UPPSALA
UNIVERSITET

UPTEC F08 062
Examensarbete 20 p
November 2008

A Broadband Measurement System for 3D Space Magnetometry

Design, Manufacturing and Characterization

Johan Sundqvist

Civilingenjörsprogrammet i teknisk fysik

Masters Programme in Engineering Physics

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

A Broadband Measurement System for 3D Space Magnetometry

Johan Sundqvist

This master's thesis presents the development of a high-bandwidth electronic measurement system, to be used with a magnetoresistive sensor currently under development at the Ångström Space Technology Centre.

The report covers the working principles of the different subsystems and explains the process of two-layer printed circuit board (PCB) manufacturing. Furthermore, the most important of the manufactured circuits are presented and characterized in terms of linearity, bandwidth and noise level.

With size and weight being of outmost importance for today's space missions, the produced instrumentation amplifier PCBs have been subsequently reduced in size, down to the area of an ordinary 1 euro coin. The final measurement system is able to amplify (by approximately 10-100 times), sample and store low-voltage signals ranging from 0 to 60 MHz in frequency. Some electromagnetic interference was observed for the miniaturized circuits, but a voltage noise density of 80 nanovolts per root hertz has been measured and is expected from the final amplifier with an integrated sensor. Higher bandwidths and lower noise levels can be achieved, but will require multi-layer PCBs.

Handledare: Anders Persson
Ämnesgranskare: Hugo Nguyen
Examinator: Tomas Nyberg
ISSN: 1401-5757, UPTEC F08 062
Sponsor: Rymdstyrelsen, VINNOVA

List of frequently used abbreviations and circuits

Abbreviations

ADC	Analog-To-Digital Converter
ADS	Advanced Design System
DAQ	Data Acquisition
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
EVM	Evaluation Module
IC	Integrated Circuit
instAmp	Instrumentation amplifier
op-amp	Operational amplifier
PCB	Printed Circuit Board
SDT	Spin-Dependent Tunneling

Circuits

Name	**Description**	**Manufacturer**
ADS6145	Analog-To-Digital Converter	Texas Instruments
AAL-002	Magnetic sensor	NVE Corporation
LT1763	Voltage regulator	Linear Technology
MAX4304	Operational amplifier	Maxim IC
MAX4305	Operational amplifier	Maxim IC
OPA657	Operational amplifier	Texas Instruments
TSW1100	Data acquisition module	Texas Instruments

Contents

1 Introduction **1**

2 Development of Measurement system **3**
2.1 Theory for the measurement system 4
2.1.1 The SDTM sensor 4
2.1.2 The instrumentation amplifier 5
2.1.3 Analog-to-digital conversion and data acquisition . . . 10
2.2 Printed Circuit Boards 13
2.2.1 Design 13
2.2.2 Milling 14
2.2.3 Component placement and soldering 15
2.2.4 Produced PCBs 17
2.3 ADC and DAQ system 23
2.4 Software development 25

3 Characterization of developed circuits **27**

4 Results **29**

5 Discussion **33**

6 Conclusions **37**

Acknowledgments **39**

Bibliography **41**

Appendices **42**

A Survey on Magnetoresistive Magnetometers for Space **45**

B DAQ communication program source code **53**

C MIDAS users's manual **67**

Chapter 1

Introduction

The Spin-Dependent Tunneling Magnetometer project

Magnetic sensing is often of great interest and importance in scientific space missions. With the ongoing trend of satellite miniaturization, largely originating from the fact that launch costs are high and that low weight is of great importance, conventional scientific instruments must also be miniaturized. This implies Microelectromechanical systems (MEMS) based technology.

The Spin-Dependent Tunneling Magnetometer (SDTM) is a MEMS-based sensor, aimed to be employed on micro- or nanosatellites.

Scope and purpose of thesis

The intended scope and purpose of this thesis is to develop and characterize a measurement system for the new type of SDTM sensor that is currently being developed at the Ångström Space Technology Centre (ÅSTC). An existing requirements specification states that the final measurement system should be able to amplify, sample and store low-voltage signals ranging from DC to 500 MHz in frequency [1]. Furthermore, the measurement system should be held as small and light-weight as possible. Throughout the work, great care has been taken in order to maximize bandwidth and reduce size.

The Ångström Space Technology Centre

The ÅSTC, originally founded in the year 2000, is a research group within the Department of Engineering Sciences at Uppsala University. Located at the Ångström Laboratory, research at ÅSTC is focused on MEMS development for space applications. After a period of extensive spin-offs that created companies such as *NanoSpace AB*, *Rotundus AB*, *Ångström Aerospace Corporation AB* and *Kalogi AB*, ÅSTC was re-established again in 2006 with financial help from *The Swedish National Space Board*, *VINNOVA* and Uppsala University.

Apart from the SDTM project, current research at ÅSTC includes projects such as OCOM (optical communication for formation flying spacecraft) and DADU (a miniaturized autonomous submarine).

Previous work

Previous work on this topic has been carried out by Lukas Karlsson as a part of the Master's thesis *Development of Electronics for Ultra-Broadband Space Magnetometry* [2]. The thesis was originally intended to be focused on breadboard system design but gradually shifted towards process development for making prototype circuit boards, and included an extensive, theoretical study of important aspects for high-frequency electronics.

The breadboard and circuit board designs were based on a modular principle where each stage was built on a separate board - an approach allowing for quick evaluation of a large amount of similar circuits and components. Many circuits were evaluated, and most of the individual modules performed well when tested separately but were susceptible to Electromagnetic Interference (EMI) when connected. Hence, the need for integrated electronic circuits emerged.

Chapter 2

Development of Measurement system

The measurement system is composed of several smaller components or subsystems. Some of these operate in the analog domain, others in the digital and a few in both domains. Figure 2.1 shows the schematics of the complete measurement system.

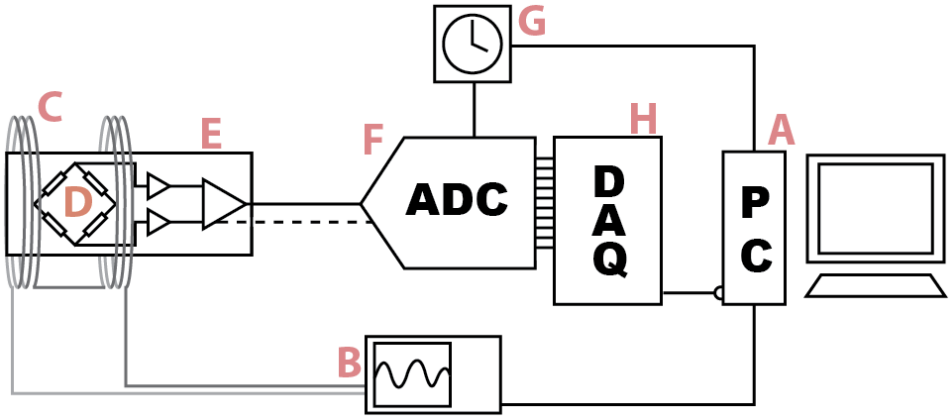

Figure 2.1: Schematics of the complete measurement system.

The process of generating a magnetic field and recording the corresponding output of the magnetic sensor is entirely computer-controlled. A *computer* [A] triggers a *function generator* [B], which generates a current flowing through a *coil* [C]. This applies a magnetic field to the *magnetometer* [D] which puts out a low-voltage signal that is amplified by an *instrumentation amplifier* [E]. The amplified signal is then sampled, by an *Analog-to-Digital Converter* (ADC) [F] with a sample rate set by a *clock circuit* [G]. A high-speed parallel stream of digital signals is produced and temporarily stored, buffered, in a *Data Acquisition* (DAQ) [H] circuit. When a certain amount

of samples has been stored, it is transferred to the computer where it can be processed and stored permanently.

2.1 Theory for the measurement system

2.1.1 The SDTM sensor

This section will explain not so much the physical but the electronic principles of the SDTM sensor. In Appendix A, more information on the physics of Magnetoresistance and Spin-Dependent Tunneling is found.

The basic building blocks of the SDTM sensor is an array of Magnetic Tunneling Junctions (MTJs), connected in parallell. Each MTJ is composed of several layers made of different materials, shown in Fig. 2.2.

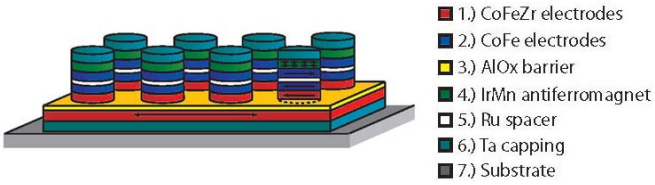

Figure 2.2: Array of Magnetic Tunneling Junctions.

The elements are magnetoresistive, meaning their resistance will change when they are exposed to a magnetic field. In order to measure this change to a high degree of accuracy, a number of MTJ arrays are connected in series, forming a classic measuring instrument called a Wheatstone bridge. A schematic of this circuit is shown in Fig. 2.3.

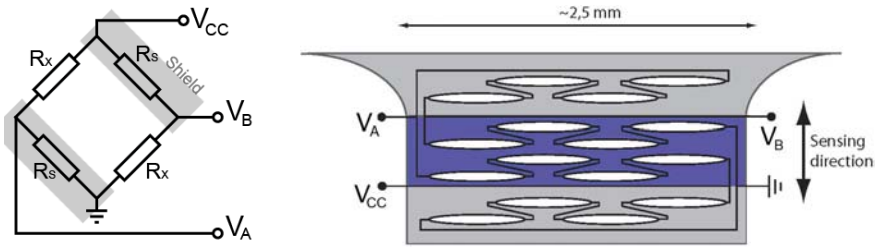

Figure 2.3: Schematic of the Wheatstone bridge (left) and its physical layout (right), as intended for the SDTM project.

Two of the elements, acting as reference resistors, are shielded from magnetic fields, while the other two are left unshielded and exposed to magnetic fields. To derive the expression for how a change in resistance effects the output

voltage of the Wheatstone bridge, let R_s and R_x be the resistances of the shielded and unshielded elements, respectively. By using the rule of voltage division, the voltage at points A (V_A) and B (V_B) is found to be:

$$V_A = \frac{R_s}{R_s + R_x} V_{CC} \tag{2.1}$$

and

$$V_B = \frac{R_x}{R_s + R_x} V_{CC}, \tag{2.2}$$

where V_{CC} is the constant bias voltage. Furthermore, the output voltage V_{out} $(= V_A - V_B)$ becomes:

$$V_{out} = \left(\frac{R_s - R_x}{R_s + R_x} \right) V_{CC}, \tag{2.3}$$

which implies that, if no magnetic field is present ($R_x = R_s$), the sensor's output voltage will be zero and that, if all resistances change proportionally (e.g. with temperature), the output will be unaffected. Finally, if we let ΔR denote the change in resistance of R_x so that $R_x = R_s + \Delta R$ and substitute this into Eq. 2.3, we get the expression:

$$V_{out} = \left(\frac{\Delta R}{2R_s + \Delta R} \right) V_{CC}. \tag{2.4}$$

From Eq. 2.1 and Eq. 2.2 it can be seen that the output signal will be biased around $V_{CC}/2$. It should also be noted that the above equations are valid only if a negligible amount of current is flowing from A to B. To make sure this is the case, the load connected to the sensor must be of high impedance.

2.1.2 The instrumentation amplifier

Since the sensor is configured as a Wheatstone bridge, its output will be a differential signal. The amplitude of this signal will vary with the magnetic field applied, but will generally be a factor of 10-100 lower than the DC bias. This kind of signal is not suitable to pass on directly to the ADC.

In order to remove the bias, and to amplify the differential signal so that it better fits the dynamic range of the ADC, we introduce the simple, yet powerful, *instrumentation amplifier*. In its basic configuration, the instrumentation amplifier consists of three operational amplifiers (op-amps), and converts a differential signal to a single-ended signal. Hence, it is a form of difference amplifier that is able to amplify the difference between two signals while rejecting the common-mode signal present at both inputs. A schematic of the classic instrumentation amplifier is shown in Fig. 2.4. The left and right part of the schematic will henceforth be referred to as the *buffer stage* and the *difference amplifier stage*, respectively.

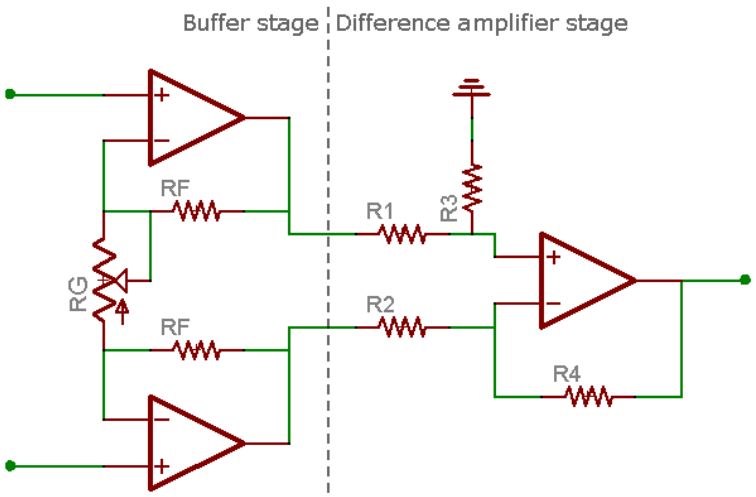

Figure 2.4: Schematic of the classic instrumentation amplifier

The difference amplifier stage

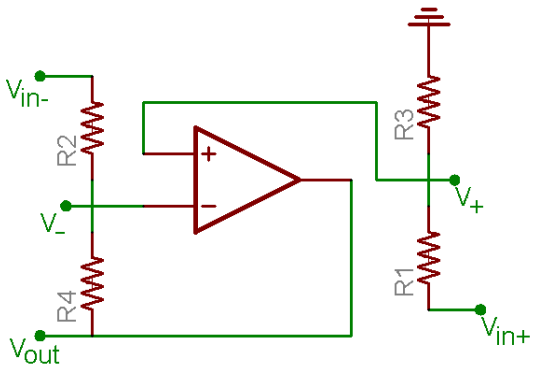

Figure 2.5: Rearranged schematic of the difference amplifier.

To derive the gain of the difference amplifier, we rearrange the resistors according to Fig. 2.5. It can now be seen that the resistor pairs to the left and right of the op-amp act as simple voltage dividers. The relation between the two input voltages (V_{in+}, V_{in-}) and the voltages at the op-amp's terminals (V_+, V_-) is given by:

$$V_- = (V_{out} - V_{in-}) \left(\frac{R_2}{R_2 + R_4} \right) + V_{in-} \tag{2.5}$$

and

$$V_+ = V_{in_+} \left(\frac{R_3}{R_1 + R_3} \right). \tag{2.6}$$

By using that $V_- = V_+$ for an ideal negative-feedback op-amp [3], it is possible to merge the two equations and rearrange them to obtain an expression for the difference amplifier's output:

$$V_{out} \left(\frac{R_2}{R_2 + R_4} \right) = V_{in_+} \left(\frac{R_3}{R_1 + R_3} \right) - V_{in_-} \left(\frac{R_4}{R_2 + R_4} \right), \tag{2.7}$$

which, after some further simplification, gives the final expression for the connection between in- and output of the difference amplifier:

$$V_{out} = V_{in_+} \left(\frac{R_3}{R_1 + R_3} \right) \left(1 + \frac{R_4}{R_2} \right) - V_{in_-} \left(\frac{R_4}{R_2} \right). \tag{2.8}$$

Equation 2.8 might look a bit discouraging, as it states that the difference amplifier amplifies the two inputs to different extents and does not reject the common-mode signals very well. However, this is solved by restricting the resistor values so that:

$$\frac{R_1}{R_3} = \frac{R_2}{R_4}. \tag{2.9}$$

Then, by setting $R_1 = R_2$ and $R_3 = R_4$, the output voltage V_{out} becomes:

$$V_{out} = \left(\frac{R_3}{R_1} \right) (V_{in_+} - V_{in_-}). \tag{2.10}$$

Now, it is clear that the differential amplifier is capable of both rejecting the DC offset from the sensor, and amplifying the differential signal. Still, this is not all that is needed to connect the sensor to the ADC.

