Generic Embedded Sensor Development

High-Speed Analogue Preconditioning and Data Acquisition Systems

By
Francis Leonard B. Sc.

A THESIS PRESENTED FOR THE AWARD OF
MASTER OF ENGINEERING

Research Supervisor
Mr. Jim Dowling

School of Electronic Engineering

August 2006

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Master of Engineering is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: Grace Lawrence ID No.: S1183315

Date: 14.9.06

Abstract

High-Speed Analogue Preconditioning
and Data Acquisition Systems
Francis Leonard

The design of a high-speed, low-cost, generic analogue preconditioning system, which is capable of interfacing with a wide variety of transducers, is described. A multi-purpose, configurable data acquisition module, which is also capable of data generation, is presented. Software support is provided by advanced signal analysis and innovative presentation algorithms that are implemented on a powerful embedded digital signal processor.

Precision voltage-feedback and high-speed current-feedback operational amplifiers are combined to form adaptable, front-end multi-stage composite amplifiers. These distinctive composite amplifiers retain the individual qualities of their incorporated parts. The amplifier's output voltage range is designed to match a standard ADC input voltage range. Easy to implement data converter circuitry and front-end signal amplification components were implemented on a detachable printed circuit daughter-board. A resourceful method of interfacing allows a series of Texas Instruments DSP mother-boards to be accessible. The PCB features single and dual input channel operation, adjustable input voltage ranges and sampling rates under software control. Bi-directional capability adds diagnostic functionality to the already versatile high-performance data acquisition system.

The high-speed preconditioning and data converter element's vulnerability to inconsistent and non-ideal real world effects such as noise, signal interference, parasitics, proximity effects and other layout problems is discussed.

Real-time digital filtering, spectrum and phase analysis sub-routines form code-intensive multi-functional programmes that simultaneously display visually and record numerically the input signal properties in a comprehensive fashion. A user directed, automated, calibration software programme was developed to correct for precision-based errors caused by the signal conditioning instrumentation.

Acknowledgements

The author would like to thank Mr. Jim Dowling, project supervisor, for his guidance, constructive criticism and for giving me the opportunity to attempt this project in the first place.

Sincere thanks to all the PEI Technologies personnel, past and present, with special thanks to Liam Sweeney for his considerable experience with analogue systems, his ideas, comments and valuable assistance throughout.

Many thanks to the entire Electronic Engineering Faculty and staff, especially to the laboratory technician, Connor Maguire for always having a calm and efficient nature even in a crisis.

I am eternally grateful to my parents, Greg and Elsa, for their strength and constant understanding.

To my soul mate, Sandra, words cannot express the feelings of gratitude I have in my heart for you. Your boundless support and encouragement made the difference.

Thanks are also due to the rest of my family and friends. Thank you one and all.

Finally, I would like to thank God and the Holy Spirit for always answering my prayers and having the power to enlighten my mind.

Contents

List of Tables

List of Figures

List of Abbreviations

3G	Third Generation Wireless Service
4PDT	Quadruple-Pole, Double-Throw
AC	Alternating Current
AC/DC	Alternating Current/Direct Current
ACE	Asynchronous Chip Enable
A-D	Analogue to Digital
ADC	Analogue to Digital Converter
AD(I)	Analog Devices (Incorporated)
AFE	Analogue Front-End
AIN	Analogue Input
AINB	Analogue Input Bar
ANSI	American National Standards Institute
ARE	Asynchronous Read Enable
ARM	Advanced RISC Machine
ASIP	Application Specific Instruction-set Processor
A_v	Closed-loop gain
AWE	Asynchronous Write Enable
BW	Bandwidth
Cc/2	Common-mode Capacitance
CCS	Code Composer Studio
Cd	Differential Capacitance
CE	Computational Engine
C_{ext}	External (parasitic) Capacitance
CFA	Current Feedback Amplifier
CMOS	Complementary Metal Oxide Semiconductor/Silicon
CMRR	Common Mode Rejection Ratio
CPU	Central Processing Unit
CT	Continuous Time
D-A	Digital to Analogue
DAC	Digital to Analogue Converter
dB	Decibel

DC	Direct Current
DCU	Dublin City University
DF	Discrete Frequency
DIF	Decimal In Frequency
DIL	Dual In-Line
DIP	Dual In-Line Pins/Package
DPDT	Double-Pole, Double-Throw
DSK	Digital Signal Processor Starter Kit
DSP	Digital Signal Processor
DT	Discrete Time
ECLKOUT	EMIF Clock Output
EDMA	Enhanced Direct Memory Access
EEPROM	Electrically Erasable Programmable Read-Only Memory
EI	Enterprise Ireland
EMIF	External Memory Interface
F_A	Alias Frequency
FET	Field Effect Transistor
FF	Folding Frequency
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
F_s	Sampling Frequency
FS	Full Scale
GHz	Gigahertz
GPS	Global Positioning System
HF	High Frequency
HPI	Host Port Interface
HS	High-Speed
I^2C	2-wire Inter-Integrated Circuit
I_B	Bias Current
IC	Integrated Circuit
ICD	In-Circuit Debugger
IDE	Integrated Development Environment
IEEE	Institute of Electrical & Electronic Engineers
IIR	Infinite Impulse Response

I/O	Input / Output
ISS-bus	Integrated Smart Sensor bus
kHz	kilohertz
LCD	Liquid Crystal Display
LCR	Inductance Capacitance Resistance (based circuit)
LED	Light Emitting Diode
LSB	Least Significant Bit
MAC	Multiply and Accumulate
McBSPs	Multi-channel Buffered Serial Ports
MCM	Multi Chip Module
MCU (μC)	Micro Controller Unit
MEMS	Micro Electro-Mechanical Systems
MFLOPS	Million Floating-Point Operations Per Second
MHz	Megahertz
MIPS	Million Instructions Per Second
MPLAB	Microchip Programmable Laboratory
MSB	Most Significant Bit
MSPS	Million Samples Per Second
MTYPE	Memory Type
MWPS	Million Words (32-bit word) Per Second
NC	Normally Closed
NCAP	Network Capable Application Processor
NV	Non-Volatile
PaRAM	Parameter Random Access Memory
PC	Personal Computer
PCB	Printed Circuit Board
p.d.	potential divider
PEI	Power Electronics Ireland
PGA	Programmable Gain Amplifier
PIC	Programmable Integrated Circuit
PLCC	Plastic Leaded Chip Carriers
PLL	Phase-Locked Loop
PQFP	Plastic Quad Flat Package
PWM	Pulse Width Modulator

QDMA	Quick Direct Memory Access
QFP	Quad Flat Package
Q.N	Number Quantised into N bits
RAM	Random Access Memory
RF	Radio Frequency
RISC	Reduced Instruction Set Computing
ROM	Read Only Memory
R_{on}	On-Resistance
RS	Register Select
RS-232	Recommended Standard-232
RTC	Resistance Temperature Coefficient
R_w	Wiper Resistance
SBSRAM	Synchronous Burst Static Random Access Memory
SDRAM	Synchronous Dynamic Random Access Memory
SIL	Single In-Line
SMD	Surface Mount Device
SMPS	Switched Mode Power Supply
SNR	Signal to Noise Ratio
SO	Small Outline
SOIC	Small Outline Integrated Circuit
SOP	Small Outline Package
SOS	Second Order Section
SPDT	Single-Pole, Double-Throw
SPI	Series Peripheral Interface
SPICE	Simulation Program with Integrated Circuit Emphasis
SPST	Single-Pole, Single-Throw
SR	Slew Rate
STIM	Smart Transducer Interface Module
TA	Turn-Around
TC	Temperature Coefficient
TEDS	Transducer Electronic Data Sheets
T/H	Track-and-Hold
TI	Texas Instruments
TPDT	Triple-Pole, Double-Throw

T_s	Sampling interval
TTI	Thurlby Thandar Instruments
UART	Universal Asynchronous Receiver / Transmitter
USIC	Universal Sensor Interface Chip
USIM	Universal Sensor Interface Module
VFA	Voltage Feedback Amplifier
VLF	Very Low Frequency
VLIW	Very Long Instruction Word
VLSI	Very Large-Scale Integration
V_{os}	Offset Voltage
VTC	Voltage Transfer Curve
X_s	Sensor Impedance
ZIF	Zero Insertion Force

1

Introduction

A human body's nervous system is exceptionally versatile. A fully functional nervous system is capable of managing five dissimilar sensory data types simultaneously; and with bi-directional capability the nervous system receives and transmits data between sensory organs and the brain, where the data is processed. The characteristics of an electronic based signal conditioning element might be compared to those of the nervous system in terms of adaptability and multi-functionality.

Nowadays a software-supported microprocessor resides at the nerve centre of any high-performance data acquisition system. When programmed appropriately, a powerful microprocessor can deal with complex signalling problems almost instantaneously. The principal software algorithms analyse the sensor data of interest once any unwanted signals or noise, which could possibly become significant during analogue signal conditioning and susceptible interfacing stages, have been extracted.

Even with the vast array of commercially available hardware, it is rarely possible to meet system performance and cost goals with exclusively off-the-shelf products. Thus, in the context of this work, the design of alternative adaptable signal conditioning elements to be compatible with a series of powerful processors was addressed. To increase the resultant (high-speed) data acquisition system's appeal, the developed sensor interface is flexible enough to be used with a wide variety of transducers. Advanced software programmes with real-time signal analysis, signal presentation and diagnostic functionality support the interface, instrumentation, and embedded digital signal processing.

Furthermore, as products and systems grow in complexity, there is an increasing need for embedded sensors, creating so-called "Smart Products" – products that interact intelligently with their users and with their environments. Indeed, embedded sensors are already an essential means of ensuring reliability and safety in many applications. However, cost again remains a significant barrier, in this case, to the adaptation of embedded sensors within many products. From automotive control to environmental monitoring systems, transducers with different characteristics are involved, each requiring different circuitry for signal conditioning. Thus, in general terms, the project was concerned with the integration of low-cost, easy to use sensors into instrumentation systems designed to handle a range of signals with contrasting characteristics. The key element of the project was the development of a configurable system to interrogate the transducers, condition the resultant data and output the data in a format for transmission.

This work was undertaken as part of a larger research project entitled "Generic embedded sensor development." The project began in January of 2001 and was completed by August of 2004. Three PEI Technologies (formerly known as Power Electronics Ireland) centres collaborated on this project. The centres were the University of Limerick Analogue Centre (ULA), the University of Limerick Thermal Centre (ULT) and the Dublin City University (DCU) centre. The University of Limerick Analogue centre (ULA) was the principal investigator and had overall management responsibility for the project. PEI Technologies was one of several government sponsored national Programmes of Advanced Technology (PAT) that involved the partnership between Enterprise Ireland, industry and the universities [1–1].

The DCU centre was responsible for the selection, design, linearisation, compensation, calibration and implementation of the signal conditioning/processing elements. The specific part of the DCU work discussed in this thesis includes (as defined by the following project objectives): analogue preconditioning, amplification, data conversion, calibration and digital signal processing (DSP). Figure 1.1 illustrates, the systems involved.

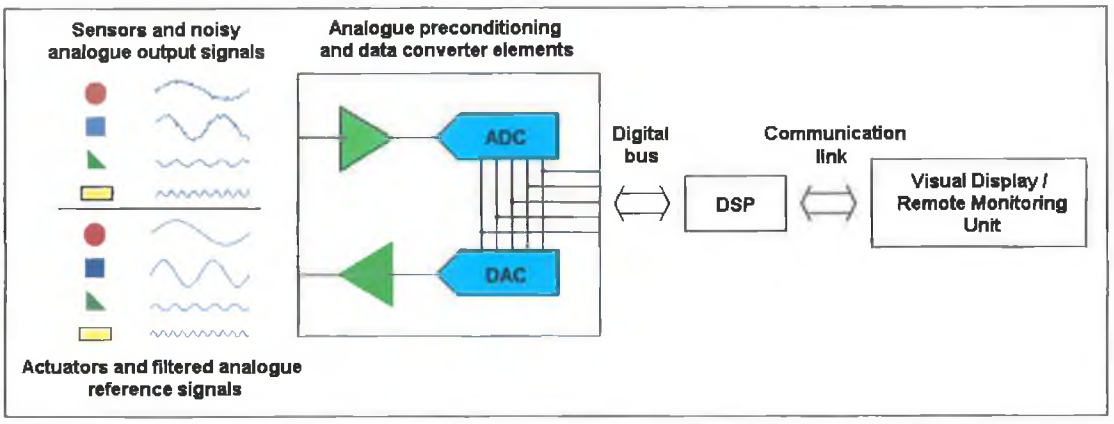


Figure 1.1: Generic data acquisition and data generation signal flow diagram

1.1 Project Objectives

This work at DCU specifically set out to develop a high-performance, low-cost, generic analogue preconditioning system capable of interfacing with a wide variety of transducers. In particular, the goals were to develop a single configurable, high-speed, multi-purpose, data acquisition (and data generation) module and to implement a real-time signal analysis and signal presentation system on an embedded digital signal processor with diagnostic functionality. Methods of compensating for offset and/or sensitivity errors caused by signal conditioning instrumentation were also investigated.

1.2 Summary of Achievements

A standard series of composite analogue preconditioning amplifier designs were implemented. This facilitated a range of sensor output signals and ensured that the resulting data acquisition system has a broad application base potential. The preconditioning amplifiers and data converter modules, which combine to form a high-performance data acquisition system, were separately and collectively evaluated in terms of speed and precision using different temperature conditions during the precision analysis. The data acquisition/generation devices were incorporated onto a printed circuit daughter-board that is designed to interface with a range of powerful

DSP mother-boards. This daughter-board is a useful early prototype for future on chip integration.

Software programmes were written to control the data acquisition system's primary features, which include the level of amplification and the system's sample rate. A set of multi-functional programmes supporting digital filtering functions and advanced signal processing algorithms were developed and tested. A user directed automated calibration software programme was also developed to compensate for any significant system offset and/or sensitivity errors.

1.3 Organisation of the Thesis

A literature review of the recent advances in sensor interfacing and instrumentation is presented in Chapter 2. This survey highlights the amount of academic interest in sensor-based instrumentation, circuit design and signal processing.

If an analogue-to-digital converter (ADC) is treated as part of the signal conditioning circuitry, the instrumentation preceding the ADC (for example an amplifier) is termed a preconditioning element with respect to the ADC. When attempting to create a data acquisition system, this preconditioning element is of crucial importance to the system's overall performance. An entire chapter (Chapter 3) has been devoted to modelling and simulating the analogue amplifier's designs.

Chapter 3 begins by establishing the high-performance standard that an amplifier system must reach before it can be considered for implementation. The composite amplifier design presented provides a method of optimising an amplifier system that combines two or more op-amps, with contrasting specifications, in cascade form. Precision and high-speed op-amps are considered with the support of PSpice simulation software. A cost-effective and compact gain selection system is also presented.

Methods of implementing the particular composite amplifiers are documented in Chapter 4. Specific layout and hardware assembly issues are discussed. The

composite amplifier's dc precision and ac analysis test results are illustrated, predominantly by tabular and graphical means.

Chapter 5 initially deals with the difference amplifier functioning as a signal level shifter. Suitable high-speed ADC and DAC devices are discussed and tested. Calibration techniques, capable of minimising system non-linearity errors, are introduced and illustrated with the use of a flowchart diagram. A selection of the DSP platforms available from Analog Devices (AD) and Texas Instruments (TI) is also reviewed. A successful physical method of interfacing the signal conditioning and signal processing sub-systems is discussed. A printed circuit daughter-board design is described in detail; this incorporates the data acquisition/generation devices and is designed to interface with a range of powerful DSP mother-boards.

In Chapter 6, the advanced signal processing software is discussed in detail. A series of DSP algorithms (digital filtering and FFT), implemented on the TI low-cost TMS320C6711 DSP development platform, is discussed. Information required to optimise the external memory interface (EMIF) and enhanced direct memory access (EDMA) controller are presented, leading to high-speed data transfer rates between the peripheral daughter-board and the DSP's internal memory. Real-time signal analysis functions are also incorporated into high-level C language programmes as important signal properties are displayed within the TI Code Composer Studio (CCS) and/or Matlab applications.

Chapter 7 is devoted to summarising the major points of the thesis. Referenced source code functions and programmes are contained, in full, within the appendices. The daughter-board schematic diagrams and the component list that form the PCB are also contained in the appendices.

2

Literature Review

In an effort to identify and understand the existing research, which is significant for the project while providing a context for the thesis, a literature review was undertaken. As a result of this review, the initial and general direction of the project was established. The topics researched range from transducer interfaces and instrumentation design to data acquisition system processing. In addition to reviewing relevant technical papers, the works of a number of academic institutions have been highlighted within this chapter.

2.1 Research Papers

The reviewed papers are organised such that the commercially available products are outlined first, while the purely academic based research systems complete the descriptive study. The most crucial concepts and issues that are addressed with respect to this project include low-cost, multi-purpose, transducer interface, programmable analogue front-end control, high-speed precision instrumentation designs and embedded support of advanced processing algorithms. There is a degree of repetition of these topics across the papers surveyed, which indicates a relationship between the different research works investigated.

The author(s), bibliography referred number and year in which the paper was published are shown for each paper reviewed. Despite the age of some of these articles, they still form useful background information that is relevant to understanding and investigating further the topics being researched.

2.1.1 Low-Noise Data Acquisition System

A. Bindra [2-1] 1999 - *Ultra-low-noise data acquisition IC tackles multiple sensors.* Author reviews the Cirrus Logic (USA) product in Electronic Design Magazine.

This article describes an on-chip multiplexer, novel amplifier, and analogue to digital converter (ADC) simplified analogue front-end design for precise measurement of low-level signals. Cirrus Logic developed a system-level data acquisition converter family, the CS533x. By integrating all of the functions needed to directly link sensors to microcontrollers or digital systems, the CS533x drastically simplifies the analogue front-end. Above all, the chip delivers the lowest noise level compared to other instrumentation amplifier solutions.

2.1.2 Integrated Sensor Interface

McCartney D., Sherry A., Meany T., Cummins T., Brannick D., MacManus L. [2-2] 1999 - *A fully integrated sensor interface chip.* Authors are from: Analog Devices (Ireland), Accutron Ltd. (Ireland), Trinity College (Ireland).

A single-chip sensor signal conditioning system is described. The chip contains two 20-bit sigma-delta ADCs. The main ADC has a rail-to-rail input buffer and Programmable Gain Amplifier (PGA). There is a 12-bit digital to analogue converter (DAC) and current source for sensor excitation and circuitry for sensor fault detection. An 8052 microcontroller core is provided for processing the ADC outputs. There are 640-bytes of user flash memory and 8-kbytes of programme flash. Clocking is from a 32kHz watch crystal oscillator. An on-chip Phase-Locked Loop (PLL) multiplies up this frequency to provide programmable clocks for the microcontroller. The die area is $17mm^2$ on a 0.5μm Triple-Poly, Triple-Metal (TPTM) flash memory process. Power consumption is 10mW at 3V.

The authors propose that, while previous solutions [2-3] have addressed the analogue signal conditioning and ADC components of a sensor system, a need exists to integrate the digital signal processing (DSP) also. Moreover an emerging standard, IEEE 1451.2 [2-4], has partitioned the sensor interfacing system into two aspects: the measurement aspect is dealt with by the Smart Transducer Interface Module (STIM) while the application aspects are handled by the Network Capable Application Processor (NCAP). Clearly there is a need for a fully integrated sensor interface chip; such a chip is described in this paper.

2.1.3 Standard Transducer Interface

Cummins T., Byrne E., Brannick D., Dempsey D.A. [2-5] 1998 - *An IEEE 1451 standard transducer interface chip with 12-bit ADC, two 12-bit DACs, 10-kbyte flash EEPROM and 8-bit microcontroller.* Authors are from: Analog Devices.

A single-chip implementation of an IEEE 1451 Standard Transducer Interface Module (STIM) is presented. It integrates an eight-channel, 12-bit ADC, two 12-bit DACs, and an 8-bit microcontroller with 256-byte SRAM and 10.5-kbyte flash EEPROM. Process complexity is simplified by using a split-gate flash EEPROM cell and metal-poly capacitors with a calibration algorithm in the ADC. The chip is 5.0 x 5.0mm in 0.6μm CMOS, operates over 2.7-5.5V, draws 13mA at 3V/12MHz, and is packaged in a 52-pin plastic quad flat pack package.

The recently approved IEEE 1451 standard aims to standardise the interface and communication protocol between networked host control systems and various types of sensors and transducers. The main elements of IEEE 1451 are a network-capable application processor (NCAP), a standard transducer interface module (STIM), and a ten-way transducer independent interface containing serial data and control signals. The NCAP typically contains a network transceiver and a processor implementing the network communication protocol. The STIM portion contains the sensor interface electronics, signal conditioning and conversion, calibration, linearisation, basic communication capability, and a non-volatile (NV) 565-byte transducer electronic data sheet (TEDS).

While this chip can be used as a general-purpose programmable converter, its combination of features makes it specifically suitable for the STIM implementation. A 12-bit ADC and DAC enable signal conditioning and linearisation for gas sensors, battery monitors, and other applications requiring high-resolution conversion. The 640-byte NV data memory provides the TEDS storage. Since the UART serial port will usually be dedicated to programme-code downloading, the second serial port provides the 1451.2 serial data and control pins.

2.1.4 Microsensor Interface

Malcovati P., Baltes H., Maloberti F. [2-6] 1996 – *Progress in microsensor interfaces.* Authors are from: Physical Electronics Laboratory (Switzerland) and Università di Pavia (Italy).

This is a general review of microsensor interfacing, it gives an overview of the integration of signal processing on a single chip and the architecture used. It also investigates in more detail:

- voltage sensing
  - thermoelectronic infrared sensor
  - thermal converter
- current sensing
  - magnetic sensors
  - UV diodes
- resistive and capacitive sensing
  - thermal pressure sensor
- sensor calibration
- and muti-sensor interfaces

2.1.5 Sensor Interface Microinstruments

Kraver K.L., Guthaus M.R., Strong T.D., Bird P.L., Sig Cha G., Höld W., Brown R.B., [2-7] 2001 - *A mixed-signal sensor interface microinstrument.* Authors are from: University of Michigan (USA), Kwangwoon University (South Korea), National Semiconductor Corp. (Germany).

The paper describes a single chip implementation of a microinstrument system (called MS-8). The chip incorporates voltage, current and capacitive sensor interfaces, a temperature sensor, programmable signal amplification, a ten channel 12-bit ADC, an 8-bit microcontroller with 16-bit hardware multiplier and 40-bit accumulator. Serial and parallel interfaces allow digital communication with a host system. The die is fabricated in a standard 0.35µm digital CMOS process, occupies 3.8mm x 4.1mm, and requires a 3V, 16mA supply.

The chip is the first implementation of an ongoing project to develop a low power, low-cost, multi-purpose sensor interface and data acquisition system. This single chip microinstrument contains a programmable analogue front end capable of interfacing to a variety of sensors. The integrated microcontroller supports the digital filtering

and compensation of sensor outputs, the timing control for sampling multiple sensors, and the communication with a host system.

The MS-8 is optimised, from instruction set definition to analogue functionality, to economically support embedded sensor applications. A minimal version of the microinstrument can be implemented with as few as eight pins. The instruction set provides excellent code density and supports the ANSI C programming language. To aid code debugging, the MS-8 includes hardware support for a single breakpoint and trace event and a development system interface provides instructions for halting the processor, single stepping through code and reading/writing system registers. In addition, a clock manager provides programmability of the clock frequency for the digital and analogue circuits and each analogue block can be individually powered down under software control.

2.1.6 Generic Interface Chip

Yazdi N., Mason A., Najafi K., Wise K.D. [2-8] 2000 – *A generic interface chip for capacitive sensors in low-power multi-parameter microsystems*. Authors are from: University of Michigan (USA).

The paper presents a generic low-power sensor interface chip compatible with smart microsystems and a wide range of capacitive transducers. The interface chip is highly programmable, can communicate with an external microcontroller using a nine-line sensor bus standard, contains a switched-capacitor readout circuit, supports sensor self-test and includes a temperature sensor. The circuit can interface with up to six external sensors and contains three internal programmable reference capacitors in the range of 0.15-8pF. The chip measures 3.2 x 3.2mm, dissipates less than 2.2mW from a single 5V supply, and can resolve input capacitance variations of less than 1fF in 10MHz bandwidth.

The research programmes at the University of Michigan's Centre for Integrated Microsystems are pioneering new advances in many relevant aspects of microsystems. The sensor bus is a central part of the microsystem architecture that

provides access to multiple sensors while using a limited number of interconnects. The heart of the microsystem is a microcontroller unit (MCU) that provides stored programme control and data handling as well as sensor-specific software routines for in-module sensor calibration, digital compensation and self-test. Communication with the MCU is performed over the sensor bus and is handled by the bus interface unit. Serial data transmitted over the bus is received, decoded, and applied to control various interface chip blocks.

The sensor bus interface unit handles the communication between the microsystem controller and the capacitive interface chip through the sensor bus. The bus interface unit consists of a series of shift registers that load the input serial data, logic circuitry that decodes incoming instructions, on-chip memory to store data written to the interface chip and an output multiplexer. A self-test DAC is another important circuit building block. The DAC output voltage is typically buffered and is available at one of the chip output pads. The DAC output also modulates the amplitude of the clocks that drive the sense and reference capacitors. This chip therefore satisfies all the requirements of a capacitive type microsystem: it can interface with a large variety of capacitive sensors with base-capacitance and sensitivity spread over a wide range and support communication with any microcontroller over a standard sensor bus. In addition it has programmable gain and offset control, supports sensor self-test and occupies very small die area with no external components.

A number of these generic capacitive interface circuits are currently employed in a low-power wireless muti-element microsystem for environmental monitoring. While this wireless battery-powered system provides a prototype for future small portable microsystems, the presented generic interface chip and its combined features will be essential for the successful realisation of similar microsystems of the future.

2.1.7 Future Sensor Interface Electronics

Yamasaki H. [2-9] 1996 – *The future of sensor interface electronics*. Author is from: Yokogawa Research Institute Corporation (Japan).

"The Future of Sensor Interface Electronics" is about future technologies on image processing. The functions of interface electronics are expanding in several directions:

- extension towards multidimensional sensing systems
- extension towards multilayer signal processing systems
- extension towards multimodal sensing systems
- progress towards more precise sensing systems
- progress towards more reliable sensing systems
- progress towards more human-friendly sensing systems
- progress towards more compact sensing systems

2.1.8 On-Chip Sensor Electronics

Najafi K. [2-10] 1987 – *Sensor-system interface: The influence of on-chip electronics.* Author is from: University of Michigan (USA).

This paper reviews the influence of on-chip signal conditioning circuitry on the performance of solid-state integrated sensors. The functions and characteristics of signal conditioning circuitry, which interfaces directly with the sensor, are first reviewed. Then advantages and disadvantages of integrating the circuitry on the sensor substrate are discussed. Performance, cost and yield considerations are analysed and reviewed. Finally two silicon-based sensors, a multielectrode recording probe and a capacitive tactile imaging array are compared with regard to the above considerations.

2.1.9 Microsensors and Packaging

Baltes H., Brand O. [2-11] 2001 - *CMOS-based microsensors and packaging.* Authors are from: Physical Electronics Laboratory (Switzerland).

This paper reviews a post-CMOS approach, technology for use in CMOS microsensors and presents three CMOS-based microsensors developed at the Physical Electronics Laboratory (PEL), these being:

- CMOS thermal imager for presence detection,
- CMOS chemical microsystem for detection of volatile organic compounds in air,
- CMOS temperature and stress microsensors for investigation of wire bonding processes.

Regarding the second of these microsensors, CMOS-based chemical microsensors, using spray-coated polymer films as chemically sensitive layers, are developed for the detection of volatile organic compounds in air. The CMOS chemical microsystems are packaged using special flip-chip (FC) bonding or chip-on-board [2-12,-13] technology. The sensor die is flip-chip mounted onto a chemically inert ceramic substrate, where the microsystems can strongly benefit from these established IC packaging techniques.

2.1.10 Digital Bus Interface

Correia J.H., Cretu E., Bartek M., Wolffenbuttel R.F. [2-14] 1997 – *A low-power low-voltage digital bus interface for MCM-based microsystems*. Authors are from: Delft University of Technology (The Netherlands).

The paper describes a digital local bus interface that is designed for use in a multi-chip-composed microsystem. The chip area using a CMOS 1.6μm n-well technology is $1mm^2$. Power consumption for 5V at 100kHz is less than 500μW and for 5V at 4MHz less than 2mW due to a smart power management of all functional blocks. The bus interface is able to transmit: voltage, frequency, duty-cycle signal data and also provides calibration facilities, service request and interrupt request for the smart sensors or microactuators.

The bus interface should be versatile enough to ensure efficient communication between all sensors and systems on the platform, but simple enough to be on-chip merged within the platform. As an additional feature the bus interface should be able to handle both digital and semi-digital signals, such as pulse width and frequency modulated pulse series. Moreover, self-test should be implemented over the bus by using analogue excitation signals and simultaneous semi-digital or digital readout.

Available bus protocols lack the flexibility that is needed to deal with a multi-sensor system on the die level. This paper presents an upgraded version of the basic Integrated Smart Sensor bus (ISS-bus) [2-15]. This upgrade is based on a single controller to co-ordinate the activity on the bus and includes: a maskable interrupt mechanism, calibration facilities, small size, low-power consumption that makes it very suitable for implementation on microsystems.

2.1.11 FIR Digital Filtering

Ramanathan S., Visvanathan V., Nandy S.K. [2-16] 1999 – *A computational engine for mutirate FIR digital filtering*. Authors are from: Indian Institute of Science (India).

This paper proposes a computational engine (CE) to serve as an ASIP (Application Specific Instruction-set Processor) implementing compute-intensive/power-critical multirate finite impulse response (FIR) digital filtering algorithms in embedded systems. The CE is programmable and consists of a data path and a control path. Control sequences necessary to realise a particular filter operation can be programmed onto the control path of the CE. The versatility and efficacy of the proposed CE are demonstrated in this paper.

2.1.12 All-Digital Front-End Sensors

Ben Romdhane M. S., Madisetti V. K. [2-17] 1996 – *All-digital oversampled front-end sensors*. Authors are from: Georgia Institute of Technology (USA).

This paper proposes the use of all-digital oversampling front-end sensors, resulting in a performance/cost-effective DSP application. DSP circuits usually use analogue front-end anti-aliasing filters to process sensor inputs. These analogue filters have a number of disadvantages with respect to cost, power, stability, and ease of integration in VLSI. Hence the proposed all-digital front-end version could theoretically overcome these disadvantages once implemented.

2.2 Research Institutions

The papers reviewed are collected from a variety of global academic sources, however particular recognition needs to be given to the three research institutions that are profiled here below. These research institutions have advanced sensor instrumentation and its associated technologies to the point where the resultant systems are having a pervasive impact on the future of the microelectronics industry in application fields ranging from automotive systems to environmental monitoring.

2.2.1 Physical Electronics Laboratory

Physical Electronics Laboratory (PEL), Zurich, Switzerland. The laboratory is led by Prof. H. Baltes, Dr. O. Brand, Dr. A. Hierlemann and Ch. Hagleitner.

Research is targeted to silicon-integrated micro and nano systems in collaboration with silicon IC manufacturers and system users. It includes device physics and chemistry, post-CMOS fabrication, packaging, signal conditioning IC and simulation tools, as needed for the demonstration of novel integrated micro and nano systems.

2.2.2 University of Michigan

University of Michigan, Department of Electrical Engineering and Computer Science, USA. Professors of interest: Khalil Najafi, Richard B. Brown and Kensall D. Wise.

The Center for Integrated Microsystems within the Department of Electrical Engineering is focused on the intersection of three key areas: microelectronics, wireless communications, and microelectromechanical systems (MEMS). The resulting integrated microsystems will soon provide button-sized information-gathering nodes for distributing sensing applications ranging from environmental monitoring to healthcare.

The Solid-State Electronics Laboratory also within the Department of Electrical Engineering is at the forefront of research in microelectronics and optoelectronics with major thrusts in micromachined integrated sensors and actuators, automated semiconductor manufacturing, compound semiconductor materials, ultra-high-speed microwave and millimeter-wave devices, optoelectronic devices and monolithic analogue and digital integrated circuits for different applications.

2.2.3 Delft University of Technology

Delft University of Technology, Department of Electrical Engineering, The Netherlands. The industrial research programme is led by Prof. Joachim N. Burghartz, Dr. Marian Bartek, Prof. Johan H. Huijsing and Dr. R.F. Wolffenbuttel.

The research within DIMES is spread over four main themes: high-frequency technology for communications, integrated smart microsystems, nano-electronics and large area electronics

2.3 Summary and Project Direction

Research in the area of high-performance data acquisition systems has followed several avenues. Early work by Najafi [2-10] was concerned with the influence of on-chip low-noise signal conditioning circuitry on the performance of solid-state integrated sensors, while more recent work [2-11] considered the importance of sensor IC package techniques. Methods of standardising transducer interfaces [2–5,–6,–8] and the integration of sensor signal conditioning and signal processing component designs were also addressed [2-1,-2,-7,-9] in a number of related papers. Several researchers [2-16,-17] investigated the problem and the development of software support and appropriate compensation routines. Further diagnostic facilities [2-14] were evolved for testing digital bus functionality. DSP architecture [2-16], ideal for high-speed applications, was also discussed.

From the state-of-the-art instrumentation addressed in these articles the project's initial design aims and targets were formed: to develop a high-performance, low-cost, generic, analogue preconditioning system capable of interfacing with a wide variety of transducers. This front-end analogue signal conditioning system could be combined with compatible high-speed bi-directional data converter instrumentation to form a multi-purpose data acquisition (and data generation) module with diagnostic functionality. A real-time embedded digital signal processing device should provide communication between a host PC and the instrumentation.

The amplifier circuit design requires a certain amount of innovation to establish a high-precision and wide-bandwidth preconditioning system. At present, separate op-amp devices are manufactured for each distinct application. Therefore a reliable method of incorporating two or more op-amps, with contrasting technologies, is essential for maintaining the systems overall desired performance. Specific precision and frequency range targets are outlined in Chapter 3. Extensive software simulating and practical testing is required to confirm any design's performance. Also real-world problems such as signal interference and parasitic effects need to be resolved prior to assembling a final prototype low-noise PCB. To implement a desirable cost effective system, only competitively priced parts with high-performance properties should be selected. A further restriction on the IC dimensions and package types could ensure that a designed prototype system with soldered parts is easily reproduced. A compatible low-cost DSP device needs to be selected that should add software support to any implemented instrumentation. On a suitable processing platform, programmes for analysing a range of signal properties could be developed.

3

Analogue Amplifier Designs

In any data acquisition system a sensor's output signal needs to be measured, recorded, manipulated mathematically or simply observed. These operations could possibly be performed in either the analogue or the digital domain. Typically, the more complex operations are accomplished in the digital domain where operations can be programmed and reprogrammed to perform a variety of functions without modifying the hardware. There are situations however where using analogue techniques are simpler, more economical and most importantly more appropriate [3−1]. For example, the process of signal amplification is generally associated with the analogue domain and is accomplished using analogue circuitry.

Amplifiers are essential in order to amplify low-level signals (e.g. ultrasonic, magnetic and RF signals) to a level that enables them to be further processed. The preconditioning devices and circuitry required to amplify, or for that matter attenuate, a sensor's output voltage range in order to complement an ADC's typical input range are discussed in detail. The operational amplifier (op-amp) is an essential component in many analogue preconditioning designs that form a front-end sensor interface of a corresponding data acquisition system.

3.1 Amplifier Standards

Op-amp circuits are used to amplify low-level output signals that are inherently available from sensor or transducer elements. Low, medium and high output voltage sensor ranges are displayed in Table 3.1. The op-amp is a versatile sub-system that can be used in various configurations. The circuit diagrams given within this chapter reflect this fact. A multitude of semiconductor fabrication methods, from bipolar to FET technologies, has resulted in distinct categories of op-amps being manufactured. Precision and high-speed op-amps are two separate and contrasting amplifier classes that are designed specifically to handle their corresponding application type. To facilitate a broad application base, a method of combining the two technologies is necessary.

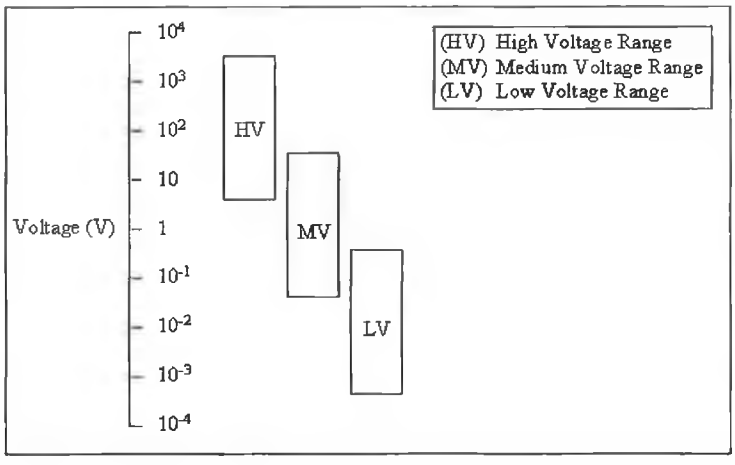


Table 3.1: Sensor output voltage ranges

A composite amplifier is a design option that provides a means of connecting two or more op-amps with contrasting specifications. This option allows for a precision voltage feedback amplifier (VFA) device with low-offset and low-noise properties to be combined with a high-speed current feedback amplifier (CFA) with a wide bandwidth to form an amplifier system with optimised dc and ac characteristics. The composite amplifier, of Figure 3.1, offers important performance advantages. The CFA operates within the feedback loop of the VFA, thus its generally poorer input dc and noise characteristics become insignificant when referred to the input of the composite device. Moreover, if most of the signal swing is provided by the CFA, the

slew-rate requirements of the VFA are significantly relaxed, thus ensuring high bandwidth capabilities for the composite configuration [3-2].

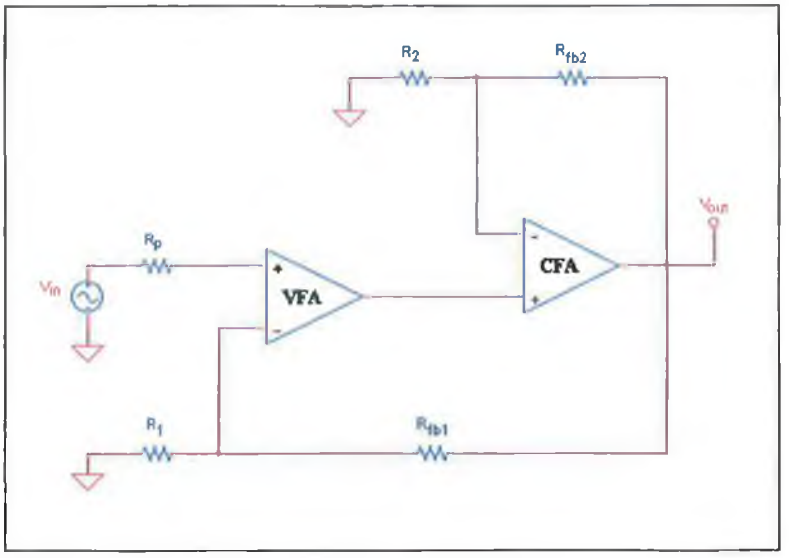


Figure 3.1: VFA-CFA composite amplifier in non-inverting mode

High-performance standards need to be reached by the front-end composite module before the amplifier can be classified as a high-performance device. Specific parameters include:

- wide operating frequency range from dc to 30MHz (high frequency band limit)
- high to low gain/attenuation level (defined in Table 3.2) flexibility
- overall amplifier input offset voltage (V_{OS}) of $\leq \pm 0.1$mV
- typical input bias current (I_B) of $\leq \pm 2$pA
- negligible drift in set gain level over a temperature gradient of 0-40°C

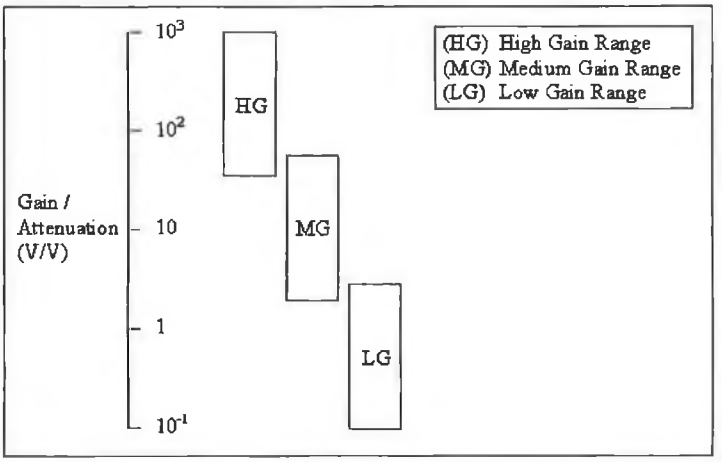


Table 3.2: Amplifier system gain/attenuation ranges

An amplifier module that maintains these parameters can service a wide spectrum of transducers from low-speed thermoelectric to high frequency electromagnetic devices, without signal distortion. A short list of suitable op-amps can be obtained from manufacturers' selection guides and data sheets. Selection guides list devices and their specifications in tabular form thus allowing comparisons to be made quickly and easily. Orcad's Capture and PSpice simulation software packages are recommended for rapid testing of the ac and dc specifications and to confirm that the required high-performance criteria is attainable with the implementation of the selected devices.

3.2 Front-End Precision Devices and Simulations

When amplifying small signals, as is the case for a wide range of sensors, V_{OS} and I_B are of great importance. From the precision op-amp properties listed in Table 3.3, these two features are relatively low which minimises any resulting input errors. Therefore, the initial investigation into finding suitable candidates for the task of a front-end precision interfacing circuit, for a range of sensors, has determined three possibilities. A more detailed analysis process must also be carried out to determine which op-amp will satisfy our specific application requirements.

Device Name	GBW Typ (MHz)	Slew Rate Typ (V/us)	VOS (25 deg C) Max (mV)	IB Max (pA)	Vn at 1kHz Typ (nV/rtHz)	Number of Channels	Approx. 1KU Price (US$)	Vs Min (V)	Vs Max (V)	IQ per channel Max (mA)	Available Channels	Shutdown	Offset Drift Typ (uV/C)	CMRR Min (dB)	Single Supply
OPA637	80	135	0.1	1	3.2	1	9.63	9	36	7.5	8	No	0.4	106	No

(a)

Device Name	BW @ Ad (MHz)	Slew Rate (V/us)	Vos Max (mV)	Ib Max (uA)	Vn at Flatband Typ (nV/rtHz)	Number of Channels	Approx. 1KU Price (US$)	Supply Voltage +/- 5 (V)	Supply Voltage 5 (V)	Ad, min stable gain (V/V)	Settling Time (0.1%) Typ (ns)	Io Typ (mA)	THD (Fc=1 MHz) Typ (dB)	Diff Gain (%)	Diff Phase (deg)
OPA621	500	500	0.5	25	2.3	1	6.95	Yes		2	25	100	-80	0.05	0.05
OPA656	500	290	0.6	0.0000057		1	5.85	Yes	Yes	1	16	70	-74	0.02	0.05

(b)

Table 3.3: (a) General precision and (b) wide-band precision op-amp parametric selection guides from Texas Instruments

The OPA637 op-amp has the lowest maximum V_{OS} and I_B of the listed devices. For both of these features, a factor of at least five exists between this op-amp and the other two devices of Table 3.3. This difference is a considerable margin. The OPA637 op-amp provides a high-performance level even with respect to other precision FET

op-amps. This is an ideal candidate for systems with the single concern of precision. On the other hand, its gain-bandwidth product of 80MHz and slew rate of 135V/μs are relatively limited properties when compared to the alternative devices listed and may be a problem for wide-band, high-speed applications.

The OPA656 op-amp shows a broader range of specifications necessary for both precision and high-speed analogue circuitry. In particular it combines a very wide 500MHz unity-gain bandwidth with a FET-input stage to offer ultra-high dynamic-range capability for ADC buffering applications. Exceptionally low dc errors are retained to give good precision with respect to the more general-purpose op-amp devices. Therefore, this device could successfully implement an amplifier for a larger range of sensor signals that include high frequencies, without the output signal being attenuated or distorted.

The OPA621 features the same bandwidth as the OPA656 yet with an improved slew rate of 500V/μs. This rate permits even faster settling times that are suitable for high-speed applications. Unfortunately, this device's I_B appears to be considerably compromised. This trade-off of precision for greater speed restricts its operational range for pre-amplifying sensor type signals.

Each device in Table 3.3 has a number of individual merits, from extremely low dc errors of precision to wide bandwidths and exceptional high-speed processing. Simulation software was used to attempt to determine the most suitable VFA for implementation within a composite design. The gain and phase properties were observed from simulated frequency response plots, while time-domain offset errors were illustrated for each device.

Figure 3.2 displays the basic single-stage OPA656 in non-inverting form used to analyse its transfer function. This schematic forms a template, which is duplicated for the other two devices examined, as their spectrum and phase responses are simultaneously analysed. Figures 3.3(a) and (b) show the ac analysis results for all three op-amps with the same voltage gain of 2V/V being displayed in both voltage absolute magnitude and decibel (dB) forms respectively. Figures 3.4(a) and (b) give similar plots for a greater voltage gain of 10V/V, while Figure 3.4(c) illustrates the

op-amps corresponding time domain signal response for the gain of 10V/V. This simulation series is completed by the phase response plots for all three precision op-amps given in Figure 3.5.

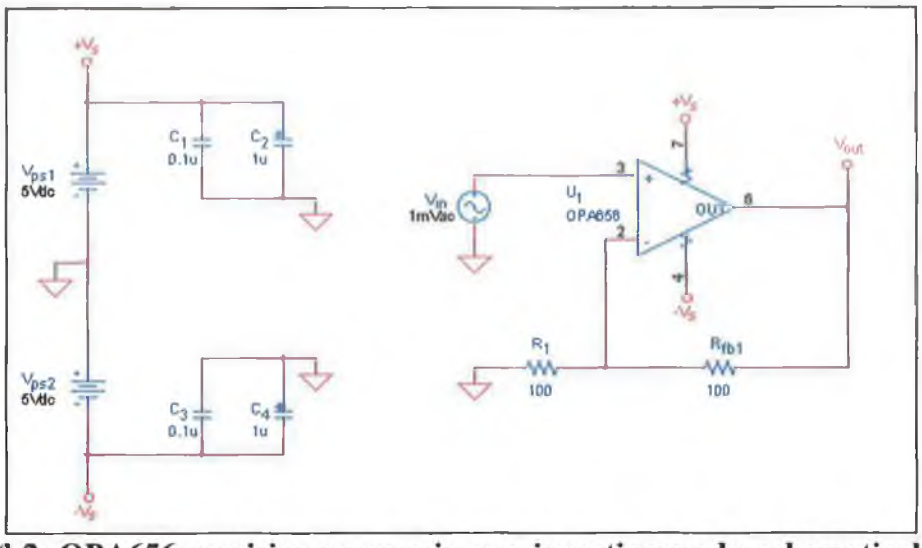


Figure 3.2: OPA656 precision op-amp in non-inverting mode, schematic diagram modelled with Orcad's Capture software

In Figure 3.2 a practical implementation factor (not necessary of PSpice simulations) is included, specifically, supply to ground terminal bypassing capacitors are used. When physically implemented supply bypassing is extremely critical and must always be used, especially when driving high current loads. Both power-supply leads (V_{ps1} and V_{ps2}) are bypassed to ground. The 1μF electrolytic capacitors and the op-amp's data sheet recommended parallel 0.1μF capacitors are added, at each supply pin. Hence, stray signal coupling from the power supplies to the inputs will be minimised when physically implemented.

To retain a controlled frequency response for the non-inverting voltage amplifier, the manufacturer's recommendation is that the parallel combination of $R_1||R_{fb1}$ should always be <200Ω. To achieve a relatively low closed-loop gain of 2V/V, both R_1 and R_{fb1} resistor values are set to 100Ω. The external resistor-gain relationship is obviously:

$$Gain_{ideal} = \frac{V_{out}}{V_{in}} = \frac{R_x + R_{fbx}}{R_x} \qquad (3.1)$$

For a multiple stage composite system, the subscripted x's in Equation 3.1 relate to a specific gain stage. In this single-stage op-amp situation, each x is simply replaced by a 1. The sensor input is represented by an ac voltage source (V_{in}) with an appropriate small signal magnitude of 1mV. The source frequency is varied to produce the responses given in the following simulation graph. The logarithmic frequency scale is sub-divided to allow for more accurate analysis.

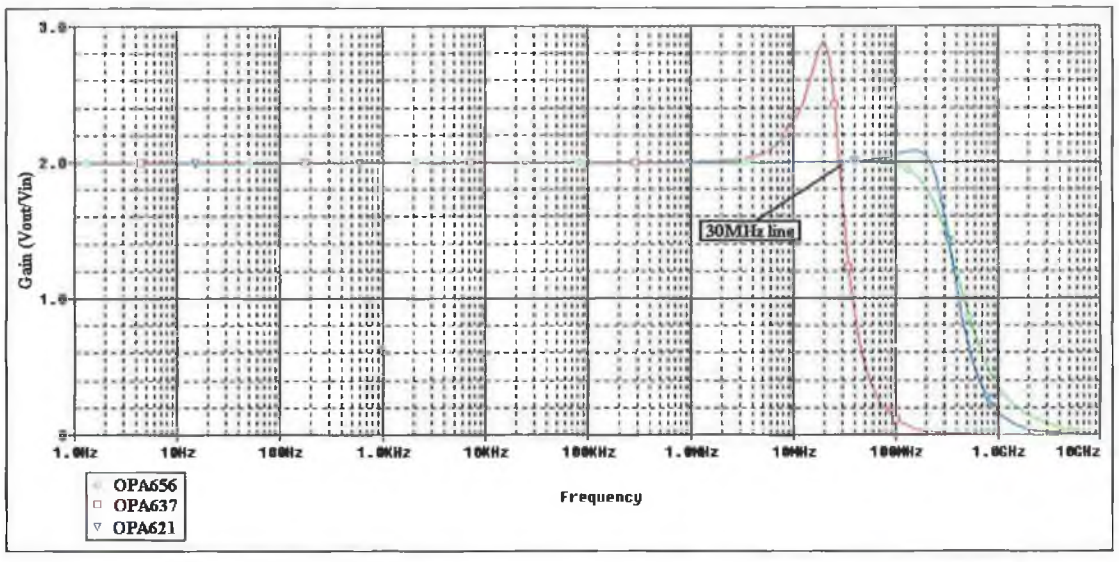


Figure 3.3(a): **AC analysis frequency responses of the OPA656, OPA637 and OPA621 op-amps in non-inverting mode with expected gains of 2V/V. Simulations performed simultaneously with Orcad's PSpice software**

Figure 3.3(a) exhibits the frequency responses for all three single stage op-amps with an expected closed-loop dc gain of 2V/V. The OPA656's response remains constant well above 30MHz and rolls-off gradually. The OPA637 and OPA621 amplifiers both exhibit peaking response characteristics, with the OPA637's peaking occurring at a lower frequency and with a greater magnitude then the OPA621's response. Hence, the OPA656 is established as the only one of the analysed precision amplifiers that appears to function properly through and beyond the high frequency band while maintaining a constant flat gain of 2V/V.

Figure 3.3(b) complements the voltage gain results in the alternative and more practical decibel format. To study significant gain changes and cut-off frequencies (-3dB point), this logarithmic scale becomes increasingly useful. This simulation also

displays the op-amps roll-off slopes as a function of frequency. The graph clearly shows a –40dB/dec slope for all three op-amps examined.

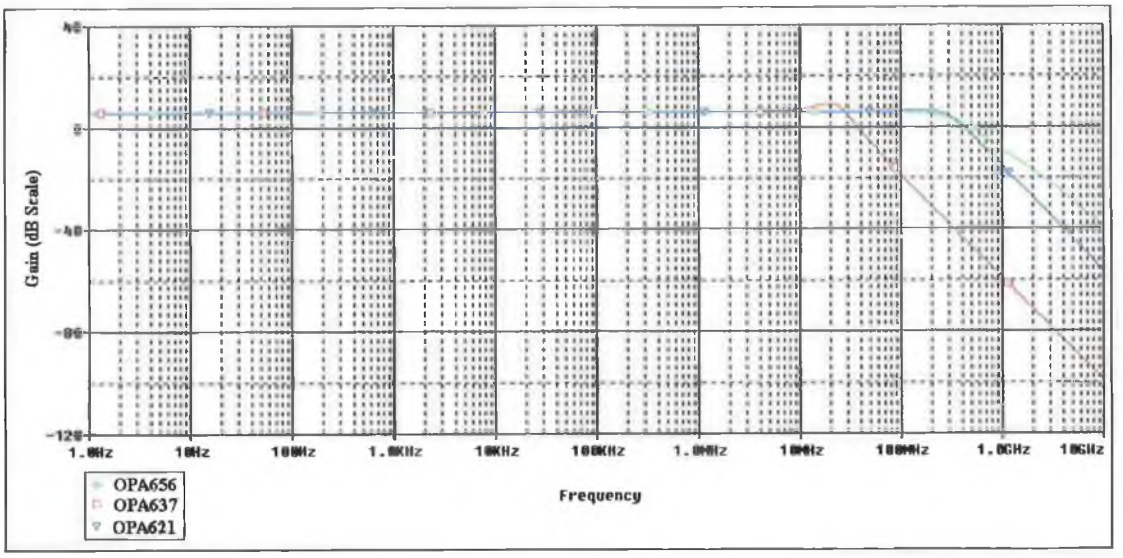


Figure 3.3(b): Bode frequency responses of the OPA656, OPA637 and OPA621 op-amps in non-inverting mode with expected gains of 2V/V (approximately equal to 6dB)

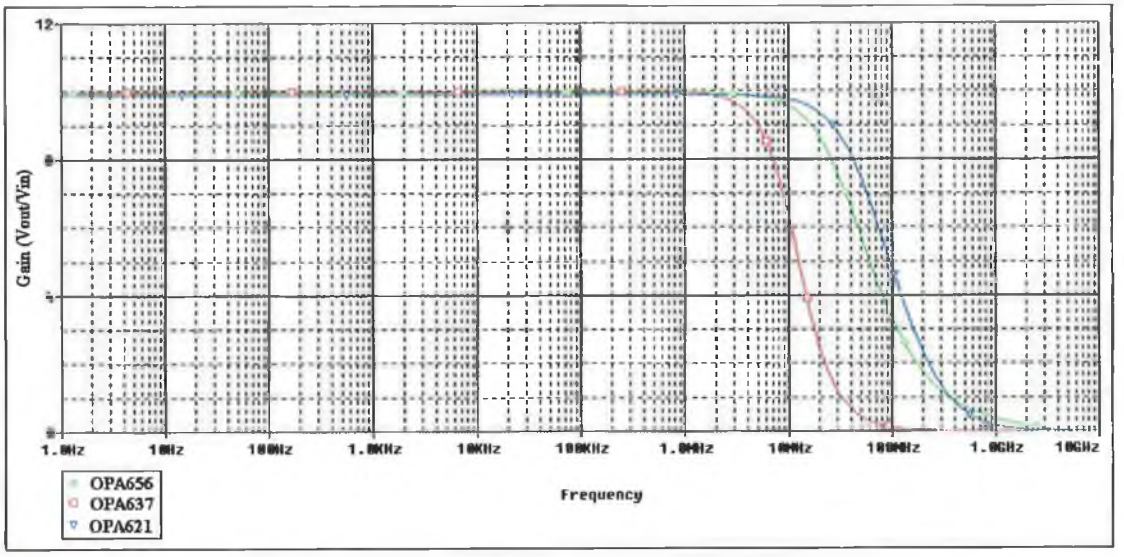


Figure 3.4(a): AC analysis frequency responses of the OPA656, OPA637 and OPA621 op-amps in non-inverting mode with expected gains of 10V/V

To analyse the effects of a higher voltage gain of 10V/V on the precision op-amps, R_{fb1} was adjusted to 900Ω. Both voltage and dB gain plots were then re-simulated. These results can be seen in Figures 3.4(a) and (b) respectively. The plots demonstrate

that at a dc voltage gain of 10V/V, flat responses occur for all of the analysed op-amps. However as the gain is increased the bandwidth of each response is reduced by a similar factor with respect to the 2V/V gain results. As expected the gain-bandwidth product is maintained.

