TEKNISKA HÖGSKOLAN

HÖGSKOLAN I JÖNKÖPING

Development of a computer interface for a clamp-on ultrasonic flow meter.

Peter Sundin

Master Thesis 2007

Embedded electronics and computer systems

TEKNISKA HÖGSKOLAN

HÖGSKOLAN I JÖNKÖPING

Development of a computer interface for a clamp-on ultrasonic flow meter.

Peter Sundin

Company supervisors: Kent Lindström and Olle Penttinen at SP Technical Research Institute of Sweden.

School supervisor: Shashi Kumar.

Examinator: Shashi Kumar

Scope: 20 points (D-level)

Date:

Archivenumber:

Sammanfattning

Sektionen för volym, flöde och temperatur hos SP Sveriges Tekniska Forskningsinstitut utför mätningar av volym, flödeshastighet och temperatur hos vätskor.

Flödesmätare kalibreras bäst i sin installation för att ta hänsyn till felkällor som installationseffekter och mediet. Om detta kan göras utan att behöva sätta in utrustning i röret innebär det flera praktiska fördelar.

Det finns sedan länge s.k. ”clamp-on” flödesmätare på marknaden, men även om de har blivit bättre har de fortfarande en osäkerhet som är 5-10 gånger för stor för att fungera som referensmätare vid kalibrering.

Denna magisteruppsats behandlar analys med hjälp av ”reversed engineering” samt vidareutveckling av funktionerna och databehandlingen hos en befintlig ”clamp-on” ultraljuds flödesmätare och dess datorgränssnitt.

Målet med projektet är att utvärdera och vidareutveckla flödesmätarens befintliga datorgränssnitt i syfte att erbjuda möjligheten att utnyttja Microsoft Excel och Visual Basic för datainsamling och mätning av vätskors flödeshastighet.

Abstract

The section for volume, flow and temperature at SP Technical Research Institute of Sweden performs measurements of volume, flow and temperature in liquids.

Flow meters are best calibrated in its installation to take sources of error like installation effects and the medium into account. If this can be done without having to place measurement equipment inside the pipe it will mean several practical benefits.

Since many years, clamp-on ultrasonic flow meters have been available on the market. But even with today's improvements they still have a measurement uncertainty in the measurements that is five to ten times too big to make them useful as references for calibration procedures.

This thesis focuses on analysis, using reversed engineering, of an existing clamp-on ultrasonic flow meter.

The goal of the project is evaluation and further development of the ultrasonic flow meter's existing computer interface with the purpose of offering the option of using Microsoft Excel and Visual Basic for data acquisition and measurement of the flow rate of liquids.

Acknowledgements

I would like to thank Olle Penttinen and Kent Lindström at SP Technical Research Institute of Sweden, as well as Tomas Bengtsson and Professor Shashi Kumar at Jönköping School of Engineering, for excellent supervision and support.

Finally I would also like to thank Alf Johansson at Jönköping School of Engineering for great advices and support during the development of the RS232-switch and I^2C to RS232-bridge used in this project.

Table of contents

1 Introduction

This thesis is performed at SP Technical Research Institute of Sweden.

The goal of the thesis ”Development of computer interface for a clamp-on ultrasonic flow meter” is to develop a new and improved computer interface for a clamp-on ultrasonic flow meter. The primary goal is to make the ultrasonic flow meter easier to use and get the flow rate-information in real-time.

1.1 Background

The section for volume, flow and temperature at SP Technical Research Institute of Sweden performs measurement of volume, flow and temperature in liquids.

Flow meters are best calibrated in its situation to take sources of error, like installation effects and the medium, into account. If this can be done without placing measurement equipment inside the pipe, the process can proceed, undisturbed, when calibrating the flow meter of interest. This is associated with many practical benefits. Since many years – clamp-on flow meters have been available on the market. They offer a solution to non-intrusive flow measurements. But even with today’s improvements they still have a measurement uncertainty that is five to ten times too big to make them useful as references for calibration-procedures. There are, however, opportunities for improvement by combining the measurement of flow rate with additional information and improved data processing.

1.2 Clamp-on ultrasonic flow meter

The ultrasonic flow meter [1] measures the flow rate of liquids in a pipe through the use of ultrasound using a technique called Transit Time Differential Measurement [2]. The equipment (Figure 1.1) consist of a sensor device with two ultrasonic transducers that are clamped on to the pipe, a meter-unit containing electronics for digital signal processing (DSP) [3] of the signals received from the sensor-unit, user interface consisting of display and simple keyboard and a computer interface with application software.

