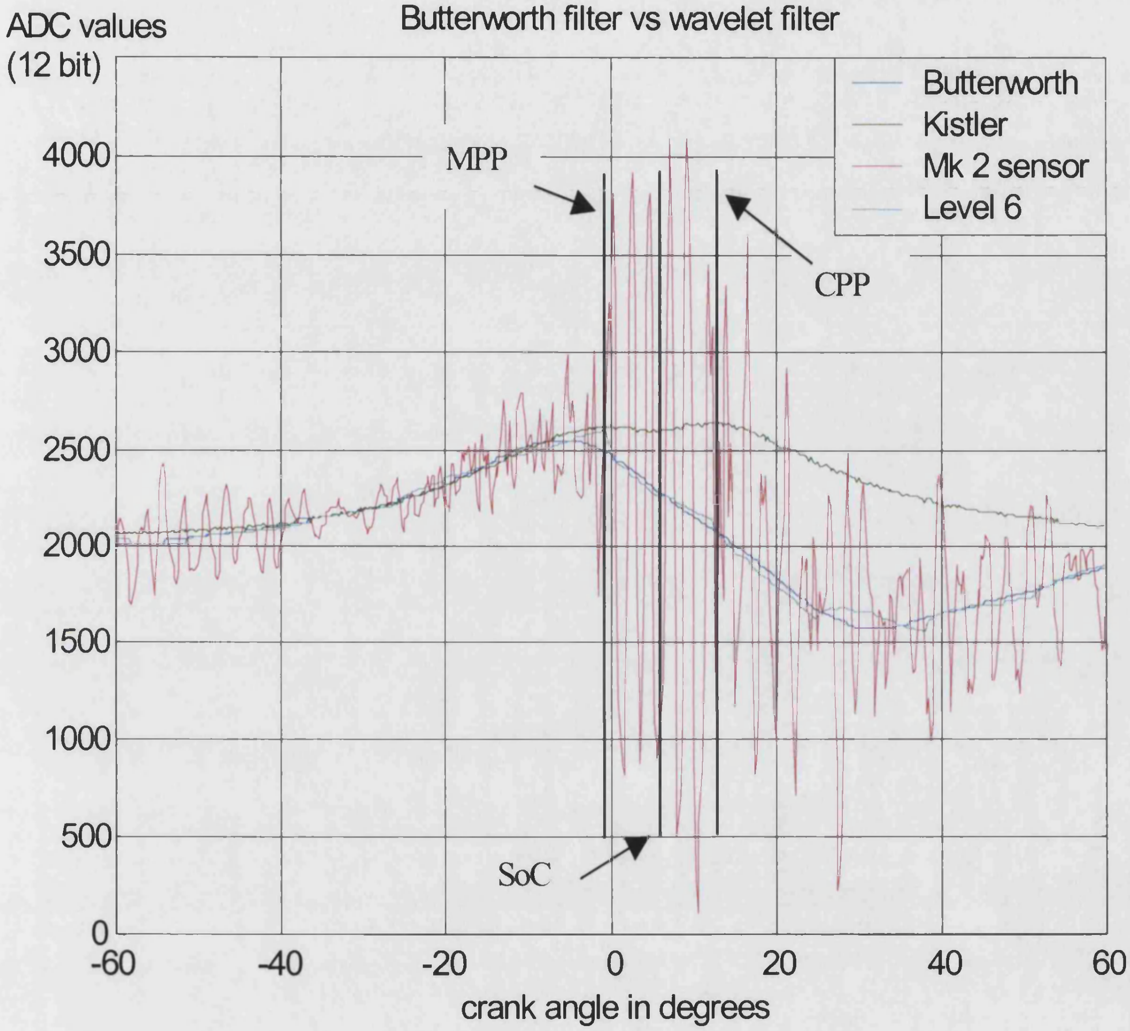


Figure 50 Performance of Butterworth filter compared with wavelet filter

6.7 Discussion

This Chapter presented a series of results taken from the decomposition of the Mk 2 sensor data, described in section 3.5. The decomposition used four wavelets from the Daubechies wavelet series. The choice of these wavelets was based on the need for an accurate detection of the position of motored peak pressure. Wavelets with small time windows offered the best performance at detecting rapid changes in the underlying signal; thus the Daubechies db2 to db4 were chosen for this analysis. The position of motored peak pressure is characterised by a rapid change in the frequency and amplitude of the signal. This frequency was identified in section 3.6 as 4.5kHz. The wavelets which have a frequency span of 3.75 to 7.5 kHz performed well at detecting this change – i.e. level 3, db2, db3 and db4 decompositions. However, as the longer wavelets are used (db5 and greater) so the location of the change of frequency becomes more difficult. This is due to the longer time window causing 'wind up'. Other wavelets, which offered multiples of the 4.5 kHz frequency, could also detect the start of MPP (Db2 level 2, 7.5 to 13kHz).

The fundamental purpose for this Chapter was to demonstrate the ability of wavelets in decomposing a non-stationary, non-deterministic signal into a series of time-frequency coefficients. Once these coefficients had been generated, processing to detect the desired features could proceed.

7 FEATURE DETECTION

7.1 Introduction

This Chapter presents the final stage in the development of a system capable of indicating the start of combustion and the location of peak pressure based on a non-intrusive sensor. In section 3.6 the non-deterministic, non-stationary nature of the signals from the Mk 2 sensor was identified, and the difficulties in locating the start of combustion and location of peak pressure, using traditional techniques, were demonstrated. In Chapter 5 the concepts were developed to enable the analysis of these non-stationary, non-deterministic signals, and wavelets were identified as a method to isolate, in time and frequency, the desired features of the signal. In Chapter 6 this ability of wavelets to isolate in time and frequency was used to analyse the Mk 2 sensor data and the results showed that wavelets perform well in locating peak pressure.

The point with wavelet decomposition is that it is a linear transform. This property will preserve the original identity of the signal, even its non-deterministic nature. This non-deterministic behaviour of the wavelet coefficients leads to difficulty in identifying individual components within the waveform responsible for peak pressure. However, the power of the wavelet decomposition technique stems from the ability to separate time and frequency and to extract the desired features. Once the features have been extracted, then other techniques can be used to tackle the feature detection from the non-deterministic nature of the signal.

This Chapter presents two methods which were used to perform this feature detection: non-linear thresholding and artificial neural networks.

7.2 Noise removal and signal reconstruction

A main advantage of using the Daubechies wavelets is that they enable a loss-less deconstruction and reconstruction of the signal. However, by careful alteration of the wavelet coefficients the reconstructed signal can be adjusted in a non-linear manner. This is the basis for compression and noise removal.

The coefficients generated by wavelet analysis are a representation of the spectrum of the signal at discrete time intervals. If the coefficients are subjected to linear or non-linear scaling, then when the signal is re-combined certain frequencies will be enhanced or removed. The performance of thresholding relies on the fact that noise contaminating the signal is spread across all frequencies with a constant power, and that the signal is concentrated into a narrow range of frequencies with high power. A simple method for achieving the enhancement and removal of certain frequencies is thresholding [43]. In this method all coefficients in a wavelet stream are compared to a threshold value. If the coefficient is less than the threshold it is negatively scaled otherwise it is positively scaled. When reconstruction takes place coefficients of sufficient energy are maintained and others are removed. The technique for removing noise is to place the threshold just above the noise floor for the signal.

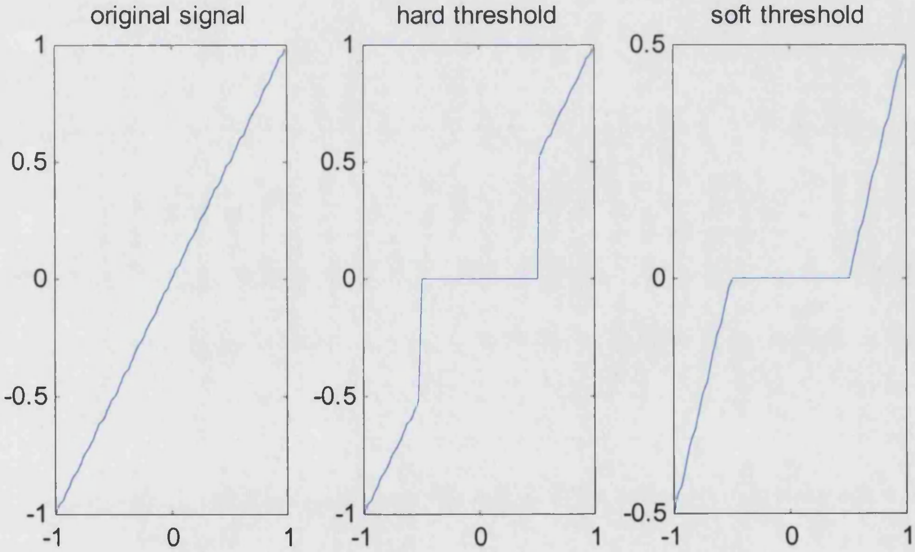


Figure 51 Thresholding of a signal

There are two common types of thresholding. Hard thresholding is achieved by setting all data to zero, which falls below the value and scaling the remainder by a linear factor. Soft thresholding scales all the coefficients by a linear factor. Figure 51 shows the two types of thresholding.

In many standard techniques using wavelets the incoming signal is decomposed by MRA into a set of wavelet coefficients. These coefficients are then passed through a threshold to remove noise. The coefficients are then recombined to produce a de-noised version of the original signal. The threshold values are the key to the success of this technique. There are several methods by which the best threshold values can be obtained, but the manual approach is very simple and effective. Figure 52 shows one of the wavelet signals from the Mk2 sensor undergoing hard thresholding. All values below the line are set to zero, and above the line are kept. The signals of sufficient energy to stay above the line are transferred to the output reconstruction.

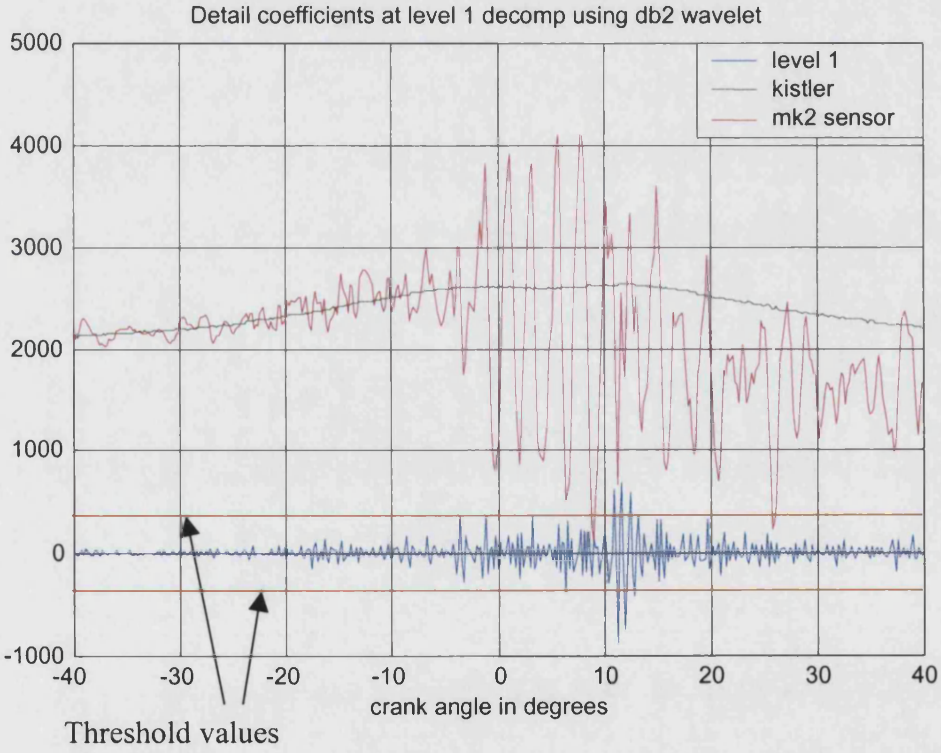


Figure 52 Threshold function of Mk 2 sensor data

Using this technique and setting the threshold values manually, a reconstructed signal can be developed which offers good performance in detecting MPP and SoC. Figure 53 shows the reconstructed graph.

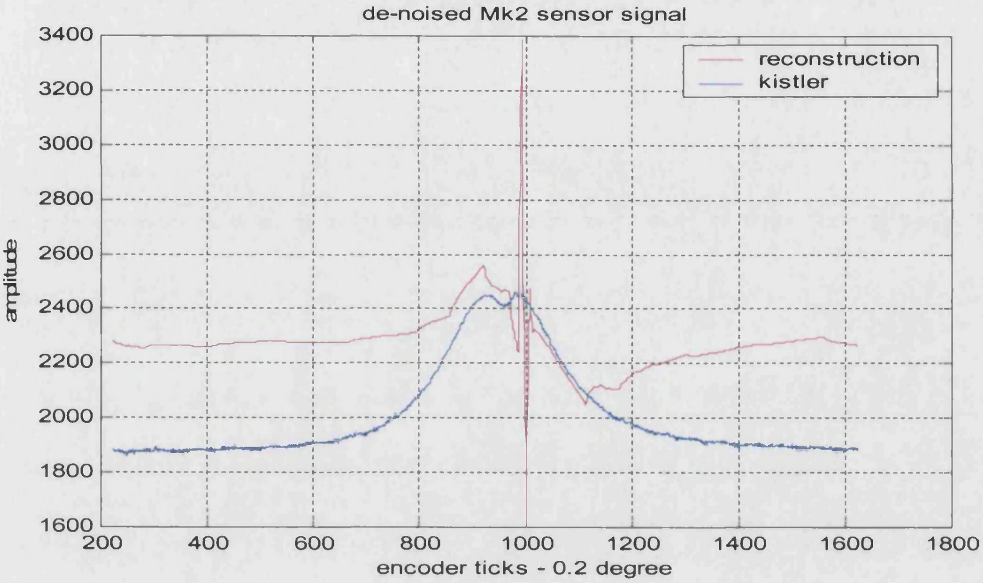


Figure 53 De-noised version of the Mk2 sensor signal

In this reconstruction the aim was to try and detect the MPP. This was achieved. The de-noising has also picked out CPP. A simple level detection could be used to find the maximum points of the waveform and this would give the corresponding crank angle. Figure 54 shows the system for de-noising the signal from the Mk2 sensor.

The main problem with this technique is that the values of the thresholding were manually adjusted until the desired output was achieved. Thus using the manual thresholding this process is difficult to automate. There are automatic estimators for thresholding which are based on the properties of white noise. However, these estimators are good at setting the threshold values to remove Gaussian white noise, but cannot perform so well in the presence of unscaled and non-white noise. In general, the setting of the values for thresholding is a supervised learning process, where the desired output is known and an error signal can be generated.

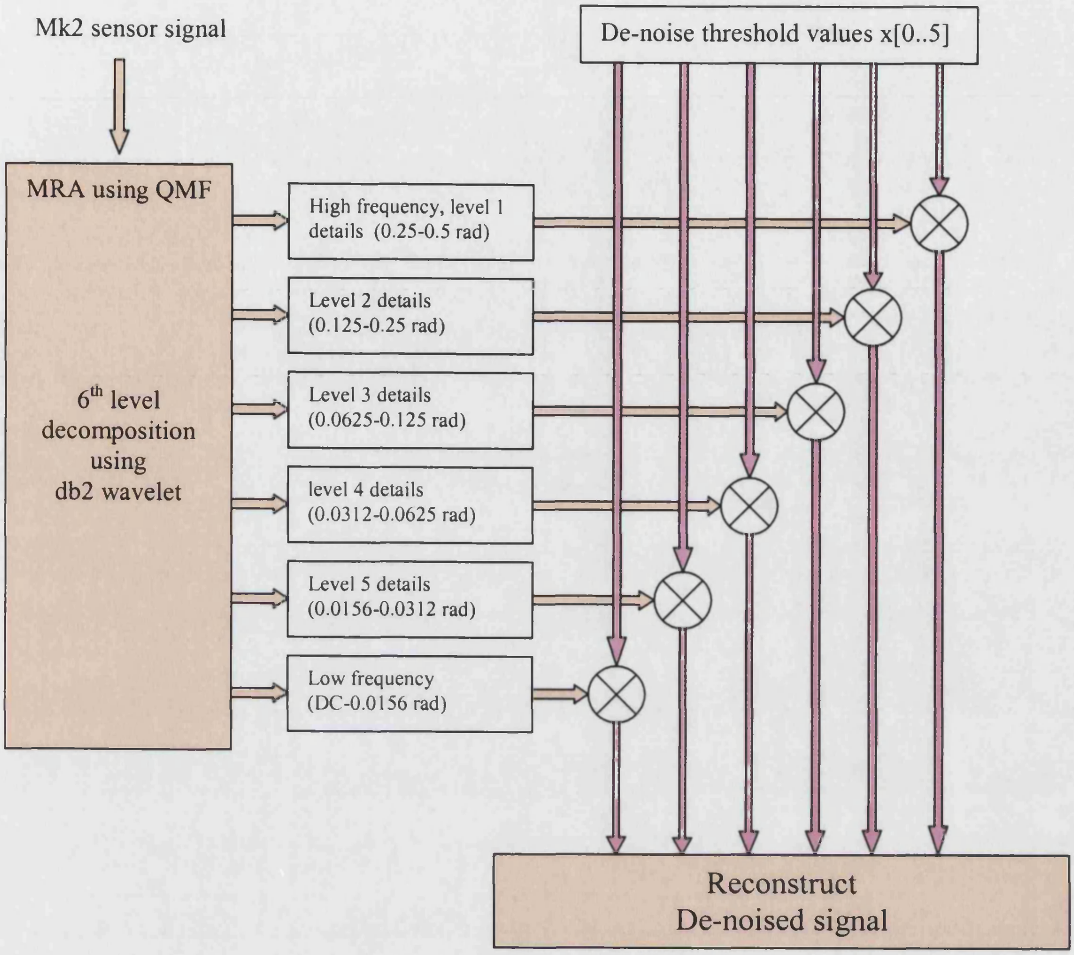

Mk2 sensor signal
De-noise threshold values x[0..5]
MRA using QMF
6th level decomposition using db2 wavelet
High frequency, level 1 details (0.25-0.5 rad)
Level 2 details (0.125-0.25 rad)
Level 3 details (0.0625-0.125 rad)
level 4 details (0.0312-0.0625 rad)
Level 5 details (0.0156-0.0312 rad)
Low frequency (DC-0.0156 rad)
Reconstruct De-noised signal

Figure 54 Thresholding of wavelet coefficients

7.3 Feature detection using neural networks

7.3.1 Introduction

The desired output signal is an indication of the state of the engine. In its simplest case a signal giving compression or combustion is required. The signal together with crank angle would determine the point when MPP has occurred. The start of combustion and CPP signals would follow a similar pattern.

This recognition of when a signal has changed state is a classic problem for a neural network. There are many examples covered in Chapter 2, which show how similar problems from different fields have been solved. The main problem in using a neural network is the selection of input features to use in the network. The following section deals with this.

7.3.2 Selection of input features

In Chapter 2, neural networks were investigated as a possible solution to the problem of feature recognition in non-stationary signals. It highlighted the need for a network to have many input features with which to train. However, in this research a single sensor was used, which produced a single output signal. Websper [15] showed that the selection of input features is critical to the performance of the network, and highlighted the need for multiple inputs to a network.

In the feature extraction process many additional signals were developed. These stemmed from the fact that many different wavelets could be used for multiple decompositions.

When a neural network is being trained it attempts to find a mapping in multi-dimensional space from the input feature domain to the output domain. The position of a vector within the input features affects the ability of the network to learn. Thus the selection of the input features aims to simplify the input/output mapping of the network. As a general rule, it is desirable to have points from one problem class mapped into close proximity in the feature space, yet distinctive from points from the other problem class. Figure 55 shows this mapping process, where the problem domain has been re-organised into the input feature domain, by the extraction of the input features. Data presented to the network in this fashion is in optimal form.

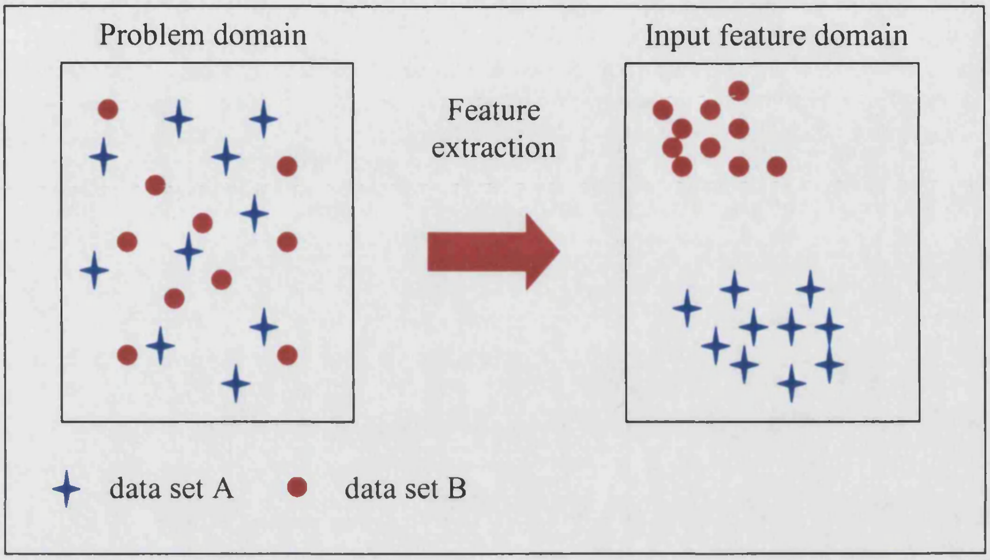


Figure 55 Feature extraction process

One method widely used for the selection of the input features is to select a subset of the input features and discard the remainder. This approach is useful if there are inputs which carry little useful information, or there is a strong correlation between sets of inputs where the same information is repeated in several variables. Following the selection of a possible subset of features it is desirable to view the performance of the selection. One possible method is cross-validation, where a network is trained with the relevant input features and the output performance is inspected. However, this process requires considerable time, when a change in the input features requires a complete retrain and test of the network. It is advantageous if the information on the performance of the selection of features can be visualised prior to training of a network.

7.3.3 Sammon algorithm

The Sammon algorithm [18] allows the visualisation of multi-dimensional data by reducing the data to an optimal two dimensional representation. This mapping occurs such that the approximate structure of the data is preserved. In this preservation the data is fitted in the two dimensional space so that the inter-point distances approximate the corresponding inter-point distance in the original dimension.

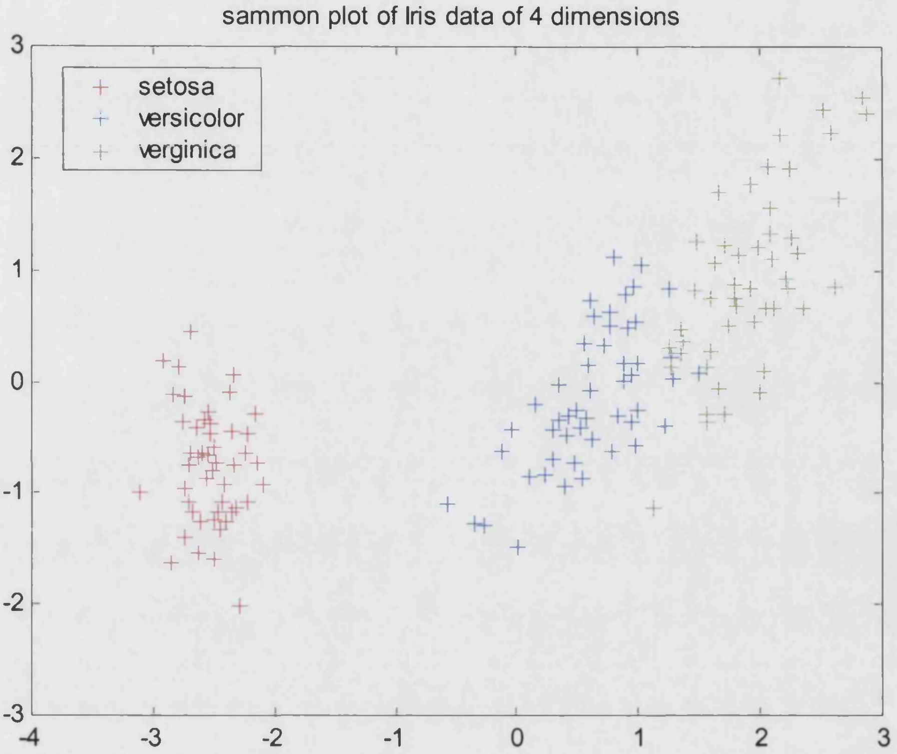


Figure 56 Sammon plot of Iris data of 4 dimensions

A standard example of the Sammon plot, taken from Vesanto [46] is shown in Figure 56. The data set consists of four measurements from 150 Iris flowers: 50 Iris-setosa, 50 Iris-versicolor and 50 Iris-virginica. The measurements are length and width of sepal and petal leaves. Each line represents a set of readings followed by a label. From the plot it is possible to see the groupings of the flowers.
When it is possible to split the data into classes without overlap on the Sammon plot, training of a network with these input features will be trivial, i.e. the network will learn in a reduced time, and will exhibit a high degree of classification accuracy. If there is a small amount of overlap between the classes on the Sammon

Plot, this will increase the training times and degrade the classification performance.

7.3.4 Visualisation of the input features

The input features selected were a subset of the 25 wavelet coefficients described in the feature extraction stage. The complete set comprised db2 wavelets to db6 wavelets from level 1 to level 5 decomposition. An initial test was done on all input features and the result is shown in Figure 57. The plot shows a fair degree of grouping of the data between before MPP and after MPP. However, the numerous stray vectors would cause the network to produce inaccuracies in the classification. The cause of the stray vectors appeared to be the uncertainty in the detection of MPP, so as an initial trial all input features which did not visually detect MPP were discarded. The subset of features were: db2 level 2, 3 & 4, db3 level 3, db4 level 3 & 4, db5 level 3, db6 level 3 & 4. Figure 58 shows this subset of 9 input features.

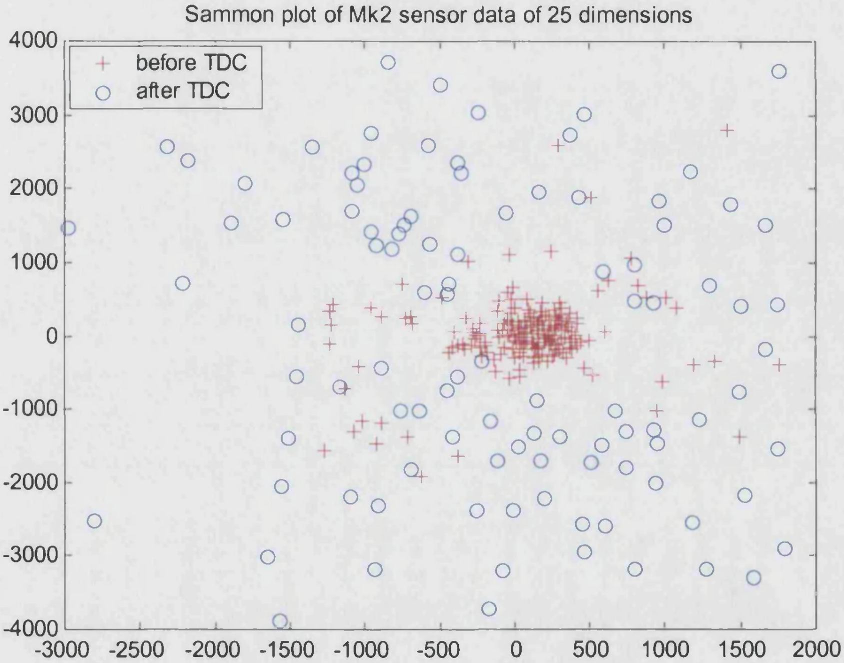


Figure 57 Sammon plot of Mk2 sensor data of 25 dimensions

Figure 58 shows this subset of 9 input features. The number of stray vectors after MPP has been reduced and a clear pattern can be seen. The separation of BMPP and AMPP has been successful at this number of input features.

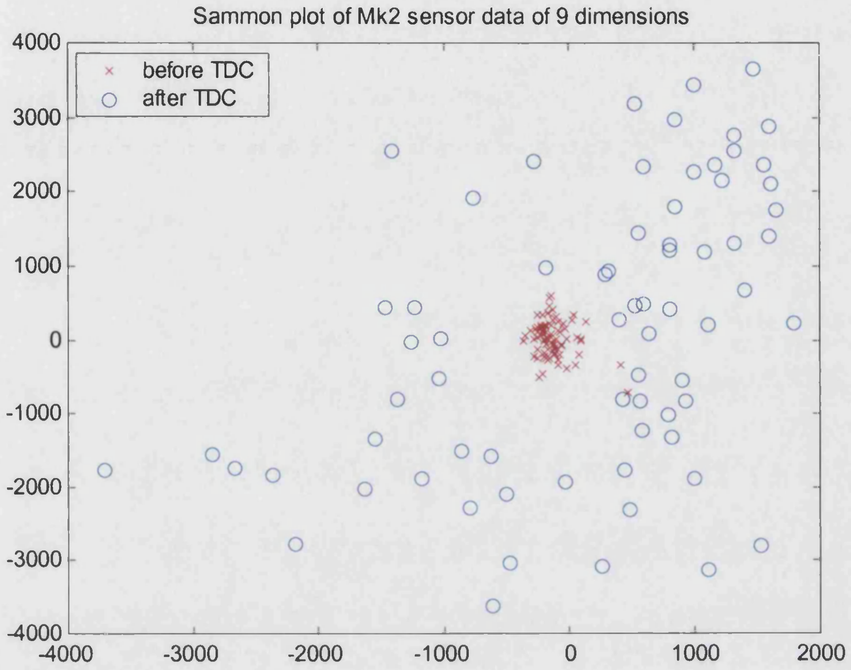


Figure 58 Sammon plot of Mk2 sensor data of 9 dimensions

Figure 59 shows the following input features: db2 level 2 & 3, db3 level 3, db4 level 4, db5 level 5, db6 level 4. This appears to be an acceptable set of alternatives.

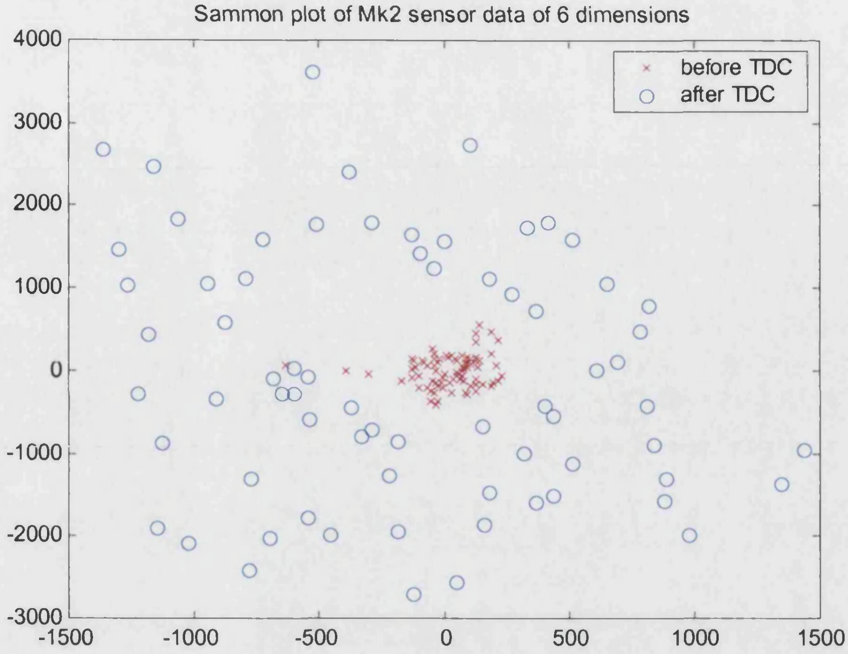


Figure 59 Sammon plot for Mk2 sensor data of 6 dimensions

7.3.5 Selection of the network

Based upon the intrusive engine data gathered from the Kistler transducer the start of combustion and location of peak pressure can be determined. Thus a Kohonen network can be trained with just the inputs from the wavelet coefficients and the in-cylinder transducer can be used to give an indication of successful classification. Figure 60 shows a block diagram of this.

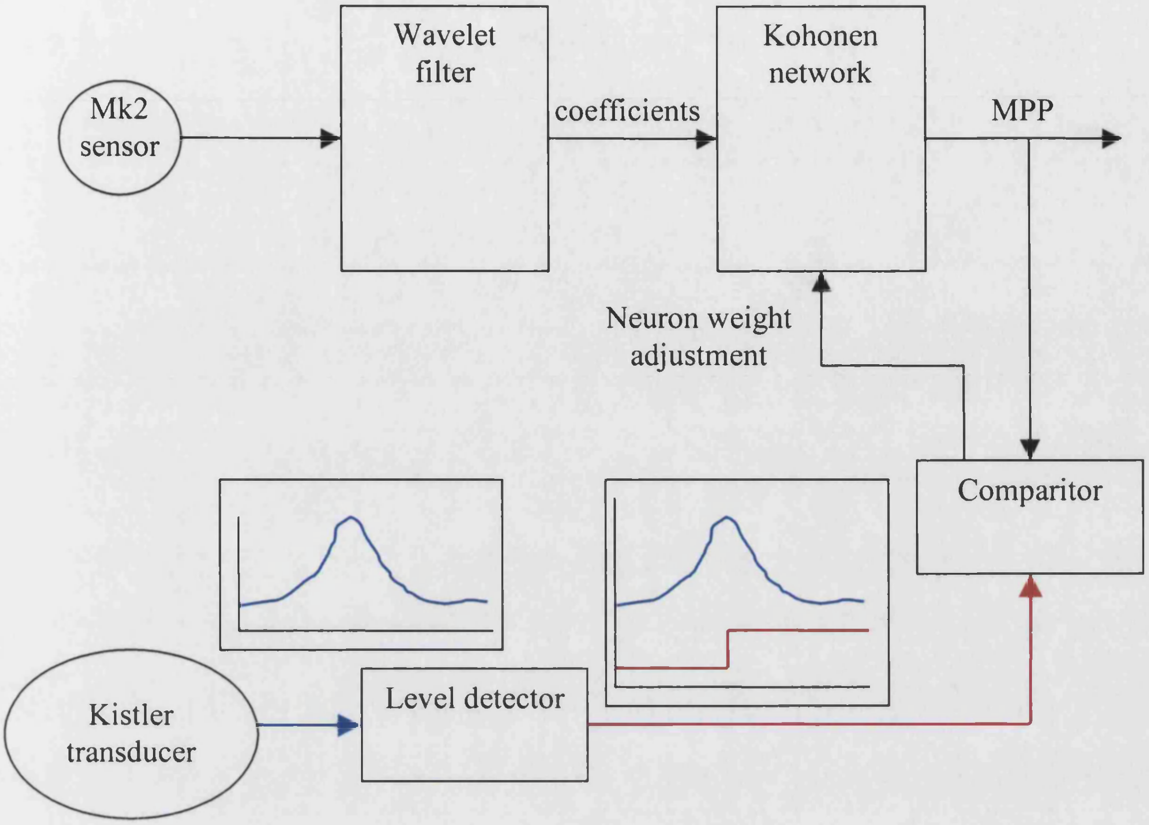


Figure 60 Kohonen network training

7.4 Discussion

In this Chapter, the features from the extraction stage were examined for use in detection. It started with a technique called thresholding, which attempts to remove the noise from unwanted frequency bands, and then reconstructed the signal. However, it was pointed out that the function of thresholding was done manually and an automated process was not readily available.

Neural networks were investigated in Chapter 2.4.1 and this Chapter used this to establish a method to detect location of peak pressure and start of combustion. The method used the Sammon algorithm to eliminate unnecessary input features and reduce the input feature space from 25 to 6. Finally the type of network is chosen and the arrangement of training was detailed.

8 CONCLUSIONS

8.1 Introduction

This research has established that it is possible to dynamically time an internal combustion engine based on a non-intrusive sensor. It built on the work started by Kirkman in determining cylinder pressure from forces exerted on the cylinder head bolts. The research utilised multi-resolution analysis combined with artificial neural networks to overcome the problem of contamination of signals from structures in the engine, which had dogged previous research. The purpose of this research was to develop a sensor and associated signal processing to enable the non-intrusive measurement of the combustion process to take place. It established that future systems would rely on measurements of the combustion process to accurately control the engine and transmission. It noted that in the field of non-intrusive pressure measurement several areas of investigation exist. One of these areas – head bolt measurement – was chosen since it matched the preliminary investigations on non-intrusive sensors undertaken for this thesis. The signal processing strategy followed the path of several similar problems from separate disciplines. These were investigated and a common approach was taken.

8.2 Sensors

The first phase of this research set out to develop a miniaturised sensor which could be used on small, single and multi-cylinder engines. The work started with the design of the Mk1 sensor. This was based on a 'washer stack' with an active element of piezoelectric ceramic material placed in the centre of the stack. Results were obtained by installing the sensor on a small (412cm^3) single cylinder diesel engine. The research reproduced the results obtained by Kirkman using different materials and techniques, and established a baseline for the design of the subsequent sensors. However, the Mk1 sensor could not survive the torque of the head bolts on the multi-cylinder engine and so the Mk2 sensor was developed. The Mk2 sensor used a load-sharing device with the majority of the head bolt torque taken by the load washer. The sensor was installed on a four cylinder diesel engine of 1800cm^3 and a set of results obtained. The results contained large amounts of signal contamination, which obscured the features of interest – namely location of peak pressure and start of combustion. This confirmed the results obtained by previous studies in showing that the signals derived from sensors placed on a secondary signal source were subject to cross contamination from many primary signal sources.

8.3 Digital signal processing system

In order to process the signals in a timely manner, a digital signal processing system was developed. The benefit of a custom built system stems from the optimisation of the development process for the algorithms and real time performance. Algorithms could be developed in a text based or graphical manner and downloaded to the DSP system.

The custom built system consisted of an off line data capture and analysis system, and a real time rapid prototyping system. The off line system enabled analysis to be conducted using standard mathematical packages, with the advantages of report and graphics generation. The real time system enabled the computational effectiveness of the algorithms to be tested.

8.4 Feature extraction

The signals captured from the Mk2 sensor contained the features of interest but these were buried in noise and contamination. It was the task of the feature extraction stage to remove the noise and contamination from the signals to leave the desired features. Traditional methods of feature extraction rely on frequency filters and the Fourier transform. However, it was shown that these are ineffective.

A survey of research conducted in other fields - mathematics, image processing, telecommunications and medicine - showed that a multi-resolution approach to the problem could produce the desired results. In order to develop a multi-resolution analysis of the signal, a set of wavelet transforms was selected. It was found that the Daubechies wavelets were particularly effective at localising signals in time and frequency. Once the wavelet transform had been chosen, the data sets captured from the engine were converted into their wavelet coefficient equivalents. The decomposition of the signals into their wavelet coefficients proved successful and the task of feature recognition then took place.

8.5 Feature recognition

For each signal captured from the Mk2 sensor there existed 25 wavelet coefficient signals. Some of these contained useful information on the location of peak pressure and start of combustion whilst others did not. The technique of thresholding was applied to the wavelet coefficients in an effort to remove the noise and contamination from the signal. This proved to be very successful and after reconstruction of the waveform, LPP and SoC could be clearly seen. However, the thresholding function was conducted by manually adjusting the weights of each of the threshold values, and this could not be automated for other signals. Thus the problem with thresholding was to find an algorithm which would give the best values for the threshold weights, and this proved elusive.

In the survey of relevant DSP techniques, artificial neural networks were identified as a method to classify complex non-linear relationships between signals. The use of neural networks for this type of problem has been well documented, and an examination of the wavelet coefficients by the Sammon algorithm revealed that there existed a strong correlation between the wavelet coefficients.

In previous research, the importance of selection of input parameters to the network was highlighted. With 25 possible inputs, a subset of wavelet coefficients needed to be found which contained the information to classify the signal into before peak pressure and after peak pressure.

Using visual judgement several wavelet coefficients were selected as possible inputs to the network. The Sammon algorithm was used to quantify this visual selection and resulted in 6 wavelet coefficients being selected as representing the best classification of the signal.

8.6 Dynamic timing

The final part to the research was on the selection of the network. The unsupervised learning scheme of the Kohonen network lent itself to the solution of the problem. The Kistler transducer was used as a learning marker for the Kohonen network to determine if the output signal should be before peak pressure (0) or after peak pressure (1).

8.7 Summary

The culmination of this research project is a low-cost sensor and signal conditioning system, which produces a high grade signals indicating the timing of start of combustion and peak pressure. This has been achieved by a combination of non-intrusive sensor, wavelet decomposition and artificial neural network to produce a useful measure of the combustion process. This low-cost measure of the combustion process enables a reduction of toxic emissions and CO_2 by the deployment of closed-loop control systems on production IC engines.

9 *FUTURE DEVELOPMENTS*

9.1 Introduction

The research reported in this thesis has successfully demonstrated a technique for inferring information about the combustion process from sensors placed on the outside of the engine. To develop this work, and to achieve the end-goal of producing a non-intrusive pressure transducer for production IC engines, further research is required. Some areas of investigation to be pursued are discussed below.