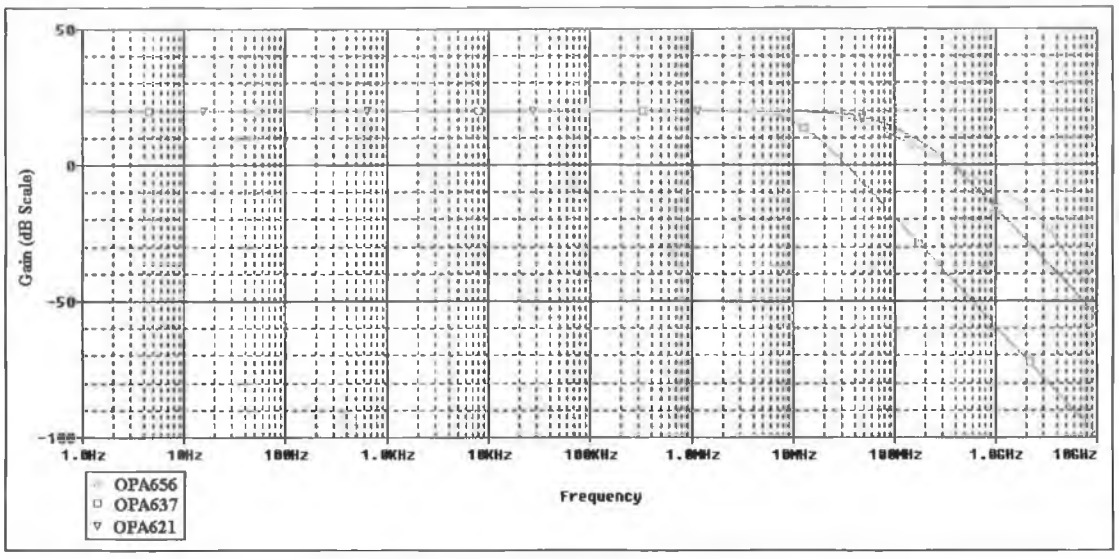


Figure 3.4(b): Bode frequency responses of the OPA656, OPA637 and OPA621 op-amps in non-inverting mode with expected gains of 10V/V (20dB)

The op-amp's respective frequency response ranges are more easily viewed from Figure 3.4(b), as the decibel scale is applied. The most noticeable outcome drawn from this graph is regarding the OPA637's inability to reach the required cut-off frequency set at the high frequency band limit. This result puts a serious doubt on this device's suitability as a possible inclusion within the developing preconditioning module.

Figure 3.4(c) demonstrates the OPA656 and OPA637 device's minor to negligible dc errors for the respective op-amps, as a passband 1MHz frequency low-level 1mV input signal is applied. However with the identical non-inverting arrangement, the OPA621 op-amp manages to output an amplified signal with an offset of +4mV. This offset value is too significant for the OPA621 device to be implemented within a high-performance preconditioning module. Therefore, the module's suitable front-end candidates are reduced to two; the OPA637 and OPA656, with the OPA656 op-amp appearing the most consistent device with respect to precision and speed.

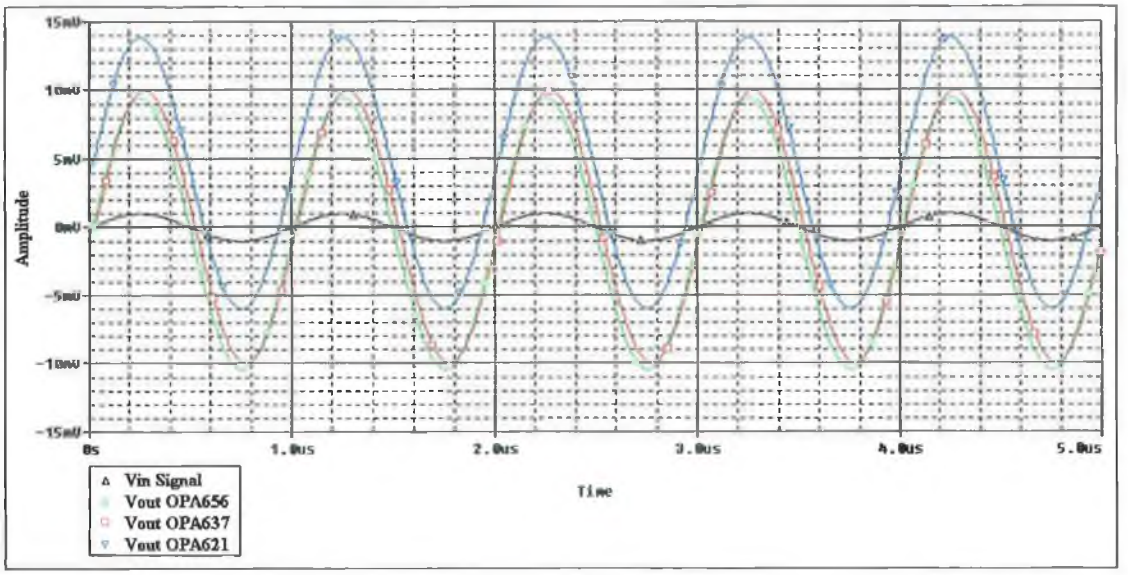


Figure 3.4(c): Signal analysis of the OPA656, OPA637 and OPA621 op-amps in non-inverting mode with expected gains of 10V/V for an input signal frequency of 1MHz

The actual implemented composite circuit will require one of these two precision op-amps as the front-end interface device. The expected gain that this front-end device will operate over will be 2 - 10 V/V (6 - 20 dB). This should explain the op-amp's selected gain values. Hence, the simulated results of Figures 3.3 and 3.4 are of critical importance to the entire system.

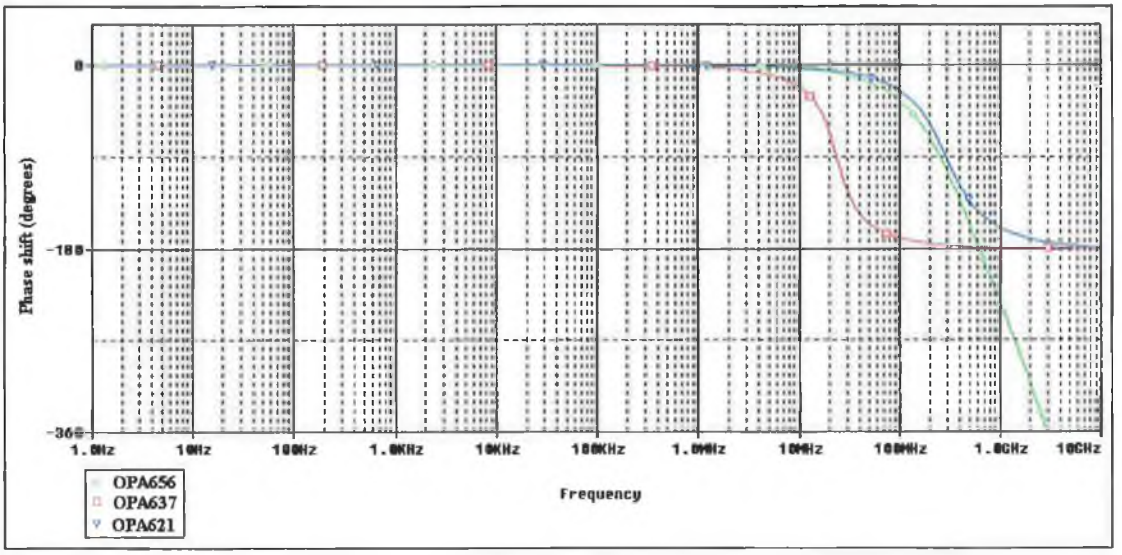


Figure 3.5: AC analysis phase responses of the OPA656, OPA637 and OPA621 op-amps in non-inverting mode

The phase response graph of Figure 3.5 shows the phase shifting effects for the op-amp trio at high frequencies. This error has a bearing on the stability of the circuits, especially if additional amplifier stages are attached. The OPA637's phase plot shows the least desirable response of the three analysed op-amps. Hence, a significant phase shift would result when any low-level input signal with a frequency at or above 10MHz is applied to the OPA637 op-amp. The other two op-amps have negligible phase error within the considered high frequency signal range and are therefore concluded to be more suited to high frequency phase based applications. It is also worth noting that the OPA637's and OPA621's phase error tails off to –180° as the frequency sweeps into the GHz range. These phase plots complete the precision op-amp simulations.

The main conclusions drawn from the simulations is that despite the OPA637 VFA's excellent precision capabilities, its frequency and phase properties fall short of the required standard set for a high-performance preconditioning system. The OPA621 also fails to reach what is required with respect to its precision properties. Therefore, the OPA656 op-amp was selected as the most suitable candidate of the three analysed devices to be implemented as the front-end component of the system.

3.3 High-Speed Circuit Modelling

The analogue preconditioned design must cater for high accuracy dc signal levels while simultaneously having the capability to measure dissimilar ac signals with elevated rates of change (slew rate). In creating such a system both ends of the wave-type spectrum, slow dc high-precision and fast ac-dynamic signals should all be successfully amplified and filtered without distortion. The previous section of this chapter presented three VFA devices manufactured by the Texas Instruments corporation. The OPA656 device was found to be the most suitable candidate, capable of managing the front-end precision aspect of this module. Selecting a suitable device that is capable of achieving the high-speed characteristics of a preconditioning system was also considered.

Device Name	Title	Channels	BW at Av+1 (MHz)	BW at Av+10 (MHz)	Slew Rate (V/us)	Maximum Supply Voltage (Volt)	Offset Voltage (mV)	Input Bias MaxCurrent (nA)	Feedback Type	Settling Time
CLC414	Quad,Low Power Monolithic Op Amp	4	107	57	1000	14	6	10000	Current	16nS to 0.1%
CLC415	Quad,Wideband Monolithic Op Amp	4	260	101	1500	14	5	25000	Current	12nS to 0.1%
CLC420	High Speed,Voltage Feedback Op Amp	1	300	10	1100	14	2	20000	Voltage	12nS to 0.1%
CLC425	Ultra Low Noise Wideband Op Amp	1	-	420	350	14	0.80	40000	Voltage	22nS to 0.2%
CLC428	Dual Wideband,Low Noise,Voltage Feedback Op Amp	2	160	16	500	14	2	65000	Voltage	16nS to 0.1%
CLC430	General Purpose 100MHz Op Amp with Disable	1	100	55	2000	33	7.50	20000	Current	35nS to 0.05%
CLC431	Dual Wideband Monolithic Op Amp with Disable	2	92	60	2000	33	6	16000	Current	70nS to 0.05%
CLC432	Dual Wideband Monolithic Op Amp with Disable	2	92	60	2000	33	6	16000	Current	70nS to 0.05%
CLC449	1.1GHz Ultra Wideband Monolithic Op Amp	1	1200	150	2500	12	7	60000	Current	6nS to 0.1%
CLC452	Single Supply,Low Power,High Output,Current Feedback Amplifier	1	190	105	540	14	6	31000	Current	25nS to 0.05%

Table 3.4: High-speed op-amp parametric selection guide from National Semiconductors

Table 3.4 describes some of the many high-speed CFA and VFA op-amps developed by the National Semiconductor organisation in tabular form. At this point it is worth noting that other op-amp selection guides like those in Tables 3.3 and 3.4 are also available for viewing from Analog Devices and Elantec documents and websites. The illustrated tables were chosen for their particular relevance to the signal conditioning module being investigated. Within Table 3.4, the primary categories of interest include the devices' bandwidth and slew rate. Additional features of interest within this selection guide are the stated V_{OS}, I_B and the bandwidths achieved for unity and 10V/V closed-loop gains.

The CLC449 ultra wide-band monolithic op-amp has the most suitable properties of all the listed devices for high-speed applications. It has the highest specified slew rate with a value of 2500V/μs. The device's bandwidth is exceptionally wide with a typical –3dB cut-off frequency of 1.2GHz at unity gain. This wide-band op-amp supports rise and fall times of less than 1ns and has a stated settling time of 6ns (to 0.1%). Performance advantages were achieved through improvements in National Semiconductor's proven current feedback topology combined with a high-speed complementary bipolar process. In applications using high-speed flash ADCs and

DACs, the CLC449 provides more than the necessary bandwidth (1.1GHz at a gain of +2V/V).

As in the case of the precision investigations, a number of possible candidates are initially listed for the active high-speed filter and signal conditioning functions. The CLC449 is seen to exhibit the best overall and unrivalled dynamic features of the listed devices. Therefore, the CLC449 op-amp was the only high-speed type device selected to be modelled in PSpice software for more detailed analysis. This virtual analysis process was necessary to confirm the device's gain-bandwidth and phase response properties.

Before discussing the modelling and simulating investigations, some of the main drawbacks of CFA are noted. Compared to the precision VFAs of Table 3.3, CFAs generally suffer from poorer V_{OS} and I_B characteristics. Moreover, having much wider bandwidths, typically in the high frequency band, they tend to be noisier.

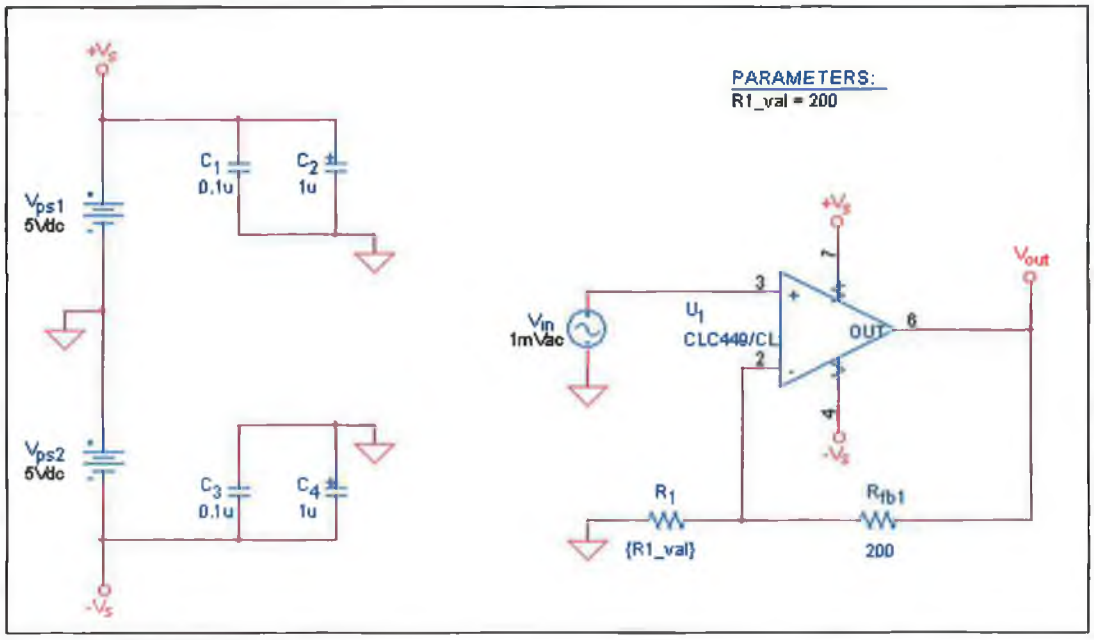


Figure 3.6: CLC449 ultra wide-band high-speed op-amp schematic diagram in non-inverting mode

Figure 3.6 displays the single-stage CLC449 op-amp in non-inverting mode. Apart from the resistor values used, this configuration is identical to the one given for the precision op-amps (Figure 3.2). Each model design was created to produce their

respective ac graphical analysis results. The CLC449 and OPA656 devices employ the same ±5V dc power levels. This permits both devices, implemented on a single circuit board, to be powered by the same supply lines.

The R_1 resistor value in Figure 3.6 is defined in PSpice as a global parameter {R1_val} with an initial value of 200Ω. With a global parameter defined, an increased number of simulation options are possible. The simulation results for a list of distinct resistor values, given in the simulation settings dialogue box, can be displayed simultaneously as each response is superimposed on the same plot. This type of approach to modelling systems allows different gain and bandwidth information to be analysed as critical circuit values are adjusted.

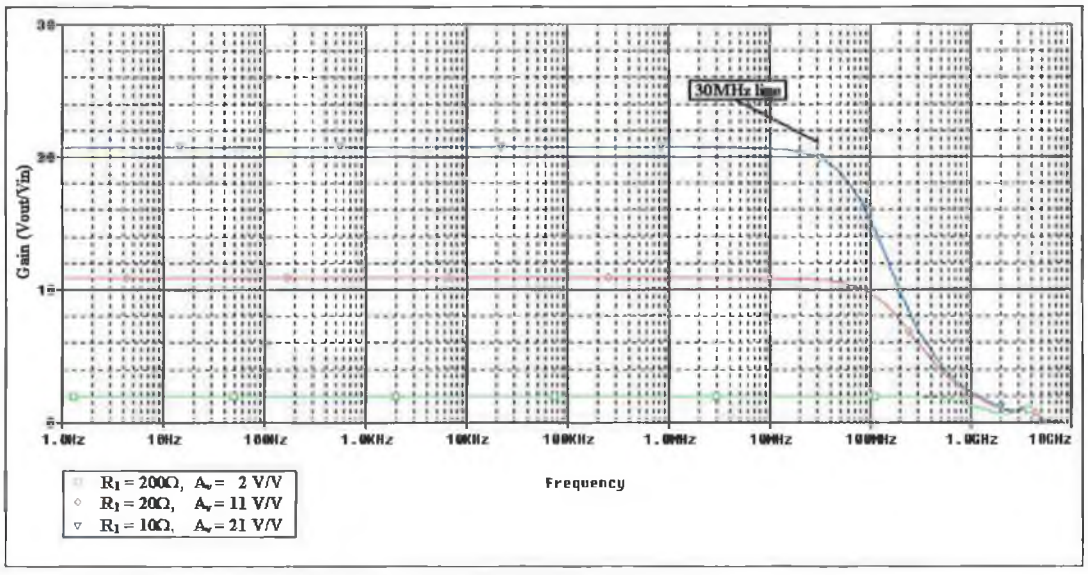


Figure 3.7(a): AC analysis frequency responses of the CLC449 op-amp in non-inverting mode with a range of gains

The normal magnitude and decibel scaled frequency response plots of the CLC449 op-amp are shown in Figures 3.7(a) and (b) respectively. The parameter option is called upon to simultaneously produce a number of different frequency and phase plots. For this example, the R_1 resistor was assigned the values: 200, 20 and 10Ω. The resulting closed-loop dc gains simulated were 2, 11 and 21V/V respectively. These values were chosen for a variety of reasons. The resulting 2V/V dc gain level allowed a direct comparison with the VFA results of Figure 3.3(a) and (b) to be attained. The response plot from the set gain of 11V/V is close enough to the specified 10V/V gain

given in the corresponding selection guide and the gain level results from Figures 3.3(a) and (b) for an approximate comparison to be made. These graphs also indicate the level of gain at which this CFA starts to attenuate high frequency (3-30MHz) signals. As expected the CFAs bandwidth exceeds the VFAs ranges. The simulated gain-bandwidths were found to closely match their specified values in the selection guide and the data sheets.

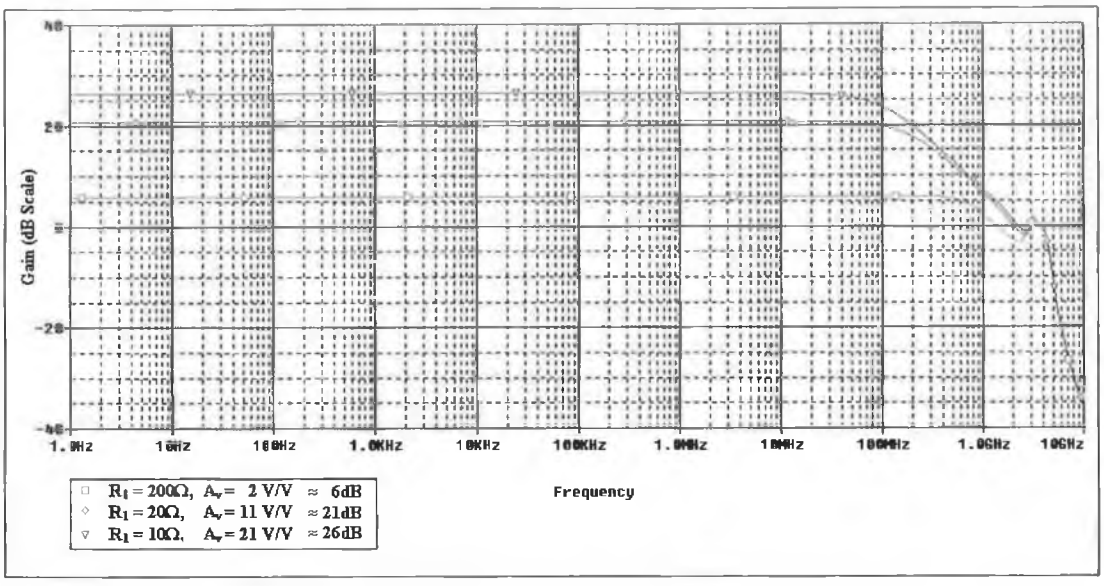


Figure 3.7(b): Bode frequency responses of the CLC449 op-amp in non-inverting mode with a range of gains

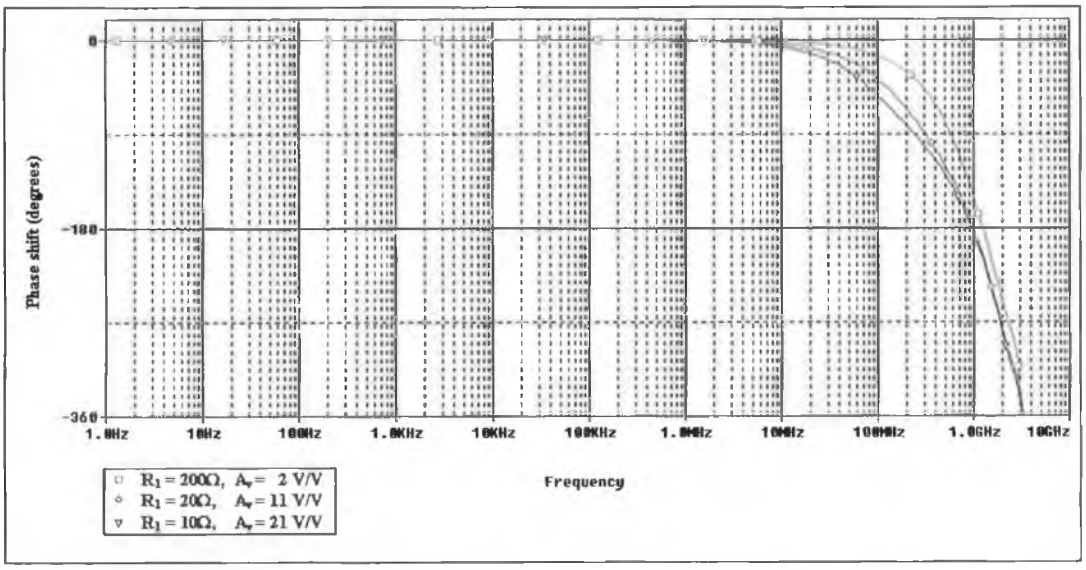


Figure 3.8: AC analysis phase responses of the CLC449 op-amp in non-inverting mode

Figure 3.8 displays the phase response plots of the CLC449 op-amp for a range of gains. These plots clearly show that input signals with frequency components in or beyond the high frequency band experience a greater phase shift as the gain level is increased.

All CFAs exploit a current topology that emphasises current-mode operation. This current-mode is inherently much faster than voltage-mode operation because it is less prone to the effects of stray node-capacitance. CFAs are fabricated using high-speed complementary bipolar processes that allows CFAs to be potentially orders of magnitude faster than VFAs. The modelling results confirmed that the CLC449 wide-band op-amp to be a suitable candidate for high-speed applications and the CLC449 was therefore chosen for the composite system design.

3.4 Composite Amplifier Designs

A composite amplifier system combines two or more op-amps to achieve improved overall performance characteristics [3-2]. The specific composite designs documented in this section connect precision VFAs, in cascade form, with high-speed CFAs. The idea of this composite system is to incorporate the best qualities from each constituent device with a minimum amount of trade-off.

3.4.1 Two-Stage Typical Composite System

The composite amplifier's output voltage range was designed to match a standard ADC input voltage range of ±0.5V. With the amplifier's output range fixed, specific gain/attenuation levels required to accommodate a range of low to medium sensor output signals were considered. The actual gain/attenuation levels used were 500, 50, 5 and 0.5V/V. A two-stage composite amplifier is displayed in Figure 3.9, with an overall gain level of 5V/V. Both op-amps are in non-inverting mode and the overall or individual gains provided by each op-amp are found using Equation 3.1's external resistor value relationship. The distributed gain ratio is 2.5:2 as the front-end precision

op-amp handles more of the overall gain. The VFA could have provided all the gain, CFA acting as a unity gain buffer, without any adverse effect on the system.

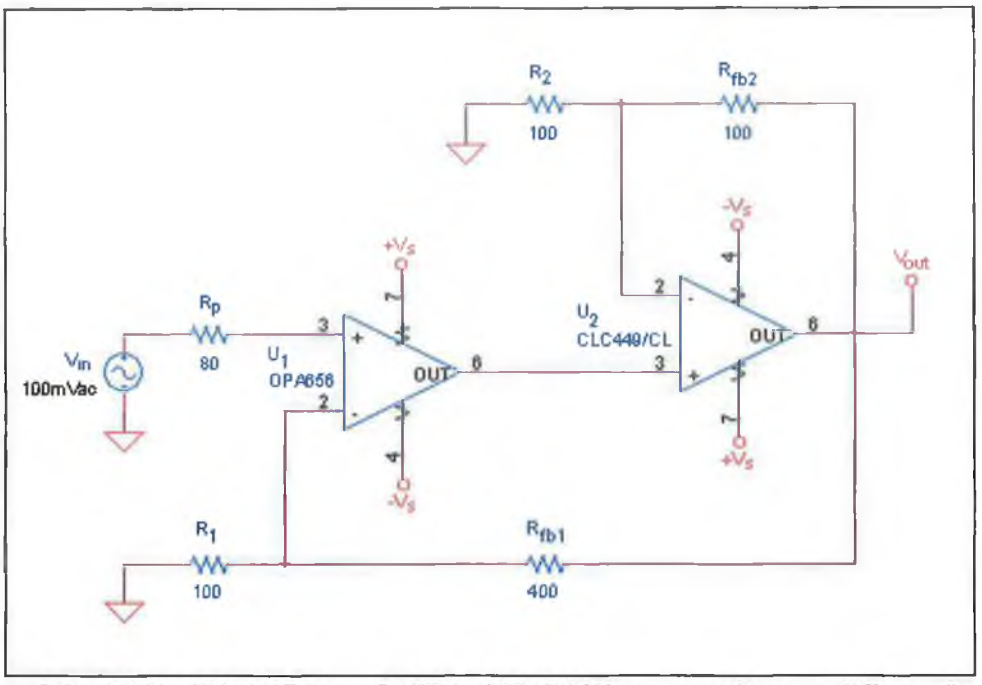


Figure 3.9: VFA (OPA656) and CFA (CLC449) composite amplifier schematic diagram in non-inverting mode with a gain of 5V/V

To ensure that both VFA inputs see the same dc driving impedance and to minimise bias current errors, R_p was chosen as the parallel combination of R_1 and R_{fb1} according to the equation:

$$R_p = \frac{R_1 * R_{fb1}}{R_1 + R_{fb1}} \tag{3.2}$$

Signal and frequency response simulations found this composite circuit to be capable of amplifying signals beyond the high frequency range (>30MHz) without any noticeable distortion being introduced. Alternative gains are achieved simply by adjusting the composite amplifier's external resistor values. This typical composite amplifier forms a useful template design for achieving a range of gains that suit various applications. It was discovered that the selected op-amp arrangement of Figure 3.9 is ideally suited for medium voltage range applications. Such a circuit,

when simulated, yields high precision signals that can be amplified to an upper bandwidth limit of 300MHz.

3.4.2 Three-Stage Complex Composite Design

The lower gain limit of the non-inverting composite amplifier is restricted to unity, while the upper gain range is more flexible. When amplifying high frequency signals by a factor of 50V/V or greater, the three-stage composite version was found to produce an improved frequency response without distortion compared with the typical composite design. To exceed the initial composite amplifier's upper gain limit and corresponding bandwidth a three-stage composite amplifier system was designed (Figure 3.10). A third-stage CFA was added, in series, to increase the total gain of the system. With reference to the op-amps' respective data sheets, relatively low external resistor values were chosen that served to minimise instability in the composite amplifier.

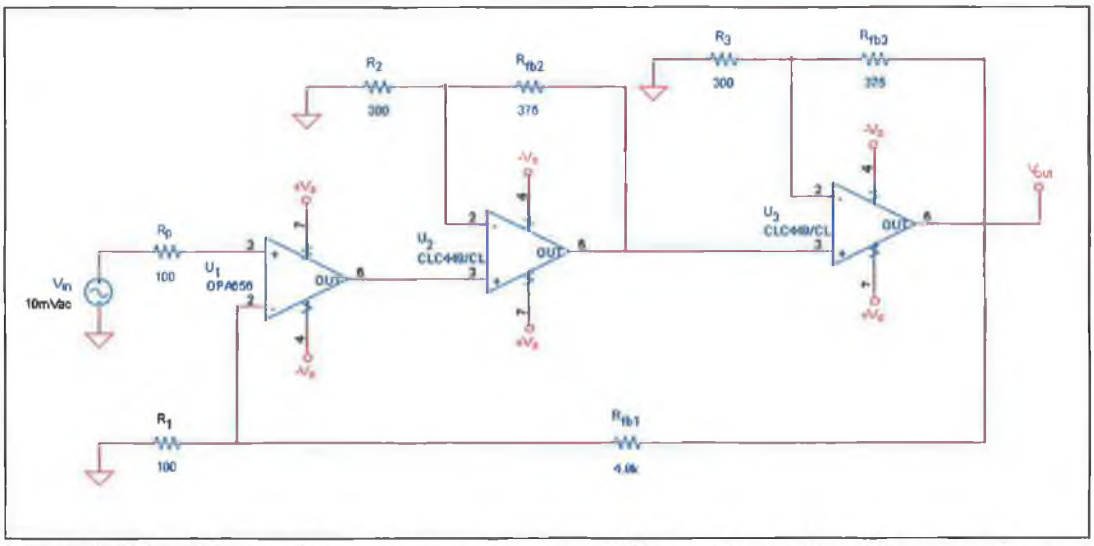


Figure 3.10: Three-stage composite amplifier schematic diagram in non-inverting mode with a gain of 50V/V

With reference to the resistor values in Figure 3.10, the two CFA local loop gains are clearly 2.25V/V. For an overall gain of 50V/V the VFA must provide a gain of approximately 9.9V/V. The combined circuitry shown in Figure 3.10 was found to

work extremely well. A bandwidth in excess of 60MHz was reached when this circuit's frequency response was simulated.

The drawback of introducing an additional op-amp into the composite system is that the op-amp comes with an inherent phase shift. Figure 3.11 accordingly analyses the intermediate and total phase response of the system. The plots show a phase shift introduced by each stage. The amount of phase shift needs to be considered for applications that require minimum phase. For general sensor applications, however, the phase shift is not significant enough to cause instability or noticeable distortion throughout the unattenuated frequency range.

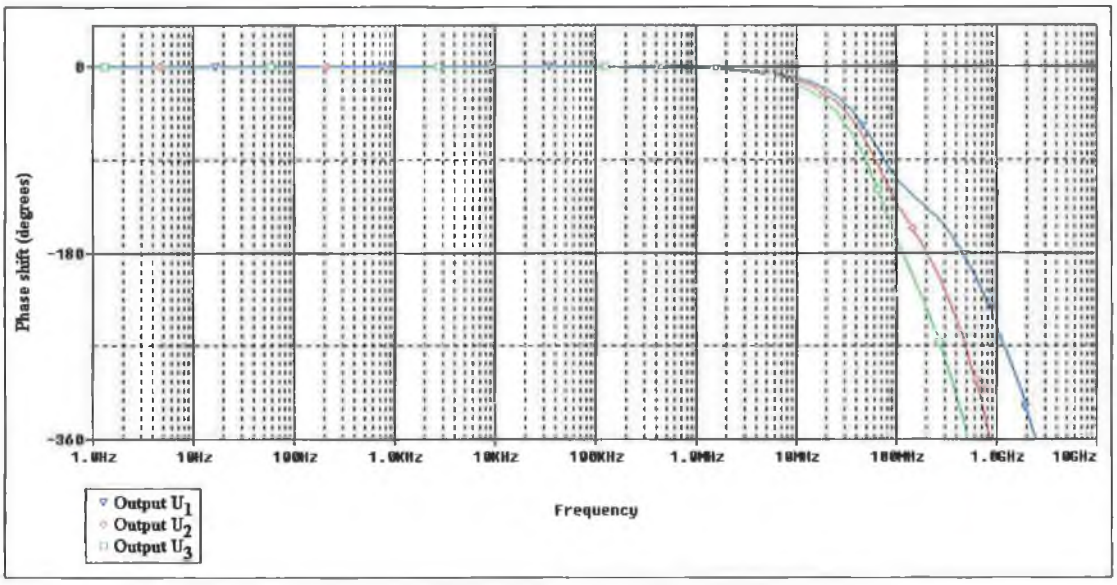


Figure 3.11: AC analysis phase responses of the complex composite amplifier, measured at each op-amp output stage

High-frequency, low-voltage (refer to Table 3.1) signals can also be amplified with the same level of performance by the three-stage composite design. The corresponding set of resistor values, required to achieve a gain of 500V/V, are listed in Table 3.5 in the summary section.

3.4.3 Stray-Capacitance Compensation

Stray input capacitance compensation was considered in attempting to extend the composite amplifiers bandwidth. All practical op-amps exhibit stray input capacitance in the order of a few pF typically. The primary capacitance sources include: the differential capacitance (C_d) at the input pins, the common-mode capacitance ($C_c/2$) of each input to ground and the external parasitic capacitance (C_{ext}) of components, leads, sockets and printed circuit traces associated with the inverting input node [3-3]. These various stray input capacitances, shown in Figure 3.12, can cause a phase lag in the amplifier system. A common way of counteracting this lag is by using a feedback capacitor (C_{fb1}) also shown in Figure 3.12, to create an opposing phase lead. The inclusion of a capacitor shunting the feedback resistor can favourably alter the bandwidth of the amplifier.

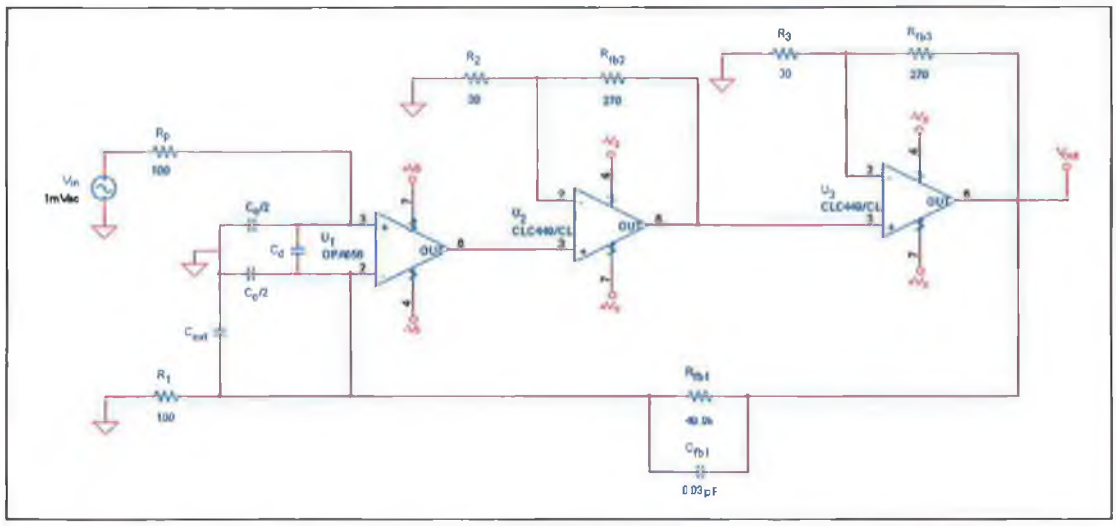


Figure 3.12: Three stage preconditioning composite amplifier schematic diagram in non-inverting mode with a gain of 500V/V illustrating stray input and feedback compensation capacitance

From Figure 3.12, the portion $C_1 = C_c/2 + C_{ext}$ is in parallel with R_1, so the original gain Equation 3.1 must be modified to include all impedance sources. The ideal gain is therefore:

$$\text{Gain}_{\text{ideal}} = \frac{Z_1 + Z_2}{Z_1} = 1 + \frac{Z_2}{Z_1} \qquad (3.3)$$

with $Z_1 = R_1 \parallel (1/j2\pi fC_1)$ and $Z_2 = R_{fb1} \parallel (1/j2\pi fC_{fb1})$, and the resulting value of C_{fb1} to make the ideal gain frequency-independent is:

$$C_{fb1} = \left(\frac{R_1}{R_{fb1}}\right) * \left(\frac{C_c}{2} + C_{ext}\right) \tag{3.4}$$

With typical values of C_c and C_{ext} at 10pF and 8pF respectively and taking the resistor values given in Figure 3.12, C_{fb1} is calculated to be approximately 0.03pF. This capacitor value is not realisable in practice but can be simulated to illustrate the compensating bandwidth effects.

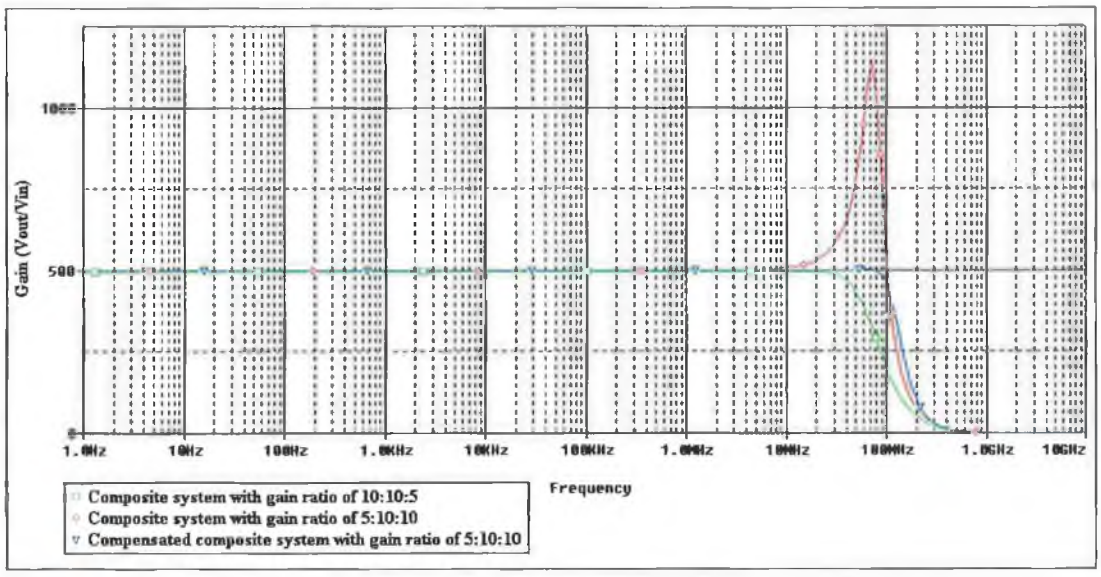


Figure 3.13: AC analysis frequency responses of the three-stage non-inverting composite amplifier with contrasting gain ratio distributions for an overall gain of 500V/V

Without a small C_{fb1} inserted into the circuit, as shown in Figure 3.12, a potentially unstable system results. When high frequency signals are applied, the presence of peaking in the uncompensated system's frequency response is manifested as ringing in the time domain. The uncompensated, third-order, under-damped system becomes unstable as the inherent phase shift of each op-amp combines with the input phase lag, caused by the stray capacitance, to produce a phase margin of zero. Figure 3.13 indicates that the stability of the system depends on how the gain is distributed across each op-amp. The system suffers most significantly when the backend CFAs are relied upon to handle the majority of the system's gain. The stabilising effect of C_{fb1} is

also shown in Figure 3.13 (refer to the blue line), which counteracts the stray capacitance, increases the phase margin and widens the system's bandwidth. Thus Figure 3.13 illustrates and compares the effects between a compensated and uncompensated system and when the overall gain is distributed more heavily across the front and back ends respectively.

3.4.4 Inverting Composite Models

The non-inverting composite designs are particularly suited to amplifying analogue signals from high impedance sources. However, without the use of a potential divider, a non-inverting amplifier's lower gain/attenuation limit is unity. To handle a wider range of signals the inverting composite arrangement, with simplified external resistor gain relationship and unrestricted amplification/attenuation range was considered.

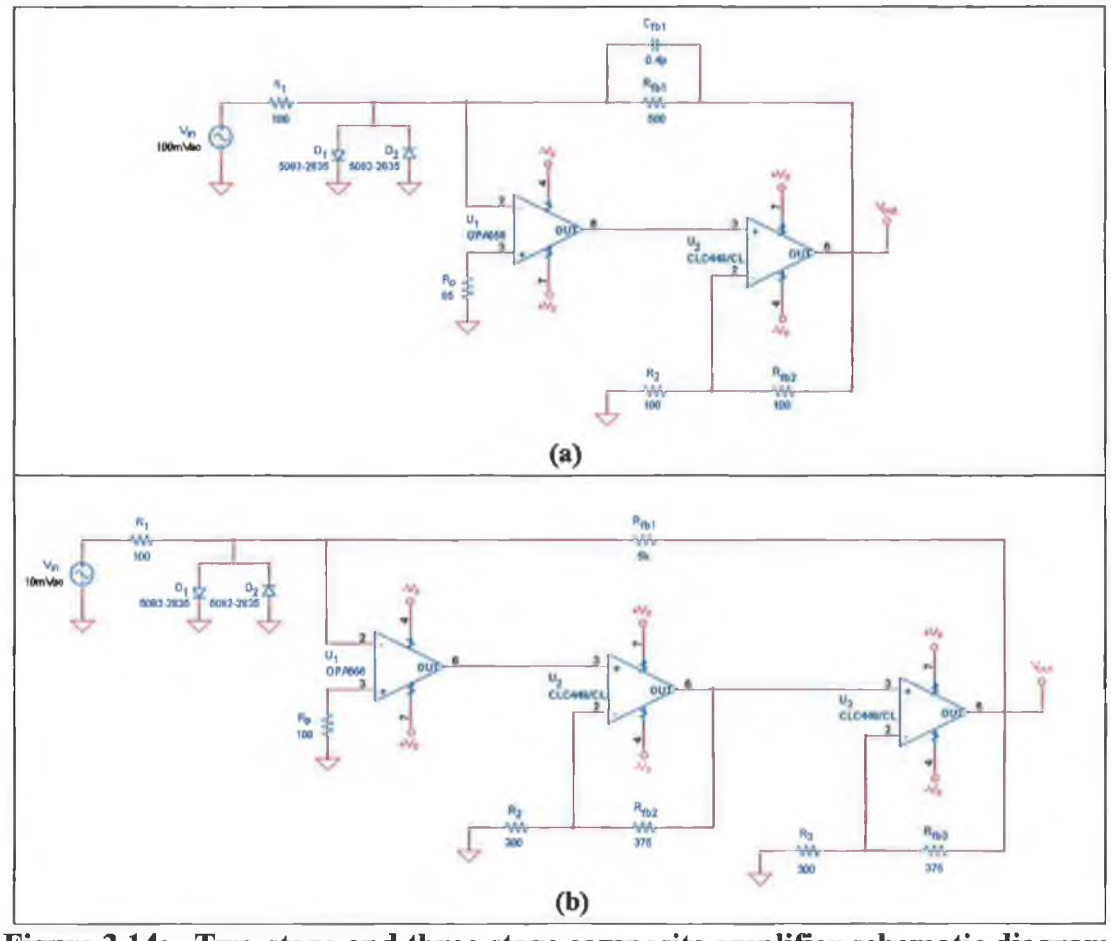


Figure 3.14: Two-stage and three-stage composite amplifier schematic diagrams in inverting modes, (a) gain set to –5V/V (b) gain set to –50V/V

Figure 3.14 displays the two inverting composite amplifier designs that can collectively produce negative gain ranges from –0.5V/V to –500V/V. The circuit diagram of Figure 3.14(a) was developed for use as a –0.5V/V and –5V/V gain system. The three-stage complex composite arrangement of Figure 3.14(b) was designed specifically for gains of –50V/V and –500V/V. Switching between one gain option and another requires resistor selection. The specific resistor values given in Figures 3.14(a) and (b) respectively produce gains of –5V/V and –50V/V.

Anti-parallel diodes were inserted across the amplifier's inverting input pin and ground. In the event that excessive voltage on the input pin is applied, this diode clamping technique protects the input stage from permanent damage. However, the capacitance value of the diodes can affect an amplifier's frequency response. PSpice simulations indicated that a diode capacitance of a few pFs, or lower, caused a negligible effect on the systems frequency response. Therefore, diodes with low capacitance values (pF range) and high switching speeds were chosen. For these reasons Schottky barrier diodes are an ideal choice. For example the 5082-28xx Series Schottky diodes with a total capacitance as low as 1pF, or the HSMP-386x Series surface mount diodes, with an even lower related capacitance of 0.2pF, are both suitable as voltage limiting devices. Both of these low capacitance high frequency Schottky barrier diodes are available from Agilent Technologies.

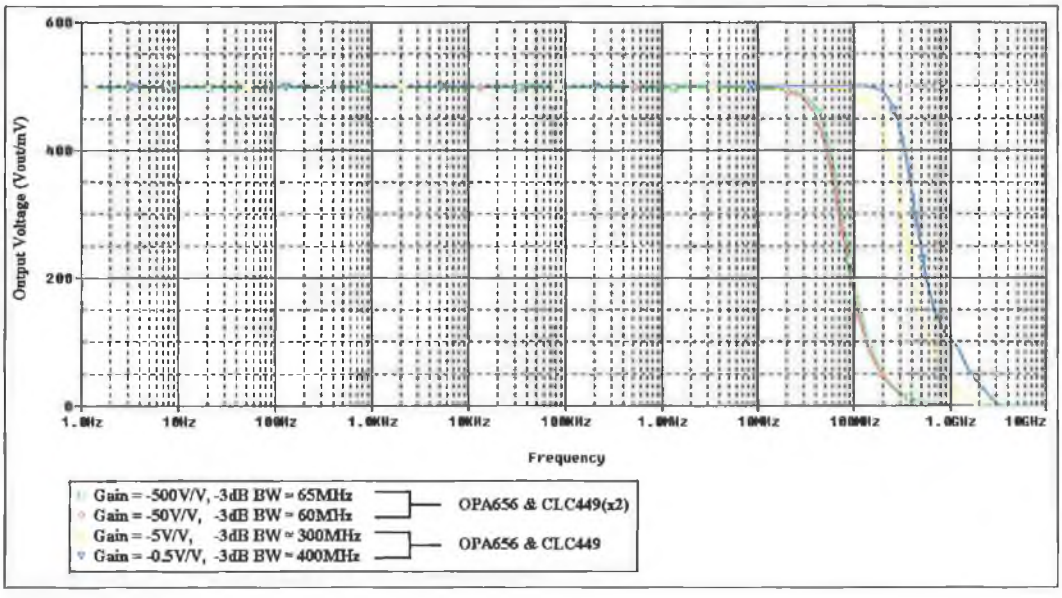


Figure 3.15: AC analysis frequency responses of the two-stage and the three-stage inverting composite amplifiers

Figure 3.15, simultaneously, shows the four gain (–0.5, –5, –50 and –500V/V) versus frequency plots. Each spectrum response of the inverting amplifier displayed similar bandwidths when compared to their corresponding non-inverting amplifier plots. Where possible the same set of resistor values for the non-inverting and inverting amplifier options were used. By retaining the same resistor values for a given gain level, only a minimum amount of modification is needed to switch between the two alternative input options. This arrangement, where there is an alternative input option for a sensor's output, widens the range of possible applications and improves the generic ability of the overall system.

3.4.5 Gain Selection Systems

An efficient and cost-effective method of adjusting the composite amplifier's gain range was investigated. A suitable variable gain arrangement can control both the individual op-amp's and overall amplifier's gain range. Digital potentiometers, analogue switches and relays were initially considered as possible gain selection devices suitable for the composite amplifier system.

The features of the digitally controlled potentiometers fit the range of resistance values to function as a variable resistor. The Analog Devices AD8403 has four variable resistors per package, each with a nominal (maximum) resistance of 1kΩ and 256 wiper positions. Other useful features, like linear taper and non-volatile wiper memory, are available from the Dallas Semiconductor Corporation.

Unfortunately, these components have inherent limitations that precludes their use in this context. In particular the devices resistance temperature coefficient (RTC) and wiper resistance (R_W) were the main sources of error. For instance, if a typical RTC is stated at 500ppm/°C, a significant change in the device's resistance will result. With an operational range in temperature (ΔT) of 40°C, the resistance temperature coefficient will be equivalent to a 2% change in resistance. The effect of this change on a feedback resistor causes an unacceptable gain error of approximately ±2% for all the gain levels considered.

R_W was also found to be around the same level of magnitude of some of the local feedback resistor values, of 50-200Ω. Hence the minimum achievable resistance is not zero but is determined by R_W. This situation is generally more suited for higher resistive applications in the MΩ scale range.

Like the digital potentiometers, analogue switches have their limitations that prevent them from being used as part of the composite amplifier's gain control system. The analogue switche's on-resistance (R_{on}) varies from one device to another with typical values being 5-500Ω which is unacceptable in a precision amplifier system.

The digital potentiometers and analogue switches are both suited for high resistive arrangements in the MΩ range whereby any internal resistance errors have a negligible effect on the overall impedance. A more accurate source of gain control is required, since the majority of the external resistor values used in conjunction with the op-amps of the composite system are of relatively low magnitudes. Matched and/or dummy circuit techniques could be used in the feedback path of the amplifier to compensate for the undesirable resistance properties associated with the digital potentiometers and analogue switches, however only with added expense and limited effect.

Mercury-wetted and dry reed relays, available from Pickering Company, were considered because of their suitability for wide bandwidth applications and any other application where exceptionally low levels of inter-terminal capacitance are required. The Omron Corporation produces other compact electromechanical PCB miniature relays, of interest. The most common maximum contact resistance stated for the Omron PCB relays is 50mΩ. This compares to 150mΩ for the dry and 75mΩ for the mercury-wetted reed relays of the Pickering series. These values are considerable low with respect to the analogue switch resistance and the R_W of the digital potentiometers. They have a negligible effect on an amplifier's set gain.

The most useful multi-pole relay contact forms include SPDT (single-pole, double-throw), DPDT, TPDT and 4PDT. However, the 103 Pickering series, low capacitance SIL reed relays appear to be the only relay type that has exceptionally low levels of

inter-terminal capacitance, which has a negligible effect on the amplifier's external impedance. This series of relays has an open switch associated capacitance that is <<1pF. As expected, these levels are considerably less than the rating capacitance values of the general-purpose multi-pole relays. At present, this low capacitance relay series is only available in one switching form (1 Form A), also referred to as a SPST. This same switching form is available in a number of different configurations, with guard and magnetic co-axial screens being of particular interest for the composite amplifier design.

In high-density applications, as in the composite arrangement, when more than one relay may be operating at any time, it is necessary to use a relay that includes internal mu-metal magnetic screening. Without this magnetic screening, the operational and release voltages of the relays will be altered by the extraneous fields from adjacent devices. This means that when temperature effects are also considered, it may not be possible to operate a relay at its nominal coil voltage. A magnetic screen is therefore important when relays are used in this way [3-4].

Two switching types are also available in this series of relays. A sputtered ruthenium type device is particularly good for switching low currents and/or voltages and is therefore the recommended type for the composite amplifier system. Since these suitable low capacitance reed relays are only available in the simplest SPST form, the multi-pole option is not currently viable. If however this situation changes and multi-pole (SPDT and/or DPDT) low capacitance relays are produced in the future, the relay arrangement required to control the gain of the composite amplifier could appear similar to the circuit diagram of Figure 3.16.

This circuit diagram also contains the least number of op-amps necessary to successfully perform all four gain ranges within a single, stand-alone, composite amplifier design. For clarity, the clamping diodes are not shown in Figure 3.16. The four SPDT and two DPDT multi-pole relays act as internal "multiplexers / demultiplexers" along the feedback path and at the intermediate op-amp stages. They distribute and transmit signals along different paths with differing impedance properties in an effort to produce the gain required. Due to the physical properties of

these relays a period of transients (ringing) should be expected (typically in the μs range) while each switch closes / opens.

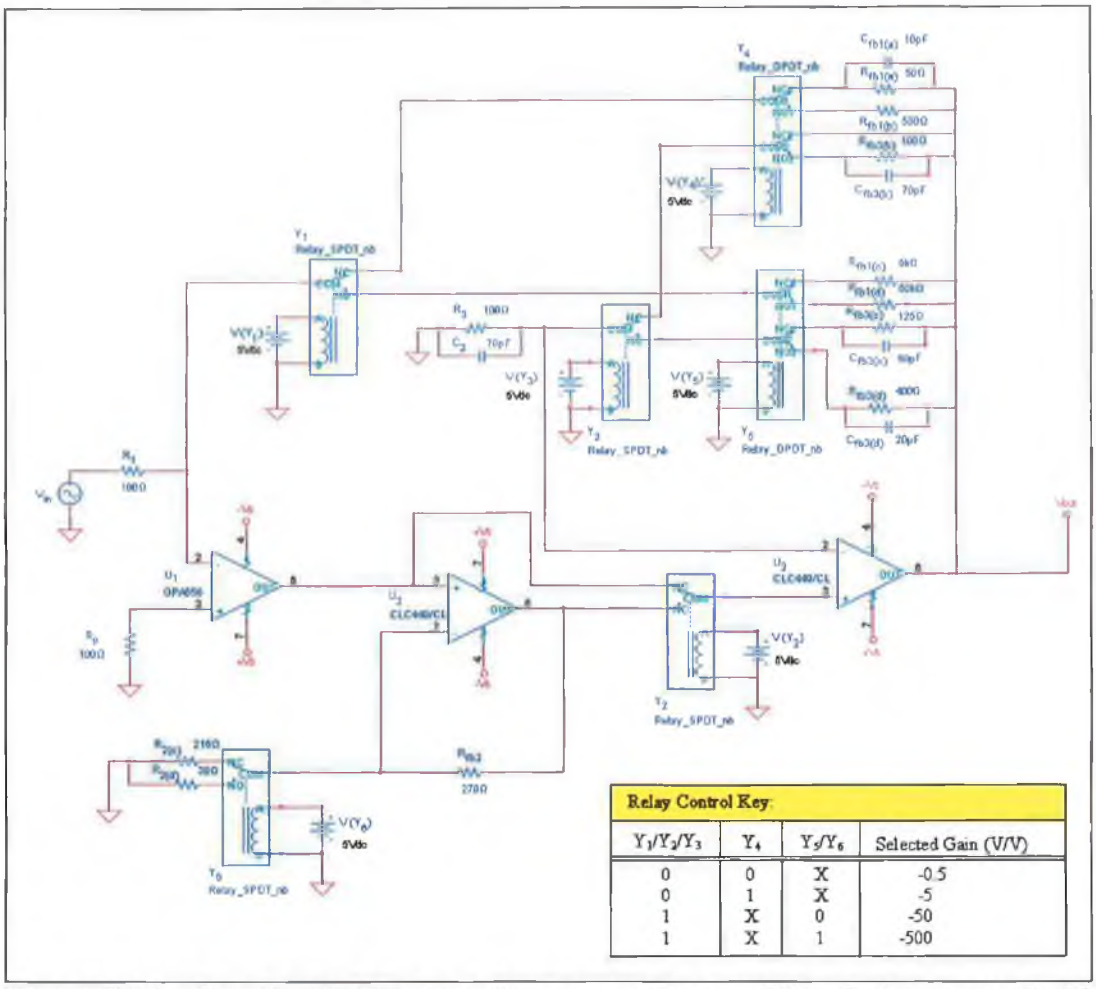


Figure 3.16: One i/p-o/p line inverting composite amplifier design controlled by multi-pole relay, gain selection system

To reduce the number of system components, some of the op-amps' external resistors (R_1, R_p, R_{fb2} and R_3) are fixed and are not controlled by any relay. This reduction causes their matched impedance components to be adjusted from their previously specified values. For example, R_2 in Figure 3.14(b) for the –50V/V gain system is stated at 300Ω when R_{fb2} is equal to 375Ω. Here R_{fb2} is restricted to 270Ω, therefore R_2 must be adjusted accordingly to 216Ω to ensure the appropriate gain of 2.25V/V is maintained.