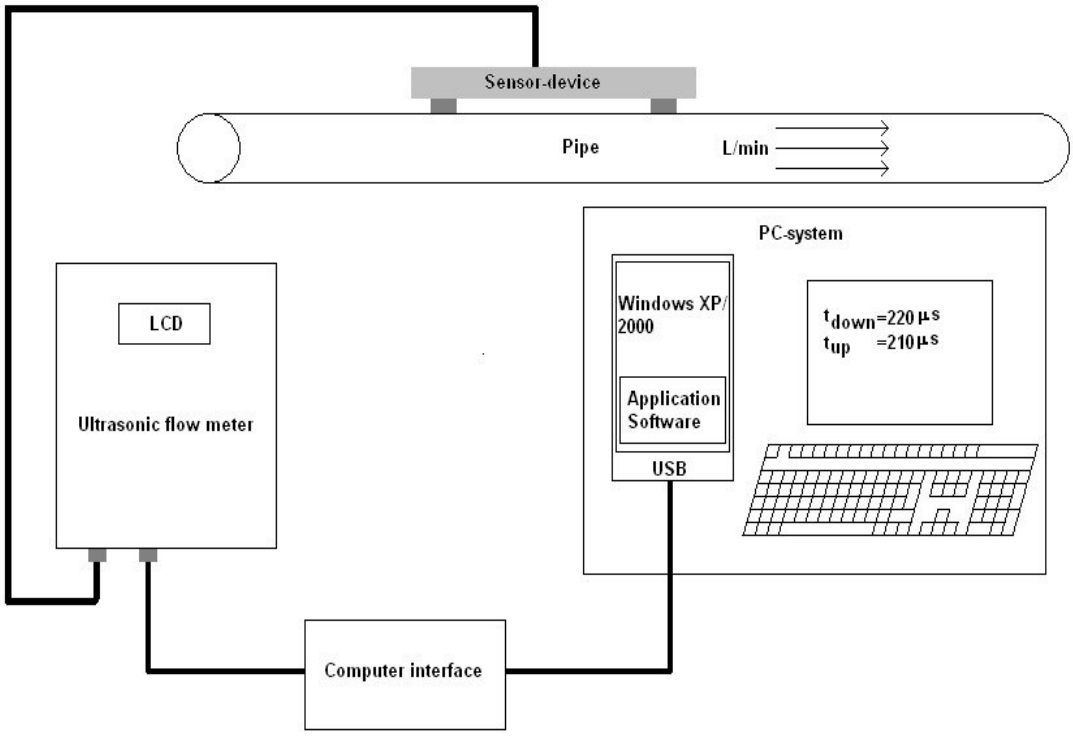

Figure 1.1: A block diagram of a complete system for clamp-on ultrasonic flow measurement using the transit time differential measurement method.

1.3 Purpose

The purpose of this thesis is to analyze components and functions in an ultrasonic flow meter meant for measurement of flow rate in liquids in a pipe. The thesis serves as a part of a main project where the object is to evaluate and do further development on the flow meter to make it fulfill the requirements of a reference meter.

The goal is to gain information about the design of the flow meter and its original computer interface and by using this information explore ways of improving the computer interface for the ultrasonic flow meter.

1.4 Description of problem

SP wishes to improve accuracy and simplify the use of the ultrasonic flow meter through the design of a new and improved computer interface. SP uses Microsoft Excel to analyze the measurements of flow rate, pressure and temperature. The user application for the ultrasonic flow meter creates a dump-file with values, which needs to be converted before it can be imported into another program, such as Microsoft Excel. Then the flow rate of the liquid in the pipe needs to be calculated from the acquired transit-times presented in the dump-file. This is very time-consuming and makes real-time measurement of the liquid's flow rate difficult.

It is possible to access the flow meter, using the original computer interface from Microsoft Excel through the use of a custom Visual Basic application and the virtual COM-port used by the computer interface. However, it quickly became obvious that the overhead associated with the RS232-protocol in combination with the interpretative nature of VBA offered a solution to slow to fulfill SP: s real-time requirements for data acquisition of the flow rates.

