Texas Instruments

Application Report
SPRA946 - August 2003

Signal Conditioning an LVDT Using a TMS320F2812 DSP

Richard Poley *DSP Field Applications*

ABSTRACT

A linear variable differential transformer (LVDT) is a type of electro-mechanical transducer capable of measuring linear displacement with a high degree of accuracy. The transducer requires external electronic circuitry to produce a stable reference oscillator, and condition the return signal in order to determine displacement. Traditional LVDT circuits are based on analog design techniques, and suffer from several well known limitations such as component aging and drift. This paper presents an alternative design methodology using a digital signal processor (DSP) which is not subject to these effects. The approach also enables the designer to use sophisticated signal conditioning techniques such as software error correction, which can significantly increase the measurement accuracy.

Contents

List of Figures

List of Tables

1 Introduction

Since the 1930s, linear variable differential transformers (LVDTs) have found widespread application in industry for the measurement of displacement. Early development was driven by the needs of the chemical industry to remotely measure process variables, and during World War II development accelerated rapidly as the LVDT found use in aircraft and weapon systems. Today, the LVDT has evolved into a highly accurate and reliable form of displacement transducer, and is widely used in many branches of industry and science.

Since the LVDT is basically a transformer, it requires an alternating signal to excite the primary coil and some form of electronic circuit to condition the return signal in order to extract displacement information. Almost all such circuits in use today are designed with analog components for both functions. While these are usually robust and cheap to produce, their design can be challenging if accurate measurements are required. This is because both the construction of the transducer and the presence of spurious phase shift in the return signal path contribute to non-linear measurement error, which can be difficult to eliminate using analog techniques.

The objectives of this paper are to describe the LVDT transducer and to present a design for a signal conditioning circuit using a digital signal processor, which permits the use of software error correction techniques.

Section 2 briefly describes the operating principle of the LVDT for readers unfamiliar with this type of transducer. Section 3 goes on to describe a typical analog signal conditioning approach and outlines some of the problems associated with it. Section 4 outlines a digital signal conditioning technique using a DSP, describes the design of the hardware circuit and embedded software, and presents some experimental results. Lastly, section 5 discusses the merits of the digital design approach.

2 Description of the LVDT

The LVDT is an electro-mechanical transducer, the input to which is a physical movement of the transducer core, and the output a change in magnetic coupling between the internal windings which can be measured using suitable conditioning electronics. Usually, the output of the conditioning electronics is a stable, DC voltage which is proportional to the core position.

Various types of LVDT transducers exist, including half-bridge configurations, and versions designed to measure rotational rather than linear displacement. For the purposes of this paper, a common type of linear transducer was selected, with two secondary windings connected in the “series opposed” configuration. The transducer was procured from RDP Electronics Limited, and has the type number ACT2000C. It has the following specifications:

- Linear range: ± 50 mm
- Sensitivity: 27.45 mV/V/mm
- Linearity: 0.16%

The transducer construction consists of a cylindrical transformer, with a single primary winding and two secondary windings. All three windings are wound onto a hollow cylindrical bobbin, and the complete assembly is housed within a rigid metal casing. A moveable core is fitted concentrically within the bobbin, and is free to move axially within the transducer body. One end of the core is normally fixed to an extension rod which emerges through the end of the transducer. In use, the transducer body is clamped in position while the extension rod is attached to the component being measured. Electrical connections to the internal windings of the transducer normally emerge through the side of the casing in a captive screened multi-core cable.

Figure 1 shows a cross-sectional diagram of a typical LVDT.

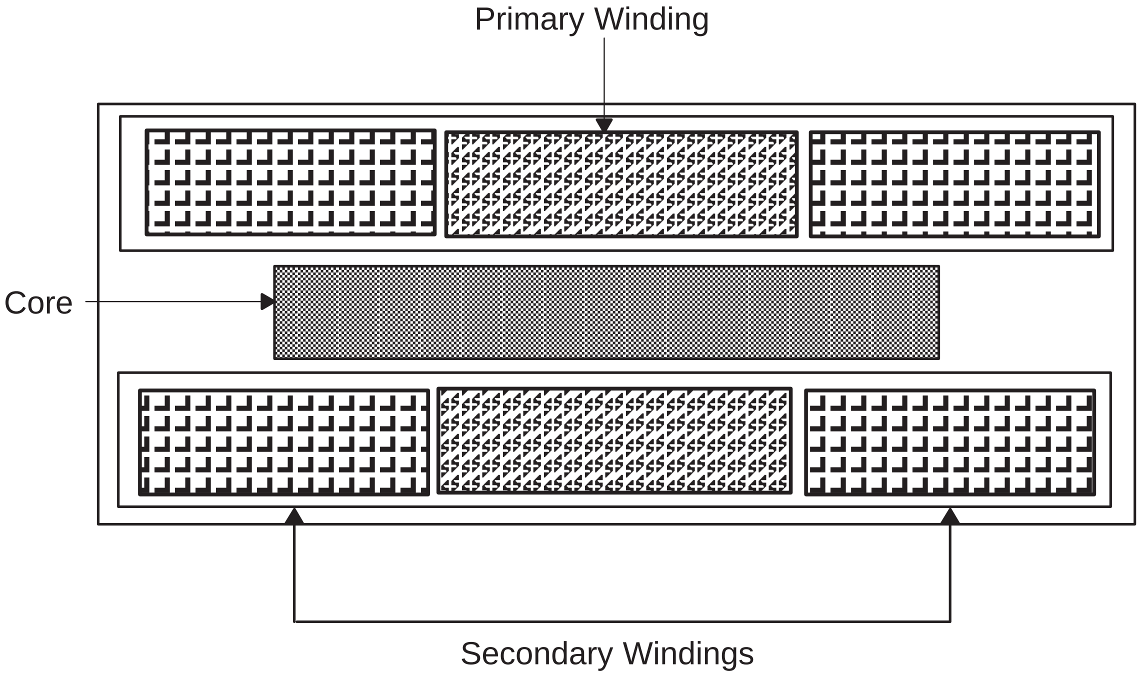


Figure 1. Cross-Section of a Typical LVDT

The primary coil is connected to a sinusoidal signal of fixed amplitude and frequency, and the core electrically couples the resultant magnetic flux onto the secondary windings. An electronic circuit measures the differential signal across the two secondary coils. This usually takes the form of a difference amplifier with a suitable passive filter at the input.

Figure 2 shows the electrical connections to the LVDT.

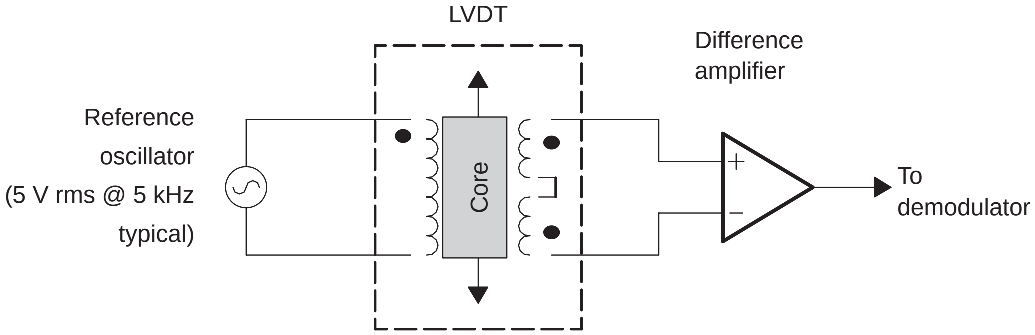


Figure 2. Electrical Connections to Transducer

With the core in its central position (shown above), magnetic flux is coupled equally onto both secondary windings and the output of the difference amplifier is theoretically zero. For this reason, the central core position is often referred to as the “null point” of the transducer.

If the core is displaced a small amount from the null point, flux coupling to one secondary winding increases while that to the other winding falls, and a net differential voltage is seen at the output of the difference amplifier. The resulting signal is a sinusoidal waveform of the same frequency as the input oscillator, and has an amplitude proportional to the displacement of the core from the null point. The signal is either fully in phase, or fully in anti-phase with the signal applied to the primary, depending on the direction of core displacement. Signal conditioning thus becomes a matter of determining the amplitude of the return signal from the secondary windings and its phase relative to the reference oscillator.

Figure 3 shows the primary signal in diagrammatic form, and the secondary signal with the core displaced in either direction from the null point.

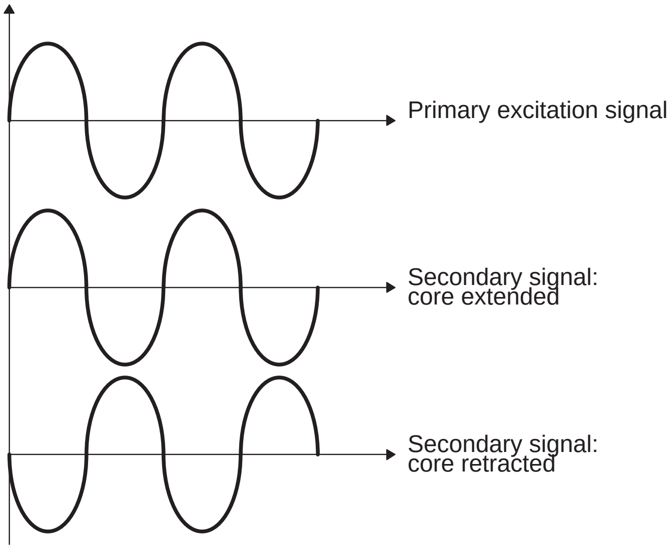

Primary excitation signal
Secondary signal: core extended
Secondary signal: core retracted

Figure 3. Primary to Secondary Phase Relationship

Since the electrical gain of the transducer depends on the turns ratio between primary and secondary windings, LVDTs can be constructed with high sensitivity and it is frequently unnecessary to amplify the return signal. This makes the transducer particularly suitable for use in electrically noisy environments. Other features of the transducer are zero hysteresis, and resolution which is theoretically limited only by the design of the signal conditioning electronics. Furthermore, since the core is not in contact with the transducer body, movement is frictionless and the mechanical life of the transducer is essentially infinite.