Similarly R_3 and R_{fb3} in Figure 3.16 relate to the R_2 and R_{fb2} respective values in Figure 3.14(a) for the two-stage amplifier only. This is a consequence of the middle

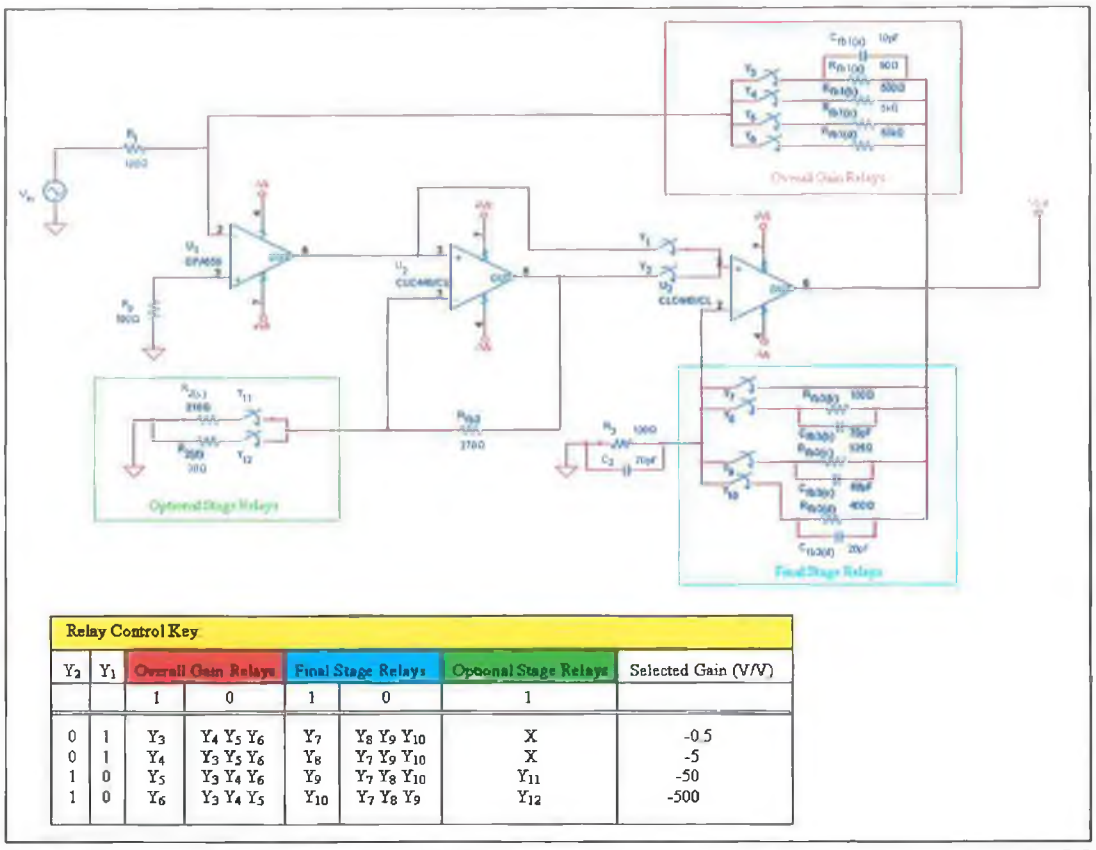
op-amp (U_2-CLC449) being bypassed by the SPDT relay labelled Y_2, which therefore permits the lower system gains (–0.5 and –5V/V) to operate. The majority of the external resistors and capacitors components have been labelled with the bracketed letters a, b, c or d. In increasing order, these letters relate to the selected components of a specific gain range: (a) refers to a gain of –0.5V/V, (b) to –5V/V, (c) to –50V/V and (d) to –500V/V.

When a low state is applied to the normally closed (NC) Y_1-Y_4 relays and with Y_5 and Y_6 having no bearing on the results (don't care, X state), the system functions as a -0.5V/V gain (attenuation) amplifier. The control key in Figure 3.16 shows how the other gain ranges are selected via the relays, where the normally open contact is closed when 5V (high state) is applied to the coil of the relay. It is recommended that the respective excitation voltages are wired together for the relays labelled Y_1-Y_3. They are always set to the same state according to the control key. This also applies to Y_5 and Y_6 to prevent any signal conflicts.

The circuit diagram of Figure 3.16 was successfully modelled and simulated producing high frequency bandwidth results, when the multi-pole relay's capacitance properties were omitted. Successful physical operation of this design is entirely dependent on (currently unavailable) low-capacitance multi-pole relays being manufactured. A more realistic gain selection system, which includes readily available relays, is illustrated in Figure 3.17. The single-pole 103 Pickering series low capacitance relays are shown as single switches in Figures 3.17, 3.18 and 3.19. These designs demonstrate the most cost-effective high-performance gain selection options available for the composite amplifiers.

Figure 3.17 shows a useful relay gain selection system with only one-input and one-output line. The limited switching capability of the single-pole relays requires twice as many relays to maintain the systems gain options, with respect to the multi-pole version given in Figure 3.16. Figure 3.17's control key lists all twelve relays and their switching states necessary to amplify an input signal by the selected gain. Logic devices can then be included to avoid any undesirable situation where complementary relay states are in conflicting modes of operation. This compact design employs three op-amps and twelve single-pole low capacitance relays. Assuming a low quantity

competitive op-amp price of €6 each and each relay at €3; the one-input, one-output line active devices in Figure 3.17 amount to a total of €54.

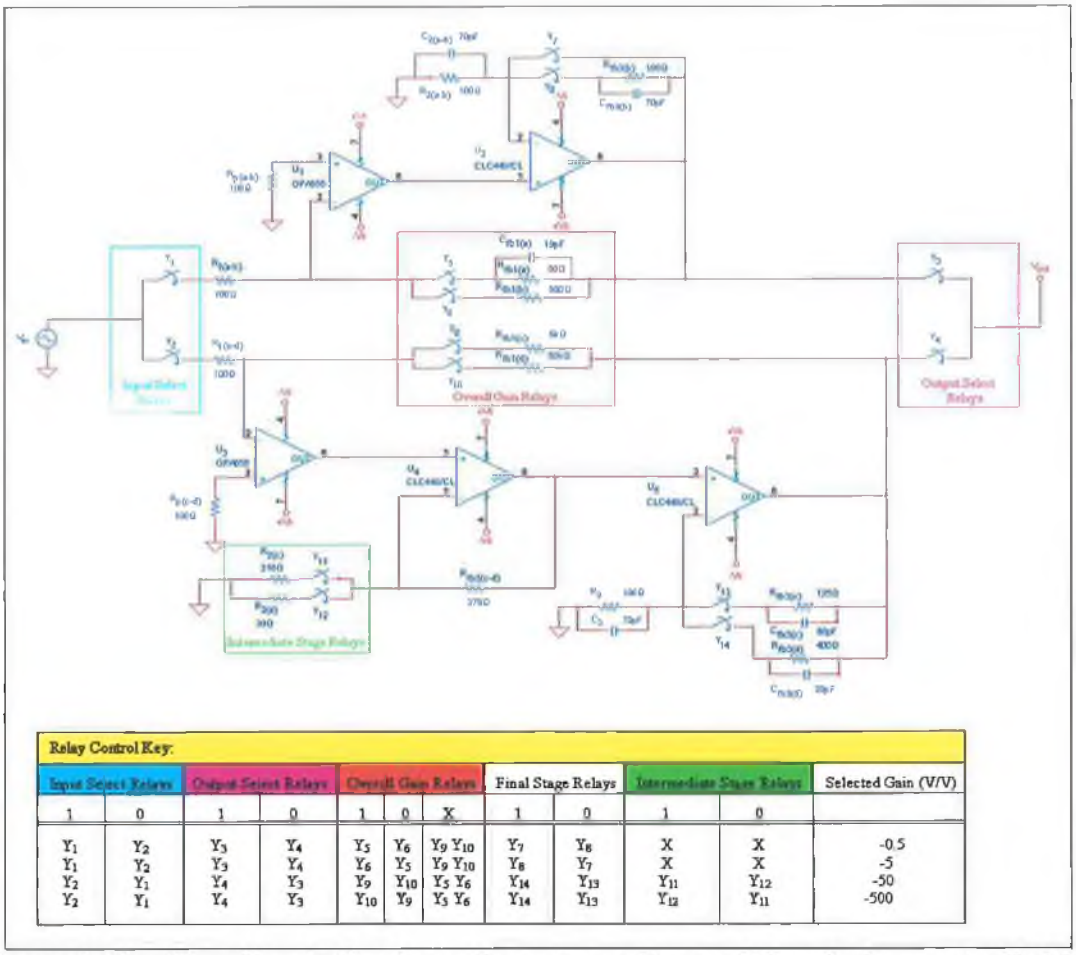


Relay Control Key

Y_2	Y_1	Overall Gain Relays		Final Stage Relays		Optional Stage Relays	Selected Gain (V/V)
		1	0	1	0	1	
0	1	Y_3	Y_4 Y_5 Y_6	Y_7	Y_8 Y_9 Y_{10}	X	-0.5
0	1	Y_4	Y_3 Y_5 Y_6	Y_8	Y_7 Y_9 Y_{10}	X	-5
1	0	Y_5	Y_3 Y_4 Y_6	Y_9	Y_7 Y_8 Y_{10}	Y_{11}	-50
1	0	Y_6	Y_3 Y_4 Y_5	Y_{10}	Y_7 Y_8 Y_9	Y_{12}	-500

Figure 3.17: One i/p-o/p line inverting composite amplifier design controlled by single-pole relay, gain selection system

Figure 3.18 shows a two-input, two-output line design that separates the lower gain two-stage composite amplifier from its higher gain three-stage equivalent. Separate lines allow for improved flexibility of the external impedance values. For instance, $R_{2(a\text{-}b)}$ (which can be seen at the top of Figure 3.18), is the appropriate value for two neighbouring gains (–0.5 and –5V/V) only, instead of the more extreme alternative for which it could apply for all four gain ranges simultaneously. The disadvantage of this circuit comes down to size and cost. An increased number of active components are required to implement the more complex design. The complexity is reflected in the relay's control key, where a larger permutation count is needed. A price tag of €72 is the result of using five op-amps and fourteen relays in the design.

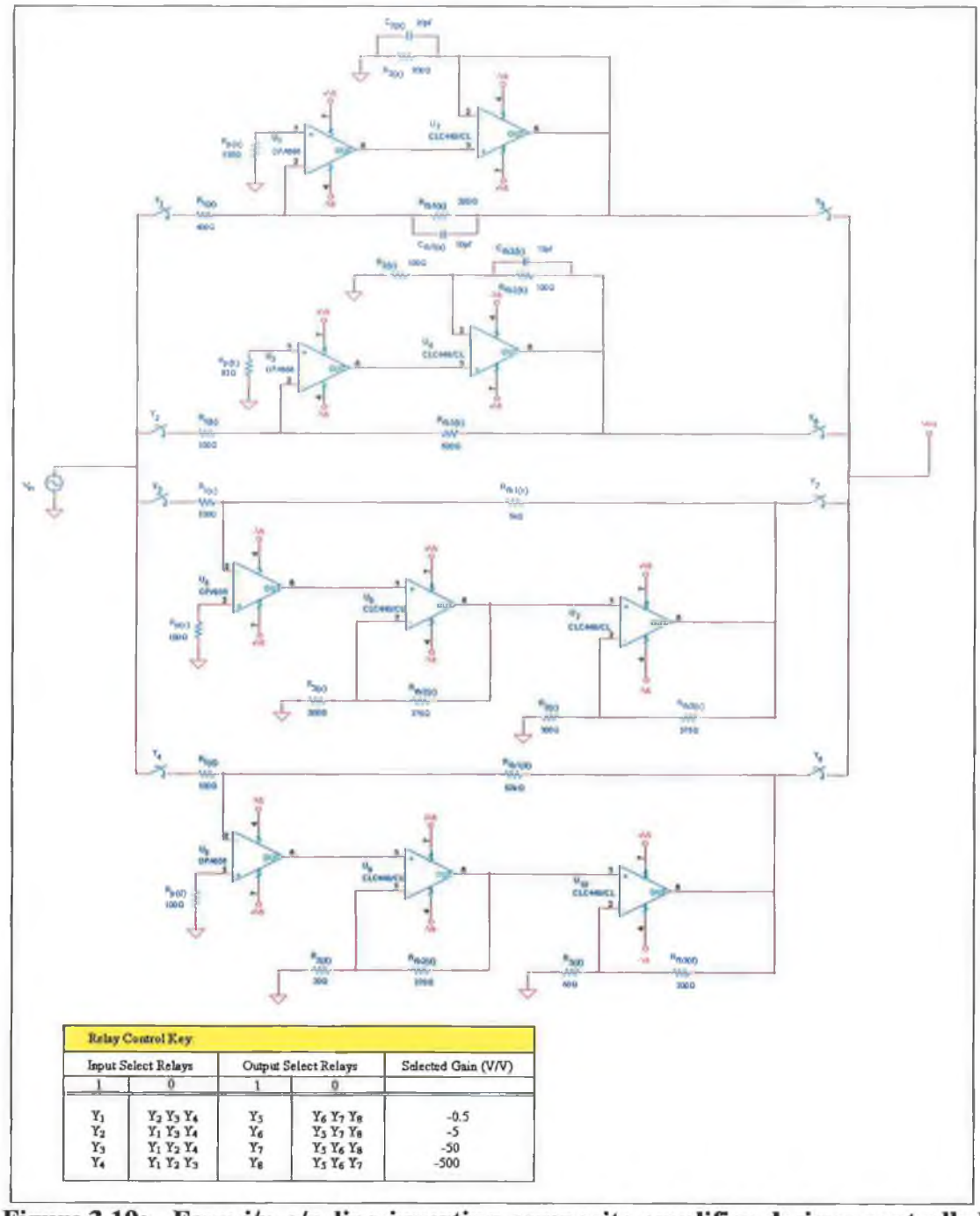


Relay Control Key:

Input Select Relays		Output Select Relays		Overall Gain Relays			Final Stage Relays		Intermediate Stage Relays		Selected Gain (V/V)
1	0	1	0	1	0	X	1	0	1	0	
Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_9 Y_{10}	Y_7	Y_8	X	X	-0.5
Y_1	Y_2	Y_3	Y_4	Y_6	Y_5	Y_9 Y_{10}	Y_8	Y_7	X	X	-5
Y_2	Y_1	Y_4	Y_3	Y_9	Y_{10}	Y_5 Y_6	Y_{14}	Y_{13}	Y_{11}	Y_{12}	-50
Y_2	Y_1	Y_4	Y_3	Y_{10}	Y_9	Y_5 Y_6	Y_{14}	Y_{13}	Y_{12}	Y_{11}	-500

Figure 3.18: Two i/p-o/p line inverting composite amplifier design controlled by single-pole relay, gain selection system

Figure 3.19 shows a four-input, four-output line gain selection system. An increased number of active devices (10 op-amps and 8 relays) is associated with this four-input, four-output line system and costs €84 approximately. The cost alone supersedes any useful advantages that may result from such a system. Therefore, the one-input, one-output line single-pole relay gain selected system was determined to be the most competitive and realistic design option for the composite amplifier.



Relay Control Key				
Input Select Relays		Output Select Relays		Selected Gain (V/V)
1	0	1	0	
Y_1	$Y_2\ Y_3\ Y_4$	Y_5	$Y_6\ Y_7\ Y_8$	-0.5
Y_2	$Y_1\ Y_3\ Y_4$	Y_6	$Y_5\ Y_7\ Y_8$	-5
Y_3	$Y_1\ Y_2\ Y_4$	Y_7	$Y_5\ Y_6\ Y_8$	-50
Y_4	$Y_1\ Y_2\ Y_3$	Y_8	$Y_5\ Y_6\ Y_7$	-500

Figure 3.19: Four i/p-o/p line inverting composite amplifier designs controlled by single-pole relay, gain selection system

3.5 Summary

Front-end analogue amplifier systems form essential sensor interfaces that condition raw data into suitable output signals that match the input range of a connected ADC. High-performance standards were established for the analogue amplifier. Specific parameters include:

- wide operating frequency range from dc to 30MHz (high frequency band limit)
- high to low gain/attenuation level (defined in Table 3.2) flexibility
- overall amplifier V_{OS} of $\leq \pm 0.1$mV
- typical I_B of $\leq \pm 2$pA
- negligible drift in set gain level over a temperature gradient of 0-40°C

These specifications allow for an extended number of applications to be handled by the same flexible amplifier system.

A short list of precision VFA and high-speed CFA devices were selected for further analysis using PSpice simulation software. When individually modelled, these particular devices produced distinctive results. In general, the CFAs were found to have greater bandwidth and speed characteristics, while the VFAs were more suitable for high precision preconditioning applications. Within a composite amplifier arrangement, precision VFAs and high-speed CFAs are combined in cascade form. The OPA656 VFA from Texas Instruments was selected to perform the front-end precision operation and the CLC449 CFA from National Semiconductor was selected to maintain the high-speed system requirements.

Composite Amplifier Type (front end input)	AC Simulated Input / V_{in} (mV)	Overall Gain / Attenuation (V/V)	Approximate Bandwidth (MHz)	Network Resistor Values (Ω)						
				R_1	R_{fb1}	R_2	R_{fb2}	R_3	R_{fb3}	R_p
Non-Inverting	≤±1	500	60	100	49.9k	30	270	50	200	100
	≤±10	50	60	100	49k	300	375	300	375	100
	≤±100	5	300	100	400	100	100	-	-	82
	≤±1000	0.5 + p.d.	300	400	200	0	100	-	-	135
Inverting	≤±1	-500	65	100	50k	30	270	50	200	100
	≤±10	-50	60	100	5k	300	375	300	375	100
	≤±100	-5	300	100	500	100	100	-	-	82
	≤±1000	-0.5	400	400	200	0	100	-	-	135

Table 3.5: Summary of modelled composite amplifier properties with an output voltage swing of ±0.5V

To explore the composite amplifier's adaptability, its inverting and non-inverting designs were analysed. The simulation results are summarised in Table 3.5. In general, both non-inverting and inverting composite model designs produced similar frequency responses for a set gain. Four-input signal categories with resultant dc gain, frequency limitations and associated resistor values are all listed in Table 3.5 for each

composite amplifier type. Whenever possible, similar magnitudes of resistors were employed for both inverting and non-inverting composite amplifier types. Therefore, a minimum amount of adjustment is required when switching between amplifier input types.

For all the subdivided and related dc gain settings, compatible bandwidth simulation results were found for the two amplifier types. The two-stage composite amplifier's bandwidths is considerably wider than that the three-stage versions. All the gain arrangements listed in Table 3.5 produce bandwidths that exceed the high frequency (3-30MHz) band. Thus, their dynamic properties are theoretically capable of handling successfully a range of high-speed sensor signals. In fact, with suitable adjustment to the op-amps gain ratios and the capacitor compensation technique used, the simulated two-stage amplifier's bandwidths listed were found to extend beyond 100MHz.

It should be noted that the dashes (-) in the R_3 and R_{fb3} resistor columns refer to the fact that a third op-amp and its external resistive components are not required by the two-stage typical composite arrangements in order to achieve the associated gains, which are listed in Table 3.5. The zero resistor value given in the R_2 column of both of the 1V input signals means a short circuit was deemed appropriate between the CLC449's output and inverting input pins for unity gain. The abbreviated letters p.d. in Table 3.5 is a reminder that the non-inverting composite amplifier requires an external potential divider circuit to achieve an overall attenuation level of 0.5V/V.

When the systems were simulated with the particular network of resistors listed in Table 3.5, it was found that the two-stage composite amplifiers with their low gain settings were more susceptible to the effects of stray input capacitance. The systems exhibited underdamped characteristics with peaking in the frequency domain being observed. To compensate for these underdamped properties, the respective –5 and −0.5V/V inverting systems required 1.4 and 0.4pF capacitors (C_{fb1}) within the overall feedback loops. The corresponding non-inverting gain systems required 0.4 and 0.7pF capacitors respectively.

A cost-effective and compact gain selection system was also considered. Relays were found to be the most suitable devices for adjusting the system's gains while retaining

the system's high-performance. A number of gain selection methods for incorporating all of the necessary two- and three-stage composite amplifiers were investigated. A one-input, one-output line, single-pole relay gain selection system was found to be the most competitive and realistic design examined.

4

Composite Amplifier Assembly Evaluation

Prototype boards are used to test a system's practical functionality. To confirm a system's actual performance, dc precision and ac signal results are analysed. The evaluation of a system's practical performance means non-ideal, real-world issues such as noise, signal interference, parasitics, proximity effects and other circuit board problems are introduced and can be observed and ideally overcome.

This chapter deals with the physical implementation and analysis of the composite amplifier system. The internal and external issues, which are particularly relevant when developing a high-performance, analogue, preconditioning system, are also discussed.

4.1 Implemented Typical Composite Amplifier

The OPA656 and CLC449 op-amps are both inexpensive devices, produced in a number of package types. The composite amplifier was implemented using the OPA656's small outline package (SOP) and the CLC449's dual-in-line package (DIP). Figure 4.1 illustrates the implemented two-stage inverting composite amplifier with a gain of –5V/V. The external amplifier components used include power supply regulators, bypassing capacitors, input clamping diodes and resistive loads.

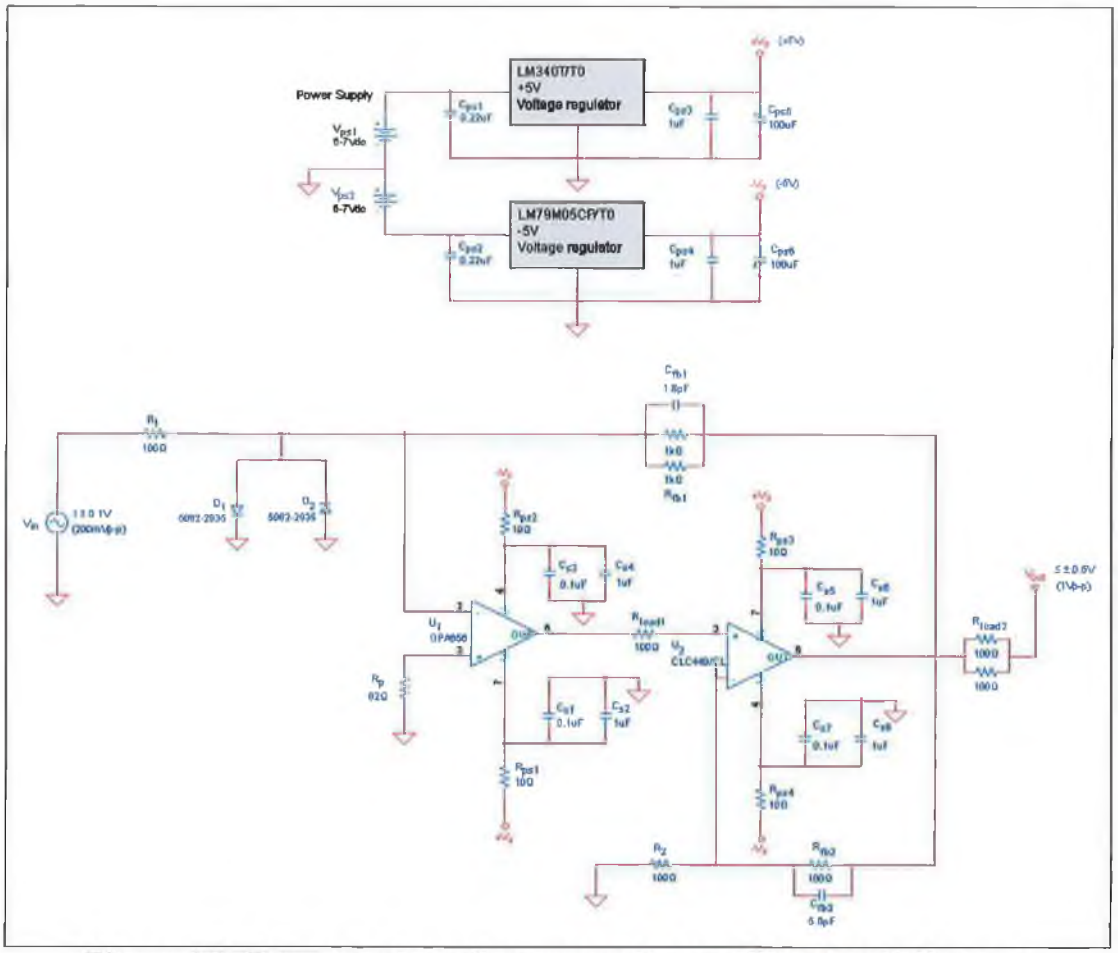


Figure 4.1: Two-stage inverting composite amplifier circuit diagram

4.1.1 Composite Amplifier Assembly

The OPA656 and CLC449 op-amps are intended for operation on ±5V supplies. The fixed output voltage regulators displayed in the upper section of Figure 4.1 will

supply the op-amps accordingly with stable dc levels. These local regulators replicate a real-world situation where a power supply applies a voltage that is insufficiently regulated. Features like internal current-limiting, thermal shutdown and safe-area compensation make these regulators essentially "indestructible" [4-1]. The capacitors C_{ps1} and C_{ps2} are required if the regulators are located far from the power supply. C_{ps3}-C_{ps6} are included for power supply decoupling purposes.

For laboratory testing purposes, the op-amp's source of excitation is provided by a linear power supply (dual tracking dc power supply, model number: GPC-3020) via the voltage regulators. However for many years the world of power supply design has seen a gradual movement away from the use of linear power supplies to the more practical switched mode power supply (SMPS). By employing high switching frequencies, the size of the power transformer and associated filtering components in the SMPS are dramatically reduced in comparison to the linear method. This means an SMPS design can produce very compact and lightweight supplies [4-2]. It is therefore recommended that a suitable SMPS replace the present linear power supply when moving from the laboratory to a purely practical environment.

The leads from the power supply to the input board terminals should be kept as short as possible. Recommended twisted pair techniques were used on all power supply leads to reduce magnetic coupling effects. This effective technique allows signals induced in successive twists to cancel out each other [4-3].

The R_{ps1}-R_{ps4}, 10Ω resistors have two functions. Firstly, to provide a physical separation of an appropriate distance between the regulators and the sensitive op-amp pins, while still allowing the op-amps to be powered via the resistance paths. This improves isolation of the source from the other lines and components on the board. Secondly, they act as electronic dampers to the current loops that could possibly resonate. The tracks of a circuit board have self-inductance properties that could have erroneous effects on a high-performance application. Therefore to counteract this induced voltage and any oscillations, resistors of suitable magnitude are appropriately placed within the circuit. A suitable value of the resistor in the LCR circuit created can be deduced from the following equation:

$$R \geq 2 * \sqrt{L / C} \tag{4.1}$$

The physical track dimensions of a PCB determine the amount of inductance (L) present with typical values ranging from 1-100nH. Considering the extreme value of 100nH and C the capacitor value of 0.1μF, R is calculated to be 2Ω, which was rounded up to 10Ω to overdamp the circuit.

C_{s1}-C_{s8}, are shunt or decoupling capacitors that are used to provide a comparatively low impedance path for alternating currents [4-4]. The 0.1μF and 1μF values match the op-amps manufacturers' recommended values. Such capacitors prevent high frequency ac signals, present at the regulator output lines, from reaching the positive and negative supply pins (V+ and V-) of the two op-amps.

The implemented amplifier system shown in Figure 4.1 requires a signal generator to analyse the composite amplifier's frequency response. The signal generator is shown diagrammatically as an ac voltage source and is labelled V_{in}.

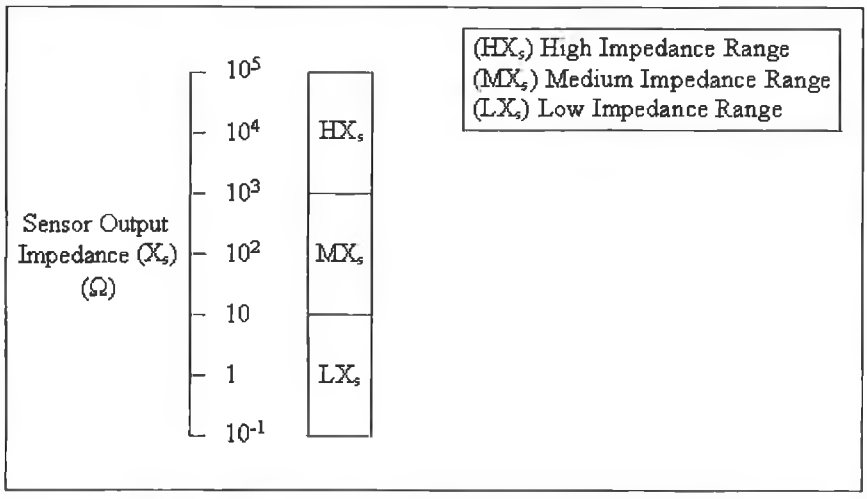


Table 4.1: Sensor output impedance ranges

To get the maximum voltage transfer from a hypothetical sensor to the amplifier's load and successfully interface a sensor with the inverting composite amplifier, the sensor's specific impedance properties need to be initially considered. For example, if a sensor has a low impedance output (X_s), which is in the range indicated in Table 4.1, a negligible effect on the amplifier's gain will be observed. This is a result of all the

R_1 values listed in Table 3.5 being significantly greater than the sensor's output impedance, which relates to a minimum amount of signal loss being experienced across the sensor's internal impedance. Consequently, the maximum voltage transfer from a sensor to the amplifier's R_1 input resistor is achieved.

A situation may arise where a sensor's internal impedance is of a similar order of magnitude as the original R_1 value. In this case, significant loading errors would result if R_1 were not increased. For the maximum voltage transfer to be maintained the amplifier's input impedance (R_1) should be far greater than the sensor's internal impedance (X_s). The process of modifying the amplifier's input impedance until $R_1 >> X_s$ affects not only the loading error but also the system's overall gain. Therefore, R_1 and R_{fb1} resistors are increased by the same factor, typically by a hundred or a thousand, to preserve the inverting $-R_{fb1}/R_1$ gain relationship.

For maximum bandwidth, the front-end op-amp parallel combination of $R_1 \| R_{fb1}$ should always be <200Ω. Therefore, the expected gain may indeed be preserved but only for low frequency. When a sensor's internal impedance is of the order of 100Ω or greater, the ideal option is to operate the composite amplifier in its non-inverting mode.

If the sensor's exact impedance value is known, an alternative method could be employed in which it's internal impedance is combined in series with R_1 and their resultant value treated as the new input load impedance. For example, if X_s is specified at 10Ω and R_1 being equal to 100Ω for the inverting –5V/V amplifier the only external component needing to be modified is R_{fb1} from 500Ω to 550Ω and therefore maintains the required gain level.

Another 100Ω resistor labelled R_{load1} was placed in series between the output pin of the OPA656 op-amp and the non-inverting input pin of the CLC449 op-amp. This ensures that any parasitic capacitive load present along this line has a negligible effect on either op-amp stages. At the output of the CLC449 op-amp two 100Ω resistors where placed in parallel to produce a terminal impedance of 50Ω (labelled R_{load2}).

Anti-parallel Schottky barrier diodes are placed across the front-end amplifier's inverting input pin and ground. The particular clamping diodes used have the manufacturing code of 5082-2835 and are produced by Agilent Technologies. The main reason for their selection was that this diode is in wire type form. This was in contrast with other Agilent Technologies diodes, surface mount device (SMD) package components, that are harder to implement in a breadboard system and are therefore more suited for miniature or PCB designs.

Along the two feedback loops of the composite amplifier, capacitors of suitable magnitude were connected across both resistor networks R_{fb1} and R_{fb2} in parallel form. These feedback capacitors (C_{fb1} and C_{fb2}) were added to reduce any unwanted high frequency noise present down to a negligible level, hence producing an ideal output signal.

4.1.2 Assembly Precautions

Careful attention to breadboard assembly and/or PCB layout is inevitably the difference between a fully functioning system and one that doesn't live up to expectations. The composite amplifier is no exception. Consequently, some relevant assembly precautions are recommended to optimise the system's performance.

- Minimise the distance from the power-supply pins to high frequency 0.1μF decoupling capacitors. Close proximity of analogue signal I/O pins and ground lines should be avoided.
- Remove ground lines underneath the amplifiers' packages.
- Keep resistor leads and board track lengths as short as possible.
- Since the output pins and inverting input pins are most sensitive to parasitic capacitance, always position the feedback resistors as close as possible to the input pins. Isolate these sensitive pins by cutting away any unused lengths of neighbouring tracks.
- Where double-sided component mounting is allowed, place the feedback resistor directly under the package on the other side of the board between the output and inverting input pins [4-5].

- Connections between the op-amps should be made with short direct lines or via the recommended resistors.
- Socketing high-speed devices is not recommended. The additional lead length and pin-to-pin capacitance introduced by the socket can create an extremely troublesome parasitic network that can make it almost impossible to achieve a smooth, stable frequency response. Best results are obtained by soldering the device onto the board. However, if a socket is the only option, flush-mounted socket pins instead of high profile sockets are endorsed. The mini-board type adapters from companies like Aries and Winslow are particularly useful when attempting to convert the OPA656 SOP pin arrangement into a DIL package.

The composite amplifier's performance is concluded to be strongly dependent on tight overall layout, proper resistive termination and adequate power supply decoupling [4-6].

4.1.3 Typical Composite AC Analysis

A suitable signal generator and two oscilloscopes were used to investigate the high-speed frequency response properties of the two-stage typical composite amplifier. A Hewlett Package, 33120A, 15MHz function/arbitrary waveform generator was employed. The periodic waveforms generated by this instrument have a maximum frequency of 15MHz. Despite the high frequency (HF) band limit of 30MHz not being able to be generated by this instrument the HF band range is represented by the 3-15MHz signal range available and is therefore a suitable limit by which to test the amplifier's ac performance. The signal generator acts as a controllable low-level ac source in place of a measured sensor output signal.

Two oscilloscopes were used separately at different stages. The Hewlett Packard, 54504A, digitising oscilloscope, 400MHz (200MSa/s), acted as the primary measurement device. The Tektronix, TDS210, digital real-time oscilloscope, 60MHz (1GSa/s) was used for recording waveforms and transferring data to the PC via an RS232 cable. The oscilloscopes were connected to input and output points of the circuit board with reliable Tektronix P6138 10X probes.

Initially, the HP 54504A oscilloscope was used to validate the two-stage composite amplifier's constancy with respect to its gain over varying frequencies. By viewing input and output waveforms at distinctly differing frequencies its gain of –5V/V was confirmed. For example, a 10kHz very low frequency (VLF) input sinusoidal signal with $200mV_{p\text{-}p}$ amplitude produced a corresponding $1V_{p\text{-}p}$ signal returned from the amplifier. As expected the inverted output had a phase shift of –180° with respect to the original input signal. This change in phase is not crucial for ac based sensor signals, however, if necessary the signal can be re-inverted at the level shifting or digital processing stages.

Figure 4.2 displays the composite amplifier's input and output waveforms as a high frequency 10MHz signal is applied. The tested two-stage composite amplifier maintains the desired ac performance for low frequency signals up into the MHz range.

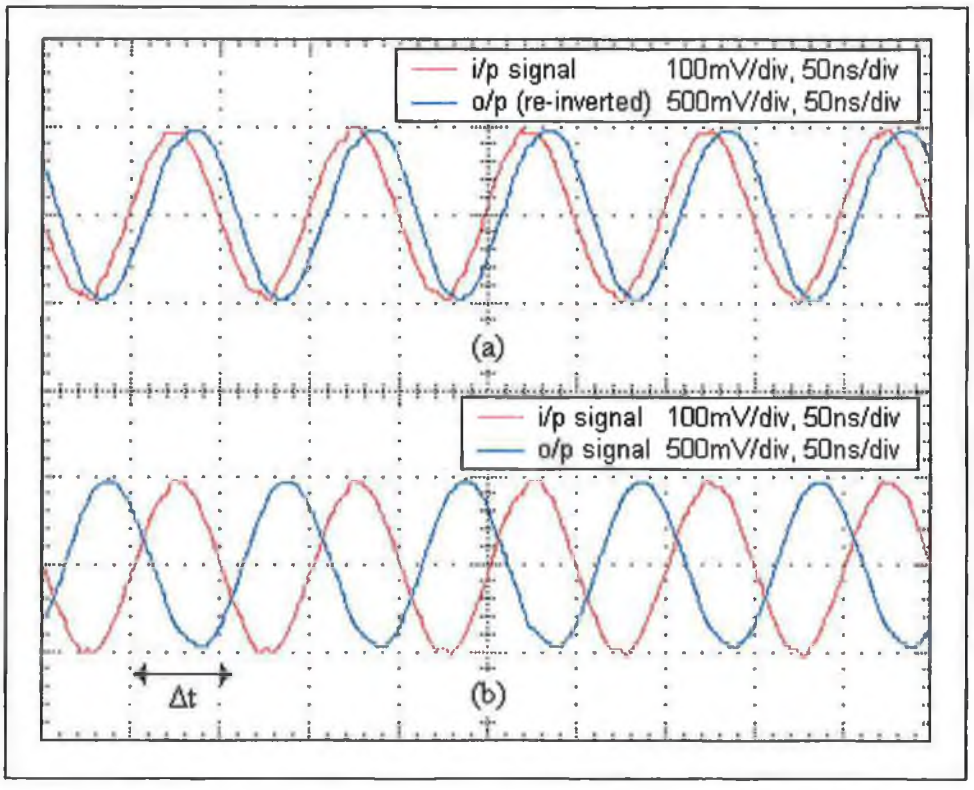


Figure 4.2: Real-time signal analysis of the typical two-stage (–5V/V gain) composite amplifier

The inherent phase shift of –180° was expected when employing the inverting amplifier arrangement. However, at the higher frequencies tested, an additional phase delay was observed. This frequency related phase delay, also shown in Figure 4.2, is a property of the op-amps used and is noticeable at a frequency around 10MHz, at which point it was measured to be approximately –18°. That is, –18° was recorded when the output signal was re-inverted mathematically by the oscilloscope and compared to the input signal. This phase delay between the input and output signals is clearly apparent from Figure 4.2(a). The phase error was measured by expanding out the time/div scaling and by using the following equation to calculate the phase lag as an angle:

$$\boxed{\theta = -360° \cdot \nu \cdot \Delta t} \tag{4.2}$$

The symbol ν in the above equation denotes the input signal's frequency, while the measured time difference between the input and the output signals are given the symbol Δt.

The true and total phase shift is actually: –180° –18° = –198°, which was measured and can also be seen in Figure 4.2(b) where the output signal is displayed in its original form. As expected the phase difference between the input and output signals increased gradually beyond the 10MHz frequency. This error is intolerable in applications requiring high phase accuracy, thus higher feedback capacitance is required to perform greater frequency compensation and hence overcome the shift in phase. Refer to Section 3.4.3 for further details for phase shift compensation.

4.1.4 Typical Composite Precision Results

High-accuracy voltage measurement instruments were selected to test the composite amplifier's dc precision properties. The Thurlby Thandar Instruments (TTI), 1906 computing multi-meter was used primarily to measure the amplifier's input voltage. This multi-meter has the ability to record potential differences in the order of 1μVs and was also used to measure the device's output voltage at room temperature.

As with the ac investigations, the op-amps of the composite design were powered by the same dual tracking dc power supply. Also, note that the input voltage to the amplifier is attached to a separate fixed source that is incorporated within this same power supply. However, the actual voltage levels received at the input to the amplifier are controlled via two potentiometers that act together as a course and a fine adjustment system.

To test the amplifier device even further, its dependence on temperature was examined. The board's dc characteristics were analysed at different temperatures, which were chosen to represent harsher climatic conditions, in which the components are expected to successfully operate. The enclosed apparatus functioning as a miniature temperature controllable environmental chamber was a modified fridge/freezer that can rapidly vary its air temperature from the sub-zero scale up to heated plus forty degree Celsius levels within minutes.

The Philips, PM2521 automatic multi-meter was also required to measure the amplified output voltage when the air temperature was artificially set above or below room temperature. The PM2521 multi-meter has a suitable accuracy down to the 10s of μV.

The high and low air temperatures at which the composite amplifier board was tested were set at 40°C and 0°C respectively. A measured room temperature of approximately 22°C was the third reference temperature used. When the amplifier's circuit board was placed inside the environmental chamber, both multi-meters were used simultaneously to measure the input and output voltage levels accordingly. With this arrangement, the chamber door could remain closed during the measuring process. This arrangement avoided any unnecessary temperature variations that could result if only one multi-meter was used and where disconnecting/reconnecting leads would be involved. All three temperatures were measured independently of the apparatus by a thermocouple probe that was placed in close proximity to the tested circuit board. The resulting temperature was digitally displayed via a Fluke 52 K/J thermometer handheld device.

The selected input voltage range for the inverting (–5V/V gain) composite amplifier represented the expected dc output voltages from a sensor under investigation. This input range, which the amplifier was designed to handle, was extended to include levels that exceeded both the amplifier's linear operating range and also the ADC's ±0.5V input range that would be returned from the amplifier. These saturated results have no bearing on the device's performance as an amplifier, since the linear range covers a sensor's entire low voltage output range. However for completeness, the amplifier's limitations with respect to its own input/output voltage ranges should be known and noted.

V_{in} (V)	Predicted V_{out} (V)	Actual V_{out} (V)	Output Error (mV)	Percentage diff. (%)
-0.800000	4.000000	3.402000	598.000	14.950
-0.700000	3.500000	3.399000	101.000	2.886
-0.600000	3.000000	3.008000	8.000	0.267
-0.500000	2.500000	2.496000	4.000	0.160
-0.400000	2.000000	1.999000	1.000	0.050
-0.300000	1.500000	1.496000	4.000	0.267
-0.200000	1.000000	0.998000	2.000	0.200
-0.100015	0.500075	0.498560	1.515	0.303
-0.090015	0.450075	0.448660	1.415	0.314
-0.080028	0.400140	0.398850	1.290	0.322
-0.070017	0.350085	0.348870	1.215	0.347
-0.060010	0.300050	0.298940	1.110	0.370
-0.050028	0.250140	0.249130	1.010	0.404
-0.040000	0.200000	0.199140	0.860	0.430
-0.030013	0.150065	0.149284	0.781	0.520
-0.020012	0.100060	0.099383	0.677	0.677
-0.010000	0.050000	0.049449	0.551	1.102
0.010023	-0.050115	-0.050403	0.288	0.575
0.020005	-0.100025	-0.100174	0.149	0.149
0.030028	-0.150140	-0.150149	0.009	0.006
0.040028	-0.200140	-0.200030	0.110	0.055
0.050009	-0.250045	-0.249790	0.255	0.102
0.060025	-0.300125	-0.299750	0.375	0.125
0.070014	-0.350070	-0.349540	0.530	0.151
0.080031	-0.400155	-0.399480	0.675	0.169
0.090000	-0.450000	-0.449160	0.840	0.187
0.100023	-0.500115	-0.499000	1.115	0.223
0.200000	-1.000000	-1.004000	4.000	0.400
0.300000	-1.500000	-1.503000	3.000	0.200
0.400000	-2.000000	-1.996000	4.000	0.200
0.500000	-2.500000	-2.508000	8.000	0.320
0.600000	-3.000000	-3.002000	2.000	0.067
0.700000	-3.500000	-3.300000	200.000	5.714
0.800000	-4.000000	-3.300000	700.000	17.500

Table 4.2: Measured and calculated dc precision results for the inverting (–5V/V gain) composite amplifier, in the ambient air temperature of 22°C

By examining the measured dc precision results in Table 4.2, the amplifiers linear and saturated levels are apparent. The predicted output voltage (V_{out}) column in Table 4.2 corresponds to the behaviour predicted by the ideal amplifier model. This behaviour equates to the input voltage being multiplied by the moderate dc gain of –5V/V, that is, V_{out} (predicted) = $-5*V_{in}$. When the predicted and actual output voltages are examined, the linear input range can be seen to exceed magnitudes of ±0.6V. The effects of saturation become significantly apparent shortly beyond this range. At this point, 100s of mV are the calculated differences between the predicted and the actual measured output voltages.

The offset effect of the amplifier is observed when an input signal close to zero is applied, thus an offset error of 0.6mV should be expected due to the properties (listed in Table 3.3 and confirm in Table 4.2) of the front-end precision op-amp.

Another indication of non-linearity can be observed from the percentage difference results, where a percentage difference in the order of a few 10s generally correspond to a saturated region. The equation used to determine the output's percentage difference has the form:

$$\text{Percentage Difference} = \left| \frac{\text{Actual } V_{out} - \text{Predicted } V_{out}}{\text{Predicted } V_{out}} \right| * 100 \qquad (4.3)$$

The difference between actual and predicted results and their percentage differences should both be reviewed in advance of any linear or non-linear ranges for the device being considered.

The ultimate cause of saturation was found to be the op-amps output voltage range limits. Both of the OPA656 and CLC449 op-amp's data sheets output voltage range is stated at ±3.3V. These range limits are confirmed, by examining Table 4.2's actual V_{out} measured results for the two-stage composite amplifier. When the air temperature was adjusted similar test results were produced. Instead of tabulating these high and low air temperature results, all three situations and their ideally predicted results were analysed further using Matlab's graphical application. Matlab's data analysis and

visualisation software allows plotted results to be examined in detail and allows drift patterns to be detected.

The information illustrated in Figure 4.3 shows the voltage transfer curves (VTCs) for the amplifier under the influence of temperature. The positive and negative saturation levels (±3.3V) of the amplifier appear to be almost identical in symmetry when the amplifier's entire measured range is viewed in Figure 4.3(a). From the low resolution scaling of Figure 4.3(a) the amplifier's slope is practically –5V/V. Therefore the gain or sensitivity of the amplifier is as expected approximately –5V/V in its linear region. By expanding the original plot, Figure 4.3(b) shows that the amplifier's offset voltage is temperature dependent.

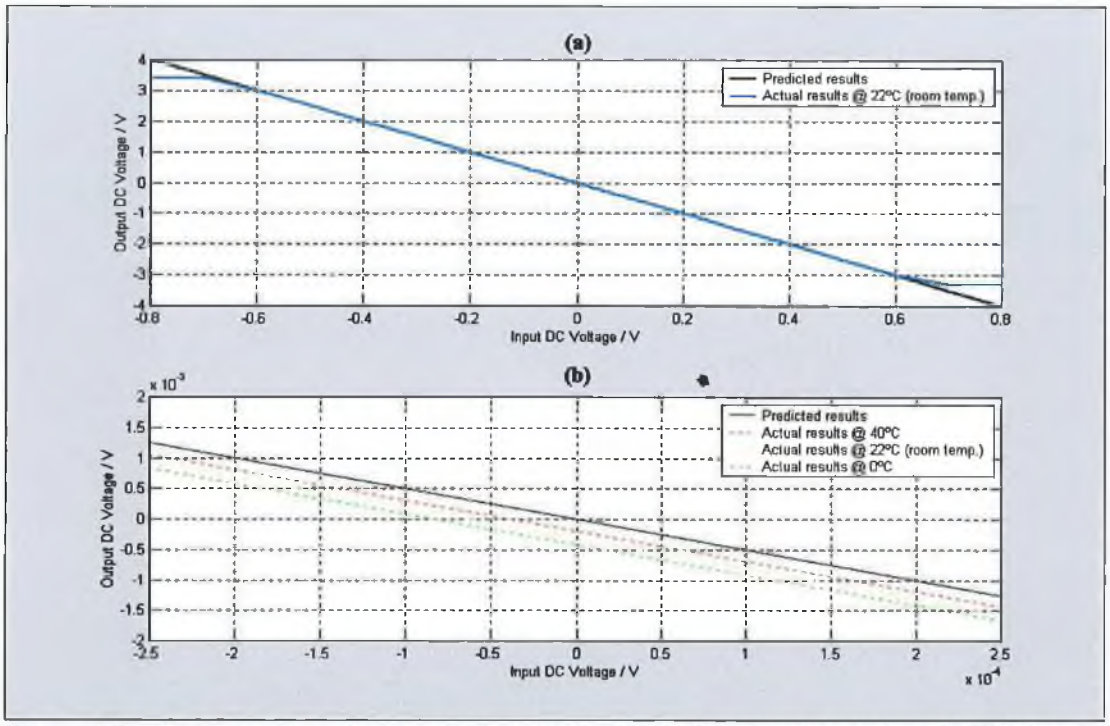


Figure 4.3: Predicted and actual i/p-o/p dc VTC for the inverting (–5V/V gain) composite amplifier: (a) over the entire recorded voltage range and (b) for an expanded section of the linear range

The specific dc-precision properties of the composite amplifier were determined using Matlab's data analysis functions. The polyfit( ) and corrcoef( ) functions are the particular functions used to determine the specific linear regression characteristics of the inverting composite amplifier. The gain and offset values are established from each distinctive data set using the polyfit( ), best straight-line approximation, function.

The corrcoef( ) function produces the degree of uncertainty by determining its correlation coefficient value, associated with a calculated gain.

Set Temperature /°C	Input Offset Voltage /µV	Uncertainty /%	Slope/Gain & Absolute Uncertainty V/V
0	-83.50	0.0159	-4.9869 ± 0.0008
22	-70.44	0.0051	-4.9874 ± 0.0003
40	-37.27	0.0103	-4.9874 ± 0.0005

Table 4.3: Linear regression calculations for the inverting (–5V/V gain) composite amplifier in different ambient air temperature conditions

Table 4.3 contains the tabulated results, which show acceptable levels of precision, for the three resulting gains and their associated uncertainties. The calculated offsets are within the offset parameters stated for the front-end precision OPA656U standard-grade op-amp that was implemented and is therefore within the high-performance amplifier standard. The offset calculations show that their resultant errors will have a negligible effect for most applications. However, it should be noted that the thermal drift characteristics of the amplifier are also evident. The equation associated with these temperature dependent properties is:

$$V_{os}(T) = V_{os}(25°C) + TC(V_{os})_{avg.} * (T - 25°C) \qquad (4.4)$$

$TC(V_{os})_{avg.}$ stands for the temperature coefficient term that is sometimes described as the average offset voltage drift. One can therefore estimate the V_{OS} at a temperature other than 25°C. For example, with $V_{OS}(25°C)$ stated at ±0.25mV and $TC(V_{OS})_{avg.}$ equal to ±2µV/°C for the OPA656U op-amp, $V_{OS}(0°C)$ is typically around –200µV. This calculated value exceeds the actual linear regression result, as is the case for the other temperature situations employed and given in Table 4.3. Consequently, the offset properties of the inverting (–5V/V gain) composite amplifier have been confirmed to perform within the low offset and drift parameters of the OPA656 op-amp.

The additional plots of Figure 4.4 show differences between the predicted and actual sets of data in percentage and absolute value form over the required output voltage

range of ±0.5V. Certain patterns exist from the plots that relate again to the thermal drift properties of the composite amplifier device. The output error values of all three data sets appear to marginally converge in Figure 4.4(b) when the unsaturated output voltage goes most negative. The percentage difference results in Figure 4.4(a) seem to converge when the magnitude of the output voltage is increased in either direction. In general, these plots show variations in the two-stage composite amplifier when the ambient air temperature is altered. Depending upon the application these variations maybe considered significant enough to consider employing temperature compensation techniques to minimise these errors.

When an input voltage of zero was applied the output voltage and errors were recorded but omitted from Figure 4.4, as significant outliers may draw some attention away from the thermal drift patterns illustrated.

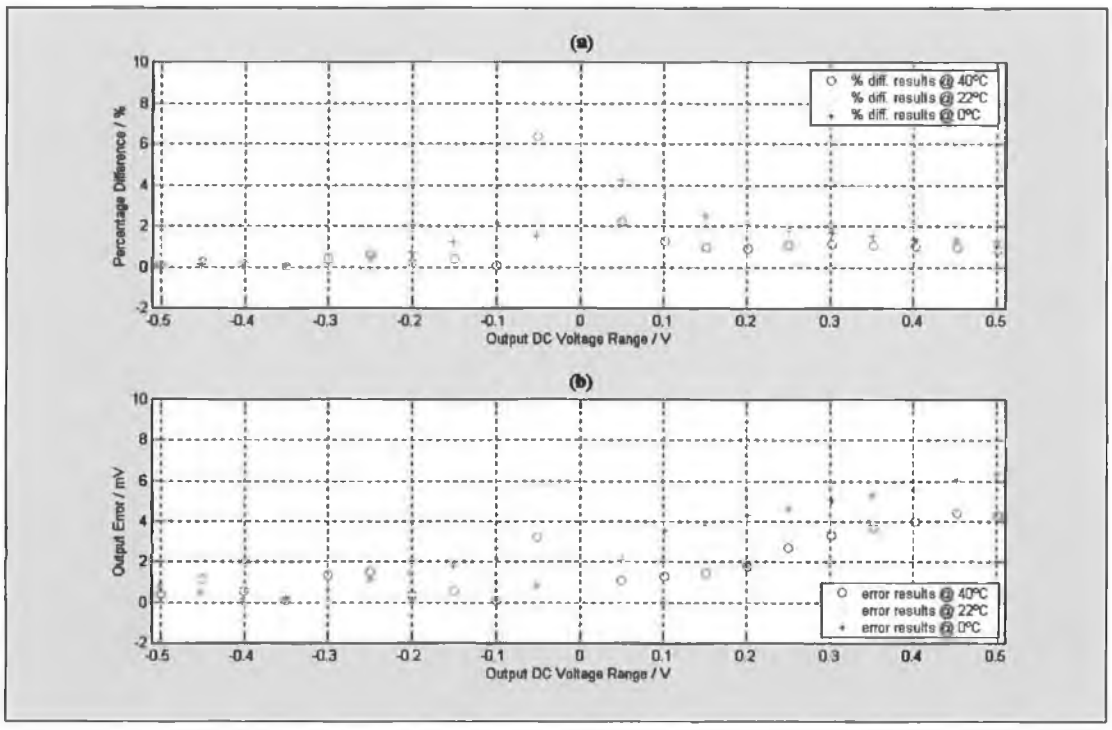


Figure 4.4: Actual dc voltage plots of the inverting (–5V/V gain) composite amplifier in various conditions of temperature producing: (a) percentage difference and (b) error relationships

4.2 Implemented Complex Composite Amplifier

As in the case of the typical composite arrangement, a three-stage inverting composite amplifier was implemented and tested. Unlike the original composite system already described, an additional high-speed CFA is incorporated within the overall feedback loop, as can be seen in Figure 4.5. This extra op-amp is included to increase the system's gain and hence permit a sensor's output signal in the low mV range (2-20mV$_{p-p}$) to be amplified accordingly to cover the complete ADC input range.

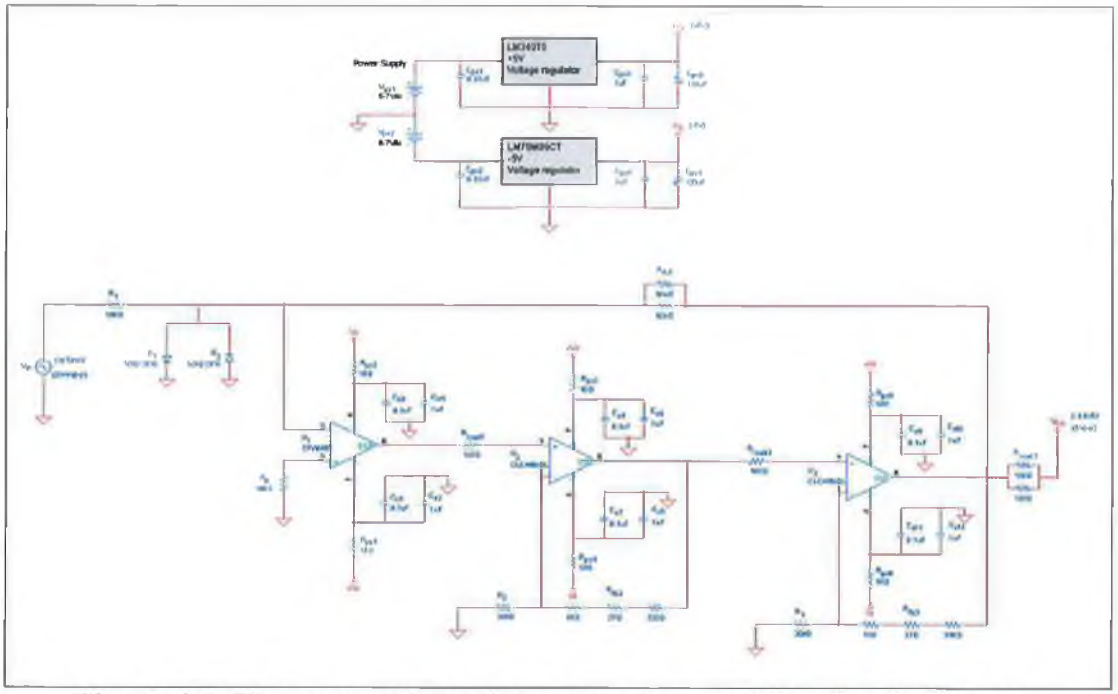


Figure 4.5: Three-stage inverting composite amplifier circuit diagram

The relevant hardware assembly and layout precaution notes of the typical composite amplifier are equally important when developing the more complex three-stage composite design. The overall arrangement is that of an inverting system, an input signal amplitude that is ≤10mV in magnitude (20mV$_{p-p}$) will be amplified to 0.5V magnitude (1V$_{p-p}$) and will be inverted with respect to the input signal. Hence, this particular design, shown in Figure 4.5, displays the specific circuitry required to implement a three-stage composite system with a gain factor set to –50V/V. R_1 is specified at 100Ω, therefore to achieve the system gain of –50V/V the impedance of the feedback resistor labelled R_{fb1} must equal 5kΩ. Two 10kΩ resistors were combined in parallel to form the desired feedback impedance. Series resistors were

used to create the R_{fb2} and R_{fb3} external resistance values for the non-inverting CLC449 op-amps of Figure 4.5.