9.2 Mk 3 sensor

Some preliminary work on the development of a Mk3 sensor has already been done. Its design is an evolution of the Mk2 sensor with four pockets of PZT crystal rather than one. This reduces the amount of load bearing surface and increases the sensitivity of the device. The integrity of the device is still maintained, and the active elements are contained within the unit, so eliminating the ingress of contaminants. Figure 61 shows the general arrangement of the sensor. This design increases the signal to noise ratio of the sensor whilst maintaining integrity and a safe failure mode. However, further work and testing are needed to verify that the Mk3 sensor meets the predicted increase in signal to noise ratio.

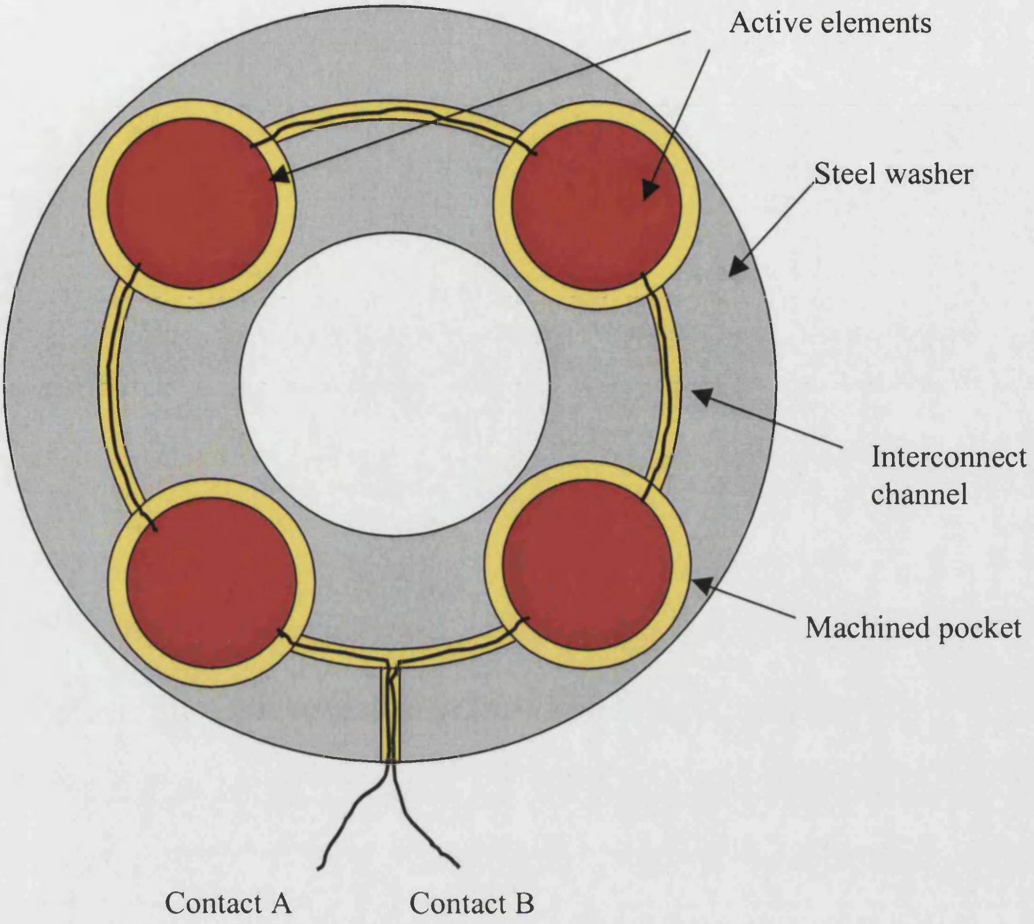


Figure 61 Cross section of Mk 3 sensor

9.3 Mk 4 sensor

Some preliminary work has also been done on a Mk4 sensor. The design stems from a detailed analysis of the best location for the sensor. Previous research by Kirkman had demonstrated that the head bolts of IC engines provide a good location. However, there are many other places which may provide additional information or convenience. The Mk 2 & 3 sensors are limited in their location and number to the positions occupied by the head bolts. However, as was seen in section 3.5, head bolts are frequently used to secure other items of engine equipment. In the engine used for this research the camshaft was located to one side of the cylinder head and used the head bolts as anchor points. Sensors placed under these head bolts were subjected to variations in camshaft load, which added noise to the signal. Therefore, using head bolts as the source of the signal limited the number and the location of the sensors.

A location for the sensors, which would avoid some of the head bolt limitations, is under the cylinder head. There is an additional advantage to this, in that sensors embedded into the head gasket can be pre-manufactured and assembled as a standard part with the engine. However, a potential problem would be that the sensors were located closer to the heat source from the combustion process.

9.4 Signal processing

The research reported in this thesis was based on a single non-intrusive sensor and an in-cylinder pressure transducer. It was noted in section 2.4.1, that neural networks were a valuable tool in drawing relationships between highly non-linear signals. However, the main problem faced early on was the lack of input features to the network. Thus the research concentrated on extracting the features from a single source and using these as the inputs to a network.

A strategy could be developed which used multiple signal sources from many sensors. It could perform analysis between signals and use these as the input to a network directly.

9.5 Summary

The research focused on extracting information from a single signal source, using wavelet decomposition and a discrete neural network function. This enabled the location of peak pressure and the start of combustion to be identified. However, the magnitude of peak pressure and the general shape of the pressure curve remained undetected.

By using a number of sensors with the wavelet decomposition and a continuous neural network function there may be a method to reproduce accurately the cylinder pressure trace.

LIST OF REFERENCES

1 Department of the Environment, Transport and the Regions (13th August 1999). “The Environmental Impacts of Road Vehicles in Use - Air Quality, Climate Change and Noise Pollution”, Cleaner Vehicles Task Force, DETR Free Literature, PO Box No 236, Wetherby LS23 7NB.

2 Vehicle Certification Agency. (2001) “Exhaust Emissions Testing”. 1 The Eastgate Office Centre, Eastgate Road, Bristol. BS5 6XX.

3 Turner, J.D. (2000). “A review of current sensor technologies and applications within automotive and traffic control systems”. *Proceedings of the Institute of Mechanical Engineers*, **Volume 214, Part D,** pp. 589-614.

4 Stone, R. (1985). “Introduction to internal combustion engines”. *MacMillan Press Ltd., ISBN 0-33-74013-0,* pp. 211-217.

5 Glavmo, M., Spadafora, P. and Bosch, R. (1999). “Closed loop start of combustion utilising ionizing sensing in a diesel engine”. *Society of Automobile Engineers, Journal of Engines,* **Paper 1999-01-0549,** pp. 801–807.

6 Geiser, F., Wytrykus, F. (1998). “Combustion control with the optical fibre fitted production spark plug”. *Society of Automobile Engineers, Journal of Engines,* **Paper 980139,** pp. 118-130.

7 Muller, R. and Hemberger, H-H. “Neural adaptive ignition control”. *Society of Automobile Engineers, Journal of Engines,* **Paper 981057,** pp. 1636–1641.

8 Sellnau, M.C. and Matekunas, F.A. (2000). “Cylinder–pressure based engine control using pressure ratio management and low cost non-intrusive cylinder pressure sensors”. *Society of Automobile Engineers International Congress and Exposition,* **Paper 00P-379.**

9 Gu, F., Jacob, P.J. and Ball, A.D. (1999). “Non-parametric models in the monitoring of engine performance and condition, part 2: non-intrusive estimation of diesel engine cylinder pressure and its use in fault detection”. *Proceedings of the Institute of Mechanical Engineers,* **Volume 213 part D,** pp. 135-143.

10 Kirkman, E.T.F. (1991). "The indirect measurement of diesel engine cylinder pressure". *M.Sc Thesis, Southampton University.*

11 Widrow, B. and Hoff, M.E. (1960) "Adaptive switching circuits". *WESCON Convention Record,* **Part IV**, pp. 96-104.

12 Kohonen, T. (1988) "The neural phonetic typewriter". *IEEE Computer Magazine,* **Vol. 21**, pp.11-22.

13 Hopfield, J.J. (1982) "Neural networks and physical systems with emergent collective computational abilities". *Proceedings of the National Academy of Science,* **Vol. 74,** pp. 2554-2558.

14 Lippmann, R.P. (1989) "Pattern classification using neural networks". *IEEE Communications Magazine,* **Vol. 21,** pp. 105-117.

15 Websper, S.P. (1997). "An A.I. based solution to the problem of adaptive distance protection". *Ph.D. thesis, University of Bath.*

16 Chen, H. and Hewit, J. (2000) "Application of wavelet transforms and neural networks to the recognition and classification of blemishes". *Mechatronics,* **Vol. 10,** pp. 699-711.

17 Haar A. (1910). "Zur Theorie der orthogonalen Funktionen-Systeme", *Mathematische Annalen,* **Vol. 69,** pp. 331-371.

18 Sammon. J.W., (1969). "A nonlinear mapping for data structure analysis". *IEEE transactions on computers,* **Vol. C-18, No. 5,** pp. 401-409.

19 Wang, W., Chirwa, E.C., Zhou, E., Holmes, K. and Nwagboso, C. (1999) "Fuzzy ignition timing control for a spark ignition engine", *Proceedings of the Institute of Mechanical Engineers,* **Vol. 214 part D,** pp. 297-306.

20 Muller, R. et al. (2000). "Combustion pressure based engine management system". *Society of Automobile Engineers International Congress and Exposition.* **paper 00P-301.**

21 Lee, S.B et al. (1996) "Micromachined interferometer for dynamic high-pressure sensing", *Sensors,* **Vol. 13, No. 6,** pp. 31-36.

22 DeJong, R.G et al, (1986). "Engine monitoring using vibration signals". *Society of Automobile Engineers International Congress and Exposition.* **Paper 861246.**

23 Ordubadi, A. (1982). "Fault detection in internal combustion engines using an acoustic signal". *Society of Automobile Engineers Proceedings*, pp.145-150.

24 Gibberty, R.J. (1987). "Correlation between detonation, pressure and vibration in a petrol engine". *M.Sc. Thesis, Institute of Sound and Vibration Research, Southampton University.*

25 Miyamoto, N. (1983) "Unique measuring method of indicator diagrams using strain history of head bolts". *Society of Automobile Engineers International Congress and Exposition.*

26 Shayler, P.J., Goodman, M. and Ma, T. (2000) "The exploitation of neural networks in automotive engine management systems". *Engineering Applications of Artificial Intelligence.* **Volume 13,** pp. 147-157.

27 Leonhardt, S., Ludwig, C. and Schwarz, R. (1995) "Real-time supervision for diesel engine injection". *Control Engineering Practice.* **Vol. 3, No. 7,** pp. 1003-1010.

28 Kirkman, E.T.F. and Elliott, C. (1995). "Condition monitoring of marine engineering equipment using neural computing". *The Institute of Marine Engineers Transactions.* **Vol. 107, Part 3,** pp. 185-192.

29 Brace, C. (1996). "Transient modelling of a DI TCi Diesel engine". *Ph.D. thesis, University of Bath.*

30 Scaife, M.W. (1993) "A neural network for fault recognition". *Society of Automobile Engineers International Congress and Exposition.* **Session 1P23.**

31 Daubechies, I (1988). "Orthonormal Bases of Compactly Supported Wavelets". *Communications on Pure and Applied mathematics.* **Vol. 31. No 8.** pp. 909-996.

32 Yao, Y., Li. X., and Yuan, Z. (1999) "Tool wear detection with fuzzy classification and wavelet fuzzy neural network". *International Journal of Machine Tools & Manufacture.* **Vol. 39,** pp.1525-1538.

33 Bakhtazad, A., Palazoglu, A. and Romagnoli, J.A.. (1999) "Process data de-noising using wavelet transform". *Intelligent Data Analysis.* **Vol. 3,** pp.267-285.

34 Daubechies, I. (1990). "The Wavelet Transform, Time-Frequency Localisation and Signal Analysis". *IEEE Transactions on Information Theory.* **Vol. 36,** pp. 961-1005.

35 Daubechies, I (1988). "Time-frequency localization operators – A geometric phase space approach". *IEEE Transactions on Information Theory.* **Vol. 34,** pp. 605-612.

36 Mallat, S. (1988). "A theory for multiresolution signal decomposition". *Ph.D. thesis, Univ. of Pennsylvania, Depts. of Elect. Eng. and Comput. Sci.*

37 Mallat, S (1989). "Multiresolution approximation and wavelets". *Trans. Am. Math. Soc.* **Vol. 135,** pp. 69-88.

38 Mallat, S (1989). "A Theory for multiresolution signal decomposition: The wavelet representation". *IEEE Trans. Pattern Anal. Machine Intell.* **Vol. 31,** pp. 674-693.

39 Mallat, S and Hwang, W.L. (1992). "Singularity Detection and Processing with Wavelets". *IEEE Transactions on Information Theory.* **Vol. 38,** pp. 617-643.

40 Mallat, S (1987). "Multifrequency channel decompositions of images and wavelet models", *IEEE Trans. Acoust. Speech Signal Processing.* **Vol. 37 no.12,** pp. 2091-2110.

41 Kovacevic, J and Vetterli, M (1992). "Nonseparable Multidimensional Perfect Reconstruction Filter Banks and Wavelet Bases for R^n". *IEEE Transactions on Information Theory.* **Vol. 38,** pp. 533-555.

42 Rioul, O and Duhamel, P. (1992). "Fast Algorithms for Discrete and Continuous Wavelet Transforms". *IEEE Transactions on Information Theory.* **Vol. 38,** pp. 569-586.

43 Lang, M., Guo, H., Odegard, J.E., Burrus, C.S. and Wells, Jr. R.O. (1996). "Noise Reduction Using an Undecimated Discrete Wavelet Transform". *IEEE Transactions on Signal Processing Letters.* **Vol. 3,** pp. 10-12.

44 Morgan Matroc Ltd, "Introduction to Piezoelectric Ceramics", Thornhill Southampton, Hants. SO19 7TG. 01703 444811.

45 Basu, S. (1998) “Multi-dimensional Filter Banks and Wavelets – A System Theoretic Perspective”. *J Franklin Institute*, **Vol. 335B**, No. 8, pp. 1367-1409.

46 Vesanto J. et al. (1999). “Self-organizing map in matlab: the som toolbox”. *Proceedings of the matlab DSP conference, Espoo, Finland.* pp. 35-40.

BIBLIOGRAPHY

Ainsworth M., Levesley J., Marletta M. and Light W.A. "Wavelets, multilevel methods and elliptical PDEs" in Numerical Maths and Scientific Computation. 1997. Clarendon Press. ISBN 0-19-850190-0.

Bellanger M. "Digital processing of signals, theory and practice". 1984. Wiley ISBN 0-471-903183.

Burrus C.S., Gopinath R.A. and Guo H. "Introduction to Wavelets and Wavelet Transforms: A Primer". 1998. Prentice Hall. ISBN 0-13-489600-9.

Chui C.K. "An Introduction to Wavelets". 1992. Academic Press Ltd. ISBN 0-12-174584-8.

Chui C.K. "Wavelets: A mathematical tool for signal analysis". 1997. SIAM. ISBN 0-89871-384-6.

Corochiere R.E and Rabiner L. "Multi-rate Digital Signal Processing". 1983. Prentice-Hall. ISBN 0-13-605162-6.

Creasey D.J. (Ed.) "Advanced Signal Processing". IEE telecommunication series 13. 1985. Peter Peregrins Ltd., ISBN 0-471-90318-3.

Folland G.B. "Fourier analysis and its applications". 1992. Brooks/Cole Publishing. ISBN 0-534-17094-3.

Ifeachor E.C and Jervis E.C. "Digital Signal Processing a Practical Approach". 1993. Addison Wesley ISBN 0-201-54413

Jaideva C., Goswami A. and Chan K. "Fundamentals of Wavelets, Theory, Algorithms and Applications". 1999. Wiley ISBN 0-471-19748-3.

Jones N.B. (Ed.) "Digital Signal Processing". IEE Control Engineering Series 22. 1982. Peter Peregrins Ltd. ISBN 0-906048-91-5.

Kaiser G. "A friendly guide to wavelets". 1994. Birkhauser. ISBN 0-8176-3711-7.

Lim J.S. and Oppenheim A.V. "Advanced Topics in Signal Processing". 1988. Prentice Hall. ISBN 0-13-013129-6.

Louis A.K., Maab P. and Rieder A. "Wavelets: theory and applications". 1997. Wiley & Sons. ISBN 0-471-96792-0.

Nievergelt. N "Wavelets made easy". 1999. Birkhauser. ISBN 0-8176-4061-4.

Stone R. "Introduction to internal combustion engines". 1985 .MacMillan Press Ltd., ISBN 0-33-74013-0

Strang G. and Nguyen T. "Wavelets and Filter Banks". 1997. Wellesley-Cambridge Press. ISBN 0-9614088-7-1.

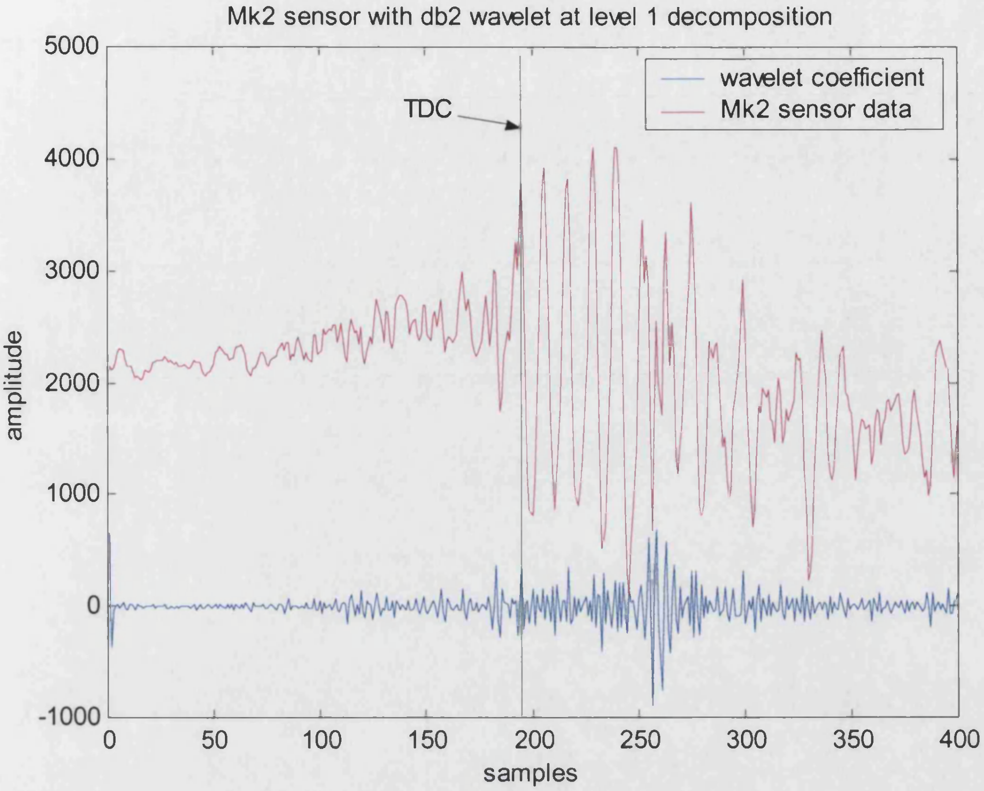
Ziemer R.E. and Peterson R.L. "Digital Communications and Spread Spectrum Systems". 1989. ISBN 0-02-431670-9.

UK Department of the Environment, Transport and the Regions.
http://www.roads.detr.gov.uk/cvft/impact/3.htm

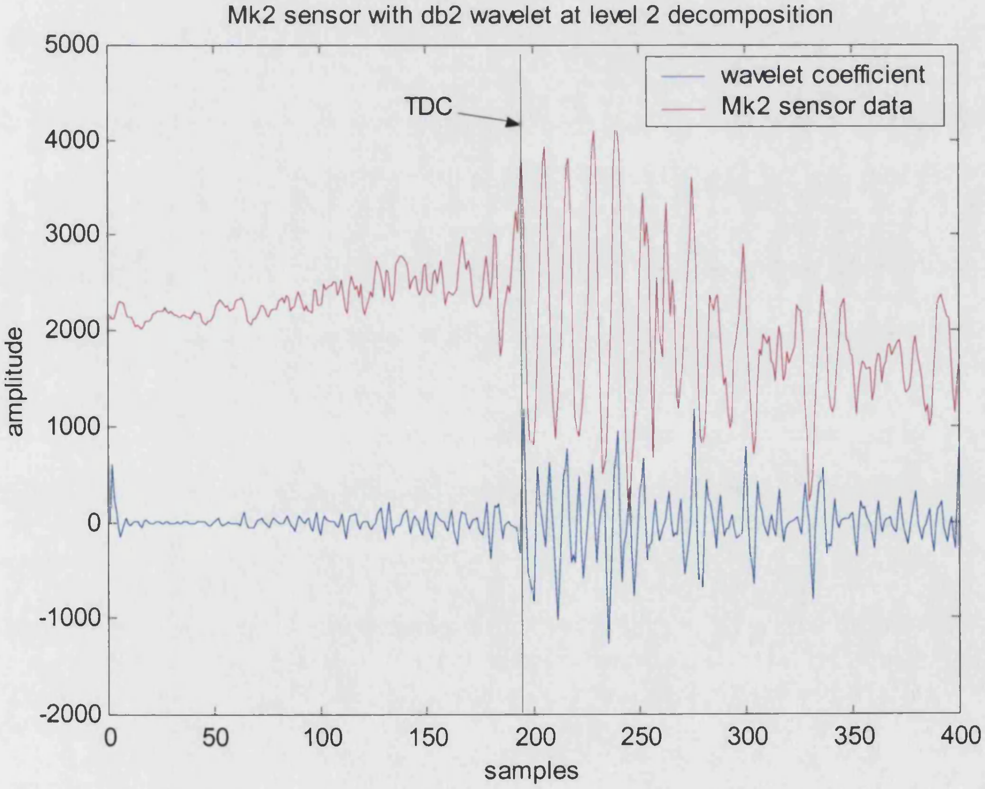
"ADSP-21000 Family Development Tools: ADDS-210xx-TOOLS"
http://www.analog.com/pdf/dds210xx.pdf

APPENDIX A – ADDITIONAL ENGINE DATA RESULTS

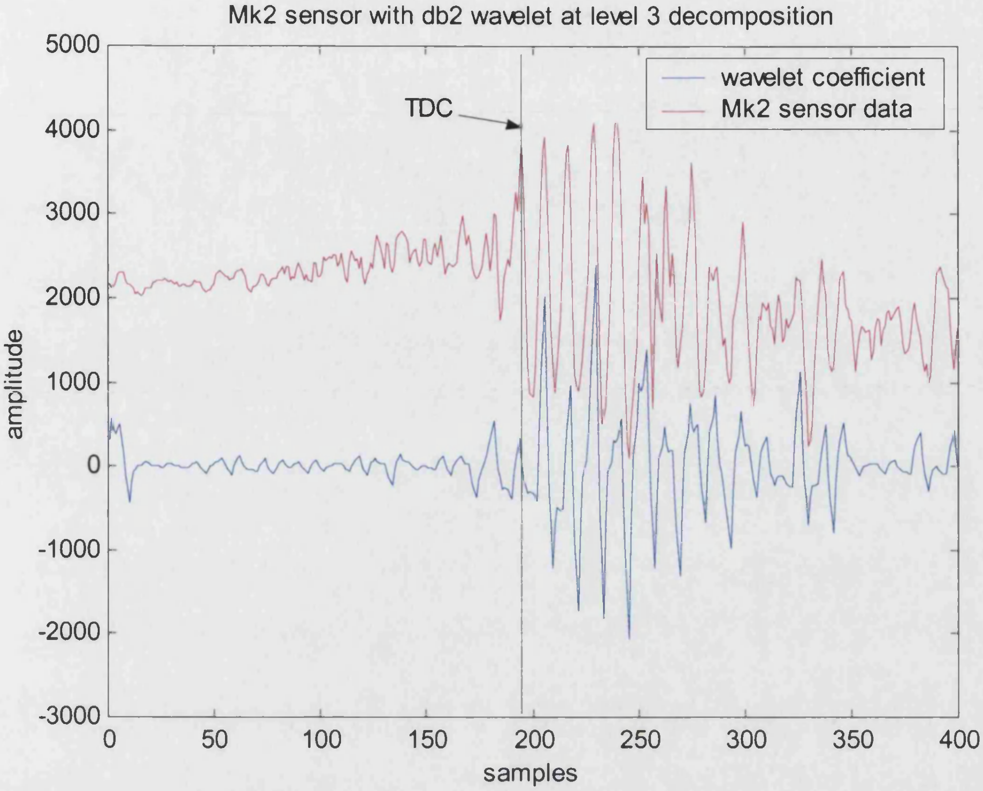
These results represent a complete set of wavelet decompositions. The decompositions start with the Daubechies db2 wavelet and finish with the db6 wavelet. For each wavelet there are 5 levels of decomposition (1 to 5).
Thus a set of 25 decompositions represent a single set from one engine test. Additional engine tests have not been included for clarity.

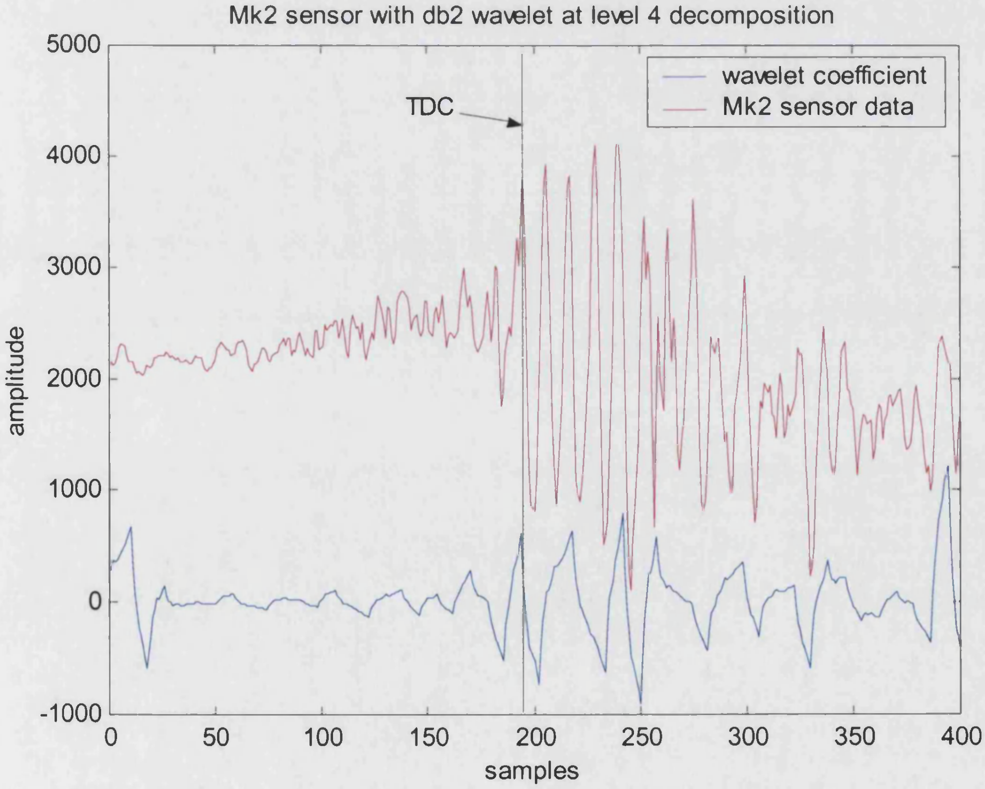


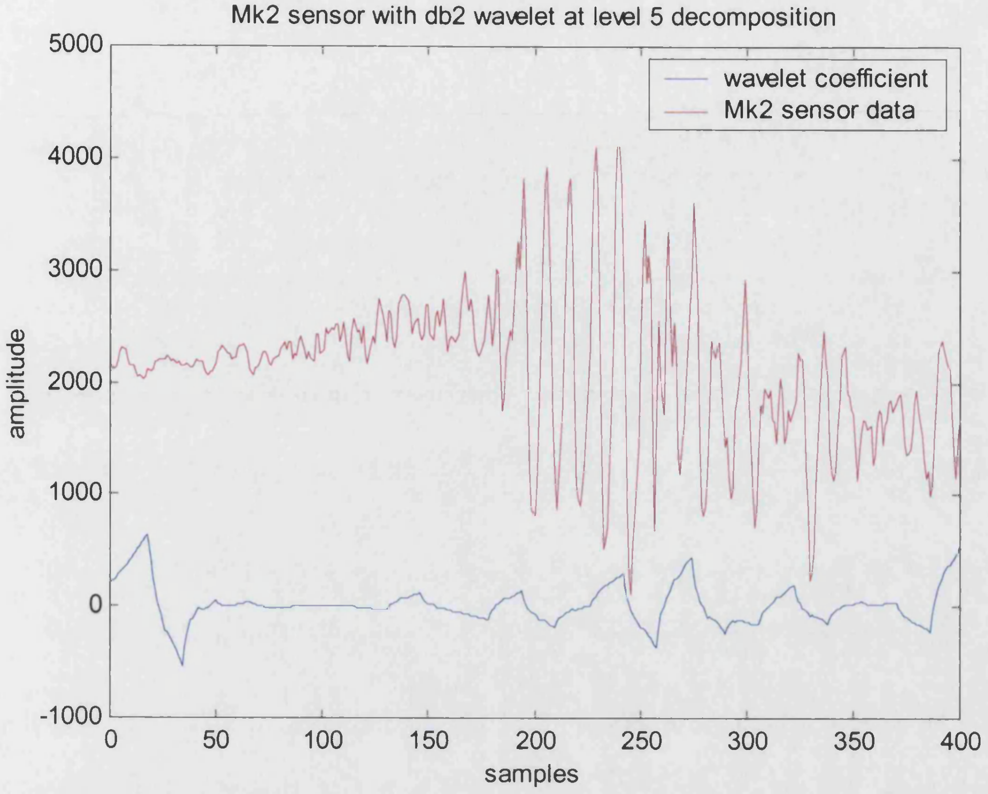
Mk2 sensor with db2 wavelet at level 1 decomposition
wavelet coefficient
Mk2 sensor data
TDC
amplitude
5000
4000
3000
2000
1000
0
-1000
0
50
100
150
200
250
300
350
400
samples

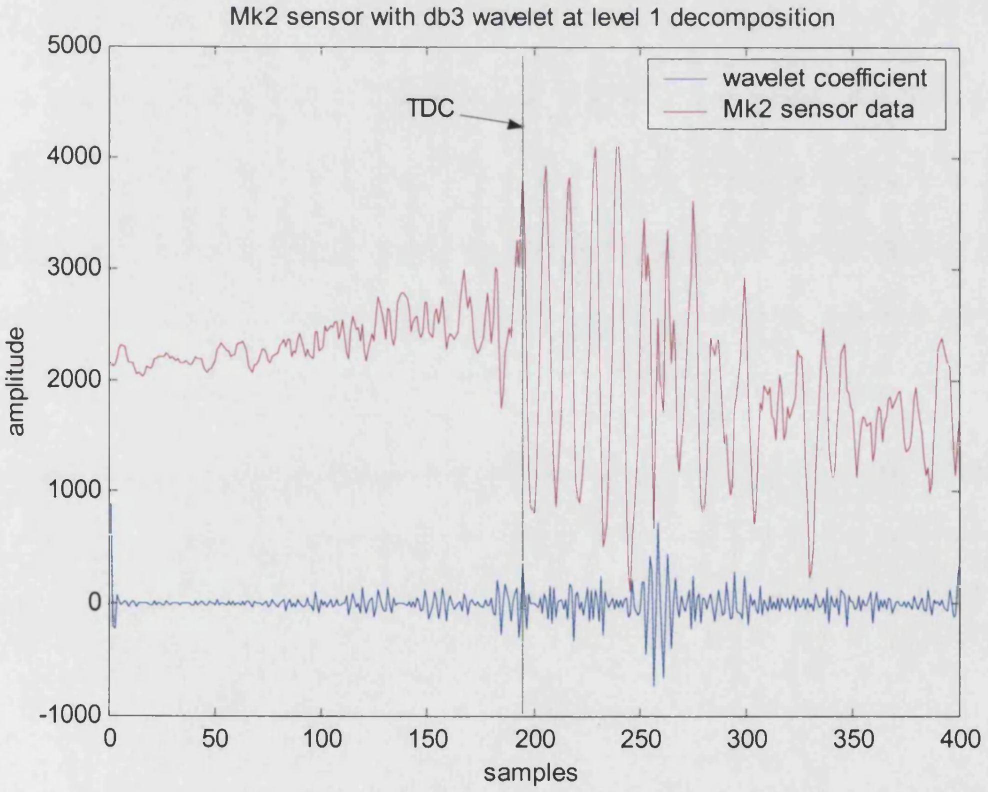


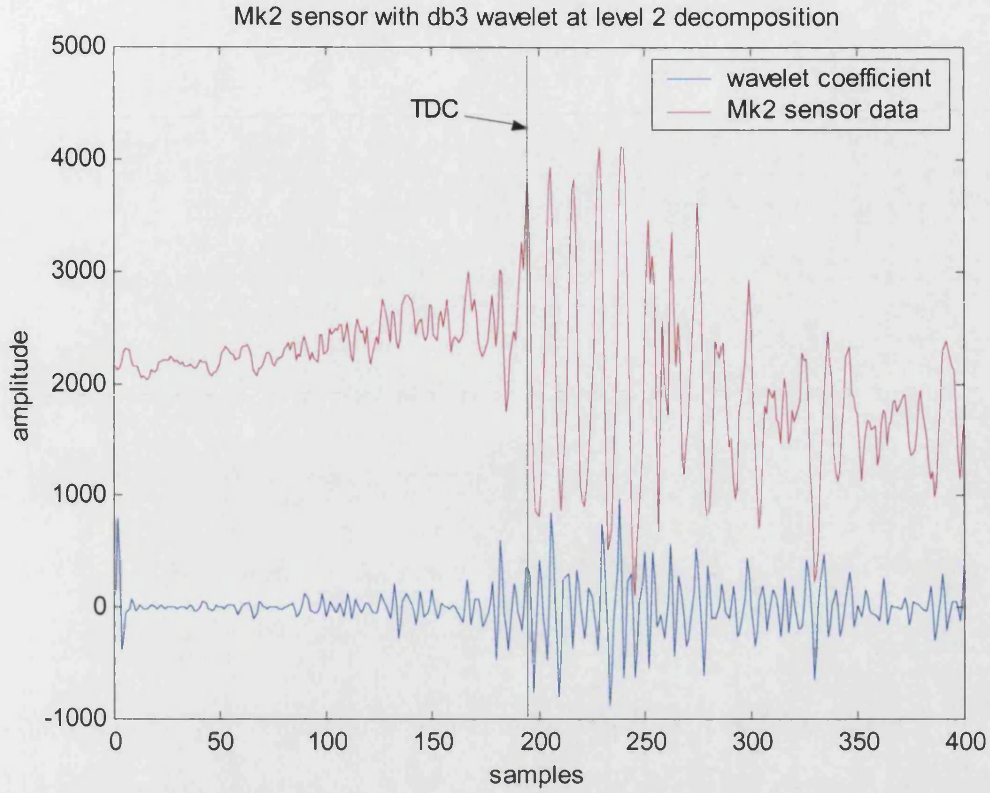
Mk2 sensor with db2 wavelet at level 2 decomposition
wavelet coefficient
Mk2 sensor data
TDC
amplitude
5000
4000
3000
2000
1000
0
-1000
-2000
0
50
100
150
200
250
300
350
400
samples

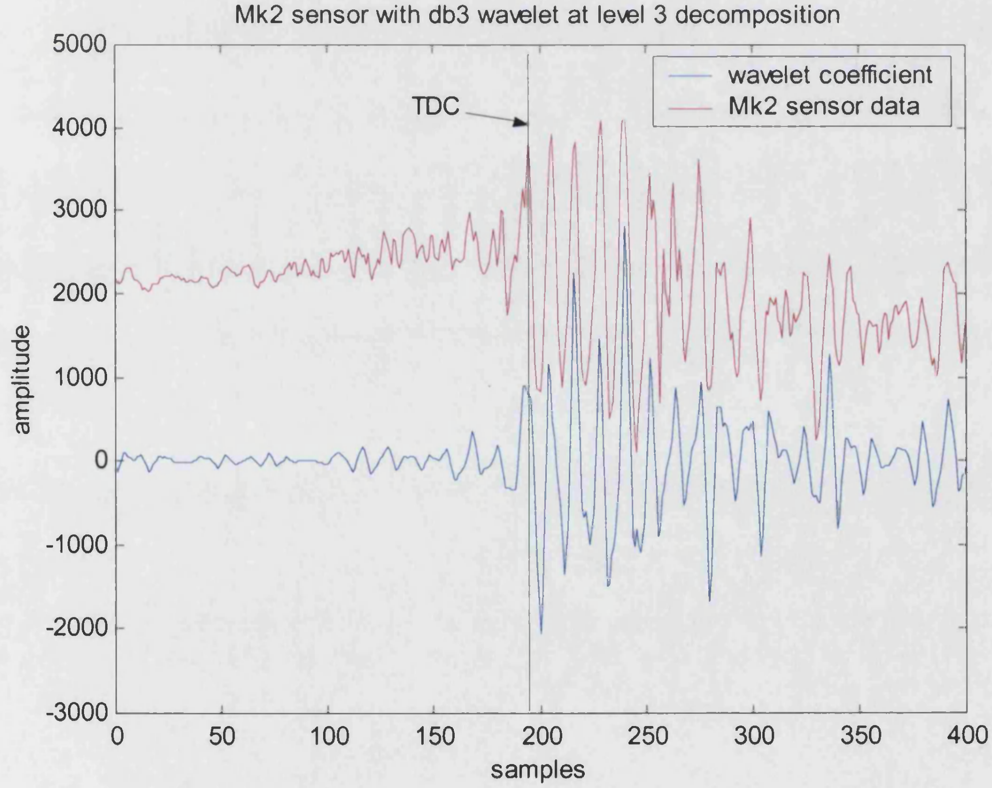

Mk2 sensor with db2 wavelet at level 3 decomposition
wavelet coefficient
Mk2 sensor data
TDC
amplitude
samples
5000
4000
3000
2000
1000
0
-1000
-2000
-3000
0
50
100
150
200
250
300
350
400

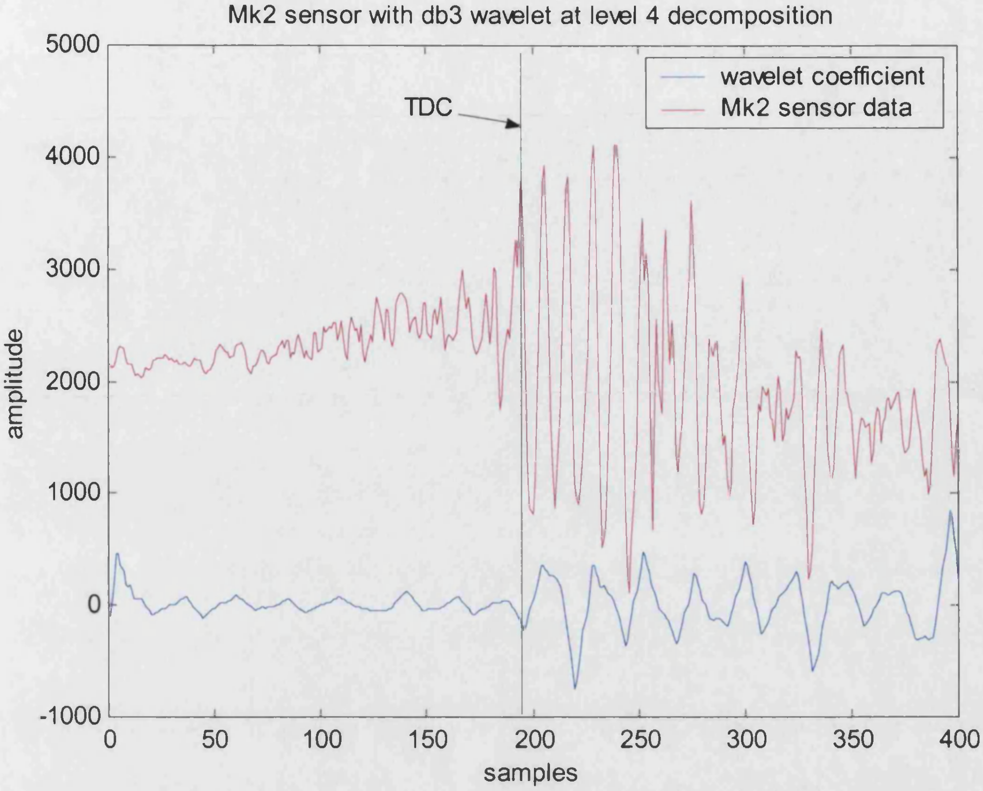

Mk2 sensor with db2 wavelet at level 4 decomposition
wavelet coefficient
Mk2 sensor data
TDC
amplitude
samples
5000
4000
3000
2000
1000
0
-1000
0
50
100
150
200
250
300
350
400