The buffer stage

There are two problems with the differential amplifier configuration. First, the two voltages V_{in_-} and V_{in_+} in Fig. 2.5 will see different input impedances [4]. The output impedance of the sensor will be large compared with these impedances, creating a situation where a non-neglectible leakage current will flow from the sensor.

The second problem arises if we want to vary the amplification by adjusting the gain of the circuit. In such a case, both R_1 and R_3, or R_2 and R_4, must change to the same extent. Both these problems will be solved by

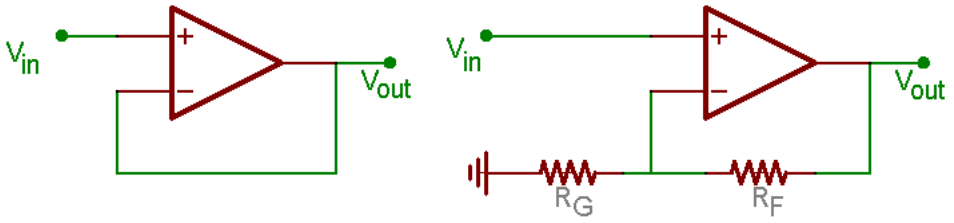

Figure 2.6: Two different buffer circuits; a voltage follower (left) and a non-inverting (right).

adding a buffer stage between the sensor and the differential amplifier. The buffer stage consists of two *non-inverting amplifiers* (or, if no amplification is required, *voltage followers*), one for each input. The schematics for these circuts are shown in Fig. 2.6. Due to the feedback, the input impedance of the amplifier circuit is larger than or equal to the op-amp's input impedance Z_{in} [5]. For most op-amps, Z_{in} is in the 10^6 to 10^9 Ω range, which is substantially higher than the output impedance of the sensor. This gives an impedance bridging effect where a very small amount of current is leaking from the sensor. Also, since the output impedance of the op-amp is small (typically a 1-10 Ω) [5], the 1-10 $k\Omega$ input impedance of the difference amplifier will no longer be a problem.

The output voltage (V_{out}) of a non-inverting amplifier is given by:

$$V_{out} = \left(1 + \frac{R_F}{R_G}\right) V_{in}, \tag{2.11}$$

where V_{in} is the input voltage and R_F, R_G are resistances according to Fig. 2.6. The gain can easily be adjusted by changing value of either R_F or R_G. However, since the buffer stage consists of two non-inverting amplifiers, one would still have to adjust two resistors in order to change the amplification of the circuit without disrupting the balanced behavior of the signal. This is solved by letting the two amplifiers share a gain resistor R_G, Fig. 2.7. This resistor makes sure that both amplifiers have the same current flowing through their feedback loop and, hence, keeps the signal balanced.

To analyze how the amplification depends on R_G, we let $V_{in_{diff}}$ and $V_{out_{diff}}$ denote the differential input, $V_{in_1} - V_{in_2}$, and output, $V_{out_1} - V_{out_2}$, respectively. Using Eq. 2.11 and, again, the fact that for an ideal op-amp $V_- = V_+$, we get:

$$V_{out_{diff}} = \underbrace{\left(V_{in_1} + \frac{R_F}{R_G} V_{in_{diff}}\right)}_{V_{out_1}} - \underbrace{\left(V_{in_2} - \frac{R_F}{R_G} V_{in_{diff}}\right)}_{V_{out_2}} \tag{2.12}$$

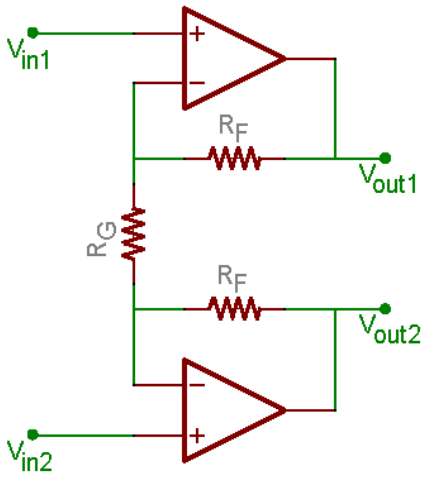

Figure 2.7: The buffer stage of the instrumentation amplifier with a common gain resistor R_G.

$$\Rightarrow V_{out_{diff}} = V_{in_{diff}} + 2\frac{R_F}{R_G}V_{in_{diff}} = \left(1 + 2\frac{R_F}{R_G}\right)V_{in_{diff}}. \tag{2.13}$$

From this equation, it is clear that the buffer stage amplifies both inputs to the same extent and does not disrupt the balanced behavior of the signal. When adding the difference amplifier in order to form the instrumentation amplifier, the relation between the differential input to the buffer stage and the single-ended output of the difference amplifier becomes:

$$V_{out} = \left(1 + 2\frac{R_F}{R_G}\right)\left(\frac{R_3}{R_1}\right)V_{in_{diff}}. \tag{2.14}$$

The gain of the whole instrumentation amplifier circuit can be set by adjusting R_G, e.g. by letting R_G be a potentiometer as in Fig. 2.4.

It is very important to note that Eq. 2.14 is valid only if the values of R_{1-4} are chosen according to Eq. 2.9. Also, the two feedback resistors, R_F, of the noninverting amplifiers in the buffer stage must be of the same value. Great care should be taken when selecting resistors, as the slightest mismatch will disrupt the circuit's common-mode rejection ability. As an example; if the amplifier is designed so that is has a gain of 1, a 0.1% mismatch of a single resistor will reduce the amplifiers common-mode rejection ratio (CMRR) to 66 dB [4].

2.1.3 Analog-to-digital conversion and data acquisition

Sampling of electronic signals is done by an Analog-to-Digital converter (ADC) circuit. An ADC has two key characteristics, namely sampling rate (f_s) and bit depth. The bit depth sets the number of discrete levels that the digital signal can have, where consequently, M bits gives 2^M different levels. Combined with the maximum voltage range, the bit depth gives the magnitude of the Least Significant Byte (LSB), often denoted q.

Quantization error

The upper graph of Fig. 2.8 shows a continuous signal (blue line) and a sampled version of the same signal (red line) with each sample marked as a red circle in the plot.

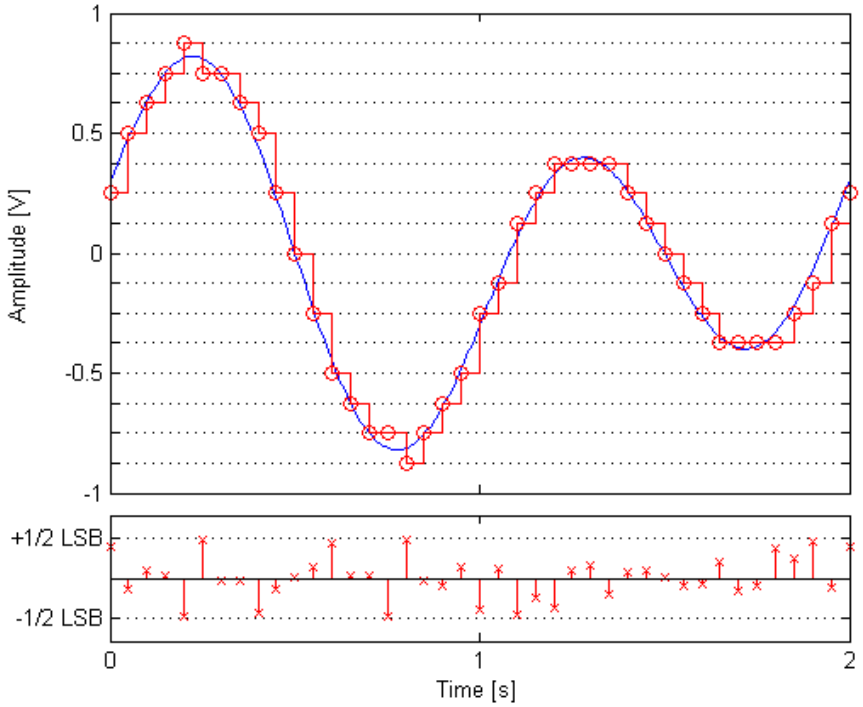

Figure 2.8: Top graph: a sampled version (red) of a continuous signal (blue). Bottom graph: the quantization error.

The bottom graph seen in Fig. 2.8 shows the quantization error, i.e. the difference in amplitude between sample and continuous signal at the time of each sample. Each such difference will have an amplitude of $-1/2$ to $+1/2$ LSB. The difference between the sampled and continuous signal is called the quantization noise. Modeling quantization noise is not trivial unless some

assumptions are made. Common assumptions are that the ADC is ideal (fully linear), and that each quantization error is uniformly distributed over the $\pm 1/2$ LSB range [6]. In such cases, the root mean square (RMS) of the quantization error is:

$$RMS_e = \frac{q}{\sqrt{12}} = \frac{1}{\sqrt{12}} \left(\frac{2A}{2^M} \right) = \frac{A}{\sqrt{3} \cdot 2^M}, \tag{2.15}$$

where A is the full-scale amplitude and M the bit depth of the ADC. Equation. 2.15 states that the quantization noise level is independent of the input signal's amplitude. In order to maximize the circuit's Signal-to-Noise Ratio (SNR), the amplitude of the input signal should be held as close as possible to the ADC's voltage range.

Sampling rate and aliasing

The Nyquist-Shannon sampling theorem sets a limit for the highest frequency, f_{max}, that can be reconstructed from a sampled signal. The relation between f_s and f_{max} is:

$$f_{max} < \frac{f_s}{2}. \tag{2.16}$$

The frequency $f_s/2$ is often called the Nyquist frequency, f_N, and frequencies higher than this will suffer from aliasing and appear 'folded' around the Nyquist frequency in a frequency-domain representation, as seen in the upper part of Fig. 2.9. Aliasing is usually an unwanted effect and ADC inputs are often low-pass filtered (such as in the middle part of Fig. 2.9) in order to attenuate frequencies higher than f_N. However, with careful filtering, aliasing can be used to work favorably through undersampling. This requires the input signal to be band-pass filtered with a passband between two multiples of f_N and allow frequencies higher than f_N to be properly sampled.

ADC and DAQ circuits

Most, if not all, manufacturers of high-speed ADCs produce evaluation modules (EVMs) for their circuits. An EVM is, as the name implies, built for the purpose of quick and easy evaluation of a specific ADC. The manufacturers, of course, want to show their ADC circuit at it's best and, hence, the EVMs are low-cost but fairly fine-tuned circuit boards. The ADC EVM takes care of the sampling and, in turn, produces a high-speed stream of parallel digital signals.

Due to the high speed and accuracy of the ADC, large amounts of data are produced: 14 bits per sample at 100 million samples per second produces 175 MB of data each second. Apart from the obvious problem of filling up an ordinary-sized Hard Disk Drive (HDD) within a few minutes, a data rate

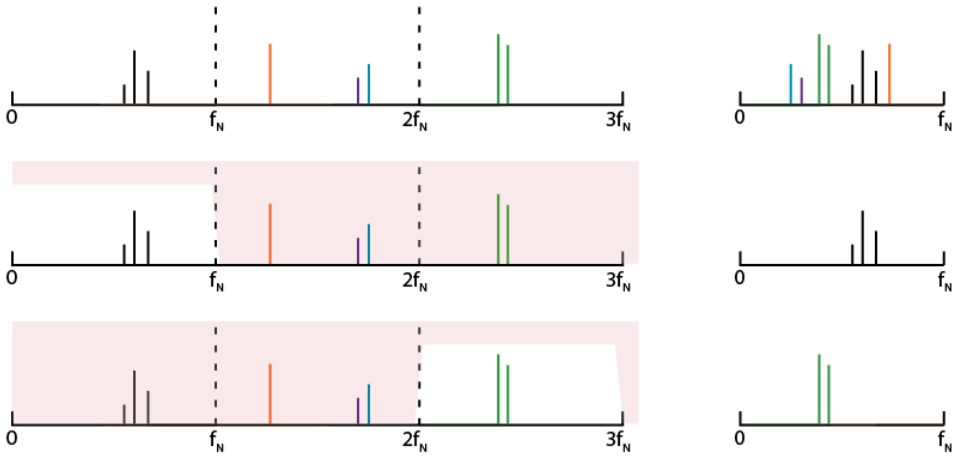

Figure 2.9: Example of aliasing and the effects of filtering. Left and right halves shows spectral content before and after sampling, respectively.

of 175 MB/s is much too fast even for a modern desktop computer. A fast 3.5" HDD spinning at 1500 rpm can write a maximum of 125 MB of data per second [7] if the data is written sequentially (which it seldom is), and a standard 33 MHz 32-bit PCI bus can transfer data at a theoretical maximum speed of 133 MB/s. Equipping a computer with the high-end components required for high transfer rates is very expensive.

To solve this problem, most manufacturers also produce data acquisition (DAQ) modules that mate with their ADC EVMs. The acquisition modules simply act as digital FIFO (First In, First Out) queues, where the high-speed stream of parallel data is processed by a Field-Programmable Gate Array (FPGA) and stored in a fast, but often relatively small, memory. When a given number of samples have been stored, the DAQ module transfers the data to a computer by USB or an other serial interface. Since the serial interface is slow compared with the ADC's parallel data stream, sequential measurements will be taken as "bursts" where the actual measurement is made during a fraction of a second and data is transferred to the computer during several seconds.

The various ADC EVM and DAQ combinations available today are substantially cheaper than their computer expansion card counterparts, often called digitizers. In Table 2.1, a quick review of two different hardware setups is shown.

Table 2.1: Hardware features for a typical ADC+DAQ and digitizer.

	ADC+DAQ	**Digitizer**
Manufacturer	Texas Instruments	National Instruments
Product name(s)	ADS6145EVM/TSW1100	NI PXI-5142
Sample rate	125Msps	100Msps*
Resolution	14 bits	14 bits
Input channels	2†	2
Maximum voltage range	$2V_{p-p}$	$10V_{p-p}$
On-board memory	16 MB/channel	256 MB/channel
Computer interface	Serial (USB)	Parallel (PXI)
Price‡	$1100	$17000

*Can produce a sample rate of 2Gsps by the use of Random interleaved sampling.
†One DAQ card can connect to two ADC EVM boards.
‡As of June 2008.

2.2 Printed Circuit Boards

A large part of the work behind this thesis has consisted of manufacturing various Printed Circuit Boards. This section will explain the manufacturing process and present some of the produced PCBs.

2.2.1 Design

Today there exists a wide range of computer software used for rapid PCB production, ranging from very extensive and somewhat complex suites (such as *OrCAD*, produced by Cadence Design Systems) aimed for large-scale industry production to more basic platforms (such as *Eagle*, produced by Cadsoft), used in large parts by hobbyists.

At ÅSTC, the software of choice is ADS, Advanced Design System, produced by an Agilent subdivision named *Agilent EEsof EDA*. ADS is mainly aimed for high-frequency design, providing tools for extensive high-frequency simulations. Since the electronics used in the SDTM project is focused at the HF-VHF [1] bands, the simulation capabilities of ADS are very useful. Another useful assistance has been the basic ADS user guide written by Lukas Karlsson as a part of his thesis [2].

The starting-point of all PCBs is the schematic. This should contain all the required components, including vias (holes that connect two layers of

[1]High Frequency to Very High Frequency, as defined by [8]

the PCB) and external connections, and show how they are interconnceted. A screenshot of an ADS schematic is shown in the left half of Fig. 2.10.