4.2.1 Complex Composite AC Analysis

The ac-signal analysis set-up described previously for the typical composite amplifier was again employed for the complex composite amplifier. With the same signal generator and digital oscilloscopes connected, the three-stage amplifier's spectrum properties were examined in real-time.

This amplifier was designed primarily for low input signals that are $\leq 20mV_{p-p}$. However, due to signal generator limitations, a $60mV_{p-p}$ sinusoidal signal was used. By simultaneously observing the input to the amplifier and its corresponding output signal over a suitable frequency range, the device's gain and phase properties were found. Throughout the kHz range and beyond to the 10MHz range, the output signals magnitude remained constant at $3V_{p-p}$. Therefore, with the inherent inversion considered, the three-stage composite amplifier's set gain was confirmed to be as expected –50V/V.

The phase property of this device is also reliable, with negligible phase delay being observed in the high frequency band. The input/output waveform relationships at 10MHz are displayed in Figure 4.6. Figure 4.6(a) shows the output signal when re-inverted by the scope and is compared to the input signal. Figure 4.6(b) gives the unmodified input/output relationship with the device's inverting –180° phase shift being apparent. Hence, the waveforms of Figure 4.6 show the amplifier's approximate gain while giving no evidence of any additional phase delay at this frequency. Therefore the three-stage composite amplifier clearly maintains the desired ac properties for high-speed applications and also shows improved phase characteristics when compared to an additional -18° phase shift at 10MHz of the two-stage composite amplifier, where feedback capacitance is also required.

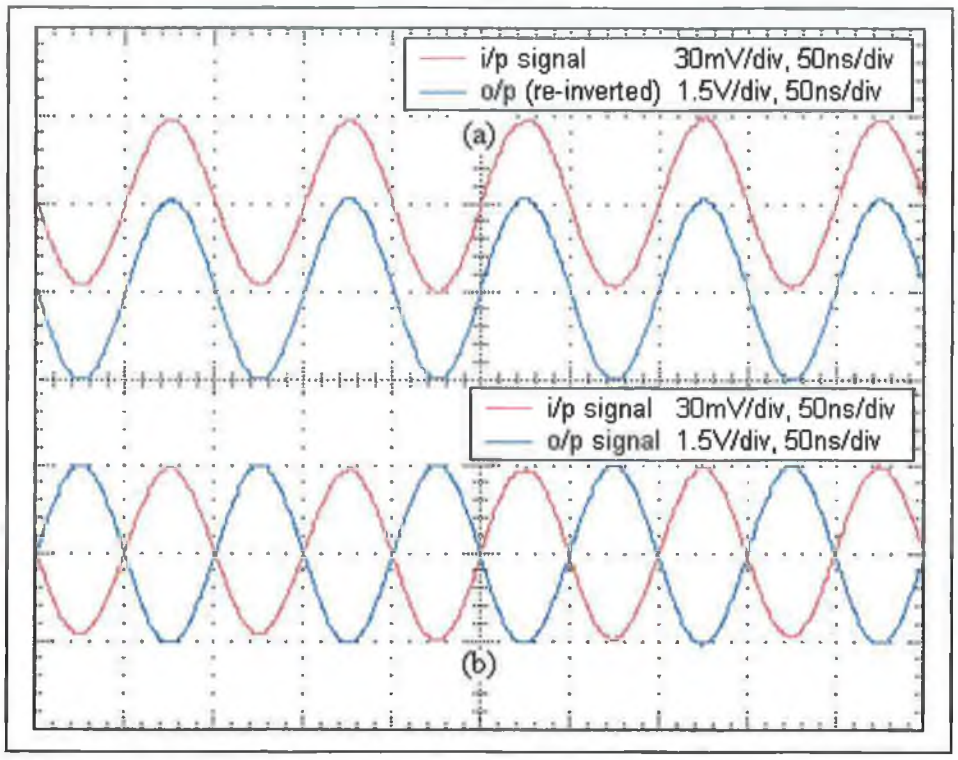


Figure 4.6: Real-time signal analysis of the complex three-stage (–50V/V gain) composite amplifier

4.2.2 Complex Composite Precision Results

The chosen input voltage (V_{in}) range of Table 4.4 was used to test the three-stage composite amplifier's dc gain performance, covering the amplifier's entire linear input/output region and beyond. The expected sensor output voltage, which the amplifier is designed to condition, is well within the amplifier's linear region.

The device's linear input range (V_{in}) extends out to ±80mV that exceeds considerably the required sensor output range of ±10mV. The voltage and percentage differences tabulated data generally increase rapidly to indicate the effects of saturation. However the output errors in Table 4.4 also shows the gradual non-linear effects being introduced when the input voltage is > ±10mV.

V_{in} (V)	Predicted V_{out} (V)	Actual V_{out} (V)	Output Error (mV)	Percentage diff. (%)
-100.000	5.00000	4.047900	952.100	19.042
-90.000	4.50000	4.050400	449.600	9.991
-80.000	4.00000	3.964600	35.400	0.885
-70.000	3.50000	3.473700	26.300	0.752
-60.000	3.00000	2.978100	21.900	0.730
-50.000	2.50000	2.481400	18.600	0.744
-40.000	2.00000	1.984500	15.500	0.775
-30.000	1.50000	1.489840	10.160	0.677
-20.007	1.00035	0.991950	8.400	0.840
-10.010	0.50050	0.495540	4.960	0.991
-9.007	0.45035	0.445630	4.720	1.048
-8.003	0.40015	0.395720	4.430	1.107
-7.003	0.35015	0.346040	4.110	1.174
-6.012	0.30060	0.296840	3.760	1.251
-5.014	0.25070	0.247420	3.280	1.308
-4.010	0.20050	0.197520	2.980	1.486
-3.027	0.15135	0.148684	2.666	1.762
-2.025	0.10125	0.098933	2.317	2.288
-1.000	0.05000	0.048025	1.975	3.950
1.022	-0.05110	-0.052420	1.320	2.583
2.003	-0.10015	-0.101141	0.991	0.990
3.024	-0.15120	-0.151937	0.737	0.487
4.003	-0.20015	-0.200640	0.490	0.245
5.009	-0.25045	-0.250420	0.030	0.012
6.004	-0.30020	-0.300030	0.170	0.057
7.010	-0.35050	-0.349970	0.530	0.151
8.007	-0.40035	-0.399590	0.760	0.190
9.013	-0.45065	-0.449460	1.190	0.264
10.018	-0.50090	-0.499420	1.480	0.296
20.000	-1.00000	-1.006000	6.000	0.600
30.000	-1.50000	-1.497000	3.000	0.200
40.000	-2.00000	-1.994000	6.000	0.300
50.000	-2.50000	-2.486000	14.000	0.560
60.000	-3.00000	-2.982000	18.000	0.600
70.000	-3.50000	-3.478000	22.000	0.629
80.000	-4.00000	-3.965000	35.000	0.875
90.000	-4.50000	-3.979000	521.000	11.578
100.000	-5.00000	-3.979500	1020.500	20.410

Table 4.4: **Measured and calculated dc precision results for the inverting (−50V/V gain) composite amplifier, in the ambient air temperature of 22°C**

Figure 4.7(a) displays the predicted VTC that would result if the amplifier were ideally linear. In addition to this linear gradient line, the actual three-stage composite amplifier VTC at room temperature is shown. The amplifier's output saturation levels of ±4V, initially observed from the tabulated data of Table 4.4, are confirmed from the VTC of Figure 4.7(a). Therefore, as long as the three-stage composite amplifier operates within its expected input voltage range (typically ±1mV to ±10mV), a remarkable agreement between the actual behaviour and the behaviour predicted by

the amplifier's ideal model results. The amplifier's temperature dependent offset voltage relationship is displayed graphically in Figure 4.7(b).

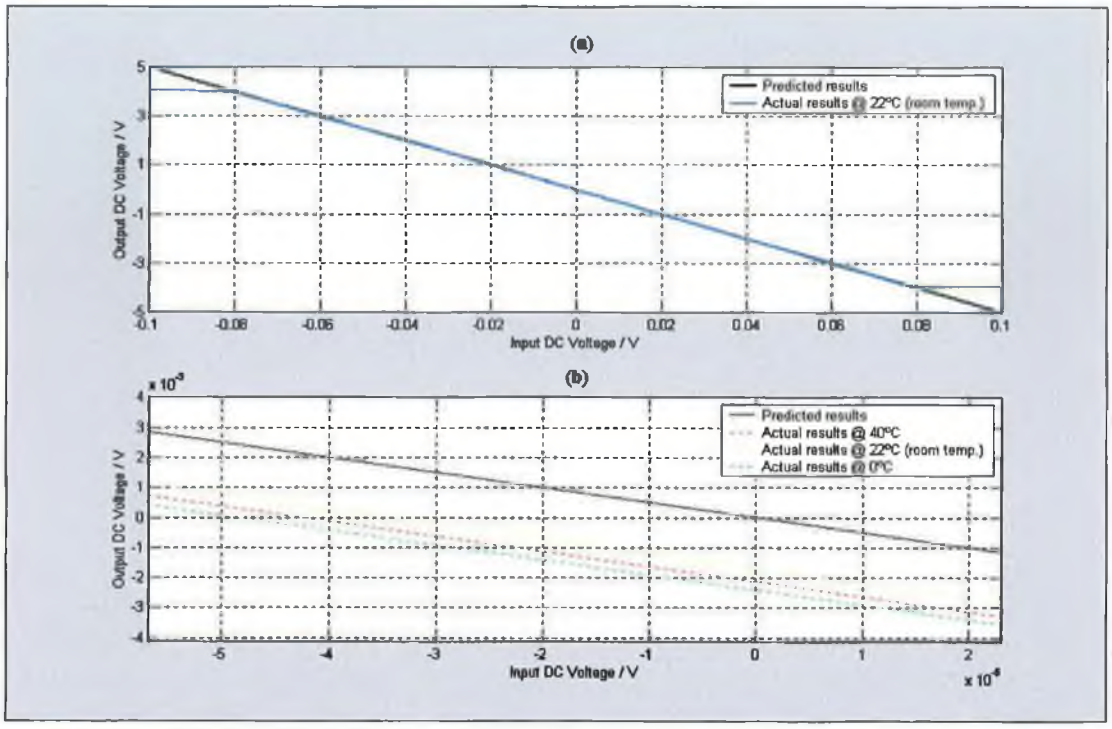


Figure 4.7: Predicted and actual i/p-o/p dc VTC for the inverting (–50V/V) composite amplifier: (a) over the entire recorded voltage range and (b) for an expanded section of the linear range

Set Temperature /°C	Input Offset Voltage /μV	Uncertainty /%	Slope/Gain & Absolute Uncertainty V/V
0	-48.63	0.093	-49.64 ± 0.05
22	-33.89	0.016	-49.68 ± 0.01
40	-42.65	0.169	-49.76 ± 0.08

Table 4.5: Linear regression calculations for the inverting (–50V/V gain) composite amplifier in different ambient air temperature conditions

Table 4.5 lists the calculated offset, gain and associated uncertainty values for each temperature, set to test the amplifier's performance. The calculated offset values of the two and three stage composite amplifiers are of the same order of magnitude. These μV range offsets will have a negligible effect on the three-stage composite amplifier's dc performance when typical input voltage ranges of ±1 to ±10mV are

applied. The amplifier's gain and associated uncertainties given in Table 4.5 are acceptable for precision-based applications, despite a slight decrease in gain occurring as the temperature decreases.

The plots of Figure 4.8 are included to indicate any secondary patterns that exist between the amplifier's predicted and actual measured output voltage over its required operational range. For example, the percentage differences shown in Figure 4.8(a) appear to decrease almost exponentially as the output voltage increases linearly. These differences also drift with temperature and therefore give a insight into the amplifier's thermal behaviour. However, the pattern that results is less regular or distinctive when compared to the corresponding plot (Figure 4.4(a)) of the −5V/V gain composite amplifier. The output error relationships of Figure 4.8(b) are generally seen to increase gradually as the output voltage becomes more positive.

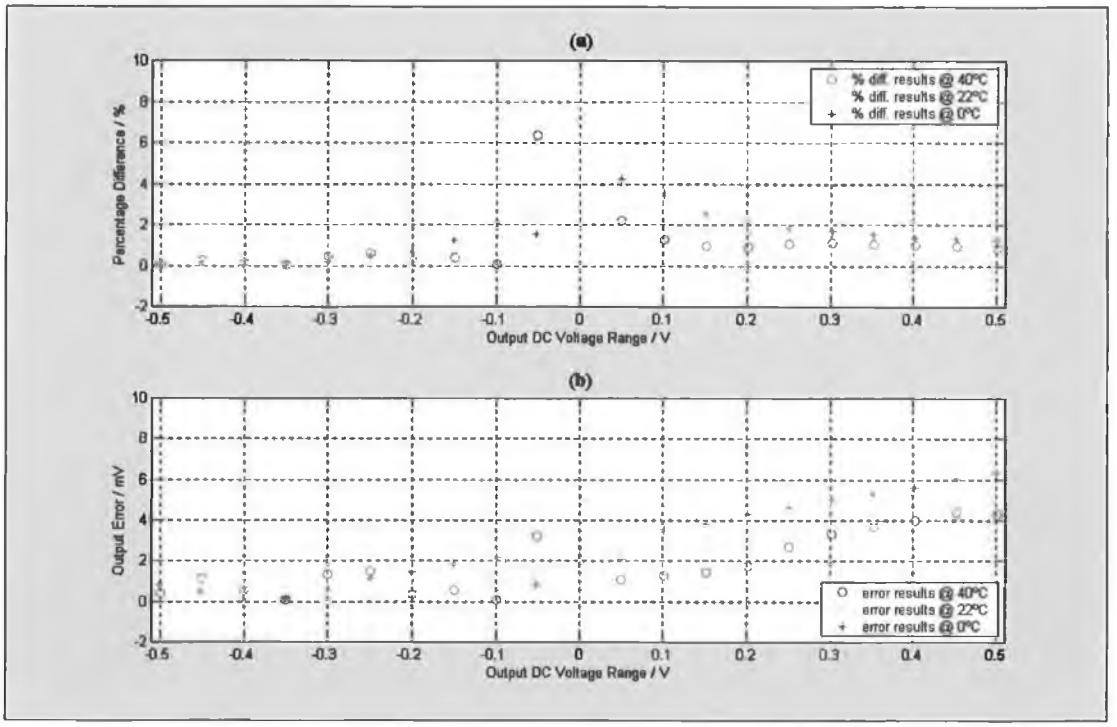


Figure 4.8: **Actual dc voltage plots of the inverting (–50V/V gain) composite amplifier in various conditions of temperature producing: (a) percentage difference and (b) error relationships**

4.3 Summary

It is becoming increasingly advantageous to carry out information processing and control functions using digital methods. However, to get to this stage of processing the data from the real world that is typically in analogue form requires some initial manipulation and conversion. Front-end amplifier circuits form essential sensor interfaces that condition raw data into suitable output signals that match the input range of a connected ADC.

Implementing a high-performance preconditioning amplifier system requires specific layout and hardware assembly issues to be followed. Tight overall layout, proper resistive termination and adequate power supply decoupling, ensures that the composite amplifier system's high-performance capability is achieved.

The high-speed properties of the implemented composite amplifiers were confirmed by viewing its input and output signal waveforms over a suitable frequency range. When analysed the implemented composite amplifiers maintained the desired ac performance for low frequency signals up into the MHz range. Hence, the aim of amplifying a low-range voltage signal up to the appropriate $1V_{p-p}$ level was comfortably achieved by both composite designs.

It was found that negligible shift in phase was observed in the high frequency band for the three-stage composite amplifier, while a marginal phase error resulted from the two-stage composite amplifier.

The composite amplifier's dc precision characteristics were analysed under a range of temperature conditions, where the amplifier's components are expected to successfully operate. The device's recorded input and output dc data were examined through Matlab software. Using linear regression techniques, the offset voltages were calculated to be within the offset parameters stated in the precision OPA656's front-end op-amp data sheets and therefore are of high-performance standard.

These results also show that the front-end precision OPA656 op-amp dominates the dc static parameters of the composite amplifier. More specifically, the input offset-drift characteristics of the OPA656 VFA are reflected by these findings and the poorer input dc characteristics of the CLC449 CFA(s) become insignificant since the CFA(s) are operated within the feedback loop of the VFA.

To conclude, this chapter contains the elements necessary to interface a sensor-based system with an ADC. The recommended devices and hardware assembly issues discussed in this chapter form a basis and a guideline to developing a high-performance amplifier for a data acquisition system. The developed preconditioning amplifier is capable of facilitating a range of output sensor signals; thus ensuring that a broad application base potential is achievable.

5

Data Conversion and Acquisition Systems

The physical world is inherently analogue, indicating that there will always be a need for analogue circuitry to condition physical signals such as those associated with transducers, as well as to convert information from analogue to digital form for processing and from digital back to analogue for reuse in the physical world [5-1]. The instrumentation involved in converting signals for processing in the digital domain generally starts with an op-amp based amplifier system. The signal conditioning elements, to which amplified signals are subsequently applied, include a difference/differential amplifier, an ADC and a data bus interface buffer.

5.1 Data Converter Circuitry

Up to this point, in the thesis, the issue of amplifying precisely a sensor's high-speed output signal to match an ADC's standard input voltage swing of ±0.5V has been discussed. However level shifting is a further conditioning requirement, which is generally involved, to ensure the amplifier's output signal and the ADC's input voltage ranges correspond. The process of dc level shifting the amplified signal is of particular importance, since the mid-scale analogue input value of an ADC is generally at a non-zero level. The Analog Devices AD9050 10-bit, 40MSPS ADC's analogue input is centred at 3.3V for example, while the AD6645 14-bit, 80MSPS device has a lower centre reference value of 2.4V. Therefore, the selected level shifter is generally used to control the appropriate offset from ground to drive the chosen ADC.

When attempting to verify the performance of an ADC, a DAC allows analogue input test signals and reconstructed digital output data to be compared in the same analogue form. Subsequently, with the use of an interface buffer, the ADC-DAC shared data lines could potentially form a bi-directional data acquisition and data generation system.

This section introduces, demonstrates and tests the level shifting, analogue to digital and the signal recovery circuitry relevant for high-speed applications. Assembling a high-performance data acquisition system involves the selection of appropriate level shifting and data conversion devices, which complement the analogue preconditioning circuitry, with high-speed and dc precision properties.

The circuit diagram of Figure 5.1 displays all of the data converter devices that were implemented on a single test board. This platform allows each device to be easily evaluated. The ADC is the pivotal device of Figure 5.1, while the difference amplifier and DAC shown were only selected to accommodate the ADC's analogue input and digital output signals respectively.

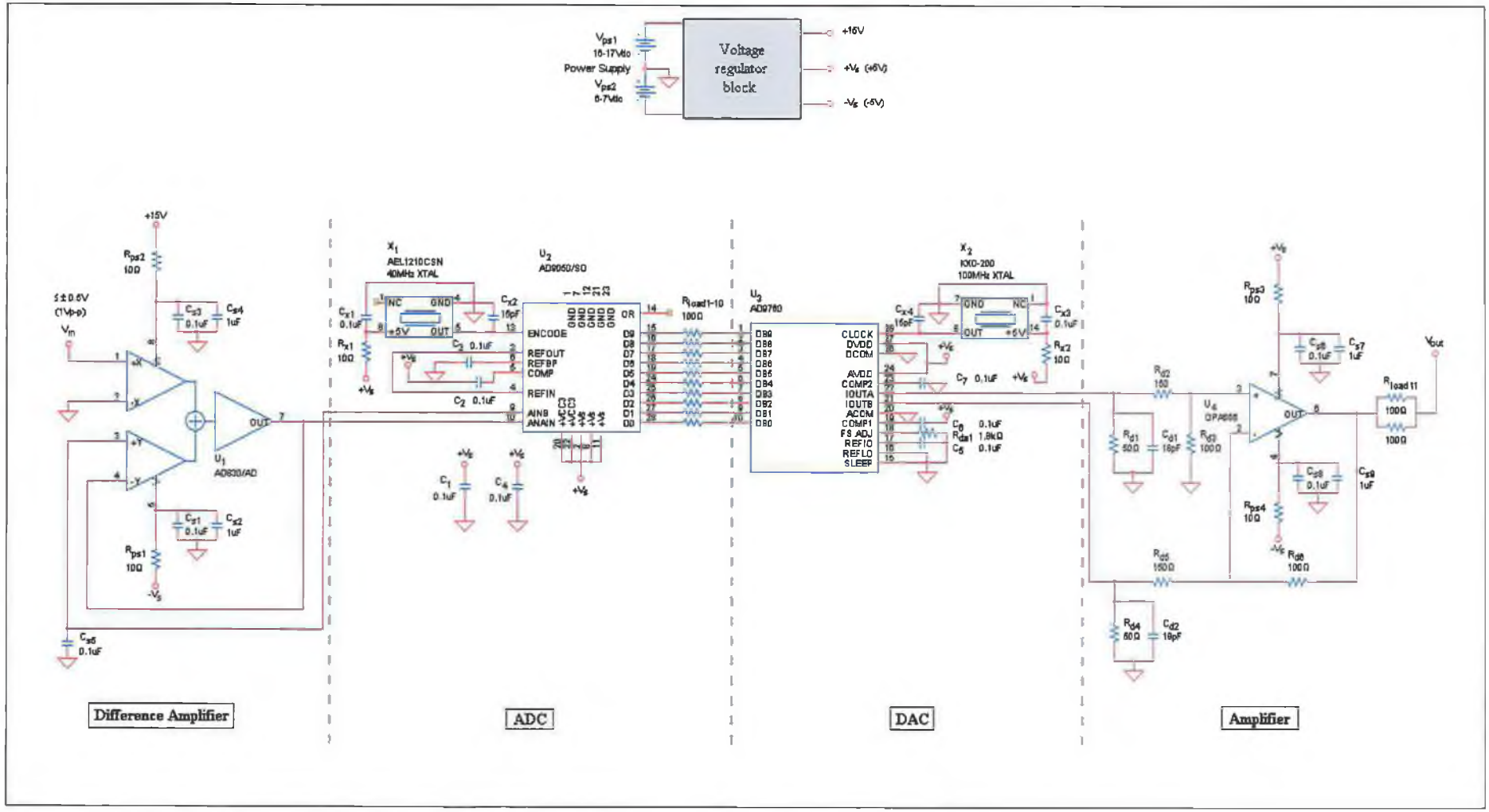


Figure 5.1: Level shifting and data converter circuit diagram

5.1.1 Difference Amplifier Design

The high-speed AD830 difference amplifier displayed in Figure 5.1 provides a unique method of providing dc level shifting for the analogue input. Its input impedance and common-mode rejection ratio (CMRR) are the same for each input connection. These features offered by the AD830's topology are very advantageous and are unlike a voltage or current feedback amplifier, where there is a distinct difference in performance between the inverting and non-inverting gain [5-2]. Using the AD830 allows a great deal of flexibility for adjusting the signal's offset voltage. Hence, the AD830 is a capable intermediate interface between a gain amplifier and the AD9050 ADC.

Prior to implementing the difference amplifier, its level shifting ability was confirmed through PSpice simulations. Figure 5.2 shows the signal analysis waveforms of the AD830 when set-up for interfacing with the AD9050 ADC. The composite amplifier's output voltage is therefore dc shifted up by 3.3V. The waveform, which resulted, was re-inverted to allow phase comparisons to be made between itself and the original sensor output signal (input to composite amplifier). This simulation was performed with an input high frequency signal set at 10MHz. An approximate phase delay of -18° was observed that could affect the performance of a phase based measurement system.

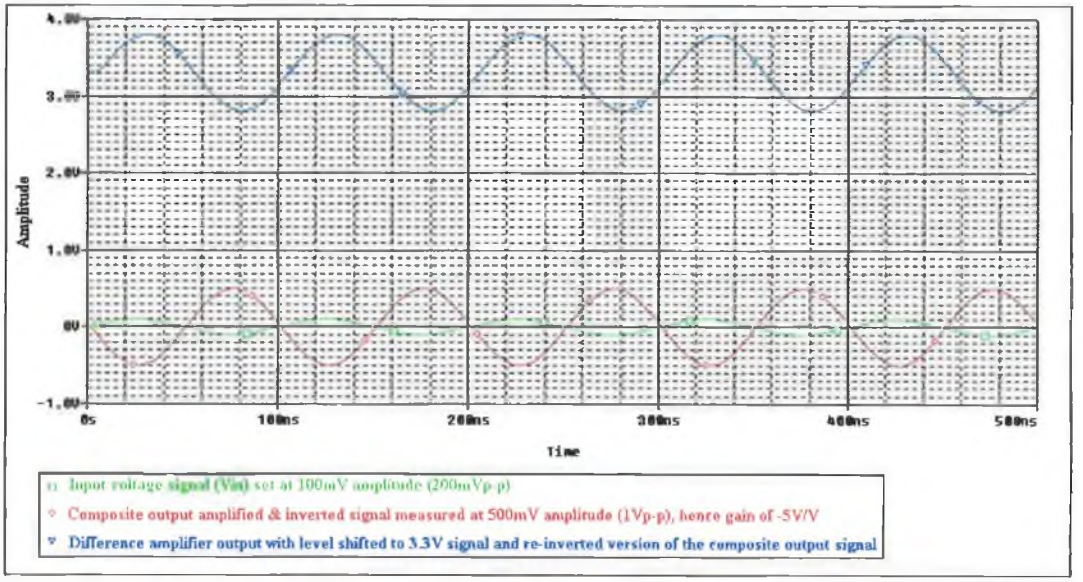


Figure 5.2: Simulated signal analysis of the difference amplifier with an input signal frequency at 10MHz

The dc shifted signal shown in Figure 5.2 was re-inverted by connecting the amplified signal to one of the inverting input pins (pin 2) of the difference amplifier. However, after some physical testing was completed, it was decided to connect the amplified signal line to one of the non-inverting input pins (pin 1) instead, while pin 2 would go to ground. This adjustment between the modelled and implemented designs was made in an effort to counteract any unwanted noise on the second non-inverting 3.3V-line. This modification had the desired effect with a significantly less noisy output signal resulting. The required 3.3V-dc level is conveniently supplied by the AINB pin (pin 9) on the ADC. The AD830 8-pin plastic mini-DIL package was physically mounted on the circuit board by using a suitable 8-way DIL IC socket.

Similar isolation, bypassing and loading techniques used for the composite amplifier designs were implemented to improve the difference amplifier physical performance. Referring to Figure 5.1, the resistor values of R_{ps1} and R_{ps2} act as electronic dampers in the current loops that could possibly resonate. The shunt capacitors, C_{s1}-C_{s5}, on the power supply and input lines of the difference amplifier provide comparatively low impedance ground paths for unwanted alternating currents.

The power supplied to the difference amplifier and the other active devices in Figure 5.1 are controlled via the voltage regulator block shown in the upper portion of this circuit diagram. The fixed regulators that form this block were positioned in close proximity to each other on the board. This arrangement allows a primary ground track to be established. The bypass components of the difference amplifier were appropriately connected to the ground reference line. This direct grounding method prevented ground loops being created around the difference amplifier.

5.1.2 Difference Amplifier AC Analysis

The sinusoidal waveforms of Figure 5.3 demonstrate the actual physical behaviour of the difference amplifier's input and output signals at the high frequency of 10MHz. For the purpose of demonstrating the device's phase delay properties, the output signal's dc shift level of 3.3V is set to the same centre line used by the input signal. When the display's scaling was expanded, a phase shift of -21.6° was calculated. This result compares closely with the simulated phase delay of -18°. The slight difference in these results could be due to the influence of any capacitive loading and/or layout parasitic effects present. Therefore the device can be concluded to be functioning close to its specifications with the ability to level shift the amplified signal, while introducing a marginal phase delay when the input signal's frequency response is within the high frequency band. Thus, for applications with a minimum phase shift requirement an alternative difference amplifier maybe required.

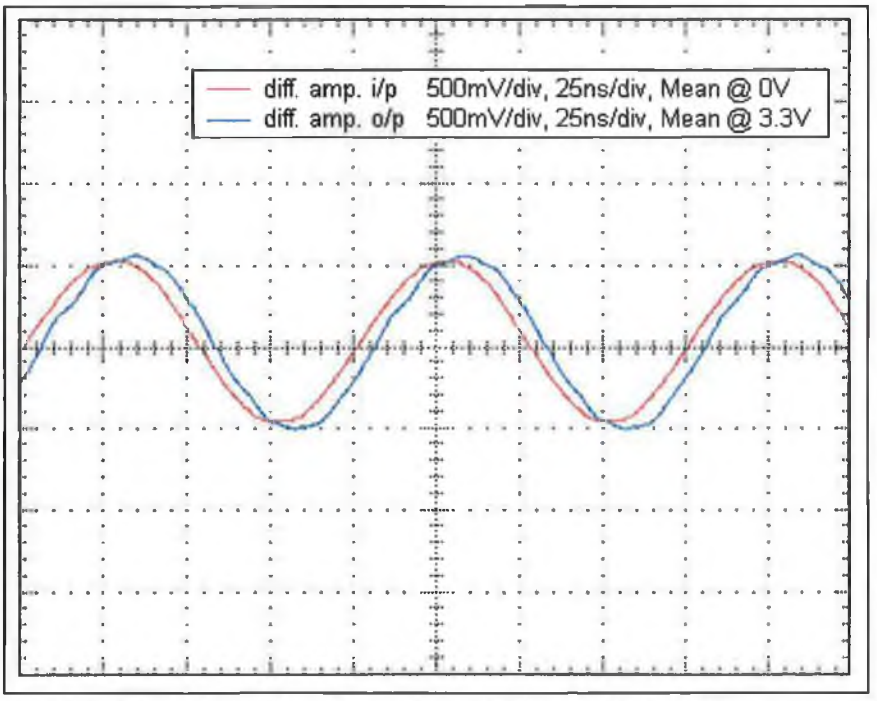


Figure 5.3: Real-time signal analysis of the AD830 difference amplifier

5.1.3 Difference Amplifier DC Precision

The AD830 difference amplifier and its driven counterpart, the AD9050 ADC were tested simultaneously to determine their relative precision. Their results however are discussed separately; the ADC's dc precision findings are addressed in Section 5.1.6 while this sub-section reveals the difference amplifier dc precision details.

Checking the ADC's precise properties involves measuring and recording multiple voltage lines concurrently. These analogue and digital signal levels were measured under specific temperature conditions. This was performed in an effort to see whether the device's dependence, with respect to temperature, would in fact emerge.

The environmental chamber door could not remain closed throughout the measuring process. This unfortunate consequence was caused by the multi-meter's (Fluke 10) probe position requiring adjustment to read all the relevant voltage lines. However to maintain some kind of constant air temperature within the apparatus, the set temperatures were reduced to 10°C and 30°C respectively. These two air temperatures and the room temperature (22°C) still make it possible to observe the devices temperature-dependent properties.

The AD830 difference amplifier was initially placed in an ambient air temperature of 22°C, as the device's level shifting dc properties were determined. The results of applying accurate dc voltage signals within an appropriate range of ±0.5V, which complements the expected amplifier's output range, are shown in Table 5.1. The actual recorded output voltage values differ from the predicted output values. The AD830 difference amplifier has four input signals: the dc test signal is applied to pin 1, pin 2 is connected to ground, the ADC's 3.3V-dc level shifting signal is applied to pin 3 and the output signal is connected to pin 4 to form a unity gain voltage follower arrangement. Each of these input signals as well as the device's own gain and offset properties are potential contributors of the +10 to −1mV determined output error swing.

V_{in} (V)	Predicted V_{out} (V)	Actual V_{out} (V)	Output Error (mV)	Percentage diff. (%)
-0.500	2.800	2.790	10.0	0.357
-0.450	2.850	2.841	9.0	0.316
-0.400	2.900	2.890	10.0	0.345
-0.350	2.950	2.943	7.0	0.237
-0.300	3.000	2.992	8.0	0.267
-0.250	3.050	3.045	5.0	0.164
-0.200	3.100	3.096	4.0	0.129
-0.150	3.150	3.146	4.0	0.127
-0.100	3.200	3.196	4.0	0.125
-0.050	3.250	3.246	4.0	0.123
0.000	3.300	3.295	5.0	0.152
0.050	3.350	3.345	5.0	0.149
0.100	3.400	3.397	3.0	0.088
0.150	3.450	3.447	3.0	0.087
0.200	3.500	3.497	3.0	0.086
0.250	3.550	3.549	1.0	0.028
0.300	3.600	3.599	1.0	0.028
0.350	3.650	3.649	1.0	0.027
0.400	3.700	3.700	0.0	0.000
0.450	3.750	3.750	0.0	0.000
0.500	3.800	3.801	1.0	0.026

Table 5.1: Measured and calculated dc precision results for the AD830 difference amplifier, in the ambient air temperature of 22°C

The assumption that the voltage applied to the other non-inverting input (pin 3) of the AD830 devices is constantly set at 3.3V is actually incorrect. This source of error, which originates from the AINB pin (pin 9) of the AD9050 ADC, directly affects the AD830's output voltage. Instead of being fixed at 3.3V, this line varies around this expected value by as much as ±5mV as it is subject to unwanted noise and onboard parasitics.

The device's input offset voltage (V_{os}) was independently tested by, initially, short circuiting all its input pins to ground, therefore removing any contributing sources of error. A suitable voltage was then applied between the first non-inverting input (pin 1) and ground to force the output voltage of the difference amplifier to zero. The measured offset voltage of –2mV, at room temperature, is within the maximum offset voltage of ±5mV specified in the AD830's data sheets. The device's specified offset voltage of ±5mV, combined with the 3.3V ±5mV level shifting input line property explains the measured variation between predicted and actual output voltages.

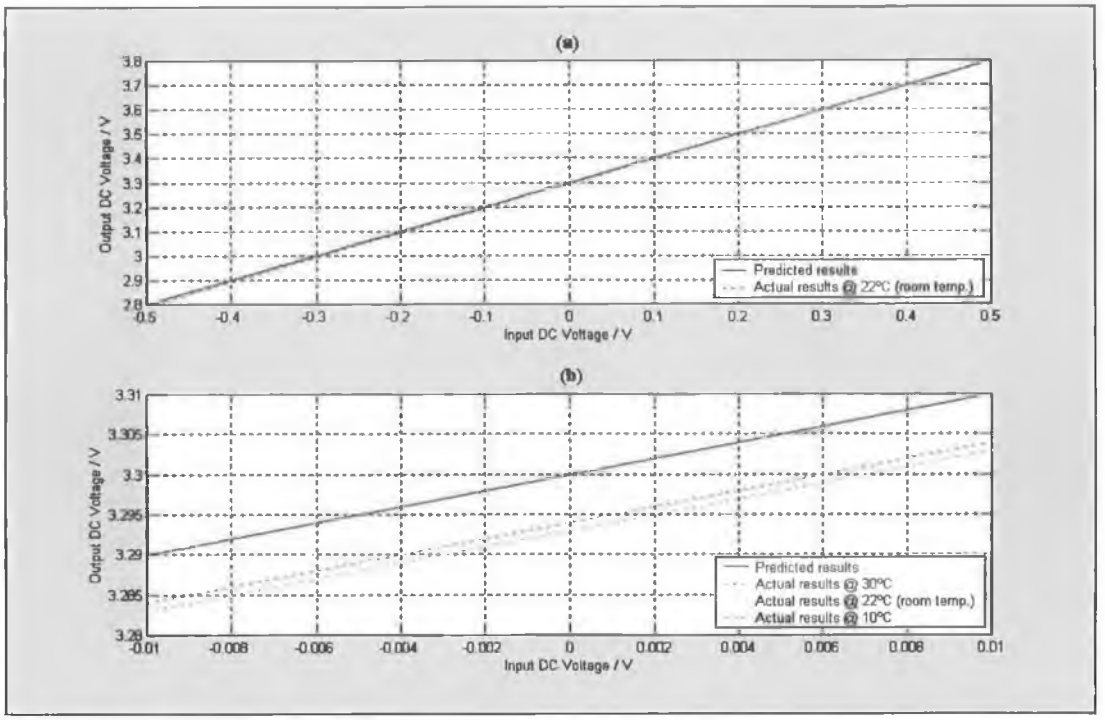


Figure 5.4: **Predicted and actual i/p-o/p dc VTC for the AD830 difference amplifier: (a) over the entire recorded voltage range and (b) for an expanded section near the centre of the range**

Table 5.1's predicted and actual sets of data are shown in graphical form in Figure 5.4(a). The devices additional temperature-dependent voltage curves are included in the plots of Figure 5.4(b). It appears from the plot of Figure 5.4(a) that the straight line associated with an ideal dc level shifting device, with an intercept value of exactly 3.3V, almost identically reflects the actual device's VTC when measured at room temperature. However when a closer look is taken, the difference amplifier's dc performance is more appropriately described as suitable though not ideal.

Figure 5.4(b) shows an expanded section of the device's VTC. Offset values are observed between the predicted curve and each of the actual transfer curves of the difference amplifier in the respective temperature conditions. The room temperature data set produces a transfer curve that has the closest resemblance to the ideal transfer curve. With the ambient air temperature adjusted above or below room temperature, the AD830's VTC results produce similar offset values with respect to each other. The two offset results, caused by adjusting the device's ambient air temperature, are greater in magnitude when referred to the room temperature offset result that is itself a marginal shift from the ideally predicted zero offset situation.

The AD830's offsets and other relevant dc characteristics are all listed in Table 5.2. These values were calculated by using linear regression techniques, with the dc level shift values relating to their intercept points when the input voltage is zero. The observed properties from Figure 5.4(b) are confirmed by these tabulated results, where the device's room temperature shifted level of 3.297V is the nearest calculated value to the ideal 3.3V level. The input offset voltages of Table 5.2 were calculated by subtracting the predicted ideal value of 3.3V from each dc-shifted result.

Set Temperature /°C	DC Level Shift /V	Input Offset Voltage /mV	Uncertainty /%	Slope/Gain & Absolute Uncertainty V/V
10	3.294	-6.4	0.076	1.0119 ± 0.0008
22	3.297	-3.3	0.063	1.0118 ± 0.0006
30	3.294	-5.8	0.066	1.0120 ± 0.0007

Table 5.2: Linear regression calculations for the AD830 difference amplifier in different ambient air temperature conditions

The composite amplifiers exclusively perform the amplification or gain requirements of the preconditioning system. Therefore, the difference amplifier should have an associated "gain" of 1V/V. The regressed findings show gains that are exceptionally close to the required level, with their respective uncertainties also available. In conclusion, the AD830's dc precision and high-speed features make it suitable for most applications. If however the exceptional ac and dc performance properties achieved by the composite amplifiers are not to be degraded in anyway, the AD830 difference amplifier may need to be either replaced or compensated. The second option is generally preferred and is discussed further with reference to the ADC's dc precision findings and calibration technique in Section 5.1.6.

5.1.4 Analogue to Digital Converters

Whenever attempting to select a suitable ADC for a given application, the factors generally considered are: (a) precision, (b) speed, (c) accuracy, (d) required supply voltages and power dissipation, (e) package type, (f) reference and clock (internally or externally supplied), (g) input impedance and analogue voltage range, (h) input

structure, (i) output structure (parallel or serial and if parallel is it "microprocessor-compatible"), and of course, (j) price [5-3]. The order in which these factors were prioritised was critical in selecting a suitable ADC.

After examining a number of ADCs available from Analog Devices and Texas Instruments, the Analog Devices AD6645 14-bit, 80MSPS ADC was initially identified as a useful example for high-speed applications. With even higher sampling rates, the AD9288 Analog Devices 100MSPS ADC, is another interesting option, however this device's resolution is reduced to 8-bits. The AD6645 is a high-performance, monolithic 14-bit ADC and costs approximately €50 for quantities of 100-499 PQFP parts. All necessary functions, including track-and-hold (T/H) and voltage reference, are included on the chip to provide a complete conversion solution [5-4]. The device's high sampling, switching and digital specifications make it an ideal choice for a data acquisition system where package type and price are of low priority.

Figure 5.1 shows the AD9050 ADC as a central component of the high-speed preconditioning system. This ADC's SOIC package is relatively easy to incorporate and became the overriding factor for its selection over other ADCs with similar sampling rates. Precision was compromised to accommodate the small package that requires the minimum amount of soldering or layout tools when testing the devices ac and dc properties. The AD9050 is competitively priced at approximately €10 and is therefore the more cost-effective solution when compared to the higher precision options.

The AD9050 is described as a complete 10-bit monolithic sampling ADC with onboard T/H and reference features. The device was designed for low cost, high-performance applications, requires a +5V supply and an external clock to achieve 40MSPS or 60MSPS sample rates with 10-bit resolution [5-5].

5.1.5 ADC AC Analysis

Important guidelines that optimise the AD9050's performance are:

- separate power lines and supplies for the analogue and digital sections
- remove all ground and power lines underneath the device
- use decoupling capacitors flush against the device's pins when required
- the analogue input sensitive pins (which can be driven single-ended or differentially) need to be isolated along with the crystal's pins
- SOIC to DIL adapter is advised for implementing the ADC on a test board, a suitable example is the W9508 mini-board from Winslow
- the Encode pin is connected to a 40MHz crystal that produces pure sine waves
- 100Ω series resistors are connected between the digital output pins and the receiving pins of the DAC, these reduce transients and improve the signal to noise ratio (SNR)

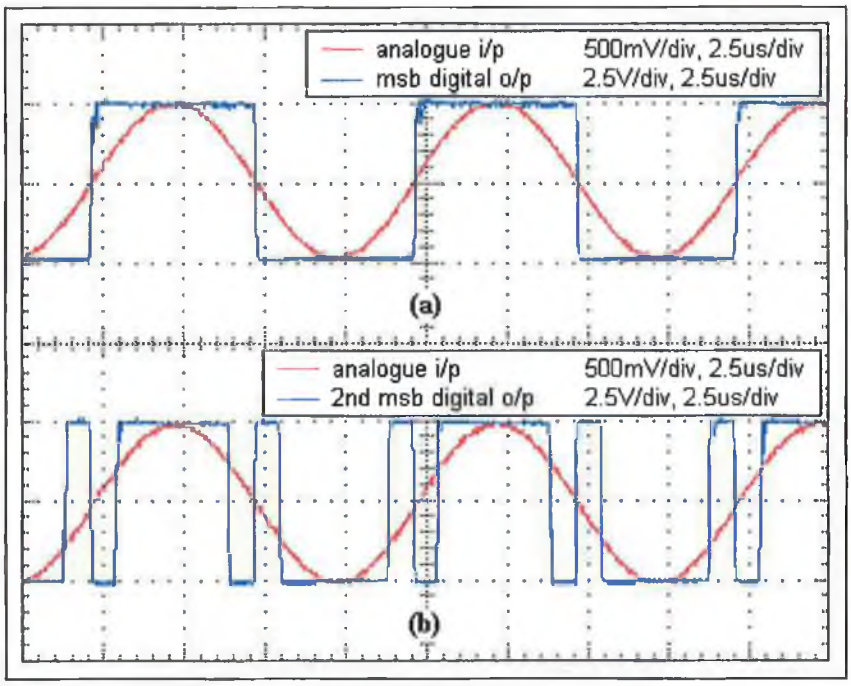


Figure 5.5: Real-time signal analysis with the ADC's two MSB illustrated

Figures 5.5, 5.6 and 5.7 include analogue input testing signals that are sinusoidal in form and have appropriate amplitudes of $1V_{p-p}$ (±0.5V). These signals correspond to the amplified inputs of the difference amplifier. They reach the ADC at pin 10 (AIN) after the 3.3V dc shift has been applied. The continuous input signals of Figures 5.5

and 5.6 were set at a relatively low frequency of 100kHz. These input signals are sampled by the AD9050. The ADC's 1st, 2nd, 3rd and 4th MSB with their respective switching characteristics are superimposed on the analogue input signals given sequentially in Figures 5.5 and 5.6.

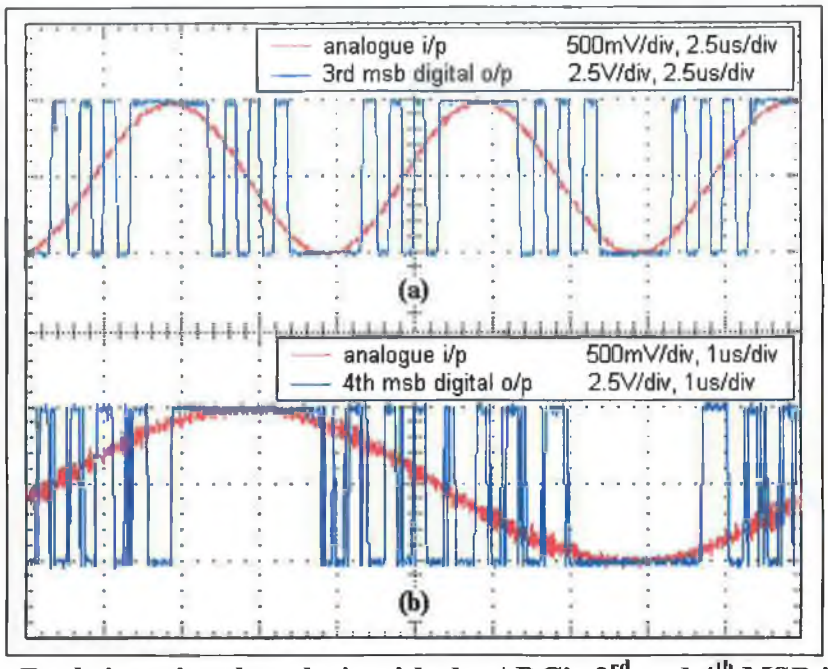


Figure 5.6: Real-time signal analysis with the ADC's 3rd and 4th MSB illustrated

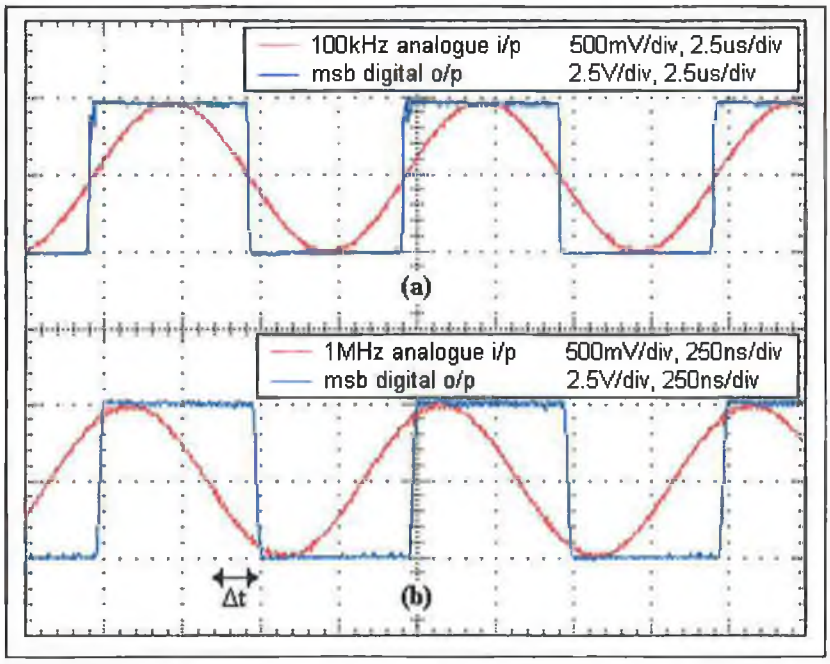


Figure 5.7: Real-time signal analysis with 100kHz and 1MHz frequencies applied and the ADC's respective MSB also illustrated

Figure 5.7(b) gives the resulting waveform of the MSB digital output when the analogue-input frequency signal was increased to 1MHz. This figure illustrates the considerable phase delay of approximately –50° introduced by the ADC and difference amplifier that was negligible when the 100kHz signal was initially applied. At this higher frequency, the ADC caused the majority of the phase lag itself. To avoid such an undesirable effect, an alternative ADC could replace the AD9050. For example, the 10-bit, 100MSPS AD9071 ADC, which is available from Analog Devices, is an ideal candidate since it has the identical package and compatible pin-out configuration as the AD9050.

5.1.6 ADC DC Precision and Calibration

When investigating an ADC dc performance certain terms and their specified limits need to be mentioned. Specifically, the AD9050's data sheets specify both the maximum differential and integral non-linearity values to be 1.75 LSB. The device's maximum gain error is stated at ±8.5% full scale (FS), while the input offset voltage has a range of –32 to +51mV over the converter's full temperature range of –40 to +85°C.

An ADC's differential non-linearity is defined as the deviation of any code width from an ideal 1 LSB step. It's integral non-linearity is similarly defined but instead compares the transfer function with a reference line measured in fractions of 1 LSB using a "best straight line" determined by a least square curve fit. Therefore, the device's non-linearity can be said to cause inaccurate code transitions to occur by up to 1.75 LSB as opposed to the ideal code transitions that occur 1 LSB apart.

Analogue V_{in} (mV)	Predicted digital o/p (dec)	Actual digital o/p (dec)	Output Error (LSB)	Percentage diff. (%)
2.8000	12	4	-8	66.67
2.8500	62	56	-6	9.68
2.9000	112	108	-4	3.57
2.9500	162	160	-2	1.23
3.0000	212	212	0	0.00
3.0500	262	264	2	0.76
3.1000	312	316	4	1.28
3.1500	362	368	6	1.66
3.2000	412	420	8	1.94
3.2500	462	472	10	2.16
3.3000	512	524	12	2.34
3.3500	562	576	14	2.49
3.4000	612	628	16	2.61
3.4500	662	680	18	2.72
3.5000	712	732	20	2.81
3.5500	762	784	22	2.89
3.6000	812	836	24	2.96
3.6500	862	888	26	3.02
3.7000	912	940	28	3.07
3.7500	962	988	26	2.70
3.8000	1012	1020	8	0.79

Table 5.3: Measured and calculated dc precision results for the AD9050 ADC, in the ambient air temperature of 22°C

A finite amount of signal noise and the converter's non-linear properties were found to cause the two LSB to exhibit fluctuating characteristics even when a steady clean analogue signal was applied. These two bits were therefore deemed indeterminate and were not considered when the ADC's actual digital output values were being measured, which are given in Table 5.3, when the ambient air temperature was recorded at 22°C. As a consequence of the inconstant nature of the 2 LSB, when recording the output values, the ADC was treated as an 8-bit resolution device with the 8 MSB retaining their original 10-bit associated weights. Therefore a +5V (high/1) output on the MSB line still corresponds to a decimal value of 512_d (2^{n-1} where n is the corresponding bit number in ascending order). This mid-range predicted value of 512_d represents 10 0000 0000_2 in binary as the equation used to determine each predicted digital output is:

$$\text{Digital o/p} = 1000 * \text{Level Shifted Analogue i/p } (V_{in} \text{ in volts}) - 2788 \qquad (5.1)$$

By rearranging this equation, the mid-range ADC's input voltage is found as expected to be 3.3V. The difference between the predicted and actual output values initially converge and after a certain point diverge as the input voltage is increased from 2.8V to 3.8V. The minimum difference is reached when a 3.0V input signal is applied while the maximum difference occurs at 3.7V. The device's gain error must therefore be the main cause of the observed variation. Hence, the ADC's gain was found to be steeper then expected. This is confirmed when the tabulated output data is contrasted in graphical form (Figures 5.8(a) and (b)) over the input voltage range.

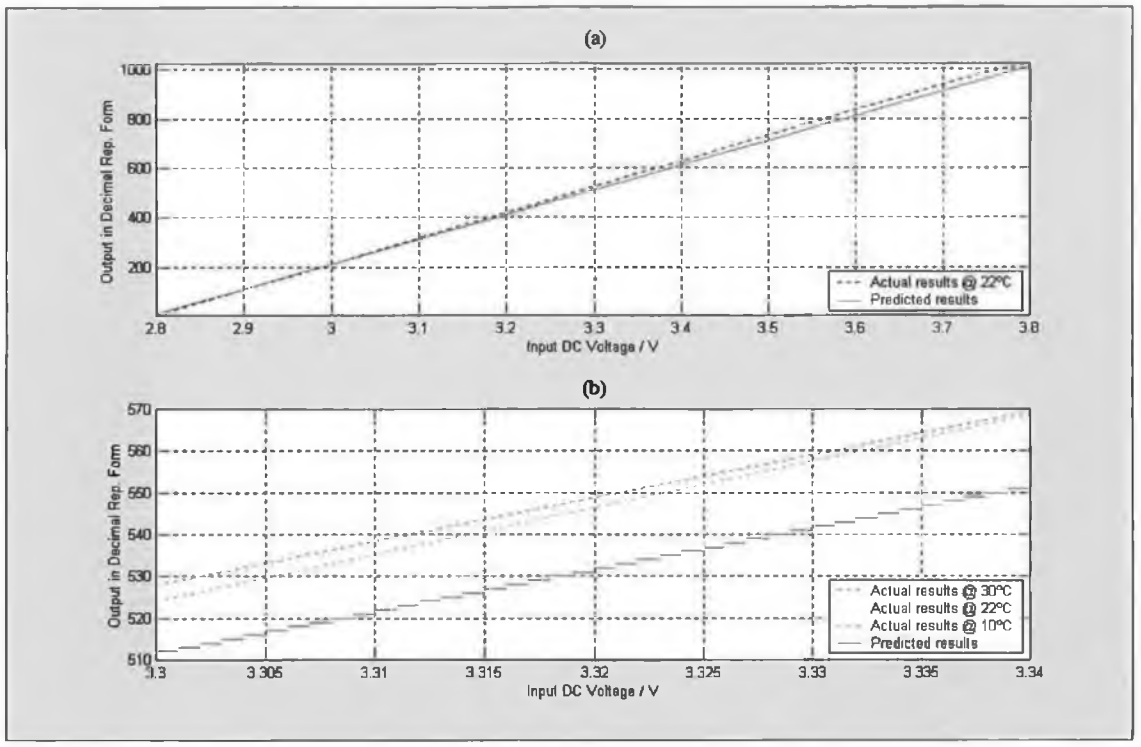


Figure 5.8: Predicted and actual i/p-o/p transfer curves for the AD9050 ADC: (a) over the entire recorded input voltage range and (b) for an expanded section near the centre of the range

The plot of Figure 5.8(a) shows the device's room temperature slope diverging away from the predicted transfer line as the input voltage level is increased. The predicted and actual transfer data are displayed in Figure 5.8(a) with solid and dashed lines respectively. The typical method of displaying an ADC's quantised nature was avoided in Figure 5.8(a), since a total of 2^{10} (1024) possible transition levels, each 1mV in width, was deemed impractical for the range and scale used. Instead, the expanded plot of Figure 5.8(b) shows the ideally predicted code transitions for the

tested ADC. Figure 5.8(b) includes the other results when the air temperature was set to 10°C and 30°C as well as the room temperature transfer curve in dashed form. Each curve represents the locus of the respective midpoints of the actual code range. The room temperature segmented transfer curve produces marginally better results for the AD9050 with respect to the other two set temperatures.