3 Analog Signal Conditioning Techniques

3.1 Description of a Typical Analog Circuit

Most LVDT circuits in use today employ an analog sine wave oscillator, such as the well-known “Wien bridge” type, to produce a fixed frequency excitation signal for the primary coil. Since the return signal is amplitude-modulated, the system is sensitive to any instability in the reference oscillator, which is amplified by the electrical gain of the transducer. Therefore, the oscillator circuit usually incorporates some form of gain control to stabilize the amplitude of the output signal.

The return signals from the two secondary windings are usually connected to an instrumentation amplifier (IA) which measures the differential voltage. This is filtered to remove unwanted noise by a band-pass filter tuned to extract the oscillator frequency. The filter output is passed through a synchronous demodulator circuit which yields a full-wave rectified signal, the polarity of which depends on the phase of the return signal relative to the oscillator. The signal is then smoothed and scaled to produce a stable DC voltage proportional to core position. Figure 4 shows a block diagram of the oscillator and signal conditioning electronics.

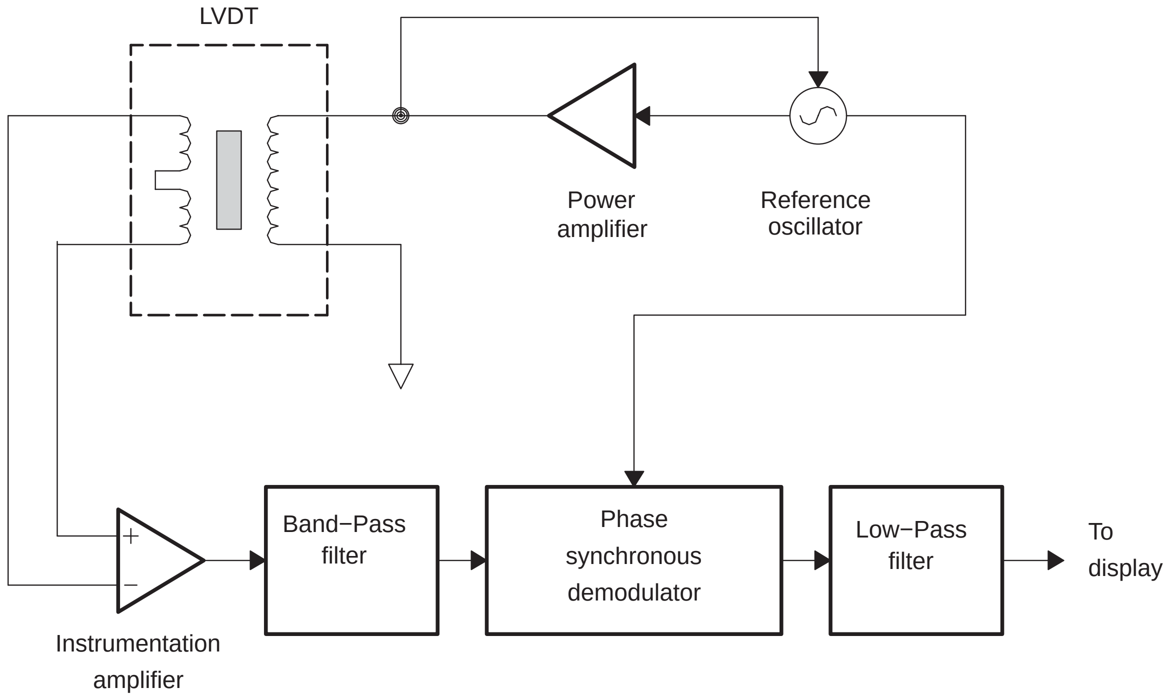


Figure 4. Analog Signal Conditioning Block Diagram

3.2 Limitations of the Analog Approach

3.2.1 Residual Null Point Signal

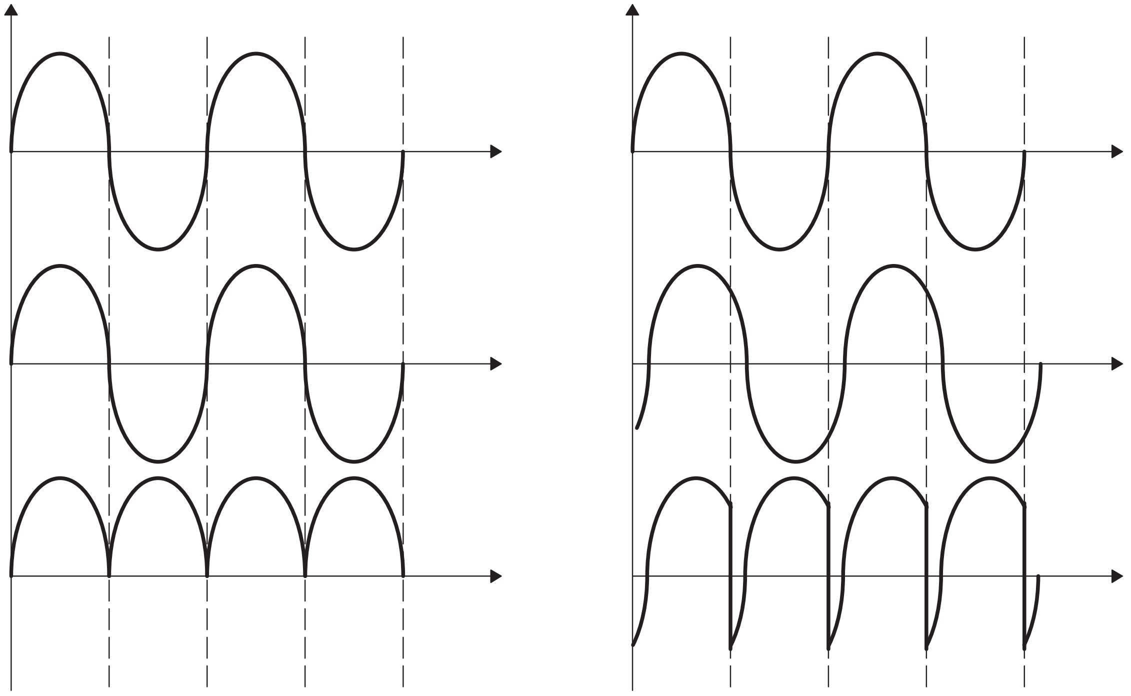
In theory, with the core at the null point, magnetic flux is coupled equally to both secondary windings and the net difference seen at the output of the IA is zero. In practice, however, a small amount of residual signal is usually observed which arises from unbalance in the windings and stray internal capacitance, as well as leakage resistance effects. A residual null signal may consist of the fundamental reference frequency as well as higher harmonics. The fundamental frequency produces a small offset at the null point and can be removed in the signal conditioning electronics, while higher harmonics can often be removed by a suitable filter. Nevertheless, most LVDT transducers do exhibit a certain amount of null residual signal which requires additional compensating circuitry, and can introduce a non-linear error into the measurement near the null point. This can be particularly inconvenient, since in many systems the central core position is just where accuracy is most important.

3.2.2 Phase Shift

The demodulation process is essentially one of multiplying the return signal with the reference oscillator being fed to the primary. If both signals are in-phase, the result is of the form $sin^2\theta$: a full-wave rectified sine wave which can be smoothed with a suitable low-pass filter to give a stable voltage linearly proportional to core position. The technique works well in the absence of phase shift between reference and return signals. However, if spurious phase shift is present in the signal chain, the multiplication process yields sum and difference terms which combine to give ‘undershoots’ in the rectified signal. These are integrated into the result during the smoothing process and contribute to linearity error in the final measurement.

Figure 5 shows diagrammatically the effect of adding a small amount of phase shift to the return signal with the core at some arbitrary displacement from the null point. The upper trace in each case is the excitation signal to the primary winding, the centre trace is the return signal (output of the difference amplifier), and the lower trace is the demodulated signal.