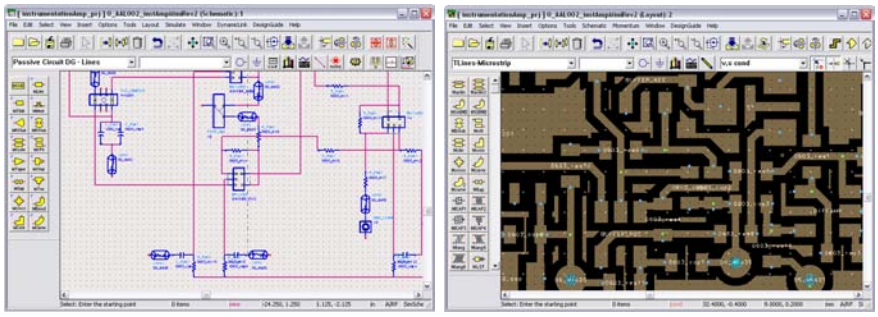

Figure 2.10: Screenshot of a schematic (left) and a layout (right) in ADS.

When all components have been placed and properly connected, the physical layout of the PCB can be made. The components in the schematic are placed, one by one, on the layout. All components have to be manually connected by signal traces drawn by the designer. ADS aids the designer by showing dotted lines between pads and traces that should be, but not yet are, connected. The right half of Fig. 2.10 shows ADS in the layout mode where red areas represent the copper pads and traces of the top layer and the black areas represent copper that is to be removed by milling or etching. The layout can often be the most time-consuming part of the design process, especially if the components are to be packed tightly.

2.2.2 Milling

The physical process of realizing the final layout on a blank PCB copper board requires either etching or milling. In this project, the latter method has been used through a *LPKF ProtoMat S62* prototyping mill shown in Fig. 2.11. After the layout is finished, it is exported to a number of Gerber files which are then loaded at the computer controlling the mill. Once all files are properly imported and the correct bits and drills are installed, the computer takes full control over the milling process, requiring manual intervention only when the board is to be flipped over for backside processing.

Figure 2.11: LPKF Protomat S62 prototyping mill.

2.2.3 Component placement and soldering

When the PCB has been milled it will look like the left part of Fig. 2.12. Before the solder paste is applied and the components are placed, all excess copper is removed so that the PCB looks like in the right half of Fig. 2.12. After cleanup the individual pads are easier to locate, which makes placement of the components less cumbersome.

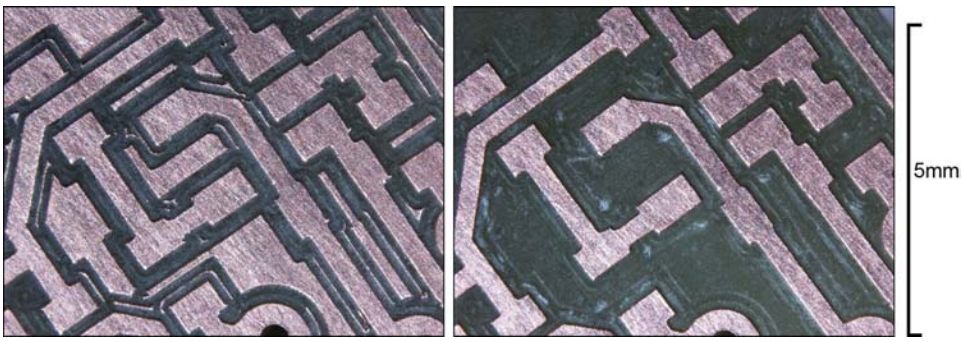

Figure 2.12: Milled PCB (left) and the same PCB after removal of the excess copper (right).

The second step of the assembly process is to apply solder paste to all component pads. Prefferably, this is done under a microscope using a sharp pointy object such as the tip of a scalpel. Applying the right amount of paste to all pads is essential, since too little paste may cause a bad or no connection to the pad, and too much paste may cause short circuits or misalignment of components. The left part of Fig. 2.13 shows an appropriate amount of solder paste applied to each pad.

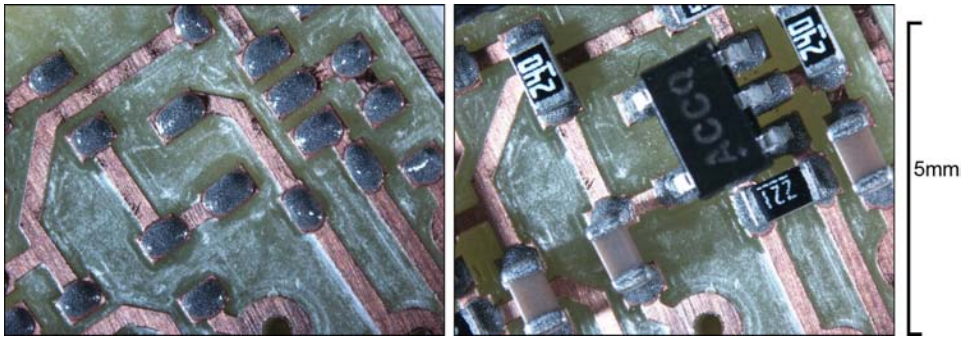

Figure 2.13: Solder paste applied (left) and components placed (right).

With solder paste applied, all surface-mount components can be placed. Due to the high component density and small packages used, bare handed placement is not possible. Tthe PCB used for demonstrational purpose in this section uses the 0603 package for resistors and capacitors, each component being 1.6×0.8 mm. Luckily, ÅSTC has a manual Pick-and-Place system to assist the assembler. The PCB is placed under the PnP's microscope and each component is lifted by the vacuum of a hollow needle tip. The needle mechanism slides along two rails, increasing hand steadiness, and releases the component as it is firmly pressed against the PCB.

Industry-scale reflow soldering often requires special ovens or infrared lamps, but previous work has shown that an ordinary temperature-controlled heated plate can be sufficient, if used properly [2]. Soldering the assembled PCB is fairly straight-forward as long as the temperature profile of the solder paste is roughly followed. After reflow soldering, the solder joints should look like those shown in Fig. 2.14.

With reflow soldering done, all that is left of the assembly process is to manually solder the through-hole components that could not be reflow soldered. This usually includes vias, coaxial jacks and power supply connectors.

Before the circuit is connected for the first time, all solder joints are thoroughly examined under a microscope.

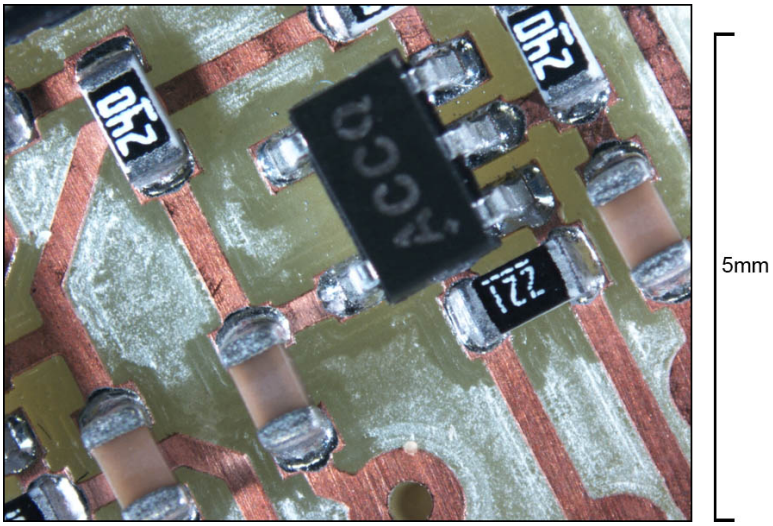

Figure 2.14: PCB after reflow soldering.

2.2.4 Produced PCBs

As a part of the SDTM project, a number of different instrumentation amplifiers (explained in section 2.1.2) has been designed, produced and tested. This section will go through a few of those designs. Although the design has changed throughout the different versions, some basic concepts have remained throughout all designs. Among the most important ones are:

- **Unbroken Ground plane**
  In order to keep the ground plane as unbroken as possible, voltage supply rails are drawn along the edges of the PCB. Also, few or no signal traces are drawn through the ground plane.

- **Decoupling of operational amplifiers**
  By adding a resistor in front of the decoupling capacitor at each voltage supply input, the voltage supply is low pass filtered. This reduces noise. Since current flows to the op-amp, there is a voltage drop over the resistor. In order to provide the amplifier with a proper voltage level, and to reduce thermal noise and power dissipation, the resistance should be held small, typically less than 10 Ω. The +5 V and -5 V voltage rails are also decoupled.

- **No excess copper**
  All copper not used as signal traces or component pads is removed, partly to improve Electromagnetic Compatibility (EMC) characteristics but mostly because it makes the process of placing components a lot easier, especially when the component density is high. Also, all

copper underneath the integrated circuits is removed from the ground plane at the bottom layer of the PCB. This is done to reduce capacitance between the ground plane and the pins of the IC. Lifting the copper is easily done under a microscope, using a scalpel and a pair of sharp tweezers.

Since the first SDTM sensor still is to be manufactured, all sensor-equipped instAmp designs include a commercially available magnetoresistive sensor from NVE Corporation, namely *AAL002*. This sensor is similar to the SDTM in terms of principle of operation, output impedance and output levels, but is of less bandwidth and higher noise than what is expected of the SDTM sensor.

instAmp v0.1

The very first version of the instrumentation amplifier was made mostly for the purpose of learning the ADS software. The PCB is shown in Fig. 2.15 and measures 43.3×33.8 mm for a total area of 1464 mm^2.

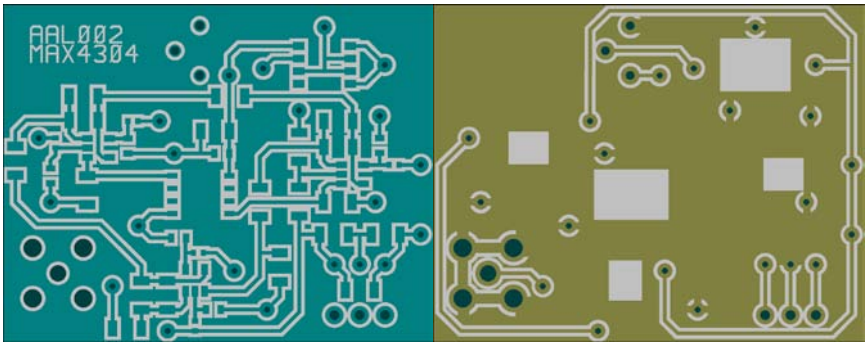

Figure 2.15: Top and bottom layer of instAmp version 0.1.

The *AAL-002* magnetic sensor is located in the middle of the PCB and the power supplied by a *LT1763* low-noise voltage regulator from Linear Technologies. With a potentiometer, the voltage supply of the sensor can be adjusted between 0 and 5 V. Since the output of the sensor is proportional to its voltage supply [9], the output of the circuit can be adjusted. The instrumentation amplifier has three *MAX4304* op-amps and fixed gain.

instAmp v1.0

In order to miniaturize the circuit further, instAmp v1.0 uses the 0603 package for its passive components, apart from a few 1206 decoupling capacitors. It also incorporates more intricate wiring, where many traces are drawn underneath and through the pads of other components. As a result of this, the PCB, shown in Fig. 2.16, measures only 32×14.5 mm for a total area of only 464 mm^2. This is only 31% the size of version 0.1.

As the circuit is made a lot smaller, the signal traces are shorter and less susceptible to electromagnetic interference. On the other hand, as the signal traces get closer to each other, EMC issues such as capacitive coupling may cause trouble at high frequencies.

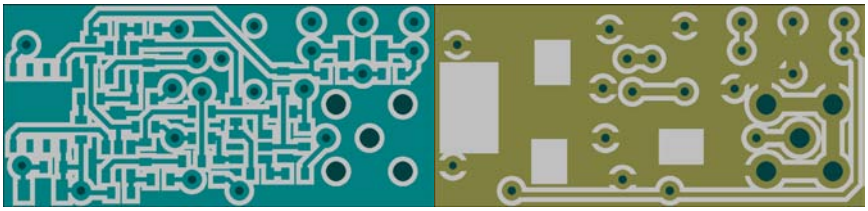

Figure 2.16: Top and bottom layer of instAmp version 1.0.

The substantial reduction of size comes at another cost, as the *LT1763* voltage regulator is removed from the design. Instead, the sensor (located near the left edge of the board) is supplied power directly by the +5 V rail. The instrumental amplifier still uses three *MAX4304* op-amps and has a variable gain that is set by a through-hole potentiometer.

instAmp v1.1

Since all vias introduce unwanted parasitic capacitances [10] [11] and act as small antennas, the through-hole potentiometer in version 1.0 is replaced by a surface-mount counterpart. This, combined with the rearrangement of a few decoupling capacitors allows instAmp v1.1 to have 7 vias fewer than its predecessor. Rearranging the decoupling capacitors introduces a small risk, as they generally should be placed as near the load as possible in order to provide a low-impedance path to ground for high-frequency disturbances.

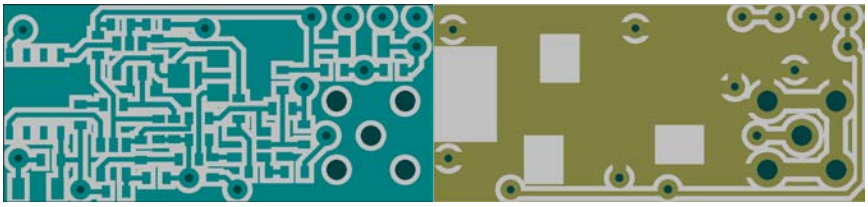

Figure 2.17: Top and bottom layer of instAmp version 1.1.

The dimensions of this circuit are 32×14 mm, which is basically the same as for version 1.0. In Fig. 2.18, a photograph of the circuit is shown with a standard BNC plug next to it as a size reference.

Figure 2.18: Fully assembled instAmp v1.1.

instAmp evaluation version

In order to measure the circuit's bandwidth and noise level, two PCBs with the layout shown in Fig. 2.19 were produced. These two PCBs were equipped with different op-amps, namely Maxim *MAX4305* and the high impedance

FET-input *OPA657* from Texas Instruments. Additionally, a third (non-miniaturized) circuit using *MAX4304* op-amps and a design similar to instAmp v0.1 was used as reference. All circuits include a positive and negative input pin instead of the *AAL002* sensor. Table 2.2 lists the most important features of the different op-amps. The comparison is not completely fair, as the two manufacturers sometimes specify their circuits for slightly different conditions.

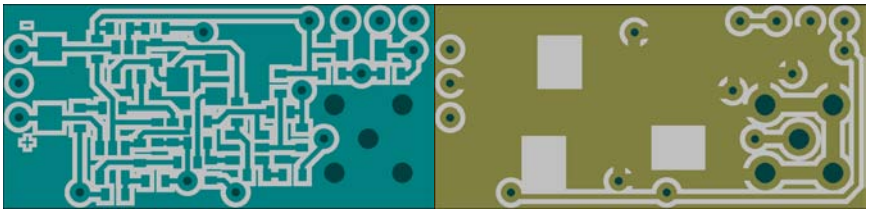

Figure 2.19: Top and bottom layer of instAmp, op-amp evaluation version.

Table 2.2: Some important specifications of the different op-amp circuits used. (From the manufacturers' datasheets [12], [13].)

	MAX4304	**MAX4305**	**OPA657**	
Z_{in} (differential)	10^4	10^4	10^{12}	[Ω]
Z_{in} (common-mode)	10^6	10^6	10^{12}	[Ω]
Quiescent supply current	20	20	14	[mA]
Full-power bandwidth	285	320	180	[MHz]
Slew rate	1000	1400	700	[V/μs]
Minimum stable gain	2	10	7	
Voltage output swing	±3.4	±3.4	±3.5	[V]
Voltage noise density	2.1	2.1	4.8	[nV/$\sqrt{\text{Hz}}$]
Current noise density	3100	3100	1.3	[fA/$\sqrt{\text{Hz}}$]
CMRR, typical (min)	95 (80)	95 (80)	89 (81)	[dB]

2.3 ADC and DAQ system

Previous work [2] included a 40 Msps ADC EVM from Linear Technology (*DC782-A*) with a corresponding DAQ module (*DC718B*). Apart from the low sampling frequency, the DAQ's small memory size puts constraints on the lowest frequencies measurable. In addition, the controller software that came bundled with the DAQ proved to be quite poor. Furthermore, no Software Development Kit (SDK) was supplied. These flaws made it clear that a new ADC and DAQ solution was needed.