Table 5.4 summarises the temperature drift effects on the AD9050. The ADC's calculated input offset voltages never exceeded the AD9050's data sheets specified offset range of –32 to +51mV over the tested temperature range. Therefore, the ADC was determined to function within specified dc parameters. However if a greater level of precision is required for a given application and the device's cost and package type are not major issues, a recommended replacement of the AD9050 is advised. The 14-bit, 80MSPS AD6645 ADC from Analog Devices is recommended. The AD6645 is available in a 52-lead power quad-4-package and has an improved typical offset error of +1.2mV, with a low quantity price of approximately €50. In applications where dc coupling is required, the most compatible differential op-amp that could be used to drive the AD6645 is the AD8138 from Analog Devices. The AD8138 op-amp provides single-ended-to-differential conversion that minimises the layout requirements [5-4].

Set Temperature /°C	Input Offset Voltage /mV	Intercept / dec	Slope/Sensitivity & Absolute Uncertainty dec/V
10	-8	-2918	1042 ± 9
22	-11	-2886	1033 ± 4
30	-15	-2872	1030 ± 4

Table 5.4: Linear regression calculations for the AD9050 ADC in different ambient air temperature conditions

The most cost-effective option involves using the AD9050 ADC in conjunction with a suitable calibration technique that corrects the linear offset error. The linear regression calculations of Table 5.4 can be used, however the number of measurements needed to determine these results is not essential. Taking only one, two or three measurement steps significantly reduces the offset error. Low measurement steps produce improved

results while restricting the measurement time to a minimum. When employing three measurement steps, a good calibration is generally obtained when the calibration points are selected in the following sequence: (a) the first point at one end of the input range of operation, (b) the second at the other end of the range, and (c) final calibration point halfway between the two previously selected points [5–6].

This basic, one-dimensional, calibration method has been theoretically tested on the AD9050 transfer function and then developed into a programmer-defined function, which was implemented with suitable source code, on the PIC-microcontroller (μC) platform. Refer to the analogue inputs and actual digital outputs of Table 5.3 for the initial theoretical situation. Taking the first, last and mid-range values will produce the linear equation: y = 1016*x - 2837. Its inverse offset error correction equation is obviously and simply determined to have the form: y = 984*x – 2739, when the ideally related function for the ADC is considered, i.e. y = 1000*x – 2788. These coefficients can be programmed into a suitable PIC-μC or DSP device. The offset error output (y) from the ADC is applied to the rearranged correction equation to determine the true compensated input (x) value. This transfer correction method can also be applied to the difference amplifier's precision results to sufficiently reduce its linear gain error.

The flowchart diagram of Figure 5.9 helps illustrate the user directed automated software programme developed for testing the calibration function. This calibration function was incorporated into a fundamental signal processing programme that could sequentially display peak-to-peak and mean voltage values of an applied analogue signal, through a LCD display, once the system has been calibrated. From the flowchart it is apparent that the implemented one-dimensional calibration method requires only two reference points, one at either end of the operating range (0-5V). The user is suitably prompted as to when and where the reference signals should be applied, with sufficient adjustment time given between prompts. After each reference signal is applied, its corresponding digital equivalent is displayed and recorded for the linear regression calculations. If any slope or intercept differences exist between the ideal and actual values, a new calibration data set is determined and stored in memory. This calibration data compensates for offset and sensitivity errors caused by the system's instrumentation.

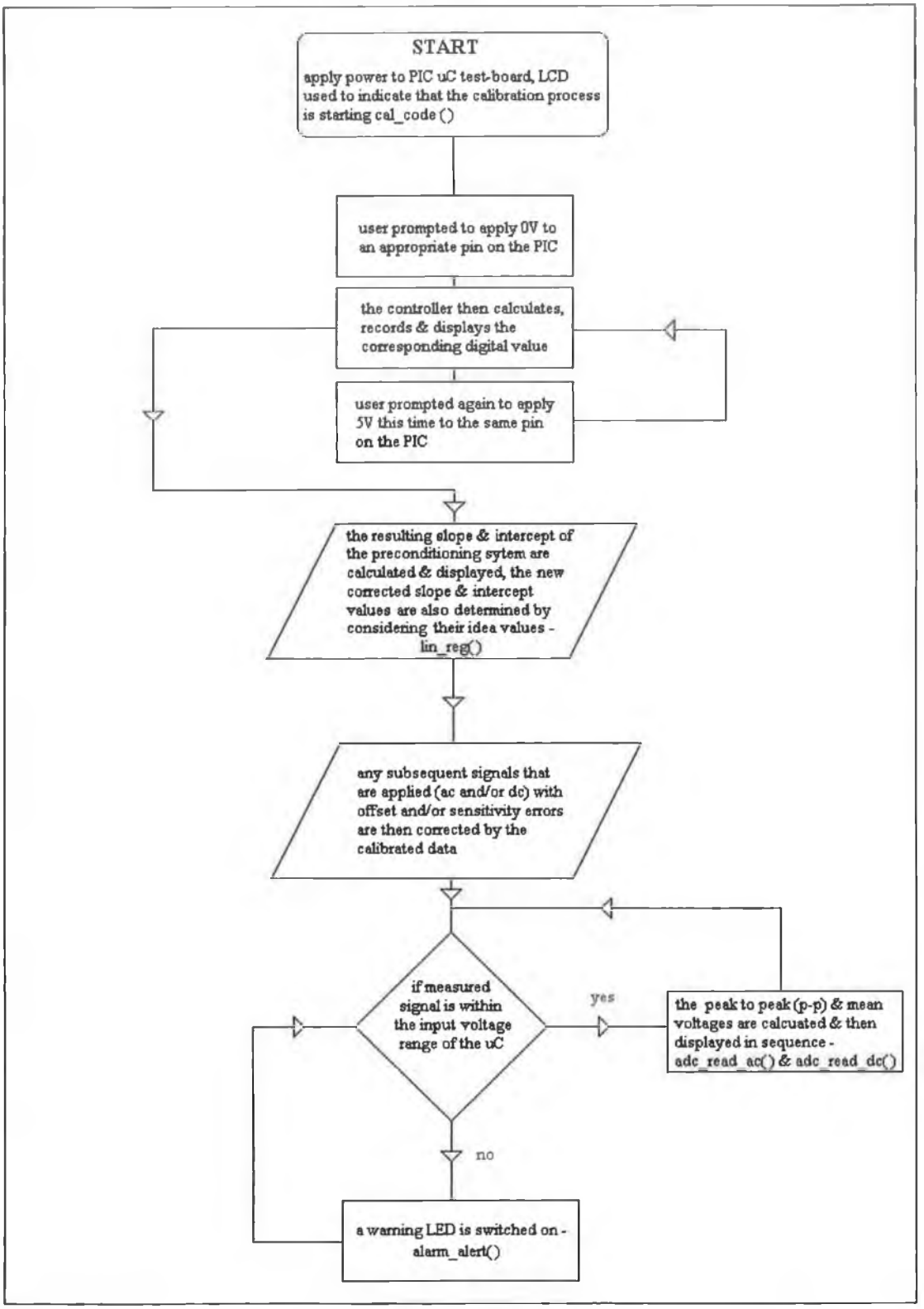


Figure 5.9: Flowchart diagram of fundamental signal processing programme with calibration function incorporated

The specific C source code including calibration function, which is associated with the flowchart diagram of Figure 5.9, is documented in Appendix Section A.1. While a

related PIC-µC "get started guide" document is available from the DCU School of Electronic Engineering. This guide forms a report that was primarily written for PEI Technologies personnel and postgraduate research students planning to commence work on a project incorporating a Microchip PIC-µC device. Originally the PIC-µC was strongly considered as a possible signal processing element for the developing data acquisition system. However, for high-speed applications this device was subsequently replaced by a more suitable high-performance embedded DSP (discussed further in Section 5.2.1). The calibration programme was initially implemented, tested and examined by hardware associated with the PIC-µC. C is a portable language therefore the linear calibration function development can be transferred and run on different microprocessor systems (including an embedded DSP) with little or no modifications needed.

5.1.7 Digital to Analogue Circuitry

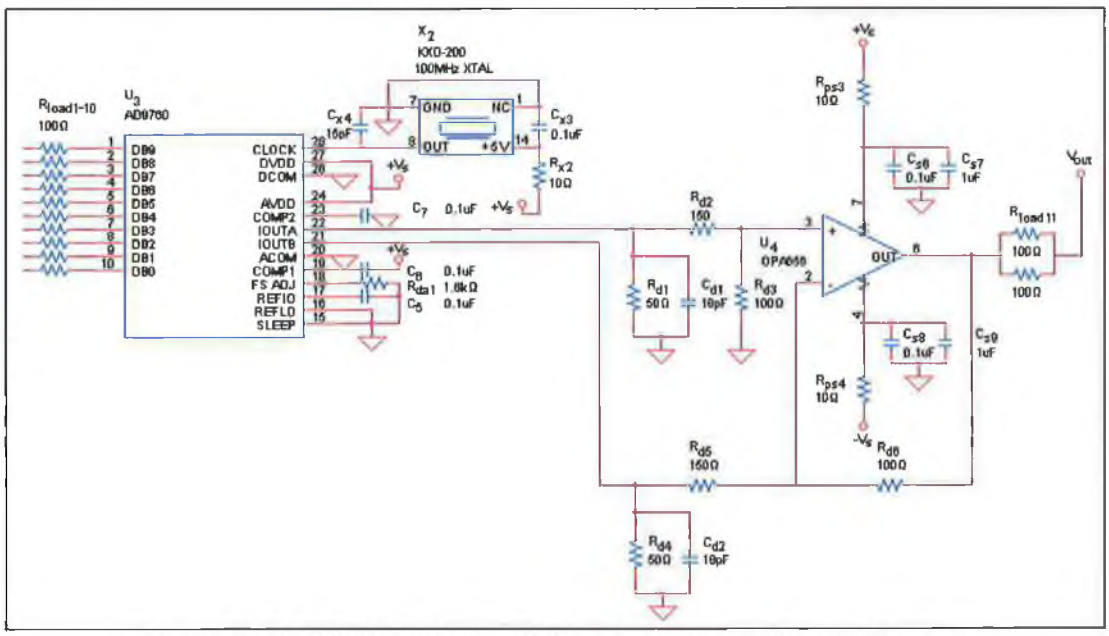


Figure 5.10: DAC device and signal recovery circuit diagram

Figure 5.10 illustrates separately the 125MSPS AD9760 DAC and its differential-to-single-ended signal recovery circuitry initially shown in Figure 5.1. The AD9760 was implemented to reconstruct and analyse the analogue test signals. The DAC's resolution is the same as the 10-bit AD9050 while the 100MHz crystal is used to

achieve an appropriate sample rate. The AD9760 provides complementary current outputs, I_{OUTA} and I_{OUTB}. The full-scale output current is regulated by the internal reference control amplifier and can be set from 2mA to 20mA via the external resistor, R_{dal} in Figure 5.10. The differential operation of the DAC helps cancel common-mode error sources associated with I_{OUTA} and I_{OUTB} such as noise, distortion and dc offsets.

Digital signal paths should be kept short and run lengths matched to avoid propagation delay mismatch. The insertion of the low value resistors (100Ω) between each AD9760 digital inputs and the AD9050 digital outputs reduce any overshooting and ringing at the digital inputs that contribute to the data feed-through [5-7]. The OPA656 op-amp was used to perform the differential to single-ended conversion as shown in Figure 5.10. The observed phase properties of the AD9050 ADC were evident by examining the DAC recovered high-speed test signals. A phase delay of approximately –60° was confirmed by comparing the ADC 1MHz sinusoidal input signal to the DAC single-ended output. This phase difference is greater than the initial difference of –50°, displayed in Figure 5.7. The DAC device, and to a lesser extent the attached VFA, are the main cause of the additional phase delay.

5.2 Data Acquisition System

Any data acquisition system consists of four essential elements: transducer, signal conditioning, signal processing and data presentation elements. Figure 5.11 displays the general sequence of elements along the signal path that combine to form a data acquisition system. Typical noise entry points are indicated with dashed lines. Proper noise-reduction measures must be applied to maintain a system's high performance.

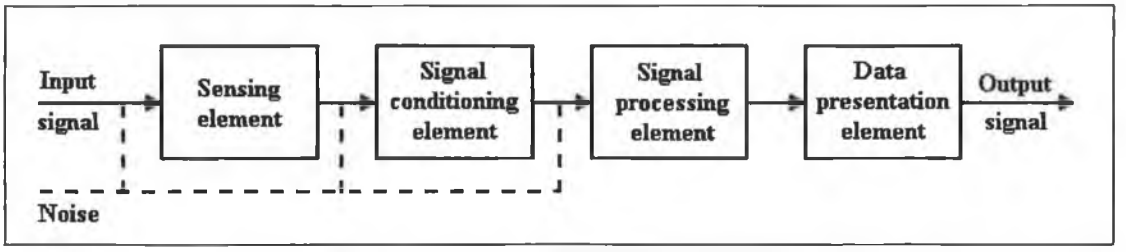


Figure 5.11: General structure of a data acquisition system

When developing a compact data acquisition system, the signal conditioning, signal processing and data presentation elements displayed in Figure 5.11 are generally merged. A PC and some application software typically present the data as a Digital Signal Processor (DSP), interfaced signal conditioning module and communication link completes the arrangement. With the principal signal conditioning devices already selected and tested, the next practical step is to find a compatible low-cost high-performance DSP that has PC communication capability.

5.2.1 Digital Signal Processors

Gathering and analysing massive amounts of test data in real-time is where DSPs excel [5-8]. These features combined with their ability to provide exceptionally fast mathematical computations makes DSP devices particular useful for high-speed applications. Two of the world's leading DSP suppliers are Analog Devices and Texas Instruments. An overview of the various platforms available from both companies is presented as relevant data sheets and application notes were evaluated. The purpose of this review is to determine the most suitable option, which is cost-effective, easy to use, with high-performance capability and a high-speed data transfer rate compatible with the established signal conditioning instrumentation.

Analog Devices portfolio includes mixed-signal and general-purpose DSPs, such as the SHARC®, TigerSHARC®, Blackfin™, and ADSP-21xx DSP families. Analog Devices DSP architectures feature simple, yet powerful, programming models and are supported by high-quality development tools.

The Analog Devices family members are code compatible. For example, all the ADSP-21xx family members share the same base architecture and assembly language. Hence, there is no need to learn or invest in new development tools when moving from one family member to another as any initial software investment is preserved. Algebraic syntax assembly language makes code easy to learn, use and read. Fast core processing and high bandwidth I/O is integrated. Large on-chip memory provides ample on-chip storage for most common DSP tasks, such as digital filtering and FFTs, thus eliminating the need for off-chip memory.

More specifically, the ADSP-218x M and N series members offer low-voltage (1.8V), low-cost (€10-30) and high-performance 80MHz, 16-bit DSPs. While the ADSP-2199x family of mixed-signal DSPs provides single-chip, high-performance solutions for embedded control and signal processing applications at a competitive price. The ADSP-2199x family represents the highest-performance mixed-signal DSPs generally available today. These products combine the 160MIPS, ADSP-219x DSP core with an 8-channel, 14-bit, 20MSPS ADC system as well as the right mix of embedded control peripherals and comprehensive development tools [5-9].

Texas Instruments incorporates similar DSP chips on platforms that are optimised for high-performance and ease-of-use with high-level language programming. The TMS320 DSP family offers an extensive selection of DSPs, with a balance of general-purpose and application-specific processors. There are three distinct Instruction Set Architectures that are completely code-compatible within the platforms, from the control optimised TMS320C2000™ DSP platform through to the power efficient C5000 and high-performance C6000 systems.

The TMS320C2000™ DSP platform provides the digital control industry with the highest level of on-chip integration and powerful computational abilities that produce unparalleled improvements in energy efficiency. The TMS320C28x™ DSP generation is the highest-performance solution for digital control. The TMS320C24x™ DSP generation is the foundation for this diverse platform. This generation delivers power and control advantages that allow designers to implement advanced cost-efficient control systems.

The TMS320C5000™ DSP platform is optimised for the consumer digital market - the heart of the mobile Internet - and it's convergence with other consumer electronics. With power consumption as low as 0.33mA/MHz, the TMS320C55x™ and TMS320C54x™ DSPs are optimised for personal and portable products like digital music players, GPS receivers, portable medical equipment, 3G cell phones, digital cameras as well as MIPS-intensive voice and data applications and are extremely cost effective for single and multi-channel applications. Based on the C55x DSP core, the OMAP5910 processor integrates a C55x DSP core with a TI-enhanced ARM925 on a single chip for the optimal combination of high-performance with low

power consumption. This unique architecture offers an attractive solution for both DSP and Advanced RISC Machine (ARM) developers, providing the low power real-time signal processing capabilities of a DSP coupled with the command and control functionality of an ARM. The OMAP5910 is optimal for designers working with devices that require embedded applications processing in a connected environment.

Raising the bar in performance and cost efficiency, the C6000™ DSP platform offers a broad portfolio of the industry's fastest DSPs running at clock speeds up to 1 GHz. The platform consists of the TMS320C64x™ and TMS320C62x™ fixed-point generations as well as the TMS320C67x™ floating-point generation. They range in price from around €20-30. The C6000 DSP platform's performance ranges from 1200 to 8000 MIPS for fixed-point and 600 to 1350 MFLOPS for floating point, optimal for designers working on targeted broadband infrastructure, performance audio, imaging and precision instrumentation based applications.

This review has indicated a number of competitively priced systems that are particularly suited for high-speed applications. After a more detailed study, the TMS320C6711 DSP was identified as the most appropriate candidate of the systems reviewed. The TMS320C67x™ DSPs comprises the floating-point DSP family in the TMS320C6000™ DSP platform. The C6711 device is based on the high-performance, advanced VelociTI™ very long instruction word (VLIW) architecture developed by Texas Instruments, making these DSPs an excellent choice for multi-channel and multi-function applications.

The C6711 DSP have application-specific hardware logic, on-chip memory, and additional on-chip peripherals. The C6711 uses a two-level cache-based architecture and has a powerful and diverse set of peripherals. The Level 1 programme cache (L1P) is a 32-kbit direct-mapped cache. The Level 2 memory/cache (L2) consists of a 512-kbit memory space that is shared between programme and data space. L2 memory can be configured as mapped memory, cache, or a combination of the two. The peripheral set includes two multi-channel buffered serial ports (MCBSPs), two general-purpose timers, a host-port interface (HPI), and a glueless external memory interface (EMIF) capable of interfacing to SDRAM, SBSRAM and asynchronous peripherals [5-10].

The C6711 DSP's equivalent development board (TMS320C6711 DSK) includes standard cross-platform peripheral connectors, which allow for a designed-to-specification prototype board, with implemented signal conditioning and data conversion instrumentation to be rapidly evaluated. A compatible prototype board, which forms the DSP's daughter-board, can accelerate the application design process and add new functionality for several existing C6000 and C5000 systems. Signals are brought to a prototype board through two 80-pin connectors, with one connector primarily for peripheral power and control signals and the other primarily for the external memory interface. A suitable low cost DSP with enhanced architecture and extended memory size combined with an easy-to-reproduce prototype board with high-speed signal analysis and generation instrumentation could create a powerful high-performance system well worth developing.

The TMS320C6711 DSK was selected as the data acquisition system's signal processing element that communicates with a PC via a parallel port cable. With the processor system introduced, a single easy to implement, compatible printed circuit daughter-board can be formed.

5.2.2 PCB Design and Physical Layout

Orcad Capture and Layout files were initially developed that contained schematic circuit design, footprint component types, standard board boundary, track dimension, plane and component orientation information for the prototype PCB design. The format of the post processing (Gerber and Excellon) files needed converting, before the LPKF ProtoMAT C305 apparatus can be used to physically machine the board. CircuitCAM first imports and then suitably modifies these post processing files after a number of board related calculations have been performed. A further application, BoardMaster, accepts these modified files and controls the actual machining/drilling processes performed by the LPKF ProtoMAT C305 apparatus.

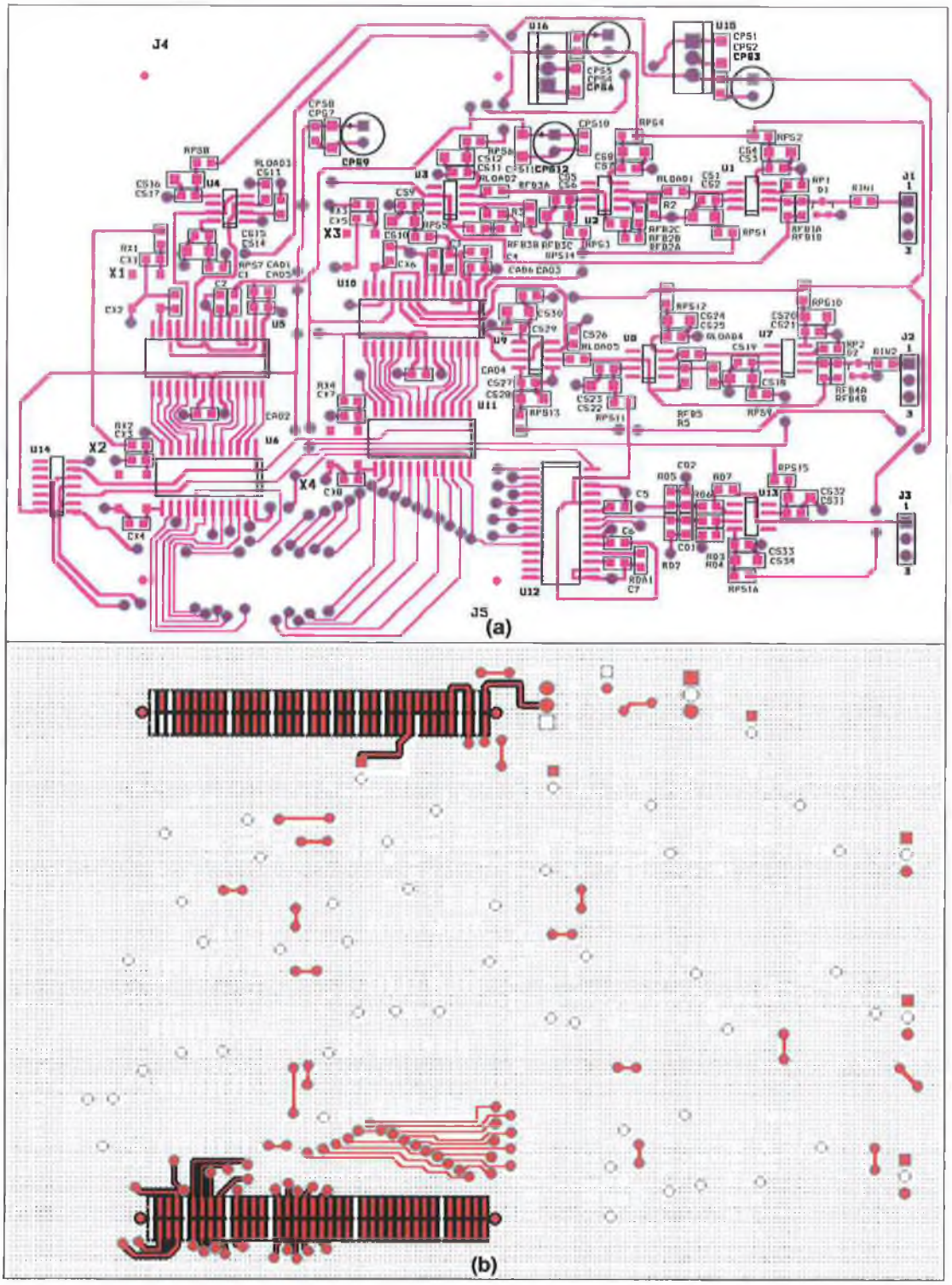


Figure 5.12: Printed circuit daughter-board layout, (a) top layer (b) bottom layer

Figure 5.12 illustrates the layout of a user specific PCB that is approximately 133cm^2 (14 x 9.5cm) in area. This prototype board embodies multi-channel front-end sensor interfaces, signal conditioning instrumentation, analogue to digital and digital to analogue modules and DSP data bus interface. Power supply regulators and a transfer

data logic control device with matching lines are included. Two matching female daughter-board connectors, available from Samtec, form a physical link with the complementary gender connectors on the DSP mother-board.

With the purpose of decreasing the board's size while allowing the soldering work to be a manageable activity, the chipset selection process was restricted. Only SOICs and SMDs were chosen over the larger DIL or miniaturised alternatives. For example, alternative SMDs replaced the original DIL crystal oscillators shown in Figure 5.1. A fully reproducible, simple-to-implement, arrangement is the substantial effect of this approach to developing a prototype board.

This prototype board combines the essential analogue amplifier and data converter devices already selected and tested in this and the previous chapters. The additional 10-bit bus interface buffers (U6 and U11) are implemented to prevent damage to the output pins of the respective ADCs (U5 and U10) when output bus data meant for the DAC (U12) is applied. Such data can be software controlled as synthesised signals through the DAC allows diagnostic system tests to be performed.

Careful attention to the PCB layout is inevitably the difference between a fully functioning system and one that doesn't live up to expectations. Hence a number of practical precautions were employed to ensure that a low distortion, high-speed and robust data acquisition (and data generation) system would result. Some of the most effective measures included, a tight overall layout design with the minimum track length used, decoupling the power supply lines and isolation of sensitive pins from parasitic capacitance.

The aim of facilitating a range of output sensor signals was initially investigated (in Section 3.4.5) as a span of somewhat elaborate variable gain systems were controlled by a set of relays. The implemented prototype design is a streamlined version of one of the gain selection systems investigated. Two separate front-end input channels are provided on the prototype board, each with a distinctive gain that can be modified for a specific application. Therefore, the developed board is capable of high-speed interfacing with multiple output-voltage sensor ranges. The Appendix Section A.2

contains the PCB schematic diagrams, while each component of the PCB is described in Section A.3 where the board's chipset is listed.

5.3 Summary

The output sensor range and the input voltage limits of an ADC collectively determine the required level of front-end signal amplification. When the ADC's mid-scale value is non-zero a difference/differential amplifier is necessary to suitably dc shift the amplified signal. A logically-controlled, data buffer device completes the conversion process, as the selected ADC becomes the key signal conditioning component.

Certain factors took priority as an appropriate ADC was selected. The order of priority was: package type, followed by cost, speed and then accuracy. This led to the 10-bit AD9050 ADC with its small outline packages, €10 price tag, 40/60MSPS sample rates and 7mV typical offset error to be selected. A compatible difference amplifier (AD830), data buffer (SN74ABT821A) and DAC (AD9760) devices were subsequently chosen.

With the use of a test board, the data converter devices were initially implemented and then analysed. The AD830 difference amplifier and AD9050 ADC's graphical and tabular data that resulted, illustrated the respective devices independent speed and precision properties. Despite varying the test board's ambient air temperature, all the implemented devices remained functioning within specifications. Only when the input signal's frequency was set at or above 10MHz was a phase delay observed through the difference amplifier. A more noticeable phase lag of –50° was recorded for the ADC as a 1MHz level shifted input signal was applied, while below 500kHz no phase delay was observed.

In the event that these devices are found to have significant offset or sensitivity errors, a user directed automated calibration programme was developed to compensate for these linear errors.

A DAC device was initially implemented on a test board to allow analogue input signals and reconstructed digital signals to be compared in the same analogue form. This approach allows the data conversion instrumentation's high-speed performance to be rapidly verified.

To add high-speed processing power to the data acquisition system the TMS320C6711 DSK mother-board was selected. This low-cost DSP development board includes standard cross-platform peripheral connectors that allow for a prototype PCB with signal conditioning and data conversion instrumentation to be easily implemented. The compatible prototype board, which forms a DSP's detachable daughter-board, can accelerate the application design process and add new functionality for several existing C6000 and C5000 systems. Signals are brought to a prototype board through two 80-pin connectors, with one connector primarily for peripheral power and control signals and the other primarily for the external memory interface. This PCB incorporates a multi-channel front-end sensor interface, high-performance signal conditioning instrumentation, analogue to digital and digital to analogue modules and DSP data bus interface.

6

Advanced Signal Processing Software

The functionality of the established data acquisition system is determined not only by it's high-performance analogue conditioning instrumentation elements and integral digital signal processing hardware, but can be modified as software support is added and executed. Such programmability ensures the practical issues of front-end gain control and sample rate adjustment complement a range of signal amplitudes and frequencies. Implementing advanced digital filtering and Fast Fourier Transform (FFT) algorithms is a recommended processing enhancement, ideal for sensor-based applications where noise and interference are issues. Real-time diagnostic programmes, which utilise digitally generated test signals to determine a system's performance, are also advised.

The DSP development platform selected to implement the multi-functional programmes is the low cost TMS320C6711 DSK. Supplied with this target system is the Code Composer Studio (CCS) application software developed by Texas Instruments that allows a user to interface with the DSP system through a PC. CCS provides tools for configuring, building, debugging, tracing and analysing programmes. Numerical and graphical data visualisation facilities are also supplied by this powerful application.

A series of DSP algorithms, implemented on the TMS320C6711 DSK through the CCS application, are presented in this chapter. Prior to presenting specific processing methods and designs, a number of critical peripheral and external access memory issues are discussed.

6.1 EMIF and EDMA

Figure 6.1 shows the block diagram of the printed circuit daughter-board. This board was designed to act as a peripheral memory device, capable of working on a number of compatible C6000 and C5000 DSP mother-boards, through a standard interface. Signals are brought to the daughter-board through two 80-pin connector headers, with one connector (J4) primarily for peripheral signals and the other connector (J5) primarily for the external memory interface (EMIF).

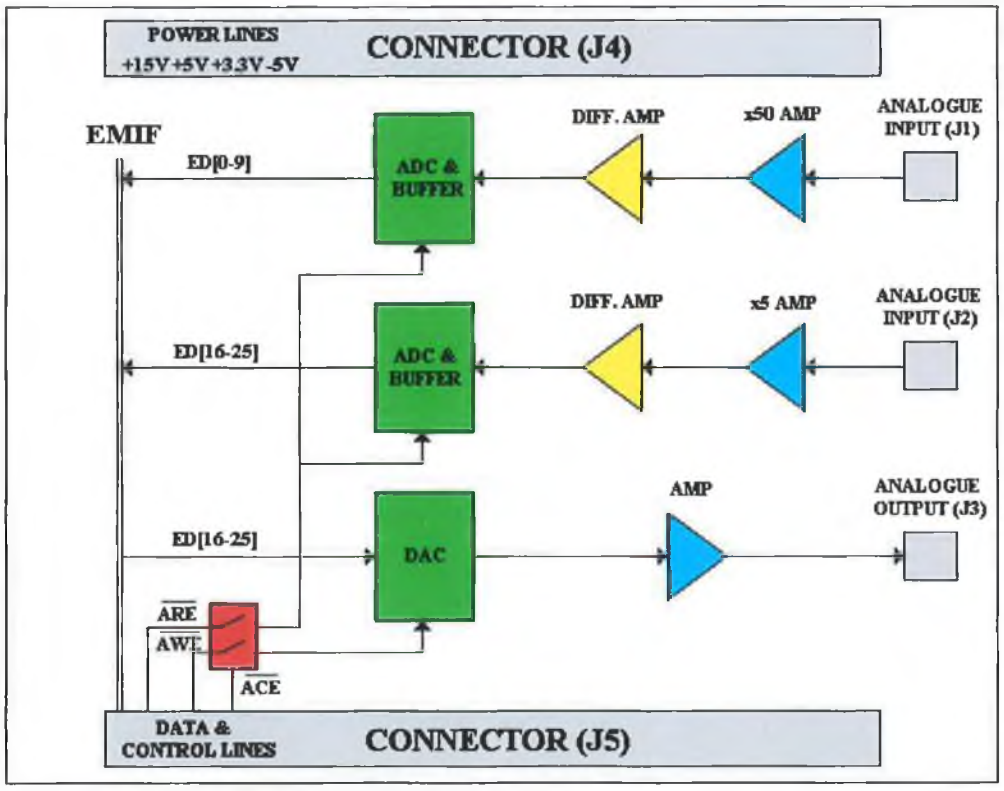


Figure 6.1: Printed circuit daughter-board block diagram

The selected TMS320C6711 DSK mother-board provides an asynchronous memory interface capable of communicating with 8-, 16- or 32-bit memory. As only 20 lines

(ED[0-9] and ED[16-25]) of the possible 32 external data lines are used by the daughter-board, caution is needed when 32-bit memory mode is selected. An appropriate software function can ensure that only the 10-bit sampled data of each channel is considered, as the remaining unused data lines are ignored. There are four chip selects (CE[0-4]) that are used when accessing a specific memory location (e.g. if you try to access asynchronous memory, 0xB000 0000H, then CE3 will be activated). The memory is controlled with separate asynchronous active low read ($\overline{\text{ARE}}$), write ($\overline{\text{AWE}}$) and chip enables ($\overline{\text{ACE}}$) signals [6-1].

Figure 6.2 shows the EMIF bi-directional path as the processor services requests of the external bus from the on-chip enhanced direct-memory access (EDMA) controller. Also indicated is the on-chip two-level (L2) architecture for programme and data memory. The first level programme cache is designated L1P, and the first level data cache is designated L1D. Both the programme and data memory share the second level 64-kbytes of internal memory designated L2. The EDMA controller handles all data transfers between the L2 cache/memory controller and the device peripherals on the TMS320C6711 DSK. These data transfers include cache servicing, non-cacheable memory accesses, user-programmed data transfers, and host accesses [6-2].

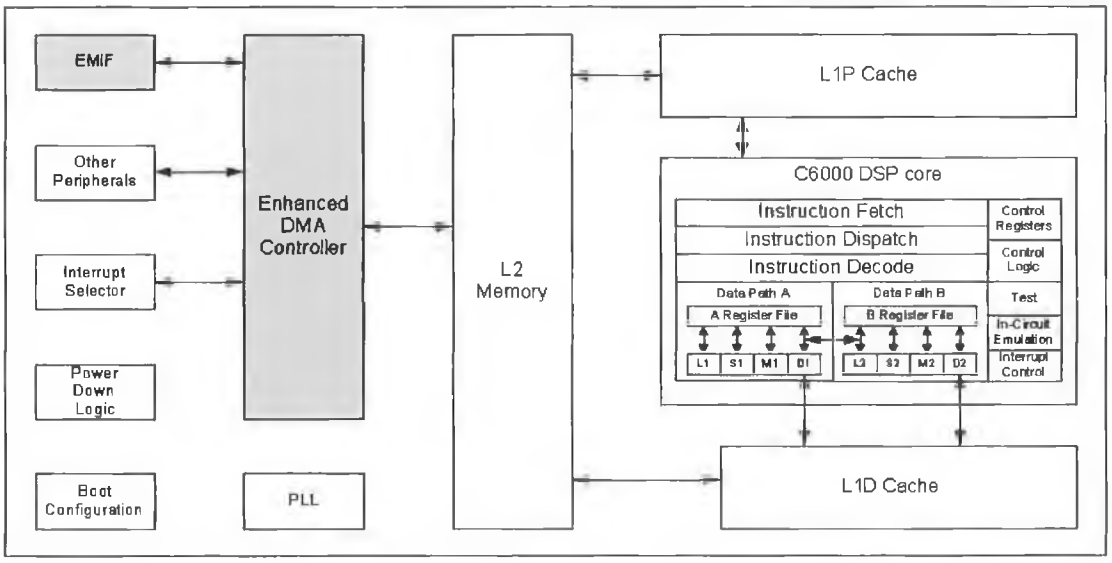


Figure 6.2: Block diagram of the external memory interface in the TMS320C671x DSPs

Through proper configuration, the EDMA channels can be set up to operate continuously without requiring CPU intervention or reprogramming. This allows the CPU to use its MIPS for data processing, while the EDMA handles data management in the background. For the C6711, 16 channels plus a Quick DMA (QDMA) register set is programmable to perform data transfers during CPU operation. EDMA channels and QDMA register sets are useful to transfer data to/from any location in the DSP's memory map. All transfers are synchronised and each channel has a dedicated synchronisation event. Note that QDMA transfers are synchronised by the CPU. Each requestor (L2 controller, EDMA channel) submits a transfer request to be processed by the EDMA. The requests are queued according to priority, with higher priority requests serviced first. Because of the EDMA's structure, transfers requested through different queues (though submitted according to priority) can occur simultaneously. This maximises the bandwidth available to transfer data and allows for the efficient transferring of data without hindering the performance of the cache.

In conclusion, proper configuration of the EDMA channels enables servicing of all incoming and outgoing data streams to/from the DSP, without requiring significant processing time by the CPU to manage the transfers. Thus, the CPU is primarily left to focus on data processing [6-3].

6.1.1 Enhanced DMA Transfer Example

The on-chip EDMA controller is the backbone of the system used by the two-level cache architecture. Synchronous background data transfers are configurable in a special on-chip parameter RAM (PaRAM). There are 16 channels, which can be configured in PaRAM, with each channel corresponding to a specific synchronisation event to trigger the transfer. The EDMA channels may be configured to access any location in the device's memory map. This includes internal memory, external memory, on-chip peripherals and external analogue front-end (AFE) circuits.

The following configurable parameters are available for each EDMA channel:

- Options: transfer configuration settings
- Source address: the memory location from whence the elements are transferred
- Destination address: the memory location to which the elements are transferred
- Array/frame count: the number of arrays or frames to be transferred minus 1
- Element count: the number of elements in an array or frame
- Array/frame index: the offset used to calculate the starting address of each array/frame
- Element index: the spacing between the addresses of elements within a frame
- Link address: the parameter RAM address of the parameters to be loaded upon completion of the current transfer

```
/*
  ----------------------------------------------------------------------------------------
  Function:     submit_qdma()
  Description:  Submit a QDMA request to transfer the sampled data to internal (L2) memory.
  Inputs:       -
  Outputs:      -
  Returns:      -
  ----------------------------------------------------------------------------------------
*/

void submit_qdma(void){

    EDMA_Config config;

    config.opt = (Uint32)                          /* EDMA channel options selected (0x20200001): */
          ((EDMA_OPT_PRI_HIGH     << _EDMA_OPT_PRI_SHIFT    )   /* High priority EDMA transfer,  */
         | (EDMA_OPT_ESIZE_32BIT  << _EDMA_OPT_ESIZE_SHIFT  )   /* 32-bit element size,          */
         | (EDMA_OPT_2DS_NO       << _EDMA_OPT_2DS_SHIFT    )   /* 1-dimensional source,         */
         | (EDMA_OPT_SUM_NONE     << _EDMA_OPT_SUM_SHIFT    )   /* Fixed source address mode,    */
         | (EDMA_OPT_2DD_NO       << _EDMA_OPT_2DD_SHIFT    )   /* 1-dimensional destination,    */
         | (EDMA_OPT_DUM_INC      << _EDMA_OPT_DUM_SHIFT    )   /* Increment destination address,*/
         | (EDMA_OPT_TCINT_NO     << _EDMA_OPT_TCINT_SHIFT  )   /* Transfer complete indicator   */
         | (EDMA_OPT_TCC_DEFAULT  << _EDMA_OPT_TCC_SHIFT    )   /* disabled, with no code set,   */
         | (EDMA_OPT_LINK_NO      << _EDMA_OPT_LINK_SHIFT   )   /* Event parameter link disabled,*/
         | (EDMA_OPT_FS_YES       << _EDMA_OPT_FS_SHIFT     )); /* Frame/block synchronised.     */

    config.src = (unsigned int)MEM_SRC;          /* Source address for transfer      (0xB0000000) */
    config.cnt = (unsigned int)EL_COUNT;         /* Element count for transfer       (0x00001000) */
    config.dst = (unsigned int)MEM_DST;          /* Destination address for transfer (0x00006000) */
    config.idx = (unsigned int)0;                /* Element index offset addressing  (0x00000000) */

    EDMA_qdmaConfig(&config);

    return;

} /* end submit_qmda */
```

Figure 6.3: Example code to perform Quick DMA (QDMA) data transfer during CPU operation

The EDMA also has the capability of performing unsynchronised transfers though the use of a QDMA request by the CPU that is the quickest way to perform any transfer. The QDMA is used to issue single, independent transfers to move data quickly, rather than perform periodic or repetitive transfers through the EDMA channels. The sub-routine (displayed in Figure 6.3) contains the essential parameters, where the QDMA is used to transfer blocks of data as the AFE is continuously serviced. The QDMA

does not have the capability to reload a count or link. Therefore the count reload/link address register is not available to the QDMA and consequently the linking parameter in this example is appropriately excluded as sampled data is transferred to internal memory.

A programme using this function would initially define the memory source, memory destination and element count values as symbolic constants. When executed the sampled data at the source address (0xB000 0000H), relating to the EMIF CE3 location, would be transferred through the EMIF to the L2 internal memory (with a starting address at 0x0000 6000H). In this example, 1000H (4k 32-bit sample) elements are transferred in a time dependent on the EMIF CE3 control register settings. This type of transfer is valid for block sizes of less than 64k elements, this equates to filling the L2 internal memory in one move. The following section (6.1.2) highlights the EMIF CE3 control register settings that can be optimised for high-speed sampling.

6.1.2 EMIF CE3 Control Register

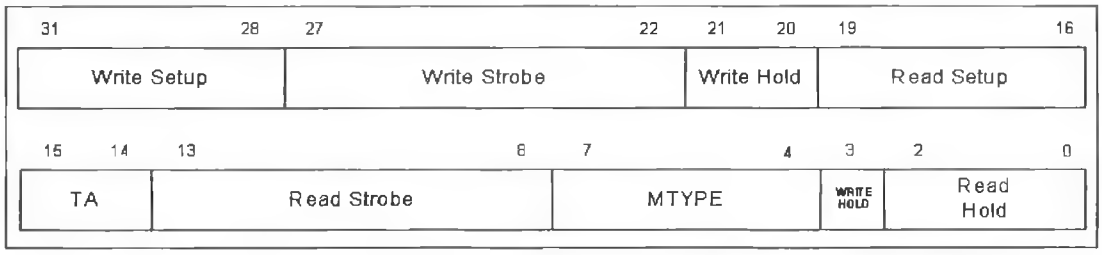


Figure 6.4: TMS320C671x DSP's EMIF CE3 space control register (CE3CTL)

Access to external memory devices requires the EMIF control registers to be appropriately configured. The specific CE3 space control register (CE3CTL), shown in Figure 6.4, corresponds to the CE3 memory space supported by the EMIF. There is a total of four CE space control registers corresponding to the four external CE signals. The MTYPE field identifies the memory type for the corresponding CE space. If the MTYPE field selects a synchronous memory type (SDRAM, SBSRAM), the remaining fields in the register have no effect. If an asynchronous type is selected (ROM or asynchronous), the remaining fields specify the shaping of the address and

control signals for access to that space. The programmed values in the CE3CTL refer to ECLKOUT (EMIF clock cycle), with a frequency of 100MHz.

The EMIF allows a high degree of programmability for shaping asynchronous access. The programmable parameters that allow this are:

- Setup: The time between the beginning of a memory cycle ($\overline{ACE}$ low, address valid) and the activation of the read or write strobe
- Strobe: The time between the activation and deactivation of read ($\overline{ARE}$) or write strobe ($\overline{AWE}$)
- Hold: The time between the deactivation of the read or write strobe and the end of the cycle (that can be either an address change of the deactivation of the $\overline{ACE}$ signal)
- TA: Turn-around time controls the number of cycles between a read and a write operation

These parameters are programmed in terms of ECLKOUT cycles. Separate set-up, strobe and hold timing parameters are available for read and write accesses [6-4]. Minimum values for asynchronous data transfer are as follows:

- SETUP $\geq$1 (0 treated as 1)
- STROBE $\geq$ 1 (0 treated as 1)
- HOLD $\geq$ 0

Therefore the maximum sampling rate possible, with no additional time delays present, is 50MSPS when the read setup and read strobe are both set to one clock cycle and read hold is set to zero cycles. Similarly, when transmitting data only, the write setup and write strobe are both set to one clock cycle and write hold is set to zero cycles to achieve 50MWPS.

The asynchronous MTYPE definitions for the processor are:

- MTYPE = 0000b: 8-bit-wide asynchronous interface
- MTYPE = 0001b: 16-bit-wide asynchronous interface
- MTYPE = 0010b: 32-bit-wide asynchronous interface

Therefore, the follow two lines of code in Figure 6.5 set the CE3 control register to 32-bit asynchronous memory mode with a theoretical read transfer data rate of 50MSPS.

```
#define EMIF_CE3    0x01800014
*(int *)EMIF_CE3 = 0x00010120;
```

Figure 6.5: Code to define and configure the EMIF CE3 control register

Unfortunately this theoretical limit is not realised, as external memory interface delays of less then one cycle are incurred before a read or write command is issued. However a more reliable transfer rate of 25MSPS is achieved with negligible delays being observed as, for example, when the read setup/strobe/hold values are set to 1/2/1 cycles respectively.

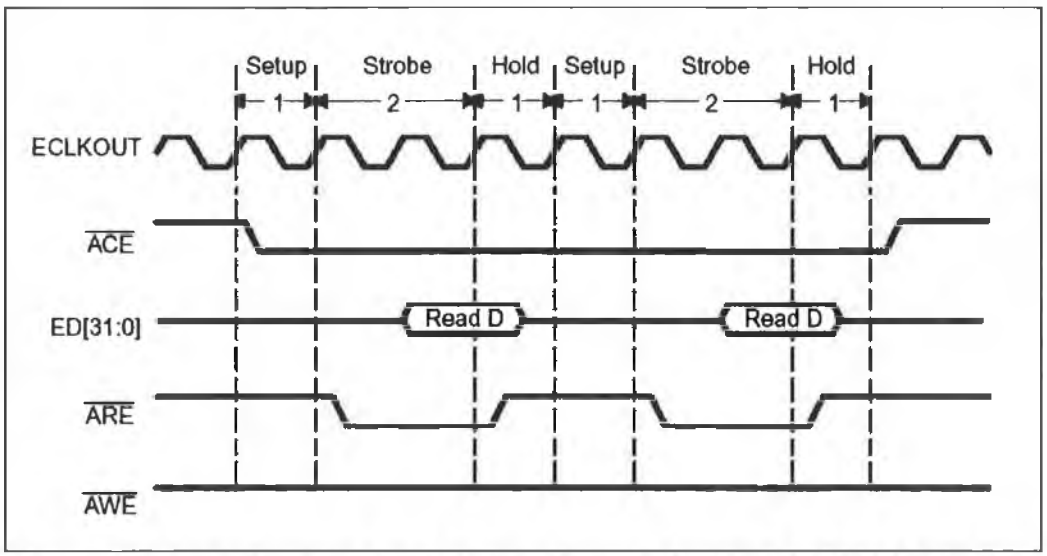


Figure 6.6: Asynchronous read timing diagram for the TMS320C671x DSPs

The timing diagram of Figure 6.6 illustrates the sequence of events that are critical to successfully transferring data through the EMIF. Here the asynchronous read with the setup, strobe and hold parameters are programmed with the values 1, 2 and 1 respectively. At the beginning of the setup period $\overline{\text{ACE}}$ becomes active (low). Then one clock cycle later, at the beginning of the strobe period, $\overline{\text{ARE}}$ becomes active. At the beginning of a hold period, $\overline{\text{ARE}}$ becomes inactive (high). Data is sampled on the

ECLKOUT rising edge concurrent with the beginning of the hold period (the end of the strobe period) and just prior to the $\overline{ARE}$ low-to-high transition [6-4]. The $\overline{ACE}$ signal goes high just after the final programmed hold period. Throughout the read sequence the $\overline{AWE}$ signal remains inactive (high).

The $\overline{AWE}$ signal becomes active (low) when the data stream direction is reversed from internal to off-chip memory. In the asynchronous write case a complementary sequence occurs with the $\overline{ARE}$ and $\overline{AWE}$ waveform active and inactive periods being reversed. As can be seen in Figure 6.1, the $\overline{ACE}$, $\overline{ARE}$ and $\overline{AWE}$ control signals are applied to a logic device on the daughter-board. The signals that result control the flow of data to and from the DAC/ADC devices respectively.

With the EMIF and EDMA's relevant registers and software code configured for high-speed data rates, advanced algorithms can be introduced and then applied to process and analyse the received high-speed signals.

6.2 Digital Filter Algorithms

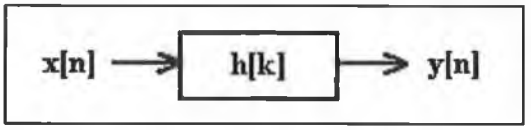


Figure 6.7: A discrete-time system

One of the many useful applications of DSP's is the real-time implementation of digital filters. A digital filter can be viewed as a discrete-time (DT) system, as shown in Figure 6.7, where an input signal x[n] is filtered by a system with the DT impulse response h[k], producing an output signal y[n]. The input and output signals to the filter are related by the convolution sum, which is given in Equation 6.1 for the FIR and 6.2 for the IIR filter.

$$y[n] = \sum_{k=0}^{N-1} h[k] * x[n-k] \qquad (6.1)$$

$$y[n] = \sum_{k=0}^{\infty} h[k] * x[n-k] \tag{6.2}$$

It is evident from these equations that, for IIR filters, the impulse response is of infinite duration whereas for FIR is of finite duration, since h[k] for the FIR has only N values. In practice, it is not feasible to compute the output of the IIR filter using Equation 6.2 because the length of its impulse response is too long (infinite in theory). Instead, the IIR filtering equation is expressed in a recursive form:

$$y[n] = \sum_{k=0}^{\infty} h[k] * x[n-k] = \sum_{k=0}^{N} b[k] * x[n-k] - \sum_{k=1}^{M} a[k] * y[n-k] \tag{6.3}$$

Equation 6.3 is the required difference equation to represent the output of an IIR DT filter system in the time domain, where a[k] and b[k] are the coefficients of the filter. In Equation 6.3, the current output samples (y[n]) is a function of past outputs as well as present and past input samples, therefore the IIR is a feedback system of some sort. This should be compared with the non-recursive FIR equation where the current output sample (y[n]) is a function only of past and present values of the input [6-5]. To find the values of h[k], for FIR, or a[k] and b[k], for IIR, from a set of given design parameters a computer programme such as Matlab is used. Once the filter coefficients have been calculated the next design stage involves the representation of the filter by a suitable structure, which is termed realisation.

Realisation involves converting a given DT system transfer function, H(z), into a suitable filter structure, where H(z) is the z-transform of the DT impulse response (h[n]). Block or flow diagrams are often used to depict filter structures and they show the computational procedure for implementing the digital filter. The three structures commonly used are the direct, cascade and parallel forms. Figure 6.8 shows a general illustration of these forms.

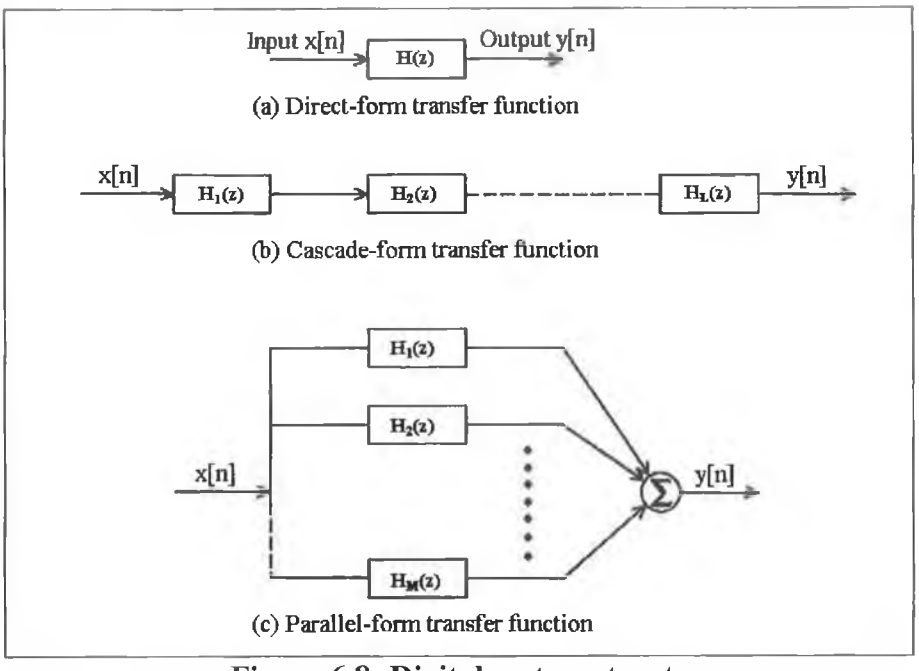


Figure 6.8: Digital system structures

The parallel and cascade structures are most widely used for IIR filters. This is because they lead to simpler filtering algorithms and are far less sensitive to the effects of implementing the filter using a finite number of bits with respect to the direct structure. This point is expanded upon in greater detail in Section 6.2.2, where the cascade form is found to perform closest to its unquantised result when relatively high order filters with finite wordlengths are analysed.

Digital filtering algorithms have the ability to improve the quality of a signal by removing or reducing any noise observed. The contamination of a signal of interest by other unwanted, often larger, signals or noise is a problem encountered in many applications. Where the signal and noise occupy fixed and separate frequency bands, conventional linear filters with fixed coefficients are normally used to extract the signal.

Depending upon a given application, different types of processing algorithms are selected. This is particularly true when digital filtering is involved. The choice between FIR and IIR filters depends largely on the relative advantages of the two filter types. FIR filters can have an exact linear phase response. The implication of this is that no phase distortion is introduced into the signal by the filter. This is an

important requirement in many applications, for example data transmission, digital audio and image processing. The phase responses of IIR filters are non-linear, especially at the band edges and the stability of IIR filters cannot always be guaranteed unlike FIR filters.

FIR requires more coefficients for sharp cut-off filters than IIR. Thus for a given amplitude response specification, more processing time and storage will be required for a FIR implementation. Therefore for real-time, high-throughput, application requirements, IIR filters should be used. However, if the number of filter coefficients is not too large and, in particular, if little or no phase distortion is desired, FIR is the ideal option [6-5].

6.2.1 FIR Filter Design and Implementation

Through Matlab's signal processing toolbox, digital filtering functions calculate coefficients according to the specification. These coefficients represent and describe the designed filter. With the coefficients determined, the important filter properties can easily be obtained, analysed and compared to specification. General filter properties of interest include; frequency and impulse responses, phase plots while for IIR filters stability information is critical.

In Matlab the *fir1( )* function can calculate suitable filter coefficients once the filter order and cut-off frequency corresponding to half the sampling rate have been specified. The returned coefficients must be converted to Q.15 format and then, for convenience, stored in a separate C assessable file. By defining the coefficients in a separate file, the main programme can load the coefficients to the desired memory location when the programme is run.

Figure 6.9 illustrates the characteristics of a low-pass FIR filter generated in Matlab, with a specified cut-off frequency of 2MHz. The two frequency response plots comprehensively confirm the specified cut-off frequency at 2MHz that is defined as the frequency at which the gain level has been attenuated to half (-6dB) the passband level. This traditional cut-off value of –6dB is associated with all Matlab FIR filter

designs, whereas IIR (including the Butterworth filter design of Section 6.2.2) use the –3dB cut-off level. As expected, within the passband a linear phase response is observed while the impulse response is finite, symmetrical and stable.

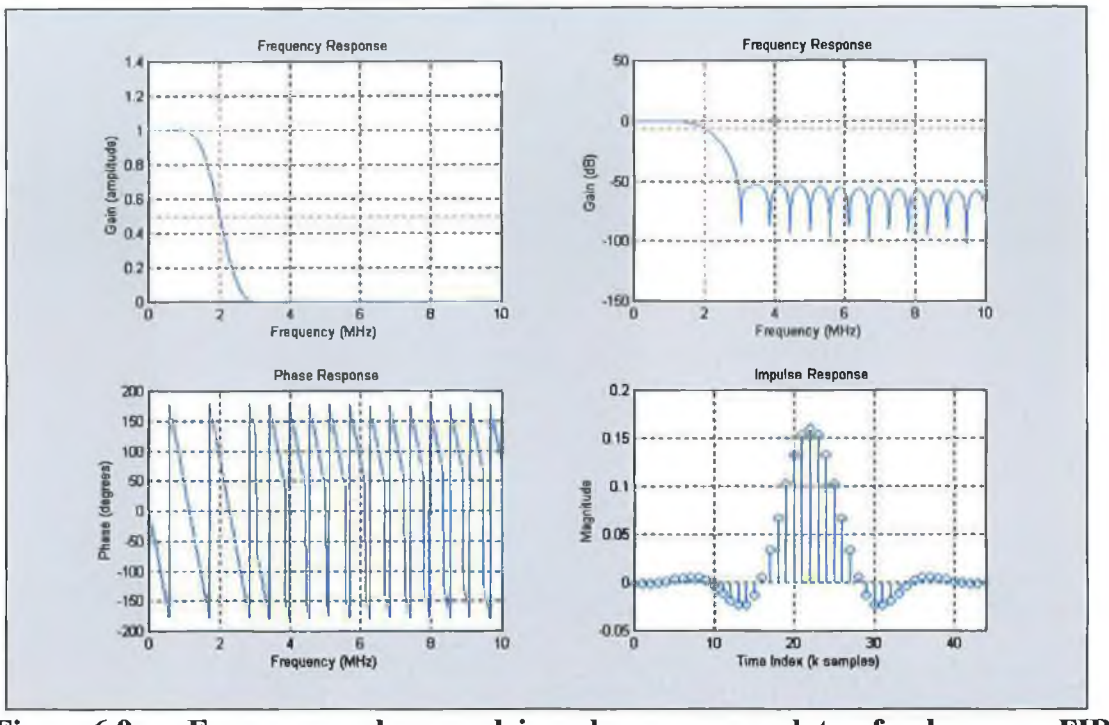


Figure 6.9: **Frequency, phase and impulse response plots of a low-pass FIR digital filter with a designed 2MHz cut-off frequency (-6dB)**

The sequence of events, from filter specification and calculating coefficients to analysing the filter properties prior to implementation can be entirely incorporated into a single Matlab (m.file) programme. This method allows for rapid altering of a specific filter property such as its frequency response.