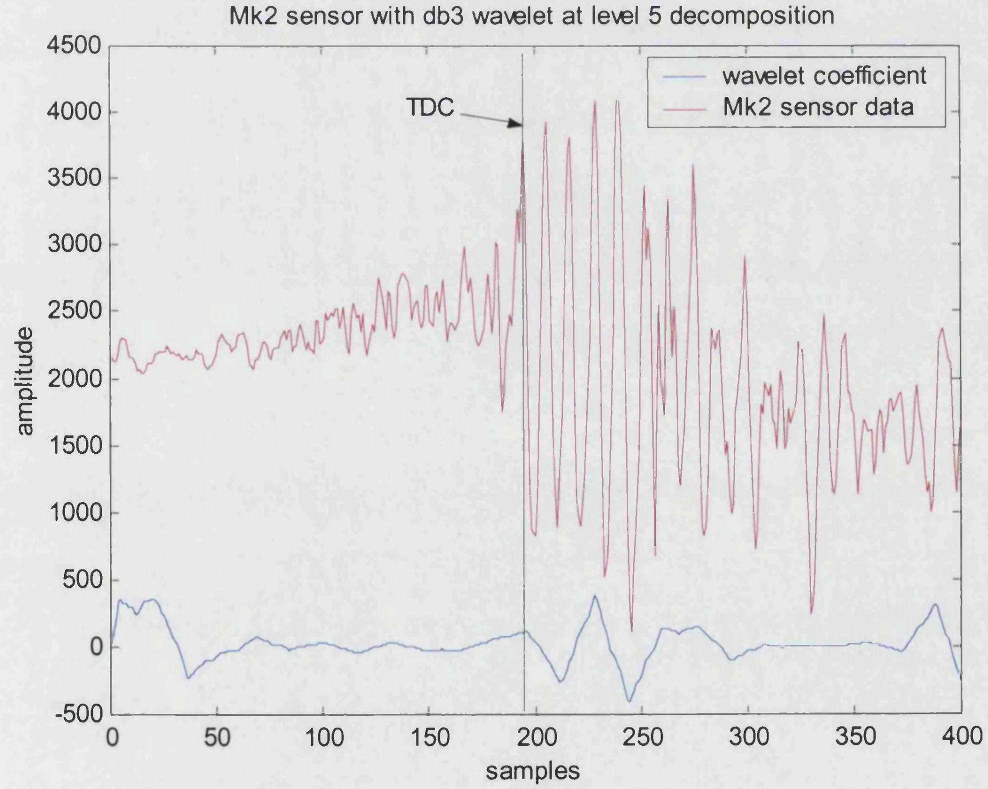

Mk2 sensor with db2 wavelet at level 5 decomposition
wavelet coefficient
Mk2 sensor data
amplitude
5000
4000
3000
2000
1000
0
-1000
0
50
100
150
200
250
300
350
400
samples

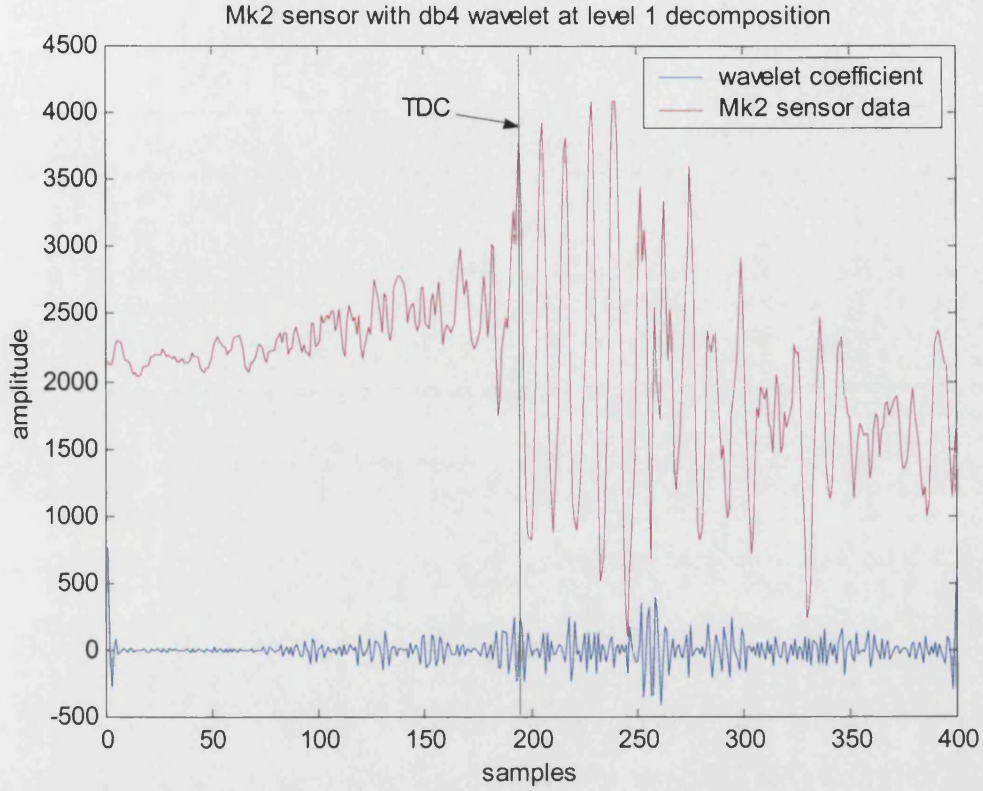

Mk2 sensor with db3 wavelet at level 1 decomposition
wavelet coefficient
Mk2 sensor data
TDC
amplitude
5000
4000
3000
2000
1000
0
-1000
0
50
100
150
200
250
300
350
400
samples

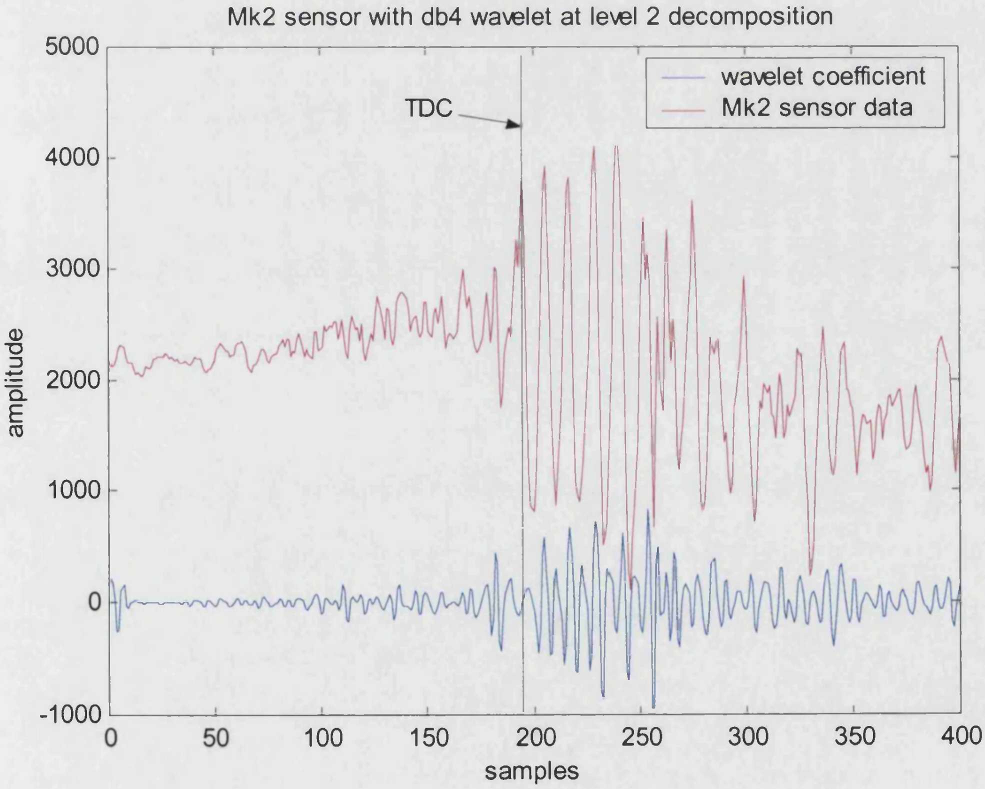

Mk2 sensor with db3 wavelet at level 2 decomposition
wavelet coefficient
Mk2 sensor data
TDC
amplitude
5000
4000
3000
2000
1000
0
-1000
0
50
100
150
200
250
300
350
400
samples

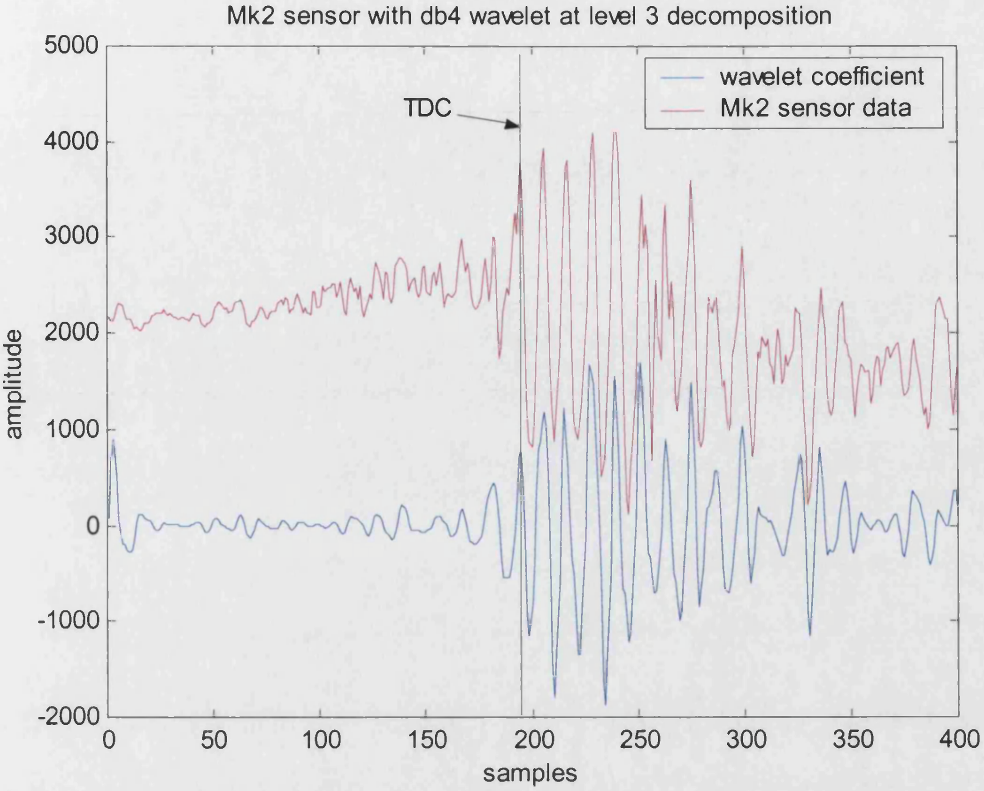

Mk2 sensor with db3 wavelet at level 3 decomposition
wavelet coefficient
Mk2 sensor data
TDC
amplitude
5000
4000
3000
2000
1000
0
-1000
-2000
-3000
0
50
100
150
200
250
300
350
400
samples

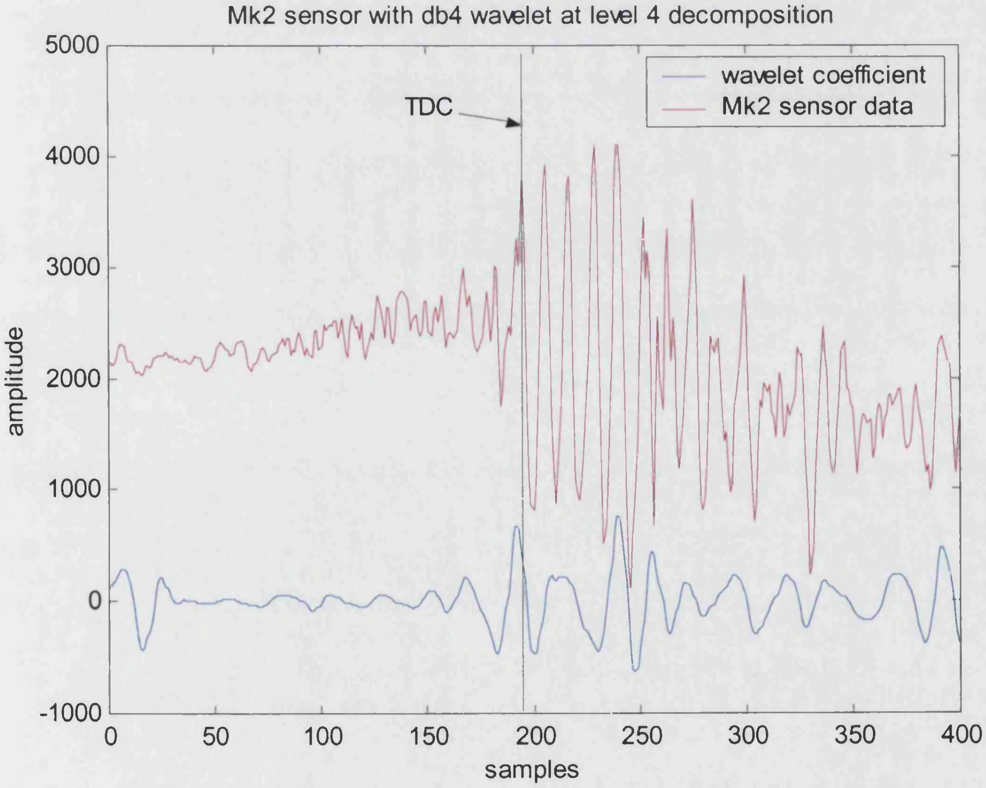

Mk2 sensor with db3 wavelet at level 4 decomposition
wavelet coefficient
Mk2 sensor data
TDC
amplitude
samples
5000
4000
3000
2000
1000
0
-1000
0
50
100
150
200
250
300
350
400

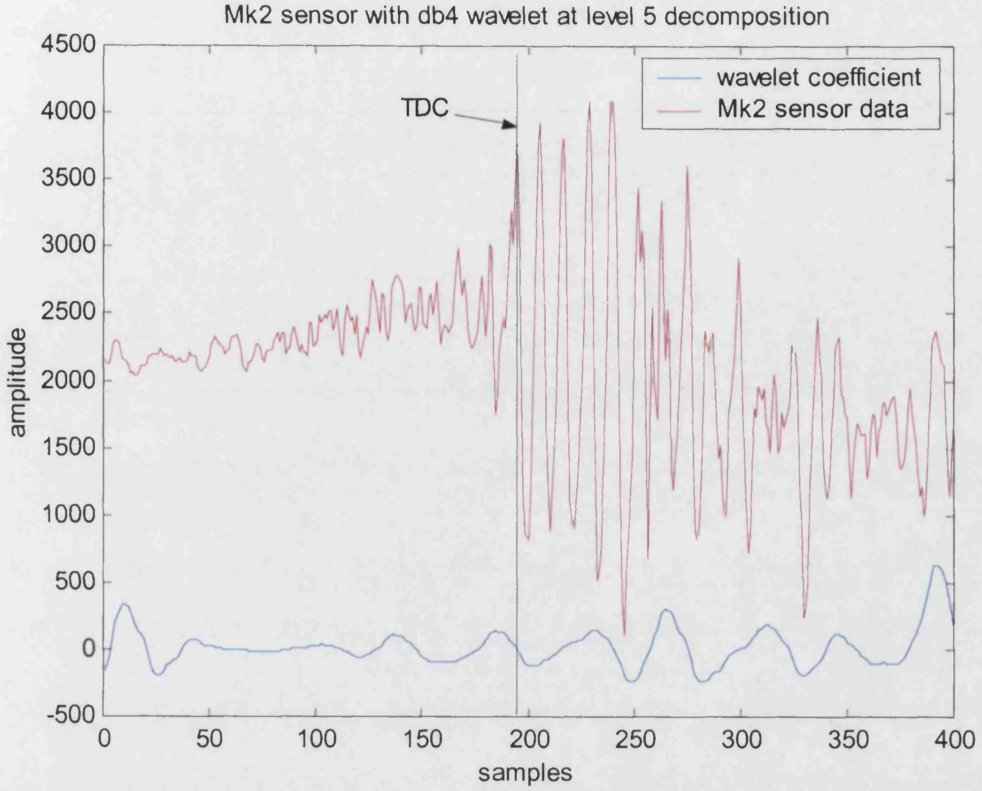

Mk2 sensor with db3 wavelet at level 5 decomposition
wavelet coefficient
Mk2 sensor data
TDC
amplitude
samples
4500
4000
3500
3000
2500
2000
1500
1000
500
0
-500
0
50
100
150
200
250
300
350
400

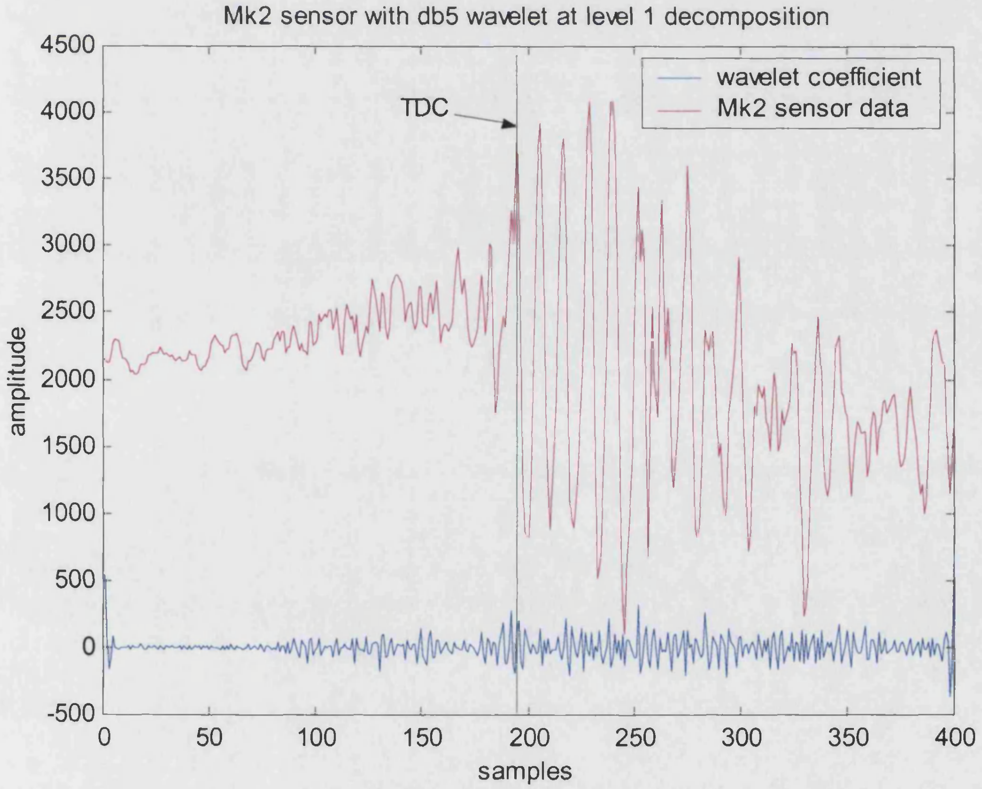

Mk2 sensor with db4 wavelet at level 1 decomposition
4500
4000
3500
3000
2500
2000
1500
1000
500
0
-500
amplitude
0
50
100
150
200
250
300
350
400
samples
TDC
wavelet coefficient
Mk2 sensor data

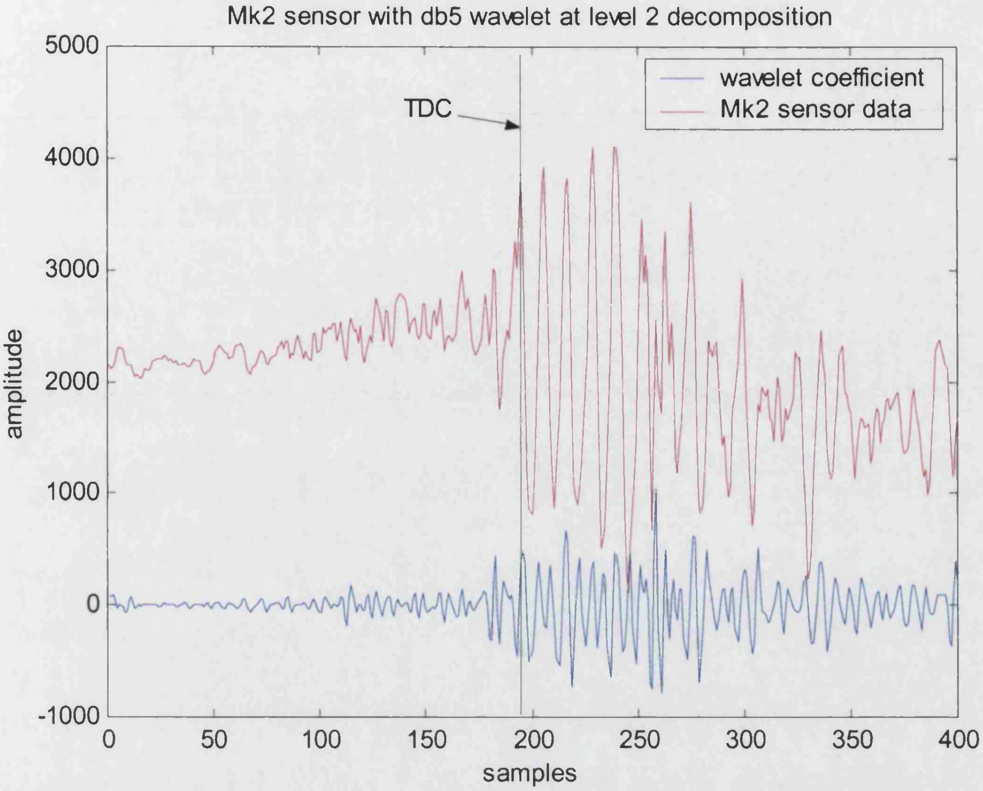

Mk2 sensor with db4 wavelet at level 2 decomposition
5000
4000
3000
2000
1000
0
-1000
amplitude
0
50
100
150
200
250
300
350
400
samples
TDC
wavelet coefficient
Mk2 sensor data

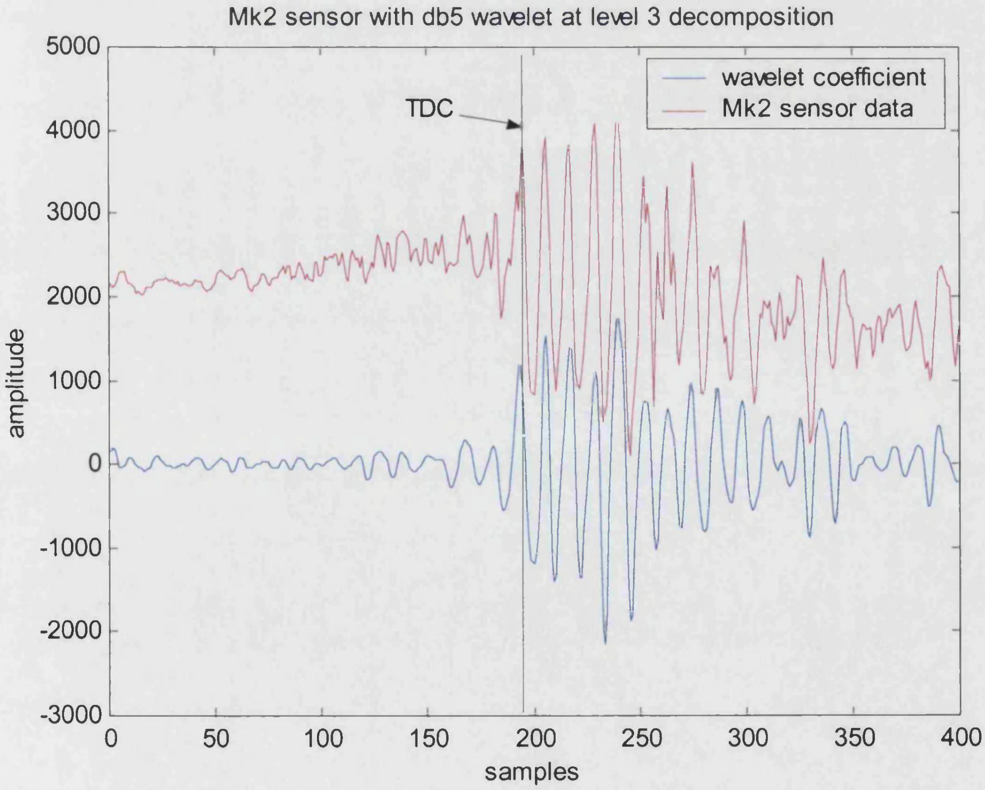

Mk2 sensor with db4 wavelet at level 3 decomposition
wavelet coefficient
Mk2 sensor data
TDC
amplitude
samples
5000
4000
3000
2000
1000
0
-1000
-2000
0
50
100
150
200
250
300
350
400

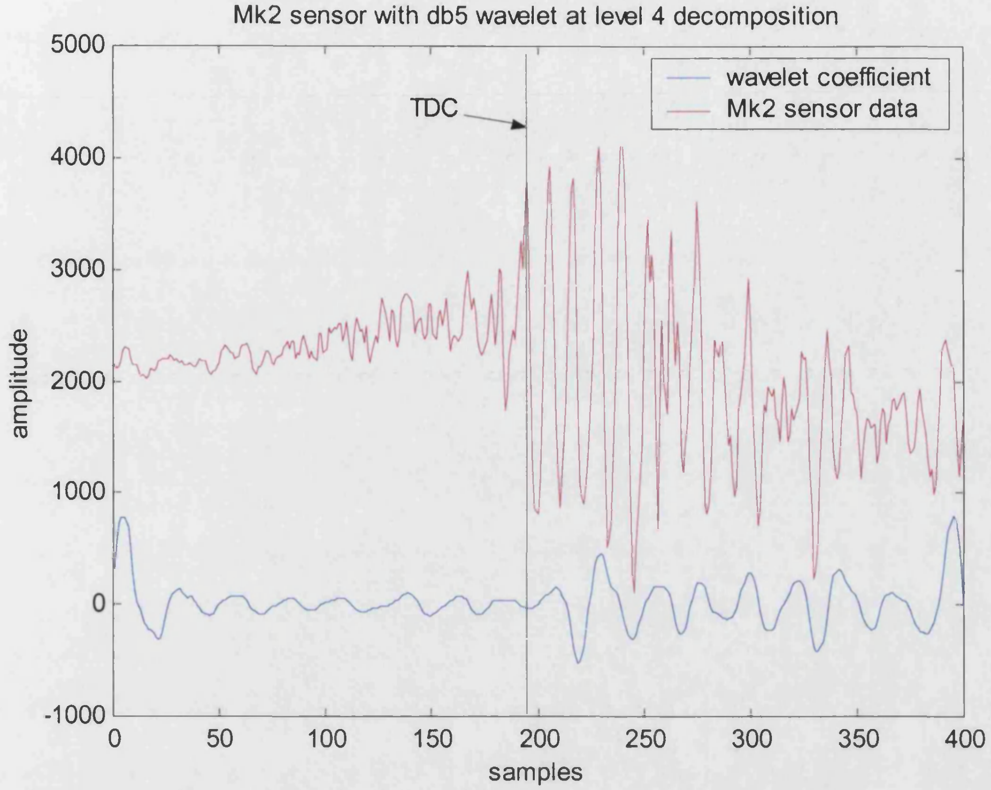

Mk2 sensor with db4 wavelet at level 4 decomposition
wavelet coefficient
Mk2 sensor data
TDC
amplitude
samples
5000
4000
3000
2000
1000
0
-1000
0
50
100
150
200
250
300
350
400

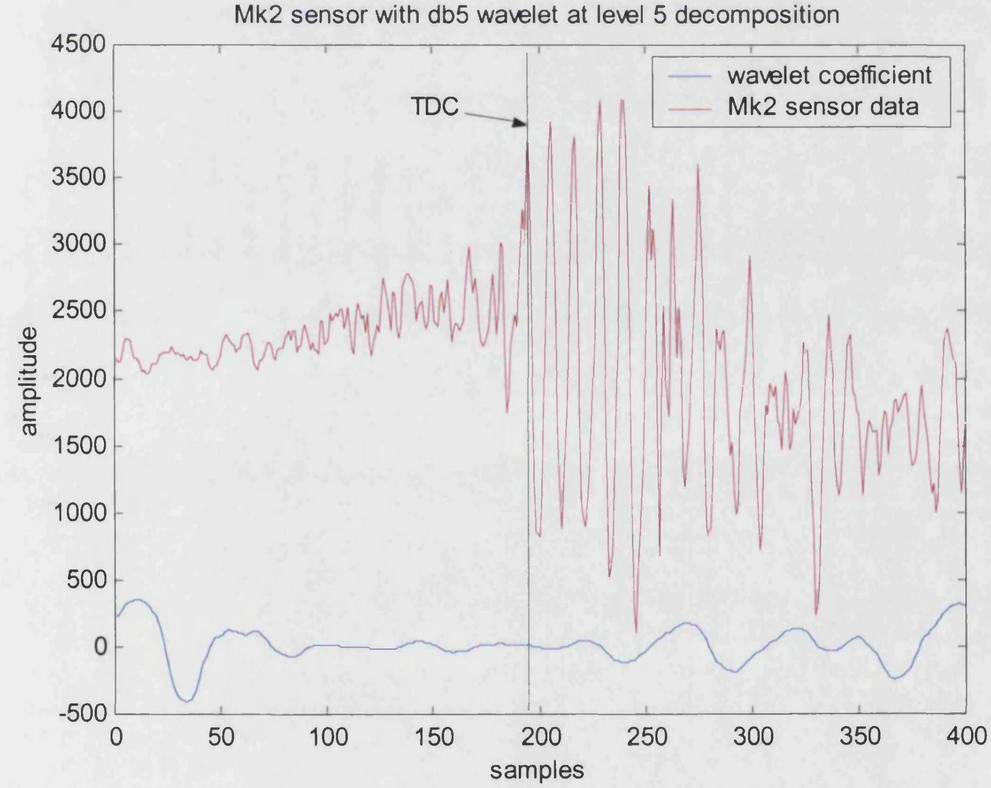

Mk2 sensor with db4 wavelet at level 5 decomposition
wavelet coefficient
Mk2 sensor data
TDC
amplitude
samples
4500
4000
3500
3000
2500
2000
1500
1000
500
0
-500
0
50
100
150
200
250
300
350
400

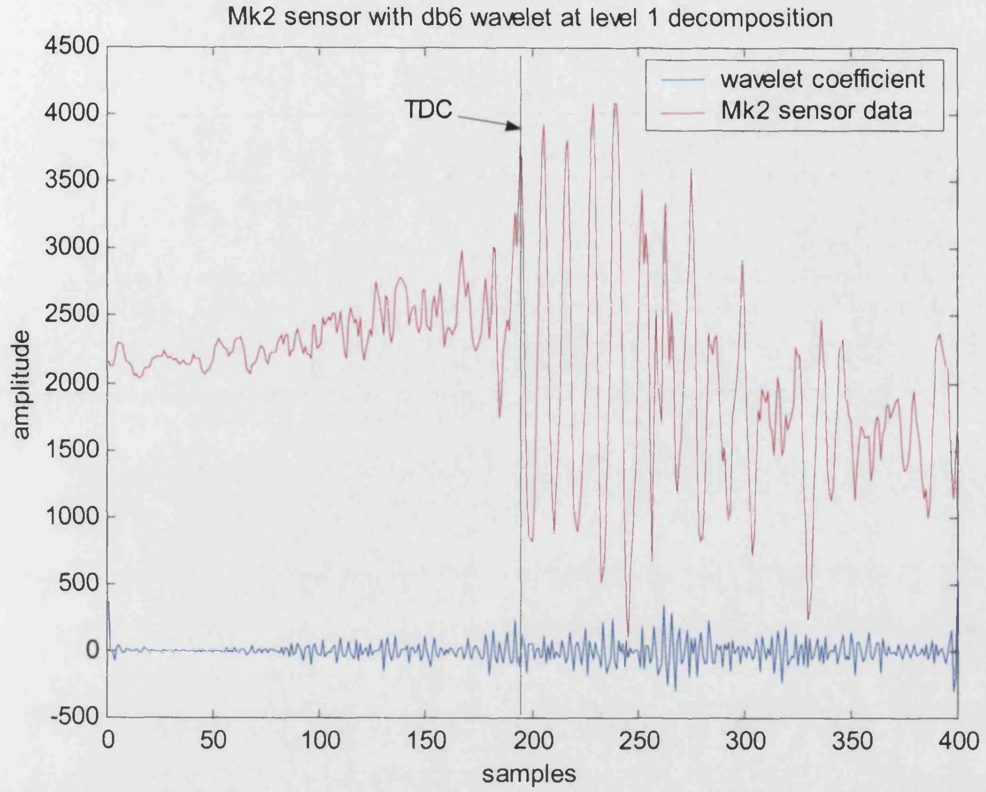

Mk2 sensor with db5 wavelet at level 1 decomposition
wavelet coefficient
Mk2 sensor data
TDC
amplitude
samples
4500
4000
3500
3000
2500
2000
1500
1000
500
0
-500
0
50
100
150
200
250
300
350
400

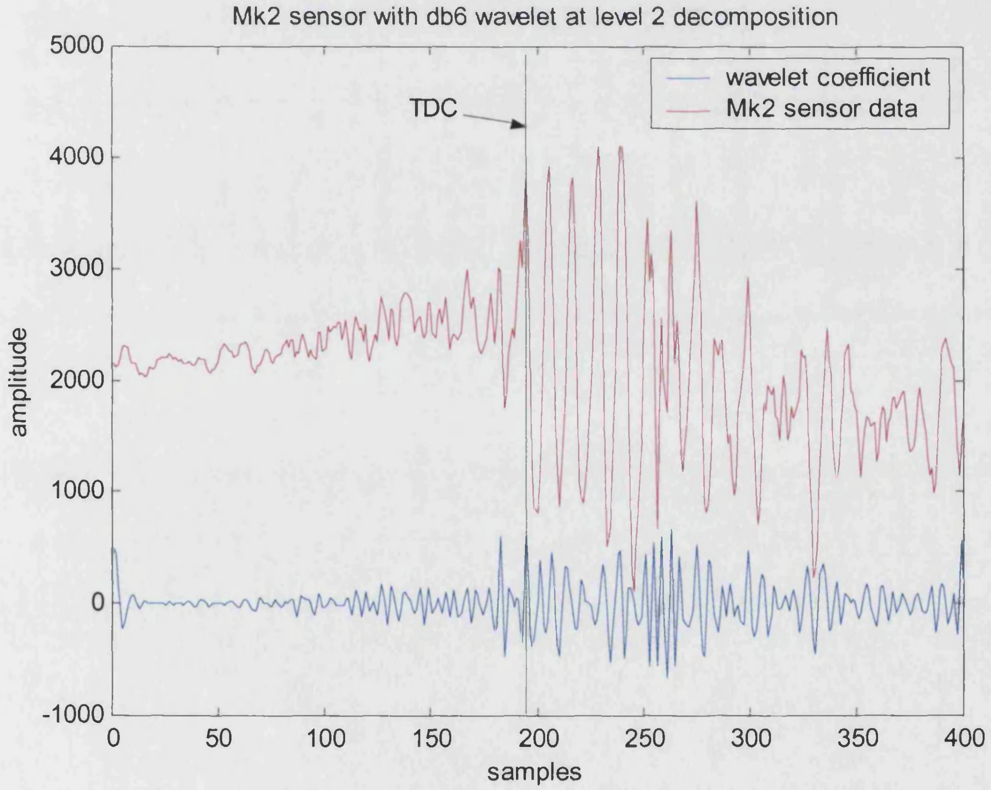

Mk2 sensor with db5 wavelet at level 2 decomposition
wavelet coefficient
Mk2 sensor data
TDC
amplitude
samples
5000
4000
3000
2000
1000
0
-1000
0
50
100
150
200
250
300
350
400

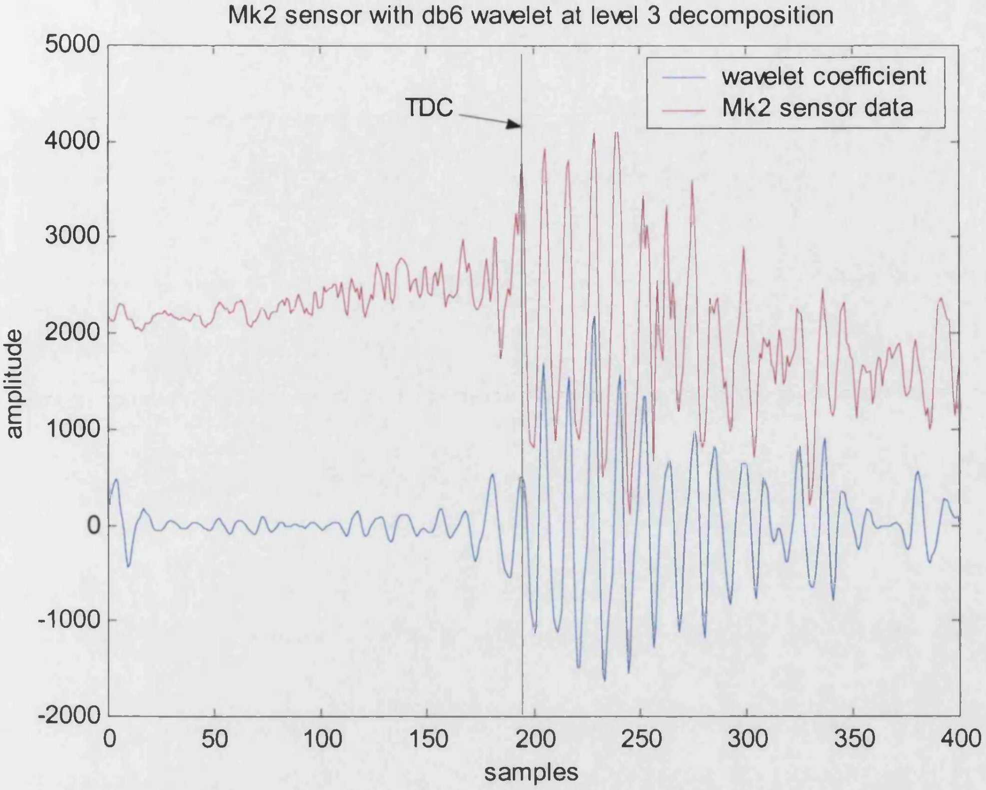

Mk2 sensor with db5 wavelet at level 3 decomposition
wavelet coefficient
Mk2 sensor data
TDC
amplitude
samples
5000
4000
3000
2000
1000
0
-1000
-2000
-3000
0
50
100
150
200
250
300
350
400

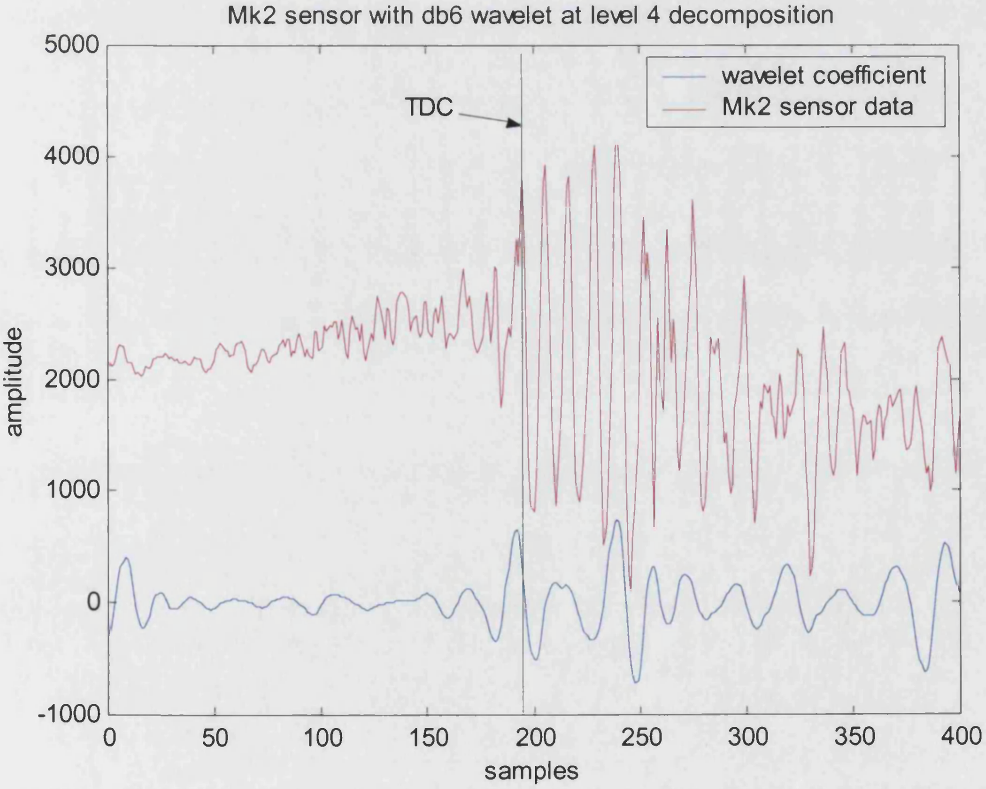

Mk2 sensor with db5 wavelet at level 4 decomposition
wavelet coefficient
Mk2 sensor data
TDC
amplitude
samples
5000
4000
3000
2000
1000
0
-1000
0
50
100
150
200
250
300
350
400