In order to find the most suitable combination of ADC and DAQ boards, different manufacturers' datasheets were thoroughly examined. Specifications of some of the more interesting combinations are presented in table 2.3.

Table 2.3: Specifications of ADC evaluation modules and their corresponding DAQ modules.

	Input type [SE/DI][2]	$\mathbf{f_{max}}$ [Msps]	**Input range** [V_{p-p}]
AD AD9254-150EBZ	DI	150	2
AD AD9246-125EBZ	SE	125	2
LT DC782A-S	SE	125	2
Maxim MAX1255EVKIT	SE	95	2.2
Maxim MAX19586EVKIT	SE	80	2.56
NS ADC14V155LFEB	SE	155	2
TI ADS6145EVM	SE	125	2

	Inputs	**Memory**	**Mates with**
AD HSC-ADC-EVALB	2	2x32 kB	AD9246-125EBZ
LT DC718B	1	128 kWord	DC782A-S
NS WAVEVSN BRD 5.1	1	Up to 8 MB	ADC14V155LFEB
TI TSW1100	2	2x1 MWord	ADS6145EVM

Analog Devices' AD92xx-series and Texas Instruments's ADS6145 are neck and neck, with AD having the slight advantage of a differential input. However, due to the bad and sometimes even missing documentation of Analog Devices' DAQ modules, the final choice fell on the TI ADS6145 and TSW1100 combination.

The software that comes bundled with the two modules is decent, but can

[2]Single-ended/Differential

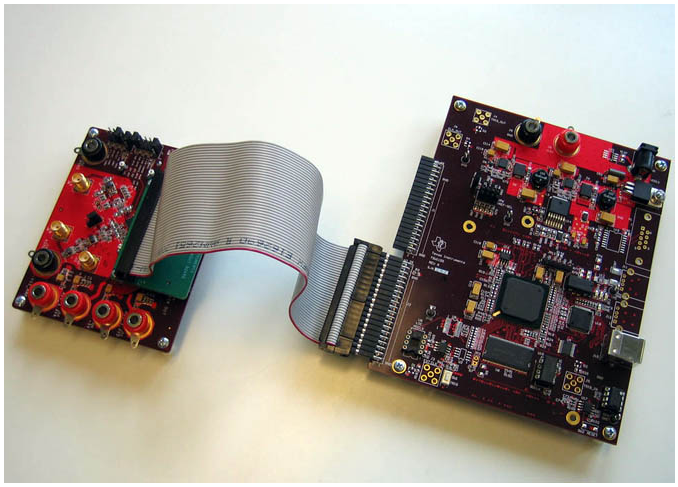

Figure 2.20: Texas Instruments's ADS6145EVM ADC (left) and TSW1100 DAQ (right).

only take one measurement at a time and requires the user to save the acquired data manually by clicking buttons and entering filenames. This might be sufficient for the kind of quick evaluation measurements that the ADC EVM is aimed for, but is not very optimal for the SDTM test bench. Fortunately, Texas Instruments sends a SDK bundled with the software. The SDK includes a DLL[3] with an accompanying header file that, combined, allow a user to write programs that can communicate with the DAQ board. Such programs can be written in various high-level programming languages such as C or C++.

The ADC requires a clock signal to operate. This signal is supplied by an *ispClock5620A* clock generator circuit from Lattice Semiconductor. This circuit is In-System Programmable (ISP) and can output clock signals ranging from 8 to 400MHz at both single-ended and differential low-voltage logic levels such as LVTTL, LVCMOS, LVDS, eHSTL, etc.

[3]Dynamic-link library, a library of subroutines that are imported to the main program at runtime as opposed to static libraries that are included when the program is compiled.

2.4 Software development

Communication with the TSW1100 DAQ board.

As previously stated in section 2.3, the software that comes bundled with the DAQ board is not well suited for the SDTM test bench. Therefore, a simple program has been written that handles communication with the DAQ. By calling the program with a number of commandline parameters and arguments, the user can control a variety of settings such as number of samples, capture channel(s) and filename(s). The source code for this program is included in Appendix B.

Matlab GUI

In order to enhance user friendliness and add further functionality, a Matlab Graphical User Interface (GUI) has also been developed. A screenshot of the GUI is shown in Fig. 2.21

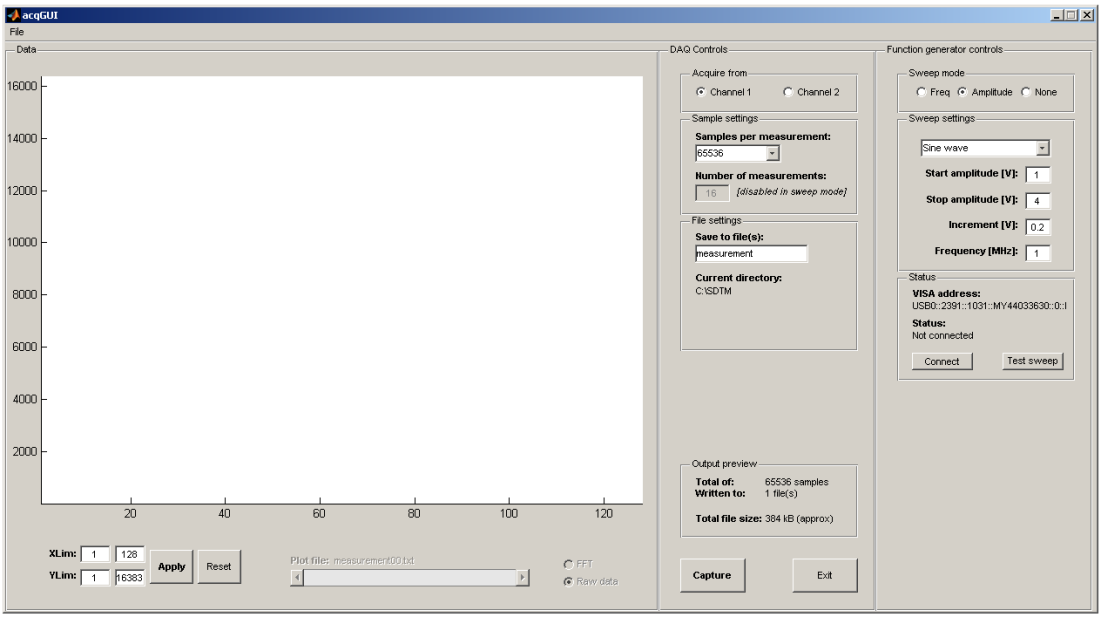

Figure 2.21: Screenshot of the MIDAS software.

The program, named *MIDAS* (Matlab Interface for Digital Acquisition of Signals), lets the user specify things like input channel, number of samples per measurement, total number of measurement and save path. When the 'Capture' button is pressed, the communication program (explained in the previous section) is called repeatedly until all measurements are made. Then all data is imported back to Matlab where it is plotted and can be further

processed. The user can browse through the most recently captured files, plotting either the raw data or the signal's spectral content (using the FFT algorithm with a variety of window functions).

Additionally, a connected function generator can be controlled. The user can let the frequency or amplitude sweep through a certain range while taking subsequent measurements, offering a crude estimate of the connected system's transfer function.

Further information about the MIDAS software is found in the MIDAS user's manual, Appendix C.

Chapter 3

Characterization of developed circuits

This chapter will explain how the most important measurements have been carried out.

Hysteresis sweep

As a first step towards testing and verifying the design, a simple hysteresis sweep was made by placing an instAmp v1.0 circuit inside a coil, with the sensor's sensitive axis aligned in parallel to the magnetic field. The setup is shown in Fig. 3.1.

A direct current was applied to the coil, and the output voltage of the circuit was measured as a function of this current.

Low-frequency AC measurement

In addition to the hysteresis sweep, a 10 kHz AC measurement was made using an instAmp v1.0 circuit. The circuit was placed inside a coil (as seen in Fig. 3.1) and the sensor's positive output signal, versus ground, was probed to an oscilloscope. At the same time, the amplified output of the circuit was measured by the same oscilloscope.

Bandwidth measurement

Using a HP 4195A Network Spectrum Analyzer (NSA), the frequency responses of two different instAmp evaluation circuits were measured. One circuit had *OPA657* and the other had *MAX4035* op-amps. In addition, a non-miniaturized circuit (much like instAmp version 0.1 but without sensor) was used as reference. This circuit incorporated *MAX4304* op-amps.

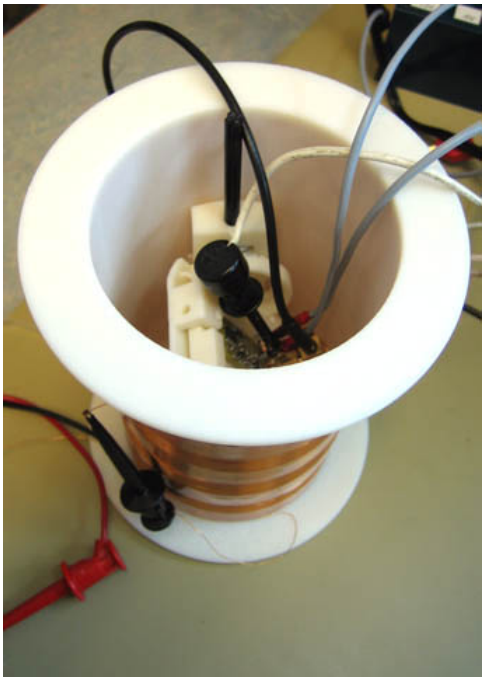

Figure 3.1: Setup for the hysteresis sweep and low-frequcncy AC measurement.

Noise measurement

Finally, a noise measurement was carried out for each of the three circuits described above. In order to examine the noise characteristics of the *AAL002*, a non-miniaturized circuit (instAmp v0.1) with sensor was also measured for noise. Power was connected to each circuit and all inputs were terminated with 10 kΩ resistors. This was done to simulate the source impedance of the sensor. Since the two miniaturized circuits suffer from EMI in the 90-110 MHz range, the output signals were low-pass filtered. All outputs were connected to the HP 4195 NSA.

Chapter 4

Results

Hysteresis sweep

Figure 4.1 shows the performed hysteresis sweep. The circuit's output is linear in the (-200)-(-40) and 40-200 mA ranges. Higher currents produce a magnetic flux density high enough to saturate the sensor, and at lower currents the measurements becomes somewhat unreliable due to lack of precision in instruments and interference from the earth's magnetic field.

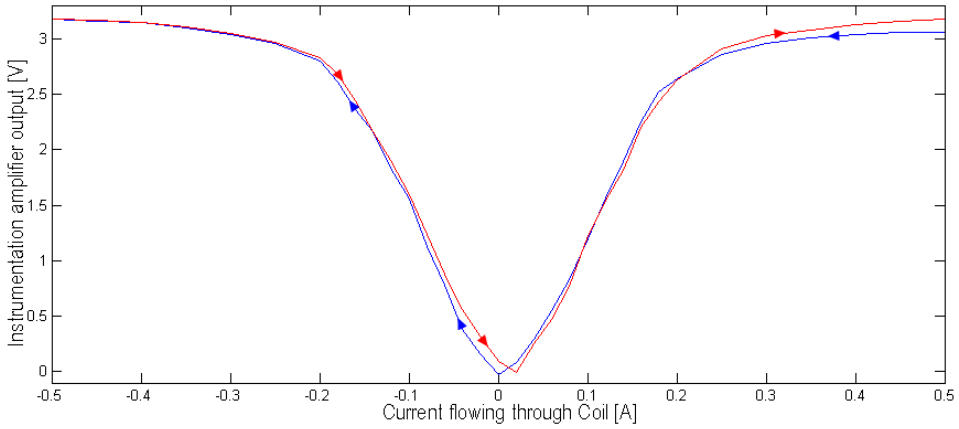

Figure 4.1: Plot of the circuit output voltage as a function of the current flowing through the coil in which the circuit is placed. The blue and red line are sweeps from 0.5 to -0.5A and -0.5 to 0.5A, respectively.

Low-frecuency AC measurement

In Fig. 4.2, the probed signal is compared with the amplified signal of the instAmp output. All DC bias has been removed from both outputs.

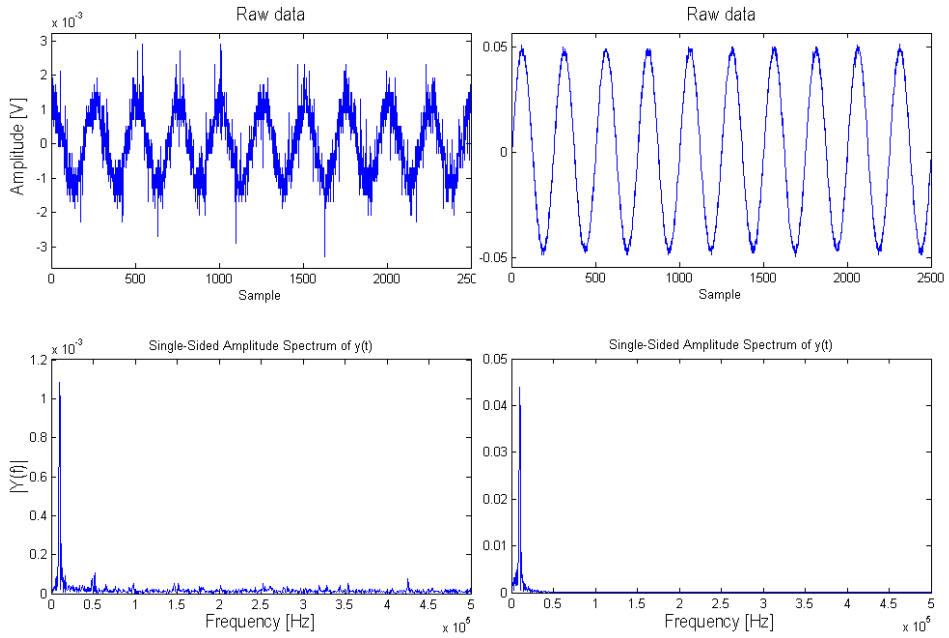

Figure 4.2: Comparison of sensor (left) and circuit (right) output.

Bandwidth measurement

The results of the frequency reponse measurements are shown in Fig. 4.3. Approximate -3 dB bandwidths are 60 MHz for the *MAX4304* and *OPA657* circuits, and 20 MHz for the *MAX4305* circuit.

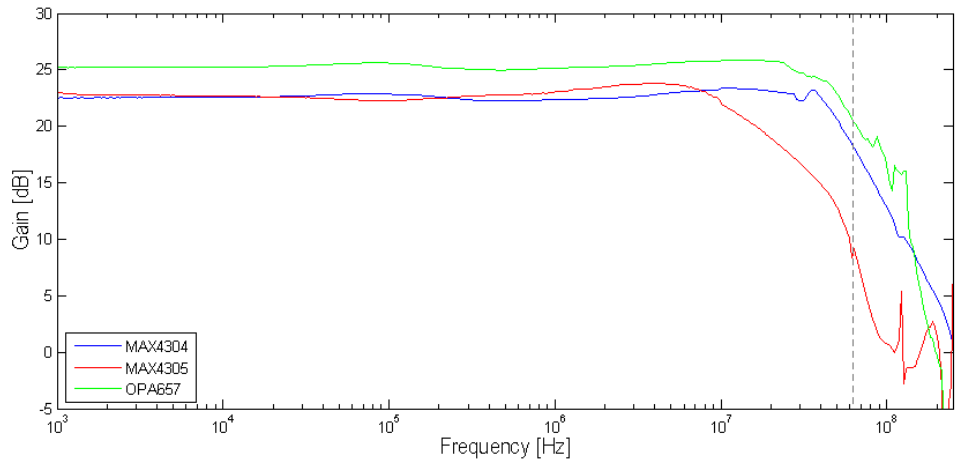

Figure 4.3: Frequency responses for the different instAmp circuits. The dotted gray line marks the maximum Nyquist frequency for the ADS6145.