Figure 5(a) shows the ideal case in which there is no phase shift between excitation and return signals. In Figure 5(b), a small amount of phase shift has been added. The zero crossing points of the return signal no longer align with those of the reference signal, and negative “spikes” are seen in the rectified trace at the demodulator output which leads to non-linear measurement error. In a practical system, even quite small amounts of additional phase shift can produce significant measurement error.



a) Core extended − no phase shift

b) Core extended − phase shift in return signal

Figure 5. Effect of Phase Shift on Demodulated Signal

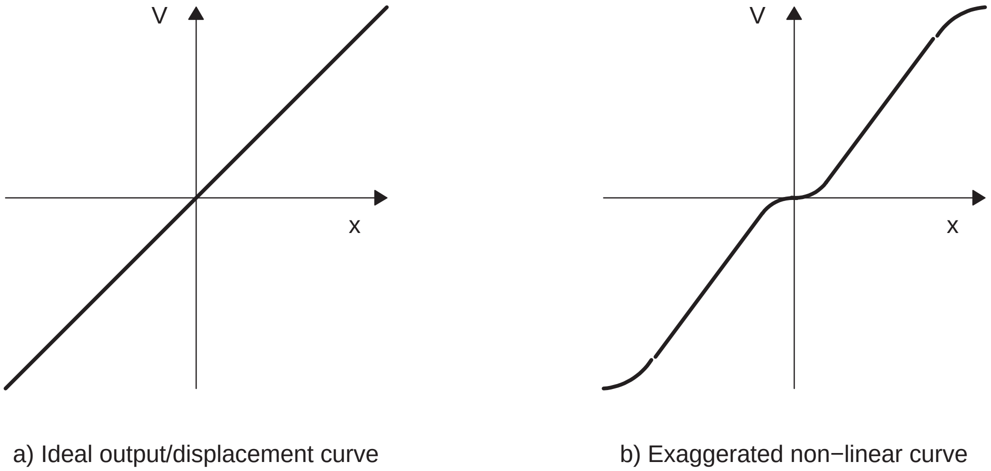
In practice, some additional phase shift is almost certain to exist in the return signal due to the effect of stray capacitance in the transducer and connecting leads, as well as from tolerance and drift effects associated with analog components in the signal path.

3.2.3 Transducer Non-Linearity

The transducer itself can exhibit a certain amount of non-linearity due to its construction, and although this is not directly related to the design of the signal conditioning circuit, its effect cannot be neglected from the total measurement error.

As the core displacement approaches the limit of its movement in either direction, the magnetic flux coupling to the secondaries becomes increasingly non-linear. This arises partly because the core starts to emerge from the end of the transducer body, and the effect on the transducer output is to diminish the signal amplitude near the limits of displacement. To some extent this can be corrected by good transducer design, however some degree of signal 'roll-off' can be expected, and unless corrected can limit the effective measurement range.

Figure 6 shows the result of the above effects on the output vs. displacement curve. The graphs show output voltage (V) against core displacement (x). Figure 6(a) shows the ideal case in which output is linearly proportional to input. Figure 6(b) shows the exaggerated non-linear case for a real transducer. Roll-off near the extremes of core displacement arises from transducer construction and phase effects, while non-linearity near the zero crossing is the effect of residual signal at the null point of the transducer. In a purely analog system, both effects are very difficult to correct.



a) Ideal output/displacement curve

b) Exaggerated non−linear curve

Figure 6. Output / Displacement Curves

3.2.4 Component Tolerance, Aging and Drift

Errors due to component tolerance, aging and drift are always present in any practical analog circuit, and it is left to the skill of the circuit designer to ensure their combined worst case effect does not exceed an acceptable limit for the system.

4 Design Using a Digital Signal Processor

The use of a digital signal processor offers the designer the ability to compensate for, or to completely avoid, the limitations outlined in the last section. The effects of residual null signal and transducer non-linearity can be compensated by digital error correction, while the use of digital demodulation techniques make the system insensitive to spurious phase shifts up to ±90 degrees.

A software correction algorithm is easily able to compensate for static errors in the output measurement by means of a software correction map. The software determines a correction offset by interpolation between the nearest points in the map and applies it to the measurement – a simple process taking only a few CPU cycles. The correction map is stored in internal non-volatile memory, and can be programmed during routine calibration of the transducer.

The DSP selected for this experiment is capable both of generating a highly stable reference signal for the primary coil, and of conditioning the returned signal. The processor is a member of the C28x family of devices from Texas Instruments. These are 32-bit fixed-point DSPs which have a rich selection of integrated peripherals including PWM generators, various serial communication interfaces, and a 12-bit A/D converter, making them ideal for applications with challenging cost or space constraints. The particular device chosen for this application runs at a maximum CPU frequency of 150 MHz.

4.1 Hardware Description

The following description deals specifically with a circuit used for experimental purposes in the laboratory. It is acknowledged that the design can be improved in various ways to meet the stringent cost and performance requirements of a production system.

An eZdsp™ DSP development board from Spectrum Digital (part no. 761128) was used in the experiment. This is available as a development kit comprising DSP board, DC power supply, on-board JTAG-compliant emulator, and the Code Composer Studio™ suite of development tools (including debugger IDE and ANSI-C compiler). The board has all peripheral signals from the DSP connected to unpopulated header connectors, making it easy to interface the DSP to custom hardware.

The development board was mounted on a prototyping card from DSP Global (part no. DSPG-LITE-2407), on which was constructed the analog interface circuitry to the transducer. A small daughtercard was constructed to carry the power components, and this was mounted above a free area of the prototype board. Removable connectors were fitted to carry power and transducer connecting leads. A standard bench power supply (RS model 7748) was used to provide the ±15V power rails for the oscillator circuit and instrumentation amplifier.

Figure 7 shows a photograph of the experimental setup. A simplified schematic of the electronic interface circuit is shown in Appendix A. Power connections and some component details have been omitted for clarity.

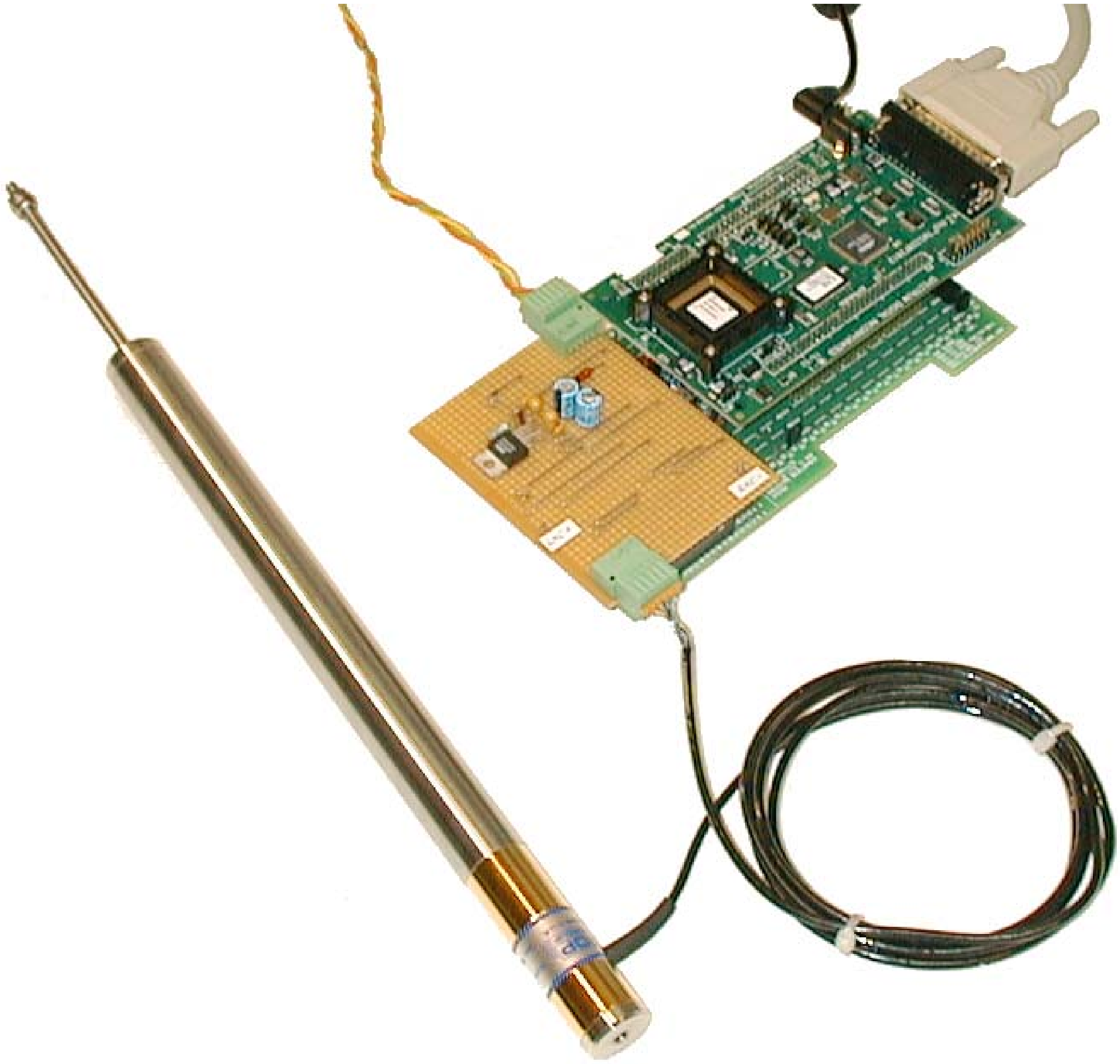


Figure 7. Photograph of Experimental Rig

4.1.1 Reference Oscillator

The TMS320F2812 DSP used in this implementation is equipped with two sophisticated integrated PWM pattern generators called event managers, each of which is capable of producing multiple PWM pulse trains of programmable frequency and duty cycle. Such pulse trains can be smoothed with a simple low-pass filter to produce an analog voltage which is proportional to the PWM duty cycle (see ref. #4). The transducer selected for this application has its minimum internal phase shift at a specified excitation frequency of 5 KHz, which is the chosen frequency of modulation.