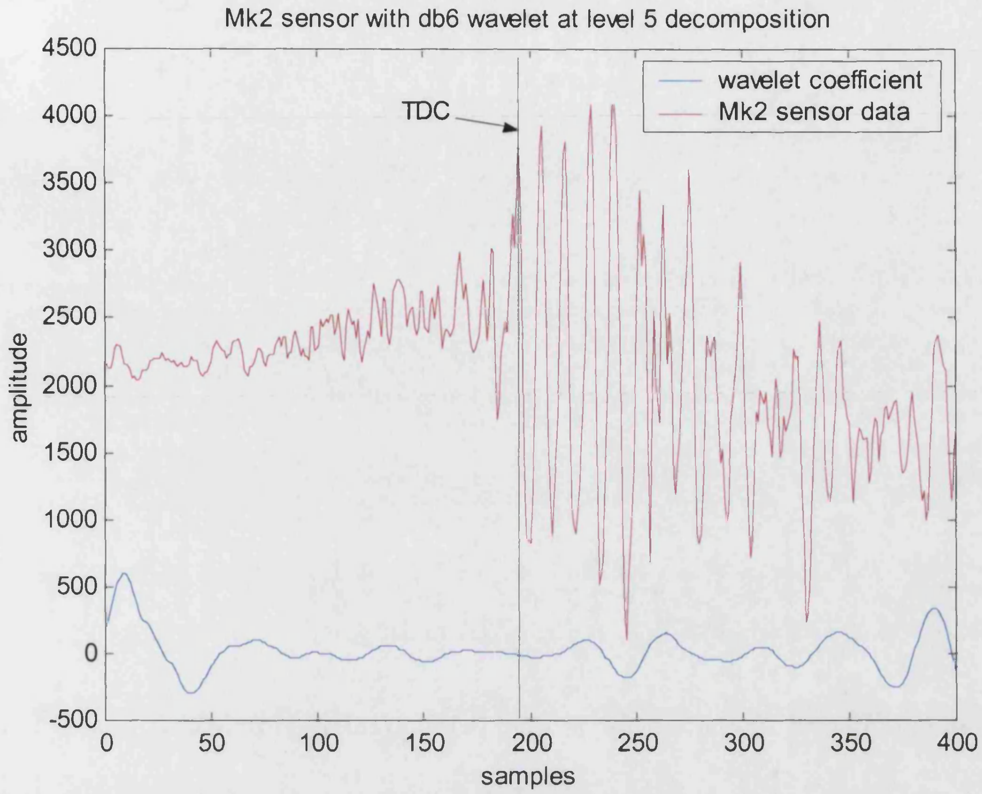

Mk2 sensor with db5 wavelet at level 5 decomposition
wavelet coefficient
Mk2 sensor data
TDC
amplitude
samples
4500
4000
3500
3000
2500
2000
1500
1000
500
0
-500
0
50
100
150
200
250
300
350
400


Mk2 sensor with db6 wavelet at level 1 decomposition


Mk2 sensor with db6 wavelet at level 2 decomposition

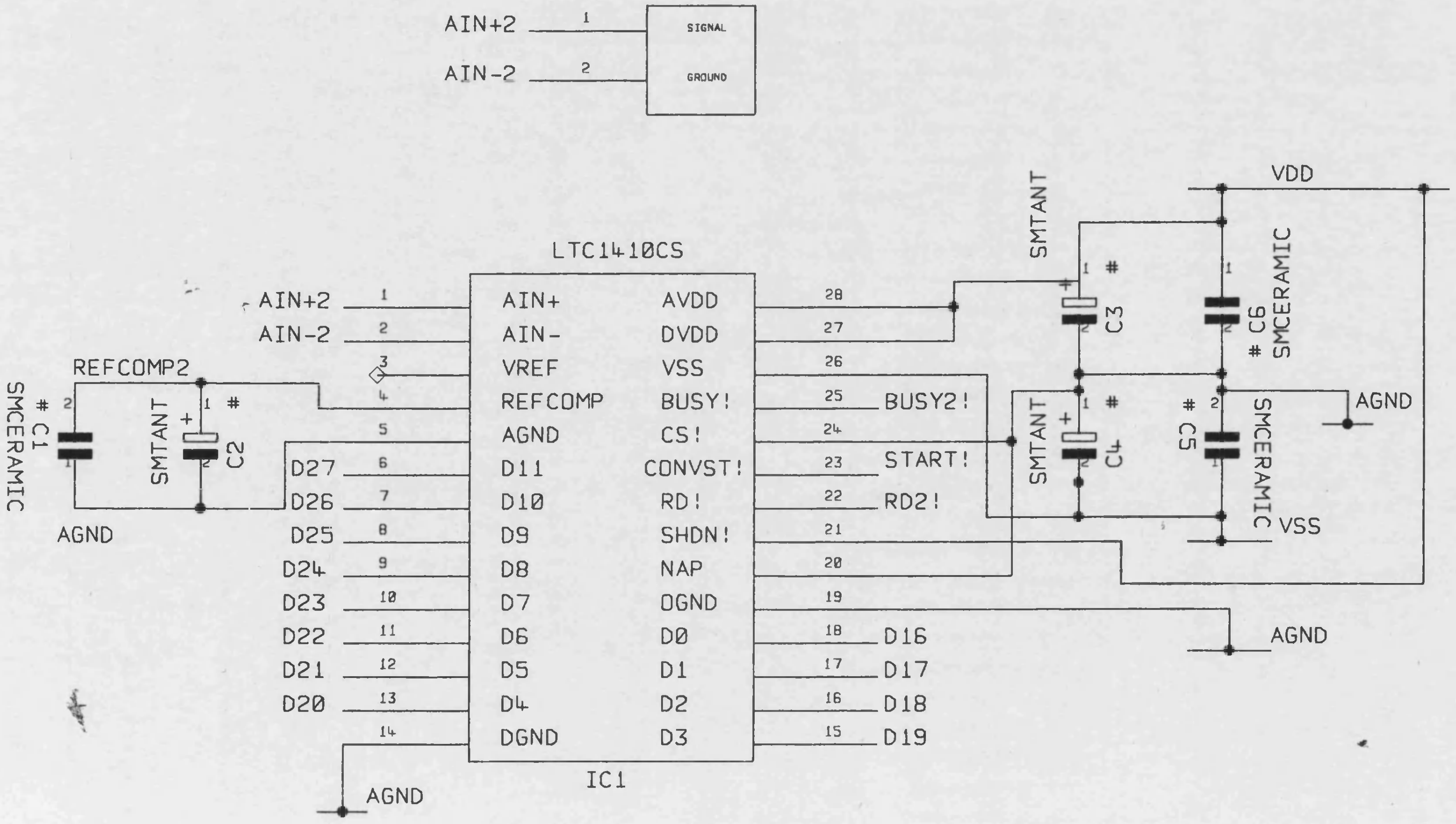

Mk2 sensor with db6 wavelet at level 3 decomposition
wavelet coefficient
Mk2 sensor data
TDC
amplitude
samples
5000
4000
3000
2000
1000
0
-1000
-2000
0
50
100
150
200
250
300
350
400

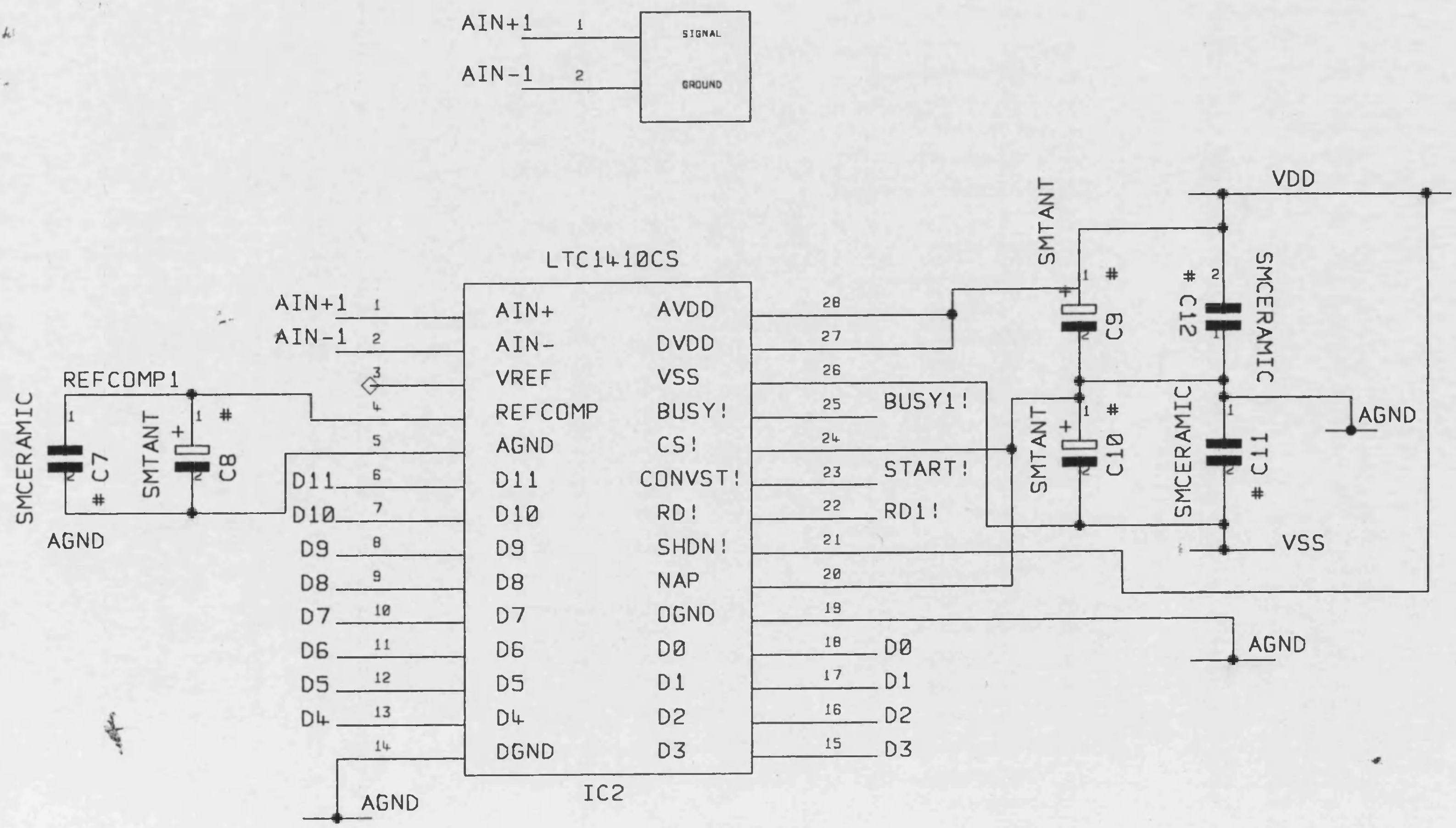

Mk2 sensor with db6 wavelet at level 4 decomposition
wavelet coefficient
Mk2 sensor data
TDC
amplitude
samples
5000
4000
3000
2000
1000
0
-1000
0
50
100
150
200
250
300
350
400

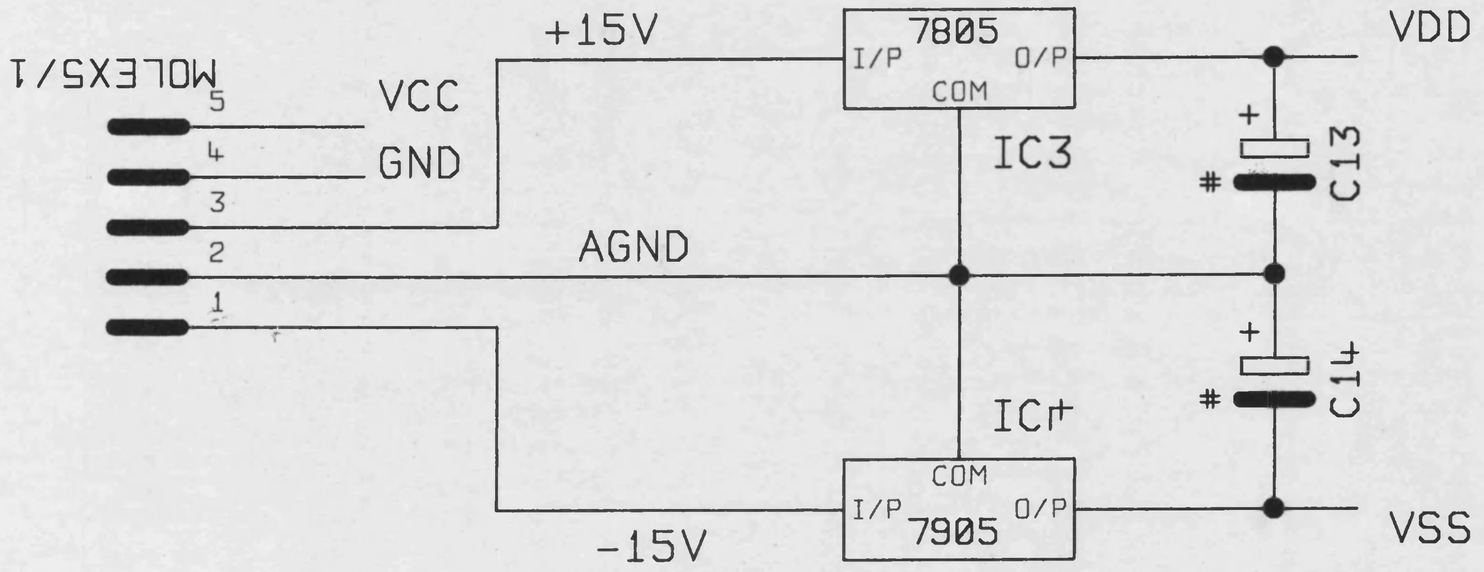

Mk2 sensor with db6 wavelet at level 5 decomposition
wavelet coefficient
Mk2 sensor data
TDC
amplitude
samples
4500
4000
3500
3000
2500
2000
1500
1000
500
0
-500
0
50
100
150
200
250
300
350
400

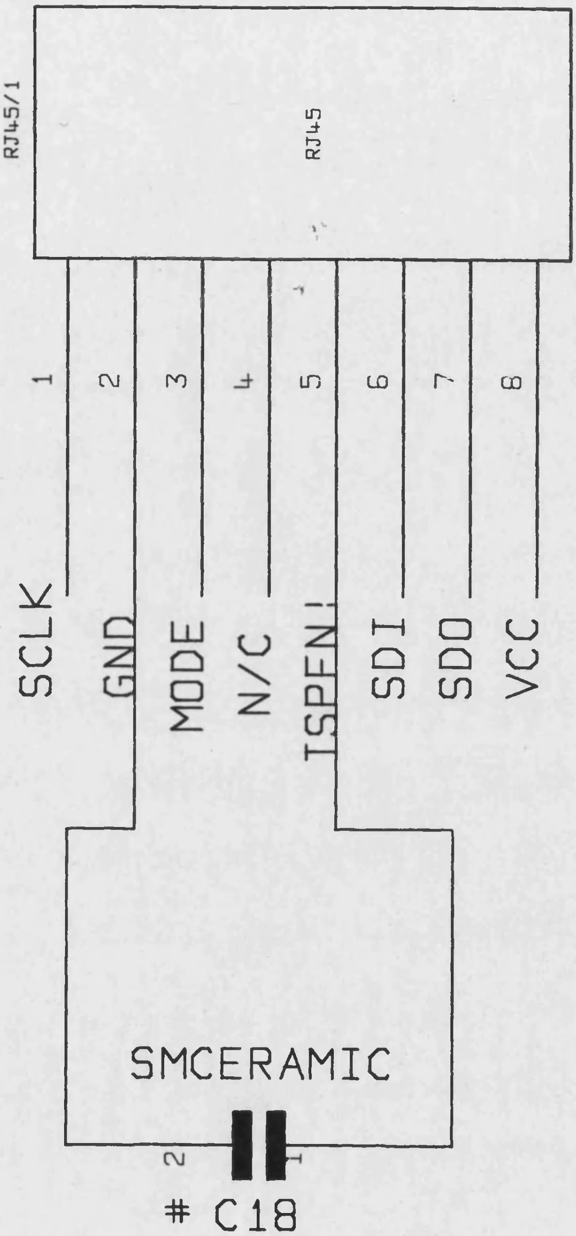
APPENDIX B - SHARC EZ-LAB ANALOG TO DIGITAL CONVERTER SUB-SYSTEM

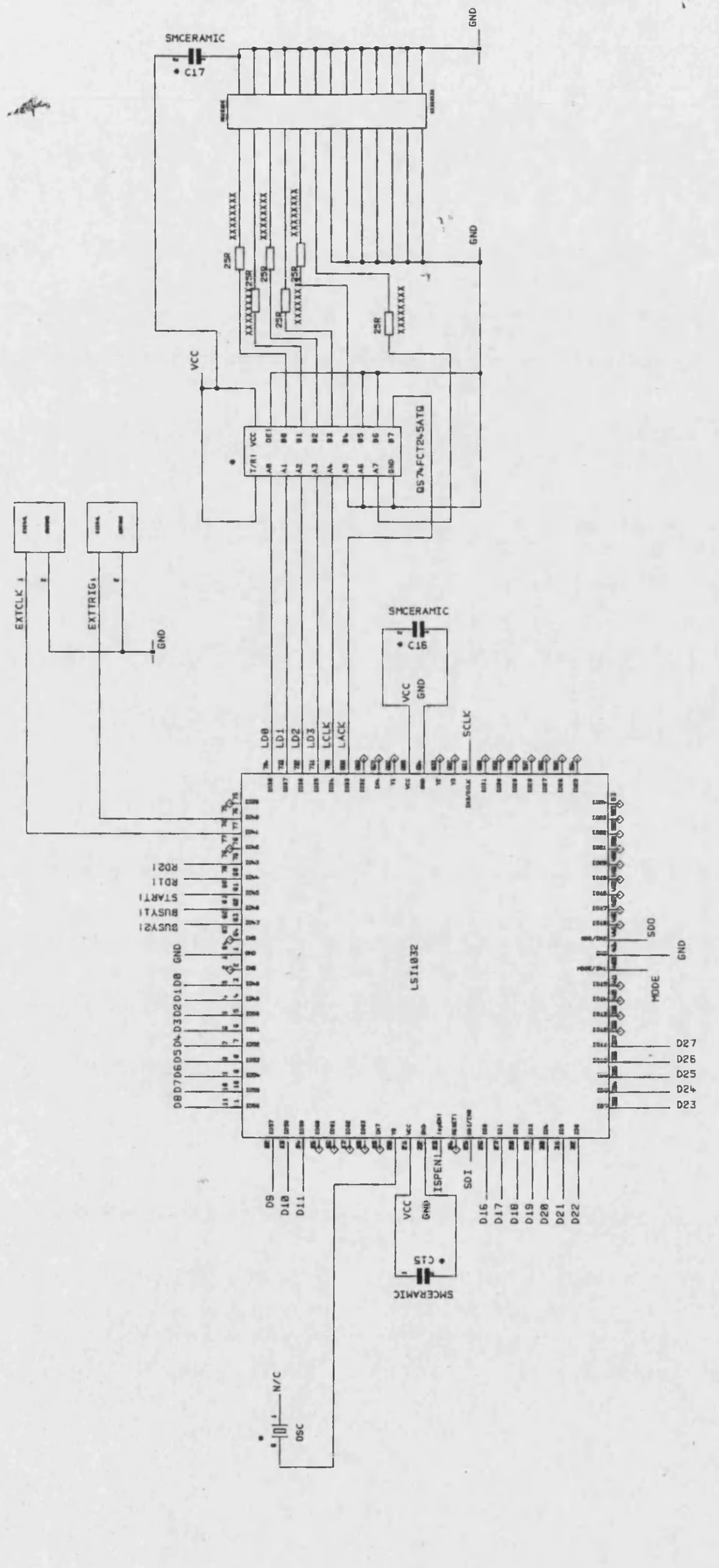
This appendix contains the drawings and program files for the analog to digital converter system. They are presented in the following order: circuit diagrams, pcb layout and epld file.


AIN+2
AIN-2
1
2
SIGNAL
GROUND
LTC1410CS
AIN+2
AIN-2
1 AIN+
2 AIN-
3 VREF
4 REFCOMP
5 AGND
6 D11
7 D10
8 D9
9 D8
10 D7
11 D6
12 D5
13 D4
14 DGND
28 AVDD
27 DVDD
26 VSS
25 BUSY!
24 CS!
23 CONVST!
22 RD!
21 SHDN!
20 NAP
19 DGND
18 D0
17 D1
16 D2
15 D3
IC1
REFCOMP2
# C1
SMCERAMIC
SMTANT
C2
AGND
D27
D26
D25
D24
D23
D22
D21
D20
AGND
BUSY2!
START!
RD2!
D16
D17
D18
D19
VDD
SMTANT
C3
# C6
SMCERAMIC
AGND
SMTANT
C4
# C5
SMCERAMIC
VSS
AGND


AIN+1
1
AIN-1
2
SIGNAL
GROUND
LTC1410CS
AIN+1
AIN-1
AIN+
AIN-
VREF
REFCOMP
AGND
D11
D10
D9
D8
D7
D6
D5
D4
DGND
AVDD
DVDD
VSS
BUSY!
CS!
CONVST!
RD!
SHDN!
NAP
DGND
D0
D1
D2
D3
1
2
3
4
5
6
7
8
9
10
11
12
13
14
28
27
26
25
24
23
22
21
20
19
18
17
16
15
D11
D10
D9
D8
D7
D6
D5
D4
BUSY1!
START!
RD1!
D0
D1
D2
D3
IC2
AGND
REFCOMP1
SMCERAMIC
C7
SMTANT
C8
AGND
SMTANT
C9
SMTANT
C10
SMCERAMIC
C12
SMCERAMIC
C11
VDD
AGND
VSS
AGND


MOLEX5/1
5
4
3
2
1
VCC
GND
+15V
AGND
-15V
7805
I/P
O/P
COM
IC3
IC4
COM
I/P
O/P
7905
VDD
VSS
+
#
C13
+
#
C14


RJ45/1
RJ45
1
2
3
4
5
6
7
8
SCLK
GND
MODE
N/C
ISPEN!
SDI
SDO
VCC
SMCERAMIC
2
1
# C18



LSI1032
QS74FCT245ATQ
SMCERAMIC
C15
C16
C17
OSC
N/C
VCC
GND
ISPEN1
SDI
SDO
SCLK
MODE
D9
D10
D11
D16
D17
D18
D19
D20
D21
D22
D23
D24
D25
D26
D27
D8D7D6D5D4D3D2D1D0
BUSY2!
BUSY1!
START!
RD1!
RD2!
LD0
LD1
LD2
LD3
LCLK
LACK
EXTCLK
EXTTRIG
25R
XXXXXXXX
T/R1
OE1
A0
A1
A2
A3
A4
A5
A6
A7
B0
B1
B2
B3
B4
B5
B6
B7

EPLD FILES FOR ADC CONVERTER BOARD

```
// Mon Jun 03 14:30:59 1996
// C:\RESEARCH\CRIMSON\DIGITIV2\EPLDS\LINKINTF.LDF generated using
Lattice pDS Version 2.8

LDF 1.00.00 DESIGNLDF;
DESIGN sharc link interface;
REVISION rev 1;
AUTHOR chris mobley;
PROJECTNAME crimsoniv2;
DESCRIPTION
link interfacec for sharc processor.;

PART ispLSI1032-80LJ84;

OPTION PULLUP ON;

DECLARE
END;  //DECLARE

SYM GLB  A7  1  ;
SIGTYPE BUSY OUT;
SIGTYPE START OUT;
EQUATIONS
BUSY = BUSY1 & BUSY2;
START = !G3&!G2&!G1&G0;
END
END;

SYM GLB  D7  1  ;
SIGTYPE [A0..A3] REG OUT;
EQUATIONS
A0.ptclk = SRCLOCK;
A0 =VCC;
A1 = A0&!LOAD # D24&LOAD;
A2 = A1&!LOAD # D20&LOAD;
A3 = A2&!LOAD # D16&LOAD;
END;
END;

SYM GLB  D6  1  ;
SIGTYPE [A4..A7] REG OUT;
EQUATIONS
A4.ptclk = SRCLOCK;
A4 = A3&!LOAD # VCC&LOAD;
A5 = A4&!LOAD # D8&LOAD;
A6 = A5&!LOAD # D4&LOAD;
A7 = A6&!LOAD # D0&LOAD;
END;
END;

SYM GLB  D5  1  ;
SIGTYPE [B0..B3] REG OUT;
EQUATIONS
```

```
B0.ptclk = SRCLOCK;
B0 = VCC;
B1 = B0&!LOAD # D25&LOAD;
B2 = B1&!LOAD # D21&LOAD;
B3 = B2&!LOAD # D17&LOAD;
END;
END;

SYM GLB  D4  1  ;
SIGTYPE [B4..B7] REG OUT;
EQUATIONS
B4.ptclk = SRCLOCK;
B4 = B3&!LOAD # VCC&LOAD;
B5 = B4&!LOAD # D9&LOAD;
B6 = B5&!LOAD # D5&LOAD;
B7 = B6&!LOAD # D1&LOAD;
END;
END;

SYM GLB  D3  1  ;
SIGTYPE [C0..C3] REG OUT;
EQUATIONS
C0.ptclk =SRCLOCK;
C0 = EXTTRIG;
C1 = C0&!LOAD # D26&LOAD;
C2 = C1&!LOAD # D22&LOAD;
C3 = C2&!LOAD # D18&LOAD;
END;
END;

SYM GLB  D2  1  ;
SIGTYPE [C4..C7] REG OUT;
EQUATIONS
C4.ptclk = SRCLOCK;
C4 = C3&!LOAD # VCC&LOAD;
C5 = C4&!LOAD # D10&LOAD;
C6 = C5&!LOAD # D6&LOAD;
C7 = C6&!LOAD # D2&LOAD;
END;
END;

SYM GLB  D1  1  ;
SIGTYPE [E0..E3] REG OUT;
EQUATIONS
E0.ptclk = SRCLOCK;
E0 = EXTCLK;
E1 = E0&!LOAD # D27&LOAD;
E2 = E1&!LOAD # D23&LOAD;
E3 = E2&!LOAD # D19&LOAD;
END;
END;

SYM GLB  D0  1  ;
SIGTYPE [E4..E7] REG OUT;
EQUATIONS
E4.ptclk = SRCLOCK;
E4 = E3&!LOAD # VCC&LOAD;
```

```
E5 = E4&!LOAD # D11&LOAD;
E6 = E5&!LOAD # D7&LOAD;
E7 = E6&!LOAD # D3&LOAD;
END;
END;

SYM GLB  A0  1  ;
SIGTYPE [G0..G3] REG OUT;
EQUATIONS
G0.CLK = CLOCK;
G0 =   !G3&!G2&!G1&!G0&EXTCLK
      #!G3&!G2&!G1&G0&!BUSY
      #!G3&!G2&G1&!G0
      #!G3&G2&!G1&!G0
      #!G3&G2&G1&!G0
      #G3&!G2&!G1&!G0
      #!G3&!G2&G1&G0&!LACK
      #G3&!G2&G1&G0&!LACK;

G1 =   !G3&!G2&!G1&G0&BUSY
      #!G3&!G2&G1&!G0
      #!G3&G2&!G1&G0
      #!G3&G2&G1&!G0
      #G3&!G2&!G1&G0
      #!G3&!G2&G1&G0&!LACK
      #G3&!G2&G1&G0&!LACK;

G2 =   !G3&!G2&G1&G0&LACK
      #!G3&G2&!G1&!G0
      #!G3&G2&!G1&G0
      #!G3&G2&G1&!G0;

G3 =   !G3&G2&G1&G0
      #G3&!G2&!G1&!G0
      #G3&!G2&!G1&G0
      #!G3&!G2&G1&G0&!LACK
      #G3&!G2&G1&G0&!LACK;
END;
END;

SYM GLB  A1  1  ;
SIGTYPE LOAD OUT;
SIGTYPE SRCLOCK OUT;
SIGTYPE RD1 OUT;
SIGTYPE RD2 OUT;
EQUATIONS
LOAD = !G3&!G2&G1&!G0;
RD1 = !G3&!G2&G1&!G0;
RD2 = !G3&!G2&G1&!G0;
SRCLOCK=             !(!G3&!G2&!G1&!G0
            #!G3&!G2&!G1&G0
            #!G3&!G2&G1&!G0);

END;
END;

SYM IOC  IO0  1  ;
```

```
XPIN IO DIN0 LOCK 3;
IB11(D0,DIN0);
END;

SYM IOC  IO1  1  ;
XPIN IO DIN1 LOCK 4;
IB11(D1,DIN1);
END;

SYM IOC  IO2  1  ;
XPIN IO DIN2 LOCK 5;
IB11(D2,DIN2);
END;

SYM IOC  IO3  1  ;
XPIN IO DIN3 LOCK 6;
IB11(D3,DIN3);
END;

SYM IOC  IO4  1  ;
XPIN IO DIN4 LOCK 7;
IB11(D4,DIN4);
END;

SYM IOC  IO5  1  ;
XPIN IO DIN5 LOCK 8;
IB11(D5,DIN5);
END;

SYM IOC  IO6  1  ;
XPIN IO DIN6 LOCK 9;
IB11(D6,DIN6);
END;

SYM IOC  IO7  1  ;
XPIN IO DIN7 LOCK 10;
IB11(D7,DIN7);
END;

SYM IOC  IO8  1  ;
XPIN IO DIN8 LOCK 11;
IB11(D8,DIN8);
END;

SYM IOC  IO9  1  ;
XPIN IO DIN9 LOCK 12;
IB11(D9,DIN9);
END;

SYM IOC  IO10  1  ;
XPIN IO DIN10 LOCK 13;
IB11(D10,DIN10);
END;

SYM IOC  IO11  1  ;
XPIN IO DIN11 LOCK 14;
IB11(D11,DIN11);
```

```
END;

SYM IOC  IO16  1  ;
XPIN IO DIN16 LOCK 26;
IB11(D16,DIN16);
END;

SYM IOC  IO17  1  ;
XPIN IO DIN17 LOCK 27;
IB11(D17,DIN17);
END;

SYM IOC  IO18  1  ;
XPIN IO DIN18 LOCK 28;
IB11(D18,DIN18);
END;

SYM IOC  IO19  1  ;
XPIN IO DIN19 LOCK 29;
IB11(D19,DIN19);
END;

SYM IOC  IO20  1  ;
XPIN IO DIN20 LOCK 30;
IB11(D20,DIN20);
END;

SYM IOC  IO21  1  ;
XPIN IO DIN21 LOCK 31;
IB11(D21,DIN21);
END;

SYM IOC  IO22  1  ;
XPIN IO DIN22 LOCK 32;
IB11(D22,DIN22);
END;

SYM IOC  IO23  1  ;
XPIN IO DIN23 LOCK 33;
IB11(D23,DIN23);
END;

SYM IOC  IO24  1  ;
XPIN IO DIN24 LOCK 34;
IB11(D24,DIN24);
END;

SYM IOC  IO25  1  ;
XPIN IO DIN25 LOCK 35;
IB11(D25,DIN25);
END;

SYM IOC  IO26  1  ;
XPIN IO DIN26 LOCK 36;
IB11(D26,DIN26);
END;
```

```
SYM IOC  IO27  1  ;
XPIN IO DIN27 LOCK 37;
IB11(D27,DIN27);
END;

SYM IOC  IO32  1  ;
XPIN IO STARTOUT LOCK 81;
OB11(STARTOUT,START);
END;

SYM IOC  IO12  1  ;
XPIN IO BUSYIN1 LOCK 82;
IB11(BUSY1,BUSYIN1);
END;

SYM IOC  IO13  1  ;
XPIN IO BUSYIN2 LOCK 83;
IB11(BUSY2,BUSYIN2);
END;

SYM IOC  IO14  1  ;
XPIN IO EXTCLKIN LOCK 77;
IB11(EXTCLK,EXTCLKIN);
END;

SYM IOC  IO15  1  ;
XPIN IO EXTTRIGIN LOCK 76;
IB11(EXTTRIG,EXTTRIGIN);
END;

SYM IOC  IO47  1  ;
XPIN IO LD0 LOCK 74;
OB11(LD0,A7);
END;

SYM IOC  IO46  1  ;
XPIN IO LD1 LOCK 73;
OB11(LD1,B7);
END;

SYM IOC  IO45  1  ;
XPIN IO LD2 LOCK 72;
OB11(LD2,C7);
END;

SYM IOC  IO44  1  ;
XPIN IO LD3 LOCK 71;
OB11(LD3,E7);
END;

SYM IOC  IO43  1  ;
XPIN IO LCLKOUT LOCK 70;
OB11(LCLKOUT,SRCLOCK);
END;

SYM IOC  IO42  1  ;
XPIN IO LACKIN LOCK 69;
```

```
IB11(LACK,LACKIN);
END;

SYM IOC  IO34  1  ;
XPIN IO RDOUT1 LOCK 80;
OB11(RDOUT1,RD1);
END;

SYM IOC  IO33  1  ;
XPIN IO RDOUT2 LOCK 79;
OB11(RDOUT2,RD2);
END;

SYM IOC  Y0  1  ;
XPIN CLK EXTCLOCK LOCK 20;
IB11(CLOCK,EXTCLOCK);
END;
END;  //LDF DESIGNLDF
```

APPENDIX C - ' COMPASS' - RAPID PROTOTYPING DSP SYSTEM

This appendix contains the documents used to create the 'Compass' rapid prototyping system. It starts with an overview of the system, which was written as a user guide introduction. The next section contains the circuit diagrams and printed circuit board diagrams for the three boards which make up the 'Compass' system. Finally a flow diagram and a section of code is presented, to give an appreciation of some of the software ported to the system.

1. OVERVIEW

Compass has been designed to provide a flexible system for data acquisition and control. The system consists of a base module (Input Output Processor - IOP), and a number of plug-in modules. This enables a cost effective system to be quickly built to the user requirements.

The flexible design of Compass enables it to be used for a wide range of control and acquisition tasks. In a simple fixed-point control strategy the low cost, base module can be used as a stand alone controller. If a larger Floating Point model strategy is required then the Floating Point Module (FPM) can be added.

The Compass system is enclosed in equipment housings of various sizes, depending upon configuration. Three sizes of enclosure are available, as well as bare boards. The smallest enclosure measures 225x110x35 mm.

2. Introduction to FPM

Industry is turning to rapid prototyping systems to help in the development of control and acquisition systems. Previously control systems have been implemented using a variety of techniques, from simple analog systems to complicated, Personal Computer based, digital systems. Each solution required a detailed knowledge of the implementation technology before a strategy for control could be established. Rapid prototyping aims to remove the

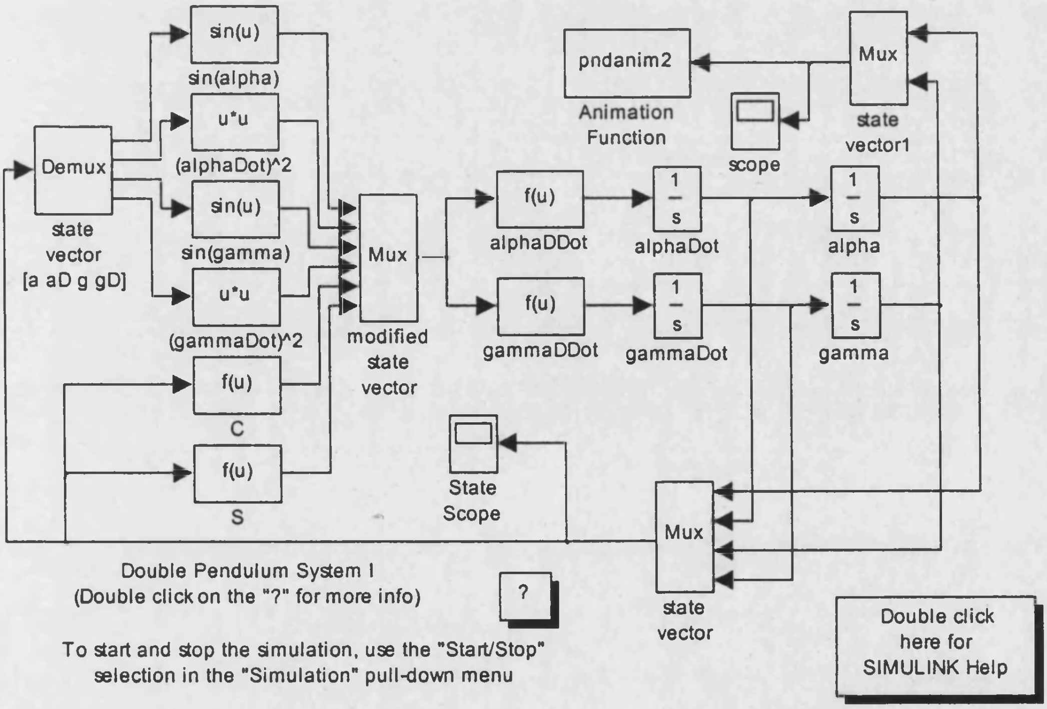


Figure 1, Simulation model

underlying implementation technology from the development of the solution, often called 'problem abstraction'. The systems being developed should allow a model of the process to be built independently of the electronic hardware.

Model development packages offer the designer a graphical interface to construct a control system using icons which represent standard (and custom) signal processing blocks. **Figure 1** shows such a model.

The icons contain standard C code to perform the operation. The icons are 'wired' together via logical connections, which produce the control flow. Once the model has been constructed it can be simulated or converted into C code and downloaded into an embedded rapid prototyping system. By using this system, 'what if' solutions can be tried and rapid design cycles completed.

In existing systems the connection of the model to the outside world is achieved by using special icons representing the underlying technology being used. This requires the correct 'libraries' for specific hardware, and somewhat negates the 'problem abstraction' goals which the system promises to deliver.

In the Compass system 'Memory Mapped Input Output' is used. This enables the designer to develop the model independently of the underlying hardware and to achieve the goal of 'hardware abstraction'.

3. Compass system architecture

The architecture of Compass has been designed to separate the model requirements from the Input Output requirements. To achieve this, two tightly coupled processors are used. One processor has responsibility for the Input & Output of the system and the other executes the model.

The capabilities of the processors are tailored to the specific task they must perform. The IOP is a complete 'system on a chip' with extensive capabilities to control the interfaces to the real world. The FMP is a high performance Floating Point Processor board, which can achieve 120M to 1200M Flops performance.
The interface between the processors is a large buffer of memory. It contains 512k bytes of read/write dual port memory. Both processors can read or write to the buffer simultaneously.

In the FMP access to the real world is via the buffer memory and the IOP connects the Model output to the physical world. The FMP is presented with a unified memory architecture, which is re-configurable, this is shown in Figure 2.

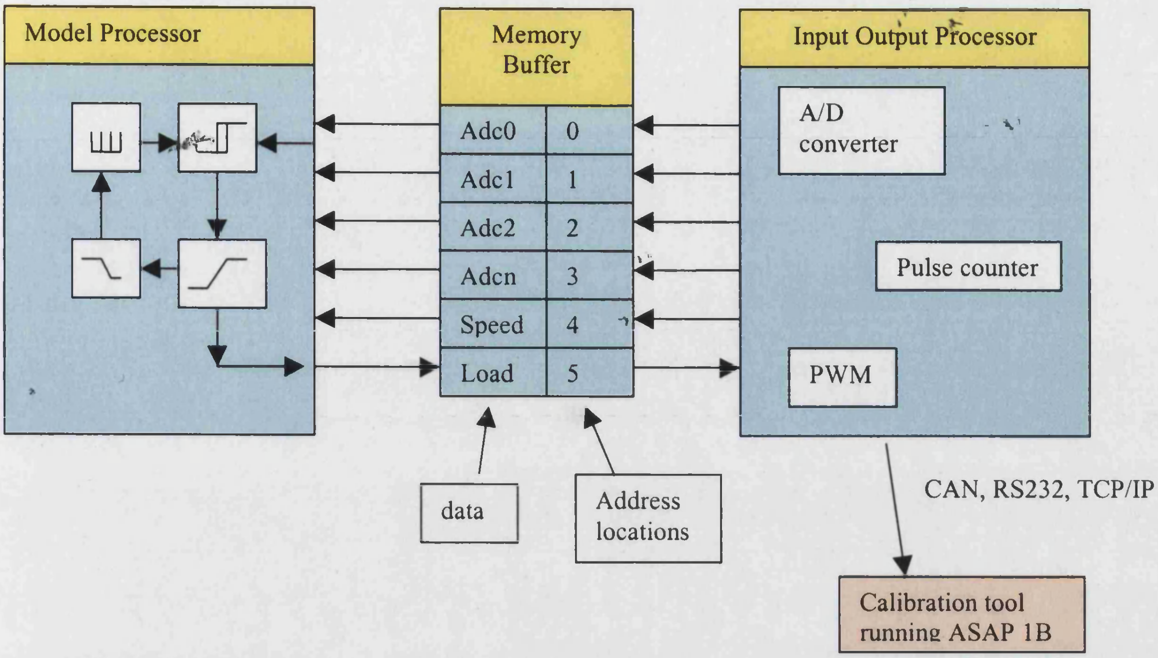


Figure 2 Memory rchitecture

The address locations in the memory buffer can be independent of the system generated by the model package. This is possible because the loading of the Memory Buffer data is controlled by the IOP, which can generate an offset from a specific address location. The order of the data cannot be changed as this is fixed by the FMP.