The coefficients of the filters were computed within Matlab, according to the given requirements, and were saved in Q.15 format to be represented with the *short* type in the DSP. The function for the implementation of the FIR filter was named *fir_filter( )*. The ANSI C code for this function is presented in Figure 6.10.

Under the control of the EDMA, the input of the function is a sample from the EMIF and is in Q.15 format. The most recent sample enters the filter delay line as a global variable called *R_in*, while each previous sample of the delay line is shifted by one place. The output (filtered) sample is also in Q.15 format. After each multiplication

(multiple and accumulate operation) the result must be shifted 15 places to the right to avoid overflow, because the samples and the filter coefficients are in Q.15 format. A *short* data type is also required for the output result. To reduce the quantisation error a 32-bit accumulator is used (*int* data type in the TMS320C6711) [6-6].

```
/*
  ----------------------------------------------------------------------------
  Function:     fir_filter()
  Description:  Sample by sample FIR filtering
  Inputs:       input   =    input sample to the filter
                shift   =    defines the Q.N format, shift the product right by 15 bits
  Outputs:
  Returns:      temp    =    filtered sample
  ----------------------------------------------------------------------------
*/

short fir_filter (short input, int shift)
{
   int i;
   short temp;
   int Acc;

   for (i=COEF-1; i>0; i--)
     R_in[i]=R_in[i-1];                          /* Shift delay samples       */
   R_in[0] = input;                              /* Update most recent sample */

   Acc = 0;
   for (i=0; i<COEF; i++)
      Acc += (int)(short)h[i] * (int)(short)R_in[i]; /* Sum operation in C      */
   temp = (short) (Acc>>shift);                  /* Output in 16-bit format   */

   return temp;

} /* end fir_filter */
```

Figure 6.10: C code function used to implement a FIR filter

The full programme (*data_in_fir+_pcb.c*) with initialisation and transfer routines is included in the Appendix Section A.4.1, with corresponding FIR filter coefficients (*fir_coeffv45_q15_fc2meg_fs25msps.c*) listed in Appendix Section A.4.2.

6.2.2 IIR Filter Design and Implementation

It should be noted that the frequency response of a digital filter cannot always be guaranteed. A source of performance degradation in digital filters can be caused by coefficient quantisation. However, the effects of using a limited number of bits to implement filters are much less severe in FIR than in IIR. Therefore when implementing an IIR filter, the type of structure used is highly important. Because of this, a Matlab programme was written to investigate and test the direct, cascade and parallel structure forms that are generally realised when implementing an IIR filter.

The findings of this investigation proved the cascade and parallel structures are far less sensitive to the effects of implementing a filter using a finite number of bits compared to the direct form. For the direct form, the system becomes unstable and as a consequence the actual frequency response varies from its desired unquantised response which is an unacceptable result. By examining the band-pass filter plots of Figure 6.11, this conclusion is confirmed. The cascade form is found to be the most reliable structure as the level of quantisation is limited to 8-bits. Therefore, the cascade filter form was the structure used to implement the IIR filter function.

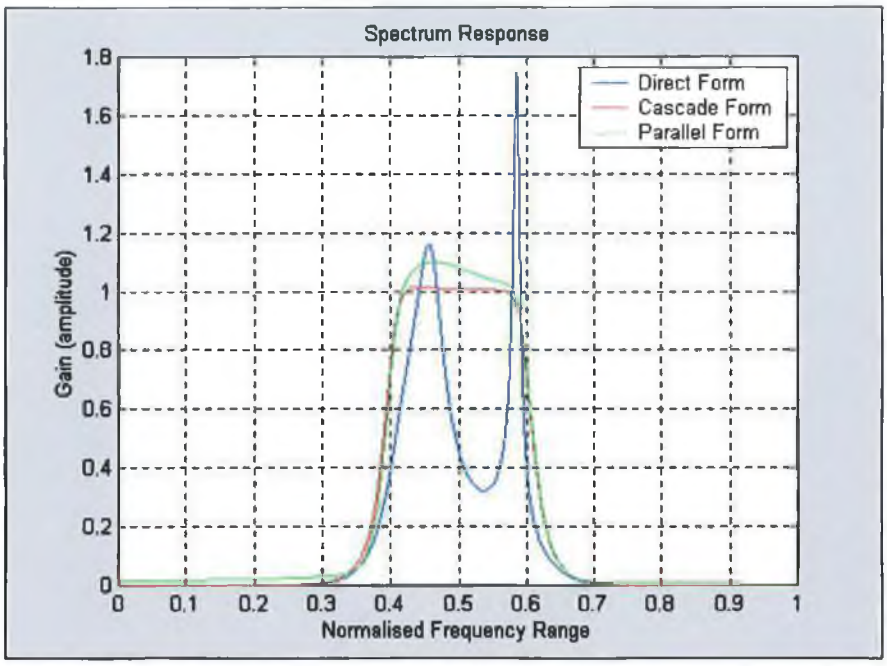


Figure 6.11: Normalised frequency response of a direct, cascade and parallel form 12th order band-pass IIR filter quantised to 8-bits

The selected IIR filter properties, in cascade form, are compared to specifications prior to implementation. For example, a practical low-pass Butterworth filter with a cut-off frequency of 2MHz (-3dB) is designed with the *butter( )* function. Butterworth filters are characterised by a magnitude response that is maximally flat in the pass-band and monotonic overall. The *tf2sos( )* function is subsequently needed to convert the transfer function representation of the Butterworth filter into an equivalent second-order section (SOS) representation. Note that a specified 8th order Butterworth filter requires four SOS stages. Each cascade form SOS coefficient is appropriately quantised to a 16-bit number with an additional sign bit. Once quantised the filter's

related spectrum, phase and stability information are analysed to ensure they remain within specification. The primary functions used to complete this visual analysis are, respectively; *freqz( ), angle( ), grpdelay( ), filter( ) and tf2zp( ).*

The absolute gain frequency response plot, of Figure 6.12, indicates no variation from the specified cut-off frequency that is defined for this filter as the frequency where the passband response falls by –3dB.

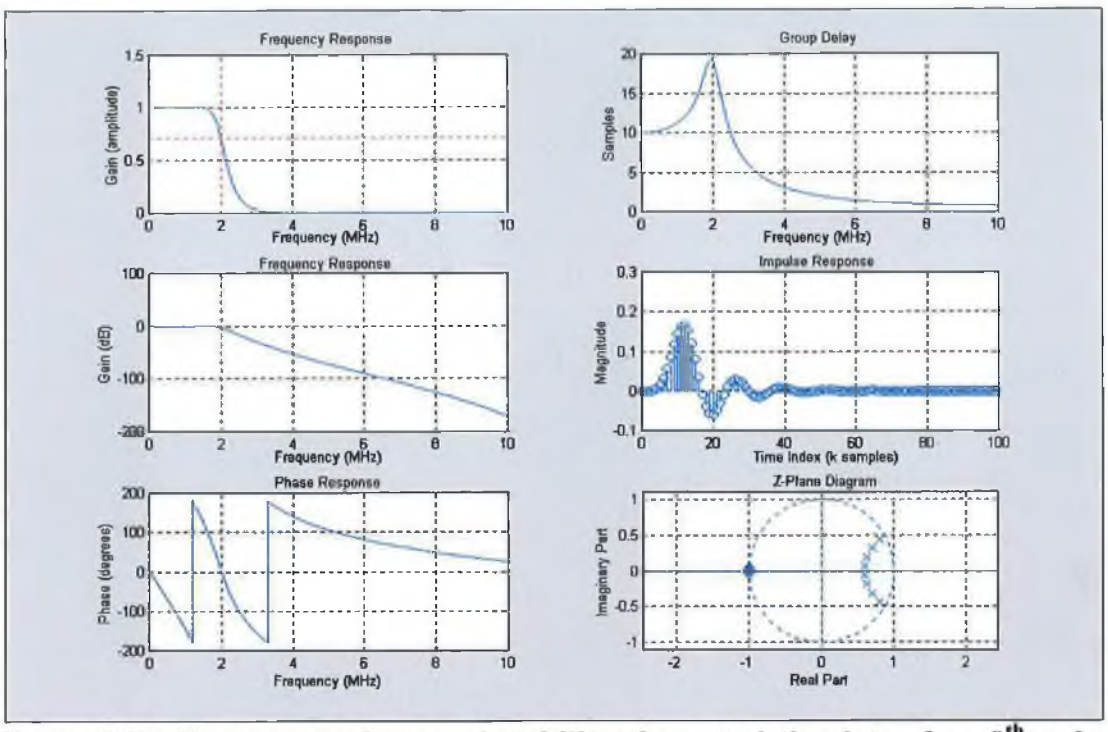


Figure 6.12: Frequency, phase and stability characteristic plots of an 8th order low-pass IIR cascade form digital filter with a designed 2MHz cut-off frequency (-3dB)

There are two phase-related plots included in Figure 6.12: phase response and group delay. When signals pass through a filter, they are possibly modified in amplitude and/or phase. The nature and extent of a signal's modification are dependent on the amplitude and phase characteristics of the filter. The phase shift (response), or group delay of the filter, provides a useful measure of how the filter modifies the phase characteristics of a signal.

If a signal consists of several frequency components, the phase delay of the filter is the amount of time delay each frequency component of the signal suffers in going

through the filter. The group delay, on the other hand, is the average time delay the composite signal suffers at each frequency. It is the negative first derivative of a filter's phase response. The phase response in this case is simply a function of the filter's length that relates to the filter coefficients. To get a zero phase or group delay response the delay is expressed in terms of the number of coefficients and so any correction can be made.

Unlike the FIR filter, the phase response of the IIR digital filter is non-linear, especially at frequencies near its cut-off value as shown in Figure 6.12. Therefore, this phase characteristic of the filter will have the greatest modifying effect on signal components passing frequencies near the filter cut-off frequency. If a signal being filtered is required to have the minimum possible phase distortion, the signal's frequency range should be maintained well inside the filter's passband range. Plots displaying phase characteristics, such as group delay, are invaluable tools used to confirm acceptable limits for a digital filter.

There are two general types of results that can be plotted to show stability properties of a digital filter system. The first being the impulse response, h[k], that completely defines the DT system in the time domain. A Linear Time Invariant (LTI) system is stable if its impulse response satisfies the condition of Equation 6.4.

$$\sum_{k=-\infty}^{\infty} |h[k]| < \infty \tag{6.4}$$

This condition is therefore satisfied if h[k] is of finite duration or if h[k] decays towards zero as k increases. Such a plot allows the designer or observer to check the stability of a system through software simulation before any hardware implementation is initiated. Several passes may be required as trade-offs are made between design specification tolerances and the processing power available. Moreover, the software implementation of a digital filter algorithm is often carried out to verify that the algorithm chosen does indeed meet the goals of the application on hand before the algorithm is implemented. The impulse response of the quantised filter of Figure 6.12 is clearly converging towards zero as the number of DT samples increase, therefore the filter can be said to be stable even when quantised.

The second method of displaying appropriate stability details is in a zero-pole plot that is another equally important analysis tool for accurate studies of digital filter systems. The z-plane diagram of Figure 6.12 confirms the respective system's stability, where all the quantised poles lie within the unit circle. Based on the plots of Figure 6.12, the conclusion is made that the cascade form structure remains within specifications and stable even when the IIR filter's coefficients are quantised.

The IIR filter's strength comes from the flexibility the feedback arrangement provides. For example, IIR filters normally require fewer coefficients than a FIR filter for the same set of specifications, which is why IIR filters are used when a sharp cut-off or a high-throughput are the important requirements [6-7].

The cascade form, displayed in Figure 6.13, is a biquadratic structure with second order building blocks. This structure has become very popular and is used predominately for practical high-order transfer functions. Figure 6.13 is a more detailed version of Figure 6.8(b).

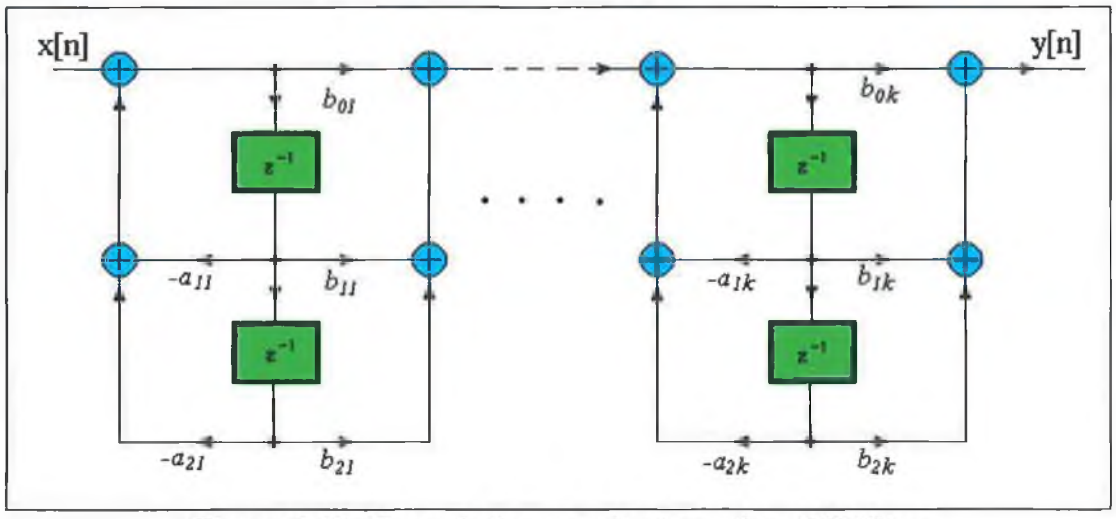


Figure 6.13: Cascade form realisation of an IIR filter

Conversion from the direct form to a cascade structure involves factorising the polynomials such that the product of the individual transfer functions in Figure 6.8(b) is equal to the single transfer function in Figure 6.8(a). In Matlab the desired

biquadratic sections are contained within a matrix, where each row (k) contains the numerator and denominator coefficients b_{ik} and a_{ik} of the SOS of H(z):

$$H(z) = \prod_{k=1}^{M} H_k(z) = \prod_{k=1}^{M} \frac{b_{0k} + b_{1k}z^{-1} + b_{2k}z^{-2}}{1 + a_{1k}z^{-1} + a_{2k}z^{-2}} \tag{6.5}$$

The function that implements the SOS cascade IIR filter is named *iir_cas5( )*, and is shown in Figure 6.14. This function is designed for 5 coefficients per biquadratic section. The algorithm's format allows *int* data type coefficients to be implemented unlike the FIR filter. This modification ensures that saved coefficients, with an absolute value greater than 1, are not a source of error when applied.

```
/*
  ----------------------------------------------------------------------------
  Function:    iir_cas5()
  Description: Second Order Section cascade IIR filter, 5 coefficients per biquadratic section,
               each coefficient in Q.16 format, 16 magnitude bits and 1 sign bit, total of 17 bits.
  Inputs:      input    =    input sample to the filter
               c        =    coefficients of the total cascade configuration
               d        =    delay states of the total cascade configuration
               n        =    number of the second order sections
               shift    =    defined by the Q.N format (minus 1), shift the product right by 15 bits
  Outputs:     -
  Returns:     temp     =    filtered sample
  ----------------------------------------------------------------------------
*/

int iir_cas5(int input, int *c, int *d, int shift, int n){

    int k0;
    int temp;
    int i;

    temp = input;
    for (i=0; i<n; i++){
      k0   = temp - ((c[5*i+1]*d[2*i+1])>>shift) - ((c[5*i+0]*d[2*i+0])>>shift);
      temp = ((c[5*i+4]*d[2*i+1])>>shift)+((c[5*i+3]*d[2*i+0])>>shift)+((c[5*i+2]*k0)>>shift);

      d[2*i+1] = d[2*i+0];
      d[2*i+0] = k0;
   }

   return temp;

} /* end iir_cas5 */
```

Figure 6.14: C code function used to implement an IIR filter

Each input sample is considered in sequence, as the most recent sample is reassigned the local variable *temp*. To avoid overflow a suitable shift is performed after each multiplication. Each sample is filtered and then returned to the main programme, which calls this filter function, while previous delay line samples are shifted appropriately. The IIR filter based C programme is similar to the one developed to implement the FIR filter, with initialisation, memory allocation and global variables being the principal differences. The full IIR filter based programme (*data_in_iir+_pcb.c*) can be seen in the Appendix Section A.4.4, with corresponding

cascade form IIR filter coefficients (*iir_cas5_stage4_fc2meg_fs25msps.c*) listed in Appendix Section A.4.5.

6.3 Spectral Analysis FFT Implementation

An important application of the FFT is for spectral analysis. The method used to implement a FFT function was the decimal in frequency (DIF) technique. The process of dividing the frequency components into even and odd parts gives this algorithm its name, DIF. More specifically, a 512-point complex radix-2 FFT algorithm, *radix_2( )* is called. For the implementation of the radix-2 algorithm the weighting factor (conventionally referred to as 'twiddle factors') must be given as inputs. These factors are generated with Matlab and are saved in Q.15 format, in a header file.

```
/*
  ---------------------------------------------------------------------------
  Function:         radix_2()
  Description:      Radix-2 FFT implementation
  Inputs:           n       =   number of points in the FFT
                    x       =   array with complex samples
                    w       =   array with twiddle factors
  Outputs:          x       =   array with the FFT complex samples
  Returns:          None
  ---------------------------------------------------------------------------
*/
void radix_2(short *x, short n, short *w)
{
          short n1,n2,ie,ia,i,j,k,l;
          short xt,yt,c,s;

          n2 = n;
          ie = 1;
          for (k=n; k > 1; k = (k >> 1) ) {
            n1 = n2;
            n2 = n2>>1;
            ia = 0;
            for (j=0; j < n2; j++) {
              c = w[2*ia];
              s = w[2*ia+1];
              ia = ia + ie;
              for (i=j; i < n; i += n1) {
                l = i + n2;
                xt      = x[2*l] - x[2*i];
                x[2*i]  = x[2*i] + x[2*l];
                yt      = x[2*l+1] - x[2*i+1];
                x[2*i+1]    = x[2*i+1] + x[2*l+1];
                x[2*l]      = (c*xt + s*yt)>>15;
                x[2*l+1]    = (c*yt - s*xt)>>15;
              }
            }
            ie = ie<<1;
          }

          return;
} /* end radix_2 */
```

Figure 6.15: C code function used to implement a radix-2 FFT

Figure 6.15 shows the C function, which a main programme calls, to determine the spectrum response of a signal. When called, this *radix_2( )* function calculates the

FFT of its input buffer x that contains complex-valued time-domain samples. The parameter n contains the number of data values in x. Overwriting the input buffer x returns the complex-valued output spectrum, to produce X[k]=DFT{x[n]}.

Combining this FFT function with a digital filter based programme adds another dimension to the signal processing system. Therefore, with the addition of this function, a system is capable of analysing the spectrum of an input signal in real-time.

6.4 Visual Analysis and Design Verification

Code Composer Studio (CCS) allows a user to create and test real-time signal processing programmes through a PC, while providing an interface link with an embedded DSP system. A primary function of any DSP system is to extract information from a signal after the quality of the original, possibly noisy, signal has been improved typically by digital filtering. The performance of these implemented filters have been verified in the CCS application. To complement the visual and data analysis facilities, which the CCS provides, powerful Matlab analysis based programmes were also developed.

6.4.1 Real-Time CCS Application

Both FIR and IIR digital filters that pass signals in the MHz range were specified, realised and analysed in Matlab. An application that provides a suitable platform to verify the performance of these implemented filter functions with actual test signals is the CCS. In this code intensive environment, C computer language programmes were developed that incorporated transfer control, digital filtering, FFT, further signal processing and presentation functions together. By combining the processing power of these programmes, with the high-speed adaptable instrumentation of the daughter-board, a multi-functional data acquisition system was established.

In the CCS, a user creates a project and adds the necessary programme files. The project is then compiled as the files are linked. An executable file is generated and

downloaded to the processor and then run. For the primarily FIR based project/programme, documented in the appendices, five (source code and support) files were compiled:

- data_in_fir+_pcb.c — C language source code file listing the programme to service the EMIF (under the control of the EDMA) and digitally filter and analyse the received data (in Section A.4.1)
- fir_coeffv45_q15_fc2meg_fs25msps.c — C language source code file listing the FIR filter coefficients (in Section A.4.2)
- csl6711.lib — Chip support library file of the target DSP
- rts6701.lib — Run time library file of the target DSP
- c6711dsk.cmd — Command linker file (in Section A.4.3)

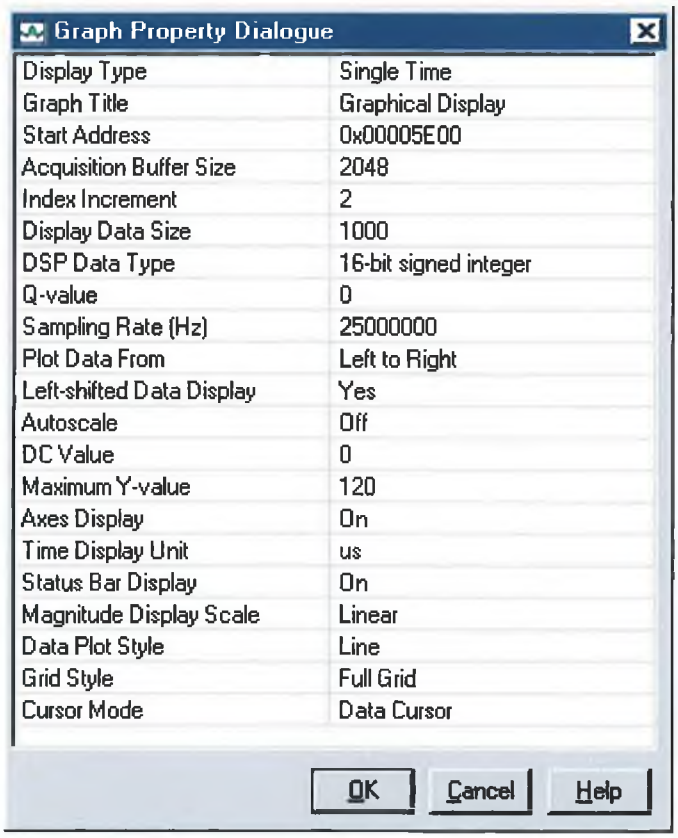


Figure 6.16: CCS's graph property dialogue box

A similar list exists for the IIR based project. Viewing the performance of either of these filtering programmes/functions is made possible by the CCS's graphical window. This window provides a variety of methods to graph data processed by a programme. For instance, Figure 6.17 illustrates the FIR low-pass filter's ability to minimise noise from a wanted signal plotted against time. This graphical window can be configured after choosing View→Graph→Time/Frequency in sequence from the Menu Bar. In the Graph Property Dialogue Box that appears, the Start Address, Acquisition Buffer Size, Index Increment, Display Data Size, DSP Data Type are the most important properties requiring change to the values shown in Figure 6.16.

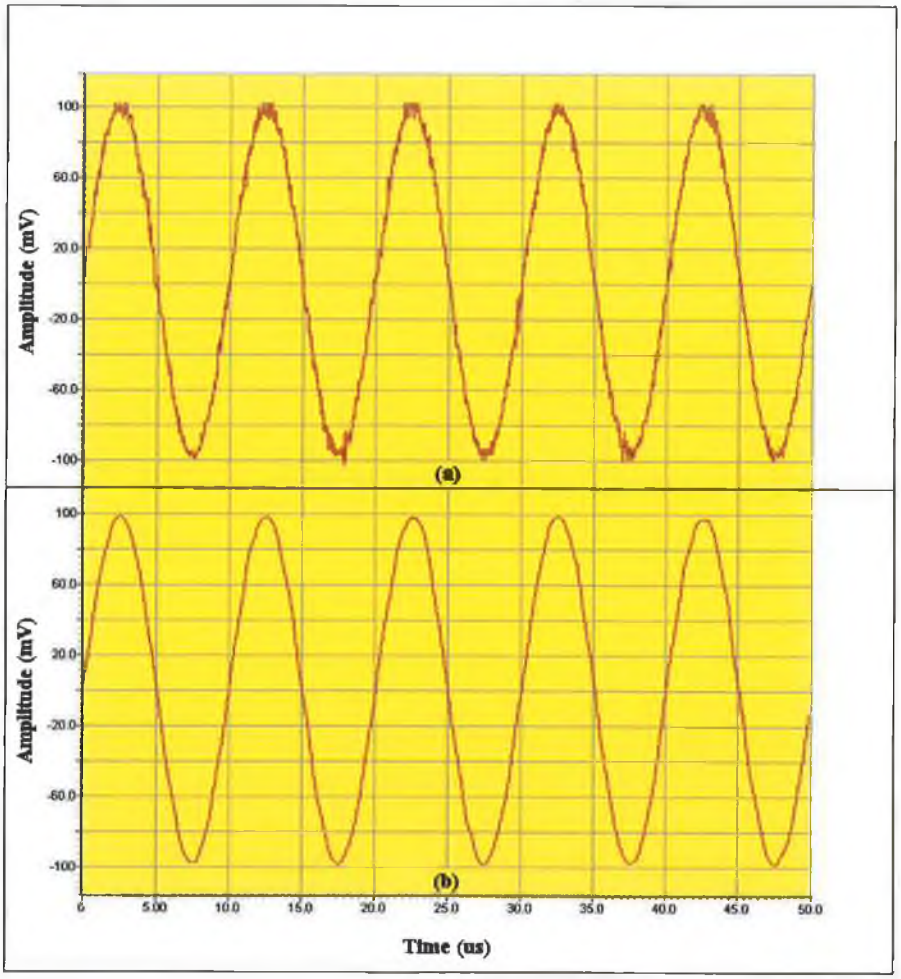


Figure 6.17: (a) Unfiltered and (b) filtered 100kHz frequency, 200mVp-p (approximately) input test signal

After the high frequency distortion has been removed, important input signal properties are more accurately analysed. The signal analysis functions can be configured to measure a range of application dependent dc and/or ac properties. By

using the standard output (stdout) window feature these results are numerically displayed. Figure 6.18 indicates the frequency and amplitude results of the filtered signal in Figure 6.17(b), as the *freq_anal( )* and *volt_anal( )* respective functions are called by the (*data_in_fir+_pcb.c*) main programme. The input channel number (with associated gain) and units of each property are also presented.

```
Ch#2(x5) Signal Results

Frequency(kHz):
100.00
Vp-p(mV):
198
```

Figure 6.18: Numerical data displayed through the standard output window

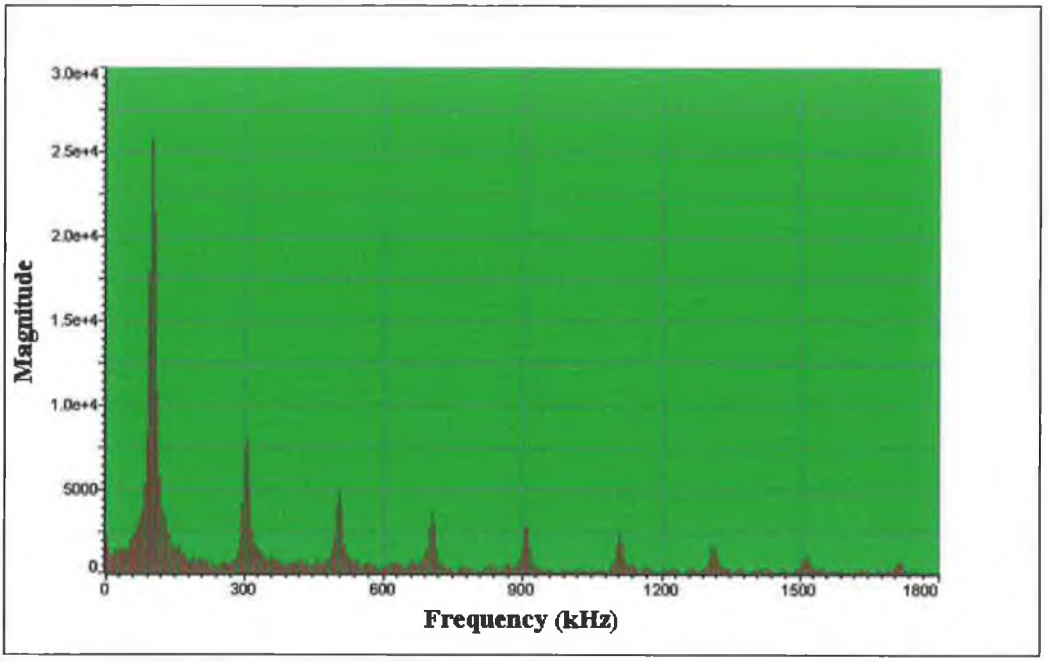


Figure 6.19: CCS power spectrum plot of a square waveform with a fundamental frequency of 100kHz

Changing the Display Type to FFT Magnitude in the Graph Property Dialogue box causes a power spectrum plot to be displayed. For example, in Figure 6.19 the separate components of an input square waveform are shown. As expected the Fourier series frequency components of a square waveform match the peak values shown in this plot. However, the corresponding power magnitude values are dependent on the FFT options selected. In particular, adjusting the FFT order and Windowing function

(Rectangular, Hamming, etc.) properties cause the conversion factor from power magnitude to a corresponding voltage level that are difficult to determine simply from the plot.

By implementing the user defined FFT algorithm discussed in Section 6.3 alternative amplitude versus frequency plots are available, as the CCS's FFT option is avoided. Further improvement to the spectrum and data analysis results are possible by transferring the recording data into a Matlab readable *.dat* format file.

6.4.2 Advanced Matlab Software Tools

To transfer data from the CCS's real-time environment to Matlab's improved presentation and detailed data analysis environment, the follow steps are required:

- Select File→Data→Save… from the CCS's Menu bar
- In the Store Data dialogue box, which appears, enter an appropriate filename and save the file type as Hex (*.dat) and then click Save
- In the next Storing Memory into File dialogue box, enter the starting memory address of the variable(s) to be converted followed by it's length in the appropriate hexadecimal format and finally click OK.

Once in Matlab, each vector variable is converted to a decimal form using the *hex2dec( )* function and then loaded into the workspace. For comparison purposes, a related waveform to the one sampled is easily superimposed, into any plot, while Matlab's *fft( )* function results are also presented in Figure 6.20(b) for the sampled and ideally generated signals. Simultaneously, the waveform's frequency components are calculated and tabulated in the workspace as Table 6.1 indicates.

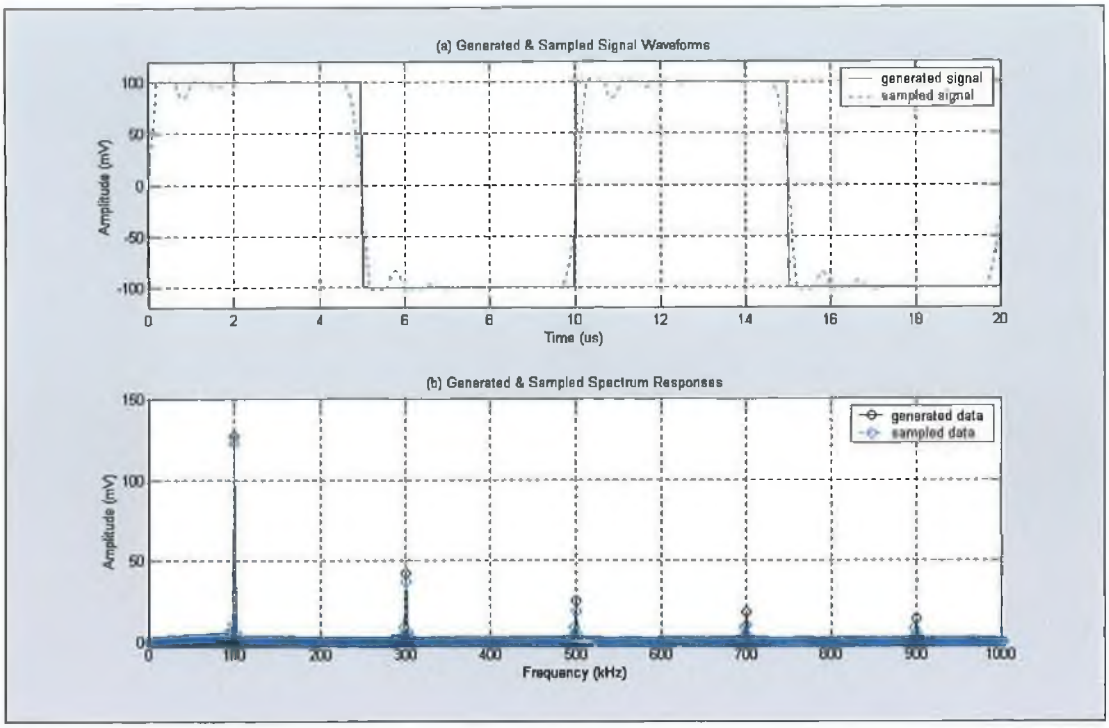


Figure 6.20: (a) Time and (b) frequency domain plots of Matlab generated and input sampled signals

Frequency Components (kHz)	Amplitude (mV)	
	Expected (Generated) Value	Actual Sampled Value
100.0000	127.3245	122.7463
300.0000	42.4429	37.0143
500.0000	25.4673	18.8993
700.0000	18.1927	10.6895

Table 6.1: Fourier analysis results of generated and sampled square waveforms

Other periodic and non-periodic waveforms, with multiple frequency components, are handled equally well by the same programme, to determine the frequency components of the square waveforms, in Figure 6.20. The Fourier series equation that relates to this square shaped waveform is:

$$f(t) = \sum_{n=1(odd)}^{n=\infty} \frac{4A}{n\pi} \sin(2 \cdot \pi \cdot n \cdot v_1 \cdot t) \qquad (6.6)$$

Where A is the amplitude of the resultant waveform (100mV), v_1 is the fundamental frequency (100kHz) and only odd harmonics (odd multiples of the fundamental frequency) are required. When the generated square waveform's frequency components are decomposed, the amplitude values equal the calculated values associated with Equation 6.6. However, due to sampled signal length being truncated and with spectral leakage effects being observed across neighbouring discrete frequencies, the amplitude of it's fundamental frequency and odd harmonics are slightly less than their expected levels.

This additional Matlab analysis option provides a quick and easy method of testing the system's performance, as evasive measures such as zero padding or adjusting the frequency resolution to compensate for a truncated signal are possibly deemed necessary. Any data stored by the DSP can be transferred into Matlab, which is particularly useful when comparing transmitted digital signals (converted into analogue form by the DAC) to data received through the ADC, as diagnostic functionality is added to the data acquisition (data generation) system. The phase delay, associated with the daughter-board's instrumentation and memory interface, can also be determined through this procedure.

6.4.3 Phase and Diagnostic Functionality

```
Ch#2(x5) Signal Results

Frequency(kHz):
100.00
Phase Delay(ns):
00
Vp-p(mV):
200
```

Figure 6.21: Numerical data returned by diagnostic programme displayed through the standard output window

Through the daughter-board's DAC, high-speed generated signals are transmitted. Such signals can be used to control an actuator, or if applied to an input channel, can identify the phase delay associated with the system's instrumentation. Thus, the phase

analysis project/programme *data_out_in_phase+_pcb.pjt/.c* was developed. The results of the programme (Figure 6.21) guarantee a zero phase delay when an input signal of 100kHz is applied.

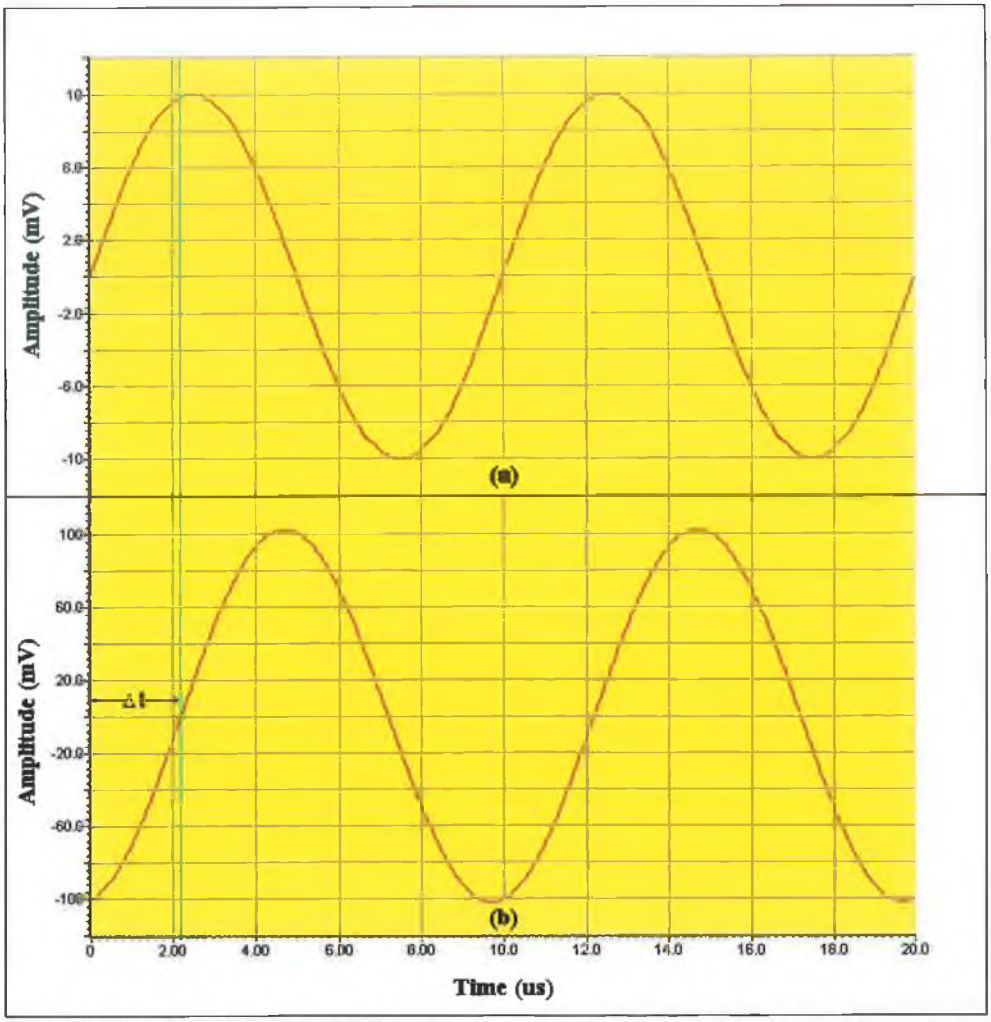


Figure 6.22: Phase difference observed between input (a) channel no.1 and (b) channel no.2

This type of phase analysis can be expanded to measure the phase difference between two independent input signals and displayed instantly in real-time. The name designated to this project/programme is *data_in_phase+_pcb.pjt/.c*. The type of visual and numerical results, returned by this programme, are presented in Figures 6.22 and 6.23 respectively. The time-lag (Δt) between the two signals is indicated in Figure 6.22, while it's corresponding phase value is calculated from Equation 4.2 and listed in Figure 6.23. A negative phase relates to the fact that channel no.2's input signal lags behind the input signal of channel no.1. The source code of these two phase

analysis programmes are documented in Appendix Sections A.4.6 and A.4.7 respectively.

```
Ch#1-Frequency(kHz):
100.00
Ch#2-Frequency(kHz):
100.00
Ch#1-Vp-p(mV):
20
Ch#2-Vp-p(mV):
203
Phase Diff.(deg.):
-79.20
```

Figure 6.23: Numerical data including phase difference displayed through the standard output window

6.5 Summary

The developed printed circuit daughter-board and interfaced DSP are fully supported by high-level C computer-language programmes. The system's ability to handle a range of different signals is achieved through these programmes, as the signal gain level and data transfer rate properties are controlled by software. Advanced programmer-defined signal processing algorithms are also implemented, in software, alongside real-time signal analysis functions. Because of these code intensive programmes an enhanced high-performance data acquisition (and data generation) system was realised.

Specifically, the DSP's EMIF control registers are optimally configured for high-speed data transfer rates, while the EDMA controller services all incoming and outgoing data streams to/from internal memory. The maximum theoretical transfer rate for the TMS320C6711 DSK mother-board is 50MSPS; however due to timing delays associated with the interface reliable high rates of 25/33MSPS are possible.

Digital filtering and FFT functions were used to improve the analytical ability of the system. For applications where little or no phase distortion is desired and if the number of filter coefficients needed are not too large, a FIR is recommended. If high-throughput and sharp cut-off filtering are important an IIR should be used. To

implement an IIR designed filter, the cascade form structure was employed. This is because it leads to a simple filtering algorithm and its form is the least sensitive to the effects of implementing a filter using a finite number of bits. The DSP's architecture is tailored to provide exceptionally fast mathematical computations required by these functions as massive amounts of data are gathered and analysed in real-time.

Graphical and numerical data are easily viewed in the TI CCS environment, while further analysis is possible as a method of loading stored data into Matlab was developed. Typical signal properties that are calculated and then displayed include magnitude, frequency and phase results, while multiple waveforms can be presented simultaneously. Therefore, when combined, these two compatible programmable applications extend and enhance the resulting system's signal processing and analysis capabilities. For diagnostic purposes, test signals can be generated through the DAC and returned as an input signal, which is then analysed, to give a measure of the system's performance. When applied to an input channel, the synthesised signals can instantaneously identify the range and phase properties associated with the system's instrumentation. The DAC's transmitted signals can also be used to control an actuator if deemed necessary.

7

Project Review

The initial paragraph of the first chapter compares, in terms of adaptability and multi-functionality, the characteristics of a signal conditioning element to those of a body's nervous system. During the course of this project, this analogy became more apt as the high-performance signal conditioning elements were designed to interface with a powerful DSP to form a versatile data acquisition system.

The front-end analogue preconditioning module has the ability to receive a range of signals from various transducers of different shapes and sizes (illustrated in Figure 1.1 in Chapter 1). This module and the connected data converter elements are implemented on a reproducible PCB. A cost-effective system was established through a thorough selection process. Precision, speed and environmental tests on the critical components and circuits that were considered for implementation were systematically carried out.

The software support provided by the interfaced DSP has the ability to configure the instrumentation's gain and sampling properties, while advanced real-time analysis, filtering and presentation functions are utilised to match a particular application.

7.1 Discussion of Results

High-performance standards were set and achieved for the front-end composite amplifier module. Specific parameters include:

- wide operating frequency range from dc to 30MHz (high frequency band limit)
- high to low gain/attenuation level flexibility
- overall amplifier V_{OS} of $\leq \pm 0.1$mV
- typical I_B of $\leq \pm 2$pA
- negligible drift in set gain level over a temperature gradient of 0-40°C

An amplifier module that maintains these parameters can service a wide spectrum of transducers from low-speed thermoelectric to high-frequency electromagnetic devices. Various low-level voltage signals received by the amplifier are amplified without distortion, as one of the possible gain/attenuation levels (±500, 50, 5 and 0.5V/V) is selected, to a level that complements a typical ADC's input voltage range. A short list of suitable op-amps were found from selection guides and data sheets, while software modelling and physical tests determined the type of amplifier circuit design to be implemented on a prototype PCB. The result of these investigations established the composite amplifier design as a flexible arrangement that would preserve the high-performance standards. Precision VFA and high-speed CFA parts were combined to form multiple op-amp stage composite systems. Where appropriate, diode protection and feedback capacitor compensation components were also incorporated.

A compatible, easy-to-implement and cost effective data converter module was also implemented and tested. The selected ADC has a high sample rate of 40MSPS and a resolution of 10-bits. A combined offset voltage in the order of 10s of mV was calculated for the difference amplifier and ADC devices. To compensate for this offset, caused by the instrumentation, a calibration software programme was developed. When a MHz range signal was applied, a significant phase delay of approximately -50° was introduced by the instrumentation. Any delay could be an issue for a phase-based measurement system.

The low-cost and powerful TMS320C6711 DSK was selected to perform real-time signal processing. This platform features a 150MHz DSP capable of executing 900MFLOPS with 64-kbytes of internal memory and 4-Mbytes of SDRAM. A 32-bit external memory interface facility for peripheral data transfer and a parallel port connector for a PC are also available. Through the peripheral, glueless interface a printed circuit daughter-board was connected. The resulting daughter-board contained the amplifier and data converter modules, while the DSP mother-board's power and control lines were utilised by the connected daughter-board. A tight overall layout ensured the daughter-board's surface dimensions were limited to 14 x 9.5cm, while sensitive pins were isolated to prevent interference from surrounding pervasive signal sources.

Theoretically the DSP's EMIF and EDMA can be configured for an optimum transfer data rate of 50MSPS. However, due to timing delays associated with the interface, practical high rates of 25/33MSPS are achieved. This loss in performance can be overcome with a faster processor system.

Digital filtering and FFT algorithms formed central functions of various DSP code intensive programmes. These functions were incorporated to enhance the analysis ability of the real-time data acquisition system. For example, digital filters have the ability to improve the quality of a desired signal through noise reduction. Both FIR and IIR filter types were designed, analysed and implemented. Direct, parallel and cascade form IIR filter structures were investigated for coefficient quantisation effects. A spectrum response of these three filters indicated the cascade form to be the least sensitive structure for implementing a filter's coefficients with a finite number of bits. The performances of the implemented FIR and IIR filtering algorithms were found to be highly satisfactory in removing noise from signals of interest. However, FIR filters require more coefficients for sharp cut-off filters than IIR. Thus, for a given amplitude response specification, more processing time and storage is required for a FIR implementation. Therefore for real-time high-throughput application requirements, IIR filters should be used. Whereas if the number of filter coefficients is not too large and, in particular, if little or no phase distortion is desired a FIR filter is the more ideal option.

System diagnostic checks were performed by applied internally synthesised signals to each input channel. Voltage range limits were instantly confirmed. A zero phase delay was determined when a 100kHz sinusoidal reference signal was applied.

7.2 Overall Summary

To form a versatile high-performance data acquisition system, high-speed analogue conditioning and powerful processing elements are combined. Single and dual input channel operation is possible. Input voltage ranges and sampling rates are adjustable under software control. Bi-directional capability adds diagnostic functionality to the system. To allow access to a range of TI DSP mother-boards, the printed circuit daughter-board's interface is completely detachable. A minimised number of implemented components ensured that the system remained cost-effective. Digital filtering, data analysis and presentation algorithms form code intensive application supported programmes.

7.3 Concluding Remarks

The investigated gain selection system containing relays was unsuitable for a low cost compact design. Therefore, an alternative, more practical, two-input channel selectable system without switching components was implemented. To power the prototype daughter-board, using the least number of resources, a dual power supply is connected to the J8 junction on the DSK (Digital Signal Processing Kit) mother-board. Here the +15V and –5V lines are found, while the standard +5V line can be applied to either this or the DSK's J4 junction, but never to both at the same time.

The printed circuit daughter-board is estimated to cost under €100 to implement, this is the primary component cost apart from the DSP chip. This amount is considerably less than other lower specification daughter-boards commercially available. For example, the ATDSK1118 daughter-board from ATE Communications costs £300 sterling. Therefore, the conclusion can be drawn that the data acquisition system,

which has been presented, is clearly cost effective while also having high-performance and versatility.

A

Appendix

A.1 PIC-µC Implemented Calibration Programme

A.1.1 cal_prog.c

```
/*
-------------------------------------------------------------------------------------------------------------------------
Filename:       cal_prog.c
Description:    This programme initially performs a basic calibration routine, which will correct offset and/or
                sensitivity errors that are associated with the analogue preconditioning amplifiers, level-
                shifter and ADC devices. The code was tested on a PIC's ADC. The calibration method employs a
                linear regression process where a point at the start of the input range of operation (0V) is
                applied and it's corresponding digital value is recorded to complete the first step of the
                calibration process. The second recorded point of interest is measured near the end of the input
                range (5V). Therefore only two measurement steps are needed to automatically correct the devices
                offset and/or sensitivity errors present. The subsequent input voltage signals applied are
                analysed as their peak to peak (p-p) & mean properties are calculated and displayed with a LCD.

                PORTB bits 0-3 are connected to the LCD data bits D4-D7 (pins 11-14:high nibble)
                PORTD bit 5 is connected to the LCD RS input (register select)
                PORTD bit 6 is connected to the LCD EN bit (enable)
-------------------------------------------------------------------------------------------------------------------------
*/

/*
-------------------------------------------------------------------------------------------------------------------------
Include files, symbolic constants and global data variables defined
-------------------------------------------------------------------------------------------------------------------------
*/

#include "c:\ht-pic\include\pic1687x.h"
#include "c:\ht-pic\samples\delay.c"

static bit LCD_RS   @ ((unsigned)&PORTD*8+5);               // Register select
static bit LCD_EN   @ ((unsigned)&PORTD*8+6);               // Enable

#define LCD_STROBE  ((LCD_EN=1),(LCD_EN=1),(LCD_EN=1),(LCD_EN=1),(LCD_EN=0),(LCD_EN=0),(LCD_EN=0),(LCD_EN=0))

bank1 unsigned int Dig_temp,y[2];                           // Global variables moved into bank 1 of the RAM, since
bank1 double m,m_new,c,c_new,Mea_volt,ptmp2;                // bank 0 is close to capacity
bank1 float p_to_p;

/*
-------------------------------------------------------------------------------------------------------------------------
Function:       init_ioports()
Description:    The following code initialises the I/O Ports as inputs or outputs.
-------------------------------------------------------------------------------------------------------------------------
*/

void init_ioports(void) {

        PORTA = 0x00;           // PORTA pins initially =   00 0000b (all pins set low)
        PORTB = 0xF0;           // PORTB pins initially = 1111 0000b (RB0-RB3 set low, RB4-RB7 set high)
        PORTC = 0x30;           // PORTC pins initially = 0011 0000b (RC4 & RC5 pins set high to test PIC with LEDs)
        PORTD = 0x00;           // PORTD pins initially = 0000 0000b (all pins set low)

        TRISA = 0x3F;           // PORTA set as INPUTS
        TRISB = 0x00;           // PORTB set as OUTPUTS
        TRISC = 0x00;           // PORTC set as OUTPUTS
        TRISD = 0x00;           // PORTD set as OUTPUTS
}

/*
-------------------------------------------------------------------------------------------------------------------------
Function:       config()
Description:    This function configures the ADC converter set-up.
-------------------------------------------------------------------------------------------------------------------------
*/

void config(void) {

        ADCON1 = 0x08;          // 6 analogue pins set but only AN0/RA0 (1 pin) used
                                // RA3 = Vref+, RA2 = Vref-
                                // Left justified A-D result format selected
                                // 6 least significant bits of ADRESL are read as '0'

        ADCON0 = 0x41;          // Fosc/8 is the A-D conversion clock selected
                                // RA0/AN0, analogue channel 0 selected
                                // A-D conversion not in progress, converter module is on
}

/*
-------------------------------------------------------------------------------------------------------------------------
Function:       lcd_write()
Description:    Writes bytes to the LCD in 4-bit mode.
-------------------------------------------------------------------------------------------------------------------------
*/

void lcd_write(unsigned char c) {

        PORTB = (PORTB & 0xF0) |  (c >> 4);
        LCD_STROBE;
        PORTB = (PORTB & 0xF0) |  (c & 0x0F);
        LCD_STROBE;
        DelayUs(80);
}
```

```
/*
-------------------------------------------------------------------------------------------------------------
Function:       lcd_clear()
Description:    Clear and home the LCD.
-------------------------------------------------------------------------------------------------------------
*/

void lcd_clear(void) {

    LCD_RS = 0;
    lcd_write(0x1);
    DelayMs(4);
}

/*
-------------------------------------------------------------------------------------------------------------
Function:       lcd_puts()
Description:    Write a string of chars to the LCD.
-------------------------------------------------------------------------------------------------------------
*/

void lcd_puts(const char * s) {

    LCD_RS = 1;               // write characters
    while(*s)
    lcd_write(*s++);
}

/*
-------------------------------------------------------------------------------------------------------------
Function:       lcd_goto()
Description:    Go to the specified position.
-------------------------------------------------------------------------------------------------------------
*/

void lcd_goto(unsigned char pos) {

    LCD_RS = 0;
    lcd_write(0x80+pos);
}

/*
-------------------------------------------------------------------------------------------------------------
Function:       lcd_init()
Description:    Initialise the LCD - put into 4-bit mode.
-------------------------------------------------------------------------------------------------------------
*/

void lcd_init(void) {

    LCD_RS = 0;               // write control bytes
    DelayMs(15);              // power on delay
    PORTB = 0x3;              // attention!
    LCD_STROBE;
    DelayMs(5);
    LCD_STROBE;
    DelayUs(100);
    LCD_STROBE;
    DelayMs(5);
    PORTB = 0x2;              // set 4 bit mode
    LCD_STROBE;
    DelayUs(40);
    lcd_write(0x28);          // 4 bit mode, 1/16 duty, 5x8 font
    lcd_write(0x08);          // display off
    lcd_write(0x0F);          // display on, blink cursor on
    lcd_write(0x06);          // entry mode
}

/*
-------------------------------------------------------------------------------------------------------------
Function:       adc_read_dc()
Description:    The following function takes a digital representation of an input analogue dc signal and converts
                it into an appropriate mean voltage value.
-------------------------------------------------------------------------------------------------------------
*/

void adc_read_dc(void) {

    int i;
    float sum_Dig_temp, mean;
    for(i=1;i<=100;++i) {
        DelayUs(20);                    // Required acquisition time for the ADC to meet its specified
        ADGO = 1;                       // accuracy ADCON0 bit set
        while(ADGO)
             continue;                  // wait for conversion to complete
        Dig_temp = ADRESH;              // Reading the A-D result register into variables, needing to be
        Dig_temp = Dig_temp << 8;       // manipulated
        Dig_temp = Dig_temp + ADRESL;
        Dig_temp = Dig_temp >> 6;
        sum_Dig_temp += Dig_temp;
        DelayUs(24);                    // Before acquisition can begin again & after conversion has completed, a
    }                                   // 2.0*T(AD) delay is required, 8*Tosc = T(AD), therefore 2*T(AD) = 4us.
    mean = sum_Dig_temp/100;            // But per 10-bit conversion 12*T(AD) is the required min.
    Mea_volt = (mean-c_new)/m_new;
}
```