To solve this problem the development of new hardware is required to offer a new computer interface that supports faster data acquisition of the transit times, compared to what is possible using the RS232-interface, as well as support for measurements using several sensors in parallel.

The first task is to, through the use of reversed engineering, study the design of the flow meter and its computer interface. SP does not have access to sufficient technical documentation for the ultrasonic flow meter. This makes reversed engineering the method of choice for studying the unit.

The second task is to gain information about the communication protocol used for the communication between the computer and the ultrasonic flow meter. SP has no access to any information about the flow meters communication protocol. This leaves experimental study of the communication between the computer and the flow meter as the only practical method.

The third task is to use the information gained, through the use of reversed engineering, in task one to develop a new and improved computer interface for the ultrasonic flow meter to enable data acquisitions of the flow rate in real-time.

1.5 Scope and Requirements

The new hardware for the improved computer interface needs to be small enough and have low weight to be portable since SP often needs to perform measurements “on the field” when visiting clients.

The project must not be too costly while at the same time be flexible enough to offer further development such as connection of multiple flow meters and other type of sensors. After a discussion with the supervisors at SP, an agreement for a solution with support for four flow meter-interfaces with the option of attaching external devices to the I^2C-bus was reached

1.6 Environmental requirements

SP has high requirements on moisture and dust-durability as well as electromagnetic compatibility (EMC) [4] because they work with sensitive equipment that must not be exposed to interference caused by other equipment.

Electromagnetic compatibility means that the device must not be susceptible to or transmit electromagnetic interference. These factors are important to consider when designing the printed circuit boards for the flow meter-interface and the selection of enclosure.

When the project is finished SP will perform environmental tests on the equipment. This means that the equipments resistance to moisture, dust and electromagnetic interference needs to fulfill the requirements stated in the EU-directive for measurement instruments – 2004/22/EC [5].

1.7 Thesis outline

First chapter - Introduction: Gives an introduction to ultrasonic flow measurement and description of the problem solved in this thesis.

Second chapter – Theoretical background: Gives a detailed introduction to different flow measurement techniques, data acquisition and serial bus-protocols. Also contains a description of the scope and requirements of this thesis.

Third chapter – Methodology: Gives a detailed introduction to the reversed engineering method and its application in this project.

Fourth chapter – Improved computer interface to the ultrasonic flow meter: Describes the new and improved computer interface for the ultrasonic flow meter.

Fifth chapter – Testing and results: Describes testing of the new computer interface and the results of the project.

Sixth chapter – Conclusions: Describes the conclusions made after completing this project.

2 Theoretical background

2.1 Flow measurement

Some flow measurement techniques are

- Mechanical flow meters.
- Vortex flow meters.
- Magnetic, ultrasonic and coriolis flow meters.

Mechanical flow meters exist in several different versions, for example -

- Piston meter
- Paddle wheel and turbine wheel-flow meter.
- Venturi measurement.

2.1.1 Mechanical flow meters

Piston meter consists of a piston rotation in a chamber of known volume. For each rotation, the flow rate can be calculated by measuring the amount of water passing through the piston chamber.

The paddle wheel and turbine wheel-flow meter consist of a paddle or turbine wheel, which translates the mechanical motion of the wheel to a frequency that is proportional to the liquid's flow rate.

The venturi measurement principle works by restricting the flow in some fashion and measure the pressure difference across the constriction in the pipe.

2.1.2 Vortex flow meters

The vortex flow measurement-technique includes placing an object (called a shredder bar) in the path of the fluid, whose flow rate is to be measured. Small disturbances called vortices are created when the fluid passes the bar. The vortices trail behind the cylinder in two rolls. The speed at which these vortices are created is proportional to the flow rate of the liquid. The sensor-bar used by the vortex flow meters utilizes piezoelectric crystals that produce a small but measurable voltage pulse each time a vortex is created. The frequency of this voltage pulse is proportional to the flow rate of the fluid and is measured by the flow meter's electronics.

2.1.3 Magnetic, ultrasonic and coriolis flow meters.

In the magnetic flow meter an electromagnetic field is applied to the meter-tube, resulting in a potential difference proportional to the flow velocity perpendicular to the flux lines in accordance to Faraday's law.

Ultrasonic flow measurement can be done using Doppler-reading or the transit time differential measurement.