Noise measurement

Figure 4.4 shows the voltage noise density for the two miniaturized circuits using the *MAX4305* and *OPA657* op-amps. Note that the plots show frequency spectras, not time-domain signals and that white noise has a flat frequency spectrum.

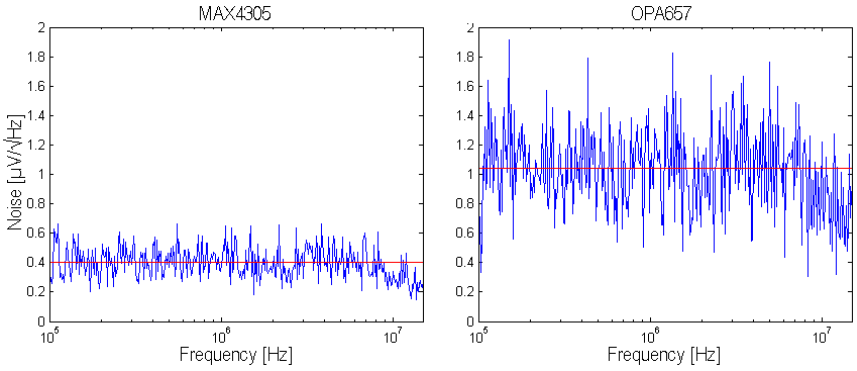

Figure 4.4: Noise measurements for instAmp evaluation version with MAX4305 and OPA657.

The noise level of the two miniaturized circuits was found to be approximately 400 and 1000 nV/$\sqrt{\text{Hz}}$ for the *MAX4305* and *OPA657* circuit, respectively. The non-miniaturized *MAX4304* showed a lot less noise, at approximately 80 nV/$\sqrt{\text{Hz}}$, Fig. 4.5

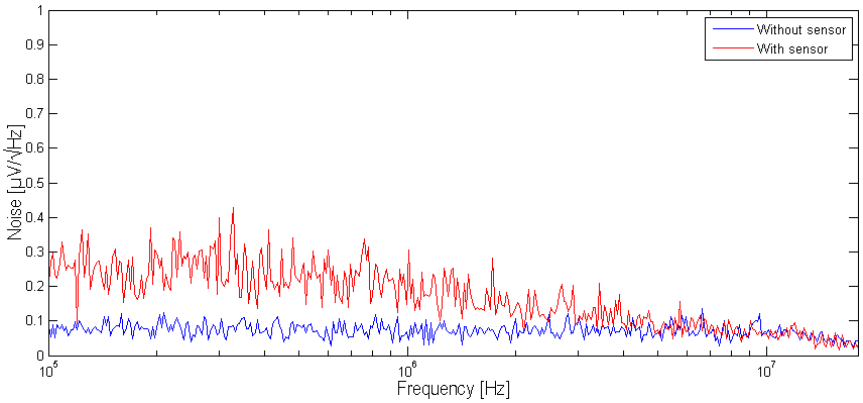

Figure 4.5: Noise measurements for instAmp v0.1 with and without sensor.

Chapter 5

Discussion

Hysteresis sweep

As a comparison to the results of the measured hysteresis sweep (Fig. 4.1, a plot of output voltage versus magnetic flux density (for various temperatures), found in the *AAL002* sensor datasheet [9], is shown in Fig. 5.1. The performed hysteresis sweeps corresponds well with the ones in the datasheet, and show that the instAmp v1.0 circuit is linear.

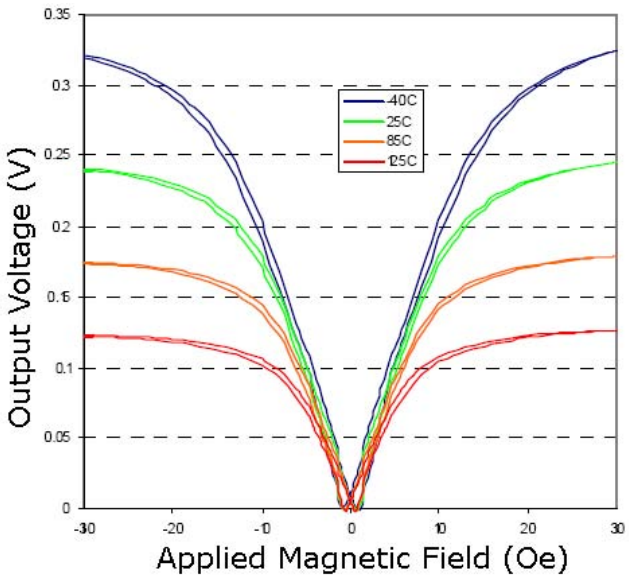

Figure 5.1: Plot of the sensor output voltage as a function of magnetic field density, at different temperatures. (From the AAL002 datasheet [9].)

Low-frequency AC measurement

Due to the coil's high impedance at 10 kHz, the magnetic field produced is weak, and the sensor output only 3-4 mV. This is near the resolution limit of the oscilloscope, which causes the probed signal to suffer quite heavily from quantization errors. This is also seen in the frequency domain as quantization noise. The amplified output of the instAmp circuit is better adapted to the oscilloscope's dynamic range and the effect of quantization noise is negligible. This is a good example showing the importance of amplifying the signal.

Bandwidth measurement

With a -3 dB bandwidth of approximately 20 MHz, the performance of the *MAX4305* circuit is clearly dissapointing. Compared with results from simulations done in ADS, Fig. 5.2, the measured bandwidths are substantially lower than those simulated. The reason for the poor bandwidths can

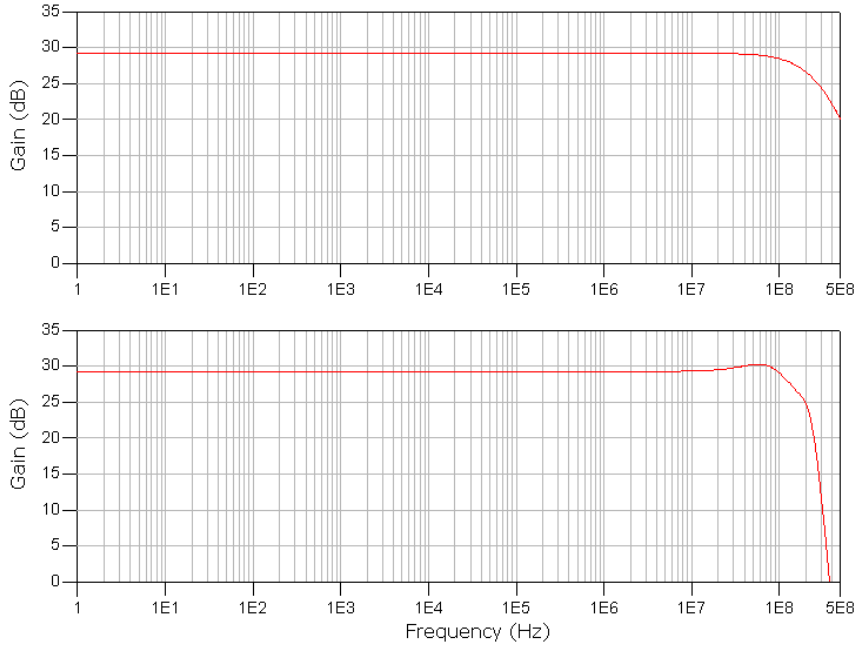

Figure 5.2: ADS Bandwidth simulations for MAX4305 (top) and OPA657 (bottom).

be partly explained by oscillations and EMC issues. For instance, drawing signal traces close to each other, and not shielding them properly, causes coupling that can result in oscillations. Using multi-layer PCBs with signal layers embedded between ground layers should greatly improve EMC characteristics and, in turn, increase the bandwidth. Both miniaturized circuits

(*MAX4305* and *OPA657*) show some interesting peaks at 90-110 MHz which are believed to be caused by electromagnetic interference from Swedish radio stations. Since the cables connecting the NSA to the PCB are unshielded, they act as antennas and are very susceptible to EMI. In retrospect, Subminiature version A (SMA) connectors and coaxial cables should have been used.

Fortunately, the unexpectedly low bandwidths of the circuits do not pose a big threat to the overall performance of the measurement system. As the maximum sample rate of the available ADC EVM is 125 Msps, the Nyquist frequency (f_N) becomes $125/2 = 62.5$ MHz, marked as a dotted gray line in Fig. 4.3. Both the *MAX4304* and *OPA657* circuit amplify frequencies up to f_N by more than 20 dB and will perform well with the components of the current measurement system.

Noise measurement

The noise levels of the two miniaturized circuts are very much higher than that of the non-minaturized *MAX4304* reference circuit. This can, to a large extent, be explained by the oscillation and EMI problems of the miniaturized circuits.

Luckily, the noise characteristics of the *MAX4304* circuit looks a lot more promising. According to table 2.2 in section 2.2.4, the voltage noise density of the individual op-amps is 2.1 and 2.8nV/$\sqrt{\text{Hz}}$ for the *MAX430x* and *OPA657*, respectively. Since the circuits are configured to have a high gain, disturbances will be amplified by a great deal. In fact, the output noise of the instrumentation amplifier ($e_{n_{out}}$) is proportional to the noise of the op-amp (e_n) and the total gain G,

$$e_{n_{(out)}} \approx G e_n, \tag{5.1}$$

given that the op-amp's voltage noise density e_n is much greater than its current noise density i_n [14]. The *MAX4304* reference circuit has a total gain of about 30-35, which in theory should yield a 63-73 nV/$\sqrt{\text{Hz}}$ voltage noise density. Compared with this, the measured 80 nV/$\sqrt{\text{Hz}}$ looks a lot more reasonable.

Due to limitations in the spectrum analyzer, the frequency spectras look noisy. Another spectrum analyzer (HP 3585A) used for measurements produced far smoother (whiter) noise spectras. Unfortunately, this analog spectrum analyzer lacked the functionality of storing digital data.

Chapter 6

Conclusions

The analog circuits developed work well in the 0-60 MHz region. With the acquired ADC and DAQ module, the measurement system can amplify and sample frequencies up to 60 MHz with an accuracy of 14 bits. Although simulations promise far greater bandwidths, real-life issues such as non-ideal components, interference of signal traces and the use of unshielded cables all degrade the frequency response characteristics. It is believed that the obtained bandwidth of 60 MHz is fairly close to what is feasible for a two-layer PCB.

The digital subsystem (ADC, DAQ and the developed software) works very well and provides the user with a simple, yet powerful, tool for sampling of high-frequency low-voltage electronic signals. This will, hopefully, prove to be useful not only for the SDTM project but also for future projects at ÅSTC.

From an electronics standpoint, miniaturization of the analog circuits did not prove to be very beneficial, rather the opposite, as some of the miniaturized circuits showed oscillatory behavior at high frequencies. However, since the ultimate goal is to integrate all electronics into a small chip, miniaturization will at some point become inevitable. In due time, it will be important to fully understand the basic principles of high-speed electronics design in order to avoid the many pitfalls associated therewith.

At the completion of this thesis, and although great progress has been made over the past months, the first SDTM sensor still remains to be manufactured. Hence, it is not yet certain if (and if so, how) the developed circuits will restrict the overall performance of the measurement system. If the sensor proves to have a very high bandwidth or exceptionally good noise characteristics, the analog circuitry will have to be improved further. Depending on the requirements, the following actions can, and should, be taken:

Higher bandwidth
For higher bandwidth, multi-layer PCBs are required. Additional copper layers allow signal traces to be properly shielded and ground-planes to be kept intact. Furthermore, it makes component placement and trace routing much less troublesome. As multi-layer PCBs are quite expensive, especially when produced in small series, the cost must be weighted against the possible gain in performance.

Lower noise
Higher bandwidth implies more noise. Hence, the easiest way to reduce noise is simply to lower the system's bandwidth. The instrumentation amplifier should be configured so that it has roughly the same bandwidth as the SDTM sensor. Multi-layer PCBs should also result in lower noise levels, mainly due to better shielding of highly sensitive signal traces.

Final notes and considerations for the end product

The developed measurement system is mainly aimed for evaluation of the soon-to-be-manufactured SDTM sensor. As such, the system contains most of the vital parts but has to be thorougly revised before it is ready to operate in space.

The SDTM sensor, analog circuits and ADC could be integrated on a single silicon chip, resulting in a highly sensitive and high-bandwidth magnetometer IC with digital output. Since the IC will produce a high-speed parallell stream of digital signals, some kind of FPGA or Application-Specific IC (ASIC) and high-speed memory will be required. Depending on the application, the FPGA/ASIC should also be configured to perform necessary digital signal processing, reducing the amount of data that has to be stored.

Acknowledgments

The following have assisted in the completion of this thesis.

At the Ångström Space Technology Center, ÅSTC,

Anders Persson, *Ph.D. Student*
Hugo Nguyen, *Assistant Professor*
Greger Thornell, *Associated Professor*
Henrik Kratz, *Assistant Professor*

At the Swedish Institute of Space Physics, IRF,

Lennart Åhlen, *Senior Research Engineer*
Sven-Erik Jansson, *Senior Research Engineer*
Walter Puccio, *Research Engineer*
Farid Shiva, *Engineer*

Sponsors,

VINNOVA, *the Swedish Governmental Agency for Innovation Systems*
Rymdstyrelsen, *the Swedish National Space Board*

Bibliography

[1] Anders Persson and Lukas Karlsson. SDTM-WP2: Requirements Specification, 2007.

[2] Lukas Karlsson. Development of electronics for ultra-broadband space magnetometry. Master's thesis, Luleå University of Technology, 2008.

[3] Douglas Hamilton and William Howard. *Basic Integrated Circuit Engineering*. McGraw-Hill, 1975.

[4] Charles Kitchin and Lew Counts. *A Designer's Guide to Instrumentation Amplifiers*. Analog Devices Inc., 3^{rd} edition, 2006.

[5] Walt Jung. *Op Amp Applications Handbook*. Newnes, 2004.

[6] Walt Kester. *Taking the Mystery out of the Infamous Formula, "SNR=6.02N + 1.76dB," and Why You Should Care*. Analog Devices Inc., 2005.

[7] Seagate Technology LLC. Seagate Cheetah 15K.5 datasheet, 2007.

[8] International Telecommunication Union. ITU-R Recommendation BS.1196-1, 2001.

[9] NVE Corporation. AA and AB-Series Analog Sensors datasheet, 2007.

[10] Lee W. Ritchey. *Right the first time - A practical handbook on high speed PCB and system design*. Speeding Edge, 2003.

[11] John Ardizzoni. A practical guide to high-speed pcb layout. *Analog Dialogue vol. 39*, 2005.

[12] Maxim IC. MAX4104/MAX4105/MAX4304/MAX4305 datasheet, 2007.

[13] Texas Instruments. OPA657 datasheet, 2006.

[14] Glen Brisebois. *Op Amp Selection Guide for Optimum Noise Performance*. Linear Technology, 2005.

Appendices

Appendix A

Survey on Magnetoresistive Magnetometers for Space

This is an excerpt from the article *Survey on Magnetoresistive Magnetometers for Space*, written by Anders Persson, Greger Thornell and Hugo Nguyen at the Ångström Space Technology Centre.

B. Anisotropic Magnetoresistance

An AMR sensor simply consists of a FM conductor, often patterned into a strip [46]. The direction of the magnetization in the strip will tend to align with the direction of an applied magnetic field. If a voltage is applied along the strip, the resistance will vary with the rotation of the magnetic field. This is illustrated in Fig. 1. Maximum resistance is reached when the field, and thus the magnetization, is aligned perpendicular to the voltage field, and minimum when parallel. The physical origin of this effect is still debated.