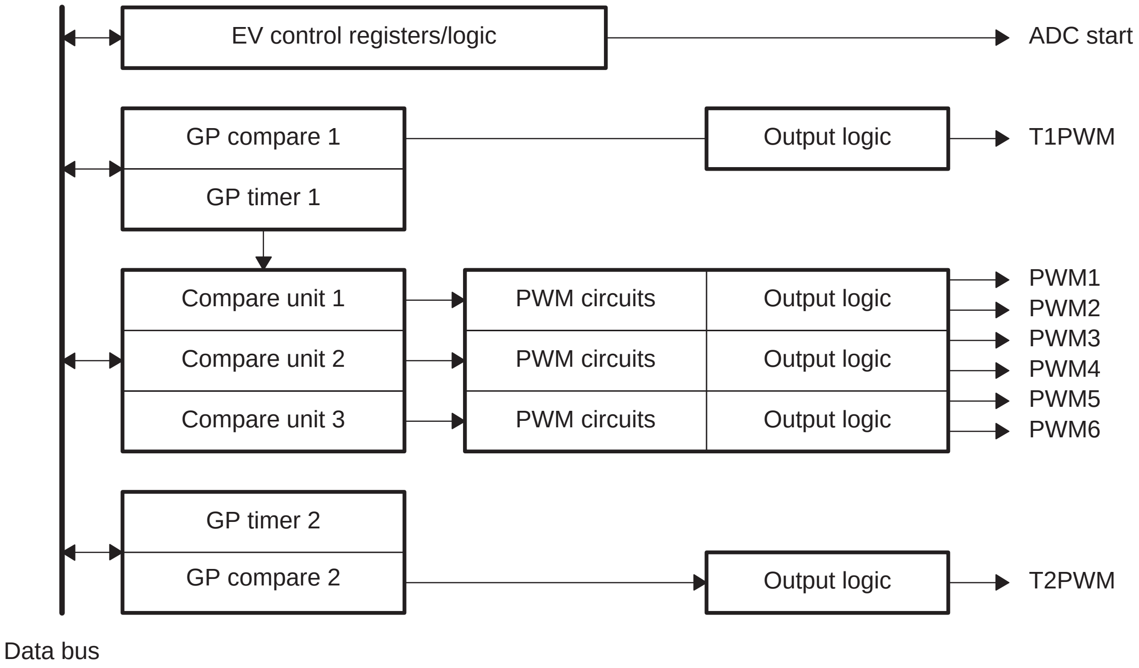


Figure 8. Simplified Block Diagram of Event Manager

A simplified diagram of one C28x event manager (EVM) is shown above. Some digital input features have been omitted for clarity. The EVM contains two free-running, programmable 16-bit timers, each connected to a digital comparator which produces a logic output according to whether the instantaneous timer count is less than or greater than the compare threshold. The timer period and compare registers are “shadowed”, meaning they can be modified without disrupting the PWM generator, the new value being updated automatically on the next pulse transition. The EVM can therefore produce two PWM outputs with independently programmable frequency and duty cycle (T1PWM and T2PWM).

Timer 1 is used as the clock source for three additional digital compare units, each capable of producing complementary output PWM streams, which have a common time base but independent duty cycle. These have more complex output logic allowing the production of complementary pair PWM signals with programmable “dead-band” for driving power devices. These outputs are not used in this application.

In this application, GP Timer 1 is configured to deliver the fixed frequency 800 KHz PWM train for the oscillator. The duty cycle of this is modulated by an interrupt service routine triggered by GP Timer 2 running at 160 KHz, which retrieves pre-calculated sine wave data points from a fixed look-up table and writes it into the duty cycle (compare) register of GP Timer 1. The resulting PWM output signal (T1PWM) has a fixed frequency of 800 KHz, and a duty cycle which is sinusoidally modulated at 5 KHz.

The 5 KHz reference sine wave is recovered by smoothing the PWM signal with a suitable low-pass analog filter. The filter used in this experiment is of the well known ‘Sallen & Key’ type, designed to have a second order low-pass characteristic and a cut-off frequency of 19.4 KHz.

4.1.2 Power Amplifier

The primary winding of the chosen transducer has a nominal impedance of 91 Ω at the oscillator frequency, and a power amplifier is necessary to provide sufficient current to drive the primary coil. An OPA544T power amplifier was used for this purpose, and the output signal AC coupled using two electrolytic capacitors to block any DC offset which would have a low impedance path to ground through the transducer coil. The output signals are bipolar, and a separate power supply is required to drive the power amplifier stage. In this case a ±15V bench power supply is used.

4.1.3 Instrumentation Amplifier

The return signals from the two secondary windings are connected to an instrumentation amplifier (IA). Texas Instruments part INA111AP was selected for this purpose on the basis of its high CMRR and high speed.

Since the LVDT transducer has electrical gain, a potential divider is often necessary to attenuate the return signal and avoid saturating the amplifier inputs. The LVDT is designed to operate with an optimum load impedance and it is necessary to consider this when selecting the attenuation resistors. In this case attenuation was found to be unnecessary, and a total resistance of approximately 100 KΩ was connected across the inputs to the board to match the optimum loading of the transducer. Operation with loads much higher or lower than the nominal value can contribute to non-linear measurement error.

Differential and common mode filtering is applied at the inputs to the IA by a simple filter circuit consisting of passive components. A differential filter with a roll-off at 36.1 KHz was used. Common mode filtering was applied at the much higher frequency of 330 KHz to avoid compromising the CMRR of the IA.

4.1.4 Anti-Aliasing Filter

Before being passed to the integrated A/D converter, it is necessary to filter the return signal to remove frequencies which might be aliased by the sampling process. The selected sample rate of 160 KSPS means that frequencies of 80 kHz and above must be effectively removed from the return signal prior to sampling by some kind of analog filter. Inevitably the use of a filter at this point introduces phase shift into the return signal path, and in a conventional analog design some means of phase compensation would need to be applied elsewhere in the signal chain to avoid corrupting the demodulator (see Section 3.2.2). However, the use of digital processing techniques permits the demodulator to be made insensitive to moderate phase shifts in the return signal, and in this example the phase shift imparted by the anti-aliasing filter was deliberately left uncorrected.

The anti-aliasing filter is based on a ‘Sallen & Key’ design, tuned to provide a 2nd order low-pass roll-off at 12.835 KHz. This introduced a phase shift of approximately 33 degrees at the oscillator frequency; an amount which would probably introduce unacceptable distortion in a conventional analog demodulator, but which the digital technique described later handles easily. The output of the filter is connected to analog channel 0 of the integrated A/D converter of the DSP.

4.2 Software Description

The 5 KHz reference oscillator signal is generated using a software look-up table with an arbitrary 8 points per quadrant, giving a required PWM update rate of 160 KHz. As described earlier, GP Timer 1 is programmed to produce the PWM time base at 800 KHz frequency, and the duty cycle is modulated by an ISR triggered from GP Timer 2 running at 160 KHz. This gave a relatively high sample rate compared with the oscillator frequency and relaxed the constraints on the anti-aliasing filter.

At this frequency, a C28x running at the maximum CPU clock speed of 150 MHz executes 937 cycles between consecutive interrupts, and all the conditioning code has to be completed in this time. With many common micro-processors and DSPs, the software engineer would be obliged to write the time-critical portions of his code in assembly language in order to meet the real-time application deadlines. However, the C28x has a highly efficient C compiler and a powerful virtual floating-point engine (IQ Math), which allowed practically all the source code to be written in C. The IQ Math library can be freely downloaded from the Texas Instruments website, and is well described in the documentation and example programs which are included therein.

The built-in profiler feature of Code Composer Studio allowed accurate measurement of ISR execution time and confirmed that the execution speed of the final program was sufficiently fast to meet the real-time requirements of the application.

The TMS320F2812 has 128 KW of integrated FLASH memory for non-volatile storage of program and data sections. Internal flash supports operation at up to 120 MIPS, so any time critical portions of code must be copied into internal RAM prior to execution in order to run at the maximum CPU speed of 150 MIPS. In the final production program, an initialization routine for each time-critical ISR is required to copy the corresponding code across from internal ROM to RAM. During the debug phase, when the code is downloaded directly from the host into internal RAM, no such initialization is required.

Figure 9 shows a block diagram of the software design. It is noted that this is similar in form to the analog circuit block diagram of Figure 4. The time-critical signal conditioning code and the sine wave generator are combined in a single interrupt service routine written in C.