The ultrasonic flow meter studied in this thesis utilizes transit time differential measurement and works by transmitting a pulse of ultrasonic energy in a downstream direction thru the pipe with an inclination angle around 30° to 45°.

The flow meter measures the transit time for the ultrasonic pulse to reach the receiver. The process is then repeated in the other direction.

The coriolis flow meter utilizes the Coriolis effect [6] and works by causing a lateral vibration to distort the pipe. The result is a direct measurement of the mass flow.

2.2 Ultrasonic flow measurement

2.2.1 Doppler measurement

In doppler ultrasonic flow meters, the ultrasonic beam is bounced off small particles or bubbles in the fluid. The flow meter measures the frequency of the received ultrasonic echo and uses doppler shift [7] to determine the liquid's flow velocity.

2.2.2 Transit Time Differential Measurement

The sensor for the clamp-on ultrasonic flow meter studied in this thesis uses a sensor that is clamped on to the pipe and the previously mentioned "Transit Time Differential Measurement"-technique.

A plane wave of ultrasonic energy is first sent by transducer A through the pipe in a down stream direction and received by the transducer B (Figure 2.1). The time T_{down} is measured. Then the process is repeated in the other direction but with the transmitter and receiver functions reversed so that the plane wave intersects the flow under study in an up stream direction. The time T_{up} is measured.

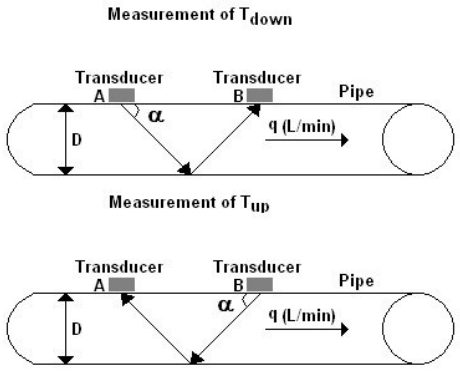

Figure 2.1: Measurement of a liquids flow rate in a pipe using the transit-time differential measurement method.

The Transit Time Differential Measurement-principle is based on the fact that a sound wave traveling in the direction of the flow is propagated at a faster rate compared to a sound wave traveling against the flow (T_{up}>T_{down}).

The transit times T_{up} and T_{down} is measured continuously. The difference in time (T_{up}-T_{down}) is directly proportional to the mean flow velocity (v) of the liquid. The volumetric flow rate per unit is the product of the mean flow velocity (v) and the cross sectional area of the pipe.

2.2.3 Calculation of the flow velocity and volumetric flow using the transit time differential measurement technique.

In the calculations below, D is the diameter of the pipe, c is the speed of sound, and α is the inclination angle of the transmitted ultrasonic pulses. Inclination angle α is normally between 30 and 45°.

A pulse traveling *with* the current from transducer A to B needs a transit time of:

$$T_{down} = T_{A\to B} = \frac{D}{\sin\alpha} \cdot \frac{1}{c + v \cdot \cos\alpha}$$

A pulse traveling *against* the current from transducer B to A needs a transit time of:

$$T_{up} = T_{B\to A} = \frac{D}{\sin\alpha} \cdot \frac{1}{c - v \cdot \cos\alpha}$$

The time difference for both pulses comes to:

$$\Delta T = T_{B\to A} - T_{A\to B} = v \cdot \frac{T_{B\to A} \cdot T_{A\to B} \cdot \sin(2\alpha)}{D}$$

From this equation the mean flow velocity v (in meters/s) can be calculated as:

$$(1) \qquad v = \frac{D}{\sin(2\alpha)} \cdot \frac{T_{up} - T_{down}}{T_{up} \cdot T_{down}} \quad \left(\frac{m}{s}\right)$$

The flow rate q is determined from the mean flow velocity v. If the pipe has a circular cross sectional area of A the following applies:

$$´(2) \qquad q = v \cdot A = v \cdot \pi \cdot \frac{D^2}{4}$$

Formula (1) inserted into formula (2) results in the formula: $q = \frac{\pi \cdot D^3}{4 \cdot \sin(2\alpha)} \cdot \frac{T_{up} - T_{down}}{T_{up} \cdot T_{down}} \quad \left(\frac{m^3}{s}\right)$

Example on calculation of a liquid's volumetric flow:

Parameters:

Inside tube diameter D: 100 mm

Inclination angle α: 45°

Measured T_{down}: 95.5862 µs

Measured T_{up}: 95.4949 µs

Calculation of the time difference Δt:

$$\Delta t = T_{down} - T_{up} = 95.5862\,\mu s - 95.4949\,\mu s = 91.3\ ns$$

Calculation of the flow velocity v:

$$v = \frac{\mathrm{D}}{\sin(2 \cdot \alpha)} \cdot \frac{\mathrm{T_{down}} - T_{up}}{T_{down} \cdot T_{up}} = \frac{100 \cdot 10^{-3}}{\sin(2 \cdot 45)} \cdot \frac{91.3 \cdot 10^{-9}}{95.5862 \cdot 10^{-6} \cdot 95.4949 \cdot 10^{-6}} (m/s) \approx 1\ m/s$$

Calculation of the volumetric flow q:

$$q = v \cdot A = v \cdot \frac{\pi \cdot D^2}{4} = 1 \cdot \frac{\pi \cdot \left(100 \cdot 10^{-3}\right)^2}{4} \approx 0.00785\ m^3/s$$

The volumetric flow in Liters Per Minute (LPM).

$$q(LPM) = \frac{q(m^3/s)}{0.001} \cdot 60 = \frac{0.00785}{0.001} \cdot 60 = 471\ LPM$$

2.3 Other applications of ultrasound

Ultrasound [8] is being used in a wide variety of measurement applications. Perhaps one of the most commonly known applications for ultrasound is the medical equipment used by the hospital for imaging of fetuses (Figure 2.2).

Figure 2.2: Ultrasonic image of fetus in womb [10].

The ultrasonic equipment (Figure 2.3) consists of piezoelectric [9] transducers (transmitter and receiver) that convert electrical energy into ultrasonic sound waves. The frequency of the ultrasound is often in the range of 20 kHz to 13MHz.

Figure 2.3: Medical ultrasonic imaging equipment [10].

The principle of operation for ultrasonic measurement equipment is based on transmitting a pulse of ultrasonic energy and measurement of the time taken to receive an echo of the pulse. The registered time for a response can later be used to calculate the speed of and distance to the target. If in addition to the time the location and strength of the echo is also registered, this information can be used to create an image of the target being examined by the ultrasonic equipment. This is the principle of operation utilized in the ultrasonic scanner used when performing medical ultrasonography [10].

Sound waves are also being used for sonic navigation and ranging (SONAR) [11]-applications by submarines and other marine equipment.

2.4 Data Acquisition Systems (DAQ)

A Data Acquisition System (DAQ) [12] is a dedicated computer system used for measurement and control applications. The system often consist of a personal computer with data acquisition hardware containing Analogue to Digital (ADC)- and Digital to Analogue (DAC)-converters as well as digital Input and Output (I/O) but could also be an embedded system, for example a portable data logger.

Data Acquisition begins with the physical property, for example flow rate, pressure and or temperature of an object to be measured. Analogue properties are measured using an Analogue to Digital converter and the data is transferred to the PC and logged for further processing. Digital properties such as time and frequency is often measured using counter/timer-hardware on the data acquisition-system and transferred to the computer.

Data acquisition software includes custom applications developed using programming languages such as LabView and Visual Basic, MatLab and applications developed by the original equipment manufacturer.

Common busses used by data acquisition systems include PCI [13], PXI [14], GPIB [15], HART [16] and USB [17].

2.4.1 PCI

PCI is one of the most commonly used buses for data acquisition-hardware for PC: s. Multifunction DAQ-cards with ADC, DAC, digital I/O, counter/timer and RAM-functionality are available from a large number of manufacturers. The PCI bus exists in a 32-bit and a 64-bit version with a clock speed of 33 to 266 MHz making it suitable for high-speed data acquisition in real-time applications.

2.4.2 GPIB (IEEE-488)

The IEEE-488 General Purpose Interface Bus (GPIB) is a specialized 8-bit bus for interconnection and control of measurement instruments. The GPIB-bus has a maximum speed of 8 and 64 Mbit/s (two versions exist) and supports addressing of up to 15 instruments.

A computer used for data acquisition using GPIB instruments requires a special GPIB-interface card.

2.4.3 PXI

PXI (Figure 2.4 and Figure 2.5) stands for PCI extension for Instrumentation and is a modular instrumentation platform originally introduced by National Instruments in 1997 and designed for measurement and automation applications that require high performance.