The so called magnetoresistance (MR) is one of the most important properties, when characterizing magnetoresistive sensors. It is defined as

$$MR = \frac{R_{\uparrow\uparrow} - R_{\uparrow\downarrow}}{R_{\uparrow\downarrow}} \tag{1}$$

where $R_{\uparrow\downarrow}$ is the maximum and $R_{\uparrow\uparrow}$ the minimum resistance of the device respectively [47]. The MR of all magnetoresistive sensors is closely related to the sensitivity, where a high MR typically corresponds to high sensitivity. The MR in AMR sensors is strongly temperature dependent with a maximum near the Curie temperature [48, 49]. Typical MR values for AMR sensors are in the order of 1−5%. The relatively low sensitivity and strong temperature dependence make AMR sensors unsuitable for scientific applications but feasible for attitude control. An attitude control sensor, based on AMR, is currently being developed by LUSOSPACE for the Astrium AEOLUS satellite[1], with launch scheduled to 2009. The Lusospace magnetometer has a field range of 70 μT, a resolution of 0.1 μT, a noise level of 40 nT, and a bandwidth of 40 Hz. This makes it comparable to most fluxgate attitude control sensors, but with a system mass of 300 g and good opportunity for further mass reduction, the Lusospace sensor should turn out to be very competitive.

It could also be noted that the structure of an AMR is very similar to that of an EHE sensor. The difference is that the signal is measured perpendicularly to the conductor in the latter case. Nevertheless the possibility of a mixed AMR-EHE sensor should be investigated.

C. Giant Magnetoresistance

GMR sensors consist of two FM electrode layers separated by a thin spacer layer of a non-magnetic metal [41]. The layers are then patterned, and a voltage is applied either along all the layers, making it a current-in-plane (CIP) GMR sensor, or perpendicularly to the layers, making it a current-perpendicular-to-plane (CPP)

[1] Vieira, Y., Martins, M. and Parracho, J., "Magnetoresistive sensors for a magnetometer", European Space Components Information Exchange System, [online database], https://escies.org/GetFile?rsrcid=1057, [Cited 2008-06-03]

GMR sensor [50]. The two types of GMR sensors are illustrated in Fig. 2 and 3. The most common shape of a GMR sensor is a strip, but also meander or yoke shapes are used, having favourable performance in broadband applications [51].

Typically, the FM layers are different alloys of Ni, Co and Fe, and the space is often made of Cu. Examples of different GMR structures are presented in Table 1. The choice of electrode material depends on the desired performance. In order to achieve a magnetoresistive effect, the properties of the two FM layers have to be different. Ideally, one of them should be influenced by the surrounding magnetic field while the other one stays unaffected. The first layer is then called the sensing layer (or the free layer), and the second is called the reference layer. The separation of the magnetic properties can be achieved either by choosing two FM materials with different coercivity, or by pinning one of them with an additional antiferromagnetic (AFM) layer.

In the first case, the layer with low coercivity (sensing layer) will react more easily to an external field than the layer with high coercivity (reference layer) [44]. Ni and Fe typically have low coercivity and are therefore called soft FM materials, while Co has a high coercivity, and consequently is called hard. The problem with the hard/soft FM configuration is that antiparallel magnetization, which represents the bias point of the sensor, occurs only at non-zero external magnetic field. This is not desirable in scientific applications. The behaviour of different FM layers in an external magnetic field is presented in Fig. 4.

Pinning, on the other hand, uses the fact that an exchange anisotropy, or exchange bias, will form in the interface between a FM and an AFM layer. The exchange anisotropy will shift the magnetic field strength that will reverse the magnetization of the FM layer along the x axis in Fig. 4 [44, 52]. Now, since the magnetization of the reference layer will remain constant in fields of less strength than the exchange anisotropy, the layer is said to be pinned. Examples of AFM materials used for pinning are IrMn and CrPtMn [53, 54].

The pinning effect is only present at the interface, and it is therefore important that the exchange anisotropy is strong enough to control the whole reference layer. The exchange anisotropy can be enhanced by optimizing the thicknesses and compositions of the pinning structure, or by using a synthetic antiferromagnetic (SAF) structure [55]. Here, the exchange anisotropy from the FM-AFM interface is amplified by the strong exchange coupling between two ferromagnets over a thin non-magnetic layer of Ru [56]. It will typically take a field of a couple of kOe to reverse the SAF reference layer. A typical SAF structure is FM-Ru-FM-AFM [57].

In addition to the improved pinning, the magnetization vectors of the two FM layers in the SAF structure will be antiparallel, due to the exchange coupling [55]. This will reduce the total magnetic moment of the reference

layer, thus reducing its magnetic fringe field impinging on the sensing layer. Therefore, the dynamic behaviour of the SAF structure turns out to be very useful when tuning the sensor performance.

Both when using a hard/soft FM configuration or a pinning structure, the two FM layers react differently to an applied magnetic field [42]. This is expressed by a difference in the orientation of their individual magnetization, and this difference will affect the conduction in the GMR sensor. When the magnetization vectors of the FM layers are parallel, electrons can move relatively unperturbed though the sensor, but when the vectors are antiparallel, vertical conduction will be hindered by scattering between electrons with different spin orientation. This behaviour will cause the magnetoresistive effect.

The MR of GMR sensors is typically in the order of 5−10%, although values up to 20 % have been observed [42]. The temperature dependence is much less pronounced than in AMRs. Most GMR sensors operate between −50˚C and 150˚C [54, 58]. They also have a reasonably low resistance, in the order of 10 Ω, enabling broadband sensors [59]. The output of a GMR sensor is typically linear in the range of ±20 Oe and the sensitivity is in the order of mV/VOe, which must be considered fairly high [60]. Thus, they are practical for intermediate to low field measurements.

Although the GMR development has been focused mostly on read-head and MRAM applications, they can be found in many different systems. For example, GMR sensors have been used for sensing the rotation of robotics arm or the position of pistons in a cylinder [54]. Highly sensitive sensors can also be used for detecting cracks in printed circuit boards via eddy-currents [61, 62], or molecules marked with magnetic beads in bio-sensors [59, 63].

The overall performance of the GMR sensor makes it a good candidate for spaceborne magnetic field measurements. The main advantage is the possibility of broadband measurements. GMR sensors can manage a 3 dB bandwidth from 1 Hz up to 1 GHz, but the sensitivity at both high and low frequencies will be reduced by intrinsic noise. The sensor performance at low frequencies and fields may therefore not be satisfactory.

D. Magnetic Tunnel Junctions

The structure of a MTJ has much in common with a GMR sensor. The main difference is that the non-magnetic conducting layer is replaced by a dielectric tunnelling barrier. The voltage is applied over the barrier in the CPP mode, which is illustrated in Fig. 3.

Even though no current will flow in a macroscopic version of such a device, electrons will start to tunnel if the barrier is made extremely thin (≤10 Å). The origin of the magnetoresistive effect, called Tunnelling Magnetoresistance (TMR), is similar to that of GMR. The spin of the valence electrons in the sensing layer will

align with the external field, and the voltage will cause them to tunnel [44]. Given a smooth and thin barrier, the spin will be conserved during tunnelling. The reference layer will have electron vacancies near the electrode-barrier interface, due to the applied voltage. These vacancies will prefer tunnelling by electrons with a spin directed along the pinning direction of the reference layer. Tunnelling will therefore be more probable if the sensing layer is aligned along the pinning vector. The relationship between the relative magnetizations of the sensing and reference layers, and the conductance can be found by solving Schrödinger's equation for the tunnelling electrons:

$$G(\theta) = G_0\left(1 + P_S P_R \cos(\theta)\right) \tag{2}$$

where, G_0 is the conductance at perpendicular alignment, P_S and P_R are the effective spin polarizations of the sensing and reference layers respectively, and θ is the angle between the magnetization of the two layers. Hence, the conductance is a linear function of cosine. Furthermore, the conductance is approximately linearly dependent on the field strength of a static external field. These fairly uncomplicated spatial dependencies prove the MTJ to be a good vector magnetometer.

Usually it takes three MTJ sensor elements in order to form a 3-axis magnetometer. However, Deak et al. have shown that it is possible to measure the field in two dimensions using only one MTJ [64]. This is accomplished by biasing the sensing layer with a triangular wave along the easy axis of the magnetization of the sensing layer, and a square wave along the hard axis (perpendicular to the easy axis). Unfortunately the frequency of the bias waves will limit the bandwidth to a couple of Hz.

The TMR effect gives rise to very large MR. It is commonly in the range of 20−500 % at room temperature [41, 42, 65]. Thanks to the high MR, MTJs can be sensitive down to the pT region (μOe) [66], and they also show very good temperature stability. For example, the MTJ made by Parkin *et al.* only suffered a 3 dB reduction of MR, when going from 4 K to room temperature [67]. This performance makes MTJs ideal for low field measurements.

It should be noted that MTJs have a relatively high resistance area product (RAP) compared to the other magnetoresistive sensor families. The RAP corresponds to the resistance of a sensor segment with an area of 1 μm^2, and RAPs between 10 $k\Omega\times\mu m^2$ and 100 $k\Omega\times\mu m^2$ are common for MTJs [55, 68, 69]. This may turn out to be problematic in a highly sensitive and broadband sensor, since high resistance typically causes increased noise, and it is therefore preferable to have a RAP lower than 5 $\Omega\times\mu m^2$ in these applications. In the past it has been difficult to accomplish low RAPs with preserved MR. Tsunekawa et al. have been able to make an MTJ with an MR of 138 % at room temperature with a RAP of just 2.4 $\Omega\times\mu m^2$ [70]. This was done by using a CoFeB(30)-

Mg(4)-MgO(8.5)-CoFeB(30)-Ru(8.5)-CoFe(25)-PtMn(150) structure (numbers in parenthesis are thicknesses in Å). By this, it should be possible to make a highly miniaturized magnetoresistive sensor comparable to the best conventional magnetometers.

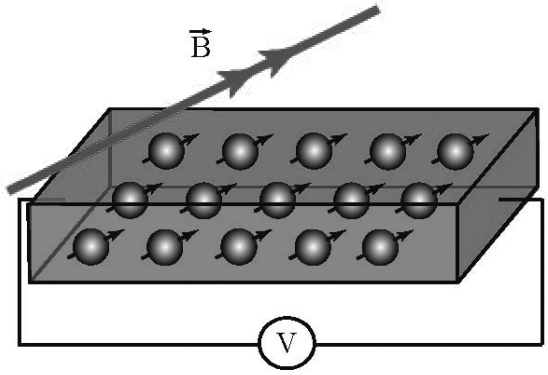

Fig. 1 The relation between the individual magnetic moments, external magnetic field, and bias voltage in an AMR sensor.

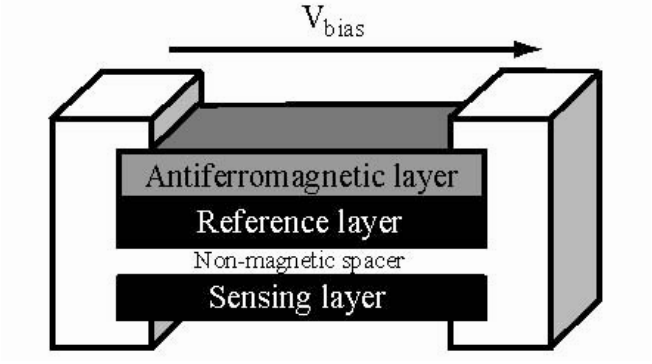

Fig. 2 In-plane biased GMR sensor.

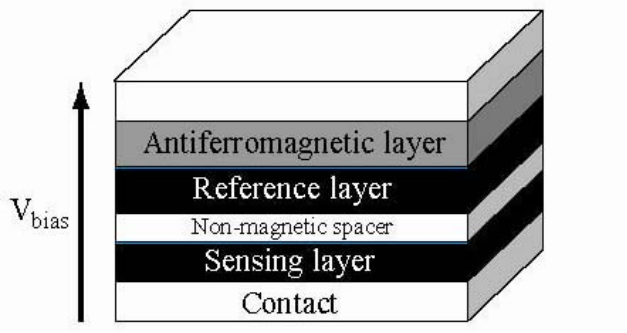

Fig. 3 GMR sensor connected in the CPP mode.

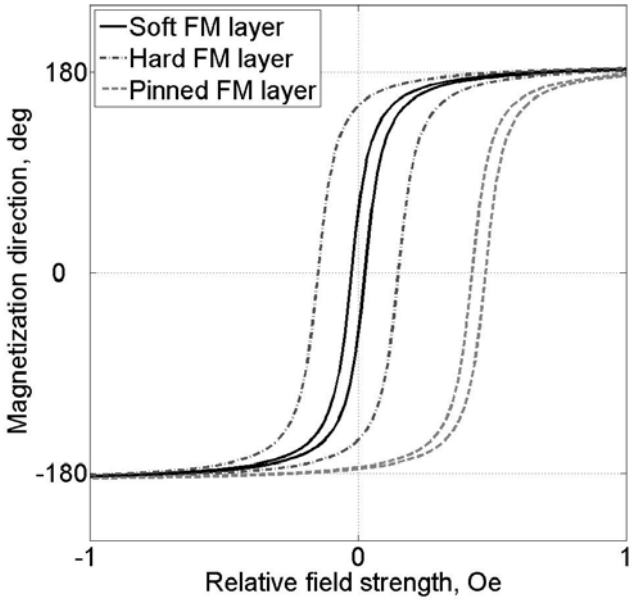

Fig.4 Hysteresis curves of different FM layers.