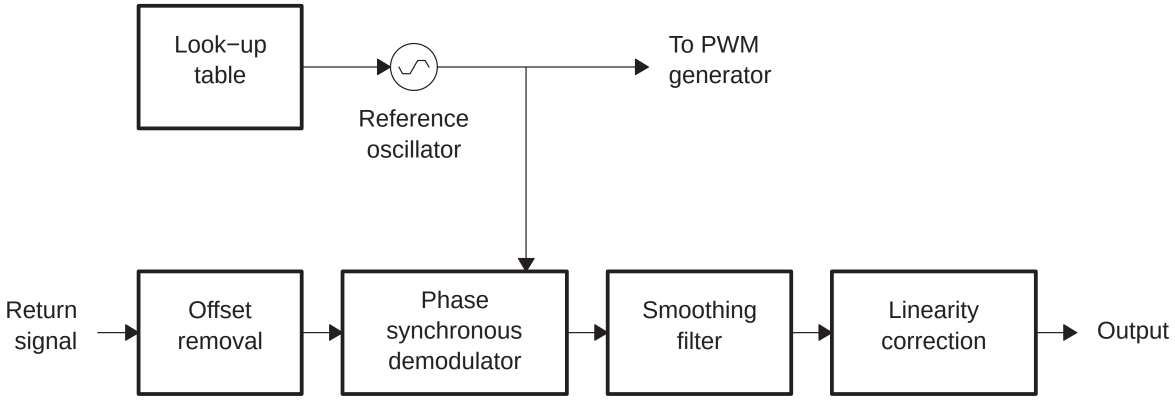


Figure 9. Software Block Diagram

As described earlier, the conditioning interrupt routine is triggered by a period event from GP Timer 2. The period event is also used to sample the analog return signal from the transducer and begin an A/D conversion: on the C28x it is possible to initiate an A/D conversion automatically in hardware on any PWM event, allowing analog input signals to be sampled at regular intervals of time and without being affected by interrupt jitter or software latency. On completion, the ADC result is stored in a memory-mapped register where it can be retrieved by the next ISR. The A/D conversion time for a single channel is about 200 ns (approx 30 instructions at 150 MHz) so the ADC easily keeps pace with the conditioning interrupt.

Next, the ISR executes a short algorithm to remove any DC offset from the signal prior to demodulation. The signal is then passed through a demodulator algorithm which determines the magnitude and sign of the return signal, and is smoothed with an averaging filter to produce a stable signal proportional to the position of the transducer core. The measured position is then modified with offset and gain coefficients, and a linearity correction applied based on values held in a look-up table.

For the purposes of this experiment, the corrected result was logged into a buffer in internal data memory for analysis using the IDE. In practical systems, the measurement could be transmitted over a high speed synchronous serial link to remote device, or sent to an external DAC in cases where the measurement output is required in analog form.

The conditioning code runs at the same rate as the sine wave generator. This enables the bandwidth of the measuring system to be kept high, which can be important if the transducer is to be used as a feedback element in a control loop. The timeline for ISR execution is shown in Figure 10.

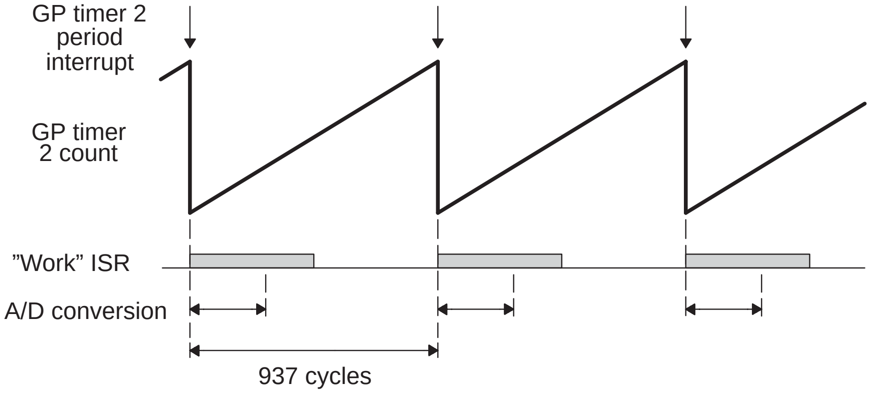


Figure 10. Interrupt Timeline

A second interrupt (background_isr) runs asynchronously at an arbitrary rate of about 2 Hz. Its task in this example is to perform simple housekeeping tasks such as re-triggering to the watchdog timer and flashing an LED.

A brief description of the code follows. Readers who are familiar with the C programming language may find it helpful to refer to the code listing in Appendix B while reading the following description.

4.2.1 Software Initialization

The main part of the program contains two parts: a short initialization section which sets up peripherals and memory, and an infinite loop containing a single instruction to increment a global counter. The initialization code is executed before any system interrupts are enabled and is therefore not time-critical. At the end of this section, the program sets up the required interrupt masks and enables the watchdog timer. Thereafter, the program enters an infinite idle loop and is driven entirely by asynchronous hardware interrupts.

4.2.2 Signal Conditioning ISR

The ISR reads fresh data from the analog to digital converter which samples channel 0 of the input multiplexer. A new A/D sequence is initiated automatically by the timer period event, so the ISR always reads the conversion result of the previous sample. Since the sine look-up table is composed of eight points per quadrant, this imparts an 11.25 deg. phase shift to the conditioned signal, but since the digital demodulation technique used is insensitive to small amounts of phase shift this does not present a problem.

The C28x has a highly efficient interrupt management system in which critical core registers are protected by an automatic hardware context save to the internal stack. Any additional context except for the specific interrupt service routine is handled by the C compiler. The user entry code manipulates the peripheral interrupt acknowledge and flag registers to enable subsequent interrupts; however, note that the global interrupt mask is set for the duration of the interrupt routine, ensuring this ISR has highest priority (it cannot be nested).

Data logging capability was built into the conditioning ISR to permit individual variables to be monitored in real-time during the debug phase. Each stage in the signal conditioning chain is now described in turn.

4.2.2.1 Sine Wave Generator

A data array holds a complete sine wave cycle in 32 integer data points which is loaded into a predefined memory block. A structure of three pointers is used to map the beginning and end of the lookup table, and store the next data point address. The data pointer is incremented on each ISR and on reaching the end of the buffer is reset to the first buffer address. In this way, each execution of the work ISR produces a new data point. A short initialisation routine is required to set up the buffer pointers and other variables.

Since the demodulation technique requires access to the quadrature (cosine) signal, the table actually consists of five quadrants (40 points total). The quadrature data is simply computed using an 8-point offset from the sine pointer.

The ISR computes the next PWM duty cycle based on the data in the lookup table and writes it to the CMPR1 compare register of event manager A, which modifies the duty cycle of the free running PWM output signal. Since the lookup table holds signed data, it is necessary to scale and offset each data point prior to modifying the 16-bit PWM compare register, which accepts an unsigned integer normalised to the PWM period.

4.2.2.2 Offset Removal Algorithm

The offset removal algorithm removes any DC offset component which may be present on the input signal and would disrupt correct operation of the demodulator. It works by maintaining a running average of the input signal and subtracting the result from each input reading. Since the average value of a sine wave is zero over a complete number of cycles, only the DC offset component remains after averaging and this can be subtracted from the ADC reading on a point-by-point basis, effectively AC coupling the input signal.

4.2.2.3 Phase Synchronous Demodulator

The second stage of the conditioning chain comprises a demodulator, which is synchronised to the reference oscillator to determine the phase shift of the return signal and yield a full-wave rectified sine wave of the appropriate polarity (see Section 3.2.2).

The use of a digital technique to produce the reference oscillator means that the quadrature signal is readily available, and this permits a simple demodulation technique to be used which is insensitive to phase shift. The return signal can be thought of as a complex vector, the real and imaginary parts of which can be computed independently by multiplying with the two quadrature components. The magnitude of the vector is computed from the root of the sum of the squares of these components.

The demodulator code is written in C using the IQ Math library, which produces highly efficient C28x assembly code while allowing the user to select a numeric format which provides sufficient range and resolution to meet the application needs. In this case, $I_{12}Q_{20}$ format was found to yield satisfactory results.

The determination of phase was based on the result of an ‘exclusive OR’ between the sign of the reference oscillator and that of the return signal: if these signals are in phase the signs should be the same and the result is false, if the signals are in anti-phase the signs are opposed and the result will be true. Phase was determined using a ‘majority vote’ based on the previous 32 results (i.e., over one complete oscillator cycle). This technique was found to give good results near the null point, where the signal-to-noise ratio is low.

4.2.2.4 Smoothing Filter

The third stage of the signal conditioning ISR is the smoothing filter, which converts the rectified signal into a smooth and stable digital result. A simple moving average filter was constructed using a window large enough to cover the previous 16 half-cycles of demodulated signal. The filter runs on a point-by-point basis: each data point from the output of the demodulator is added to a circular buffer, and the average computed by subtracting the oldest value from a running total, adding the newest value, and scaling.

4.2.2.5 Measurement Error Correction

The final stage of the conditioning ISR consists of measurement correction. Fixed offset and gain adjustments are applied, and the result is used to index into a correction map to retrieve a correction in bits which is added to the reading. The indexing technique uses simple linear interpolation to compute a correction factor from the two nearest points in the map.

4.2.3 Background ISR

A short background ISR runs asynchronously from an internal 32-bit CPU timer. It toggles an LED on the target board as a visual indication that the application is running, increments a global counter, and writes to the watchdog key register to re-trigger the timer. Since these are low priority tasks, the background ISR may be nested; meaning it can be interrupted by higher priority interrupts. The global interrupt enable mask is protected by inline functions at the start and end of the ISR which save and restore the current value of the interrupt mask register using a software stack, allowing it to be locally modified without disrupting the program.

4.3 Performance

4.3.1 Phase Correction

Most conventional switching demodulators are sensitive to small amounts of spurious phase shift in the return signal. Figure 11 shows the output of a switching demodulator used with an LVDT when the return signal carries about 30 degrees of phase shift. The upper trace is the reference oscillator, the lower the demodulator output. The core is displaced outwards (extended) a short distance from the null position.