PXI is based on Compact PCI [18] and offer all of the benefits with the PCI architecture combined the 3U or 6U (1 U=4.4 cm) Eurocard [19] -form factor of Compact PCI.

Figure 2.4: A typical PXI system [http://www.ni.com/pxi/]

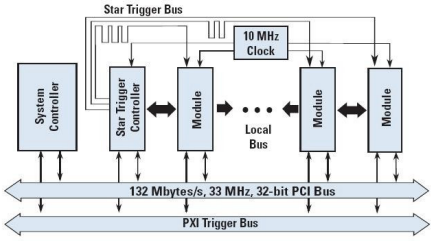

Figure 2.5: Block diagram of a PXI system. [http://www.ni.com/pxi/]

The PXI-controller typically consists of a Pentium 4-based computer running Windows XP, Windows 2000 or Linux (not yet PXISA approved).

2.5 Embedded System Buses

2.5.1 Compact PCI

Compact PCI is a 3U or 6U Eurocard-form factor industrial computer with all boards connected using a passive PCI backplane.

The pin assignments on the Compact PCI-connectors have been documented and standardized by the PCI industrial computers Manufacturing Group (PCIMG).

Like USB, Compact PCI-devices are hot plugged meaning they can be connected and disconnected while in operation.

2.5.2 Field buses

Fieldbus is an industrial network system for real-time distributed control applications.

Common fieldbusses include HART, CAN, PROFIBUS and industrial Ethernet.

Fieldbus is commonly used for interconnection of Programmable Logic Controllers (PLC) and are more flexible than older devices due to the inclusion of a CPU.

2.5.2.1 HART

Highway Addressing Remote Transmitter (HART) is an open protocol and implementation of Fieldbus developed by Rosemount Inc in the mid 1980:s originally designed as a digital communications protocol for their smart field instruments. Soon it evolved into HART and was made an open protocol in 1986.

HART utilize a 4-20 mA current loop for signaling.

The digital signal is frequency modulated.

HART supports two modes of operation – "point to point"- and multidrop-mode.

- In "point to point"-mode, the digital signals and overlaid on the 4-20 mA loop current. The polling address of the instrument is set to 0. Only one instrument can be put on each instruments cable pair.

- In multidrop-mode, the analog signal is fixed at 4 mA and only the digital signals are used. Multidrop-mode allows up to 15 instruments to be connected to the bus. Polling addresses for the instruments will be in the range 1-15. Each instrument need to have it's own unique address.

A Frequency Shift Keying (FSK)-modem connected to the PC or a hand held terminal act as master and decodes the messages.

The maximum transfer rate using HART is 1200 bits/s.

2.5.2.2 CAN-bus

The Controller Area Network (CAN)-bus [20] was originally developed for the automotive industry in the 1980: s but is now being used in data acquisition systems. Several commercial CAN-interfaces for the USB and PCI-bus are available on the market.

The CAN-bus was designed to be a robust bus with high electromagnetically noise immunity and utilizes signaling with a differential balanced line [21], making it ideal for data acquisition applications in electromagnetically noisy environments.

The CAN-bus has no clock and uses a sync-field in the packet and bit stuffing [22] to enable the receivers to synchronize to the transmitter.

Bit stuffing means that an extra bit of opposite polarity is sent after five consecutive bits of the same type have been transmitted.

Unlike other serial buses like I^2C and USB, CAN have no addressing of single devices attached to the bus. Instead the transmitted packet (Figure 2.6) has an identifier-field that identifies the contents of the packet. All devices that are interested in the information with the identifier transmitted on the bus will receive the packet.

CAN 2.0 exists in two versions:

- CAN 2.0 A with standard frame format
- CAN 2.0 B with extended frame format

The main difference between the two versions is the identifier field used to identify the content of the packet. CAN 2.0A uses a single 11-bit field while CAN 2.0B uses two identifiers of 11 and 18 bits, respectively.

The CAN-bus uses a principle commonly referred to as “wired and” as part of the bus arbitration-procedure. CAN use the name dominant and recessive bits for the logical levels zero and one. If a CAN-node transmits a dominant bit and another node transmits a recessive bit, the first node will see a dominant bit (as a result of the “wired AND”-principle, a logical zero has higher priority over a logical one). Arbitration is performed during the transmission of the CAN-packets identifier-field. The node with the lowest ID (highest priority) will win the arbitration.