Appendix B

DAQ communication program source code

The following pages presents the source code for the small binary file that handles communication with the TSW1100 board. The program was written in, and should prefferably be compiled with, *Microsoft Visual C++*.

```
#include "stdafx.h"
#include "windows.h"
#include "TSW1100Api.h"
#include <stdio.h>

#define LOCALBUFFERMAX 4096

typedef struct tChannelContext {
  int     channelid ;
  USHORT channelcontrol ;
  USHORT triggercontrol ;
  BOOL    bOutputFileUsed ;
  FILE    *logFile ;
  ULONG   lBytes ;
  ULONG   lStartAddress ;
  ULONG   lBufferSize ;
  ULONG   lPtrAddress ;
  ULONG   iInterimBytes ;
  int     iReadBufferFlag ;
  BOOL    bCounterEnable ;
  BOOL    bFifoEnable ;
  int     rbuf[LOCALBUFFERMAX] ;
  int     nBytes ;
  int     i ;
  int     iStatus ;
  int     iOldStatus ;
  BOOL    bClockPresent ;
  int     ClockOffset ;
  int     UpdateStatus ;
  char   fileName[40] ;
  char   file[40] ;
} sChannelContext ;

HINSTANCE LoadDLL (void) {
  LPVOID lpMsgBuf;
  HINSTANCE hinstLib ;
  hinstLib = LoadLibrary("tsw1100.dll") ;
  if (hinstLib == NULL) {
    if (FormatMessage(
          FORMAT_MESSAGE_ALLOCATE_BUFFER |
          FORMAT_MESSAGE_FROM_SYSTEM |
          FORMAT_MESSAGE_IGNORE_INSERTS,
          NULL,
          GetLastError(),
          MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
          (LPTSTR) &lpMsgBuf,
          0,
          NULL )) {

      // Display the string.
      MessageBox( NULL, (LPCTSTR)lpMsgBuf, "Error", MB_OK |
          MB_ICONINFORMATION );
    }
```

```
    // Free the buffer.
    LocalFree( lpMsgBuf );

    // Indicate error
    printf(" ERROR: Unable to find dll\n") ;
    return NULL ;
  }
  else
    return hinstLib ;
}

void UnLoadDLL (HINSTANCE hinstLib) {
  BOOL fFreeResult ;
  LPVOID lpMsgBuf;

  // Unload TSW1100 EVM DLL
  fFreeResult = FreeLibrary(hinstLib) ;
  if (fFreeResult == NULL)   {
    if (FormatMessage(
        FORMAT_MESSAGE_ALLOCATE_BUFFER |
        FORMAT_MESSAGE_FROM_SYSTEM |
        FORMAT_MESSAGE_IGNORE_INSERTS,
        NULL, GetLastError(),
        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
        (LPTSTR) &lpMsgBuf, 0, NULL )) {

      // Display the string.
      MessageBox( NULL, (LPCTSTR)lpMsgBuf, "Error", MB_OK |
          MB_ICONINFORMATION );
    }
    // Free the buffer.
    LocalFree( lpMsgBuf );
  }
}

//For future use
void periodicTickCh1 (void) { ; }
void periodicTickCh2 (void) { ; }

int _tmain(int argc, _TCHAR* argv[]) {
  // Hold pointers to the TSW1100 EVM DLL APIs and handle
  HINSTANCE     hinstLib ;
  TSW1100INIT    fpInit ;
  TSW1100CONTROL fpControl ;
  TSW1100STATUS  fpStatus ;
  TSW1100START   fpStart ;
  TSW1100STOP    fpStop ;
  TSW1100READ    fpRead ;
  TSW1100WRITE   fpWrite ;
  TSW1100VERSION fpVersion ;
  TSW1100CLOCKSTAT fpClockStatus ;
  TSW1100TESTMODE fpTestMode ;
  TSW1100LEDCONTROL fpLedControl ;
  TSW1100CONTROLBUFFER fpControlBuffer ;
```

```
TSW1100FIRMWAREUPDATE fpFirmwareUpdate ;
TSW1100GETCLOCKOFFSET fpGetClockOffset ;
TSW1100SETCLOCKOFFSET fpSetClockOffset ;
TSW1100SETCALLBACK    fpSetCallback ;
TSW1100FPGAUPDATE     fpFpgaUpdate ;

// Variables to store command-line switches
sChannelContext ChannelCommandLine[2] ;
int  iChannelSel ;
int  chanSel;
BOOL fRunTimeLinkSuccess = FALSE ;
int  status ;
char* SuccessStrings[5] ;
int i ;
int Offset ;
int nbrSamples = 1  ;
int nbrSamplesTaken = 0;

// Initialize variables
SuccessStrings[TSW1100_STATUSERROR+1] = "STATUSERROR" ;
SuccessStrings[TSW1100_STATUSIDLE+1] = "STATUSIDLE" ;
SuccessStrings[TSW1100_STATUSARMED+1] = "STATUSARMED" ;
SuccessStrings[TSW1100_STATUSBUSY+1] = "STATUSBUSY" ;
SuccessStrings[TSW1100_STATUSDONE+1] = "STATUSDONE" ;

ChannelCommandLine[0].channelid = TSW1100_CHANNEL1 ;
ChannelCommandLine[0].bCounterEnable = FALSE ;
ChannelCommandLine[0].bFifoEnable    = FALSE ;
ChannelCommandLine[0].bOutputFileUsed = FALSE ;
ChannelCommandLine[0].channelcontrol = TSW1100_RESETCHANNEL ;
ChannelCommandLine[0].iInterimBytes = 0 ;
ChannelCommandLine[0].lStartAddress = 0x0 ;
ChannelCommandLine[0].lBufferSize = 0x800000 ;
ChannelCommandLine[0].lPtrAddress = 0x0 ;
ChannelCommandLine[0].lBytes = 32 ;
ChannelCommandLine[0].fileName ;
ChannelCommandLine[0].triggercontrol =
    TSW1100_TRIGGERINTERNALSELECT ;
ChannelCommandLine[0].iStatus = TSW1100_STATUSIDLE ;
ChannelCommandLine[0].iOldStatus = TSW1100_STATUSIDLE ;
ChannelCommandLine[0].bClockPresent = FALSE ;
ChannelCommandLine[0].ClockOffset = TSW1100_CLOCKOFFSETMIN ;
ChannelCommandLine[0].UpdateStatus = 64 ;

ChannelCommandLine[1].channelid = TSW1100_CHANNEL2 ;
ChannelCommandLine[1].bCounterEnable = FALSE ;
ChannelCommandLine[1].bFifoEnable    = FALSE ;
ChannelCommandLine[1].bOutputFileUsed = FALSE ;
ChannelCommandLine[1].channelcontrol = TSW1100_RESETCHANNEL ;
ChannelCommandLine[1].iInterimBytes = 0 ;
ChannelCommandLine[1].lStartAddress = 0x800000 ;
ChannelCommandLine[1].lBufferSize = 0x800000 ;
ChannelCommandLine[1].lPtrAddress = 0x800000 ;
ChannelCommandLine[1].lBytes = 32 ;
```

```
ChannelCommandLine[1].fileName ;
ChannelCommandLine[1].triggercontrol =
    TSW1100_TRIGGERINTERNALSELECT ;
ChannelCommandLine[1].iStatus = TSW1100_STATUSIDLE ;
ChannelCommandLine[1].iOldStatus = TSW1100_STATUSIDLE ;
ChannelCommandLine[1].bClockPresent = FALSE ;
ChannelCommandLine[1].ClockOffset = TSW1100_CLOCKOFFSETMIN ;
ChannelCommandLine[1].UpdateStatus = 64 ;


// Parse command-line switches
for (i=0; i<argc; i++) {
  if (strcmp(argv[i],"-ch") == 0) {
    if ((strcmp(argv[i+1],"0") == 0) ||
        (strcmp(argv[i+1],"1") == 0)) {
      iChannelSel = 0 ;
  chanSel = 1;
}
  else {
       iChannelSel = 1 ;
    chanSel = 2;
  }
    ChannelCommandLine[iChannelSel].channelcontrol = (
        TSW1100_UNRESETCHANNEL | TSW1100_CHANNELINPUT) ;
    ChannelCommandLine[iChannelSel].bOutputFileUsed = TRUE ;
    i++ ;
  }
  else if (strcmp(argv[i],"-exttrig") == 0) {
    ChannelCommandLine[iChannelSel].triggercontrol =
        TSW1100_TRIGGEREXTERNALSELECT ;
  }
  else if (strcmp(argv[i],"-inttrig") == 0) {
    ChannelCommandLine[iChannelSel].triggercontrol =
        TSW1100_TRIGGERINTERNALSELECT ;
  }
  else if (strcmp(argv[i],"-bytes") == 0) {
    ChannelCommandLine[iChannelSel].lBytes = strtol(argv[i
        +1], NULL, 10) ;
  }
  else if ((strcmp(argv[i],"-buffsize") == 0) || (strcmp(argv
      [i],"-buff_size") == 0)) {
    ChannelCommandLine[iChannelSel].lBufferSize = strtol(argv
        [i+1], NULL, 10) ;
  }
  else if (strcmp(argv[i],"-cntren") == 0){
    ChannelCommandLine[iChannelSel].bCounterEnable = TRUE ;
  }
  else if (strcmp(argv[i],"-fifoen") == 0) {
    ChannelCommandLine[iChannelSel].bFifoEnable = TRUE ;
  }
  else if (strcmp(argv[i],"-clkdelay") == 0) {
    ChannelCommandLine[iChannelSel].ClockOffset = strtol(argv
        [i+1], NULL, 10) ;
    i++ ;
```

```
  }
  else if (strcmp(argv[i],"-status") == 0) {
    ChannelCommandLine[iChannelSel].UpdateStatus = strtol(
        argv[i+1], NULL, 10) ;
    i++ ;
  }
  else if (strcmp(argv[i],"-help") == 0) {
    printf("\n") ;
    printf("testfixture [OPTIONS]\n") ;
    printf("  -ch <num>     : Assigns following command-line
        switches\n") ;
    printf("                  to given channel\n") ;
    printf("  -exttrig      : Sets channel for external
        triggering\n") ;
    printf("                  MUTUALLY EXCLUSIVE with -inttrig
        \n") ;
    printf("  -inttrig      : Sets channel for internal
        triggering\n") ;
    printf("                  MUTUALLY EXCLUSIVE with -exttrig
        \n") ;
    printf("  -bytes <number>\n") ;
    printf("                : number (base 10) of bytes\n") ;
    printf("  -buffsize <number>\n") ;
    printf("                : number (base 10) of buffer size\
        n") ;
    printf("  -cntren       : Connects running counter to
        Channel\n") ;
    printf("  -fifoen       : Connects channel to fifo\n") ;
    printf("  -clkdelay <number>\n") ;
    printf("                : between 0 and 127\n") ;
    printf("  -status <number>\n") ;
    printf("                : update status every <number> of
        bytes\n") ;
    printf("  -help         : This message\n") ;
    printf("  -n <number>\n") ;
    printf("                : Number of measurements\n") ;
    printf("  -file <name>\n") ;
    printf("                : File to write to\n") ;
    printf("\n") ;
    return 0 ;
  }
else if (strcmp(argv[i], "-n") == 0) {
  nbrSamples = atoi(argv[i+1]);
  i++;
}
else if (strcmp(argv[i], "-file") == 0) {
  int n = sprintf(ChannelCommandLine[iChannelSel].file, "%s",
      argv[i+1]) ;
    i++ ;
    }
}

// Check to ensure bytes <= buffsize for each channel
for(iChannelSel = TSW1100_CHANNEL1; iChannelSel <=
```

```
    TSW1100_CHANNEL2; iChannelSel++) {
  if (ChannelCommandLine[iChannelSel].lBytes >
      ChannelCommandLine[iChannelSel].lBufferSize) {
  printf(" ERROR: Requested bytes %d for Channel %d is
      greater than buffer size %d specified.\n",
            ChannelCommandLine[iChannelSel].lBytes,
            iChannelSel+1,
            ChannelCommandLine[iChannelSel].lBufferSize) ;
    printf(" =    Changing requested number of bytes to be
        buffer size : %d\n",
            ChannelCommandLine[iChannelSel].lBufferSize) ;
    ChannelCommandLine[iChannelSel].lBytes =
        ChannelCommandLine[iChannelSel].lBufferSize ;
  }
}

// Load the DLL
hinstLib = LoadDLL() ;
if (hinstLib == NULL)
  return 0 ;

// Load DLL APIs for TSW1100 EVM
fpInit    = (TSW1100INIT)    GetProcAddress(hinstLib, "
    TSW1100_init") ;
fpControl = (TSW1100CONTROL) GetProcAddress(hinstLib, "
    TSW1100_control") ;
fpStatus  = (TSW1100STATUS)  GetProcAddress(hinstLib, "
    TSW1100_status") ;
fpStart   = (TSW1100START)   GetProcAddress(hinstLib, "
    TSW1100_start") ;
fpStop    = (TSW1100STOP)    GetProcAddress(hinstLib, "
    TSW1100_stop") ;
fpRead    = (TSW1100READ)    GetProcAddress(hinstLib, "
    TSW1100_readData") ;
fpWrite   = (TSW1100WRITE)   GetProcAddress(hinstLib, "
    TSW1100_writeData") ;
fpVersion = (TSW1100VERSION) GetProcAddress(hinstLib, "
    TSW1100_getVersion") ;
fpClockStatus = (TSW1100CLOCKSTAT)GetProcAddress(hinstLib, "
    TSW1100_clockStatus") ;
fpTestMode= (TSW1100TESTMODE)GetProcAddress(hinstLib, "
    TSW1100_testMode") ;
fpLedControl=(TSW1100LEDCONTROL)GetProcAddress(hinstLib, "
    TSW1100_ledControl") ;
fpControlBuffer=(TSW1100CONTROLBUFFER) GetProcAddress(
    hinstLib, "TSW1100_controlBuffer") ;
fpFirmwareUpdate=(TSW1100FIRMWAREUPDATE) GetProcAddress(
    hinstLib, "TSW1100_firmwareUpdate") ;
fpGetClockOffset=(TSW1100GETCLOCKOFFSET) GetProcAddress(
    hinstLib, "TSW1100_getClockOffset") ;
fpSetClockOffset=(TSW1100SETCLOCKOFFSET) GetProcAddress(
    hinstLib, "TSW1100_setClockOffset") ;
fpSetCallback   =(TSW1100SETCALLBACK) GetProcAddress(hinstLib
    , "TSW1100_setCallback") ;
```

```
fpFpgaUpdate    =(TSW1100FPGAUPDATE) GetProcAddress(hinstLib,
    "TSW1100_fpgaUpdate") ;

// Check if all DLL APIs were loaded
if ((fpInit != NULL) && (fpStatus != NULL) && (fpControl !=
    NULL) &&
    (fpStart != NULL) && (fpStop != NULL) && (fpRead != NULL)
        &&
    (fpWrite != NULL) && (fpVersion != NULL) && (
        fpClockStatus != NULL) &&
    (fpLedControl != NULL) && (fpControlBuffer != NULL) &&
    (fpFirmwareUpdate != NULL) &&
    (fpGetClockOffset != NULL) &&
    (fpSetClockOffset != NULL) &&
    (fpSetCallback != NULL) &&
    (fpFpgaUpdate != NULL) &&
    (fpTestMode != NULL)) {
  fRunTimeLinkSuccess = TRUE ;
}
else
  printf(" ERROR: Unable to Load DLL APIs\n") ;
if(fpInit == NULL)
  printf(" ERROR: Unable to load TSW1100_init\n") ;
if(fpStatus == NULL)
  printf(" ERROR: Unable to load TSW1100_status\n") ;
if(fpControl == NULL)
  printf(" ERROR: Unable to load TSW1100_control\n") ;
if(fpStart == NULL)
  printf(" ERROR: Unable to load TSW1100_start\n") ;
if(fpStop == NULL)
  printf(" ERROR: Unable to load TSW1100_stop\n") ;
if(fpRead == NULL)
  printf(" ERROR: Unable to load TSW1100_readData\n") ;
if(fpWrite == NULL)
  printf(" ERROR: Unable to load TSW1100_writeData\n") ;
if(fpVersion == NULL)
  printf(" ERROR: Unable to load TSW1100_getVersion\n") ;
if(fpClockStatus == NULL)
  printf(" ERROR: Unable to load TSW1100_clockStatus\n") ;
if(fpLedControl == NULL)
  printf(" ERROR: Unable to load TSW1100_ledControl\n") ;
if(fpControlBuffer == NULL)
  printf(" ERROR: Unable to load TSW1100_controlBuffer\n") ;
if(fpFirmwareUpdate == NULL)
  printf(" ERROR: Unable to load TSW1100_firmwareUpdate\n") ;
if(fpGetClockOffset == NULL)
  printf(" ERROR: Unable to load TSW1100_getClockOffset\n") ;
if(fpSetClockOffset == NULL)
  printf(" ERROR: Unable to load TSW1100_setClockOffset\n") ;
if(fpSetCallback == NULL)
  printf(" ERROR: Unable to load TSW1100_setCallback\n") ;
if(fpTestMode == NULL)
  printf(" ERROR: Unable to load TSW1100_testMode\n") ;
if(fpFpgaUpdate == NULL)
```

```
  printf(" ERROR: Unable to load TSW1100_fpgaUpdate\n") ;

// Run application
if (fRunTimeLinkSuccess == TRUE) {
  //Check clock status
  (fpClockStatus)(&status) ;
  if((status & TSW1100_CHANNEL1CLOCKON) ==
      TSW1100_CHANNEL1CLOCKON) {
    ChannelCommandLine[TSW1100_CHANNEL1].bClockPresent = TRUE
        ;
  }
  else {
    if (chanSel == 1) {
  printf("ERROR: Channel 1 clock not present\n") ;
  return 1 ;
    }
  }
  if((status & TSW1100_CHANNEL2CLOCKON) ==
      TSW1100_CHANNEL2CLOCKON) {
    ChannelCommandLine[TSW1100_CHANNEL2].bClockPresent=TRUE ;
  }
  else {
    if (chanSel == 2) {
      printf("ERROR: Channel 2 clock not present\n") ;
      return 1 ;
    }
  }

  // Setup Clock Offset for each channel
  for(iChannelSel = TSW1100_CHANNEL1; iChannelSel <=
      TSW1100_CHANNEL2; iChannelSel++){
    // Read current clock offset
    status = (fpGetClockOffset)(&Offset, iChannelSel) ;

    // Set clock offset
    status = (fpSetClockOffset)(&ChannelCommandLine[
        iChannelSel].ClockOffset,
              ChannelCommandLine[iChannelSel].channelid) ;
    if (status == TSW1100_UNSUCCESSFUL) {
      printf(" ERROR: Unable to change TSW1100 channel %d
          clock offset\n", (iChannelSel+1)) ;
    }
  }

  while (nbrSamplesTaken < nbrSamples) {
    //Send reset to TSW1100 EVM
    status = (fpInit)() ;
    if(status == TSW1100_UNSUCCESSFUL) {
      printf(" ERROR: TSW1100 EVM not connected\n") ;
      UnLoadDLL(hinstLib) ;
      return 0 ;
    }
    status = (fpSetCallback)((TSW1100CALLBACK)&
        periodicTickCh1,
```

```
            ChannelCommandLine[TSW1100_CHANNEL1].
                UpdateStatus,
            TSW1100_CHANNEL1) ;
  status = (fpSetCallback)((TSW1100CALLBACK)&
      periodicTickCh2,
            ChannelCommandLine[TSW1100_CHANNEL2].
                UpdateStatus,
            TSW1100_CHANNEL2) ;

for(iChannelSel = TSW1100_CHANNEL1; iChannelSel <
    TSW1100_CHANNEL2; iChannelSel++) {
  // Read status of TSW1100 EVM Channel n
  status = (fpStatus)(iChannelSel) ;

  // Reset TSW1100 EVM Channel n configuration
  status = (fpControl)(TSW1100_RESETCHANNEL, NULL,
            ChannelCommandLine[iChannelSel].lBufferSize,
            ChannelCommandLine[iChannelSel].channelid);

  if (status == TSW1100_UNSUCCESSFUL) {
    printf(" ERROR: Unable to reset TSW1100 channel %d\n",
        (iChannelSel+1)) ;
  }

  if (ChannelCommandLine[iChannelSel].channelcontrol !=
      TSW1100_RESETCHANNEL) {
    // Take TSW1100 EVM Channel n out of reset
    status = (fpControl)(TSW1100_UNRESETCHANNEL, NULL,
              ChannelCommandLine[iChannelSel].lBufferSize,
              ChannelCommandLine[iChannelSel].channelid);

    if (status == TSW1100_UNSUCCESSFUL) {
      printf(" ERROR: Unable to take TSW1100 channel %d out
          of reset\n", (iChannelSel+1)) ;
    }

    // Configure TSW1100 EVM Channel n
    status=(fpControl)
           (ChannelCommandLine[iChannelSel].channelcontrol,
            ChannelCommandLine[iChannelSel].triggercontrol,
            ChannelCommandLine[iChannelSel].lBufferSize,
            ChannelCommandLine[iChannelSel].channelid);
    status=(fpControlBuffer)
           (ChannelCommandLine[iChannelSel].lStartAddress,
            ChannelCommandLine[iChannelSel].lBufferSize,
            ChannelCommandLine[iChannelSel].lPtrAddress,
            ChannelCommandLine[iChannelSel].channelid);

    if (status == TSW1100_UNSUCCESSFUL) {
      printf(" ERROR: Unable to configure TSW1100 channel %
          d\n", (iChannelSel+1)) ;
    }
```

```
    /* Setup for CNTR_ENABLE and FIFO_ENABLE mode.
       Connects ramp counter to Channel n instead of
       actual ADDA I/F. Puts data directly into FIFO,
       rather than out to SDRAM */
    status=(fpTestMode)
           (ChannelCommandLine[iChannelSel].bCounterEnable,
            ChannelCommandLine[iChannelSel].bFifoEnable,
            ChannelCommandLine[iChannelSel].channelid);
  }
  ChannelCommandLine[iChannelSel].iStatus =
     TSW1100_STATUSIDLE;
  ChannelCommandLine[iChannelSel].iOldStatus =
     TSW1100_STATUSIDLE;
  ChannelCommandLine[iChannelSel].iInterimBytes = 0;
}

//Open Log Files if used
for(iChannelSel = TSW1100_CHANNEL1; iChannelSel <=
    TSW1100_CHANNEL2; iChannelSel++) {
  if (ChannelCommandLine[iChannelSel].bOutputFileUsed ==
     TRUE) {
    int n = sprintf(ChannelCommandLine[iChannelSel].
        fileName, "%s.txt", ChannelCommandLine[iChannelSel
        ].file);
    if ((ChannelCommandLine[iChannelSel].logFile = fopen(
        ChannelCommandLine[iChannelSel].fileName, "w+" ))
        == NULL) {
      printf("ERROR: Can't open file %s\n",
          ChannelCommandLine[iChannelSel].fileName);
    }
  }
}

//while != reset and status != done
while (((ChannelCommandLine[TSW1100_CHANNEL1].
    channelcontrol != TSW1100_RESETCHANNEL) &&
(ChannelCommandLine[TSW1100_CHANNEL1].iStatus !=
    TSW1100_STATUSDONE)) ||
  ((ChannelCommandLine[TSW1100_CHANNEL2].channelcontrol !=
     TSW1100_RESETCHANNEL) &&
  (ChannelCommandLine[TSW1100_CHANNEL2].iStatus !=
     TSW1100_STATUSDONE))) {
  for(iChannelSel=TSW1100_CHANNEL1; iChannelSel<=
     TSW1100_CHANNEL2; iChannelSel++) {
    if (ChannelCommandLine[iChannelSel].channelcontrol !=
        TSW1100_RESETCHANNEL) {
      if (ChannelCommandLine[iChannelSel].iStatus ==
          TSW1100_STATUSIDLE) {
        if (ChannelCommandLine[iChannelSel].triggercontrol
            == TSW1100_TRIGGERINTERNALSELECT) {
          //Internal trigger; start measurement
          (fpStart)(ChannelCommandLine[iChannelSel].
              channelid) ;
        }
```

```
        else  { //Else; wait for trigger
          printf(" = Channel %d: External Trigger Select.
              Waiting for DONE...\n", (iChannelSel+1)) ;
        }
      }
      ChannelCommandLine[iChannelSel].iStatus = (fpStatus)(
          ChannelCommandLine[iChannelSel].channelid) ;
      if (ChannelCommandLine[iChannelSel].iStatus !=
          ChannelCommandLine[iChannelSel].iOldStatus) {
        ChannelCommandLine[iChannelSel].iOldStatus =
            ChannelCommandLine[iChannelSel].iStatus ;
      }
    }
  }
}
// Capture is complete!

for(iChannelSel=TSW1100_CHANNEL1; iChannelSel<=
    TSW1100_CHANNEL2; iChannelSel++) {
  if (ChannelCommandLine[iChannelSel].channelcontrol !=
      TSW1100_RESETCHANNEL) {
    // Read data
    ChannelCommandLine[iChannelSel].iReadBufferFlag =
        TSW1100_STARTOFBUFFER ;

    while (ChannelCommandLine[iChannelSel].iInterimBytes <
        ChannelCommandLine[iChannelSel].lBytes) {
      if ((ChannelCommandLine[iChannelSel].lBytes -
          ChannelCommandLine[iChannelSel].iInterimBytes) >
          LOCALBUFFERMAX)
        ChannelCommandLine[iChannelSel].nBytes =
            LOCALBUFFERMAX ;
      else
        ChannelCommandLine[iChannelSel].nBytes=(
            ChannelCommandLine[iChannelSel].lBytes -
            ChannelCommandLine[iChannelSel].iInterimBytes);

      status=(fpRead)
          (&ChannelCommandLine[iChannelSel].rbuf[0],
        ChannelCommandLine[iChannelSel].nBytes,
        ChannelCommandLine[iChannelSel].iReadBufferFlag,
        ChannelCommandLine[iChannelSel].channelid);

      if (status == TSW1100_UNSUCCESSFUL) {
        printf("ERROR: Could not read from Channel %d!\n",
            1) ;
      }
      else {
        for(i=0; i<(ChannelCommandLine[iChannelSel].nBytes
            /4); i++) {
          fprintf(ChannelCommandLine[iChannelSel].logFile,
              "%04d\n", (0xFFFF-(ChannelCommandLine[
              iChannelSel].rbuf[i] & 0x0000FFFF)));
```

```
                fprintf(ChannelCommandLine[iChannelSel].logFile,
                    "%04d\n", (0xFFFF-((ChannelCommandLine[
                    iChannelSel].rbuf[i] >> 16) & 0x0000FFFF)));
              }
            }

            ChannelCommandLine[iChannelSel].iInterimBytes +=
                ChannelCommandLine[iChannelSel].nBytes ;
            ChannelCommandLine[iChannelSel].iReadBufferFlag =
                TSW1100_CONTINUEBUFFER ;
          }
        }
      }
      //Uploading finished!

      for(iChannelSel = TSW1100_CHANNEL1; iChannelSel <=
          TSW1100_CHANNEL2; iChannelSel++) {
        if (ChannelCommandLine[iChannelSel].bOutputFileUsed ==
            TRUE) {
          //Close output file
          fclose(ChannelCommandLine[iChannelSel].logFile) ;
        }
      }
      nbrSamplesTaken++;
    }
    //Unload DLL
    UnLoadDLL(hinstLib);
    }
    return 0;
}
```