Other memory mapped data can also be incorporated into the ECU32. The IOP will receive the data and map it to the MP as required.

4. Start up

The start sequence for the system will be defined as follows

- At power up, the IOP performs system checks, and holds the MP in reset.
- The IOP starts the MP from internal Flash.
- The MP copies the model from Flash memory into Sram.
- The MP runs the model from Sram

5. Generating a new Model

Figure 3 shows the design cycle for generating a new model. The main model file is the C file.This can be built using conventional programming techniques, Object Oriented Programming, or auto code generators.
For conventional programming standard pointers will be used to access the memory buffer, or library functions will be available to provide a higher level of problem abstraction. For OOP, a class library will be developed to enable objects to manipulate the memory buffer.

Auto code generators will use the standard memory mapped (pointers) techniques, and / or dedicated functions included in a library.

Once the C source file has been created, it is cross-compiled into Sharc executable by the use of the GNU ADSP21060 compiler. The executable is then sent via CAN or RS232 to the IOP for download into the MP.

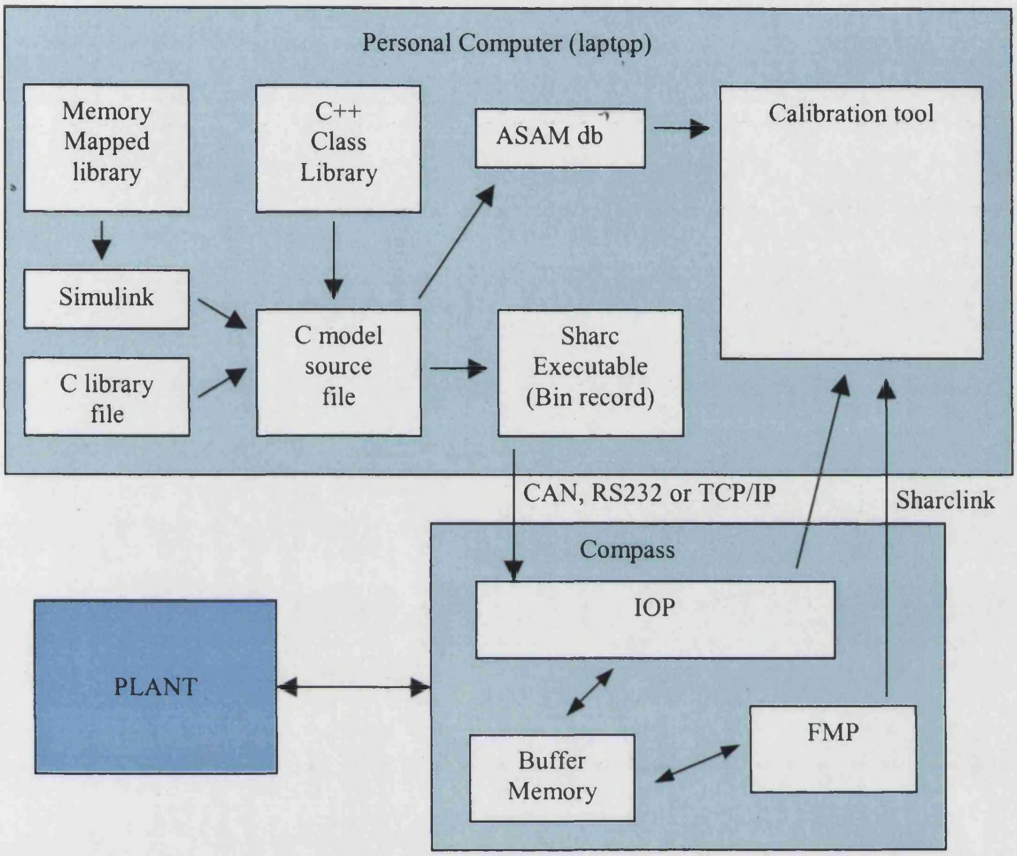


Figure 3 Design Cycle

When a command is issued to the IOP to accept a new model the following processes occur:

- A command is received by the IOP to down load a new model.
- The IOP stops the FMP.
- The IOP downloads the new model into the FMP Sram.
- A run command is received and the IOP starts the FMP.
- Commands can be issued to save the new model in Flash, or reset the FMP.

6. Modular system

The complete system offers floating point model development using standard C or graphical automation tools. The Compass system is available as separate boards and can be built up as required. For a simple control problem the base module (IOP) is capable of running the strategies.

To program the base module, C is used. This is then complied into the code required for the base module, using industry standard cross compilers, and then downloaded via RS232, CAN or TCP/IP.

7. ASAP standards

The IOP is fully ASAP 1B compliant and implements this over CAN, RS232 and TCP/IP (IOP Rev 1.3).

The Compass system can be integrated with standard CAN products or can be used standalone.

8. Data acquisition

There are several ways to perform analysis on the system during embedded control. The MP has 6 high speed parallel link ports, which can be used to transfer data from Sram to an external system. The sharclinks can provide a throughput of 240Mbytes per second. The IOP has CAN and RS232 to enable internal data to be transferred to an external system. On board Flash can be used to store acquisition data, which can be later retrieved.

9. Specifications

9.1 General operating conditions

Normal supply voltage	13.8V (nominal)
Supply range	7V to 30V
Operating temperature range	-40°C to +85°C

9.2 IOP Specifications

Motorola 68376 at 20Mhz (CPU 32 core)

This has the following embedded modules:

- Controller Area Network. Rev 2.0B
- RS232 serial port
- Queued Serial Module (QSPI)
- ADC module (10bit 32 channels)
- Time Processor Unit (16 channels)
- Counter Timer Unit (9 channels)
- System Integration Module

Main circuit board features:

- 8 layer PCB, surface mount components, single eurocard (100x160mm).
- 512K Byte Flash memory
- 2M Byte Sram
- CAN physical interface
- RS232 physical interface
- 10baseT Ethernet running TCP/IP (IOP Rev 1.3)

9.3 FPM Specifications

Floating Point Module daughter board features:

- Analog Devices ADSP21060 'Sharc' processor, 40Mhz, rated at 120 Mflops peak, 80Mflops sustained, 40MIPS. 4M bits internal Sram.
- Analog Devices ADSP21260 Sharc II, 200Mhz, 1200Mflops peak (under development).
- 4.5M Byte 0wait state Sram
- 8M Byte Flash memory
- 512K Byte Dual Port Memory
- 6 link ports
- 2 serial ports at 40M bits/s

9.4 Signal Conditioning Board

Analog input

- 15 channel 10 bit A/D. Fixed input range at 0-5v. max sample rate 10KHz aggregate. External clock.
- 16 channel 12 bit A/D. Software selectable input range : ± 10mv, ±100mV, ±1v and ±10v. Sample rate 800KHz aggregate.

Analog output

- 8 channel 12 bit D/A, 100ma, 0-10v output.

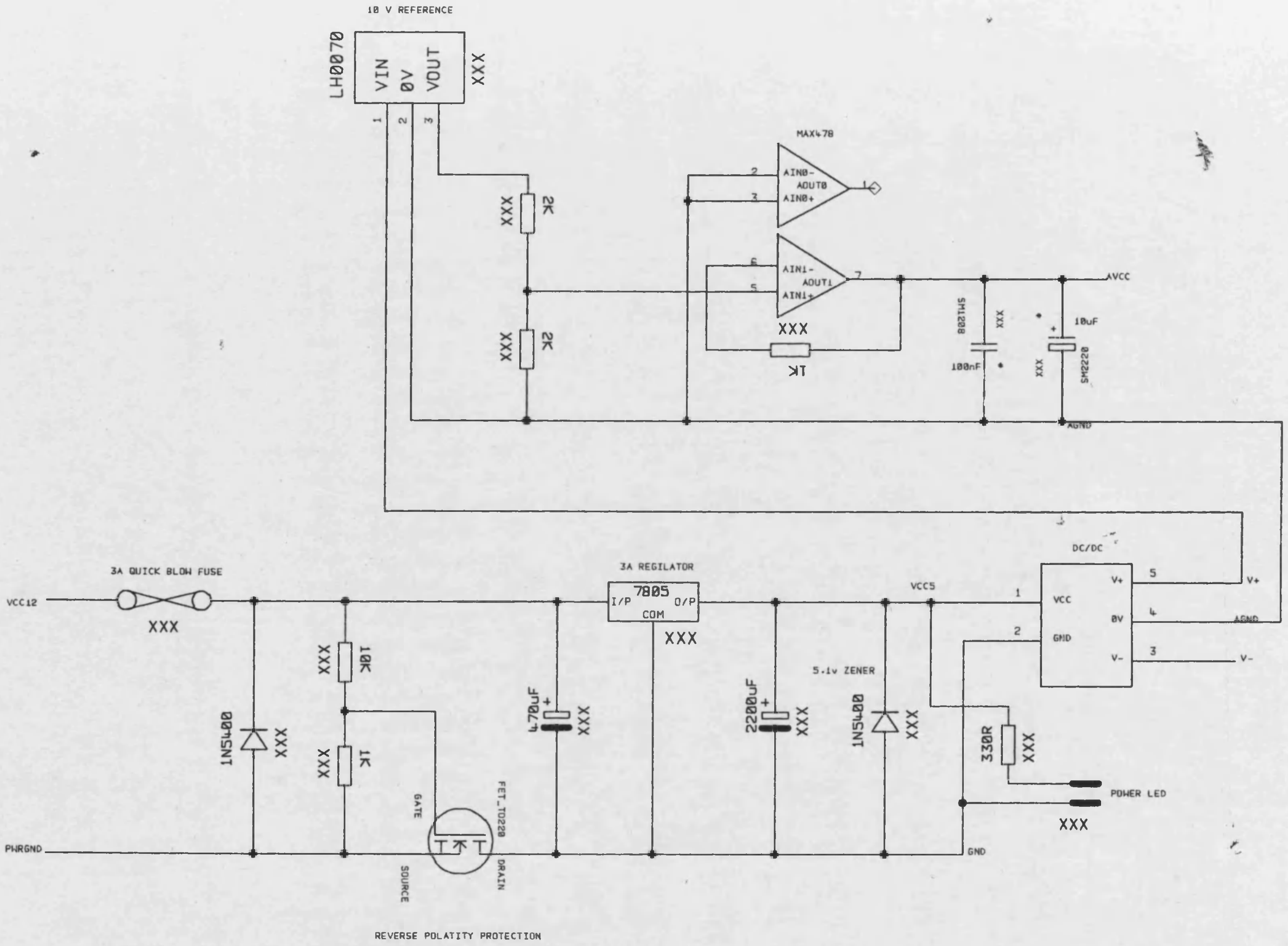
Digital input

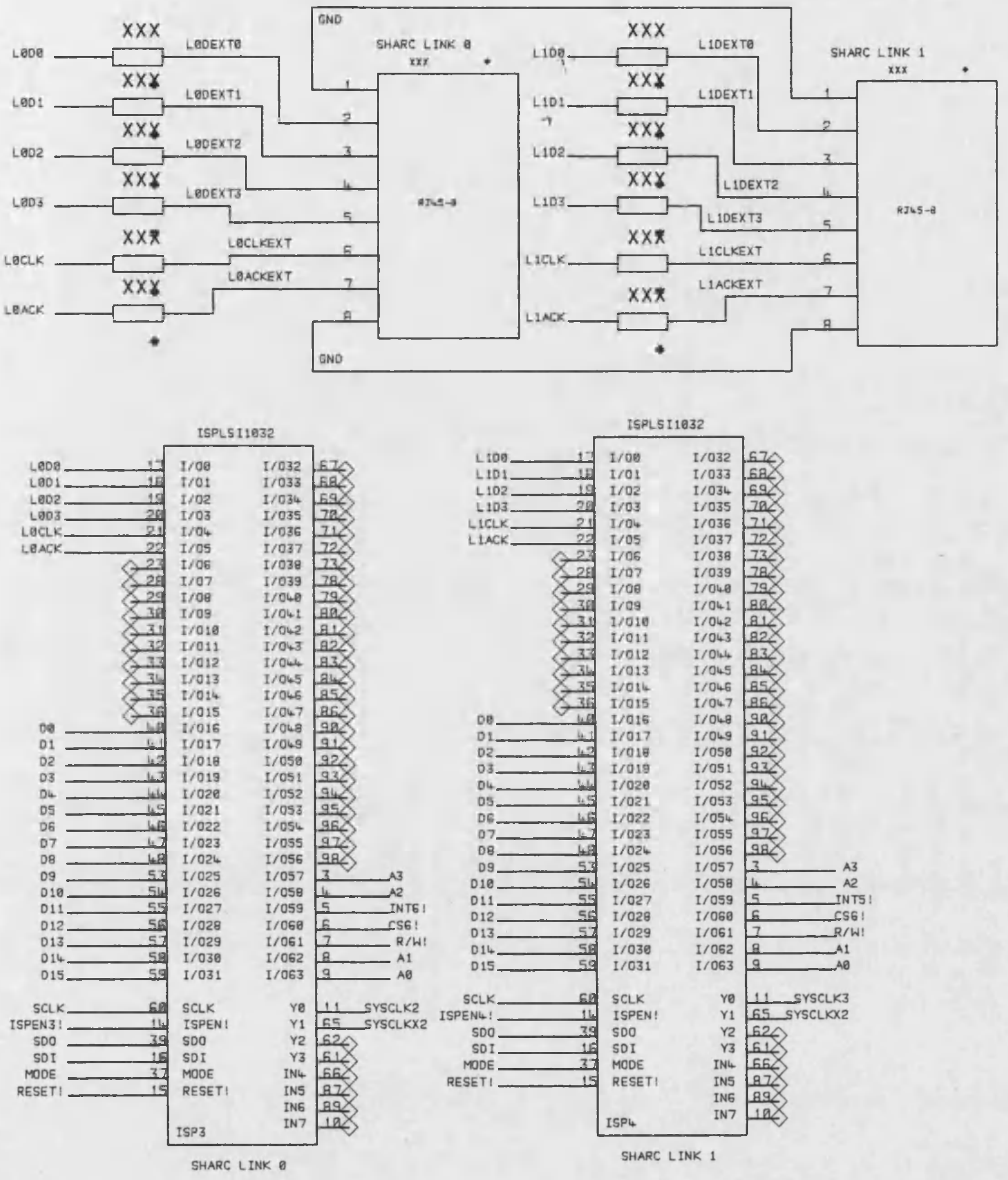
- 24 input channels, opto-isolated, 300v pk-pk tolerant.

Digital Output

- 24 output channels, 3A 50v (short cct, over voltage & temperature protected), opto-isolated.

CIRCUIT DIAGRAMS FOR COMPASS RAPID PROTOTYPING SYSTEM – ECU32 BASE BOARD


10 V REFERENCE
LH0070
VIN
0V
VOUT
XXX
1
2
3
2K
XXX
2K
XXX
MAX478
AIN0-
AOUT0
AIN0+
AIN1-
AOUT1
AIN1+
2
3
1
6
5
7
XXX
1K
AVCC
SM1208
XXX
100nF
10uF
XXX
SM2220
AGND
DC/DC
VCC
GND
V+
0V
V-
1
2
5
4
3
V+
AGND
V-
VCC5
3A QUICK BLOW FUSE
VCC12
XXX
3A REGILATOR
7805
I/P
O/P
COM
XXX
1N5400
XXX
10K
XXX
1K
XXX
470uF
XXX
2200uF
XXX
5.1v ZENER
1N5400
XXX
330R
XXX
POWER LED
XXX
GATE
FET_TO220
SOURCE
DRAIN
PWRGND
GND
REVERSE POLATITY PROTECTION

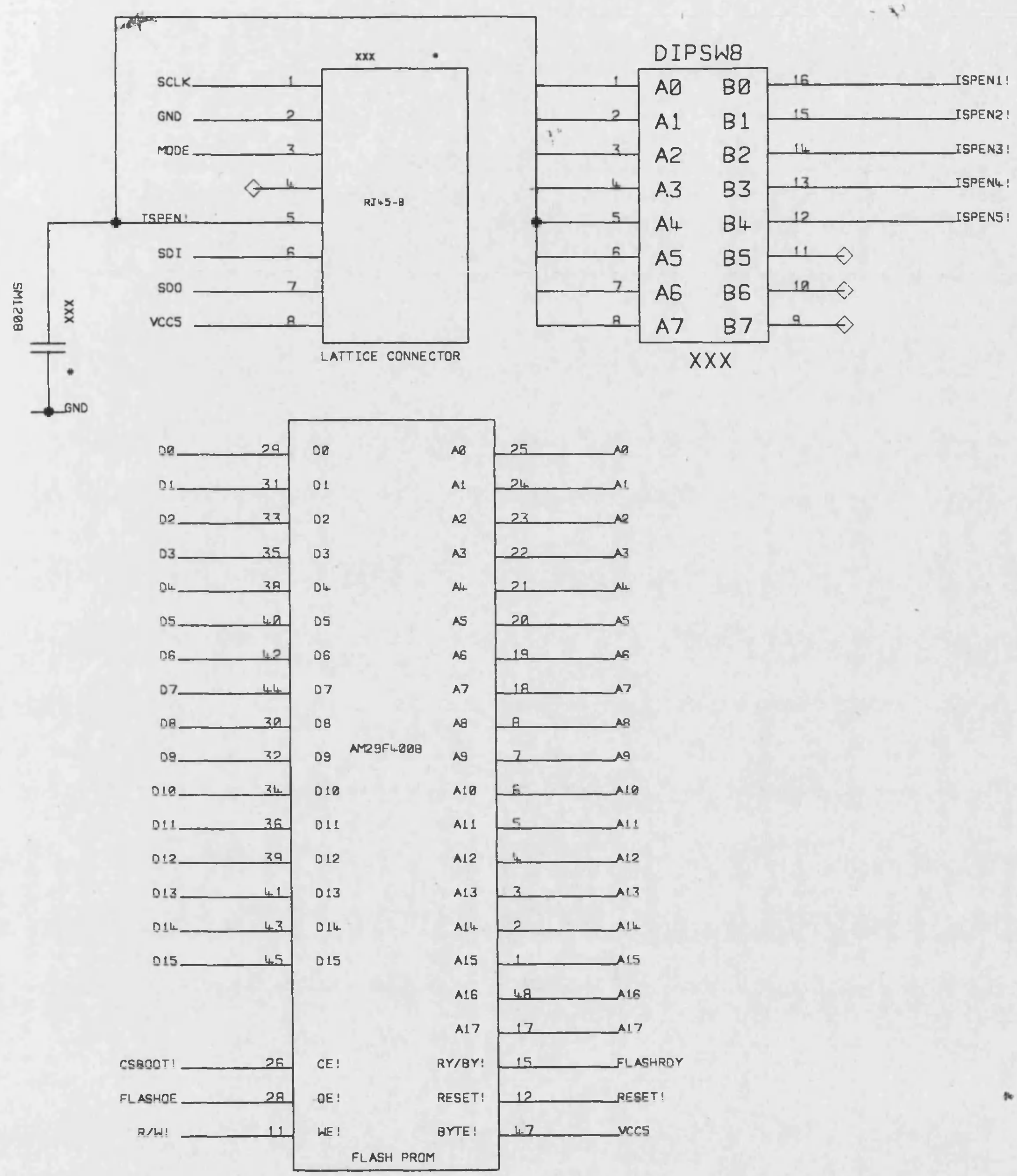


SHARC LINK 0
XXX
RJ45-8
GND
L0D0
L0D1
L0D2
L0D3
L0CLK
L0ACK
L0DEXT0
L0DEXT1
L0DEXT2
L0DEXT3
L0CLKEXT
L0ACKEXT
SHARC LINK 1
L1D0
L1D1
L1D2
L1D3
L1CLK
L1ACK
L1DEXT0
L1DEXT1
L1DEXT2
L1DEXT3
L1CLKEXT
L1ACKEXT
ISPLSI1032
ISP3
ISP4
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
SCLK
ISPEN3!
ISPEN4!
ISPEN!
SDO
SDI
MODE
RESET!
Y0
Y1
Y2
Y3
IN4
IN5
IN6
IN7
SYSCLK2
SYSCLK3
SYSCLKX2
A3
A2
A1
A0
INT6!
INT5!
CS6!
R/W!

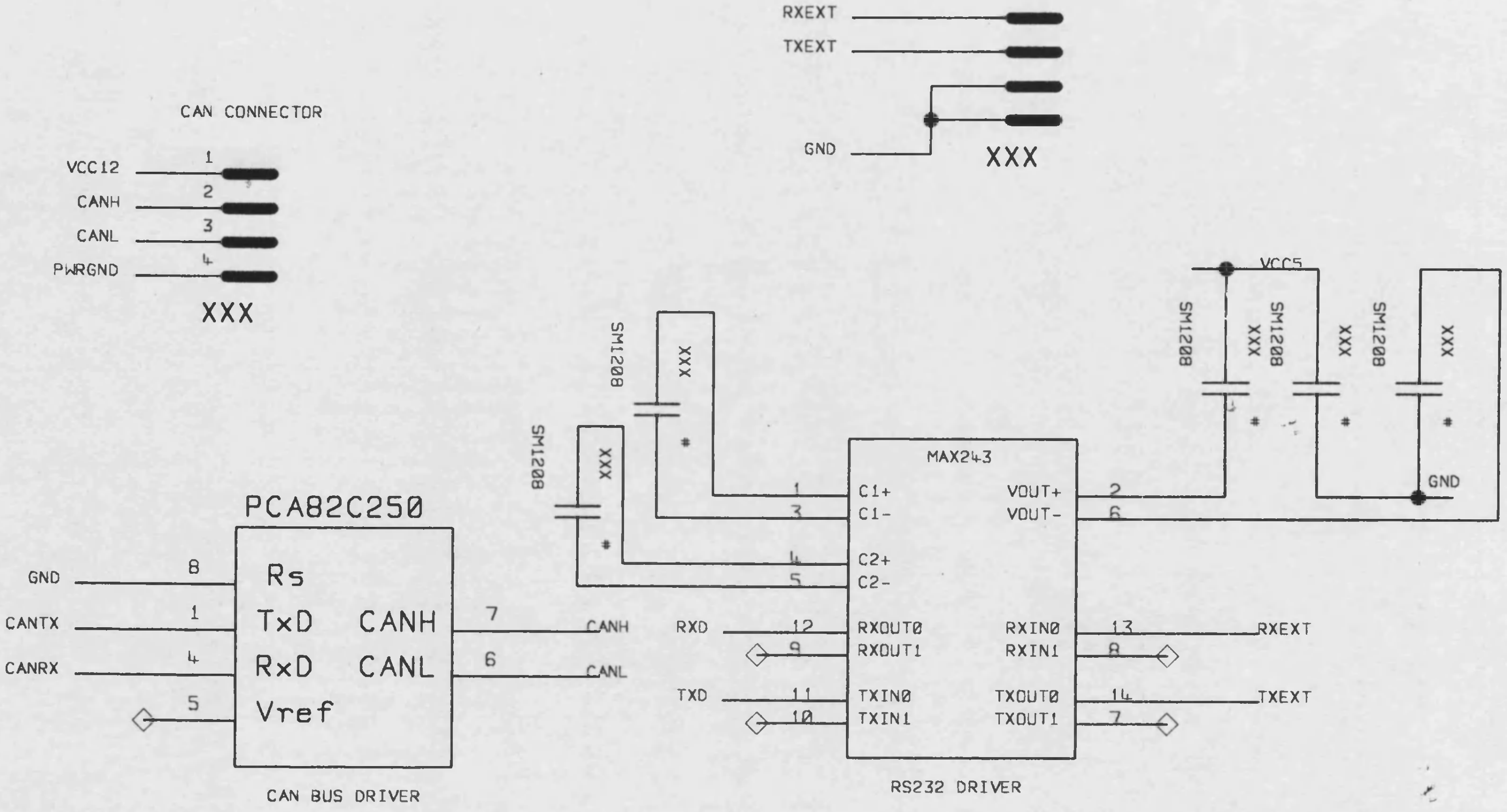
96WAY EURO CONNECTOR

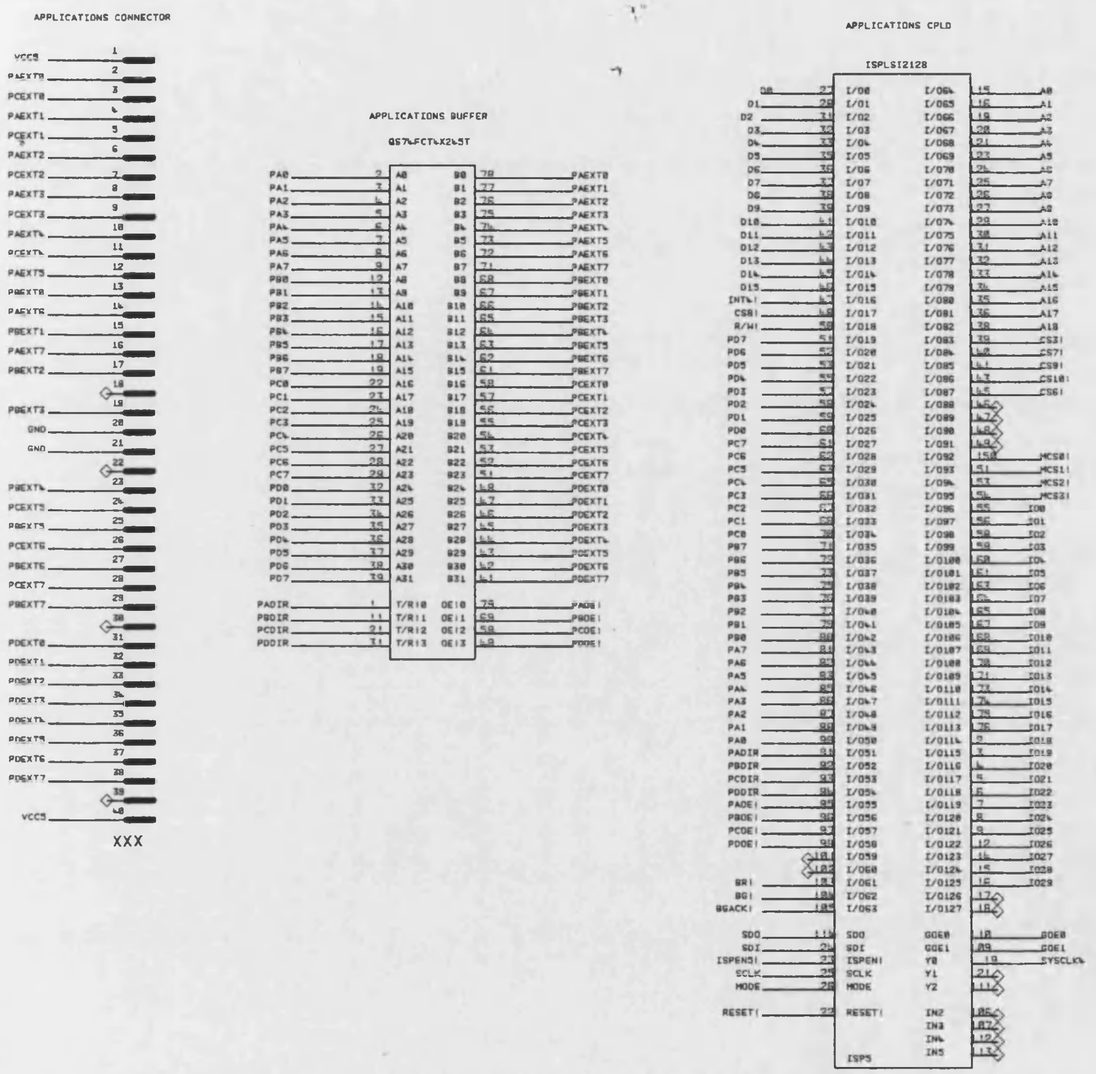
PWRGND	1a	PWRGND	1b	PWRGND	1c
IO0	2a	TPUCH0	2b	SCS0	2c
IO1	3a	TPUCH1	3b	SCS1	3c
IO2	4a	TPUCH2	4b	SCS2	4c
IO3	5a	TPUCH3	5b	SCS3	5c
IO4	6a	TPUCH4	6b	SSCK	6c
IO5	7a	TPUCH5	7b	MOSI	7c
IO6	8a	TPUCH6	8b	MISO	8c
IO7	9a	TPUCH7	9b	RXEXT	9c
IO8	10a	TPUCH8	10b	TXEXT	10c
IO9	11a	TPUCH9	11b	CTD3	11c
IO10	12a	TPUCH10	12b	CTD4	12c
IO11	13a	TPUCH11	13b	CTD9	13c
IO12	14a	TPUCH12	14b	CTD10	14c
IO13	15a	TPUCH13	15b	CPWM5	15c
IO14	16a	TPUCH14	16b	CPWM6	16c
IO15	17a	TPUCH15	17b	CPWM7	17c
IO16	18a	T2CLK	18b	CPWM8	18c
IO17	19a	GND	19b	CTM2C	19c
IO18	20a	RESET!	20b	GND	20c
IO19	21a	INT3!	21b	CANL	21c
IO20	22a	INT2!	22b	CANH	22c
IO21	23a	INT1!	23b	GND	23c
IO22	24a	GND	24b	GND	24c
IO23	25a	L0DEXT0	25b	L1DEXT0	25c
IO24	26a	L0DEXT1	26b	L1DEXT1	26c
IO25	27a	L0DEXT2	27b	L1DEXT2	27c
IO26	28a	L0DEXT3	28b	L1DEXT3	28c
IO27	29a	L0CLKEXT	29b	L1CLKEXT	29c
IO28	30a	L0ACKEXT	30b	L1ACKEXT	30c
IO29	31a	GND	31b	GND	31c
VCC12	32a	VCC12	32b	VCC12	32c

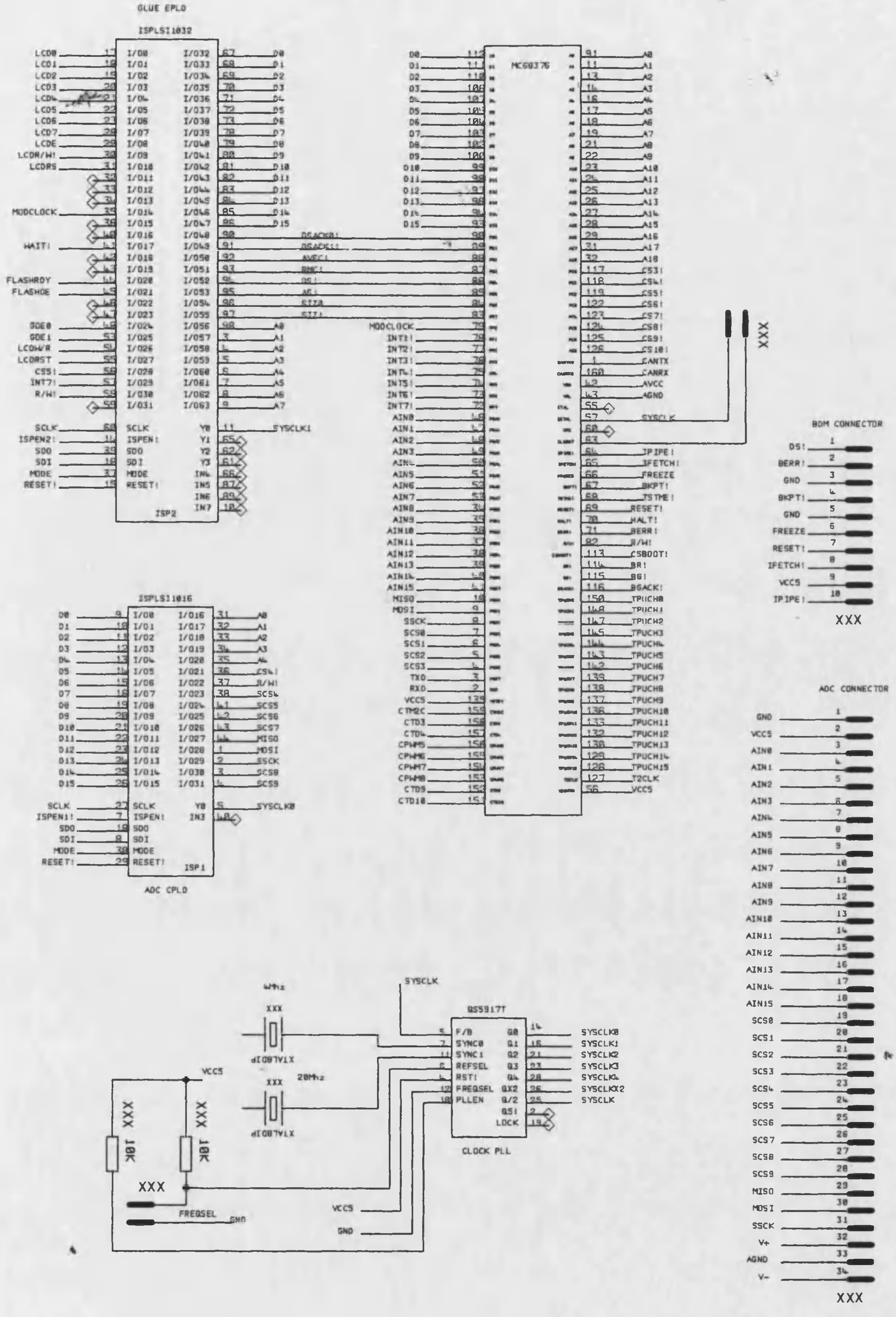
XXX

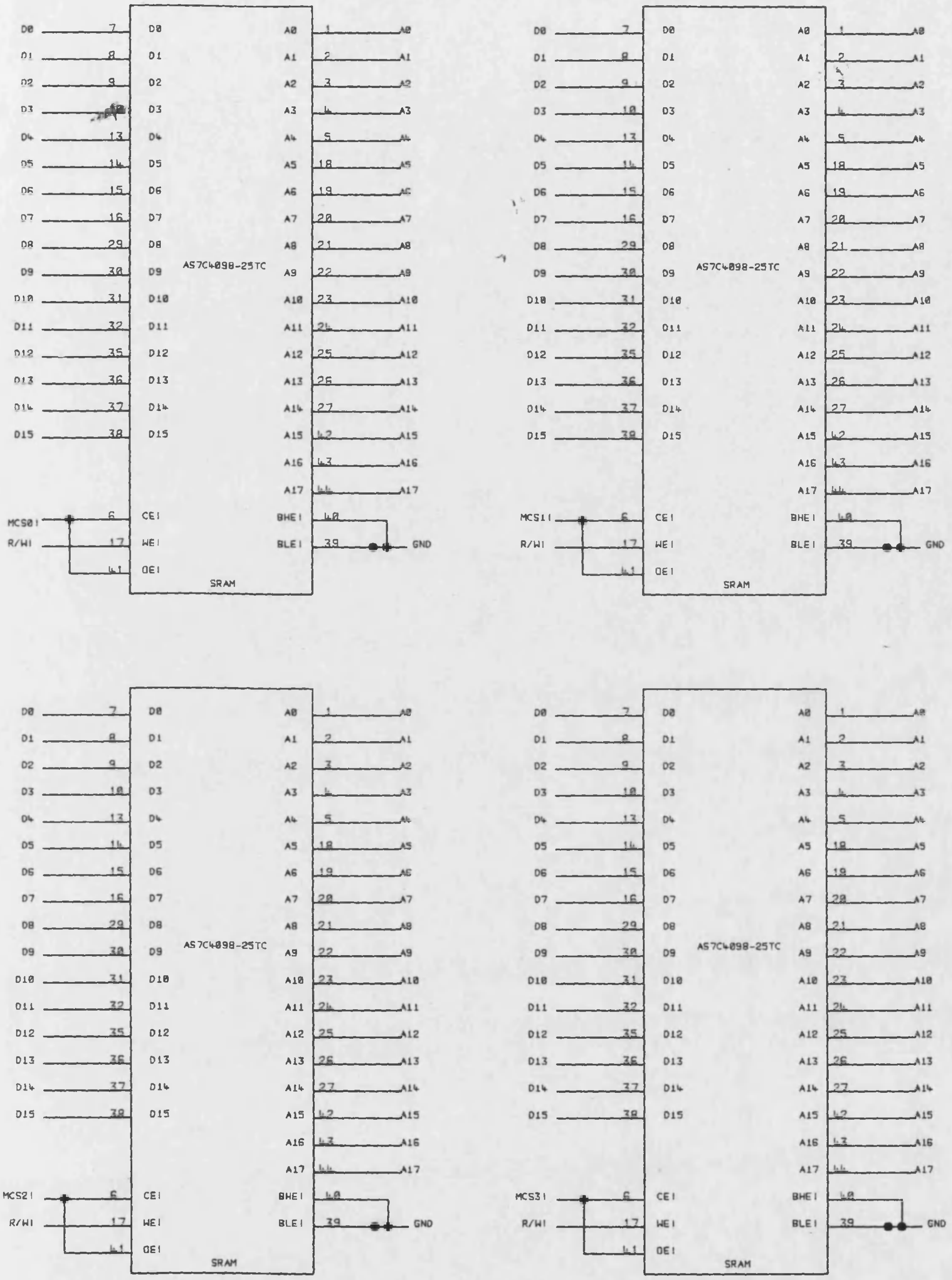


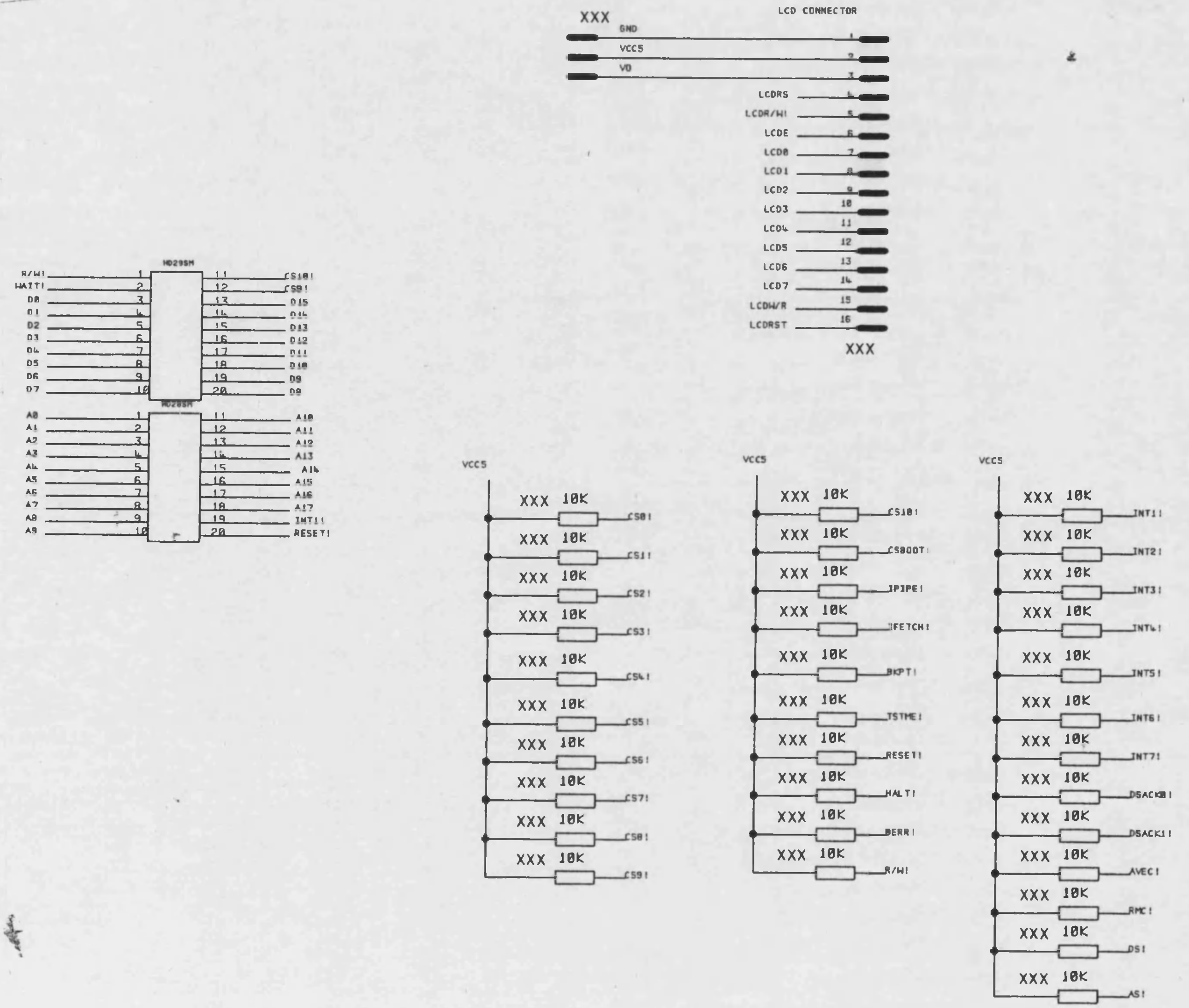
xxx
SCLK
GND
MODE
ISPEN!
SDI
SDO
VCC5
1
2
3
4
5
6
7
8
RJ45-8
LATTICE CONNECTOR
SM1208
XXX
GND
DIPSW8
A0
A1
A2
A3
A4
A5
A6
A7
B0
B1
B2
B3
B4
B5
B6
B7
16
15
14
13
12
11
10
9
ISPEN1!
ISPEN2!
ISPEN3!
ISPEN4!
ISPEN5!
XXX
D0 29
D1 31
D2 33
D3 35
D4 38
D5 40
D6 42
D7 44
D8 30
D9 32
D10 34
D11 36
D12 39
D13 41
D14 43
D15 45
A0 25
A1 24
A2 23
A3 22
A4 21
A5 20
A6 19
A7 18
A8 8
A9 7
A10 6
A11 5
A12 4
A13 3
A14 2
A15 1
A16 48
A17 17
AM29F400B
CSBOOT! 26 CE!
FLASHOE 28 OE!
R/W! 11 WE!
RY/BY! 15 FLASHRDY
RESET! 12 RESET!
BYTE! 47 VCC5
FLASH PROM