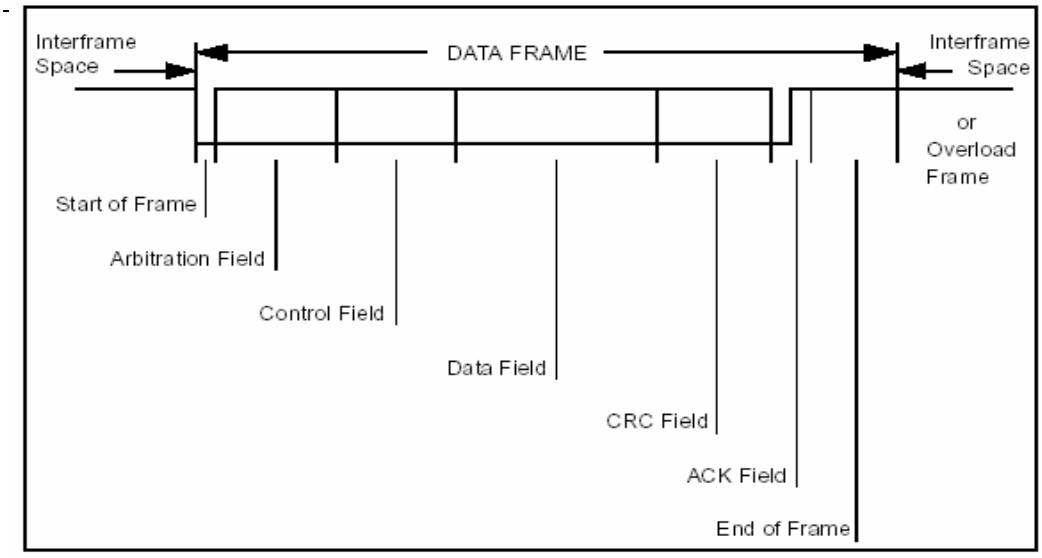

Figure 2.6: The different fields in a CAN frame.

2.5.2.3 PROFIBUS

Profibus stands for Process Field bus and is a protocol developed by Siemens in 1989 and is the most popular field bus.

Three different versions of Profibus exist –

- PROFIBUS-FMS (Field Bus Messaging Specification)
- PROFIBUS-DP (Decentralized Peripherals)
- PROFIBUS-PA (Process Automation)

PROFIBUS-FMS is based on the client-server model and is designed for the communication between automation devices.

PROFIBUS-DP is used for fast remote input and output and is used for connection of sensors and actuators to a controlling device.

PROFIBUS-PA is designed for process automation and used for the connection of field devices and transmitters to process control equipment.

2.5.2.4 Industrial Ethernet

Industrial Ethernet is an implementation of the Ethernet protocol for use in automation and production machine control applications in industrial environments.

Some of the advantages are

- Increased speed from 9.6 kbit/s to 1 Gbit/s
- Increased distance
- Increased overall performance
- Better interoperability
- Ability to use standard Ethernet-equipment such as routers, switches, hubs and cables.

2.5.3 USB

Universal Serial Bus (USB) is a popular bus often used in portable data acquisition equipment. The primary reason behind its popularity is the Plug and Play-functionality offered by USB-devices.

High Speed USB-devices have a maximum transfer rate of 60 MB/s making the bus suitable for real-time data acquisition applications.

Latency for USB-devices falls into the better category between Ethernet in the slow end and PCI and PCI Express in the fast end.

Bridges can be used to convert between USB and other protocols such as I^2C or RS232 greatly simplifying interface design.

USB-devices are self-powered and hot plugged, meaning an USB-device may be connected and disconnected while it is operating, without the need of turning of the computer. This is another reason behind the popularity of the USB-bus.

The communication interface used by the ultrasonic flow meter analyzed in this project uses a USB to RS232-bridge to offer Plug and Play-functionality.

One drawback with the USB to RS232-bridge is that it still relies on the RS232-protocol and is not able to use the full speed of the USB-bus.

Another drawback is that USB is a single master and multiple-slave bus (Figure 2.7) that needs a host (the PC).

Common USB data acquisition hardware on the market includes digital sampling oscilloscopes, data loggers and multimeters.