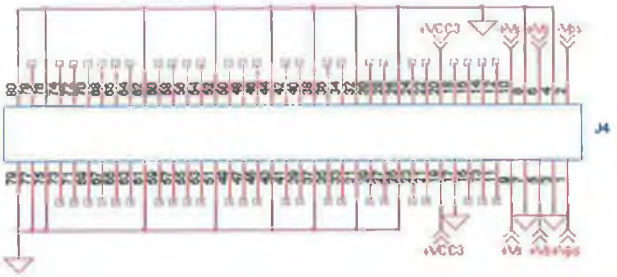
```
/*
-------------------------------------------------------------------------------
Function:         adc_read_ac()
Description:      The following function determines the peak to peak(p-p) voltage of an input ac signal.
-------------------------------------------------------------------------------
*/

void adc_read_ac(void) {

    int i;
    unsigned int min, max, Dig_temp1;
    DelayUs(80);                          // Required acquisition time for the ADC to meet its specified
    ADGO = 1;                             // accuracy, ADCON0 bit set
    while(ADGO)
          continue;                       // wait for conversion to complete
    Dig_temp1 = ADRESH;                   // Reading the A-D result register into variables, needing to be
    Dig_temp1 = Dig_temp1 << 8;           // manipulated
    Dig_temp1 = Dig_temp1 + ADRESL;
    Dig_temp1 = Dig_temp1 >> 6;
    max = Dig_temp1;
    min = Dig_temp1;
    DelayUs(60);
    for(i=1;i<=100;++i) {
          DelayUs(80);
          ADGO = 1;
          while(ADGO)
              continue;
          Dig_temp1 = ADRESH;
          Dig_temp1 = Dig_temp1 << 8;
          Dig_temp1 = Dig_temp1 + ADRESL;
          Dig_temp1 = Dig_temp1 >> 6;
          if(Dig_temp1>max)
              max=Dig_temp1;
          if(Dig_temp1<min)
              min=Dig_temp1;
          DelayUs(60);
    }
    p_to_p=(((float)(max-min))-c_new)/m_new;
}

/*
-------------------------------------------------------------------------------
Function:         display_dig_rep()
Description:      The next function indicates to the user the digitally represented value that is used to calibrate
                  the device.
-------------------------------------------------------------------------------
*/

void display_dig_rep (unsigned int D1) {

    char string[16];                              // See comments of display_cal_data() function below
    int i=0,j;                                    // for explanation of this functions code.
    unsigned int tmp,tmp1;
    tmp = D1;
    tmp1 = (tmp/1000);
    string[0] = tmp1 + '0';
    for(j=1;j<=100;j*=10) {
          tmp = (tmp-(tmp1*(1000/j)));
          tmp1 = tmp*j/100;
          i=i+1;
          string[i] =tmp1 + '0';
    }
    for (i=4;i<=15;++i)
          string[i] = ' ';
    lcd_puts(string);
}

/*
-------------------------------------------------------------------------------
Function:         display_cal_data()
Description:      The next function indicates to the user the actual calibrated data.
-------------------------------------------------------------------------------
*/

void display_cal_data(double ptmp1) {

    char string[16];                      // For the display to represent 3 significant figures before and 3 after
    int i=0,j,k,ptmp3;                    // the decimal point, the integer and real number variables are needed.
    ptmp2=ptmp1/100.0;                    // The present real ptmp2 value is converted to its integer
    ptmp3=(int)ptmp2;                     // representation with its digits below the decimal point being removed.
    string[0] = ptmp3 + '0';              // string[0] will then represent the ASCII code for the hundreds digit
    for(j=1;j<=10;j*=10) {                // The first for loop is used to display the tens and ones digits
        ptmp1 = (ptmp1-(ptmp3*(100.0/j))); // Within the outer most brackets the integer representation of the
    ptmp2 = (ptmp1/(10.0/j));             // hundreds digit is subtracted from the real value, which would result
    ptmp3 = (int)ptmp2;                   // in the tens value being the most significant number remaining,
    i=i+1;                                // therefore dividing by 10 and then finding the integer value will give
    string[i] =ptmp3 + '0';               // the suitable tens value only, the same loop continues to find the
    }                                     // ones digit.
    string[3] = '.';
    for(k=4;k<=6;++k) {                   // The second for loop performs a similar operation on the digits below
        ptmp2 = (ptmp2 - ptmp3*1.0);      // the decimal point,'0' is added to each digit to give their ASCII code
        ptmp2 = (ptmp2*10.0);             // values.
        ptmp3 = (int)ptmp2;
        string[k] = ptmp3 + '0';
    }
    for (k=7;k<=15;++k)
        string[k] = ' ';
    lcd_puts(string);                     // Finally another function is needed to write the string to the display
}
```

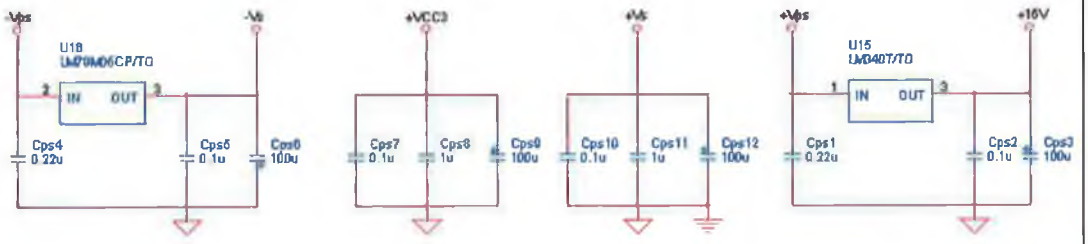
```
/*
-------------------------------------------------------------------------------------------------------------------
Function:         display_temp()
Description:      This function displays the measured voltage in an appropriate format type after the calibrated
                  data has been performed.
-------------------------------------------------------------------------------------------------------------------
*/

void display_temp(float V) {
    char string[16];
    int i,x;
    float tmp;
    tmp = V;
    x = (int)tmp;
    string[0] = x + '0';
    string[1] = '.';
    for (i=2;i<=4;++i) {
         tmp = (tmp - x) * 10;
         x = (int)tmp;
         string[i] = x + '0';
    }
    for (i=5;i<=15;++i)
         string[i] = ' ';
    lcd_puts(string);
}

/*
-------------------------------------------------------------------------------------------------------------------
Function:         alarm_alert()
Description:      A alarm alert function is now included, this is so that if the maximum or minimum voltage range
                  is reached the programmer can visually observe the desired warning.
-------------------------------------------------------------------------------------------------------------------
*/

void alarm_alert(unsigned int V2) {

    if(V2 > 1022)
         RC5 = 1;              // If the input voltage reaches 5V the red LED will turn on
    else
         RC5 = 0;
    if(V2 < 1)
         RC4 = 1;              // If the input voltage reduces to 0V the green LED will switch on
    else
         RC4 = 0;
}

/*
-------------------------------------------------------------------------------------------------------------------
Function:         lin_reg()
Description:      Linear regression routine, used in calibration process.
-------------------------------------------------------------------------------------------------------------------
*/

void lin_reg(void) {

    float x[2] = {0,5};
    double sx,sy,sxx,sxy;
    int j,k,n=1;
    sx = 0;
    sy = 0;
    sxx = 0;
    sxy = 0;
    for(j=0;j<=n;++j) {
         sx += x[j];
         sy += y[j];
         sxx += x[j] * x[j];
         sxy += x[j] * y[j];
    }
    m = (2.0 * sxy - sx * sy) / (2.0 * sxx - sx * sx);
    c = (sxx * sy - sx * sxy) / (2.0 * sxx - sx * sx);
    m_new = (620.0/3 - m) + (620.0/3);
    c_new = -c;
    lcd_goto(0);
    lcd_puts("Measured slope: ");
    DelayMs(100);
    lcd_goto(0x40);
    display_cal_data(m);
    lcd_goto(0x48);
    lcd_puts("digits/V");
    for(k=0;k<=10;++k)
         DelayMs(100);
    lcd_goto(0);
    lcd_puts("And intercept is:");
    DelayMs(100);
    lcd_goto(0x40);
    display_cal_data(c);
    lcd_goto(0x48);
    lcd_puts("digits  ");
    for(k=0;k<=10;++k)
         DelayMs(100);
}
```

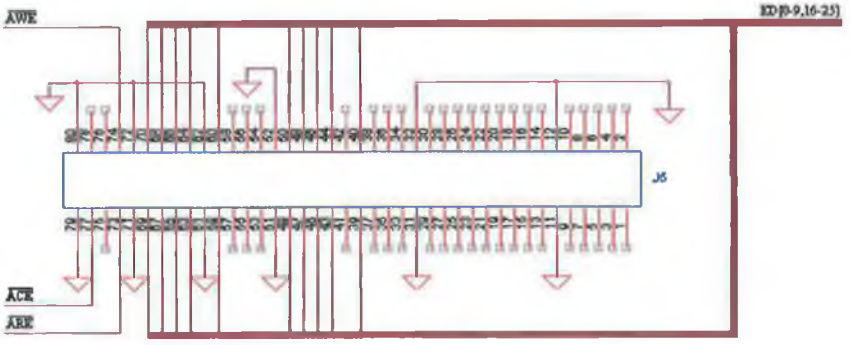
```
/*
-------------------------------------------------------------------------------------------------------------
Function:         cal_code()
Description:      Code required to step user through calibration process, with useful messages displayed.
-------------------------------------------------------------------------------------------------------------
*/

void cal_code(void) {

    int i;
    unsigned int Dig_temp_init,Dig_num_init[2];
    lcd_puts("Calibration     ");
    lcd_goto(0x40);
    lcd_puts("process starting");
    for(i=0;i<=10;++i)
        DelayMs(100);
    lcd_clear();
    lcd_goto(0);
    lcd_puts("Apply 0V level  ");
    lcd_goto(0x40);
    lcd_puts("to RA0(pin2) now");
    for(i=0;i<=20;++i)
        DelayMs(100);
    ADGO = 1;                                   // ADCON0 bit set
    while(ADGO)
        continue;                               // wait for conversion to complete
    Dig_temp_init = ADRESH;                     // Reading the A-D result register into variables, needing to be
    Dig_temp_init = Dig_temp_init << 8;         // manipulated.
    Dig_temp_init = Dig_temp_init + ADRESL;
    Dig_temp_init = Dig_temp_init >> 6;
    Dig_num_init[0] = Dig_temp_init;
    y[0] = Dig_num_init[0];
    DelayUs(24);
    lcd_goto(0);
    lcd_puts("Dig rep of 0V is");
    DelayMs(100);
    lcd_goto(0x40);
    display_dig_rep(Dig_num_init[0]);
    for(i=0;i<=20;++i)
        DelayMs(100);
    lcd_goto(0);
    lcd_puts("Finally apply   ");
    lcd_goto(0x40);
    lcd_puts("5V to RA0       ");
    for(i=0;i<=100;++i)
        DelayMs(100);
    ADGO = 1;
    while(ADGO)
        continue;
    Dig_temp_init = ADRESH;
    Dig_temp_init = Dig_temp_init << 8;
    Dig_temp_init = Dig_temp_init + ADRESL;
    Dig_temp_init = Dig_temp_init >> 6;
    Dig_num_init[1] = Dig_temp_init;
    y[1] = Dig_num_init[1];
    DelayUs(24);
    lcd_goto(0);
    lcd_puts("Dig rep of 5V   ");
    DelayMs(100);
    lcd_goto(0x40);
    display_dig_rep(Dig_num_init[1]);
    for(i=0;i<=10;++i)
        DelayMs(100);
    lin_req();
}
```

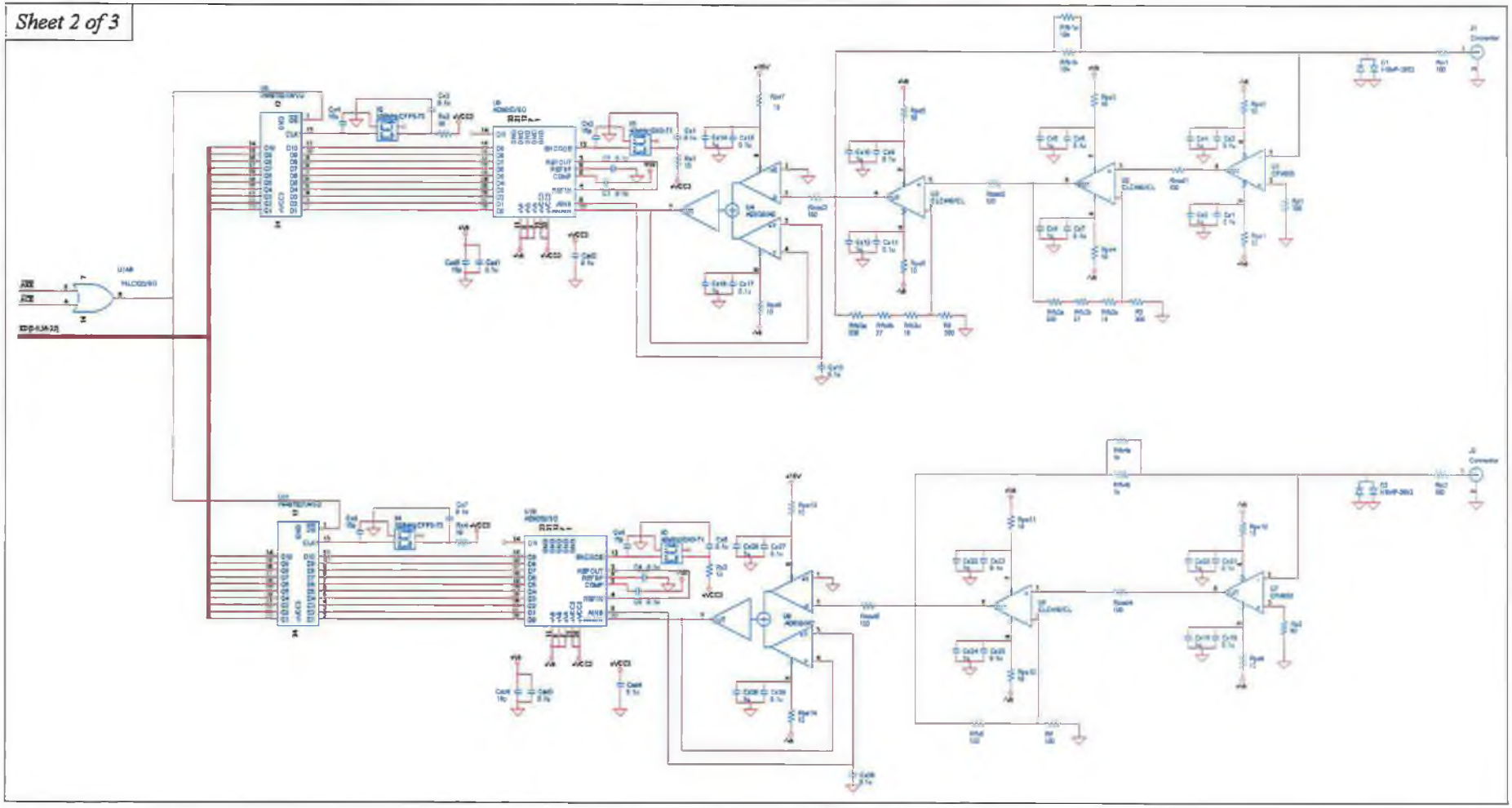
```
/*
-------------------------------------------------------------------------------
Main Programme
-------------------------------------------------------------------------------
*/


main() {

    int i;                      // main programme calls the initial set-up routines, the calibrate, measurement and
    init_ioports();             // display functions.
    config();
    DelayMs(255);
    lcd_init();
    lcd_clear();
    DelayMs(255);
    cal_code();
    DelayMs(100);

    for(;;) {
         adc_read_ac();
         lcd_goto(0);
         lcd_puts("P-P Voltage is: ");
         DelayMs(100);
         lcd_goto(0x40);
         display_temp(p_to_p);
         lcd_goto(0x45);
         lcd_puts("Vp-p       ");
         for(i=1;i<=5;++i)
          DelayMs(100);
         adc_read_dc();
         lcd_goto(0);
         lcd_puts("                ");
         lcd_goto(0x40);
         lcd_puts("                ");
         DelayMs(100);
         lcd_goto(0);
         lcd_puts("Mean voltage is:");
         DelayMs(100);
         lcd_goto(0x40);
         display_temp(Mea_volt);
         lcd_goto(0x45);
         lcd_puts("V          ");
         for(i=1;i<=5;++i)
          DelayMs(100);
         lcd_goto(0);
         lcd_puts("                ");
         lcd_goto(0x40);
         lcd_puts("                ");
         alarm_alert(Dig_temp);
         DelayMs(100);
    }
}
```

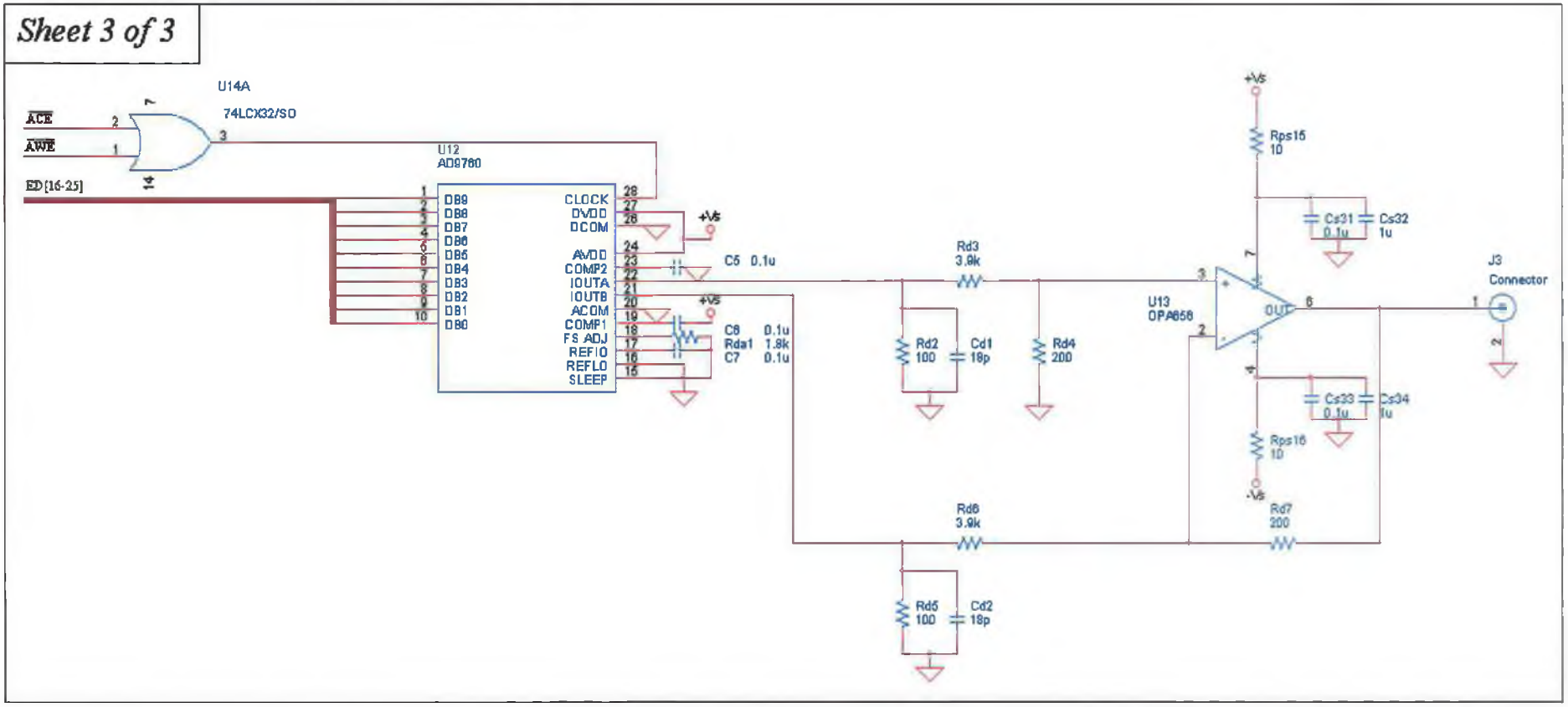
A.2 Prototype PCB Schematic Diagrams
Sheet 1 of 3








Sheet 2 of 3


Sheet 3 of 3
U14A
74LCX32/SO
ACE
AWE
ED[16-25]
U12
AD9760
DB9
DB8
DB7
DB6
DB5
DB4
DB3
DB2
DB1
DB0
CLOCK
DVDD
DCOM
AVDD
COMP2
IOUTA
IOUTB
ACOM
COMP1
FS ADJ
REFIO
REFLO
SLEEP
+Vs
C5 0.1u
C8 0.1u
Rda1 1.8k
C7 0.1u
Rd3
3.9k
Rd2
100
Cd1
18p
Rd4
200
Rd6
3.9k
Rd5
100
Cd2
18p
U13
OPA656
OUT
Rps15
10
Rps16
10
-Vs
Cs31
0.1u
Cs32
1u
Cs33
0.1u
Cs34
1u
Rd7
200
J3
Connector

A.3 List of Prototype PCB Components

Component Reference	Component Type and Footprint	Manufacturer	Manufacturer Number	Quantity
U1, U7, U13	Op-Amp SOIC 8 pin IC SOG.050/8/WG.244/L.225	Texas Instruments	OPA656U	3
U2, U3, U8	Op-Amp SOIC 8 pin IC SOG.050/8/WG.244/L.225	National Semiconductor	CLC449AJE	3
U4, U9	Difference Amplifier SOIC 8 pin IC SOG.050/8/WG.244/L.225	Analog Devices	AD830J	2
U5, U10	10-bit 40MSPS ADC SOIC 28 pin SOG.050/28/WG.420/L.725	Analog Devices	AD9050BR	2
U6, U11	10-bit Bus 125MHz Interface Buffer SOIC 24 pin SOG.050/24/WG.420/L.625	Texas Instruments	SN74ABT821A	2
U12	10-bit 125MSPS DAC SOIC 28 pin SOG.050/28/WG.420/L.725	Analog Devices	AD9760AR	1
U14	Quad 2-Input OR Gate SOIC 14 pin SOG.050/14/WG.244/L.350	Fairchild	74LCX32M	1
U15	+15V Regulator 3-Lead TO-220 TO220AB	National Semiconductor	LM340T-15	1

Component Reference	Component Type and Footprint	Manufacturer	Manufacturer Number	Quantity
U16	-5V Regulator 3-Lead TO-220 TO220AB	National Semiconductor	LM79M05CT	1
X1, X3	40MHz Crystal Oscillator SMD SM4XTAL (user created footprint)	C-MAC	SPXO010054 (IQXO-71)	2
X2, X4	100MHz Crystal Oscillator SMD SM4XTAL (user created footprint)	C-MAC	SPXO009437 (CFPS-73)	2
J1, J2, J3	Connector Header Square Pin BLKCON.100/VH/TM1SQ/W.100/3	Molex	(Farnell 143-140)	3
J4, J5	Connector Surface Mount Header SM80HDR (user created footprint)	Samtec	TFM-140-32-S-D-LC	2
D1, D2	Dual RF Diode SOT-23 SM/SOT23_123	Agilent Technologies	HSMP-3862 (Farnell 994-431)	2
Rin1, Rin2, Rp1, R5, Rfb5, Rload1, Rload2, Rload3, Rload4, Rload5, Rd2, Rd5	100Ω Resistor CRG0805 SM/R_0805	Multicomp	100R 1% tol. (Farnell 911-732)	12
Rps1, Rps2, Rps3, Rps4, Rps5, Rps6, Rps7, Rps8, Rps9, Rps10, Rps11, Rps12, Rps13, Rps14, Rps15, Rps16, Rx1, Rx2, Rx3, Rx4	10Ω Resistor CRG0805 SM/R_0805	Neohm	10R 1% tol.	20

Component Reference	Component Type and Footprint	Manufacturer	Manufacturer Number	Quantity
Rfb1a, Rfb1b	10kΩ Resistor CRG0805 SM/R_0805	Neohm	10k 1% tol.	2
R2, R3	300Ω Resistor CRG0805 SM/R_0805	Neohm	300R 1% tol.	2
Rfb2a, Rfb3a	330Ω Resistor CRG0805 SM/R_0805	Neohm	330R 1% tol.	2
Rfb2b, Rfb3b	27Ω Resistor CRG0805 SM/R_0805	Neohm	27R 1% tol.	2
Rfb2c, Rfb3c	18Ω Resistor CRG0805 SM/R_0805	Neohm	18R 1% tol.	2
Rp2	82Ω Resistor CRG0805 SM/R_0805	Neohm	82R 1% tol.	1
Rfb4a, Rfb4b	1kΩ Resistor CRG0805 SM/R_0805	Neohm	1k 1% tol.	2
Rd3, Rd6	3.9kΩ Resistor CRG0805 SM/R_0805	Neohm	3k9 1% tol.	2
Rd4, Rd7	200Ω Resistor CRG0805 SM/R_0805	Neohm	200R 1% tol.	2

Component Reference	Component Type and Footprint	Manufacturer	Manufacturer Number	Quantity
Rda1	1.8kΩ Resistor CRG0805 SM/R_0805	Neohm	1k8 1% tol.	1
Cs1, Cs3, Cs5, Cs7, Cs9, Cs11, Cs13, Cs15, Cs17, Cs19, Cs21, Cs23, Cs25, Cs26, Cs27, Cs29, Cs31, Cs33, C1, C2, C3, C4, C5, C6, C7, Cx1, Cx3, Cx5, Cx7, Cps2, Cps5, Cps7, Cps10, Cad1, Cad2, Cad3, Cad4	0.1uF (100nF) Capacitor 0805 SM/C_0805	Vishay Vitnamon	100nF ± 10%	37
Cs2, Cs4, Cs6, Cs8, Cs10, Cs12, Cs14, Cs16, Cs18, Cs20, Cs22, Cs24, Cs28, Cs30, Cs32, Cs34, Cps8, Cps11	1uF Capacitor 1206 SM/C_1206	AVX	1uF, 16V TAJA105K016R (Farnell 498-701)	18
Cx2, Cx4, Cx6, Cx8	15pF Capacitor 0805 SM/C_0805	Vishay Vitnamon	15pF ± 5%	4
Cd1, Cd2, Cad5, Cad6	18pF Capacitor 0805 SM/C_0805	Vishay Vitnamon	18pF ± 5%	4
Cps1, Cps4	220nF (0.22uF) Capacitor 1206 SM/C_1206	Vishay Vitnamon	220nF ± 10%	2
Cps3, Cps6, Cps9, Cps12	100uF Capacitor ECR Series – general purpose CPCYL/D.250/LS.100/.031	Multicomp	100uF ± 20%, 25V (Farnell 920-538)	4

A.4 DSP Implemented Programmes

A.4.1 data_in_fir+_pcb.c

```
/*
-------------------------------------------------------------------------------
Filename:       data_in_fir+_pcb.c
Description:    This programme uses one EDMA channel to service the EMIF. A default channel is
                used to transmit frames of data from the EMIF to the L2 internal memory. After
                the frames have been transmitted, the stored data is then digitally filtered with
                a FIR algorithm and finally analysed. The signal analysis results can be easily
                viewed numerically and graphically within the appropriate windows.
-------------------------------------------------------------------------------
*/

/*
-------------------------------------------------------------------------------
Include files
-------------------------------------------------------------------------------
*/

#include <c6x.h>
#define CHIP_6711               /* DSP chip defined */
#include <csl.h>
#include <csl_edma.h>
#include <csl_dat.h>
#include <stdio.h>
#include <math.h>
#include "c6711dsk.h"

/*
-------------------------------------------------------------------------------
Function prototypes
-------------------------------------------------------------------------------
*/

void submit_qdma(void);
void shift_input(void);
void filter_init(void);
short fir_filter (short input, int shift);
int freq_anal(int delay);
void volt_anal(int range);
void display(float var, int option);

/*
-------------------------------------------------------------------------------
Symbolic constants defined
-------------------------------------------------------------------------------
*/

#define MEM_SRC       0xB0000000        /* Source address (EMIF_CE3) for transfer          */
#define MEM_DST       0x00005E00        /* Destination address for transfer                */
#define EL_COUNT      0x2800            /* Element count for transfer                      */
#define COEF          45                /* Number of FIR filter coefficients               */
#define EMIF_CE3      0x01800014        /* EMIF chip enable No.3 control register address  */

/*
-------------------------------------------------------------------------------
External & global data variables
-------------------------------------------------------------------------------
*/

extern int shift_val;                   /* Defines the right shifting in the Q.N format    */
extern short h[COEF];                   /* FIR coefficients                                */
short R_in[COEF];                       /* Filter delay line                               */
int pass;                               /* Number of waveform cycles successfully analysed */
int gain=5;                             /* Gain value associated with the specific input   */
                                        /* channel used, Ch#1 - gain x5 & Ch#2 - gain x50  */
```

```
/*
-------------------------------------------------------------------------------
Function:       submit_qdma()
Description:    Submit a QDMA request to transfer the sampled data to internal (L2) memory.
Inputs:         -
Outputs:        -
Returns:        -
-------------------------------------------------------------------------------
*/

void submit_qdma(void){

     EDMA_Config config;
     config.opt = (Uint32)                    /* EDMA channel options selected (0x20200001):      */
         ((EDMA_OPT_PRI_HIGH    << _EDMA_OPT_PRI_SHIFT   )  /* High priority EDMA transfer,     */
        | (EDMA_OPT_ESIZE_32BIT << _EDMA_OPT_ESIZE_SHIFT )  /* 32-bit element size,             */
        | (EDMA_OPT_2DS_NO      << _EDMA_OPT_2DS_SHIFT   )  /* 1-dimensional source,            */
        | (EDMA_OPT_SUM_NONE    << _EDMA_OPT_SUM_SHIFT   )  /* Fixed source address mode,       */
        | (EDMA_OPT_2DD_NO      << _EDMA_OPT_2DD_SHIFT   )  /* 1-dimensional destination,       */
        | (EDMA_OPT_DUM_INC     << _EDMA_OPT_DUM_SHIFT   )  /* Increment destination address,   */
        | (EDMA_OPT_TCINT_NO    << _EDMA_OPT_TCINT_SHIFT )  /* Transfer complete indicator      */
        | (EDMA_OPT_TCC_DEFAULT << _EDMA_OPT_TCC_SHIFT   )  /* disabled, with no code set,      */
        | (EDMA_OPT_LINK_NO     << _EDMA_OPT_LINK_SHIFT  )  /* Event parameter link disabled,   */
        | (EDMA_OPT_FS_YES      << _EDMA_OPT_FS_SHIFT    ));/* Frame/block synchronised.        */

     config.src = (unsigned int)MEM_SRC;      /* Source address for transfer      (0xB0000000) */
     config.cnt = (unsigned int)EL_COUNT;     /* Element count for transfer       (0x00002800) */
     config.dst = (unsigned int)MEM_DST;      /* Destination address for transfer (0x00005E00) */
     config.idx = (unsigned int)0;            /* Element index offset addressing  (0x00000000) */

     EDMA_qdmaConfig(&config);

     return;

} /* end submit_qmda */

/*
-------------------------------------------------------------------------------
Function:       shift_input()
Description:    Sample data shifted to correctly represent 10-bit input with lsb.
Inputs:         -
Outputs:        -
Returns:        -
-------------------------------------------------------------------------------
*/

void shift_input(void){

    unsigned int LOC = 0, *x, org_val;

    for(LOC=0;LOC<=EL_COUNT-1;LOC++){
        x = (unsigned int*)MEM_DST+LOC;
        if(gain==50)
           org_val=*x&0x000003FF;       /* Depending on the channel in operation, (either Ch#1   */
        else{                           /* with gain set to 50 or Ch#2 with gain set to 5)       */
           org_val=*x&0x03FF0000;       /* only the respective 10-bit interfaced data lines will */
           org_val=org_val>>16;         /* be considered, all the other data lines are ignored.  */
        }                               /* An adjustment is also made which allows the selected  */
        *x=org_val;                     /* 10-bit data to be more appropriately represented.     */
    }

    return;

} /* end shift_input */

/*
-------------------------------------------------------------------------------
Function:       filter_init()
Description:    Initialisation of the filter variable
Inputs:         coef    =    number of filter coefficients
Outputs:        -
Returns:        -
-------------------------------------------------------------------------------
*/

void filter_init (void)
{
   int i;

   for(i=0; i<=COEF; i++)
        R_in[i]=0;

   return;

} /* end filter_init */
```

```
/*
-------------------------------------------------------------------------------
Function:       fir_filter()
Description:    Sample by sample FIR filtering
Inputs:         input    =      input sample to the filter
                shift    =      defines the Q.N format, shift the product right by 15 bits
Outputs:        -
Returns:        temp     =      filtered sample
-------------------------------------------------------------------------------
*/

short fir_filter (short input, int shift)
{
   int i;
   short temp;
   int Acc;

   for (i=COEF-1; i>0; i--)
     R_in[i]=R_in[i-1];                                  /* Shift delay samples        */
   R_in[0] = input;                                      /* Update most recent sample  */

   Acc = 0;
   for (i=0; i<COEF; i++)
      Acc += (int)(short)h[i] * (int)(short)R_in[i];     /* Sum operation in C         */
   temp = (short) (Acc>>shift);                          /* Output in 16-bit format    */

   return temp;

} /* end fir_filter */
```

```
/*
-------------------------------------------------------------------------------
Function:       freq_anal()
Description:    Analyses a signal's fundamental frequency by recording the average number of
                samples per cycle
Inputs:         coef    =       number of filter coefficients
Outputs:        -
Returns:        tot     =       total number of samples per cycle
-------------------------------------------------------------------------------
*/

int freq_anal(int coef){

    unsigned int LOC = 0, *x;
    int delay,sample[2]={0,0},pos=0,neg=0;
    int tot=0,tot9=0,type;
    float ref,freq;

    pass=0;
                                        /* Phase delay is a function of the number of        */
    delay=coef;                         /* coefficients used, hence only samples succeeding  */
    for(LOC=delay;LOC<=EL_COUNT-1;LOC++){   /* this delay are analysed.                      */
        x = (unsigned int*)MEM_DST+LOC;
        sample[0]=*x;
        if(sample[0]<512)               /* The total number of samples per cycle are         */
            ++neg;                      /* determined by using the mid-range value as a      */
        else                            /* crossing-point and therefore indicate the         */
            ++pos;                      /* start/end of a cycle.                             */
        if(sample[0]>=512 && sample[1]<512){
            ++pass;
            tot=pos+neg-1;
            ref=ceil((float)(tot*0.35));
            if(neg<=ref || pos<=ref){
                pass=0;
                tot9=0;                 /* The sample count is reset if an unexpected or     */
            }                           /* out of range signal is encountered.               */
            else                        /* By knowing the EMIF's sampling rate(Fs~100/4 MSPS) */
                tot9+=tot;              /* & the number of samples recorded over a specified */
            if(tot<=9)                  /* number of cycles (9), a signal's fundamental      */
                pass=0;                 /* frequency can be found:                           */
            if(pass==9){                /* signal freq = Fs / no. of samples per cycle       */
                if(gain==50){
                    puts("Ch#1(x50) Signal Results");
                    puts(" ");
                }
                else{
                    puts("Ch#2(x5) Signal Results");
                    puts(" ");
                }
                puts("Frequency(kHz):");    /* The samples of 9 successive cycles are used   */
                freq=(25e6)/(tot9/9.0);     /* instead of a single cycle in an effort to     */
                type=0;                     /* completely describe or cover an expected waveform */
                display(freq, type);        /* with a whole number of samples.               */
                break;                      /* E.g. The expected number of samples required to */
            }                               /* accommodate a 3MHz signal should need 100 samples */
            pos=1;                          /* over 9 cycles, with a sampling rate of 100/3 MSPS. */
            neg=0;                          /* Considering a lower number of cycles could result */
        }                                   /* in some signal data loss with part of a waveform  */
        sample[1]=sample[0];                /* being left unsampled and therefore could cause an */
    }                                       /* unnecessary error in the frequency calculation.   */
    return tot;                             /* The display() function is also called after the   */
                                            /* measured property (including units) have been     */
} /* end freq_anal */                       /* indicated within the standard output window.      */
```

```
/*
-------------------------------------------------------------------------------
Function:       volt_anal()
Description:    A signal's mean peak-to-peak voltage is determined by analysing the waveform's
                maximum & minimum levels over a set number of cycles.
Inputs:         sample_tot      =       total number of samples per cycle
Outputs:        -
Returns:        -
-------------------------------------------------------------------------------
*/

void volt_anal(int sample_tot){

     unsigned int LOC=0,LOC1=0,*x;
     int sample[2]={512,512},i,N=9;        /* The omitted, initial filtered data which possibly */
     int max=512,min=512,p_to_p=0,type;    /* includes some transience has been extended beyond */
                                           /* the expected phase delay period. This is          */
     for(LOC=50;LOC<=EL_COUNT-1;LOC++){    /* performed to ensure that the peak-to-peak voltage */
       x = (unsigned int*)MEM_DST+LOC;     /* result is limited to the signal's steady-state    */
       sample[0]=*x;                       /* range.                                            */
        if(sample[0]>=512 && sample[1]<512){
            for(i=0;i<=N-1;++i){
                for(LOC1=LOC;LOC1<=sample_tot+LOC;LOC1++){
                    x = (unsigned int*)MEM_DST+LOC1;
                    if(*x>max)
                        max=*x;
                    if(*x<min)
                        min=*x;
                }                          /* The peak-to-peak voltage of a set number of       */
                LOC=LOC1;                  /* waveforms are summed together. The maximum &      */
                p_to_p+=max-min;           /* minimum levels are reset for each cycle. The mean */
                max=512;                   /* Vp-p is then displayed with an appropriate comment */
                min=512;                   /* with units included. This mean result should      */
            }                              /* compensate for any excess signal noise.           */
            break;                         /* The initial if statement is required only once to */
        }                                  /* indicate the starting point of the voltage        */
        sample[1]=sample[0];               /* measurement, hence a break command is called to   */
    }                                      /* prevent the recorded voltage measurement from     */
    i=0;                                   /* being overwritten and suitable causes an          */
    while(i<1){                            /* unconditional exit from the primary for loop.     */
      ++i;
      if(pass<9)                           /* If the targeted 9 successive periodic waveforms   */
         break;                            /* within range are not found, the potentially       */
      if(sample_tot<=8||sample_tot>=750)   /* erroneous voltage result is ignored. By using the */
         break;                            /* samples per cycle value, upper and lower signal   */
       else{                               /* frequency limits are set at approx. 3MHz & 30kHz  */
         puts("Vp-p(mV):");                /* respectively. Voltage results relating to signals */
         if(gain==50)                      /* at frequencies beyond the expected measurement    */
             p_to_p=p_to_p/(N*50);         /* range are rejected.                               */
         else
            p_to_p=p_to_p/(N*5);
         type=1;
         display(p_to_p,type);
         puts(" ");
         puts(" ");
      }
    }

    return;

} /* end volt_anal */
```

```
/*
-------------------------------------------------------------------------------
Function:       display()
Description:    To display results by utilising the standard output (stdout) window this function
                converts measured decimal results into the appropriate string character format.
Inputs:         var     =   variable result (e.g. frequency in kHz or p-p voltage in units of mV)
                option  =   fixed or floating-point display options
Outputs:        -
Returns:        -
-------------------------------------------------------------------------------
*/

void display(float var, int option){

    char string[8];             /* To display a maximum of 4 significant figures before and 2    */
    int i,j,k,x,y;              /* after the decimal point, integer and real number variables    */
    float tmp,tmp2;             /* are declared.                                                 */

    if(option==0){              /* Two primary display type options are available; option 0 is   */
      tmp = var / 1000000;      /* used to display floating-point variables, while option 1      */
      tmp2 = var / 1000;        /* applies to integer-type variables, as required.               */
    }
    else{
      tmp = var / 1000;
      tmp2= 0;                  /* The real tmp value is converted into its integer              */
    }                           /* representation causing the digits below the decimal point to */
    x = (int)tmp;               /* be removed, as is also the case for the tmp2 variable.        */
    y = (int)tmp2;              /* Hence, string[0] will therefore initially represent the ASCII*/
    string[0] = x + '0';        /* code for the thousand digit (thousand kHz / thousand mV).     */
    for(i=1 ; i<=3 ; i++) {     /* The first for loop is used to display, the hundred, ten and   */
      tmp = (tmp - x) * 10;     /* one digits. Within the brackets of the initial statement      */
      x = (int)tmp;             /* of the for loop, the thousand integer digit is subtracted     */
      string[i] = x + '0';      /* from its floating point value which causes the remaining most*/
    }                           /* significant hundreds digit being below the radix point.       */
    string[4] = '.';            /* Therefore * by 10 and then finding the integer value will     */
    for(j=5 ; j<=6 ; j++) {     /* give the suitable hundreds value only, the same loop          */
      tmp2 = (tmp2 - y) * 10;/* continues to find the tens & ones digit. The second for loop */
      y = (int)tmp2;            /* repeats the previous procedure for the digits below the       */
      string[j] = y + '0';      /* decimal point, '0' is added to these digits to give their     */
    }                           /* ASCII code character values.                                  */
    string[7] = '\0';
    if(option==1) {             /* The conditional statements are included to allow the most     */
      for(k=4;k<=7;k++)         /* significant non-zero figure to be displayed first, while      */
          string[k] = '\0';     /* removing any unnecessary digits below the decimal point.      */
    }
    if(string[0] == '0') {
      for(k=0;k<=7;k++)
          string[k-1] = string[k];
          if(string[0] == '0') {
              for(k=0;k<=6;k++)
                  string[k-1] = string[k];
          }
          else {
              for(k=0;k<=6;k++)
                  string[k] = string[k];
          }
    }
    else {
      for(k=0;k<=7;k++)
          string[k] = string[k];
    }
    puts(string);               /* Finally another function is called to write the string of     */
                                /* characters in the standard output (stdout) window.            */
    return;


} /* end display */
```

```
/*
-------------------------------------------------------------------------------
 Main programme
-------------------------------------------------------------------------------
*/

void main(void){
                                        /* Main programme variables declared, the EMIF CE3   */
  unsigned int LOC=0, *x,*y;            /* control register is configured with one setup,    */
  int sample_len,mod_val;               /* three strobe & zero hold clock (ELCKOUT) cycles.  */
  short temp,output;                    /* Therefore an approx. sample rate of 100MSPS/4 is  */
  *(int *)EMIF_CE3 = 0x00010320;        /* employed. Also note the memory type is set to     */
                                        /* 32-bit asynchronous memory.                       */
  while(1){
      submit_qdma();                    /* Interface buffer to internal memory data transfer */
      shift_input();                    /* 10-bit input data selected & fittingly adjusted   */
      filter_init();                    /* Initialise the FIR filter variable

      for(LOC=0;LOC<=EL_COUNT-1;LOC++){ /* Sample by sample filtering with signal limits set */
          x = (unsigned int*)MEM_DST+LOC;
          temp = (short)(*x & 0xffff);
          output =  fir_filter(temp,shift_val);
          if(output>1023)
              output=1023;
      else if(output<0)
              output=0;
          else
              output=output;
          *x = output;
      }

      sample_len = freq_anal(COEF);     /* Sampled signal's frequency calculated & displayed */
      volt_anal(sample_len);            /* Signal's mean Vp-p calculated & displayed         */

      for(LOC=0;LOC<=EL_COUNT-1;LOC++){ /* Filter delay considered                           */
          x = (unsigned int*)MEM_DST+LOC+((COEF-1)/2);
          y = (unsigned int*)MEM_DST+LOC;
          mod_val=*x;
          *y=mod_val;
      }

      for(LOC=0;LOC<=EL_COUNT-1;LOC++){ /* Signal data finally modified, to view graphically */
          x = (unsigned int*)MEM_DST+LOC; /* an input signal in it's signed form.            */
          if(gain==50){
              mod_val=*x/50;
              if(mod_val>=10)
                  mod_val=mod_val-10;
              else
                  mod_val=mod_val+65526;
          }
          else{
              mod_val=*x/5;
              if(mod_val>=102)
                  mod_val=mod_val-102;
              else
                  mod_val=mod_val+65434;
          }
          mod_val=mod_val;
          *x=mod_val;
      }
  }
} /* end main programme */
```

A.4.2 fir_coeffv45_q15_fc2meg_fs25msps.c

```
/*
-------------------------------------------------------------------------------
Filename:       fir_coeffv45_q15_fc2meg_fs25msps.c
Description:    This C file contains the coefficients of a 44th order (45-length vector) lowpass
                FIR filter. Designed with Matlab's fir() function, a Hamming Window method is used.
                When implemented into a system with a sample rate of 25MSPS, a cut-off frequency of
                2MHz will result. These coefficients are stored in Q.15 format as an associated
                short data type is declared.
-------------------------------------------------------------------------------
*/

int shift_val = 15;

short h[45]={
          -38,
          -38,
          -30,
           -8,
           33,
           91,
          153,
          193,
          178,
           82,
         -103,
         -351,
         -601,
         -763,
         -736,
         -433,
          184,
         1090,
         2190,
         3331,
         4328,
         5009,
         5251,
         5009,
         4328,
         3331,
         2190,
         1090,
          184,
         -433,
         -736,
         -763,
         -601,
         -351,
         -103,
           82,
          178,
          193,
          153,
           91,
           33,
           -8,
          -30,
          -38,
          -38,

};
```

A.4.3 c6711dsk.cmd

```
-stack 0x80
MEMORY
{
  VECS:       org =          0h,  len =      0x220
  I_HS_MEM:   org = 0x00000220,  len = 0x00000020
  IRAM:       org = 0x00000240,  len = 0x0000FFFF
  SDRAM:      org = 0x80000000,  len = 0x00FFD000
  FLASH:      org = 0x90000000,  len = 0x00020000
}

SECTIONS
{
  /* Created in vectors.asm */
  vectors  :> VECS

  /* Created by C Compiler  */
  .text    :> IRAM
  .stack   :> IRAM
  .bss     :> SDRAM
  .cinit   :> SDRAM
  .sysmem  :> SDRAM
  .const   :> SDRAM
  .switch  :> SDRAM
  .far     :> SDRAM
  .cio     :> SDRAM
}
```

A.4.4 data_in_iir+_pcb.c

```
/*
-------------------------------------------------------------------------------
Filename:       data_in_iir+_pcb.c
Description:    This programme uses one EDMA channel to service the EMIF. A default channel is used
                to transmit frames of data from the EMIF to the L2 internal memory. After the
                frames have been transmitted, the stored data is then digitally filtered with a IIR
                algorithm and also analysed using a FFT function. The signal analysis results can
                be easily viewed numerically and graphically within the appropriate time and
                frequency domain windows.
-------------------------------------------------------------------------------
*/

/*
-------------------------------------------------------------------------------
Typedefs
-------------------------------------------------------------------------------
*/

typedef struct {
  short real;
  short imag;
}COMPLEX;

/*
-------------------------------------------------------------------------------
Include files
-------------------------------------------------------------------------------
*/

#include <c6x.h>
#define CHIP_6711                   /* DSP chip defined */
#include <csl.h>
#include <csl_edma.h>
#include <csl_dat.h>
#include <stdio.h>
#include <math.h>
#include "c6711dsk.h"
#include "fft_tables.h"
#include "digit_rev.h"

/*
-------------------------------------------------------------------------------
Function prototypes
-------------------------------------------------------------------------------
*/

void submit_qdma(void);
void shift_input(void);
void filter_init(int sos_size);
int iir_cas5(int input, int *c, int *d, int shift, int n);
int freq_anal(int delay);
void volt_anal(int range);
void display(float var, int option);
void radix_2(short *xy, short n, short *w);
void bitrev_cplx(int *x, short *index, int nx);

/*
-------------------------------------------------------------------------------
Symbolic constants defined
-------------------------------------------------------------------------------
*/

#define MEM_SRC         0xB0000000          /* Source address (EMIF_CE3) for transfer           */
#define MEM_DST         0x00008000          /* Destination address for transfer                 */
#define EL_COUNT        0x1000              /* Element count for transfer                       */
#define SOS_SECTIONS    4                   /* Number of second order sections                  */
#define EMIF_CE3        0x01800014          /* EMIF chip enable No.3 control register address   */
#define SCALE_DOWN      7                   /* Scaling factor, to avoid overflow                */
#define NUMDATA         512                 /* Number of real data samples                      */
```

```
/*
-------------------------------------------------------------------------------
External & global data variables
-------------------------------------------------------------------------------
*/

extern int shift_val;                       /* Defines the right shifting in the Q.N format      */
extern int iir_coefs[5*SOS_SECTIONS];       /* IIR coefficients                                  */
int delay_line[2*SOS_SECTIONS];             /* Delay line of the SOS                             */
int pass;                                   /* Number of waveform cycles successfully analysed   */
int gain=5;                                 /* Gain value associated with the specific input     */
                                            /* channel used, Ch#1 - gain x5 & Ch#2 - gain x50    */
short g[NUMDATA];                           /* Array of scale down data                          */
COMPLEX x[NUMDATA];                         /* Array of complex DFT data                         */
int magR[NUMDATA];                          /* Squared real part of the FFT                      */
int magI[NUMDATA];                          /* Squared imaginary part of the FFT                 */
int mag[NUMDATA];                           /* Squared magnitude of the FFT                      */
int amp[NUMDATA];                           /* Amplitude of the FFT                              */

/*
-------------------------------------------------------------------------------
Function:       submit_qdma()
Description:    Submit a QDMA request to transfer the sampled data to internal (L2) memory.
Inputs:         -
Outputs:        -
Returns:        -
-------------------------------------------------------------------------------
*/

void submit_qdma(void){

    EDMA_Config config;
    config.opt = (Uint32)                   /* EDMA channel options selected (0x20200001):       */
        ((EDMA_OPT_PRI_HIGH    << _EDMA_OPT_PRI_SHIFT   )  /* High priority EDMA transfer,      */
       | (EDMA_OPT_ESIZE_32BIT << _EDMA_OPT_ESIZE_SHIFT )  /* 32-bit element size,              */
       | (EDMA_OPT_2DS_NO      << _EDMA_OPT_2DS_SHIFT   )  /* 1-dimensional source,             */
       | (EDMA_OPT_SUM_NONE    << _EDMA_OPT_SUM_SHIFT   )  /* Fixed source address mode,        */
       | (EDMA_OPT_2DD_NO      << _EDMA_OPT_2DD_SHIFT   )  /* 1-dimensional destination,        */
       | (EDMA_OPT_DUM_INC     << _EDMA_OPT_DUM_SHIFT   )  /* Increment destination address,    */
       | (EDMA_OPT_TCINT_NO    << _EDMA_OPT_TCINT_SHIFT )  /* Transfer complete indicator       */
       | (EDMA_OPT_TCC_DEFAULT << _EDMA_OPT_TCC_SHIFT   )  /* disabled, with no code set,       */
       | (EDMA_OPT_LINK_NO     << _EDMA_OPT_LINK_SHIFT  )  /* Event parameter link disabled,    */
       | (EDMA_OPT_FS_YES      << _EDMA_OPT_FS_SHIFT    ));/* Frame/block synchronised.         */

    config.src = (unsigned int)MEM_SRC;     /* Source address for transfer      (0xB0000000) */
    config.cnt = (unsigned int)EL_COUNT;    /* Element count for transfer       (0x00001000) */
    config.dst = (unsigned int)MEM_DST;     /* Destination address for transfer (0x00008000) */
    config.idx = (unsigned int)0;           /* Element index offset addressing  (0x00000000) */

    EDMA_qdmaConfig(&config);

    return;

} /* end submit_qmda */

/*
-------------------------------------------------------------------------------
Function:       shift_input()
Description:    Sample data shifted to correctly represent 10-bit input with lsb.
Inputs:         -
Outputs:        -
Returns:        -
-------------------------------------------------------------------------------
*/

void shift_input(void){

    unsigned int LOC = 0, *x, org_val;

    for(LOC=0;LOC<=EL_COUNT-1;LOC++){
        x = (unsigned int*)MEM_DST+LOC;
        if(gain==50)
           org_val=*x&0x000003FF;       /* Depending on the channel in operation, (either Ch#1   */
        else{                           /* with gain set to 50 or Ch#2 with gain set to 5)       */
           org_val=*x&0x03FF0000;       /* only the respective 10-bit interfaced data lines will */
           org_val=org_val>>16;         /* be considered, all the other data lines are ignored.  */
        }                               /* An adjustment is also made which allows the selected  */
        *x=org_val;                     /* 10-bit data to be more appropriately represented.     */
    }

    return;

} /* end shift_input */
```

```
/*
-------------------------------------------------------------------------------
Function:       filter_init()
Description:    Initialisation of the filter variable
Inputs:         sos_size   =   number of SOS stages
Outputs:        -
Returns:        -
-------------------------------------------------------------------------------
*/

void filter_init(int sos_size){

  int i;

  for(i=0; i<2*sos_size; i++)
      delay_line[i] = 0;              /* Delay line initialisation */

  return;

} /* end filter_init */

/*
-------------------------------------------------------------------------------
Function:       iir_cas5()
Description:    Second Order Section cascade IIR filter, 5 coefficients per biquadratic section,
                each coefficient in Q16 format, 16 magnitude bits and 1 sign bit, total of 17 bits.
Inputs:         input =  input sample to the filter
                c     =  coefficients of the total cascade configuration
                d     =  delay states of the total cascade configuration
                n     =  number of the second order sections
                shift =  defines the Q.N format (minus 1), shift the product right by 15 bits
Outputs:        -
Returns:        temp  =  filtered sample
-------------------------------------------------------------------------------
*/

int iir_cas5(int input, int *c, int *d, int shift, int n){

  int k0;
  int temp;
  int i;

  temp = input;
  for (i=0; i<n; i++){
      k0   = temp - ((c[5*i+1]*d[2*i+1])>>shift) - ((c[5*i+0]*d[2*i+0])>>shift);
      temp = ((c[5*i+4]*d[2*i+1])>>shift)+((c[5*i+3]*d[2*i+0])>>shift)+((c[5*i+2]*k0)>>shift);

     d[2*i+1] = d[2*i+0];
     d[2*i+0] = k0;
   }

   return temp;

} /* end iir_cas5 */
```

```
/*
-------------------------------------------------------------------------------
Function:       freq_anal()
Description:    Analyses a signal's fundamental frequency by recording the average number of
                samples per cycle
Inputs:         coef    =       number of filter coefficients
Outputs:        -
Returns:        tot     =       total number of samples per cycle
-------------------------------------------------------------------------------
*/

int freq_anal(int coef){

   unsigned int LOC = 0, *x;
   int delay,sample[2]={0,0},pos=0,neg=0;
   int tot=0,tot9=0,type;
   float ref,freq;

   pass=0;                                  /* Phase delay is a function of the number of        */
   delay=((sos_size*5)-1)/2;                /* coefficients used, hence only samples succeeding  */
   for(LOC=delay;LOC<=EL_COUNT-1;LOC++){    /* this delay are analysed.                          */
       x = (unsigned int*)MEM_DST+LOC;
       sample[0]=*x;
       if(sample[0]<512)                    /* The total number of samples per cycle are         */
           ++neg;                           /* determined by using the mid-range value as a      */
       else                                 /* crossing-point and therefore indicate the         */
           ++pos;                           /* start/end of a cycle.                             */
       if(sample[0]>=512 && sample[1]<512){
           ++pass;
           tot=pos+neg-1;
           ref=ceil((float)(tot*0.45));
           if(neg<=(ref-2) || pos<=(ref-2) || (tot<=4 && pass>0))
              pass=pass-1;                  /* The sample count is reset if an unexpected or     */
           else                             /* out of range signal is encountered.               */
              tot9+=tot;                    /* By knowing the EMIF's sampling rate (Fs~100/4MSPS) */
           if(pass==9  && tot9>=60){        /* & the number of samples recorded over a specified */
              if(gain==50){                 /* number of cycles (9), a signal's fundamental      */
                 puts("Ch#1(x50) Signal Results");
                 puts(" ");                 /* frequency can be found:                           */
              }                             /* signal freq = Fs / no. of samples per cycle       */
              else{
                 puts("Ch#2(x5) Signal Results");
                 puts(" ");
              }
              puts("Frequency(kHz):");      /* The samples of 9 successive cycles are used       */
              freq=(25e6)/(tot9/9.0);       /* instead of a single cycle in an effort to         */
              type=0;                       /* completely describe or cover an expected waveform */
              display(freq, type);          /* with a whole number of samples.                   */
              break;                        /* E.g. The expected number of samples required to   */
           }                                /* accommodate a 3MHz signal should need 100 samples */
           pos=1;                           /* over 9 cycles, with a sampling rate of 100/3 MSPS. */
           neg=0;                           /* Considering a lower number of cycles could result */
       }                                    /* in some signal data loss with part of a waveform  */
       sample[1]=sample[0];                 /* being left unsampled and therefore could cause an */
   }                                        /* unnecessary error in the frequency calculation.   */
   return tot;                              /* The display() function is also called after the   */
                                            /* measured property (including units) have been     */
} /* end freq_anal */                       /* indicated within the standard output window.      */
```

```
/*
-------------------------------------------------------------------------------
Function:       volt_anal()
Description:    A signal's mean peak-to-peak voltage is determined by analysing the waveform's
                maximum & minimum levels over a set number of cycles.
Inputs:         sample_tot      =       total number of samples per cycle
Outputs:        -
Returns:        -
-------------------------------------------------------------------------------
*/

void volt_anal(int sample_tot){

    unsigned int LOC=0,LOC1=0,*x;
    int sample[2]={512,512},i,N=9;          /* The omitted, initial filtered data which possibly */
    int max=512,min=512,p_to_p=0,type;      /* includes some transience has been extended beyond */
                                            /* the expected phase delay period. This is          */
    for(LOC=50;LOC<=EL_COUNT-1;LOC++){      /* performed to ensure that the peak-to-peak voltage */
      x = (unsigned int*)MEM_DST+LOC;       /* result is limited to the signal's steady-state    */
      sample[0]=*x;                         /* range.                                            */
       if(sample[0]>=512 && sample[1]<512){
           for(i=0;i<=N-1;++i){
               for(LOC1=LOC;LOC1<=sample_tot+LOC;LOC1++){
                   x = (unsigned int*)MEM_DST+LOC1;
                   if(*x>max)
                      max=*x;
                   if(*x<min)
                      min=*x;
               }                            /* The peak-to-peak voltage of a set number of       */
               LOC=LOC1;                    /* waveforms are summed together. The maximum &      */
               p_to_p+=max-min;             /* minimum levels are reset for each cycle. The mean */
               max=512;                     /* Vp-p is then displayed with an appropriate comment*/
               min=512;                     /* with units included. This mean result should      */
           }                                /* compensate for any excess signal noise.           */
           break;                           /* The initial if statement is required only once to */
       }                                    /* indicate the starting point of the voltage        */
       sample[1]=sample[0];                 /* measurement, hence a break command is called to   */
    }                                       /* prevent the recorded voltage measurement from     */
    i=0;                                    /* being overwritten and suitable causes an          */
    while(i<1){                             /* unconditional exit from the primary for loop.     */
      ++i;
      if(pass<9)                            /* If the targeted 9 successive periodic waveforms   */
         break;                             /* within range are not found, the potentially       */
      if(sample_tot<=6||sample_tot>=850)    /* erroneous voltage result is ignored. By using the */
         break;                             /* samples per cycle value, upper and lower signal   */
      else{                                 /* frequency limits are set at approx. 3MHz & 30kHz  */
         puts("Vp-p(mV):");                 /* respectively. Voltage results relating to signals */
         if(gain==50)                       /* at frequencies beyond the expected measurement    */
             p_to_p=p_to_p/(N*50);          /* range are rejected.                               */
         else
            p_to_p=p_to_p/(N*5);
         type=1;
         display(p_to_p,type);
         puts(" ");
         puts(" ");
      }
    }

    return;

} /* end volt_anal */
```

```
/*
-------------------------------------------------------------------------------
Function:       display()
Description:    To display results by utilising the standard output (stdout) window this function
                converts measured decimal results into the appropriate string character format.
Inputs:         var     =   variable result (e.g. frequency in kHz or p-p voltage in units of mV)
                option  =   fixed or floating-point display options
Outputs:        -
Returns:        -
-------------------------------------------------------------------------------
*/

void display(float var, int option){

     char string[8];            /* To display a maximum of 4 significant figures before and 2   */
     int i,j,k,x,y;             /* after the decimal point, integer and real number variables   */
     float tmp,tmp2;            /* are declared.                                                */

     if(option==0){             /* Two primary display type options are available; option 0 is   */
       tmp = var / 1000000;     /* used to display floating-point variables, while option 1     */
       tmp2 = var / 1000;       /* applies to integer-type variables, as required.              */
     }
     else{
       tmp = var / 1000;
       tmp2= 0;                 /* The real tmp value is converted into its integer             */
     }                          /* representation causing the digits below the decimal point to */
     x = (int)tmp;              /* be removed, as is also the case for the tmp2 variable.       */
     y = (int)tmp2;             /* Hence, string[0] will therefore initially represent the ASCII*/
     string[0] = x + '0';       /* code for the thousand digit (thousand kHz / thousand mV).    */
     for(i=1 ; i<=3 ; i++) {    /* The first for loop is used to display, the hundred, ten and  */
       tmp = (tmp - x) * 10;    /* one digits. Within the brackets of the initial statement     */
       x = (int)tmp;            /* of the for loop, the thousand integer digit is subtracted    */
       string[i] = x + '0';     /* from its floating point value which causes the remaining most*/
     }                          /* significant hundreds digit being below the radix point.      */
     string[4] = '.';           /* Therefore * by 10 and then finding the integer value will    */
     for(j=5 ; j<=6 ; j++) {    /* give the suitable hundreds value only, the same loop         */
       tmp2 = (tmp2 - y) * 10;  /* continues to find the tens & ones digit. The second for loop */
       y = (int)tmp2;           /* repeats the previous procedure for the digits below the      */
       string[j] = y + '0';     /* decimal point, '0' is added to these digits to give their    */
     }                          /* ASCII code character values.                                 */
     string[7] = '\0';
     if(option==1) {            /* The conditional statements are included to allow the most    */
       for(k=4;k<=7;k++)        /* significant non-zero figure to be displayed first, while     */
           string[k] = '\0';    /* removing any unnecessary digits below the decimal point.     */
     }
     if(string[0] == '0') {
       for(k=0;k<=7;k++)
           string[k-1] = string[k];
           if(string[0] == '0') {
               for(k=0;k<=6;k++)
                   string[k-1] = string[k];
           }
           else {
               for(k=0;k<=6;k++)
                   string[k] = string[k];
           }
     }
     else {
       for(k=0;k<=7;k++)
           string[k] = string[k];
     }
     puts(string);              /* Finally another function is called to write the string of    */
                                /* characters in the standard output (stdout) window.           */
     return;


} /* end display */
```

```
/*
-------------------------------------------------------------------------------
Function:       radix_2()
Description:    Radix-2 FFT implementation
Inputs:         n     =       number of points in the FFT
                X     =       array with complex samples
                W     =       array with twiddle factors
Outputs:        x     =       array with the FFT complex samples
Returns:        None
-------------------------------------------------------------------------------
*/
void radix_2(short *x, short n, short *w)
{
        short n1,n2,ie,ia,i,j,k,l;
        short xt,yt,c,s;

        n2 = n;
        ie = 1;
        for (k=n; k > 1; k = (k >> 1) ) {
          n1 = n2;
          n2 = n2>>1;
          ia = 0;
          for (j=0; j < n2; j++) {
            c = w[2*ia];
            s = w[2*ia+1];
            ia = ia + ie;
            for (i=j; i < n; i += n1) {
              l = i + n2;
              xt       = x[2*l] - x[2*i];
              x[2*i]   = x[2*i] + x[2*l];
              yt       = x[2*l+1] - x[2*i+1];
              x[2*i+1]    = x[2*i+1] + x[2*l+1];
              x[2*l]      = (c*xt + s*yt)>>15;
              x[2*l+1]    = (c*yt - s*xt)>>15;
            }
          }
          ie = ie<<1;
        }

        return;
} /* end radix_2 */
```

```
/*
-------------------------------------------------------------------------------
Main programme
-------------------------------------------------------------------------------
*/

void main(void){

  unsigned int LOC=0;                       /* Main programme variables declared, the EMIF CE3   */
  int temp,output,sample_len,mod_val,n,n2,k;
  unsigned int *a,*b,av,bv,count=0;         /* control register is configured with one setup,    */
  short *ab;                                /* three strobe & zero hold clock (ELCKOUT) cycles.  */
  *(int *)EMIF_CE3 = 0x00010320;            /* Therefore sample rate = 100MSPS/4. Also note the  */
                                            /* memory type is set to 32-bit asynchronous memory. */
  while(1){
       submit_qdma();                       /* Interface buffer to internal memory data transfer */
       shift_input();                       /* 10-bit input data selected & fittingly adjusted   */
       filter_init(SOS_SECTIONS);           /* Initialise the IIR delay lines                    */
       for(LOC=0;LOC<=EL_COUNT-1;LOC++){    /* Sample by sample filtering with signal limits set */
            a = (unsigned int*)MEM_DST+LOC;
            temp = (int)(*a & 0xffff);
            output =  iir_cas5(temp,iir_coefs,delay_line,shift_val,SOS_SECTIONS);
            if(output>1023)
                 output=1023;
            else if(output<0)
                 output=0;
            else
                 output=output;
            *a = output;
       }
       sample_len = freq_anal(SOS_SECTIONS);/* Sampled signal's frequency calculated & displayed*/
       volt_anal(sample_len);               /* Signal's mean Vp-p calculated & displayed         */
       for(LOC=0;LOC<=EL_COUNT-1;LOC++){    /* Signal data finally modified, to view graphically */
            a = (unsigned int*)MEM_DST+LOC; /* an input signal in it's signed form.              */
            if(gain==50){
                 mod_val=*a/50;
                 if(mod_val>=10)
                      mod_val=mod_val-10;
                 else
                      mod_val=mod_val+65526;
            }
            else{
                 mod_val=*a/5;
                 if(mod_val>=102)
                      mod_val=mod_val-102;
                 else
                      mod_val=mod_val+65434;
            }
            mod_val=mod_val;
            *a=mod_val;
       }
       for(n=0;n<NUMDATA/2;n++){            /* Code used to convert stored data into more ideal  */
            a = (unsigned int*)MEM_DST+n;   /* compressed form, required for the fft function.   */
            b = (unsigned int*)MEM_DST+n+1;
            av=*a;
            bv=*b<<16;
            *a=bv+av;
            *b=0;
            ++count;
            for(n2=count;n2<NUMDATA-count;n2++){
                 a = (unsigned int*)MEM_DST+n2;
                 b = (unsigned int*)MEM_DST+n2+1;
                 av=*a;
                 bv=*b;
                 *a=bv+av;
                 *b=0;
            }
       }
       count=0;
       for(n=0;n<NUMDATA;n++){
            ab=(short*)MEM_DST+n;
            g[n] = (short)(*ab)>>SCALE_DOWN;  /* Scale down the input signal to avoid overflows.*/
       }
       for (n=0; n<NUMDATA; n++){           /* Complex 512 point FFT                             */
            x[n].real = g[n];               /* x1(n) = g(2)                                      */
            x[n].imag = 0;                  /* Zero the imaginary part and compute the FFT of    */
       }                                    /* x(n) to get X(k)->X(k)=DFT(x(n))                  */
       radix_2((short *)x,NUMDATA,(short *)W4);
       bitrev_cplx((int*)x, index, NUMDATA); /* DSP digit reversal lib function                  */
       for (k=0; k<NUMDATA; k++){           /* Compute the magnitude of the FFT                  */
            magR[k] = (x[k].real * x[k].real);
            magI[k] = (x[k].imag * x[k].imag);
            mag[k] = magR[k] + magI[k];
            amp[k] = sqrt(2*mag[k]);
       }
  }
} /* end main programme */
```