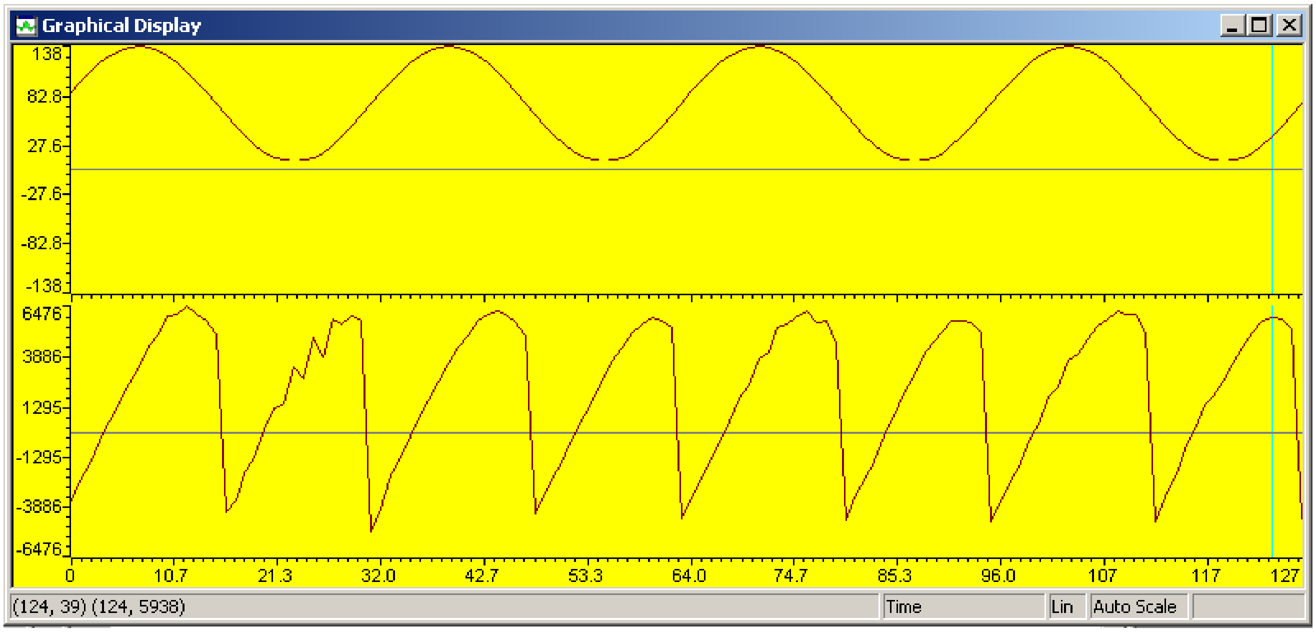


Figure 11. Output From Typical Switching Demodulator

There is clearly a significant phase shift between primary and return signals (compare the positions of the peaks of the two traces). The presence of this causes the polarity inverter to switch before the zero crossing point of the return signal, and induces prominent negative spikes in the demodulated signal. These are integrated into the measurement by the smoothing filter which follows the demodulator, and lead to distortion of the overall input/output curve (Fig. 6b).

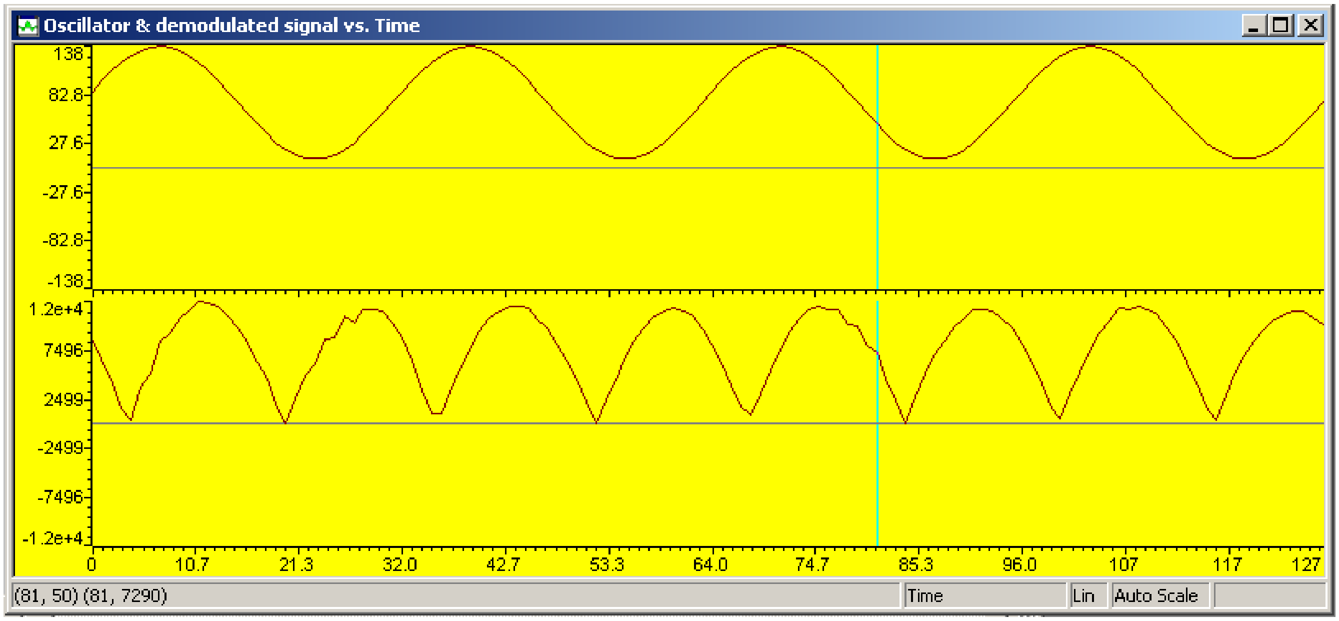


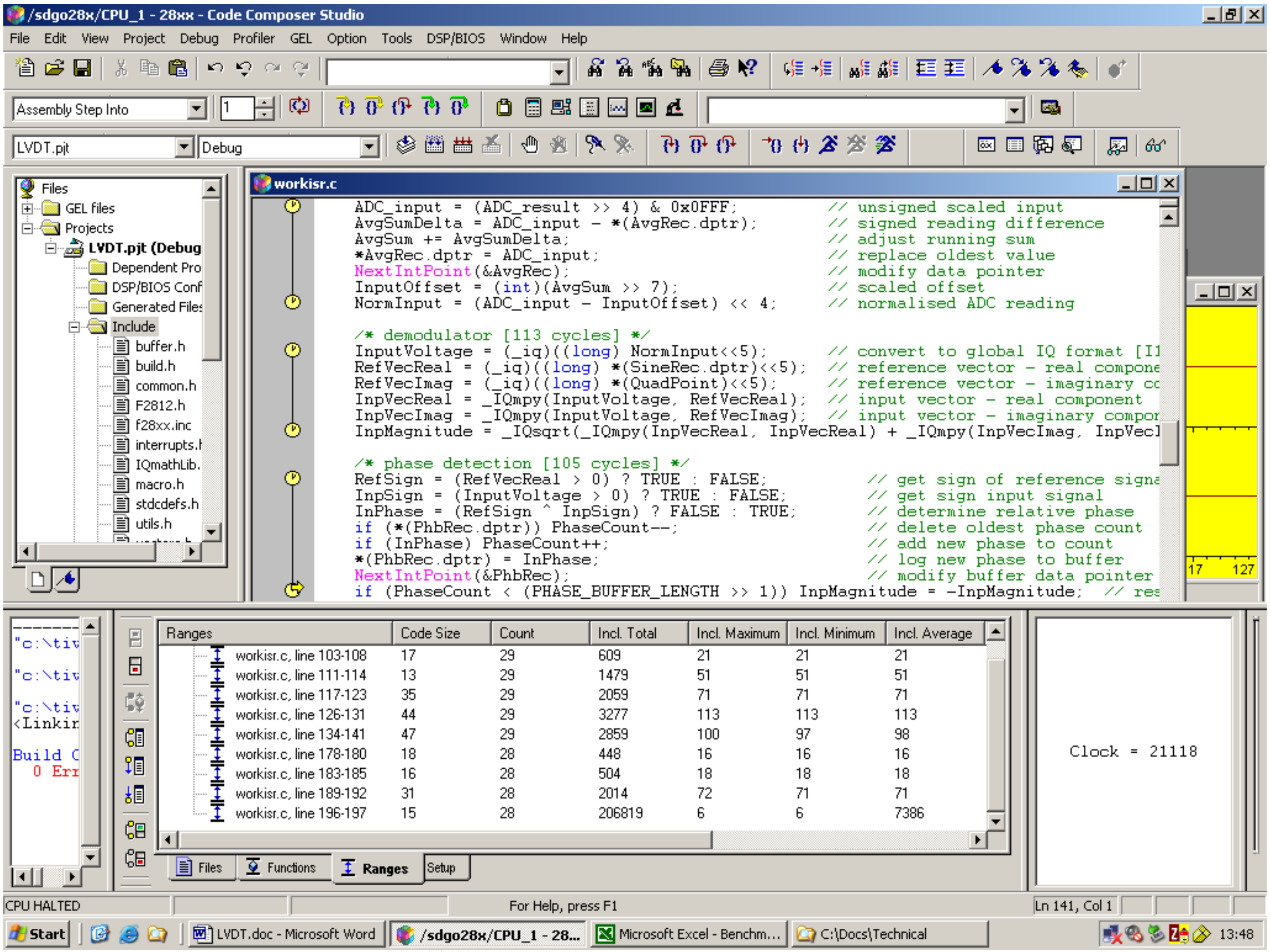
Figure 12. Output From Digital Demodulator

In the second case, the demodulated signal exhibits no negative spikes, even though a similar amount of phase shift is present. This is due to digital demodulation technique used (described in 4.2.2.3). It is interesting to compare the waveforms in Figure 11 and Figure 12 with those shown diagrammatically in Figure 5.