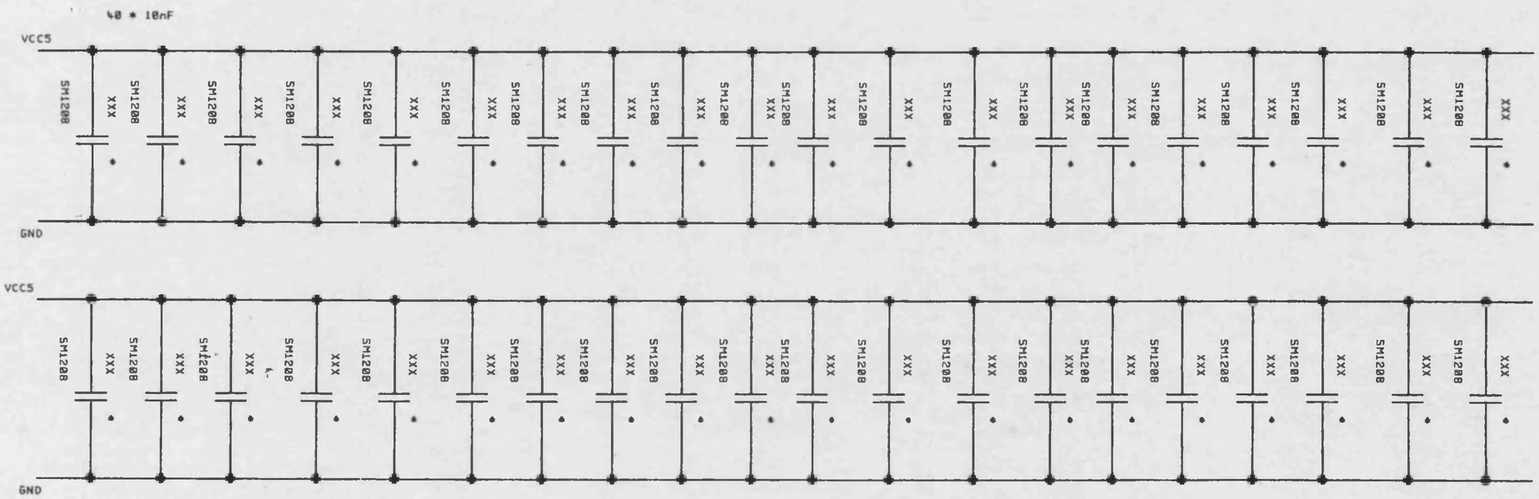

RXEXT
TXEXT
GND
XXX
CAN CONNECTOR
VCC12
CANH
CANL
PWRGND
1
2
3
4
XXX
VCC5
SM1208
XXX
SM1208
XXX
SM1208
XXX
SM1208
XXX
SM1208
XXX
MAX243
C1+
C1-
C2+
C2-
VOUT+
VOUT-
GND
PCA82C250
Rs
TxD
RxD
Vref
CANH
CANL
GND
CANTX
CANRX
8
1
4
5
7
6
CANH
CANL
RXD
TXD
RXOUT0
RXOUT1
TXIN0
TXIN1
RXIN0
RXIN1
TXOUT0
TXOUT1
1
3
4
5
12
9
11
10
2
6
13
8
14
7
RXEXT
TXEXT
CAN BUS DRIVER
RS232 DRIVER


APPLICATIONS CONNECTOR
VCC5
PAEXT0
PCEXT0
PAEXT1
PCEXT1
PAEXT2
PCEXT2
PAEXT3
PCEXT3
PAEXT4
PCEXT4
PAEXT5
PBEXT0
PAEXT6
PBEXT1
PAEXT7
PBEXT2
PBEXT3
GND
GND
PBEXT4
PCEXT5
PBEXT5
PCEXT6
PBEXT6
PCEXT7
PBEXT7
PDEXT0
PDEXT1
PDEXT2
PDEXT3
PDEXT4
PDEXT5
PDEXT6
PDEXT7
VCC5
XXX
APPLICATIONS BUFFER
QS74FCT4X245T
PADIR
PBDIR
PCDIR
PDDIR
T/R10
T/R11
T/R12
T/R13
OE10
OE11
OE12
OE13
PAOE1
PBOE1
PCOE1
PDOE1
APPLICATIONS CPLD
ISPLSI2128
INT41
CS81
R/W1
BR1
BG1
BGACK1
SDO
SDI
ISPEN31
SCLK
MODE
RESET1
GOE0
GOE1
SYSCLK4
MCS01
MCS11
MCS21
MCS31
CS31
CS71
CS91
CS101
CS61
ISP5


GLUE EPLD
ISPLSI1032
ISP2
ISPLSI1016
ISP1
ADC CPLD
MC68376
BDM CONNECTOR
DS!
BERR!
GND
BKPT!
FREEZE
RESET!
IFETCH!
VCC5
IPIPE!
XXX
ADC CONNECTOR
AIN0
AIN15
SCS0
SCS9
MISO
MOSI
SSCK
V+
AGND
V-
SYSCLK
QS5917T
CLOCK PLL
F/B
SYNC0
SYNC1
REFSEL
RST!
FREQSEL
PLLEN
LOCK
SYSCLK0
SYSCLK1
SYSCLK2
SYSCLK3
SYSCLK4
SYSCLKX2
4Mhz
20Mhz
XTALDBIP
10K
VCC5
GND
FREQSEL


D0 7 D0
D1 8 D1
D2 9 D2
D3 10 D3
D4 13 D4
D5 14 D5
D6 15 D6
D7 16 D7
D8 29 D8
D9 30 D9
D10 31 D10
D11 32 D11
D12 35 D12
D13 36 D13
D14 37 D14
D15 38 D15
A0 1 A0
A1 2 A1
A2 3 A2
A3 4 A3
A4 5 A4
A5 18 A5
A6 19 A6
A7 20 A7
A8 21 A8
A9 22 A9
A10 23 A10
A11 24 A11
A12 25 A12
A13 26 A13
A14 27 A14
A15 42 A15
A16 43 A16
A17 44 A17
AS7C4098-25TC
MCS0I 6 CEI
R/WI 17 WEI
41 OEI
BHEI 40
BLEI 39 GND
SRAM
D0 7 D0
D1 8 D1
D2 9 D2
D3 10 D3
D4 13 D4
D5 14 D5
D6 15 D6
D7 16 D7
D8 29 D8
D9 30 D9
D10 31 D10
D11 32 D11
D12 35 D12
D13 36 D13
D14 37 D14
D15 38 D15
A0 1 A0
A1 2 A1
A2 3 A2
A3 4 A3
A4 5 A4
A5 18 A5
A6 19 A6
A7 20 A7
A8 21 A8
A9 22 A9
A10 23 A10
A11 24 A11
A12 25 A12
A13 26 A13
A14 27 A14
A15 42 A15
A16 43 A16
A17 44 A17
AS7C4098-25TC
MCS1I 6 CEI
R/WI 17 WEI
41 OEI
BHEI 40
BLEI 39 GND
SRAM
D0 7 D0
D1 8 D1
D2 9 D2
D3 10 D3
D4 13 D4
D5 14 D5
D6 15 D6
D7 16 D7
D8 29 D8
D9 30 D9
D10 31 D10
D11 32 D11
D12 35 D12
D13 36 D13
D14 37 D14
D15 38 D15
A0 1 A0
A1 2 A1
A2 3 A2
A3 4 A3
A4 5 A4
A5 18 A5
A6 19 A6
A7 20 A7
A8 21 A8
A9 22 A9
A10 23 A10
A11 24 A11
A12 25 A12
A13 26 A13
A14 27 A14
A15 42 A15
A16 43 A16
A17 44 A17
AS7C4098-25TC
MCS2I 6 CEI
R/WI 17 WEI
41 OEI
BHEI 40
BLEI 39 GND
SRAM
D0 7 D0
D1 8 D1
D2 9 D2
D3 10 D3
D4 13 D4
D5 14 D5
D6 15 D6
D7 16 D7
D8 29 D8
D9 30 D9
D10 31 D10
D11 32 D11
D12 35 D12
D13 36 D13
D14 37 D14
D15 38 D15
A0 1 A0
A1 2 A1
A2 3 A2
A3 4 A3
A4 5 A4
A5 18 A5
A6 19 A6
A7 20 A7
A8 21 A8
A9 22 A9
A10 23 A10
A11 24 A11
A12 25 A12
A13 26 A13
A14 27 A14
A15 42 A15
A16 43 A16
A17 44 A17
AS7C4098-25TC
MCS3I 6 CEI
R/WI 17 WEI
41 OEI
BHEI 40
BLEI 39 GND
SRAM

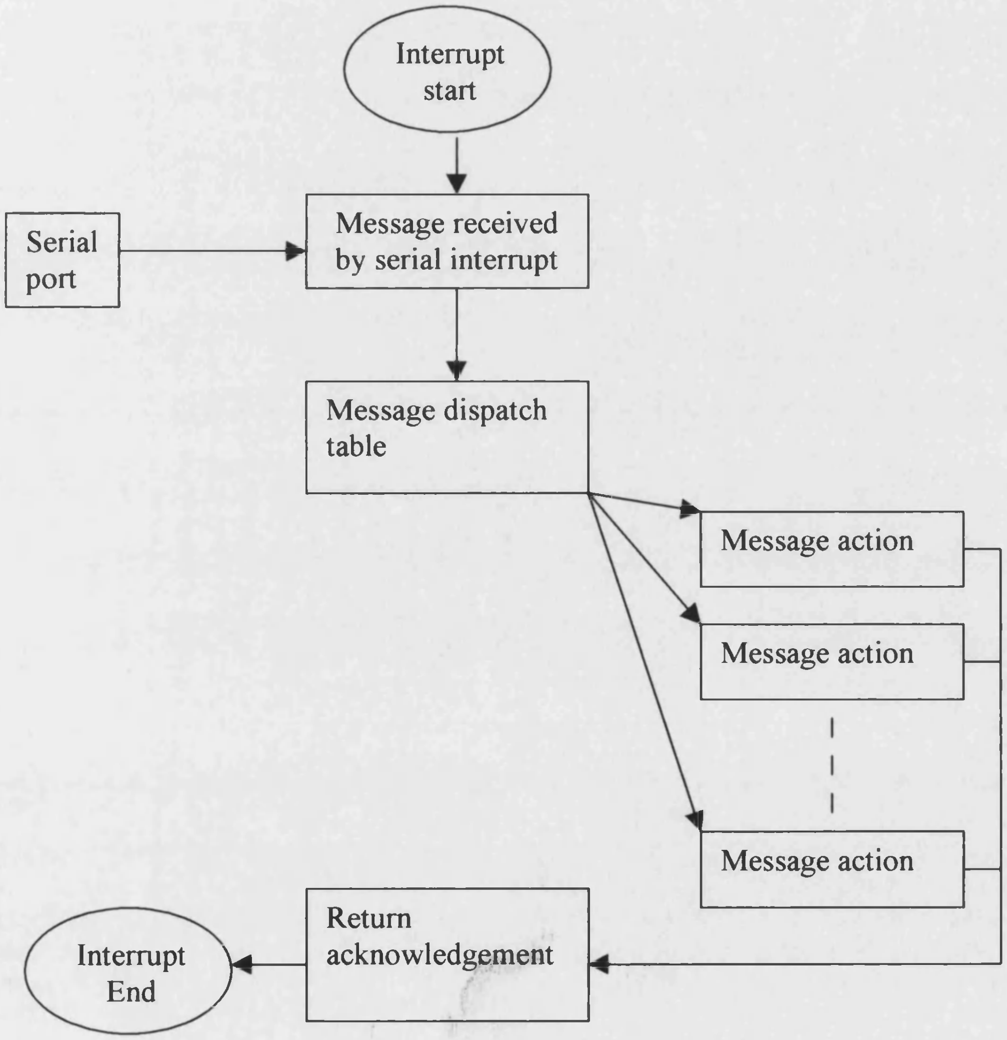

XXX
LCD CONNECTOR
GND
VCC5
VD
LCDRS
LCDR/W!
LCDE
LCD0
LCD1
LCD2
LCD3
LCD4
LCD5
LCD6
LCD7
LCDW/R
LCDRST
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
XXX
HD20SM
R/W!
WAIT!
D0
D1
D2
D3
D4
D5
D6
D7
CS10!
CS9!
D15
D14
D13
D12
D11
D10
D9
D8
HD20SM
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
A17
INT1!
RESET!
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
VCC5
XXX 10K
CS0!
CS1!
CS2!
CS3!
CS4!
CS5!
CS6!
CS7!
CS8!
CS9!
VCC5
XXX 10K
CS10!
CSBOOT!
IPIPE!
IFETCH!
BKPT!
TSTME!
RESET!
HALT!
BERR!
R/W!
VCC5
XXX 10K
INT1!
INT2!
INT3!
INT4!
INT5!
INT6!
INT7!
DSACK0!
DSACK1!
AVEC!
RMC!
DS!
AS!


40 * 10nF
VCC5
GND
VCC5
GND
SM1208
XXX

MAIN OPERATING SOFTWARE FOR COMPASS SYSTEM.

This software contains the layer stack for the Can Communication Protocol. This has initially been developed to run over the serial port. A description and flow diagram of the software is as follows.

The communication protocol follows the CCP message structure. This structure defines an 8 byte message for each communication action. Each action sent to the slave (the Compass ECU) from the master a Personal Computer running a calibration tool is acknowledged. The actions taken by the slave depend upon the message. The main loop does very little & the CCP stack runs under interrupts.



Main Definition file

```
/*
**********************************************************
**********************************************************
**                                                    **
**    Crimson os version 1.0                          **
**    MAIN DEFINES FILE                               **
**                                                    **
**    Rev 1.0 - 25/2/2000                             **
**          Can Communication Protocol ver 2.1        **
**          and simple serial port driver             **
**          Runs on Crimson I ecu with GPIO board     **
**                                                    **
**                                                    **
**********************************************************
**********************************************************/

/* interrupt stuff */

#define           VECTOR_BASE_ADDRESS     0x80000


#define           TRAP0       0x20
#define           SERIALINT   0x40
#define           TIMERINT    0x42
#define           TRIGINT           0x43
#define           ADCINT            0x67


#define           TIMER_INT   (TIMERINT*4)+VECTOR_BASE_ADDRESS
#define           TRIG_INT    (TRIGINT*4)+VECTOR_BASE_ADDRESS
#define           ADC_INT     (ADCINT*4)+VECTOR_BASE_ADDRESS
#define           SERIAL_INT  (SERIALINT*4)+VECTOR_BASE_ADDRESS
#define           TRAP0_INT   (TRAP0*4)+VECTOR_BASE_ADDRESS



/* System integration module defines */

#define SIMCR           0xFFFA00
#define PFPAR           0xFFFA1E
#define SYPCR           0xFFFA20
#define PICR            0xFFFA22
#define CSPAR0          0xFFFA44
#define CSPAR1          0xFFFA46
#define CSBARBT         0xFFFA48
#define CSORBT          0xFFFA4A
#define CSBAR0          0xFFFA4C
#define CSBAR1          0xFFFA50
#define CSBAR2          0xFFFA54
#define CSBAR3          0xFFFA58
#define CSBAR4          0xFFFA5C
#define CSBAR5          0xFFFA60
#define CSBAR6          0xFFFA64
```

```
#define CSBAR7          0xFFFA68
#define CSBAR8          0xFFFA6C
#define CSOR0           0xFFFA4E
#define CSOR1           0xFFFA52
#define CSOR2           0xFFFA56
#define CSOR3           0xFFFA5A
#define CSOR4           0xFFFA5E
#define CSOR5           0xFFFA62
#define CSOR6           0xFFFA66
#define CSOR7           0xFFFA6A
#define CSOR8           0xFFFA6E

// Queued serial module defines

#define QSMCR           0xFFFC00          // control register
#define QSMIR            0xFFFC04         // interrupt control reg
#define SCCR0            0xFFFC08         // Serial Comms CR0
#define SCCR1            0xFFFC0A         // Serial Comms CR1
#define SCSR             0xFFFC0C         // Serial Comms STATUS REG
#define SCDR             0xFFFC0E         // Serial Comms DATA REG
#define PORTQS           0xFFFC14         // PQS DATA REG
#define PQSPAR           0xFFFC16         // PQS DATA DIR REG
#define SPCR0            0xFFFC18         // SPI CONTROL REG0
#define SPCR1            0xFFFC1A         // SPI CONTROL REG1
#define SPCR2            0xFFFC1C         // SPI CONTROL REG2
#define SPCR3            0xFFFC1E         // SPI CONTROL REG3
#define RR0         0xFFFD00        // RECIEVE RAM
#define TR0         0xFFFD20        // TRANSMIT RAM
#define CR0         0xFFFD40        // CONTROL RAM
#define CTS        0x0020                 // clear to send (CTS=0)
#define NOCTS           0x0000                  // not clear to send
(CTS=1)


// Time Processor Unit

#define TPUMCR           0xFFFE00         // TPU module control
register
#define TICR             0xFFFE08         // TPU interrupt
Configuration Reg
#define CIER             0xFFFE0A         // Channel Int Enable Reg
#define CFSR0            0xFFFE0C         // Channel Function select
reg 0
#define CFSR1            0xFFFE0E         // Channel Function select
reg 1
#define CFSR2            0xFFFE10         // Channel Function select
reg 2
#define CFSR3            0xFFFE12         // Channel Function select
reg 3
#define HSQR0            0xFFFE14         // Host s ence select
register 0
#define HSQR1            0xFFFE16         // Host s ence select
register 1
#define HSRR0            0xFFFE18         // Host service r est
register 0
```

```
#define HSRR1            0xFFFE1A         // Host service r est
register 1
#define CPR0             0xFFFE1C         // Channel priority reg 0
#define CPR1             0xFFFE1E         // Channel priority reg 1
#define CISR             0xFFFE20         // Channel interrupt status
reg
#define LR         0xFFFE22         // link register
#define SGLR             0xFFFE24         // service grant latch
register
#define DCNR             0xFFFE26         // decoded channel number
register
#define TPUMCR2          0xFFFE28         // TPU2 module configuration
register 2
#define CH0PBR           0xFFFF00         // channel 0 parameter base
register
#define CH1PBR           0xFFFF10         // channel 1 parameter base
register
#define CH2PBR           0xFFFF20         // channel 2 parameter base
register
#define CH3PBR           0xFFFF30         // channel 3 parameter base
register
#define CH4PBR           0xFFFF40         // channel 4 parameter base
register
#define CH5PBR           0xFFFF50         // channel 5 parameter base
register
#define CH6PBR           0xFFFF60         // channel 6 parameter base
register
#define CH7PBR           0xFFFF70         // channel 7 parameter base
register
#define CH8PBR           0xFFFF80         // channel 8 parameter base
register
#define CH9PBR           0xFFFF90         // channel 9 parameter base
register
#define CH10PBR          0xFFFFA0         // channel 10 parameter base
register
#define CH11PBR          0xFFFFB0         // channel 11 parameter base
register
#define CH12PBR          0xFFFFC0         // channel 12 parameter base
register
#define CH13PBR          0xFFFFD0         // channel 13 parameter base
register
#define CH14PBR          0xFFFFE0         // channel 14 parameter base
register
#define CH15PBR          0xFFFFF0         // channel 15 parameter base
register
#define PARAM0           0x0         // parameter 0
#define PARAM1           0x2         // parameter 1
#define PARAM2           0x4         // parameter 2
#define PARAM3           0x6         // parameter 3
#define PARAM4           0x8         // parameter 4
#define PARAM5           0xA         // parameter 5
#define PARAM6           0xC         // parameter 6
#define PARAM7           0xE         // parameter 7
```

```
// Counter Timer Unit

#define BIUMCR           0xFFF400         // BIUSM module config reg
#define BIUTBR           0xFFF404         // Time base register
#define CPCR             0xFFF408         // CPSM control reg
#define MCSM2SIC   0xFFF410         // MCSM2 control reg
#define MCSM2CNT   0xFFF412         // MCSM2 Counter register
#define MCSM2ML          0xFFF414         // MCSM2 Modulus Latch
#define DASM3SIC   0xFFF418         // DASM cont reg
#define DASM3A           0xFFF41A         // DASM register A
#define PWM8A            0xFFF442
#define PWM8B            0xFFF444
#define PWM7A            0xFFF43A
#define PWM7B            0xFFF43C
#define PWM7SIC          0xFFF438
#define PWM8SIC          0xFFF440

// Controller Area Network

#define CANMCR           0xFFF080         // CAN Module Control
Register
#define CANICR           0xFFF084         // CAN Int reg
#define CANCTRL          0xFFF086         // Control reg 1 (high byte)
& 0 (low byte)
#define CPRESDIV   0xFFF088         // Prescale divider & Cont reg 2
#define CANTIMER   0xFFF08A         // Free running timer
#define CRXGMSKH   0xFFF090         // Receive global mask High
#define CRXGMSKL   0xFFF092         // Receive global mask low
#define CRX14MSKH  0xFFF094         // Receive buffer 14 mask high
#define CRX14MSKL  0xFFF096         // Receive buffer 14 mask low
#define CRX15MSKH  0xFFF098         // Receive buffer 15 mask high
#define CRX15MSKL  0xFFF09A         // Receive buffer 15 mask low
#define CESTAT           0xFFF0A0         // Error and Status Register
#define CIMASK           0xFFF0A2         // interrupt mask
#define CIFLAG           0xFFF0A4         // interrupt flags
#define CRTXECTR   0xFFF0A6         // Receive & Transmit Error
counters
#define CANMB0           0xFFF100         // message buffer 0 base reg
#define CANMB1           0xFFF110         // message buffer 1 base reg
#define CANMB2           0xFFF120         // message buffer 2 base reg
#define CANMB3           0xFFF130         // message buffer 3 base reg
#define CANMB4           0xFFF140         // message buffer 4 base reg
#define CANMB5           0xFFF150         // message buffer 5 base reg
#define CANMB6           0xFFF160         // message buffer 6 base reg
#define CANMB7           0xFFF170         // message buffer 7 base reg
#define CANMB8           0xFFF180         // message buffer 8 base reg
#define CANMB9           0xFFF190         // message buffer 9 base reg
#define CANMB10          0xFFF1A0         // message buffer 10 base reg
#define CANMB11          0xFFF1B0         // message buffer 11 base reg
#define CANMB12          0xFFF1C0         // message buffer 12 base reg
#define CANMB13          0xFFF1D0         // message buffer 13 base reg
#define CANMB14          0xFFF1E0         // message buffer 14 base reg
#define CANMB15          0xFFF1F0         // message buffer 15 base reg


// QADC registers
```

```
#define QADCMCR          0xFFF000         // Module config register
#define QADCINT          0xFFF004         // Interrupt register
#define PORTQAB          0xFFF006         // data ports A & B
#define DDRQA            0xFFF008         // Data direction register
for port A
#define QACR0            0xFFF00A         // Control register 0
#define QACR1            0xFFF00C         // Corntrol Regiser 1
#define QACR2            0xFFF00E         // CR 2
#define QASR             0xFFF010         // status register
#define CCW        0xFFF030         // start of conversion word table
#define RWT        0xFFF0B0         // start of result word table

/********************************************************
*                                         *
*     CCP stuff                           *
*                                         *
*                                         *
*********************************************************/

#define PKTLEN          8     // length of my packet!

#define DAQLIST   6     /* maximum number of DAQ lists */
#define ODTLIST   42    /* maximum number of ODT lists in a DAQ */

#define     FALSE 0
#define TRUE 1

#define LOWMEM 0x00000000  /* lowest memory address accessable by CCP
*/
#define HIGHMEM 0xFF280000 /* highest memory address accessable by CCP
*/

/* resource mask */

#define CAL 0x00   //0x01
#define DAQ 0x02   //0x02
#define DNLD 0x00 //0x04
#define PROG 0x04 //0x40

/* other station specific stuff */

#define STATIONID 0x3133      /* this station's address */
#define SLAVEID 0x3433  /* slave ID 16 bit (I made it up) */
#define VER       0x32  /* CCP version 2.1 */
#define REL       0x31
#define IDLENGTH 0x02         /* ID length is 16 bits */
#define IDTYPE    0x00

#define BUSY      0x00  //0x00
#define     START 0x31  //0x01
#define     STOP  0x30  //0x00
#define     SYNCH 0x32  //0x02
#define RUN       'r'


#define CALSEED 0x00FA67BB    /* 32 bit seed */
#define DAQSEED 0x00FA7522
```

```
#define DNLDSEED  0x00EB89AC
#define PROGSEED  0x0012EFAB

#define nConnect        'c'   //0x01
#define nGet_CCP_Version      'v'   //0x1B
#define Exchange_ID           'e'   //0x17
#define nGet_seed             0x12
#define nUnlock               0x13
#define nDisconnect           'd'   //0x07
#define nSet_S_Status         0x0C
#define nGet_S_Status         0x0D
#define nBuildchecksum        0x0E
#define nTest                 0x05
#define nDiag_Service         0x20
#define nAction_Service       'a'   //0x21
#define nSetmta               's'   //0x02
#define nDnload               'l'   //0x03
#define nDnload6        0x23
#define nUpload               'u'   //0x04
#define nShortup        0x0f
#define nClearmemory          0x10
#define nProgram        0x18
#define nProgram6       0x22
#define nMove                 0x19
/* data acquisition commands */
#define nSelect_Cal_Page      0x11
#define nGet_Daq_Size         0x14
#define nSet_Daq_Ptr          0x15
#define nWrite_Daq            0x16
#define nStart_Stop           0x06


// RETURN CODES


#define ACK       'a'   //0x00            /* acknowledge */
#define PACKETID 'p'    //0xFF            /* packet ID */

#define ACCESSLOCKED 0x22
#define ACCESSDENIED 0x33
#define OUTOFRANGE 0x32

#define DAQOVERLOAD     0x01
#define COMMANDBUSY     0x02
#define DAQBUSY         0x03
#define TIMEOUT         0x04
#define KEYREQUEST      0x05
#define STATUSREQ 0x06
#define COLDSTARTREQ    0x07
#define CALDATAINITR    0x08
#define DAQLISTINITR    0x09
#define CODEUPDATERE    0x0a
#define UNKNOWNCOMMAND  0x0b
#define COMMANDSYNTAX   0x0c
#define OVERLOAD  0x0d
```

```
typedef struct canframetype
{
      short int cmd;
      short int id;
      short int crodata[8];
}canframetype_t;

/* stuff for control */

typedef struct control
{

      short int station_address;
   short int status;
   short int connectstatus;
   short int masterid;
   short int slaveid;
   short int slaveidsize;
   short int slaveidtype;
   short int requestedmask;
   short int resourcemask;
   short int resourcepromask;
   short int diagserv;
   short int diagservsize;
   short int diagservtype;
   int actionserv;
   short int actionservsize;
   short int actionservtype;
   short int ccpver;
   short int ccprel;

   int calseeddata;
   int daqseeddata;
   int progseeddata;
   int dnldseeddata;
   int calibrationkey;
   int daqkey;
   int flashprogkey;
   int downloadkey;

   short int busy;

}control_t;

/* stuff for memory access */

typedef struct memory
{
      int *mta0;
      int mta0ext;
      int *mta1;
      int mta1ext;
}memory_t;
```

```
/* Stuff for data acquisition */

typedef struct element
{
      int address;      /* the address of the data */
      int offset;             /* an offset (if applicable) */
      short int size;         /* the size of the data (1,2 or 4) bytes
*/
}element_t;

typedef struct odt
{
      element_t elements[6];  /* 7 elements in the odt list */
      short int currentelement;     /* the current element pointed to
*/
      int PID;          /* the id of the odt (0<n<0xFD) */
}odt_t;
typedef struct daq
{
      odt_t odtlist[ODTLIST];       /* the list of odt's (maximum of 42
per daq list)*/
      short int currentptr;         /* the current point in the list */
      short int totalinlist;        /* the total number of odt's in
list */
      int can_id;             /* can id associated with daq list */
      int calibpage;
      short int eventchnumber;
      short int txnrate;
      short int mode;
}daq_t;



void rxcanmsg();
void txcanmsg();
void rxscimsg();
void txscimsg();
void serialint();
void dispatchmsg();

void connect();
void disconnect();
void getccpversion();
void commanderr();
void exchangeid();

void setmta();
void dnload();
void upload();
void getdaqsize();
void setdaqptr();
void writedaq();
void startstop();
void actionserv();

void timerint();
```

Main startup file

```
**********************************************************
**********************************************************
**                                                      **
**    Crimson os version 1.0                            **
**    MAIN START UP FILE                                **
**                                                      **
**    Rev 1.0 - 25/2/2000                               **
**          Can Communication Protocol ver 2.1          **
**          and simple serial port driver               **
**          Runs on Crimson I ecu with GPIO board       **
**          No FPU board as yet!                        **
**                                                      **
**********************************************************
**********************************************************/


SERIALINT         equ   40H


* 68020 Startup Routine for Crossware C Compiler

      INCLUDE     defines.inc

__NoRom equ 1
__LowestRomLocation equ       082000H
__HighestRomLocation equ      0FFFFFH
__LowestRamLocation equ       100000H
__HighestRamLocation equ      27FFFEH

      xref  _main,__initdata,__HP
      xdef  __cstart

* These sections are required by the Crossware C Compiler:

      section     __STACK,data,high Linker places this at highest
available ram location
      ds.w  1                 1 word minimum to create section

      section __HEAP,data,postpone  Linker places this after all ram
sections except __STACK
      ds.w  1                 1 word minimum to create section


      org   __LowestRomLocation     Lowest user ram location
      even                    Just in case an odd address has be put
into the linker settings


* setup the processor's chip selects and other internal registers
```

```
            move.w                  #$604f,(SIMCR)
            move.w                  #$ff,(PFPAR)
            move.w                  #$0,(SYPCR)
            move.w                  #$0,(PICR)
            move.w                  #$3fff,(CSPAR0)
            move.w                  #$03ff,(CSPAR1)
            move.w                  #$0006,(CSBARBT)
            move.w                  #$0806,(CSBAR0)
            move.w                  #$1006,(CSBAR1)
            move.w                  #$1806,(CSBAR2)
            move.w                  #$2006,(CSBAR3)
            move.w                  #$2800,(CSBAR4)
            move.w                  #$2808,(CSBAR5)
            move.w                  #$2810,(CSBAR6)
            move.w                  #$2818,(CSBAR7)
            move.w                  #$2820,(CSBAR8)
            move.w                  #$6830,(CSORBT)
            move.w                  #$7830,(CSOR0)
            move.w                  #$7830,(CSOR1)
            move.w                  #$7830,(CSOR2)
            move.w                  #$7830,(CSOR3)
            move.w                  #$7830,(CSOR4)
            move.w                  #$78F0,(CSOR5)    * 7833 0ws 78b0 2ws
78f0 3ws
            move.w                  #$7830,(CSOR6)
            move.w                  #$7830,(CSOR7)
            move.w                  #$7830,(CSOR8)

            move.l                  #__HighestRamLocation,A7            *
setup the stack


*****************************************************************
*                                               *
*     Serial Port setup                         *
*                                               *
*****************************************************************

* setup the serial port for interrupts

            move.w                  #$008d,(QSMCR)          * iarb of 13 for
the serial port
            move.w                  #$0700+SERIALINT,(QSMIR)      * Rx int
level at 5, vector 40h

*     9600 baud 1Ah for 8Mhz 41h for 20 Mhz

            move.w                  #$41,(SCCR0)            * baud rate of
9600 selected (41)
            move.w                  #$2c,(SCCR1)            * enable rx & tx

* set the handshaking to hardware, using portqs bit 5 as Clear to send
(output)
* amd bit 6 as Request to Send (input)

            move.w                  #$6020,(PQSPAR)         * port qs
register (use cs2 (output) for CTS)
```

```
                                          * and cs3 (input) for RTS)
            move.w            #NOCTS,(PORTQS)         * set the CTS to
OFF!



*****************************************************************
*                                               *
*     Setup CTM module                          *
*     This module generates the 1 Second Int                *
*                                               *
*****************************************************************
*
* set up Modulus Counter Sub Module (MCSM2) to count 9600h pulses
(19660800/512).
* BIUSM is set to interrupt number 40h. The MCSM2 is used as the timer
* Channel, which is an offset of 2 from the base int number ie 42h.
*
            move.w            #$0921,(BIUMCR)   * set the int for 42h &
iarb of 1
            move.w            #$4215,(MCSM2SIC) * int level 4
            move.w            #$69ff,(MCSM2CNT) * load the counter with
(ffff-9600)h

* start all the counters going

            move.w            #$0000,(CPCR)     * stop the prescaler

*****************************************************************
*                                               *
*     Setup CAN module                          *
*     The can timing is set up as follows:                  *
*     for 1 bit thetiming requirements are predefined as    *
*     +-------+-------+--------+--------+             *
*     | Sync      | Prop      | Phase1 | Phase2 |                   *
*     | Seg | Seg | Seg   | Seg         |                   *
*     +-------+-------+--------+--------+             *
*     | 1 tq      | 1 tq      | 1-7 tq | 1-7 tq |                   *
*     +-------+-------+--------+--------+             *
*     Thus for a 1 bit time of 1μ ie 1 MBaud rate the tq    *
*     will be 100ns ie 10 Mhz. The CPU clk is / by 2.       *
*     Phase 1 is set to 4 & Phase 2 is set to 4       *
*                                               *
*                                               *
*****************************************************************

            move.w            #0,(CANCTRL)

* set can timing for 10mhz time quanta, Jump width to max (3), Phase1 =
4 Phase2 = 4

            move.w            #$01E4,(CPRESDIV)

            move.w            #$0123,(CANMB0+6)
            move.w            #$4567,(CANMB0+8)
            move.w            #$89AB,(CANMB0+10)
            move.w          #$CDEF,(CANMB0+12)

            move.w          #0,(CANMB0+4)
            move.w          #$0018,(CANMB0+2)
            move.w          #$00C7,(CANMB0+0)

            move.w          #0,(CANMCR) * can module register


* setup the interrupts


            move.l          #$80000,d0
            movec       d0,VBR
            move.w          #$2300,SR         * set int level at 3


__cstart

        move.l      #__HEAP,__HP


        jsr   _main

__exit

forever     bra   forever


        end
```