A.4.5 iir_cas5_stage4_fc2meg_fs25msps.c

```
/*
-------------------------------------------------------------------------------------------------
Filename:       iir_cas5_stage4_fc2meg_fs25msps.c
Description:    This C file contains the coefficients of an 8th order (9-length numerator and
                denominator vectors) cascade form lowpass IIR filter. Designed with Matlab's
                butter() function (Butterworth digital filter) and then converted into 4 second-
                order section (SOS) stages using the tf2sos() function (with 5 non-unity
                coefficients per cascaded stage).  The number of stages required is determined by
                the fact that 8 non-unity denominator coefficients initially returned by the
                butter() function are converted into SOS stages with only two non-unity denominator
                coefficients permitted per stage. When implemented into a system with a sample rate
                of 25MSPS, a cut-off frequency of 2MHz will result. These coefficients are stored
                in Q.16 format as an associated int data type is declared.
-------------------------------------------------------------------------------------------------
*/


int shift_val = 15;

int iir_coefs[20] =
{
      -44770,     15383,       352,      705,       352,
      -46654,     17409,       904,     1836,       932,
      -50588,     21640,       819,     1638,       819,
      -56851,     28377,      2926,     5763,      2839
};
```

A.4.6 data_out_in_phase+_pcb.c

```
/*
-------------------------------------------------------------------------------
Filename:       data_out_in_phase+_pcb.c
Description:    This programme determines the phase delay associated with the system's
                instrumentation. Internally generated test signals are transmitted to an external
                peripheral device (DAC) via the EMIF. The resultant signal is applied to an input
                channel, which is returned to internal memory for analysis. By comparing the
                transmitted and received signals, any phase delay between the two signals is
                calculated and displayed. To complete the signal analysis results, the input
                signal's frequency and amplitude are also indicated. To avoid any additional delay
                caused by the filtering algorithm, only after the phase analysis has been completed
                is the input signal digitally filtered.
-------------------------------------------------------------------------------
*/

/*
-------------------------------------------------------------------------------
Include (functional dependent) files
-------------------------------------------------------------------------------
*/

#include <c6x.h>
#include "c6711dsk.h"

/*
-------------------------------------------------------------------------------
Function prototypes
-------------------------------------------------------------------------------
*/

void trans_rec_sample(void);
void shift_input(void);
int freq_anal(int delay);
void phase_anal(int range);
void volt_anal(int range);
void display(float var, int option);
void filter_init(void);
short fir_filter (short input, int shift);

/*
-------------------------------------------------------------------------------
Symbolic constants defined
-------------------------------------------------------------------------------
*/

#define MEM_SRC1        0x00006000   /* Source address (internal memory) for transfer No.1      */
#define MEM_DST1        0xB0000000   /* Destination address (EMIF_CE3) for transfer No.1        */
#define MEM_SRC2        0xB0000000   /* Source address (EMIF_CE3) for transfer No.2             */
#define MEM_DST2        0x0000A000   /* Destination address (internal memory) for transfer No.2 */
#define EL_COUNT        3500         /* Element count for transfer                              */
#define EMIF_CE3        0x01800014   /* EMIF chip enable No.3 control register address          */
#define COEF            45           /* Number of FIR filter coefficients                       */


/*
-------------------------------------------------------------------------------
External & global data variables
-------------------------------------------------------------------------------
*/

extern int shift_val;                /* Defines the right shifting in the Q.N format            */
extern short h[COEF];                /* FIR coefficients                                        */
short R_in[COEF];                    /* Filter delay line                                       */
int pass;                            /* Number of waveform cycles successfully analysed         */
int gain=5;                          /* Gain value associated with the specific input           */
                                     /* channel used, Ch#1 - gain x5 & Ch#2 - gain x50          */
```

```
/*
-------------------------------------------------------------------------------------------------
Function:     trans_rec_sample()
Description:  Transmits data from internal memory to external peripheral device and receives
              returned data from the EMIF back to internal memory.
Inputs:       -
Outputs:      -
Returns:      -
-------------------------------------------------------------------------------------------------
*/

void trans_rec_sample(void){

  unsigned int LOC = 0;

  for(LOC=0;LOC<=EL_COUNT-1;LOC+=4){
      *(unsigned int *)MEM_DST1 = *(unsigned int *)(MEM_SRC1 + LOC);
      *(unsigned int *)(MEM_DST2 + LOC) = *(unsigned int *)MEM_SRC2;
  }

  return;

} /* end trans_rec_sample */

/*
-------------------------------------------------------------------------------------------------
Function:     shift_input()
Description:  Sample data shifted to correctly represent 10-bit input with lsb.
Inputs:       -
Outputs:      -
Returns:      -
-------------------------------------------------------------------------------------------------
*/

void shift_input(void){

  unsigned int LOC = 0, *x, org_val;

  for(LOC=0;LOC<=EL_COUNT-1;LOC++){
      x = (unsigned int*)MEM_DST2+LOC;
      if(gain==50)
         org_val=*x&0x000003FF;        /* Depending on the channel in operation, (either Ch#1   */
      else{                            /* with gain set to 50 or Ch#2 with gain set to 5)       */
         org_val=*x&0x03FF0000;        /* only the respective 10-bit interfaced data lines will */
         org_val=org_val>>16;          /* be considered, all the other data lines are ignored.  */
      }                                /* An adjustment is also made which allows the selected  */
      *x=org_val;                      /* 10-bit data to be more appropriately represented.     */
  }

  return;

} /* end shift_input */
```

```
/*
-------------------------------------------------------------------------------
Function:       freq_anal()
Description:    Analyses a signal's fundamental frequency by recording the average number of
                samples per cycle
Inputs:         coef    =       number of filter coefficients
Outputs:        -
Returns:        tot     =       total number of samples per cycle
-------------------------------------------------------------------------------
*/

int freq_anal(int coef){

    unsigned int LOC = 0, *x;
    int delay,sample[2]={0,0},pos=0,neg=0;
    int tot=0,tot9=0,type;
    float ref,freq;

    pass=0;
                                        /* Phase delay is a function of the number of       */
    delay=coef;                         /* coefficients used, hence only samples succeeding */
    for(LOC=delay;LOC<=EL_COUNT-1;LOC++){   /* this delay are analysed.                     */
        x = (unsigned int*)MEM_DST2+LOC;
        sample[0]=*x;
        if(sample[0]<512)               /* The total number of samples per cycle are        */
            ++neg;                      /* determined by using the mid-range value as a     */
        else                            /* crossing-point and therefore indicate the        */
            ++pos;                      /* start/end of a cycle.                            */
        if(sample[0]>=512 && sample[1]<512){
            ++pass;
            tot=pos+neg-1;
            ref=ceil((float)(tot*0.35));
            if(neg<=ref || pos<=ref){
                pass=0;
                tot9=0;                 /* The sample count is reset if an unexpected or    */
            }                           /* out of range signal is encountered.              */
            else                        /* By knowing the EMIF's sampling rate(Fs= 2MSPS)   */
                tot9+=tot;              /* & the number of samples recorded over a specified */
            if(tot<=9)                  /* number of cycles (9), a signal's fundamental     */
                pass=0;                 /* frequency can be found:                          */
            if(pass==9){                /* signal freq = Fs / no. of samples per cycle      */
                if(gain==50){
                    puts("Ch#1(x50) Signal Results");
                    puts(" ");
                }
                else{
                    puts("Ch#2(x5) Signal Results");
                    puts(" ");
                }
                puts("Frequency(kHz):");    /* The samples of 9 successive cycles are used  */
                freq=(2e6)/(tot9/9.0);      /* instead of a single cycle in an effort to    */
                type=0;                     /* completely describe or cover an expected waveform */
                display(freq, type);        /* with a whole number of samples.              */
                break;                      /* E.g. The expected number of samples required to */
            }                               /* accommodate a 3MHz signal should need 100 samples */
            pos-1;                          /* over 9 cycles, with a sampling rate of 100/3 MSPS. */
            neg=0;                          /* Considering a lower number of cycles could result */
        }                                   /* in some signal data loss with part of a waveform */
        sample[1]=sample[0];                /* being left unsampled and therefore could cause an */
    }                                       /* unnecessary error in the frequency calculation. */
    return tot;                             /* The display() function is also called after the  */
                                            /* measured property (including units) have been    */
} /* end freq_anal */                       /* indicated within the standard output window.     */
```

```
/*
-------------------------------------------------------------------------------
Function:       phase_anal()
Description:    A signal's phase delay is determined by analysing the waveform's mid-value crossing
                point and compares this point to the generated reference signal.
Inputs:         sample_tot =total number of samples per cycle
Outputs:        -
Returns:        -
-------------------------------------------------------------------------------
*/

void phase_anal(int sample_tot){

  unsigned int LOC=0,*x;
  int sample[2]={0,0},i=0,type;
  float phase;

  for(LOC=0;LOC<=sample_tot-1;LOC++){          /* Number of iteration counted relates to      */
        x = (unsigned int*)MEM_DST2+LOC;      /* number of samples which in turn determines  */
     sample[0]=*x;                            /* any time delay.                             */
     if(sample_tot<=8||sample_tot>=750)
        break;
     else if(sample[0]>=512 && sample[1]<512){
        phase = ((i-1)/2)*1000;
        puts("Phase Delay(ns):");
        phase=(int)(phase);
        type=1;
        display(phase,type);
        break;
     }
     ++i;
     sample[1]=sample[0];
  }

  return;

}/* end phase_anal */
```

```
/*
-------------------------------------------------------------------------------------------------
Function:       volt_anal()
Description:    A signal's mean peak-to-peak voltage is determined by analysing the waveform's
                maximum & minimum levels over a set number of cycles.
Inputs:         sample_tot      =       total number of samples per cycle
Outputs:        -
Returns:        -
-------------------------------------------------------------------------------------------------
*/

void volt_anal(int sample_tot){

     unsigned int LOC=0,LOC1=0,*x;
     int sample[2]={512,512},i,N=9;        /* The omitted, initial filtered data which possibly */
     int max=512,min=512,p_to_p=0,type;    /* includes some transience has been extended beyond */
                                           /* the expected phase delay period. This is          */
     for(LOC=50;LOC<=EL_COUNT-1;LOC++){    /* performed to ensure that the peak-to-peak voltage */
       x = (unsigned int*)MEM_DST+LOC;     /* result is limited to the signal's steady-state    */
       sample[0]=*x;                       /* range.                                            */
        if(sample[0]>=512 && sample[1]<512){
            for(i=0;i<=N-1;++i){
                for(LOC1=LOC;LOC1<=sample_tot+LOC;LOC1++){
                    x = (unsigned int*)MEM_DST+LOC1;
                    if(*x>max)
                        max=*x;
                    if(*x<min)
                        min=*x;
                }                          /* The peak-to-peak voltage of a set number of       */
                LOC=LOC1;                  /* waveforms are summed together. The maximum &      */
                p_to_p+=max-min;           /* minimum levels are reset for each cycle. The mean */
                max=512;                   /* Vp-p is then displayed with an appropriate comment*/
                min=512;                   /* with units included. This mean result should      */
            }                              /* compensate for any excess signal noise.           */
            break;                         /* The initial if statement is required only once to */
        }                                  /* indicate the starting point of the voltage        */
        sample[1]=sample[0];               /* measurement, hence a break command is called to   */
    }                                      /* prevent the recorded voltage measurement from     */
    i=0;                                   /* being overwritten and suitable causes an          */
    while(i<1){                            /* unconditional exit from the primary for loop.     */
      ++i;
      if(pass<9)                           /* If the targeted 9 successive periodic waveforms   */
         break;                            /* within range are not found, the potentially       */
      if(sample_tot<=8||sample_tot>=750)   /* erroneous voltage result is ignored. By using the */
         break;                            /* samples per cycle value, upper and lower signal   */
      else{                                /* frequency limits are set at approx. 3MHz & 30kHz  */
         puts("Vp-p(mV):");                /* respectively. Voltage results relating to signals */
         if(gain==50)                      /* at frequencies beyond the expected measurement    */
             p_to_p=p_to_p/(N*50);         /* range are rejected.                               */
         else
            p_to_p=p_to_p/(N*5);
         type=1;
         display(p_to_p,type);
         puts(" ");
         puts(" ");
      }
    }

    return;

} /* end volt_anal */
```

```
/*
-------------------------------------------------------------------------------
Function:       display()
Description:    To display results by utilising the standard output (stdout) window this function
                converts measured decimal results into the appropriate string character format.
Inputs:         var     =   variable result (e.g. frequency in kHz or p-p voltage in units of mV)
                option  =   fixed or floating-point display options
Outputs:        -
Returns:        -
-------------------------------------------------------------------------------
*/

void display(float var, int option){

     char string[8];            /* To display a maximum of 4 significant figures before and 2   */
     int i,j,k,x,y;             /* after the decimal point, integer and real number variables   */
     float tmp,tmp2;            /* are declared.                                                */

     if(option==0){             /* Two primary display type options are available; option 0 is   */
       tmp = var / 1000000;     /* used to display floating-point variables, while option 1     */
       tmp2 = var / 1000;       /* applies to integer-type variables, as required.              */
     }
     else{
       tmp = var / 1000;
       tmp2= 0;                 /* The real tmp value is converted into its integer             */
     }                          /* representation causing the digits below the decimal point to */
     x = (int)tmp;              /* be removed, as is also the case for the tmp2 variable.       */
     y = (int)tmp2;             /* Hence, string[0] will therefore initially represent the ASCII*/
     string[0] = x + '0';       /* code for the thousand digit (thousand kHz / thousand mV).    */
     for(i=1 ; i<=3 ; i++) {    /* The first for loop is used to display, the hundred, ten and  */
       tmp = (tmp - x) * 10;    /* one digits. Within the brackets of the initial statement     */
       x = (int)tmp;            /* of the for loop, the thousand integer digit is subtracted    */
       string[i] = x + '0';     /* from its floating point value which causes the remaining most*/
     }                          /* significant hundreds digit being below the radix point.      */
     string[4] = '.';           /* Therefore * by 10 and then finding the integer value will    */
     for(j=5 ; j<=6 ; j++) {    /* give the suitable hundreds value only, the same loop         */
       tmp2 = (tmp2 - y) * 10;  /* continues to find the tens & ones digit. The second for loop */
       y = (int)tmp2;           /* repeats the previous procedure for the digits below the      */
       string[j] = y + '0';     /* decimal point, '0' is added to these digits to give their    */
     }                          /* ASCII code character values.                                 */
     string[7] = '\0';
     if(option==1) {            /* The conditional statements are included to allow the most    */
       for(k=4;k<=7;k++)        /* significant non-zero figure to be displayed first, while     */
           string[k] = '\0';    /* removing any unnecessary digits below the decimal point.     */
     }
     if(string[0] == '0') {
       for(k=0;k<=7;k++)
           string[k-1] = string[k];
           if(string[0] == '0') {
               for(k=0;k<=6;k++)
                   string[k-1] = string[k];
           }
           else {
               for(k=0;k<=6;k++)
                   string[k] = string[k];
           }
     }
     else {
       for(k=0;k<=7;k++)
           string[k] = string[k];
     }
     puts(string);              /* Finally another function is called to write the string of    */
                                /* characters in the standard output (stdout) window.           */
     return;

} /* end display */
```

```
/*
-------------------------------------------------------------------------------
Function:       filter_init()
Description:    Initialisation of the filter variable
Inputs:         coef    =    number of filter coefficients
Outputs:        -
Returns:        -
-------------------------------------------------------------------------------
*/

void filter_init (void)
{
   int i;

   for(i=0; i<=COEF; i++)
        R_in[i]=0;

   return;

} /* end filter_init */


/*
-------------------------------------------------------------------------------
Function:       fir_filter()
Description:    Sample by sample FIR filtering
Inputs:         input   =       input sample to the filter
                shift   =       defines the Q.N format, shift the product right by 15 bits
Outputs:        -
Returns:        temp    =       filtered sample
-------------------------------------------------------------------------------
*/

short fir_filter (short input, int shift)
{
   int i;
   short temp;
   int Acc;

   for (i=COEF-1; i>0; i--)
     R_in[i]=R_in[i-1];                                    /* Shift delay samples        */
   R_in[0] = input;                                        /* Update most recent sample  */

   Acc = 0;
   for (i=0; i<COEF; i++)
      Acc += (int)(short)h[i] * (int)(short)R_in[i];       /* Sum operation in C         */
   temp = (short) (Acc>>shift);                            /* Output in 16-bit format    */

   return temp;

} /* end fir_filter */
```

```
/*
-------------------------------------------------------------------------------
Main programme
-------------------------------------------------------------------------------
*/

void main(void){
                                /* Main programme variables declared, the EMIF CE3 control   */
  unsigned int LOC=0, *x,*y;    /* register is configured with two setup, three strobe & zero*/
  int sample_len,mod_val;       /* hold read/write clock(ELCKOUT) cycles. Therefore sample   */
  short temp,output;            /* rate is 2MSPS due to this active read/write period of     */
  *(int *)EMIF_CE3 = 0x20C20320;/* 100ns (10 cycles/100MHz) being combined with the inherent */
                                /* inactive CPU period between successive read/write commands*/
  while(1){                     /* being 400ns (15*4/150MHz), with this data transfer method.*/
    trans_rec_sample();         /* Data transferring                                         */
    shift_input();              /* 10-bit input data selected & fittingly adjusted           */
    filter_init();              /* Initialise the FIR filter variable
    sample_len = freq_anal(COEF); /* Input signal's frequency calculated & displayed         */
    phase_anal(sample_len);     /* Phase delay between input and reference signal calculated */
    volt_anal(sample_len);      /* Input signal's mean Vp-p calculated & displayed           */

    for(LOC=0;LOC<=EL_COUNT-1;LOC++){     /* Sample by sample filtering with signal limits set */
      x = (unsigned int*)MEM_DST2+LOC;
      temp = (short)(*x & 0xffff);
      output =  fir_filter(temp,shift_val);
      if(output>1023)
          output=1023;
      else if(output<0)
          output=0;
      else
          output=output;
      *x = output;
    }

    for(LOC=0;LOC<=EL_COUNT-1;LOC++){     /* Filter delay considered                   */
      x = (unsigned int*)MEM_DST2+LOC+((COEF-1)/2);
      y = (unsigned int*)MEM_DST2+LOC;
      mod_val=*x;
      *y=mod_val;
    }
  }
} /* end main programme */
```

A.4.7 data_in_phase+_pcb.c

```
/*
-------------------------------------------------------------------------------
Filename:       data_in_phase+_pcb.c
Description:    This programme determines the phase difference between two independent input
                signals. The calculated result is displayed instantly in real-time. The frequency
                and amplitude of each signal is also calculated and displayed, after the signals
                have been digital filtered.
-------------------------------------------------------------------------------
*/

/*
-------------------------------------------------------------------------------
Include files
-------------------------------------------------------------------------------
*/

#include <c6x.h>
#define CHIP_6711                   /* DSP chip defined */
#include <csl.h>
#include <csl_edma.h>
#include <csl_dat.h>
#include <stdio.h>
#include <math.h>
#include "c6711dsk.h"

/*
-------------------------------------------------------------------------------
Function prototypes
-------------------------------------------------------------------------------
*/

void submit_qdma(void);
void adjust_input(void);
float freq_anal(void);
void phase_anal(float range);
void volt_anal(float range);
void display(float var, int option);
void filter_init(void);
short fir_filter (short input, int shift);


/*
-------------------------------------------------------------------------------
Symbolic constants defined
-------------------------------------------------------------------------------
*/

#define MEM_SRC          0xA0000000      /* Source address (EMIF_CE3) for transfer          */
#define MEM_DST          0x00007000      /* Destination address for transfer                */
#define EL_COUNT         0x2300          /* Element count for transfer                      */
#define COEF             45              /* Number of FIR filter coefficients               */
#define EMIF_CE3         0x01800014      /* EMIF chip enable No.3 control register address  */

/*
-------------------------------------------------------------------------------
External & global data variables
-------------------------------------------------------------------------------
*/

extern int shift_val;                    /* Defines the right shifting in the Q.N format    */
extern short h[COEF];                    /* FIR coefficients                                */
short R_in[COEF];                        /* Filter delay line                               */
int pass;                                /* Number of waveform cycles successfully analysed */
```

```
/*
-------------------------------------------------------------------------------------------------
Function:       submit_qdma()
Description:    Submit a QDMA request to transfer the sampled data to internal (L2) memory.
Inputs:         -
Outputs:        -
Returns:        -
-------------------------------------------------------------------------------------------------
*/

void submit_qdma(void){

    EDMA_Config config;
    config.opt = (Uint32)                   /* EDMA channel options selected (0x20200001):      */
        ((EDMA_OPT_PRI_HIGH    << _EDMA_OPT_PRI_SHIFT   )  /* High priority EDMA transfer,     */
       | (EDMA_OPT_ESIZE_32BIT << _EDMA_OPT_ESIZE_SHIFT )  /* 32-bit element size,             */
       | (EDMA_OPT_2DS_NO      << _EDMA_OPT_2DS_SHIFT   )  /* 1-dimensional source,            */
       | (EDMA_OPT_SUM_NONE    << _EDMA_OPT_SUM_SHIFT   )  /* Fixed source address mode,       */
       | (EDMA_OPT_2DD_NO      << _EDMA_OPT_2DD_SHIFT   )  /* 1-dimensional destination,       */
       | (EDMA_OPT_DUM_INC     << _EDMA_OPT_DUM_SHIFT   )  /* Increment destination address,   */
       | (EDMA_OPT_TCINT_NO    << _EDMA_OPT_TCINT_SHIFT )  /* Transfer complete indicator      */
       | (EDMA_OPT_TCC_DEFAULT << _EDMA_OPT_TCC_SHIFT   )  /* disabled, with no code set,      */
       | (EDMA_OPT_LINK_NO     << _EDMA_OPT_LINK_SHIFT  )  /* Event parameter link disabled,   */
       | (EDMA_OPT_FS_YES      << _EDMA_OPT_FS_SHIFT    ));/* Frame/block synchronised.        */

    config.src = (unsigned int)MEM_SRC;       /* Source address for transfer      (0xB0000000) */
    config.cnt = (unsigned int)EL_COUNT;      /* Element count for transfer       (0x00002300) */
    config.dst = (unsigned int)MEM_DST;       /* Destination address for transfer (0x00007000) */
    config.idx = (unsigned int)0;             /* Element index offset addressing  (0x00000000) */

    EDMA_qdmaConfig(&config);

    return;

} /* end submit_qmda */

/*
-------------------------------------------------------------------------------------------------
Function:       adjust_input()
Description:    Sample data adjusted to correctly represent 10-bit data of each input channel.
Inputs:         -
Outputs:        -
Returns:        -
-------------------------------------------------------------------------------------------------
*/

void adjust_input(void){

    unsigned int LOC = 0, org_val;
    short *x;
    float y;

    for(LOC=0;LOC<=EL_COUNT-1;LOC++){
       x = (short*)MEM_DST+LOC;
       org_val=*x&0x03FF;          /* With both channels being used simultaneously, each 10-bit */
       y=org_val;                  /* sampled data must be considered separately, while the     */
       *x=(short)(y);              /* remaining unused bits are ignored.                        */
    }

    return;

} /* end adjust_input */
```

```
/*
-------------------------------------------------------------------------------
Function:       freq_anal()
Description:    Analyses the fundamental frequency of each input signal by recording the average
                number of samples per cycle
Inputs:         -
Outputs:        -
Returns:        samp =   total number of samples per cycle
-------------------------------------------------------------------------------
*/

float freq_anal(void){

   unsigned int LOC = 0;
   short *x;
   int sample[2]={0,0},pos=0,neg=0;
   int tot=0,tot9=0,type;
   float ref,freq,samp;
   pass=0;
   for(LOC=0;LOC<=EL_COUNT-1;LOC+=2){      /* Channel No.1's signal frequency is now determined */
      x = (short*)MEM_DST+LOC;
      sample[0]=*x;
      if(sample[0]<512)
         ++neg;
      else
         ++pos;
      if(sample[0]>=512 && sample[1]<512){
         ++pass;
         tot=pos+neg-1;                   /* The total number of samples per cycle are         */
         ref=ceil((float)(tot*0.35));     /* determined by using the mid-range value as a      */
         if(neg<=ref || pos<=ref){        /* crossing-point and therefore indicate the         */
             pass=pass-1;                 /* start/end of a cycle.                             */
         }
         else                             /* By knowing the EMIF's sampling rate(Fs~100/4MSPS) */
             tot9+=tot;                   /* & the number of samples recorded over a specified */
         if(tot<=9)                       /* number of cycles (9), a signal's fundamental      */
             pass=0;                      /* frequency can be found:                           */
         if(pass==9  && tot9>=60){        /* signal freq = Fs / no. of samples per cycle       */
             samp=tot9/9.0;
             puts("Ch#1-Frequency(kHz):");/* The samples of 9 successive cycles are used       */
             freq=25e6/(tot9/9.0);        /* instead of a single cycle in an effort to         */
             type=0;                      /* completely describe or cover an expected waveform */
             display(freq, type);         /* with a whole number of samples.                   */
             tot9=0;                      /* Eg. The expected number of samples required to    */
             break;                       /* accommodate a 3MHz signal should need 100 samples */
         }                                /* over 9 cycles, with a sampling rate of 100/3MSPS. */
         pos=1;                           /* Considering a lower number of cycles could result */
         neg=0;                           /* in some signal data loss with part of a waveform  */
      }                                   /* being left unsampled and therefore could cause an */
      sample[1]=sample[0];                /* unnecessary error in the frequency calculation.   */
   }
   pass=0;
   for(LOC=1;LOC<=EL_COUNT-1;LOC+=2){      /* Channel No.2's signal frequency is now determined */
    x = (short*)MEM_DST+LOC;
      sample[0]=*x;
      if(sample[0]<512)
         ++neg;
      else
         ++pos;
      if(sample[0]>=512 && sample[1]<512){
         ++pass;
         tot=pos+neg-1;
         ref=ceil((float)(tot*0.35));
         if(neg<=ref || pos<=ref){
             pass=pass-1;
         }
         else
             tot9+=tot;
         if(tot<=9)
             pass=0;
         if(pass==9  && tot9>=60){
             samp=tot9/9.0;
             puts("Ch#2-Frequency(kHz):");
             freq=25e6/(tot9/9.0);
             type=0;
             display(freq, type);         /* The display() function is also called after the   */
             tot9=0;                      /* measured property including units have been       */
             break;                       /* indicated within the standard output window.      */
         }
         pos=1;
         neg=0;
      }
      sample[1]=sample[0];
    }
    return samp;

}/* end freq_anal */
```

```
/*
-------------------------------------------------------------------------------
Function:       phase_anal()
Description:    Determines the phase difference between two input signals
Inputs:         sample_tot   =   total number of samples per cycle
Outputs:        -
Returns:        -
-------------------------------------------------------------------------------
*/

void phase_anal(float sample_tot){

     unsigned int LOC=0;
     short *x, sample[2]={512,512};
     int i=0,j=0,type;
     float phase;

     for(LOC=2;LOC<=(sample_tot*2)-1;LOC+=2){     /* Number of iterations relates to the       */
        x = (short*)MEM_DST+LOC;                  /* number of samples before a mid-value      */
        sample[0]=*x;                             /* crossing-point is found for each signal   */
        if(sample_tot<=8||sample_tot>=750)
            break;
        else if(sample[0]>=512 && sample[1]<512)
            break;
        else{
            ++i;
            sample[1]=sample[0];
        }
     }
     sample[0]-512;
     sample[1]-512;

     for(LOC=3;LOC<=(sample_tot*2)-1;LOC+=2){
        x = (short*)MEM_DST+LOC;
        sample[0]=*x;
        if(sample_tot<=8||sample_tot>=750){
            sample[0]=512;
            sample[1]=512;
            i=0;
            j=0;
            break;
        }
        else if(sample[0]>=512 && sample[1]<512){ /* Once the sample difference between the    */
            phase = ((i-j)/sample_tot)*360*1000;  /* two signals is determined, the number of  */
            if(phase<0){                          /* samples per cycle and an appropriate      */
                puts("Phase Diff.(deg.):");       /* multiplying factor are used to calculate  */
                type=2;                           /* the phase difference in units of degrees. */
                display(phase,type);
                puts(" ");
                sample[0]-512;
                sample[1]-512;
                i=0;
                j=0;
                break;
            }
            else{
                puts("Phase Diff.(deg.):");
                type=0;
                display(phase,type);
                puts(" ");
                sample[0]=512;
                sample[1]-512;
                i=0;
                j=0;
                break;
            }
        }
        else{
            ++j;
            sample[1]=sample[0];
        }
    }
    return;

}/* end phase_anal */
```

```
/*
-------------------------------------------------------------------------------
Function:       volt_anal()
Description:    Determines the mean peak-to-peak voltage of each input signal
Inputs:         sample_tot   =   total number of samples per cycle
-------------------------------------------------------------------------------
*/

void volt_anal(float sample_tot){
     unsigned int LOC=0,LOC1=0;
     short *x;
     int sample[2]={512,512},i,N=9;          /* The omitted, initial filtered data which possibly */
     int max=512,min=512,p_to_p=0,type;      /* includes some transience has been extended beyond */
     sample_tot=(int)(sample_tot);           /* the expected phase delay period. This is          */
     for(LOC=0;LOC<=EL_COUNT-1;LOC+=2){      /* performed to ensure that the peak-to-peak voltage */
          x = (short*)MEM_DST+LOC;           /* result is limited to the signal's steady-state    */
          sample[0]=*x;                      /* range.                                            */
          if(sample[0]>=512 && sample[1]<512){
             for(i=0;i<=N-1;++i){
                for(LOC1=LOC;LOC1<=(sample_tot*2)+LOC;LOC1+=2){
                    x = (short*)MEM_DST+LOC1;
                    if(*x>max)
                       max=*x;
                    if(*x<min)
                       min=*x;
                }                            /* The peak-to-peak voltage of a set number of       */
                LOC=LOC1;                    /* waveforms are summed together. The maximum &      */
                p_to_p+=max-min;             /* minimum levels are reset for each cycle. The mean */
                max=512;                     /* Vp-p is then displayed with an appropriate        */
                min=512;                     /* comment with units included. This mean result     */
             }                               /* should compensate for any excess signal noise.    */
             break;                          /* The initial if statement is required only once to */
          }                                  /* indicate the starting point of the voltage        */
          sample[1]=sample[0];               /* measurement, hence a break command is called to   */
     }                                       /* prevent the recorded voltage measurement from     */
     i=0;                                    /* being overwritten and suitable causes an          */
     while(i<1){                             /* unconditional exit from the primary for loop.     */
          ++i;
          if(pass<9)                         /* If the targeted 9 successive periodic waveforms   */
             break;                          /* within range are not found, the potentially       */
          if(sample_tot<=8||sample_tot>=750)/* erroneous voltage result is ignored. By using the */
             break;                          /* samples per cycle value, upper and lower signal   */
          else{                              /* frequency limits are set at approx. 3MHz & 30kHz  */
             puts("Ch#1-Vp-p(mV):");         /* respectively. Voltage results relating to signals */
             p_to_p=p_to_p/(N*50);           /* at frequencies beyond the expected measurement    */
             type=1;                         /* range are rejected.                               */
             display(p_to_p,type);
             p_to_p=0;
          }
     }
     for(LOC=1;LOC<=EL_COUNT-1;LOC+=2){      /* The received signal of channel No.2 is now        */
          x = (short*)MEM_DST+LOC;           /* analysed.                                         */
          sample[0]=*x;
          if(sample[0]>=512 && sample[1]<512){
             for(i=0;i<=N-1;++i){
                for(LOC1=LOC;LOC1<=(sample_tot*2)+LOC;LOC1+=2){
                    x = (short*)MEM_DST+LOC1;
                    if(*x>max)
                      max=*x;
                    if(*x<min)
                       min=*x;
                }
                LOC=LOC1;
                p_to_p+=max-min;
                max=512;
                min=512;
             }
             break;
          }
          sample[1]=sample[0];
     }
     i=0;
     while(i<1){
          ++i;
          if(pass<9)
             break;
          if(sample_tot<=8||sample_tot>=750)
             break;
          else{
             puts("Ch#2-Vp-p(mV):");
          p_to_p=p_to_p/(N*5);
             type=1;
             display(p_to_p,type);
          }
     }
     return;
  }/* end volt_anal */
```

```
/*
-------------------------------------------------------------------------------------------
Function:       display()
Description:    To display results by utilising the standard output (stdout) window this function
                converts measured decimal results into the appropriate string character format.
Inputs:         var     =   variable result (e.g. frequency in kHz or p-p voltage in units of mV)
                option  =   fixed or floating-point display options
Outputs:        -
Returns:        -
-------------------------------------------------------------------------------------------
*/

void display(float var, int option){

     char string[8];            /* To display a maximum of 4 significant figures before and 2   */
     int i,j,k,x,y;             /* after the decimal point, integer and real number variables   */
     float tmp,tmp2;            /* are declared.                                                */

     if(option==0){             /* Two primary display type options are available; option 0 is   */
       tmp = var / 1000000;     /* used to display floating-point variables, while option 1     */
       tmp2 = var / 1000;       /* applies to integer-type variables, as required.              */
     }
     else{
       tmp = var / 1000;
       tmp2= 0;                 /* The real tmp value is converted into its integer             */
     }                          /* representation causing the digits below the decimal point to */
     x = (int)tmp;              /* be removed, as is also the case for the tmp2 variable.       */
     y = (int)tmp2;             /* Hence, string[0] will therefore initially represent the ASCII*/
     string[0] = x + '0';       /* code for the thousand digit (thousand kHz / thousand mV).    */
     for(i=1 ; i<=3 ; i++) {    /* The first for loop is used to display, the hundred, ten and  */
        tmp = (tmp - x) * 10;   /* one digits. Within the brackets of the initial statement     */
        x = (int)tmp;           /* of the for loop, the thousand integer digit is subtracted    */
        string[i] = x + '0';    /* from its floating point value which causes the remaining most*/
     }                          /* significant hundreds digit being below the radix point.      */
     string[4] = '.';           /* Therefore * by 10 and then finding the integer value will    */
     for(j=5 ; j<=6 ; j++) {    /* give the suitable hundreds value only, the same loop         */
        tmp2 = (tmp2 - y) * 10;/* continues to find the tens & ones digit. The second for loop */
        y = (int)tmp2;          /* repeats the previous procedure for the digits below the      */
        string[j] = y + '0';    /* decimal point, '0' is added to these digits to give their    */
     }                          /* ASCII code character values.                                 */
     string[7] = '\0';
     if(option==1) {            /* The conditional statements are included to allow the most    */
       for(k=4;k<=7;k++)        /* significant non-zero figure to be displayed first, while     */
           string[k] = '\0';    /* removing any unnecessary digits below the decimal point.     */
     }
     if(string[0] == '0') {
        for(k=0;k<=7;k++)
            string[k-1] = string[k];
            if(string[0] == '0') {
                for(k=0;k<=6;k++)
                    string[k-1] = string[k];
            }
            else {
                for(k=0;k<=6;k++)
                    string[k] = string[k];
            }
     }
     else {
        for(k=0;k<=7;k++)
            string[k] = string[k];
     }
     puts(string);              /* Finally another function is called to write the string of    */
                                /* characters in the standard output (stdout) window.           */
     return;


} /* end display */
```

```
/*
-------------------------------------------------------------------------------
Function:       filter_init()
Description:    Initialisation of the filter variable
Inputs:         coef    =    number of filter coefficients
Outputs:        -
Returns:        -
-------------------------------------------------------------------------------
*/

void filter_init (void)
{
   int i;

   for(i=0; i<=COEF; i++)
        R_in[i]=0;

   return;

} /* end filter_init */


/*
-------------------------------------------------------------------------------
Function:       fir_filter()
Description:    Sample by sample FIR filtering
Inputs:         input   =       input sample to the filter
                shift   =       defines the Q.N format, shift the product right by 15 bits
Outputs:        -
Returns:        temp    =       filtered sample
-------------------------------------------------------------------------------
*/

short fir_filter (short input, int shift)
{
   int i;
   short temp;
   int Acc;

   for (i=COEF-1; i>0; i--)
     R_in[i]=R_in[i-1];                                    /* Shift delay samples        */
   R_in[0] = input;                                        /* Update most recent sample  */

   Acc = 0;
   for (i=0; i<COEF; i++)
      Acc += (int)(short)h[i] * (int)(short)R_in[i];       /* Sum operation in C         */
   temp = (short) (Acc>>shift);                            /* Output in 16-bit format    */

   return temp;

} /* end fir_filter */
```

```
/*
-------------------------------------------------------------------------------
Main programme
-------------------------------------------------------------------------------
*/

void main(void){
                                        /* Main programme variables declared and the EMIF CE3 */
  unsigned int LOC=0;                   /* control register is configured with one setup,     */
  int sample_len,mod_val;               /* three strobe & zero hold clock (ELCKOUT) cycles.   */
  short *x,*y,temp,output;              /* Therefore an approx. sample rate of 100MSPS/4 is   */
  *(int *)EMIF_CE3 = 0x00010320;        /* employed. Also note the memory type is set to      */
                                        /* 32-bit asynchronous memory.                        */
  while(1){
    submit_qdma();                      /* Interface buffer to internal memory data transfer  */
    adjust_input();                     /* 10-bit input data selected & fittingly adjusted for */
    filter_init();                      /* each channel, then initialise the filter variable  */

    for(LOC=0;LOC<=EL_COUNT-1;LOC+=2){  /* Sample by sample filtering with signal limits set  */
        x = (short*)MEM_DST+LOC;        /* for channel No.1.                                  */
        temp = (short)(*x & 0xffff);
        output =  fir_filter(temp,shift_val);
        if(output>1023)
            output=1023;
        else if(output<0)
            output=0;
        else
            output=output;
        *x = output;
    }

    for(LOC=0;LOC<=EL_COUNT-1;LOC+=2){
        x = (short*)MEM_DST+LOC+((COEF-1));
        y = (short*)MEM_DST+LOC;
        mod_val=*x;
        *y=mod_val;
    }

    for(LOC=1;LOC<=EL_COUNT-1;LOC+=2){  /* Sample by sample filtering with signal limits set  */
        x = (short*)MEM_DST+LOC;        /* for channel No.2.                                  */
        temp = (short)(*x & 0xffff);
        output =  fir_filter(temp,shift_val);
        if(output>1023)
            output=1023;
        else if(output<0)
            output=0;
        else
            output=output;
        *x = output;
    }

    for(LOC=1;LOC<=EL_COUNT-1;LOC+=2){  /* Filter delay considered                             */
        x = (short*)MEM_DST+LOC+((COEF-1));
        y = (short*)MEM_DST+LOC;
        mod_val=*x;
        *y=mod_val;
    }

    sample_len = freq_anal();           /* Input signal's frequency calculated & displayed    */
    volt_anal(sample_len);              /* Input signal's mean Vp-p calculated & displayed    */
    phase_anal(sample_len);             /* Phase difference between input signals calculated  */
                                        /* and displayed.                                     */
  }
} /* end main programme */
```

Bibliography

[1-1] http://www.pei-tech.ie/

[2-1] A. Bindra, "Ultra-Low-Noise Data Acquisition IC Tackles Multiple Sensors," Electron. Design, Vol. 47, No. 22 (1999), pp. 59-60, 62.

[2-2] D. McCartney, A. Sherry, T. Meany, T. Cummins, D. Brannick, L. MacManus, "A Fully Integrated Sensor Interface Chip," ESSCIR'99, pp. 222-225.

[2-3] D. McCartney, A. Sherry, J. O'Dowd, P. Hickey, "A Low-Noise Low-Drift Transducer ADC," IEEE Journal of Solid-State Circuits, Vol. 32, No. 7, pp. 959-967, July 1997.

[2-4] IEEE Standard for a Smart Transducer Interface for Sensors and Actuators – Transducer to Microprocessor Communication Protocols and TEDS format, 1451.2, July 1997.

[2-5] T. Cummins, E. Byrne, D. Brannick, D.A. Dempsey, "An IEEE 1451 Standard Transducer Interface Chip with 12-bit ADC, Two 12-bit DACs, 10-kbyte Flash EEPROM and 8-bit Microcontroller," IEEE Journal of Solid-State Circuits, Vol. 33, No.12 (1998), pp. 2112-2120.

[2-6] P. Malcovati, H. Baltes, F. Maloberti, "Progress in Microsensor Interfaces," Sensors Update, Vol. 1, Issue 1 (1996), pp. 143-171.

[2-7] K.L. Kraver, M.R. Guthaus, T.D. Strong, P.L. Bird, G. Sig Cha, W. Höld, R.B. Brown, "A Mixed-Signal Sensor Interface Microinstrument," Sensors and Actuators A, Vol. 91, Issue 3, 15 July 2001, pp. 266-277.

[2-8] Navid Yazdi, Andrew Mason, Khalil Najafi, Kensall D. Wise, "A Generic Interface Chip for Capacitive Sensors in Low-Power Multi-Parameter Microsystems," Sensors and Actuators A, Vol. 84, Issue 3, 1 September 2000, pp. 351-361.

[2-9] Hiro Yamasaki, "The Future of Sensor Interface Electronics," Sensors and Actuators A, Vol. 56, 1996, pp. 129-133.

[2-10] Khalil Najafi, "Sensor-System Interface: The Influence of On-Chip Electronics," IEEE International Symposium on Circuits and Systems, 1987, pp. 233-236.

[2-11] Henry Baltes, Oliver Brand, "CMOS-Based Microsensors and Packaging," Physical Electronics Laboratory, Zurich, Switzerland, Sensors and Actuators A, Vol. 92, 2001, pp. 1-9.

[2-12] A. Koll, CMOS Capacitive Chemical Microsystems for Volatile Organic Compounds, Ph.D. Thesis No. 13460, ETH Zurich, Switzerland, 1999.

[2-13] A. Koll, S. Kawahito, F. Mayer, C. Hagleitner, D. Scheiwiller, O. Brand, H. Baltes, "Flip-Chip Packaged CMOS Chemical Microsystem for Detection of Volatile Organic Compounds," Proc. SPIE 3328 (1998), pp. 223-232.

[2-14] J. H. Correia, E. Cretu, M. Bartek, R. F. Wolffenbuttel, "A Low-Power Low-Voltage Digital Bus Interface for MCM-Based Microsystems," Delft University of Technology, Dept. of Electrical Engineering.

[2-15] J. H. Huijsing, F. Riedijk, G. v. Horn, "Developments in Integrated Smart Sensors," Proc. Transducers '93, pp. 320-326.

[2-16] S. Ramanathan, V. Visvanathan and S.K. Nandy, "A Computational Engine for Multirate FIR Digital Filtering," Signal Processing, Elsevier Science, Vol. 79, Issue 2, December 1999, pp. 213-222.

[2-17] M. S. Ben Romdhane, V. K. Madisetti, "All-Digital Oversampled Front-End Sensors," IEEE Signal Processing Letters, Vol. 3, No. 2, February 1996, pp. 38-39.

[3-1] Ronald J. Tocci, Neal S. Widner, Digital Systems: Principles and Applications, Seventh Edition, p. 8, © 1998 by Prentice-Hall, Inc.

[3-2] Sergio Franco, Design with Operational Amplifiers and Analog Integrated Circuits, Third Edition, pp. 384-390, © 2002, by The McGraw-Hill Companies, Inc.

[3-3] Sergio Franco, Design with Operational Amplifiers and Analog Integrated Circuits, Third Edition, pp. 359-362, © 2002, by The McGraw-Hill Companies, Inc.

[3-4] Pickering Series 103, Low Capacitance SIL Reed Relay Data Sheets www.pickering.co.uk

[4-1] LM340/LM78XX Series, 3-Terminal Positive Regulators Data Sheets, (DS007781), © 2001 National Semiconductor Corporation.

[4-2] www.semiconductors.philips.com/acrobat/applicationnotes/APPCHP2.pdf, Power Semiconductor Applications, p. 107, © Philips Semiconductors.

[4-3] Paul Horowitz and Winfield Hill, The Art of Electronics, Second Edition, p. 456, © Cambridge University Press 1989.

[4-4] E.C. Young, The Penguin Dictionary of Electronics, Second Edition, p. 47, © Market House Books Ltd, 1988.

[4-5] OPA656 Wideband, Unity-Gain Stable, FET-input Op-Amp Data Sheets, (SBOS196A), © 2001, Texas Instruments Incorporated.

[4-6] CLC449 1.1GHz Ultra Wideband Monolithic Op-Amp Data Sheets, (DSO12715), © 2001, National Semiconductor Corporation.

[5-1] Sergio Franco, Design with Operational Amplifiers and Analog Integrated Circuits, Third Edition, p. xi, © 2002, by The McGraw-Hill Companies, Inc.

[5-2] AD830 High-Speed, Video Difference Amplifier Data Sheets, © Analog Devices, Inc.

[5-3] Paul Horowitz and Winfield Hill, The Art of Electronics, Second Edition, p. 635 © Cambridge University Press 1989.

[5-4] AD6645, 14-bit, 80MSPS A-D Converter Data Sheets, © Analog Devices, Inc., 2002.

[5-5] AD9050 10-bit, 40/60MSPS A-D Converter Data Sheets, © Analog Devices, Inc., 1997.

[5-6] Gert van der Horn and Johan L. Huijsing, Integrated Smart Sensors, Design and Calibration, p. 81, © Kluwer Academic Publishers 1998.

[5-7] AD9760 10-bit, 125MSPS TxDAC® D-A Converter Data Sheets, © Analog Devices, Inc., 2000.

[5-8] Steve Connors, Tim Ludy and Joseph Facca, Real-Time DSP Data Acquisition for High-Speed Measurement, Global DSP, Vol. 3, Issue 2, February 2004, pp.1-6.

[5-9] www.analog.com/dsp

[5-10] www.ti.com/

[6-1] TMS320 Cross-Platform Daughtercard Specifications Revision 1.0 (SPRA711), p.17, Copyright © 2000 Texas Instruments Incorporated.

[6-2] TMS320C6000 Peripherals Reference Guide (SPRU190D), pp.3-2 to 3-15, Copyright © 2001 Texas Instruments Incorporated.

[6-3] Applications Using the TMS320C6000 Enhanced DMA (SPRA636A), p.1, Copyright © 2001 Texas Instruments Incorporated.

[6-4] TMS320C6000 Peripherals Reference Guide (SPRU190D), pp.10-91 to 10-102, Copyright © 2001 Texas Instruments Incorporated.

[6-5] Emmanuel C. Ifeachor and Barrie W. Jervis, Digital Signal Processing, A Practical Approach, Second Edition, pp. 320-321, © Pearson Education Limited 1993, 2002.

[6-6] Galanis D. Michail and Zigouris Th. Evangelos, DSP Systems Design and Implementation, Electronics Laboratory, University of Patras, 2002.

[6-7] Emmanuel C. Ifeachor and Barrie W. Jervis, Digital Signal Processing, A Practical Approach, Second Edition, p. 455, © Pearson Education Limited 1993, 2002.