Note: The above traces were prepared using the realtime debug feature of the C28x processor. Data was logged to two buffers held in internal data memory, and a graph window opened in Code Composer Studio to view the contents. Real-time debug mode was enabled in Code Composer Studio and the graph windows set to refresh continuously. The effect is similar to connecting an oscilloscope probe directly to the software variable being logged, and is an extremely useful feature when debugging a system in which dynamic signals are treated in software.

4.3.2 Benchmark Timings

Code Composer Studio incorporates several advanced features which facilitate rapid and accurate benchmarking of code. The screenshot below shows the “profiler”, which produces statistical information on the execution of selected blocks of code.



```
ADC_input = (ADC_result >> 4) & 0x0FFF;            // unsigned scaled input
AvgSumDelta = ADC_input - *(AvgRec.dptr);          // signed reading difference
AvgSum += AvgSumDelta;                             // adjust running sum
*AvgRec.dptr = ADC_input;                          // replace oldest value
NextIntPoint(&AvgRec);                             // modify data pointer
InputOffset = (int)(AvgSum >> 7);                  // scaled offset
NormInput = (ADC_input - InputOffset) << 4;        // normalised ADC reading

/* demodulator [113 cycles] */
InputVoltage = (_iq)((long) NormInput<<5);         // convert to global IQ format [I1
RefVecReal = (_iq)((long) *(SineRec.dptr)<<5);     // reference vector - real compone
RefVecImag = (_iq)((long) *(QuadPoint)<<5);        // reference vector - imaginary cc
InpVecReal = _IQmpy(InputVoltage, RefVecReal);     // input vector - real component
InpVecImag = _IQmpy(InputVoltage, RefVecImag);     // input vector - imaginary compor
InpMagnitude = _IQsqrt(_IQmpy(InpVecReal, InpVecReal) + _IQmpy(InpVecImag, InpVecI

/* phase detection [105 cycles] */
RefSign = (RefVecReal > 0) ? TRUE : FALSE;             // get sign of reference signa
InpSign = (InputVoltage > 0) ? TRUE : FALSE;           // get sign input signal
InPhase = (RefSign ^ InpSign) ? FALSE : TRUE;          // determine relative phase
if (*(PhbRec.dptr)) PhaseCount--;                      // delete oldest phase count
if (InPhase) PhaseCount++;                             // add new phase to count
*(PhbRec.dptr) = InPhase;                              // log new phase to buffer
NextIntPoint(&PhbRec);                                 // modify buffer data pointer
if (PhaseCount < (PHASE_BUFFER_LENGTH >> 1)) InpMagnitude = -InpMagnitude;  // res
```

```
-------
"c:\tiv

"c:\tiv

"c:\tiv
<Linkir

Build C
  0 Err
```

Ranges	Code Size	Count	Incl. Total	Incl. Maximum	Incl. Minimum	Incl. Average
workisr.c, line 103-108	17	29	609	21	21	21
workisr.c, line 111-114	13	29	1479	51	51	51
workisr.c, line 117-123	35	29	2059	71	71	71
workisr.c, line 126-131	44	29	3277	113	113	113
workisr.c, line 134-141	47	29	2859	100	97	98
workisr.c, line 178-180	18	28	448	16	16	16
workisr.c, line 183-185	16	28	504	18	18	18
workisr.c, line 189-192	31	28	2014	72	71	71
workisr.c, line 196-197	15	28	206819	6	6	7386

```
Clock = 21118
```

Figure 13. Screenshot of Code Composer Studio profiler

Code Composer Studio for C28x includes a cycle accurate execution clock, as well as a non-intrusive, real-time debug capability. In conjunction with the profiler shown above, these features make accurate benchmarking of code execution very straightforward.

The following table shows the code size and execution timings for each block of code in the conditioning ISR module. The total execution time is 464 CPU cycles, which lies well within the maximum 937 cycles available between consecutive interrupts. In other words, the work ISR consumes less than 50% of the available CPU bandwidth.

The sole contribution to interrupt jitter comes from the background ISR which disables global interrupts for short intervals of time while saving and restoring the context. The longest period during which interrupts are disabled occurs on entry to the background ISR and is 28 CPU clock cycles (about 187 ns at 150 MHz), which is the worst case interrupt jitter. This corresponds to less than 3% variation in the interrupt period. Since the sine wave generator uses a PWM technique with shadowed compare registers as described earlier, and the base PWM period is five times that of the modulation interrupt, this small variation is not expected to affect the oscillator performance.

Table 1. Conditioning Code Benchmarks

Code Block	Code Size (Words)	Cycle Count
ISR entry routine	17	21
Sine wave generator	13	51
Offset removal	35	71
Demodulator	44	113
Phase detector	47	97
Smoothing filter	18	16
Fixed offset and gain corrections	16	18
Linearity correction	31	71
Reset A/D converter	15	6
Total	236	464

The total program size can be determined by inspecting the “map” text file produced by the linker. In this case the program was found to occupy only 1,517 words of the total available flash memory of 128 KW, without invoking the C compiler optimizer.

Since the objective of this paper is simply to demonstrate the concept of LVDT signal conditioning using DSP, a detailed analysis of performance was not carried out except in the areas of immunity to spurious phase shift and code benchmark timings. The output of the program is simply a software variable which can be examined in a watch window within the debug environment. Adjustment of the software gain and offset factors described above indicated that a result with resolution of 12 bits and less than 1 LSB noise was easily achievable on the experimental rig. It is felt that a more refined hardware interface circuit in which careful attention was paid to the layout of the signal connections to the A/D converter would achieve much superior performance for the expense of very little design effort.

5 Discussion

The use of digital signal processing techniques affords the designer several advantages over the conventional analog approach. Foremost among these are the ability to desensitize the system to the presence of phase shift in the return signal, and to compensate for various measurement errors through the use of programmable software correction.

The correction map is programmed into internal non-volatile memory during the normal calibration process, and can be as simple or complex as the application requires. Measurement accuracy can be improved by increasing the number of points in the correction map at the expense of longer time required for transducer calibration. In addition to the correction map, stored data can include correction coefficients for transducer gain and offset.

LVDTs are sensitive to changes in operating temperature: a rise in temperature of the primary coil has the effect of increasing the electrical resistance of the winding and reducing the primary current. The amplitude of the output signal for constant excitation falls correspondingly, and the electrical gain of the transducer is diminished. This effect could be compensated in software by monitoring the temperature and applying a correction to the digital output reading. Since transducer sensitivity has a non-linear dependency on temperature, the adjustment could take the form of first and second order correction curves, the correction data being stored either in the form of a map, or as polynomial coefficients.

The LVDT transducer is known to be dependent on a stable excitation signal for accurate measurement, and many oscillator designs include some form of gain control circuit to maintain the excitation signal amplitude at a constant level. Although the simple example described here did not implement this feature, it would be a straightforward matter to monitor the oscillator output amplitude via an unused ADC channel and affect a correction inside the conditioning ISR, a few extra CPU cycles being required.