Main routines file

```
/*
**********************************************************
**********************************************************
**                                                    **
**    Crimson os version 1.0                          **
**    MAIN ROUTINES                                   **
**                                                    **
**    Rev 1.0 - 25/2/2000                             **
**    Can Communication Protocol ver 2.1              **
**    and Serial Communication Protocol ver 1.0       **
**    Runs on Crimson I ecu with GPIO board           **
**    No FPU board as yet!                            **
**                                                    **
**    The CCP and SCP use the same protocol layer     **
**    stack. The only difference is in the physical   **
**    layer. The Can and Serial interrupt routines    **
**    are responsible for creating the correct format **
**    for the message                                 **
**                                                    **
**********************************************************
**********************************************************/

#include "compassos.h"

/*
      each daqlist has up to 42 odt lists. There are 6 daq lists and
//    each odt is assigned a unique PID number. This is calculated
//    by PID = DAQ list number + odt number. Therefor there can be
//    a maximum of 0xfd PID numbers. This is because 0xff is reserved
//    for the PACKETID, and 0xfe is an event message.
//    Thus 42*6=252 or 0xfc
//    There are 6 daq lists, each list can have 46 odtlists
*/

// NOTE MALLOC DOES NOT WORK - CAUSES AN ERROR!!!



// INTERRUPTS
// The two system interrupts are Serial Int and the Timer Int.
// The Serial int has a level of 5 and the Timer Int has an int level
of 4
// The CPU accepts interrupts of 4 and above until.
// the timer is enabled and disabled by starting/stoping the master
clock source.
// We can't use the CPU int level as the selection because it can't be
changed
// from within an interrupt!


// global variable - not good but necessary for the interrupt routines
to access
daq_t daqlist[DAQLIST];   /* pointer to daq list */
```

```
canframetype_t canframe;
control_t ctrl;
memory_t mem;
short int daqlistptr;
short int odtlistptr;
short int elementptr;
short int serialpacket[10];
short int serialpktno;
short int runmode;
short int runstate;


short int *scsr;
short int *scdr;
short int *portqs;
short int *cpcr;

void (*rxpacket)();
void (*coderun)();



main()
{

      int i;

      void (*intvect)();              // temp storage for interrupt
vector address
      long *serialintptr=SERIAL_INT;        // interrupt vector address
in table for serial int
      long *timerintptr=TIMER_INT;    // interrupt vector address in
table for time int

      intvect = serialint;            // get address of int routine & put
in temp variable
      *serialintptr = intvect;        // put temp variable int int vect
table

      intvect = timerint;
      *timerintptr = intvect;

      serialpktno=0;

      rxpacket = rxscimsg;            // store the address of the packet
routine


// set up the addresses for the port accesses (can't be done inline
cause of a complier error !

      scsr = SCSR;
      scdr = SCDR;
      portqs = PORTQS;
      cpcr = CPCR;

      *cpcr = 0x0000;           // stop the timer
      runmode = 0;

#asm
      move.w            #$2300,SR         * set int level at 3 to
accept 4 & above
#endasm




/* setup the station parameters */
      ctrl.station_address = STATIONID;
      ctrl.resourcemask = CAL|DAQ|DNLD|PROG;            /* all functions
available */
      ctrl.resourcepromask = CAL|DAQ|DNLD|PROG;         /* none require
security */
      ctrl.slaveid = SLAVEID;
      ctrl.slaveidsize = IDLENGTH;                      /* 32 bit slave
id */
      ctrl.slaveidtype = IDTYPE;                        /* not
implemented */
      ctrl.ccpver = VER;
      ctrl.ccprel = REL;
      ctrl.busy = BUSY;                                 /* slave is ready
*/
      ctrl.calseeddata=CALSEED;
      ctrl.daqseeddata=DAQSEED;
      ctrl.dnldseeddata=DNLDSEED;
      ctrl.progseeddata=PROGSEED;
      ctrl.calibrationkey=(CALSEED*7)+0x15;                   /* simple
but effective security */
      ctrl.daqkey=(DAQSEED*11)+0x53;
      ctrl.flashprogkey=((PROGSEED*3)-0x38119AFC)*2;
      ctrl.downloadkey=(DNLDSEED*23)+0x45;


      runstate=0;
      while(1)
      {
            if(runstate==RUN)
            {
                  coderun = ctrl.actionserv;
                  coderun();
                  runstate=0;
            }
      }




      return;


}

/* interrupt routine for the on board CAN Controller */
```

```
void rxcanmsg()
{

      dispatchmsg();


#asm
      unlk a6
      rte
#endasm
      return;
}


/* transmit routine for the can controller */


void txcanmsg()
{



      return;

}


// interrupt routine for the serial controller interface (sci)
// The data from the serial port is passed via pointer placed on the
// stack, so the first operation is to get this address off the
// stack.
// The interrupt sequence places an exception frame on the stack
// this is 3 words with the SR at the highest address, the
// high word of the program counter next and the low word of
// the PC last (lowest address)
// The pointer is placed on the stack before the frame
//


void rxscimsg()
{
      int i;


      canframe.cmd = serialpacket[0];
      for(i=1;i<8;i++)
            canframe.crodata[i] = serialpacket[i];



      dispatchmsg();

      txscimsg(); // return the packet back to the master

      if(runmode=='1')
            *cpcr = 0x000b;         // start the timer
      else
            *cpcr = 0x0000;          // stop the timer

      return;
}

/* transmit routine for the sci controller */

void txscimsg()
{
      short int i;



      while(!(*scsr & 0x100));
            *scdr = canframe.id;

      for(i=1;i<8;i++)
      {
            while(!(*scsr & 0x100));
            *scdr = canframe.crodata[i];
      }

      return;
}



/* message dispatcher for the CCP message */

void dispatchmsg()
{


        if(canframe.cmd==nConnect)
                  connect();
            else if(canframe.cmd==nDisconnect)
            disconnect();
      else if(ctrl.connectstatus == 1)                  // we must be
connected befor accessing any other commands
      {
            switch(canframe.cmd)
            {

                  case nGet_CCP_Version:
                        getccpversion();
                  break;

                  case Exchange_ID:
                        exchangeid();
                  break;

                  case nSetmta:
                        setmta();
                  break;

                  case nDnload:
                        dnload();
                  break;

                  case nUpload:
                        upload();
                  break;

                  case nGet_Daq_Size:
                        getdaqsize();
                  break;

                  case nSet_Daq_Ptr:
                        setdaqptr();
                  break;

                  case nWrite_Daq:
                        writedaq();
                  break;

                  case nStart_Stop:
                        startstop();
                  break;

                  case nAction_Service:
                        actionserv();
                  break;



                  default:
                        commanderr();

            }

      }
      else
          commanderr();

   return;
}

void serialint()
{


/**************************************************************
*
*     Serial interrupt routine
*
*     Receives a character from the serial port, puts it in
*     A buffer. After 10 characters calls the CCP layer stack
*     with the address of the buffer on the stack.
*     This uses CTS/RTS hardware handshakeing
*
**************************************************************/
```

```
      register short int temp;

      *portqs = NOCTS;              // de-assert the cts line
      temp = *scsr;                       // clear int pending
      serialpacket[serialpktno] = *scdr;        // get data from
register and put in buffer

      serialpktno++;
      if(serialpktno>=PKTLEN)             // have 10 chars been
received ?
      {
            serialpktno=0;                // reset pkt pointer
            rxscimsg();             // yes so call the CCP layer stack
      }

      *portqs = CTS;                      // re-assert CTS


#asm
      unlk  a6
      rte
#endasm

}



// control commands


// Connect
//    This command establishes a logical point to point connecton with
the
//    slave. All following protocol commands refer to this station
only,
//    until another station is selected. A Connect command to another
station
//   temporarlly disconnects the active station (see disconnect-> A
slave device
//   does not respond to any commands unless being addressed by a prior
connect
//   command with the correct station address. The station address is
specified
//   with low-byte first.

void connect()
{


      short int ctr,temp;

      ctr = canframe.crodata[1];
      temp =  canframe.crodata[3]<<8;
      temp = temp |  canframe.crodata[2];
```

```
	if( ctrl.busy)
		canframe.crodata[1]=ACCESSDENIED;
	else if(temp ==  ctrl.station_address)
	{
		canframe.crodata[1]=ACK;	// ACK
		ctrl.connectstatus = 1;		// we are connected!
	}
		else
	{
		canframe.crodata[1]=OUTOFRANGE;		// OUTOFRANGE
		ctrl.connectstatus = 1;		// we are temporarilly
disconnected!
	}


	canframe.id=PACKETID;	// PACKETID
	canframe.crodata[2] = ctr;


	return;
}


// Disconnect
//	This command disconnects the slave. The disconnect can be
temporary
//	slave.



void disconnect()
{


	short int temp,ctr;

	ctr = canframe.crodata[1];
	temp =  canframe.crodata[5]<<8;
	temp = temp |  canframe.crodata[4];

	if( ctrl.busy)
		canframe.crodata[1]=ACCESSDENIED;
	else if(temp ==  ctrl.station_address)
	{
		canframe.crodata[1]=ACK;	// ACK
		if(!canframe.crodata[2])
			ctrl.connectstatus = 2;		// we are in temporary
disconnect
		else
			ctrl.connectstatus = 0;		// we are disconnected!

	}
		else
		canframe.crodata[1]=OUTOFRANGE;		// ERROR - OUTOFRANGE


	canframe.id=PACKETID;	// PACKETID
```

```
        canframe.crodata[2] = ctr;

        return;
}

/* Get CCP Version

        This command performs a mutal identification of the protocol
version
        used in the master & slave. This command should be executed
BEFORE
        the Exchangeid command is issued

*/


void getccpversion()
{

        short int ctr;
        ctr = canframe.crodata[1];
        canframe.id = PACKETID;         //PACKETID;
        canframe.crodata[1] = ACK;              //ACK;

        canframe.crodata[3] = ctrl.ccpver;
        canframe.crodata[4] = ctrl.ccprel;

        canframe.crodata[2] = ctr;

        return;
}

void exchangeid()
{

        short int ctr,temp;
        ctr = canframe.crodata[1];

        temp =  canframe.crodata[2]<<8;                 // master id (mot
format)
        ctrl.masterid  = temp |  canframe.crodata[3];   // ie big endian


        canframe.id = PACKETID;
        canframe.crodata[1] = ACK;
        canframe.crodata[2] = ctr;
        canframe.crodata[3] = ctrl.slaveidsize;
        canframe.crodata[4] = ctrl.slaveidtype;
        canframe.crodata[5] = ctrl.resourcemask;
        canframe.crodata[6] = ctrl.resourcepromask;

        mem.mta0ext = 0;
        mem.mta0 = &ctrl.slaveid;


        return;
```

```
}

void setmta()
{

      short int ctr,mtano;
      int ltemp;
      ctr = canframe.crodata[1];    // ctr returned

      mtano = canframe.crodata[2];


      ltemp = (canframe.crodata[4] <<24) & 0xff000000;
      ltemp = ltemp | ((canframe.crodata[5] <<16) & 0x00ff0000);
      ltemp = ltemp | ((canframe.crodata[6] <<8) & 0x0000ff00);
      ltemp = ltemp | (canframe.crodata[7]  & 0x000000ff);


      canframe.crodata[1] = ACK;          //ACK;

      if(ltemp>HIGHMEM || ltemp<LOWMEM)
              canframe.crodata[1] = OUTOFRANGE;
      else if(mtano==1)
            mem.mta1 = (int *) ltemp;   // 32 bit write
      else
            mem.mta0 = (int *) ltemp;

      canframe.id = PACKETID;       //PACKETID;
      canframe.crodata[2] = ctr;

      return;

}
void dnload()
{
      short int ctr,stemp,*t,size;
      int ltemp;
      ctr = canframe.crodata[1];    // ctr returned

      size = canframe.crodata[2];   // size of required data block

      if(size == 4)
      {

            ltemp = (canframe.crodata[3] <<24) & 0xff000000;
            ltemp = ltemp | ((canframe.crodata[4] <<16) & 0x00ff0000);
            ltemp = ltemp | ((canframe.crodata[5] <<8) & 0x0000ff00);
            ltemp = ltemp | (canframe.crodata[6]  & 0x000000ff);

            *mem.mta0 = ltemp;   // 32 bit write

            mem.mta0++;

            canframe.crodata[1] = ACK;


      }
      else if(size == 2)
      {
            stemp = (canframe.crodata[3] <<8) & 0xff00;
            stemp = stemp | (canframe.crodata[4]  & 0x00ff);

            t = (short int *) mem.mta0;   // cast to short int for 16
bit write

            *t = (short int) stemp;

            t++;
            mem.mta0 = (int *) t;

            canframe.crodata[1] = ACK;          //ACK;

      }
      else
            canframe.crodata[1] = OUTOFRANGE;




      canframe.id = PACKETID;       //PACKETID;
      canframe.crodata[2] = ctr;

      return;

}
void upload()
{
      short int ctr,size,*t;
      int temp;
      ctr = canframe.crodata[1];    // ctr returned

      size = canframe.crodata[2];   // size of required data block

      if(size == 2 | size == 4)
      {
            temp = (int ) *mem.mta0;

            canframe.crodata[1] = ACK;          //ACK;


            canframe.crodata[3] = (short int) (temp >>24)&0x00ff;
            canframe.crodata[4] = (short int) (temp >>16)&0x00ff;
            canframe.crodata[5] = (short int) (temp >>8)&0x00ff;
            canframe.crodata[6] = (short int) temp &0x00ff;

            if(size==2)
            {
                  t = (short int *) mem.mta0;
                  t++;
                  mem.mta0 = (int *) t;
            }
            else
                  mem.mta0++;
```

```
	}
	else
		canframe.crodata[1] = OUTOFRANGE;


	canframe.id = PACKETID;		//PACKETID;
	canframe.crodata[2] = ctr;

	return;

}
/* This routine initalises the appropriate daq list */
void getdaqsize()
{
	short int ctr,i,x;
	int ltemp;
	ctr = canframe.crodata[1];	// ctr returned

	if(canframe.crodata[2]<0 | canframe.crodata[2]>DAQLIST)
		canframe.crodata[3] = 0;
	else
	{
		daqlistptr = canframe.crodata[2];

		ltemp = (canframe.crodata[4] <<24) & 0xff000000;
	// can id for daq lists
		ltemp = ltemp | ((canframe.crodata[5] <<16) & 0x00ff0000);
		ltemp = ltemp | ((canframe.crodata[6] <<8) & 0x0000ff00);
		ltemp = ltemp | (canframe.crodata[7]  & 0x000000ff);

		daqlist[daqlistptr].can_id = ltemp;
		x=daqlistptr*ODTLIST;

		for(i=0;i<ODTLIST;i++)
			daqlist[daqlistptr].odtlist[i].PID = x+i;


		canframe.crodata[3] = ODTLIST;
		canframe.crodata[4] = 0;				//
first PID of list
	}


	canframe.id = PACKETID;		//PACKETID;
	canframe.crodata[1] = ACK;		//ACK;

	canframe.crodata[2] = ctr;

	return;

}
void setdaqptr()
{
	short int ctr;
	ctr = canframe.crodata[1];	// ctr returned
```

```
      if(canframe.crodata[2]<0 | canframe.crodata[2]>DAQLIST)
            canframe.crodata[1] = OUTOFRANGE;
      else if(canframe.crodata[3]<0 | canframe.crodata[3]>ODTLIST)

            canframe.crodata[1] = OUTOFRANGE;
      else if(canframe.crodata[4]<0 | canframe.crodata[4]>6)
            canframe.crodata[1] = OUTOFRANGE;

      else
      {
            daqlistptr = canframe.crodata[2];
            odtlistptr = canframe.crodata[3];
            elementptr = canframe.crodata[4];

            canframe.crodata[1] = ACK;          //ACK;

      }


      canframe.id = PACKETID;       //PACKETID;


      canframe.crodata[2] = ctr;

      return;
}
void writedaq()
{

      short int ctr;
      int ltemp;
      ctr = canframe.crodata[1];    // ctr returned

      daqlist[daqlistptr].odtlist[odtlistptr].elements[elementptr].size
= canframe.crodata[2];

      ltemp = (canframe.crodata[4] <<24) & 0xff000000;            //
can id for daq lists
      ltemp = ltemp | ((canframe.crodata[5] <<16) & 0x00ff0000);
      ltemp = ltemp | ((canframe.crodata[6] <<8) & 0x0000ff00);
      ltemp = ltemp | (canframe.crodata[7]  & 0x000000ff);

      daqlist[daqlistptr].odtlist[odtlistptr].elements[elementptr].addr
ess = ltemp;


      canframe.id = PACKETID;       //PACKETID;
      canframe.crodata[1] = ACK;          //ACK;

      canframe.crodata[2] = ctr;

      return;
}
void startstop()
{

      short int ctr;
      short int *cpcr = CPCR;

      ctr = canframe.crodata[1];    // ctr returned
      runmode = (short int) canframe.crodata[2];     // run mode for
the timer interrupt

      canframe.id = PACKETID;       //PACKETID;
      canframe.crodata[1] = ACK;          //ACK;

      canframe.crodata[2] = ctr;

      return;

}
void commanderr()
{
      short int ctr;
      ctr = canframe.crodata[1];    // ctr returned
      canframe.id = PACKETID;       //PACKETID;
      canframe.crodata[1] = ACK;          //ACK;

      canframe.crodata[2] = ctr;

      return;

}
void timerint()
{

      short int temp;
      short int *statusreg = MCSM2SIC;

      temp = *statusreg;
      *statusreg = 0x4215;

      while(!(*scsr & 0x100));
      *scdr = '.';




#asm
      unlk  A6
      rte
#endasm



}
void actionserv()
{

      short int ctr;
      int ltemp;
      ctr = canframe.crodata[1];    // ctr returned

      runstate = (short int) canframe.crodata[2];
      ltemp = (canframe.crodata[4] <<24) & 0xff000000;            //
action service address
      ltemp = ltemp | ((canframe.crodata[5] <<16) & 0x00ff0000);
      ltemp = ltemp | ((canframe.crodata[6] <<8) & 0x0000ff00);
      ltemp = ltemp | (canframe.crodata[7]  & 0x000000ff);

      ctrl.actionserv = ltemp;


      canframe.id = PACKETID;       //PACKETID;
      canframe.crodata[1] = ACK;          //ACK;

      canframe.crodata[2] = ctr;

      return;


}
```

APPENDIX D – PUBLISHED WORK

1. Mobley C.G. (1999). " Non-Intrusive In-Cylinder Pressure Measurement of Internal Combustion Engines". *Society of Automobile Engineers International Congress and Exposition.* **Session 99P-191.**
2. Mobley C.G. (2000). "Wavelet analysis of non-intrusive pressure transducer traces". *Society of Automobile Engineers International Congress and Exposition.* **Session 00P-129.**

99P-191

Non-Intrusive In-Cylinder Pressure Measurement of Internal Combustion Engines

C.Mobley
University of the West of England

ABSTRACT

The monitoring of in-cylinder pressure on internal combustion engines is a costly and intrusive process. Current methods include high cost, miniature pressure transducers mounted in the cylinder head. The transducers have direct access to cylinder pressure, and give a high degree of accuracy. The disadvantage of using this intrusive method is the cost of the sensors and modification of the cylinder head. The monitoring of in-cylinder pressure is usually restricted to research engines or large marine diesels. However, the information carried in the pressure trace is significant and is of benefit in condition monitoring and control.

The aim of this research is to produce low cost, non-intrusive sensors to enable the monitoring of in-cylinder pressure, on internal combustion engines. This would enable the use of pressure information to be extended into small in-service internal combustion engines.

Preliminary results are very encouraging and show a high correlation between the non-intrusive sensors, and the in-cylinder pressure transducers.

This paper details the development of the sensors, instrumentation, algorithms and shows the preliminary results.

The conclusions drawn in this paper are that a system for the measurement of in-cylinder pressure using low cost non-intrusive sensors is achievable. The results from this work has shown that the integration of real-time pressure information into the control system is now possible using low cost components.

INTRODUCTION

The concept of using external sensors to detect the performance of internal combustion engine has been well documented. It is common practice to detect the onset of engine knock by monitoring engine vibration. Glibbery [1] investigated this effect and summarised much of the earlier work in this field.

Ordubadi [2] conducted work into obtaining cylinder pressure information from the excitation of the cylinder block. The vibration transfer function was developed and pressure information reproduced. The information suffered from the excitation of the block by all four cylinders, structural resonance within the block and contamination by other noise sources.

Kirkman and Elliott [3] conducted research into strain gauges placed under single cylinder head bolts of a large diesel engine. In their conclusions they indicate that for engines large enough to warrant individual cylinder heads, or large single cylinder engines, cylinder pressure correlates well with the measured bolt strain.

In the previous work, the cost and size of the sensors were a limiting factor in reconstruction of cylinder pressure. Also, the cost and performance of Digital Signal Processing equipment contributed to the problems encountered in isolating the effects of multiple cylinders.

The aims of the research presented in this paper are: 1, to reproduce the results obtained by Kirkman and Elliott by developing a sensor for incorporation into a single cylinder diesel engine of 412cc, and 2, to migrate this to a 4 cylinder diesel engine of 1800cc.

MAIN SECTION

Piezoelectric sensors

Piezoelectricity is a property of certain classes of crystalline materials. When mechanical pressure is applied to one of these materials, the crystalline structure produces a voltage proportional to the pressure. Conversely, when an electric field is applied to one of these materials, the crystalline structure changes shape, producing dimensional changes in the material. Piezoelectric properties occur naturally in some crystalline materials and can be induced in other polycrystalline materials. Many contemporary applications of piezoelectricity use polycrystalline ceramics instead of natural piezoelectric crystals. These piezoelectric ceramics are more versatile and their properties tailored to a specific use. The material used in this application was PZT 5A (Navy Type II) [4]. The dimensions of the material were a 2mm thick, 15x25mm washer.

$$\text{Capacitance} = \frac{\text{Relative dilectric constant} * \in_o * \Pi * (od^2 - id^2)}{4 * \text{thickness}}$$

For PZT 5A   Relative dilectric constant = 1700 at 25°C

$\in_o = 8.85 \times 10^{-12}$ farads/meter

$$\text{Capacitance} = \frac{1700*8.85\text{x}10^{-12} * \Pi * (0.025 - 0.015)}{0.004}$$

For the washer of PZT used in the MK1 sensor = 118 nF

Equation 1: Device size and capacitance

In the work by Kirkman & Elliott, the change in load on a head bolt was measured by strain gauges, another method is to use the piezoelectric effect. A PZT washer placed under the head bolt will be subjected to the load variations caused by the change in cylinder pressure, and can reproduce this as a change in charge. Equation 1 shows the relationship between device size and capacitance for the material PZT 5A. When the element is subjected to an external force it generates a charge, which can be thought of as a current source in series with a capacitance. Standard amplifier techniques can be used to amplify this signal. However, The piezoelectric effect can only register a change in charge and cannot give an absolute reading.

The system used for data acquisition was specifically designed for this project. It consists of a high speed analogue to digital module, a digital signal processor and a controller module. Figure 1 shows the arrangement of the system. The system has been designed for the high speed analysis and control of mechanical systems, and enables rapid prototyping for the non-intrusive sensor project.

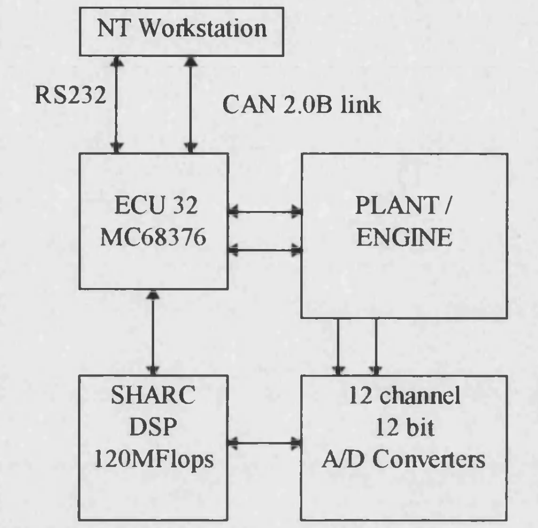


Figure 1: Analysis & Control System

Single Cylinder Engine Trials with Mk1 Sensor

To achieve the first aim of this research a single cylinder 'Robin' diesel engine of 412cc was used. This engine was equipped with a Kistler transducer, mounted inside the cylinder head, in order to accurately measure 'in-cylinder' pressure. The engine is an air cooled unit with exposed cylinder head studs, and has sufficient clearance for mounting of the Mk 1 sensor.

Figure 2 shows the construction of the Mk1 sensor. The construction of the sensor is based upon a washer of PZT 5A. The washer of material has electrical connections on either face and contact washers are placed either side. The contact washers are made from printed circuit board, type FR4. This has a copper deposit on a glass fiber base and enables solder connections to be made. Steel load washers complete the 'washer stack'.

The Mk1 sensor was placed under one of the head studs and a torque of 40Nm was applied. The engine was run at various loads and speeds and data captured from the two transducers. Figure 5 shows a comparison of in-cylinder pressure measured by the Kistler transducer and the Mk1 sensor. By inspection a strong correlation can be seen between the two traces. Data collected at other loads and speeds also showed this correlation. This corresponds with similar results obtained by Kirkman & Elliott.

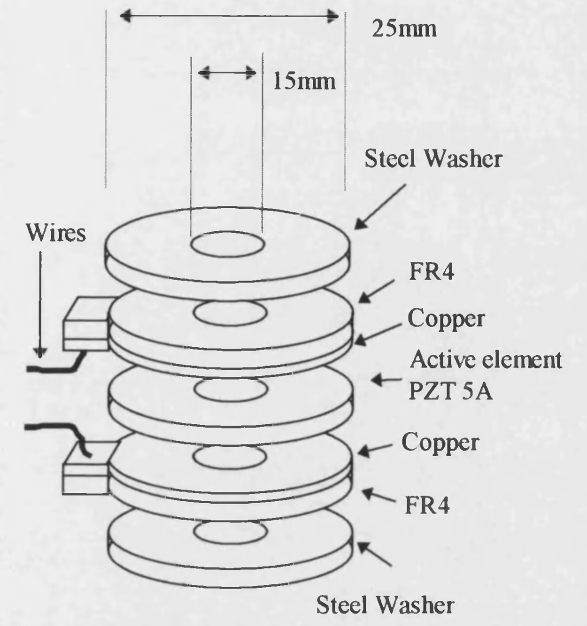


Figure 2: Mk1 sensor

Single Cylinder Engine Trials with Mk2 Sensor

The Mk1 sensor used a washer as the active element. This gave good results as the variation in bolt compression acted directly on the whole of the PZT. A compressive force of 40Nm was excerpted on the PZT by the clamping of the stud bolts, and the PZT recorded variations in compression around this mean.

It was realised that a similar arrangement could not be used on the 1800cc 4 cylinder engine as the clamping torque was 180Nm. An experiment confirmed that 180Nm was sufficient to reduced the PZT washer to powder. A new arrangement was required which could withstand the clamping torque.

The solution was to use a load sharing device which could take the majority of the torque. Figure 3 shows the arrangement of the Mk2 sensor. The active element is now a small device captured in a machined pocket within the structure. When the clamping torque is applied the majority is passed through the structure and a fraction is applied to the PZT. The PZT now measures the variation in clamping loads but experiences a reduced peak torque.

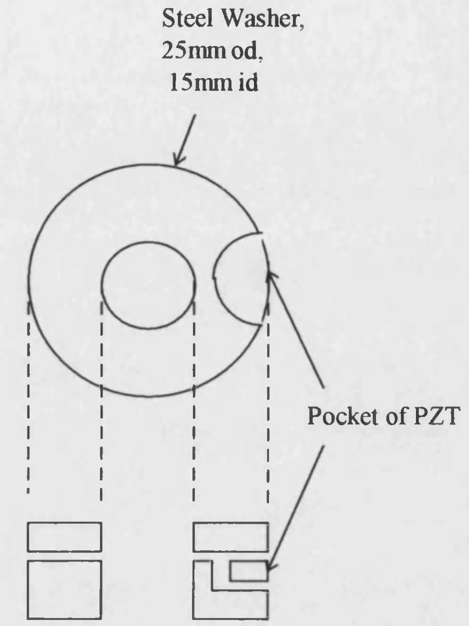


Figure 3: Mk2 Sensor design

The results of the new sensor are shown in Figure 6. At first inspection the signals are different, however if an integration is performed, a resultant is obtained which closely matches the original. Figure 7 shows the raw Mk2 sensor data and the integrated result.

Multi-Cylinder Engine Trials

Two Mk2 sensors were placed upon a 4 cylinder 1.8L diesel engine. The position of the sensors on the cylinder head is shown in Figure 4.

Sensor A is positioned under the camshaft-head nut close to Cylinder 1 and Sensor B is placed under head nut close to cylinder 4. These positions were chosen as the least likely to cause delays in the engine research programme and were not the ideal positions. Ideal conditions would have seen the 'in-cylinder' transducer mounted in number 2 or 3 cylinder, and 4 Mk2 sensors positioned around the cylinder.

In Figure 8 the Kistler in-cylinder transducer is shown with Sensor A. Figure 9 shows the Kistler and Sensor B. The traces show considerable noise and Initial inspection of the raw data is disappointing with no visible signs of correlation. However, simple low-pass filtering reveals an underlying variation in the signal which is visually similar to the Kistler data. This is shown in Figure 10.

CONCLUSION

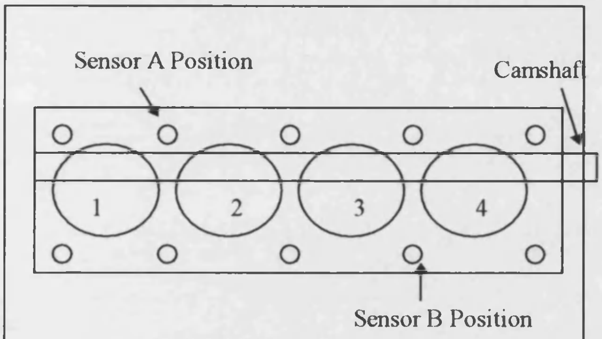


Figure 4: Arrangement of Sensors on 4 cylinder engine

This research has continued the work started by Kirkman and Elliott. They established that it was possible to infer cylinder pressure from the forces exerted on the cylinder head bolts. Their research had been based upon single cylinder, large diesel engines, or multi-cylinder engines with separate cylinder heads.

The first phase of this research, outlined in this paper, set out to develop miniaturized sensors which could be used on small, single and multi-cylinder engines. The active element chosen has resulted in sensors of 5mm in diameter being developed, with applications on small internal combustion engines. The research has reproduced the results obtained by Kirkman and Elliott using different materials and techniques, and established a baseline for design of the subsequent sensors.

The second phase of the project is to develop the algorythms which will convert the noisy multi-cylinder engine data into signals capable of being used in pressure analysis. Further work is continuing in this area and a new design of sensor is being investigated.

The new Mk3 sensor is 5mm in diameter and 1mm thick. Up to 20 of these sensors are embedded into the gasket of the engine and are arranged in an array. Ultra high speed Digital Signal Processing with SHARC processors will record and process the information. With the increased information which 20 sensors will give advanced DSP techniques may reproduce the pressure trace.

An investigation into using the sensors as a non-intrusive dynamic timing system is underway and may be able to offer standard ECU's an accurate start of combustion signal for incorporation into engine control systems.

ACKNOWLEDGMENTS

Dr. G Hawley, University of Bath.

Mr. G Bird, Ford Motor Company.

CONTACT

Chris Mobley
University of the West of England
+44 (0) 1174 965 6261
+44 (0) 1225 311749
chris@twois.demon.co.uk

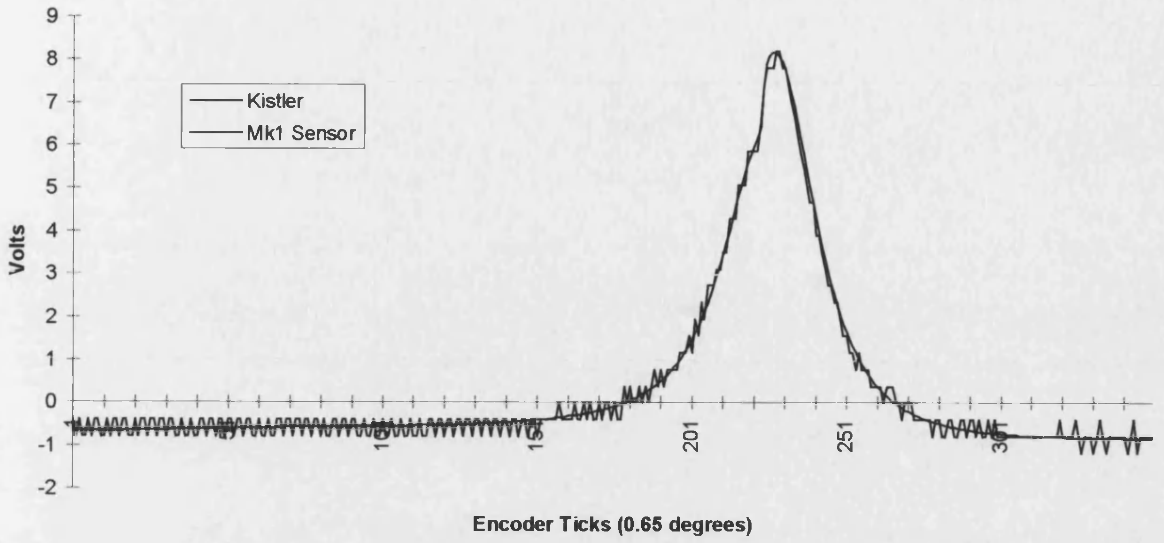


Figure 5: Mk1 Sensor & Kistler for Single Cylinder engine.

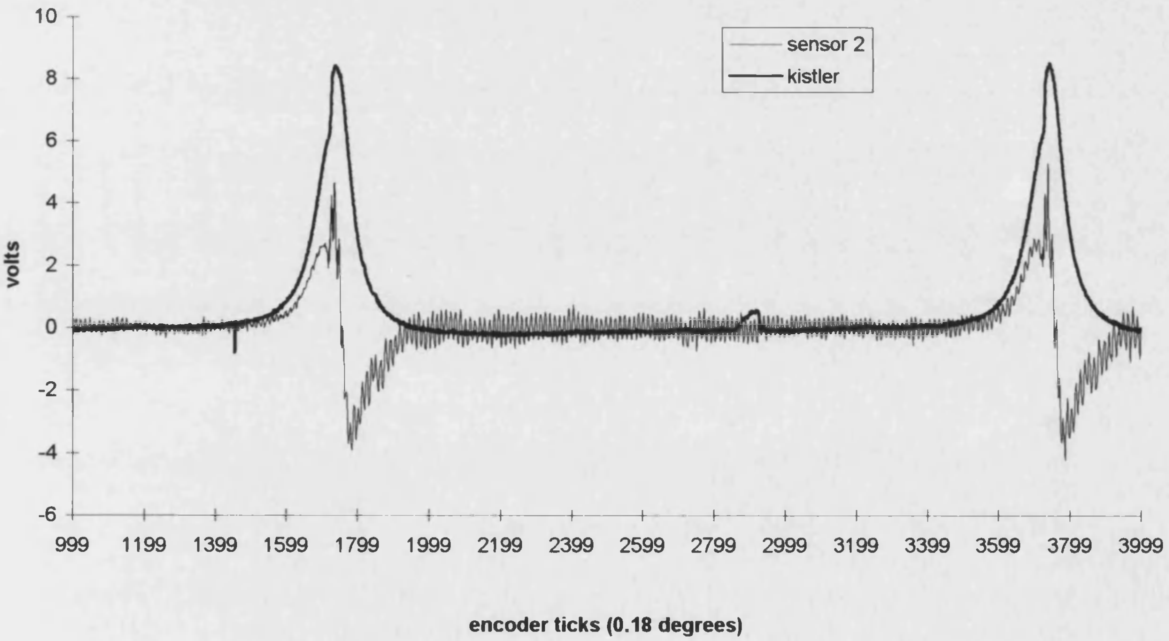


Figure 6: Kistler & Mk2 Sensor on Single Cylinder Engine

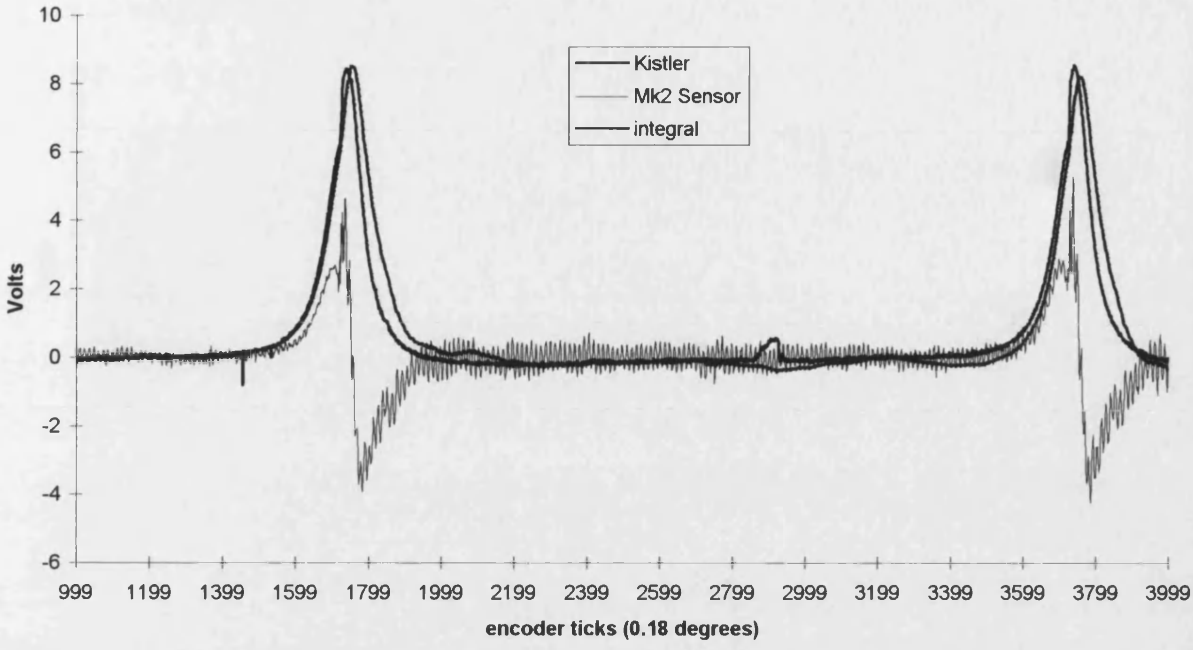


Figure 7: Kistler, Mk2 Sensor & Integral on Single Cylinder engine

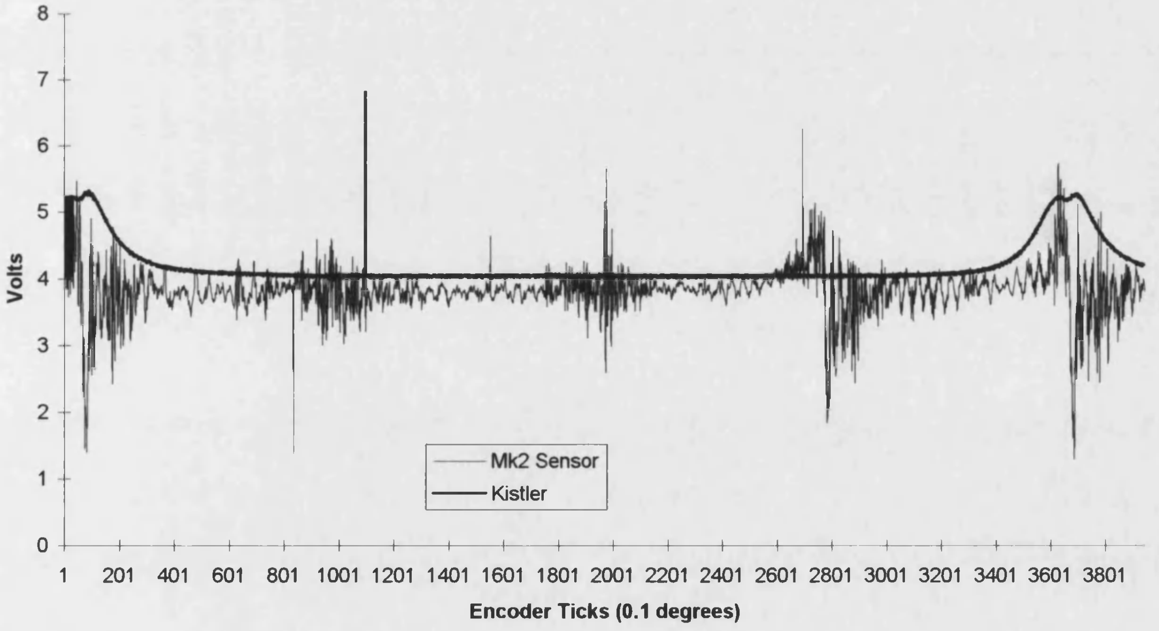


Figure 8: Kistler in No 1 Cyl. & Mk2 Sensor in position A, near Cyl. 1

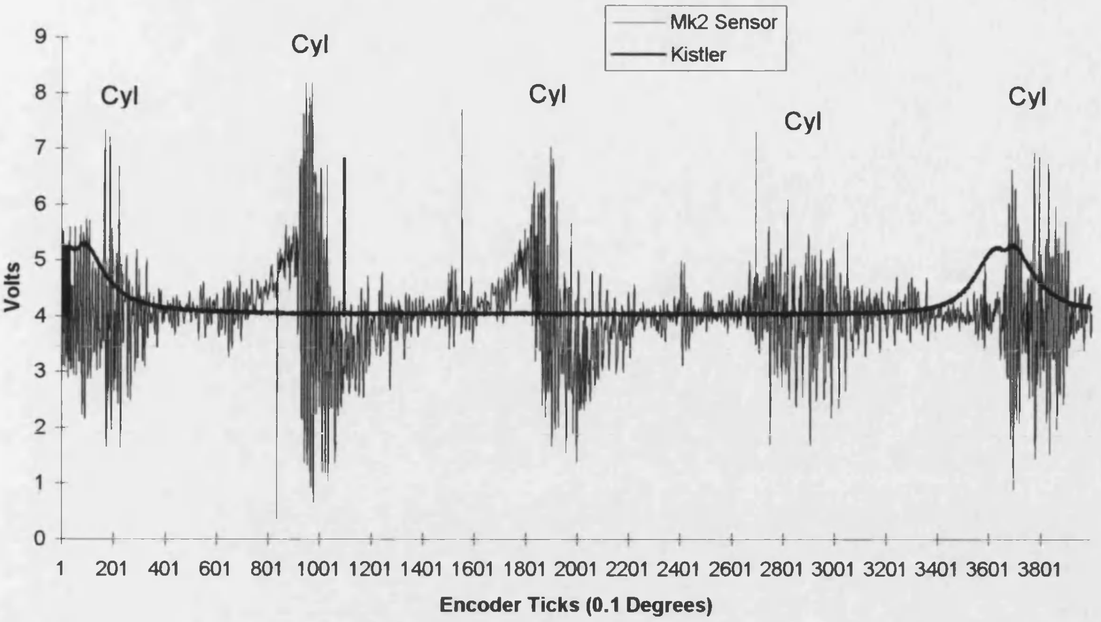


Figure 9: Kistler in No 1 Cyl. & Mk2 Sensor mounted in position B near Cyl. 4

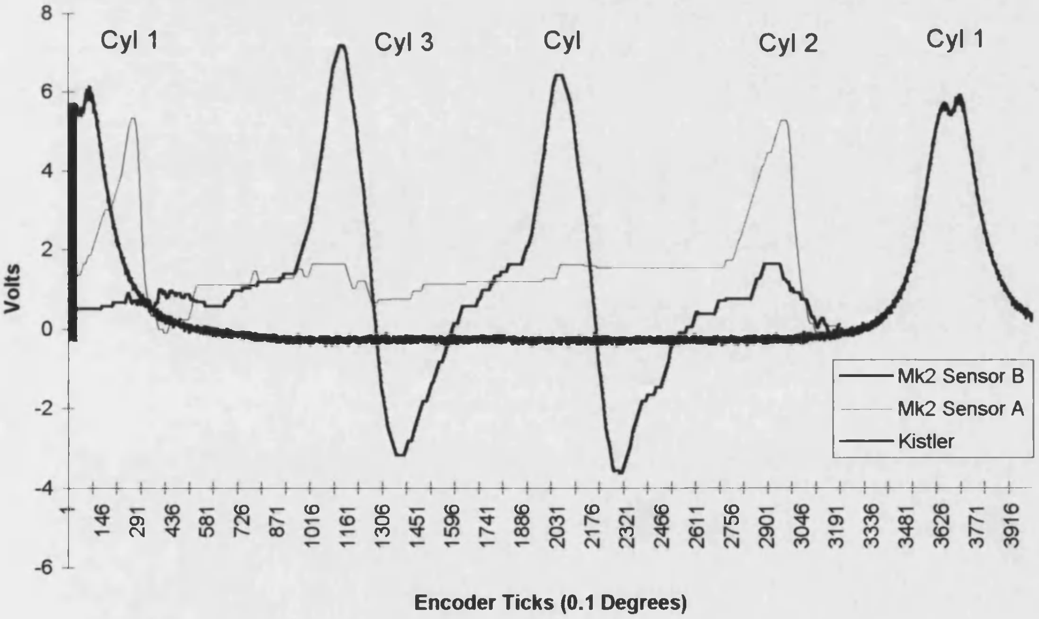


Figure 10: Kistler in No 1 Cylinder, Mk2 Sensor A & B with simple Low Pass filters.

FERENCES

libbery, R.J.: "Correlation between detonation, pressure and vibration in a petrol engine", M.Sc. Thesis, Institute of Sound and
ration Research, Southampton University, 1987.
rdubadi, A.: "Fault detection in internal combustion engines using an acoustic signal", SAE Paper 820365, Proceedings of
gine Noise Symposium, International Congress and Exposition (of the SAE), Detroit, Michigan, February 22-26, 1982.
irkman E.T.F, Elliott C.: "Condition monitoring of marine engineering equipment using neural computing", The Institute of
rine Engineers, Transactions, Volume 107, Part 3, 1995.
lorgan Matroc Limited, Thornhill, Southampton, Hampshire, SO19 7TG, UK.

00P-129

Wavelet analysis of non-intrusive pressure transducer traces

C. Mobley
University of the West of England

ABSTRACT

Wavelets have been established as an essential tool for the analysis of non-stationary signals. This paper presents a novel approach to the analysis of non-intrusive pressure traces using wavelets. It starts with a background to wavelets, how they might be useful for the analysis of signals from internal combustion engines and some results obtained from the analysis of the non-intrusive in-cylinder pressure data.

The signals obtained from the non-intrusive in-cylinder pressure sensors contain random and statistical noise. They are also contaminated with reflections and resonances from the structures within the engine. Results presented at SAE99 [ref 1] show that the signals contain useful information for the determination of timing and pressures. The purpose of this paper is to present a novel approach to the processing and analysis of these signals.

INTRODUCTION

The signals obtained from an engine are usually contaminated with noise or cross-talk. The main purpose of Digital Signal Processing (DSP) is to manipulate the information contained within the signal to enhance the 'view' of a desired feature. DSP systems can only rearrange the representation of the original data, they cannot create new data.

To manipulate the information contained within a signal mathematical transforms are used. The transforms enable the original data to be represented in a different mathematical form, called a space or domain. This new domain may show features of the data, which were concealed in the original. The choice of the transform depends upon the nature of the original signal, and the desired features to be extracted. Another important aspect of transforms is that they often allow a simplification of a mathematical process.