Appendix C

MIDAS users's manual

MIDAS

Matlab Interface for Digital Acquisition of Signals

User's manual

Introduction

The MIDAS software is a Matlab GUI that controls the Texas Instruments TSW1100 data acquisition board and the Agilent Technologies 33220A function generator. Both devices can be controlled separately or simultaneously.

Please note that this document is written specifically for use with the ADS6145EVM ADC. If any other ADC board is used, consult the corresponding user's manual for information about maximum clock- and input levels, power supply, connections and sampling frequencies.

1 System requirements

Due to limited availability of hardware components (other function generators, ADC evaluation boards etc), the software has only been tested for one setup. Other components might, or might not, be compatible with the software. A novel programmer should be able to modify the source code according to his or her hardware setup.

1.1 Hardware requirements

Texas Instruments TSW1100 data acquisition board
: *Connected to a compatible ADC evaluation board (such as the TI ADS61xxEVM series) and to the computer through an USB cable.*

Agilent Technologies 33220A function generator
: *Connected to the computer via USB or GPIB interface.*

Proper power supplies for DAQ, ADC and Clock circuits

1.2 Software requirements

Microsoft Windows XP (32-bit)

MathWorks MATLAB R2007a or later

MathWorks MATLAB instrumentation toolbox

Agilent 33220A MATLAB instrument driver[1]

Windows Drivers for the TSW1100EV

VISA drivers for the Agilent 33220A

[1] Included in latter versions of the Instrumentation toolbox, otherwise found at MATLAB Central file exchange - see http://www.mathworks.com/matlabcentral/

2 Startup procedure

Connect the TSW1100 to the ADC board. Provide power (GND to J9 and +12V DC to J8) to the TSW1100 and to the ADS6145EVM (GND to J12, J14 and +3.3V DC to J13).

Connect a USB cable between TSW1100 J10 and the computer. Wait for LED D13 to light up, showing that the TSW1100 is communicating properly with the computer. If LED D13 does not light up, or if connecting the TSW1100 for the very first time, you might need to (re-)install the drivers. Consult the TSW1100 user's manual for this.

Apply a clock signal to the ADS6145EVM SMA connector J9. The ADS6145EVM supports a wide range of clock signals but make sure that the voltage level and frequency is not too high.

Turn on the function generator and make sure that it is connected to the computer via an USB or GPIB cable.

Supply an analog input signal to the ADS6145EVM SMA connector J8. *Make sure that the input does not exceed the maximum voltage level (*$2V_{p-p}$ *for the ADS61xx family).* Also, note that low-frequency components (less than 1MHz) will be filtered at the input stage of the ADC board.

Launch the MIDAS software.

3 Operating the MIDAS software

In Fig. 1, a screenshot of the MIDAS software is shown.

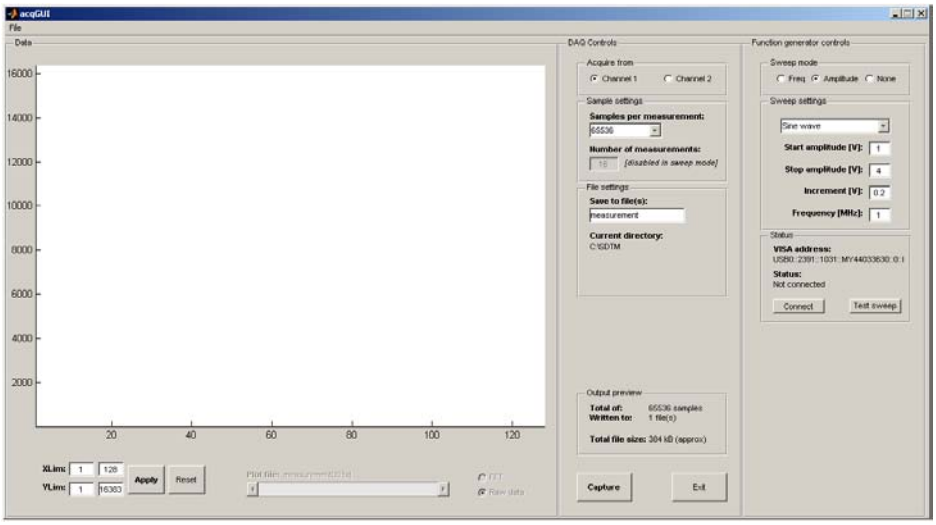

Figure 1: Screenshot of the software.

3.1 Acquiring data from the TSW1100

When capturing data, MIDAS calls an executable file that handles the communication with the DAQ. The executable saves the raw data to files, one for each measurement. The data is then imported back into matlab, where it can be viewed and further processed. Additionally, for each set of measurements, Matlab creates a file containing metadata such as clock frequency, date and time that the measurement started and ended, and so on.

In order to capture data, the user should specify the following in the "Settings..." menu (Ctrl+S to open):

Location of executable
Should point to a file named singleShot.exe.

Save Path
The path that data is saved to and read from.

Clock frequency
: Frequency of the clock signal, i.e. the sampling frequency. This setting only affects the FFT plots (and the saved metadata) and can be omitted.

To capture a set of measurements, specify which channel the ADS EVM is connected to, the number of samples and number of measurements. Optionally, specify a custom filename. With sample settings according to fig. 1, a total of 15 files named *measurement00.txt, measurement01.txt, ..., measurement14.txt* will be produced, each file containing 8192 integers ranging from 0 to 16383 ($2^{14} - 1$, since ADS6145 is a 14-bit ADC). The files are then loaded back into matlab and plotted.

3.2 Controlling the Function generator

The function generator is controlled directly through matlab. Specify the function generator's (unique) VISA address in the "Settings..." menu and connect. If successful, the screen of the function generator will display the text "MATLAB mode".

When connected, the user can perform either a frequency or an amplitude sweep. To test the sweep function, connect the function generator to an oscilloscope and press the "Test sweep" button. In order to sweep while acquiring data from the TSW1100, press the "Capture" button. This will carry out the same sweep as "Test sweep". *Note: If you connect the function generator output direcly to the analog input of the ADS EVM, make sure that the output does not exeed* $2V_{p-p}$.

4 Troubleshooting

PROBLEM: LED D13 at the TSW1100 does not light up.
Solution: Try disconnecting the USB plug and the power supply, wait for a few seconds and reconnect them. If the LED still does not light up, it is highly probable that the TSW1100 driver has crashed or failed to install/activate. Connecting the board to a different USB port might do the trick, otherwise reboot the computer.

PROBLEM: The program says "ERROR: No clock detected"
Solution: Of course, make sure that the clock is connected. Also make sure that you are trying to capture from the channel that the ADC is connected to and that LED D13 is illuminated.

PROBLEM: I can't connect to the function generator.
Solution: Make sure that your system meets the software requirements (see section 1.2), that is, that the Matlab Instrumentation toolbox, the 33220A instrument driver and the correct VISA-drivers are all installed. If Matlab is throwing a lot of strange errors, your system probably lacks at least one of the above. If MIDAS itself is throwing the error, find the function generator's VISA address and make sure that it is the same as specified in the "Settings..." menu.

5 Frequently asked questions

How do I compile the C++ source code?
Prefferably by using the Microsoft Visual C++ development platform.

Can I change the default paths?
Yes, search the m-code for "default values". Alter whatever you find necessary.

Does the program contain an easter egg?
Yes! Heavily obfuscated, search through the source code carefully.