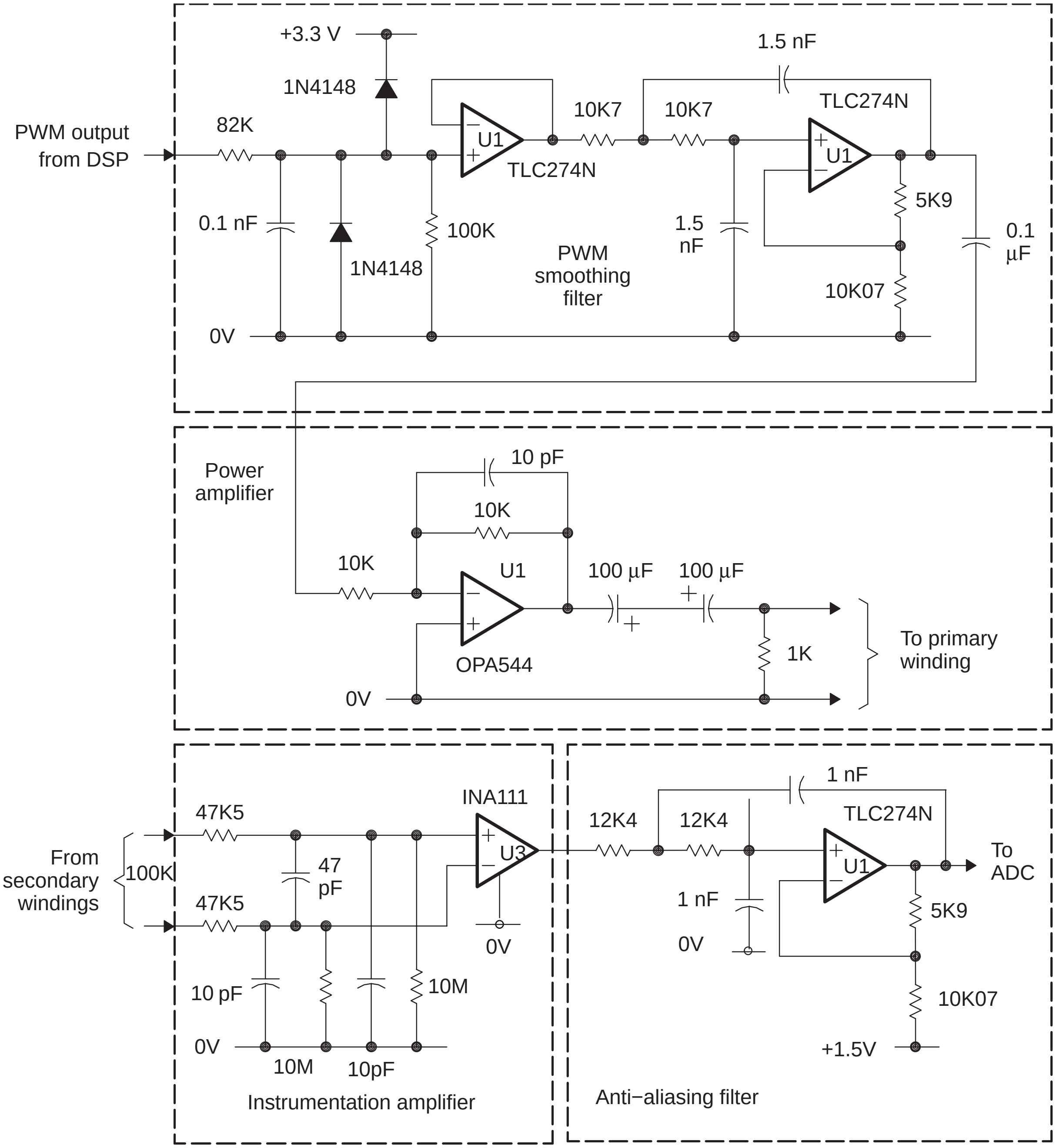
The performance benchmarks, in particular CPU usage and code size, imply that the application could be easily ported to a lower cost DSP, possibly running at a reduced clock speed and with less internal memory. With the increasing availability of high performance DSPs in small packages, it may ultimately be possible to integrate the digital processor and associated circuits physically within the casing of the transducer. Such a system could be effectively self-contained, with calibration and other specific data stored locally in non-volatile memory, allowing faulty or damaged units to be replaced without having to undertake a lengthy calibration process. In applications where system down time is critical, such as the monitoring of process variables in a factory production line, this is an important consideration.

The availability of the final measurement in digital form means that results can be transmitted directly over a noise immune serial communications link to a remote display or analysis unit. It is worth noting that several digital interface standards are beginning to emerge in the transducer industry (e.g., IEEE 1451.2) which may lead to more widespread use of digital processors for transducer signal conditioning. Indeed, it is probable that the approach presented here could be extended to other types of transducer.

6 References

1. Download C28x IQMath Library – A Virtual Floating Point Engine (SPRC087)
2. Herceg, Edward E. Handbook of Measurement and Control. Schaevitz Engineering
3. Spectrum Digital eZdsp F2812 Technical Reference
4. Szczyrbak, J. & Dr. Ernest D. D. Schmidt, B. LVDT Signal conditioning Techniques. Lucas Varity
5. *TMS320F2810, TMS320F2812 Digital Signal Processors* (SPRS174)
6. *Using PWM Output as a Digital-to-Analog Converter on a TMS320C240 DSP*. (SPRA490)
7. van Valkenberg, M. E. (1987). Analog filter design. Prentice-Hall, Inc., Englewood Cliffs, N.J. 467−479.

Appendix A Circuit Diagram



Appendix B Source Code Extract

```
// main
main()
{
     // initialisation
     InitInterrupts();                    // reset system interrupts
     InitDevice();                        // DSP core & peripherals
     LoadVectors();                       // copy PIE vector table into RAM
     InitIntNest();                       // setup interrupt nesting for background
ISR
     InitWorkISR();                       // initialise work ISR

     // start interrupts
  #ifdef REALTIME
     WriteDebugIER(REALTIME_MASK);        // setup realtime debug interrupt mask
     ERTM;                                // enable realtime debug mask (DBGM)
  #endif
     IER |= BIT12;                        // enable core interrupt 12 for TINT1
     EINT;                                // clear global interrupt mask (INTM)
     *T1CON |= BIT6;                      // start EVA timers
     Watchdog(ENABLE);                    // start watchdog timer

     // idle loop
     while (1)
     {
          IdleCount++;                    // increment loop counter
     }
}


// background ISR - triggered from CPU timer 1 on core interrupt line 13
interrupt void background_isr(void)
{
     static UINT BgCount = 0;             // ISR counter

     CLR_PM;                              // clear product mode shift
     *TIMER1TCR |= BIT15;                 // clear timer 1 interrupt flag
     IntMask(CORE_INT_MASK_2);            // enable nesting on core interrupt 2
     EINT;                                // clear global interrupt mask
     EALLOW;                              // allow access to protected registers
     WDOG_TRIG();                         // re-trigger watchdog timer
     EDIS;                                // disable access to protected registers
     BgCount++;                           // increment ISR counter
  #if (BOARD == EZDSP)
     *GPFTOGGLE = BIT14;                  // XOR bit 14 to flash LED on eZdsp board
  #endif
     DINT;                                // disable interrupts for context save
     IntUnMask();                         // remove interrupt nesting
}

// signal conditioning ISR code for LVDT - triggered by T2 period event
interrupt void work_isr(void)
{
     static UINT nDivs;                   // integral divisions in linearity map
```

```
        *CMPR1 = PWM_duty;                   // write new PWM duty cycle
        ADC_result = *ADCRESULT0;            // store ADC reading
        ADC_input = ADC_result ^ BIT15;      // bipolar input
        *PIEACK = PIEACK_GROUP3;             // acknowledge PIE group 3
        *EVAIFRB = 0x0001;                   // reset T2PINT flag

         /* sine wave generator */
        NextIntPoint((struct IBUFFER *) &SineRec);// align sine table pointer
        PWM_duty = *(SineRec.dptr);          // next sine point
        PWM_duty = (PWM_duty>>9) + 0x004B;   // scale & offset for PWM compare
        QuadPoint = SineRec.dptr + QUADRANT_LENGTH;    // cosine point for demodulator

        /* input offset removal */
        ADC_input = (ADC_result >> 4) & 0x0FFF;        // unsigned scaled input
        AvgSumDelta = ADC_input - *(AvgRec.dptr);      // signed reading difference
        AvgSum += AvgSumDelta;                         // adjust running sum
        *AvgRec.dptr = ADC_input;                      // replace oldest value
        NextIntPoint(&AvgRec);                         // modify data pointer
        InputOffset = (int)(AvgSum >> 7);              // scaled offset
        NormInput = (ADC_input - InputOffset) << 4;    // normalised ADC reading

        /* demodulator */
        InputVoltage = (_iq)((long) NormInput<<5);// convert to global IQ format
        RefVecReal = (_iq)((long) *(SineRec.dptr)<<5); // reference vector - real
        RefVecImag = (_iq)((long) *(QuadPoint)<<5);    // reference vector - imaginary
        InpVecReal = _IQmpy(InputVoltage, RefVecReal); // input vector - real
        InpVecImag = _IQmpy(InputVoltage, RefVecImag); // input vector - imaginary
        InpMagnitude = _IQsqrt(_IQmpy(InpVecReal, InpVecReal)
             + _IQmpy(InpVecImag, InpVecImag));        // calc Root-Sum-Squares

        /* phase detection */
        RefSign = (RefVecReal > 0) ? TRUE : FALSE;// get sign of reference signal
        InpSign = (InputVoltage > 0) ? TRUE : FALSE;   // get sign input signal
        InPhase = (RefSign ^ InpSign) ? FALSE : TRUE;  // determine relative phase
        if (*(PhbRec.dptr)) PhaseCount--;              // delete oldest phase count
        if (InPhase) PhaseCount++;                     // add new phase to count
        *(PhbRec.dptr) = InPhase;                      // log new phase to buffer
        NextIntPoint(&PhbRec);                         // modify buffer data pointer
        // restore sign
        if (PhaseCount < (PHASE_BUFFER_LENGTH >> 1)) InpMagnitude = -InpMagnitude;

        /* smoother */
        SmootherSum += (InpMagnitude - *(SmoothRec.dptr));  // adjust sum
        *SmoothRec.dptr = InpMagnitude;                     // replace oldest value
        NextLongPoint((struct LBUFFER *) &SmoothRec);        // modify data pointer

        /* fixed corrections */
        Stroke = OffsetCorrection + SmootherSum;       // add offset correction
        Stroke = _IQmpy(GainCorrection, Stroke);       // apply gain correction
        Displacement = (long) Stroke;                  // convert to long integer

        /* linearisation */
        nDivs = Displacement / stepSize;               // number of complete divisions
        if ((Displacement % stepSize) > (stepSize / 2))
             nDivs++;                                   // round divisions
        Displacement += cLinMap[nDivs];                 // apply linearity correction

        /* reset ADC auto-sequencer */
        if ((*ADCST & BIT2) == 0) *ADCTRL2 |= BIT14;    // reset auto-sequencer
          else status.bit.b10 = 1;                      // latch ADC overrun
    } // [total benchmark = 464 cycles]
```

IMPORTANT NOTICE

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265