The use of the Wavelet [ref 2], [ref 3] transform in digital signal processing enables certain aspects of the original waveform to be viewed and analysed more conveniently. The Wavelet transform gives an additional means to describe the original data and to extract any important features.

MAIN SECTION

Engine Data

Work carried by Kirkman and Elliott [ref 4] showed that cylinder head bolt strain caries valuable information about the combustion process. A sensor was designed to fit under the cylinder head bolt and measure the change in compressive force. The active element was a washer made of Piezo

Ceramic. Initially the sensor was tested on a single cylinder diesel engine of 412cc. The results shown in Figure 1 are a comparison between the in-cylinder pressure transducer and the non-intrusive transducer. The sensor was then migrated to a 4 cylinder diesel engine of 1800cc, but the signal suffered badly from structural resonances and other noise sources.

Figure 2 shows two waves plotted against crank angle, one is an in-cylinder pressure transducer and the other is a non-intrusive pressure transducer. Some features can be visually identified from the trace, however start of combustion and other details are entirely hidden in noise and cross-talk. We need to find a method to remove the unwanted contamination so that the features of interest are more readily identifiable. Figure 3 shows the non-intrusive Start of Combustion sensor which has been developed using techniques described in this paper.

Before describing the de-noising process used in Figure 3, some background on Fourier transforms and Wavelets is needed.

Continuous waves and the Fourier Transform

The spectrum content of a 2Π periodic function can be developed by a sine and cosine series expansion. Equation 1 shows the relationship between the complex exponential function and the trigonometric functions. This trigonometric expansion creates a sum of Sines and Cosines at frequency multiples of the original function, and is called the spectrum of the function.

Any 2Π periodic function can be broken down into its spectrum components. However, it is implicit in the expansion of the function that the function extends from -∞ to +∞. This implies that the 2Π periodic function is stationary. Figure 4 shows a square wave in the time domain and its corresponding spectrum.

Digital representation of signals

In order to process signals by a digital computer, sampling must be used. The function of sampling can be written in mathematical form Equation 2. It is the sum of the impulse functions at period T from -∞ to +∞ . The sampled signal is then the convolution of the impulse function and the continuous function over -∞ to +∞.

This convolution sum introduces a repetition into the spectrum of the signal. The spectrum of the sampled signal now contains multiples of the spectrum of the continuous function, with the repetition at the sample rate.

To develop the Fourier transform of the sampled signal it must be modified by replacing s(t) with s(nT), Equation 3. This represents the sampled signal, and we must limit the set of numbers on which the calculations can be perfomed to a finite value N. This sampled version of the Fourier transform is called the Discrete Fourier Transform or DFT.

Digital Filters

The sampling function introduced the notion of convolution. To obtain the sampled signal the impulse function was convolved with the continuous signal. If the sampled signal is convolved with a different function then the result is a linear modification of the signal Equation 4.

A practical demonstration of this is to try and remove a high frequency noise from a low frequency signal. The test signal and its frequency spectrum are given in Figure 5. In order to remove the unwanted noise a low pass filter is required. A simple method of obtaining the coefficients of the impulse response of the filter, is to expand its frequency function into a Fourier Series and produce an approximation. Figure 6 shows the Low Pass frequency spectrum characteristics superimposed on top of the noisy signal, and its impulse response.

By convolving the impulse response of the filter with the noisy signal, we have achieved the suppression of the noise but not the signal.

Time windows

In expanding a signal into trigonometric series it is implicit that the signal extends from $-\infty$ to $+\infty$. However, this may not be true of the original signal. To overcome this limitation the Fourier transform must be modified to accommodate a time limit or 'window' placed on the signal. However, if the time window is a square wave, where the signal is simply truncated, then the effect on the spectrum is to introduce large errors. Several methods exist which enable the edge effects of the window to be reduced. Such methods, for example Hanning, Hamming and Kaiser, are called windowing functions. Figure 7 shows an example of the Kaiser window.

One remarkable effect of these windows, is the ability to localise disturbances in time as well as frequency. The use of a window in time and a filter characteristic in the frequency spectrum can be used to attenuate all signals outside of this region. We now have a method for analysing signals which are non-periodic. This method is called the Short Term Fourier Transform (STFT).

To illustrate the problem of the using the DFT, it was applied to the non-intrusive sensor data. This is shown in Figure 8. In using the DFT all the signal, at all points in time, effect the spectrum. This clearly shows that the spectrum of a non-stationary wave cannot disseminate time varying information.

Wavelets and Multi-Resolution Analysis

The two main problems with the STFT are the decay of the window function and the characteristic of the frequency window. A perfect filter produces infinite ripples in time and a perfect window produces infinite harmonics in frequency. The solution is to find the compromise between the filter characteristic and the time window. This solution is called a Wavelet and its application the Discrete Wavelet Transform (DWT). The application of the DWT to different time windows and frequency spectrums is called Multi-Resolution Analysis (MRA).

One problem with using wavelets is that we cannot develop a perfect time-frequency filter. That is, we can develop a good time window, but at the expense of the frequency characteristic, and we can develop a good frequency filter at the expense of the time window.

Fortunately, the characteristic of the information contained in real world signals from many mechanical systems, is that low frequency waves tend to exist over long time periods, and high frequency events have short time durations. So (MRA) is designed to give good time resolution but poor frequency response at high frequencies and good frequency response but poor time resolution at low frequencies.

In MRA, a bank of filters could used to implement the series of DWTs. However, in the decomposition of the signal into its respective frequency bands there is much redundancy. It is possible that the signal need not be fully decomposed for features which are of interest to emerge. In a technique called 'Sub-band coding' two DWT filters are used to split the signal into low and high frequency components. These sub-signals can be further split into high and low frequencies. This splitting can proceed until a tree of sub-signals has developed. At each level of the tree a narrower frequency band results. Thus the tree is a pyramid of signals in frequency and time. The tree can be used to identify features within the signal at various levels. Figure 9 shows a filter decomposition tree, and Equation 5 details the process.

Quadrature Mirror Filters

One of the main aspects of any transform is the ability to transform back into the original domain. In Fourier analysis we need to reconstruct the signal from its transformed Fourier coefficients – the Inverse Discrete Fourier Transform (IDFT). The same is true in MRA for the DWT.

In MRA, the DWT filters are used to decompose the signal into a set of coefficients. In order to reconstruct the original signal from the set of coefficients, an inverse set of digital filters are required – the IDWT. The filters used in the decomposition & recombination process (DWT-IDWT) must be carefully chosen so that perfect reconstruction can be achieved. The filters used in the DWT-IDWT to perform decomposition and reconstruction, are called Quadrature Mirror Filters (QMFs). Figure 10 shows the arrangement of a QMF. There are a number of wavelets which can be used as QMFs [ref 5], [ref 6], Figure 11 shows the db6 wavelet and Figure 12 its corresponding frequency characteristic.

If there is no need to reconstruct the original signal from the transform, then the transform wavelet pair is not needed and the wavelet transform can be used with any wavelet.

In the process of reconstruction, the wavelet coefficients are recombined using Quadrature Mirror Filters. If the coefficients of the DWT are altered by some function then perfect reconstruction cannot take place. In some circumstances this is a desired effect. Noise can be effectively removed by modifying one or more of the wavelet coefficients.

Detection of Start of Combustion

The non-intrusive sensor waveform contains start of combustion infomation but it is obscured by other sources of signals. Wavelet analysis can be used to detect the turning points and discontinuities in the waveforms caused by the start of combustion. The method used here is to de-noise the original signal by trying to suppress all frequencies which are not related to the start of combustion. The signal is first decomposed by using a 'db2' wavelet to the fifth level of detail. To de-noise a signal all wavelet coefficients are passed through a threshold value. Any value below this level is set to zero. Thus only signals which have amplitudes above the threshold will be preserved. The threshold values were derived from trial and error. Since the threshold value is different for each set of wavelet coefficients the thresholds can be chosen to give the best 'view' of the start of combustion. Once the signals are passed through the threshold, they are then reconstructed. Figure 13 shows the reconstructed signal.

To detect the start of combustion, the reconstructed signal is then decomposed using a 'db2' wavelet to the 4th level. It is in the coefficients of the 4th level that the start of combustion shows up most clearly. Figure 3 shows the level 4 coefficients and Figure 14 shows the arrangement of the wavelet signal processing system.

SUMMARY

The main aim of signal processing is to try and isolate some desired characteristic of a signal. In this process we can examine the signal in many different domains. The time and spectrum domains are frequently used to provide this isolation function and the Fourier Transform enables the move between the two domains.

One method for isolating the desired characteristic of the signal is through the use of frequency filters. A filter is designed to remove unwanted frequencies from the spectrum of a signal. To achieve this, the time response of the filter is convolved with the time response of the signal. The problem with conventional filters is that they operate on the signal over all periods in time, and only isolate the characteristic of the signal in frequency not time.

However, there exists a special type of filter– called a 'wavelet' – which is used to provide the isolation in time and frequency. The width of the time and frequency windows of the filter is a compromise, and they cannot be minimised simultaneously. This is not such a problem with real world signals from mechanical systems as they already exhibit this characteristic.

To apply the time and frequency isolation the signal is decomposed by the Discrete Wavelet Transform. In the transfomed state the signal can be examined and unwanted elements within the signal removed. The signal is then recombined by the Inverse Discrete Wavelet Transform without the unwanted elements.

CONCLUSION

Wavelets have been used in numerous signal processing systems with varying degrees of success. They have been successfully applied to data compression, communications and other systems.

The novel application of wavelets to the detection of start of combustion has had some success. It has been shown that non-intrusive sensors can be small enough to placed under cylinder head bolts of passenger car engines. The sensors are of very low cost and can supply useful information to the engine management system. They are not a panacea for engine control and cannot replace the costly in-cylinder pressure transducer for accurate research type information. However in conjunction with other engine sensors they can help to build up a more complete picture of the engine for better control.

It has been shown that wavelets provide the DSP engineer with another powerful tool for detection and analysing waveforms, and the number of applications in the field automotive engineering is rapidly expanding.

Before a working prototype can be developed, several areas of further work need to be investigated. These are: (1) the threshold values for the de-noiser need to be produced by a repeatable process, not from trial & error. (2) The selection of wavelets for the decomposition must be optimised and (3) accuracy of the method has yet to be established.

To set the de-noise levels, standard methods rely on statistical processes such as standard deviation and mean square errors, however these are not so successful at producing the desired isolation of the data. Investigation into other techniques for estimating the noise present, and thus the de-noise level, are being conducted.

The selection of the wavelets for the process, effects the performance of the isolation technique. Investigation into the properties of the wavelet families is being conducted, with the aim of eliminating unfavorable wavelets.

Finally, the accuracy of the sensor needs to be established, with data recorded from different engines at different operating conditions.

ACKNOWLEDGMENTS

Dr G.Hawley, University of Bath

Mr. G. Bird, Ford Motor Company

Mr. A.Solomonidies, University of the West of England

CONTACT

Mr. C.G.Mobley

Faculty of Computer Studies and Mathematics,

University of the West of England,

Frenchay Campus,

Coldharbour Land,

Bristol, BS16 1QY.

UK

+(44) 1179 656261 x 3167

Email: christopher.mobley@uwe.ac.uk

REFERENCES

1. Mobley, C.G., *Non-Intrusive In-Cylinder Pressure Measurement of Internal Combustion Engines*, SAE Electronic Engine Controls 1999: Sensors, Actuators, and Development Tools (SP-1418), pp. 95-102.

2. Daubechies, I., *Orthonormal bases of compactly supported wavelets*, Comm. Pure Appl. Math., 41 (1988), pp. 909-996.

3. The Math Works Inc. *Wavelet Toolbox for MatLab.*

4. Kirkman, E.T.F and Elliott, C. *Condition monitoring of marine engineering equipment using neural computing.* The Institute of Marine Engineers. Transactions Vol. 107, Part3, 1995. pp185-194.

5. Mallat, S., *Multiresolution representation and wavelets*, Ph.D. thesis, University of Pennsylvania, Philadelphia, 1988.

6. Chui C.K., *Wavelets: A Mathematical Tool for Signal Analysis*, siam. ISBN 0-89871-384-6

APPENDIX I

Equations

Equation 1: The Fourier transform decomposes a signal to complex exponential functions of different frequencies. It is defined by the following equations:

$$X(f) = \int_{-\infty}^{\infty} x(t) * e^{-2j\Pi ft} dt$$

$$X(f) = \int_{-\infty}^{\infty} x(t) * (Cos(2\Pi ft) + Sin(2\Pi ft)) dt$$

Equation 2: The sampling equation

$$u(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$$

Equation 3: The Discrete Fourier Transform (DFT)

$$X(f) = \sum_{n=0}^{N-1} x(nT) e^{-2j\Pi fnT}$$

Equation 4: A digital filter can be realised by convoluting the impulse response of the filter h[n] with the signal x[n].

$$y[n] = x[n] \times h[n] = \sum_{k=0}^{N-1} x[k] h[n-k]$$

Equation 5: To increase the scale of a signal (rather than the filter response) the signal is sub-sampled. This can be achieved by using every other sample. Note that the h[2n-k] is the high pass filter response and the g[2n-k] is the low pass filter response.

$$y_{high}[n] = \sum_{k=0}^{N-1} x[k] h[2n-k] \qquad y_{low}[n] = \sum_{k=0}^{N-1} x[k] g[2n-k]$$

APPENDIX II

Figures

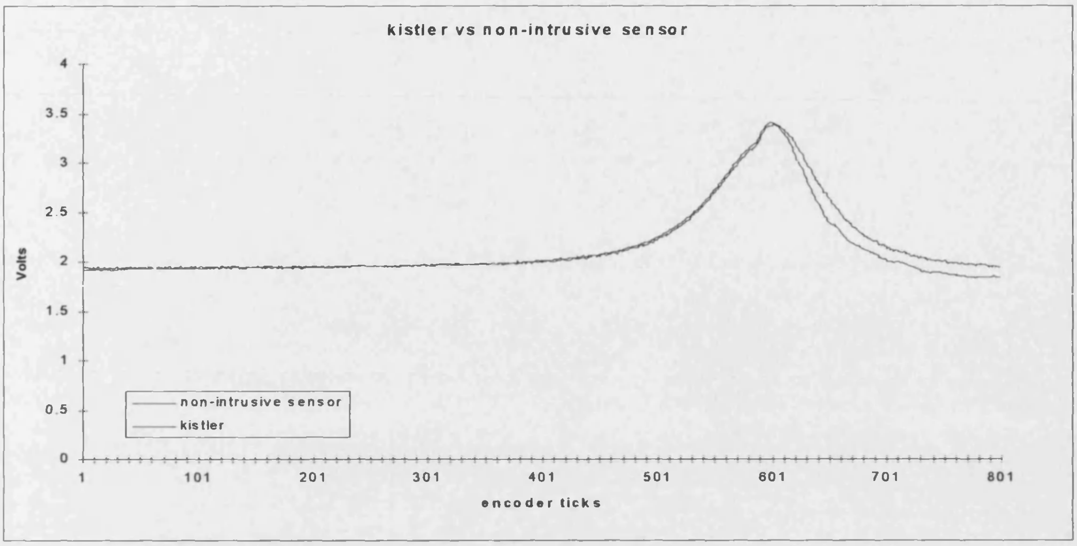


Figure 1: Kistler vs non-intrusive sensor

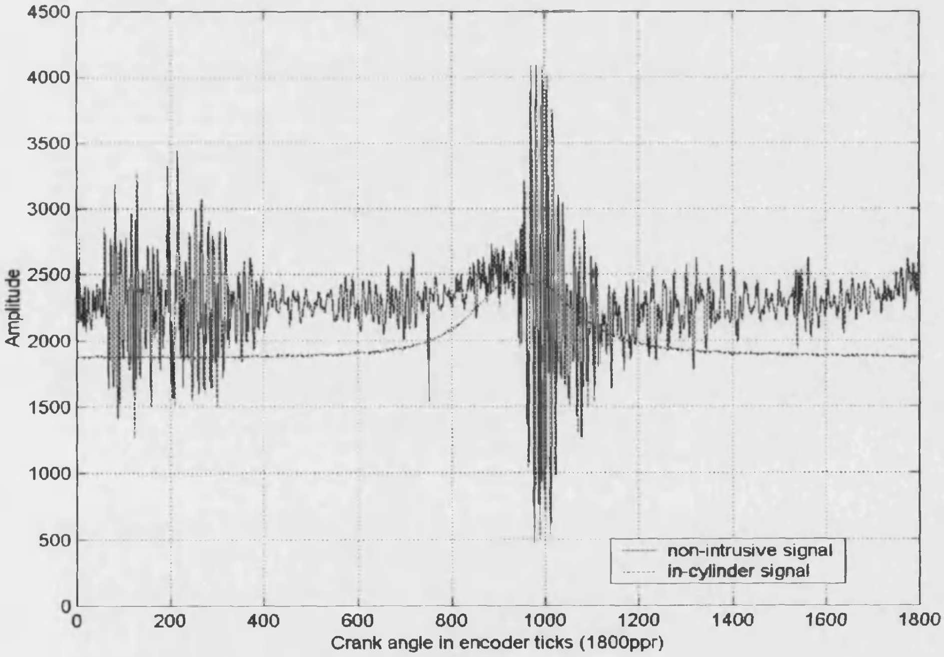


Figure 2: non-intrusive sensor vs. Kistler

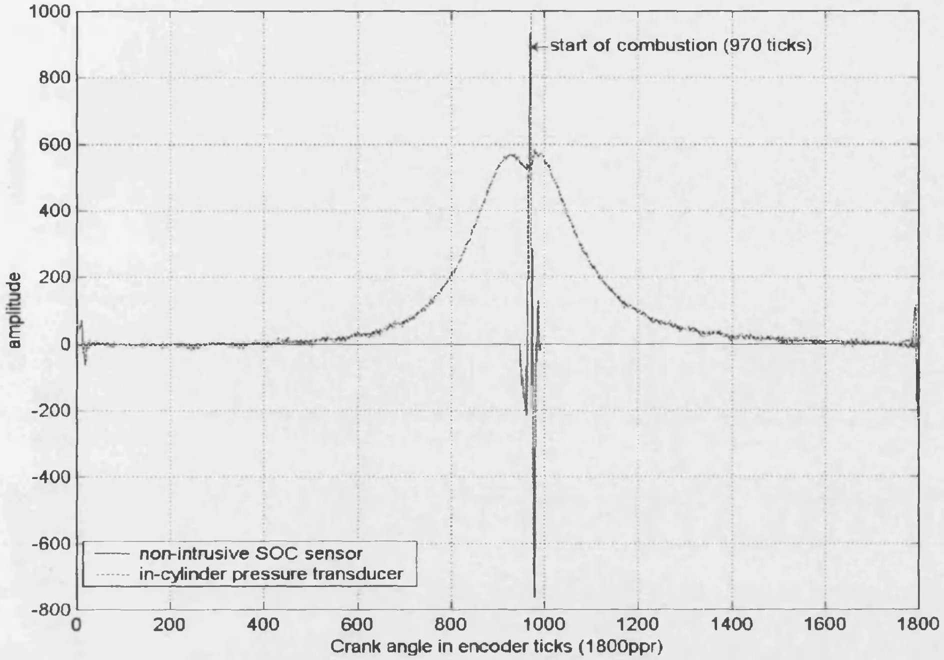


Figure 3: Start of combustion

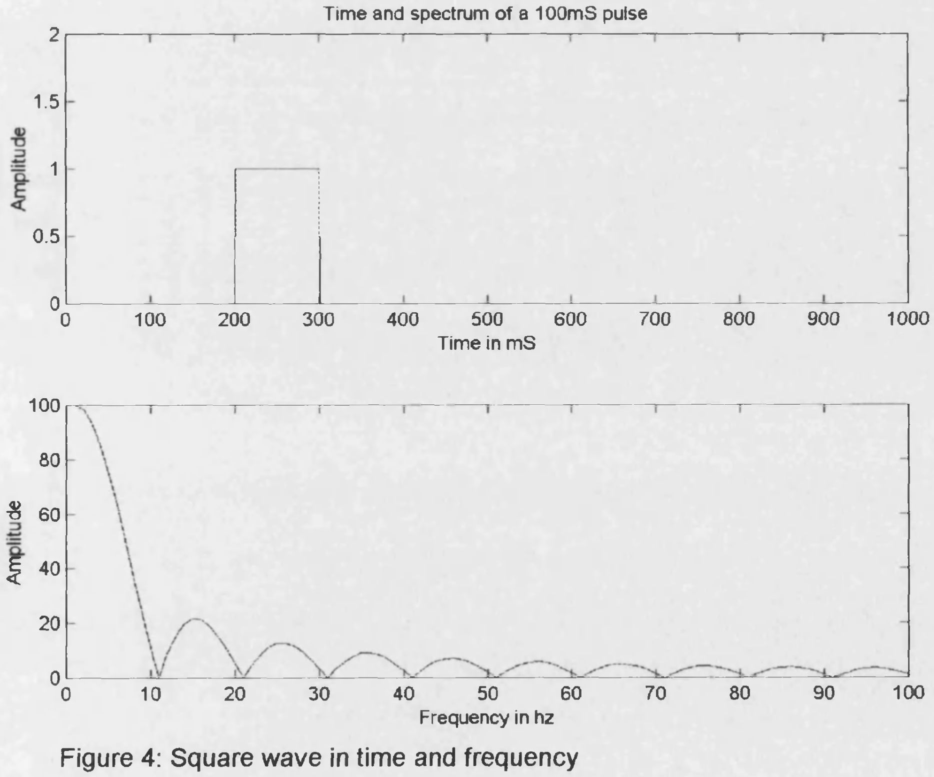


Figure 4: Square wave in time and frequency

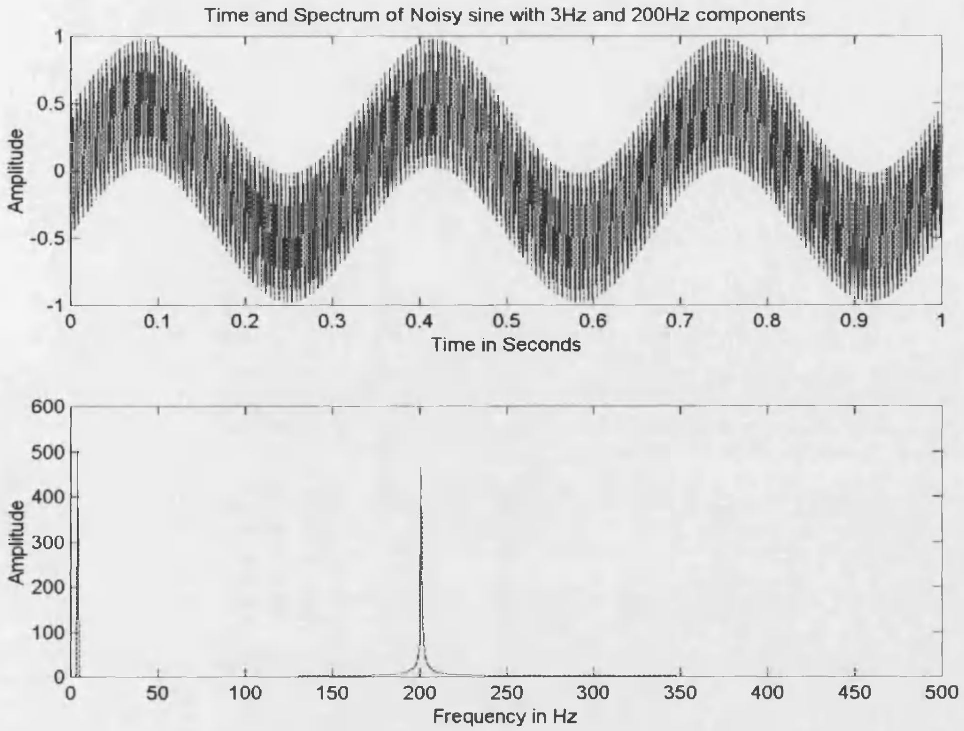


Figure 5: Noisy sine wave in time and frequency

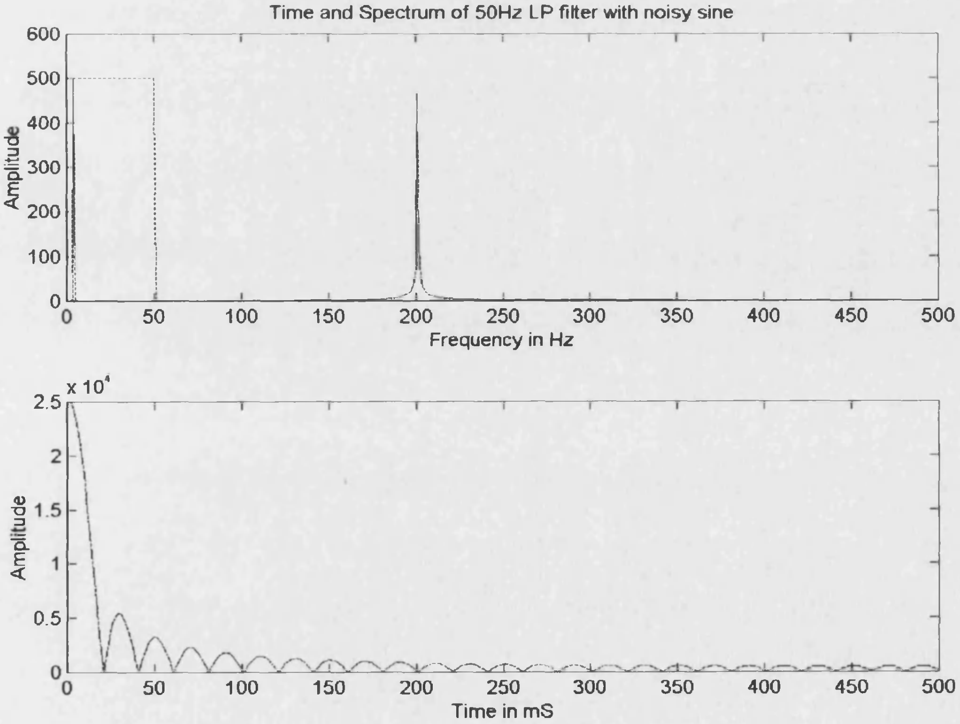


Figure 6: Low Pass Filter in frequency and time

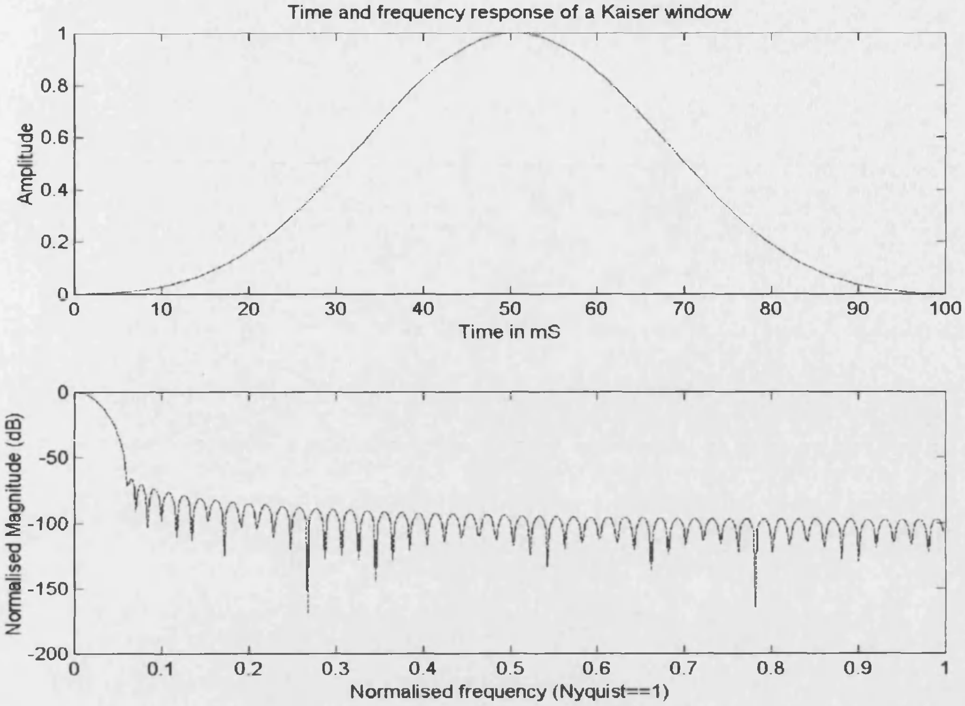


Figure 7: Kaiser window in time and frequency

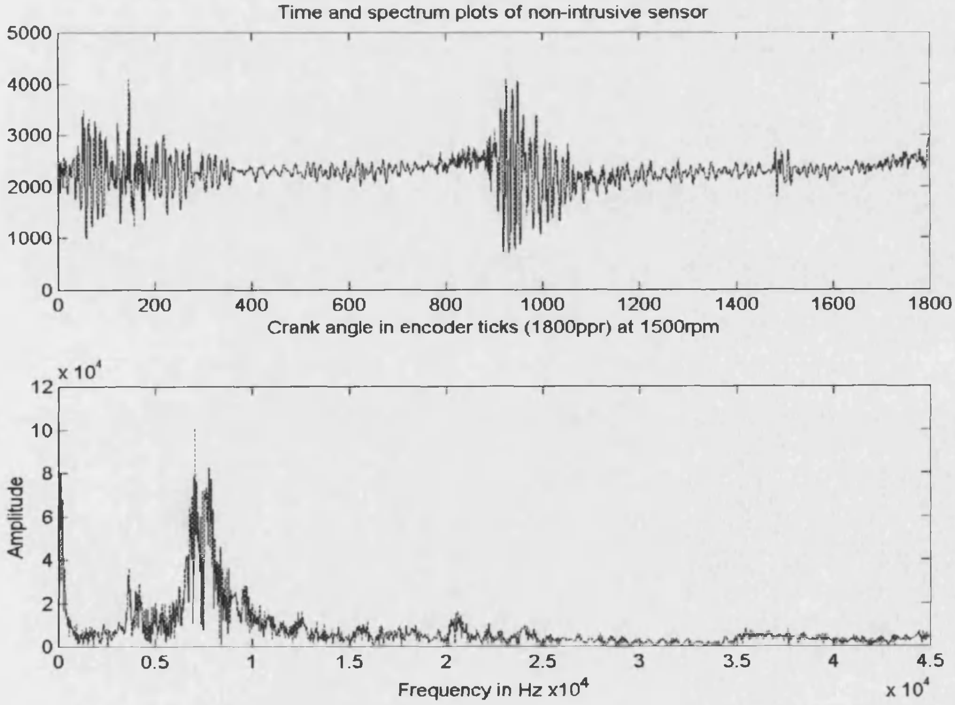


Figure 8: Fourier transform of non-intrusive cylinder pressure data

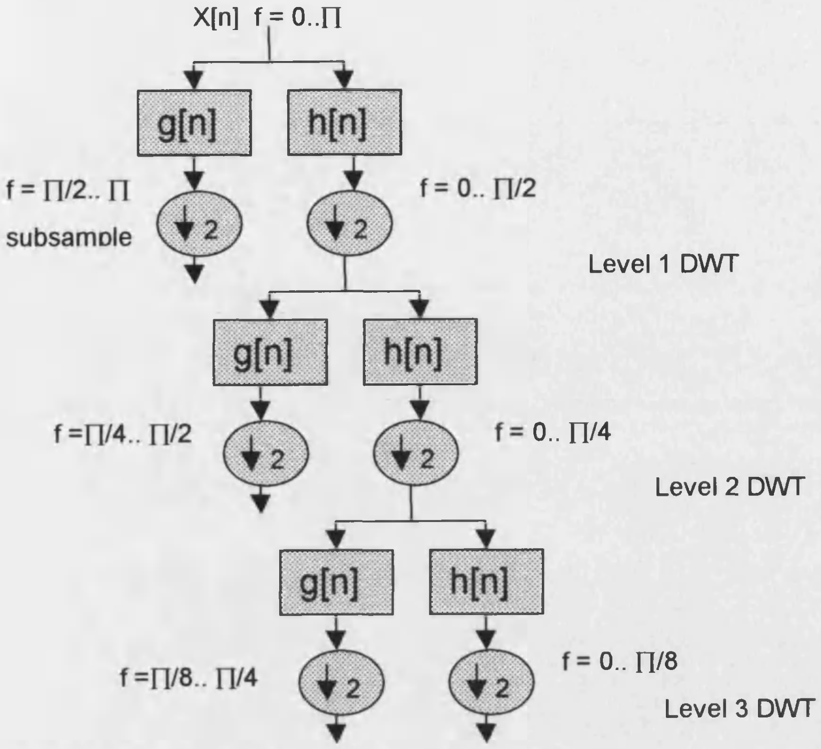


Figure 9: DWT Decomposition Tree

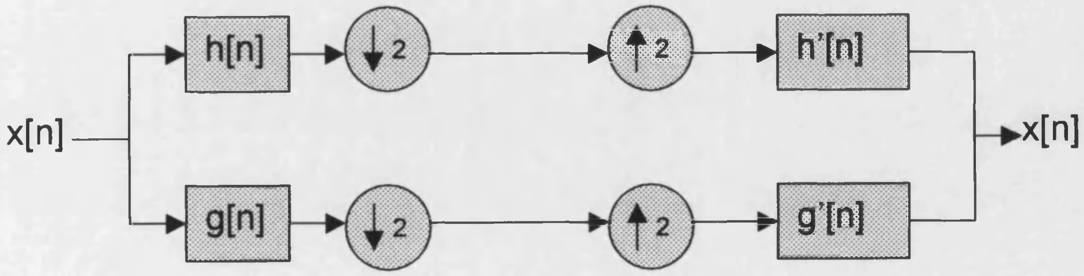


Figure 10: Arrangement of a Quadrature Mirror Filter

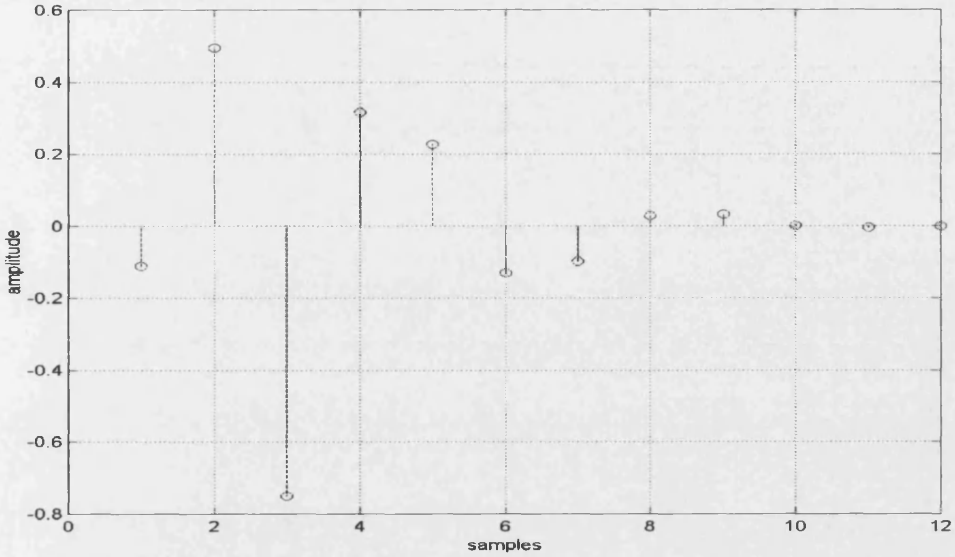


Figure 11: Impulse response of db6 wavelet

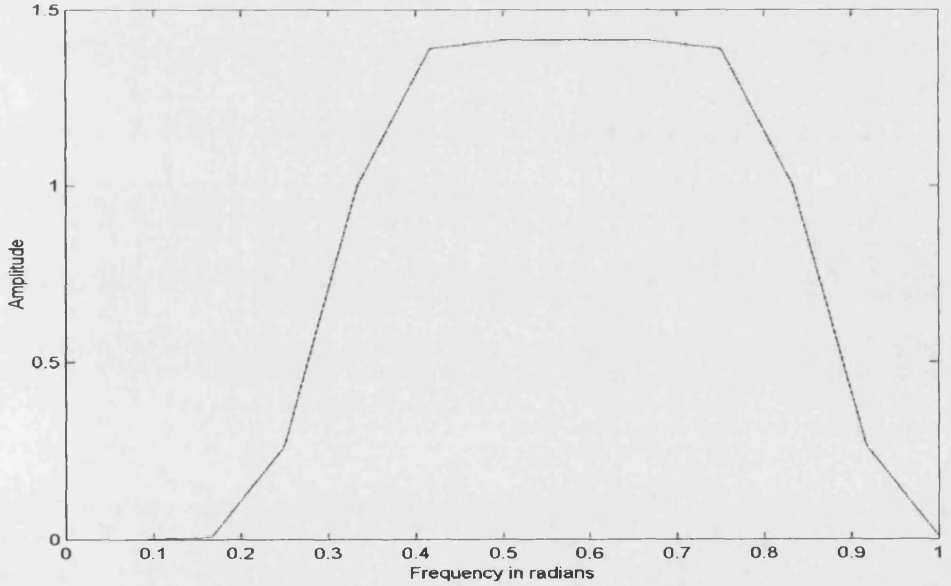


Figure 12: Frequency response of db6 wavelet

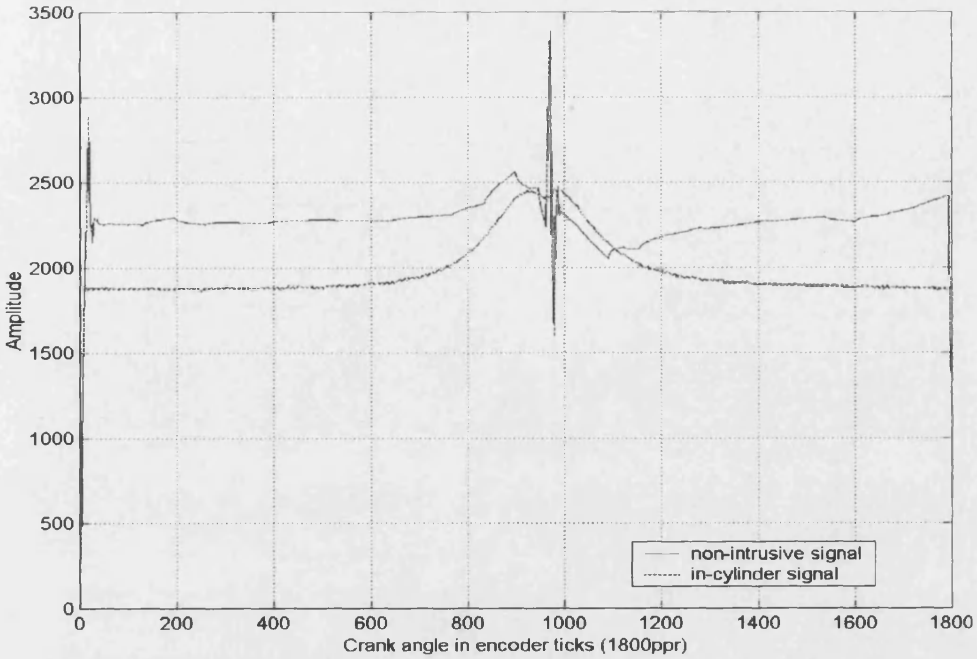


Figure 13: De-noised non-intrusive pressure transducer data

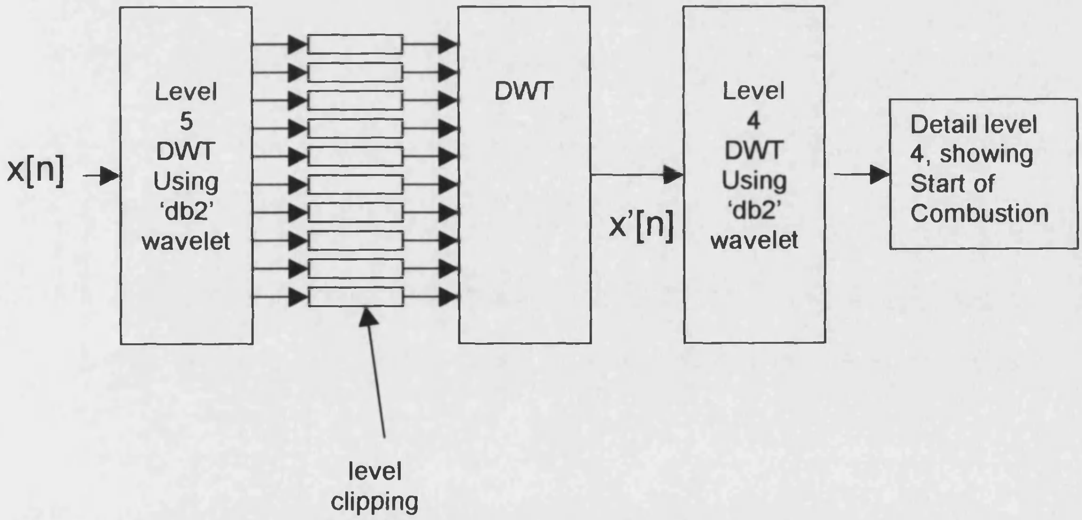

x[n]
Level 5 DWT Using 'db2' wavelet
DWT
x'[n]
Level 4 DWT Using 'db2' wavelet
Detail level 4, showing Start of Combustion
level clipping

Figure 14: Arrangement of Signal Processor