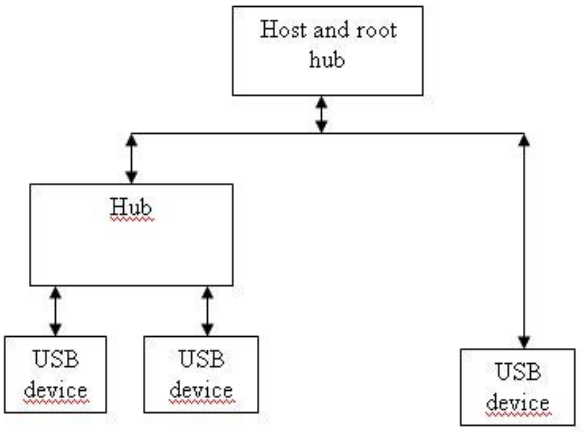

Figure 2.7: Diagram showing the tree-structure of USB

2.5.4 I^2C-bus

The I^2C-bus [23] was originally developed for use in consumer electronics devices in the early 1980:s but has evolved and could now be considered a serious alternative for data acquisition applications.

I^2C is a synchronous bus that support up to 128 devices in "standard mode" and up to 1024 devices in "fast mode". Connection of new devices to the I^2C -bus (Figure 2.8) is simple through the use of only two wires (Serial Data, SDA and Serial Clock, SCL).

I^2C exist in several versions.

- Standard speed- I^2C with a maximum transfer rate of 100 kbit/s
- Fast speed- I^2C with a maximum transfer rate of 400 kbit/s
- High speed- I^2C with a maximum transfer rate of 3.4 Mbit/s.

The fast speed-version is the version used in this project.

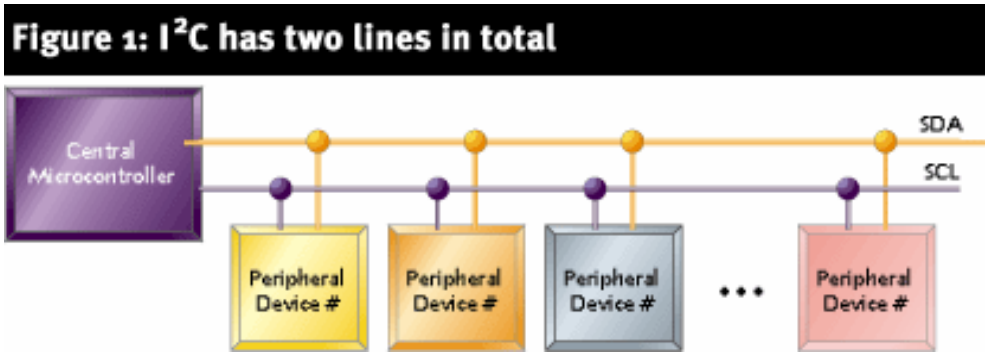

Figure 2.8: A schematic picture of the structure of the I^2C bus. [23]

The I^2C-bus is also available in a version that uses differential signaling, which improves the electromagnetic noise immunity of the bus.

The outputs of a I^2C-device are of open drain design, thus, a pull-up resistor on the SCL and SCL-lines is needed.

I^2C is a multi-master bus meaning several masters may have access to the bus. To avoid collisions some kind of bus arbitration is needed. The I^2C-bus is of the type "wired AND", meaning a master that request access to the bus sends a logical one on the bus while at the same time monitoring the signal on the bus.

If the master tries to send a one and receives a zero, it means someone else is sending and the master requesting access to the I^2C-bus continues to wait until the bus is free. The master that currently has access to the bus won't notice the other masters attempting to send. In most cases the I^2C-bus is used in its single master and multiple slave-configuration.

2.6 Data acquisition using Microsoft Excel and Visual Basic for Applications.

The use of Microsoft Excel for data acquisition (Figure 2.9) and analysis applications is possible because the software in the Microsoft Office-package supports the use of the programming language Visual Basic for Applications (VBA) [24]. VBA offers the functionality of communication with external hardware such as the ultrasonic flow meter and import of the measurements into an Excel spreadsheet.

After the data has been imported into a spreadsheet it is easy to use Microsoft Excel's built-in functions to perform further data processing, for example calculation of mean value, use of different formulas and presentation using diagrams and charts.

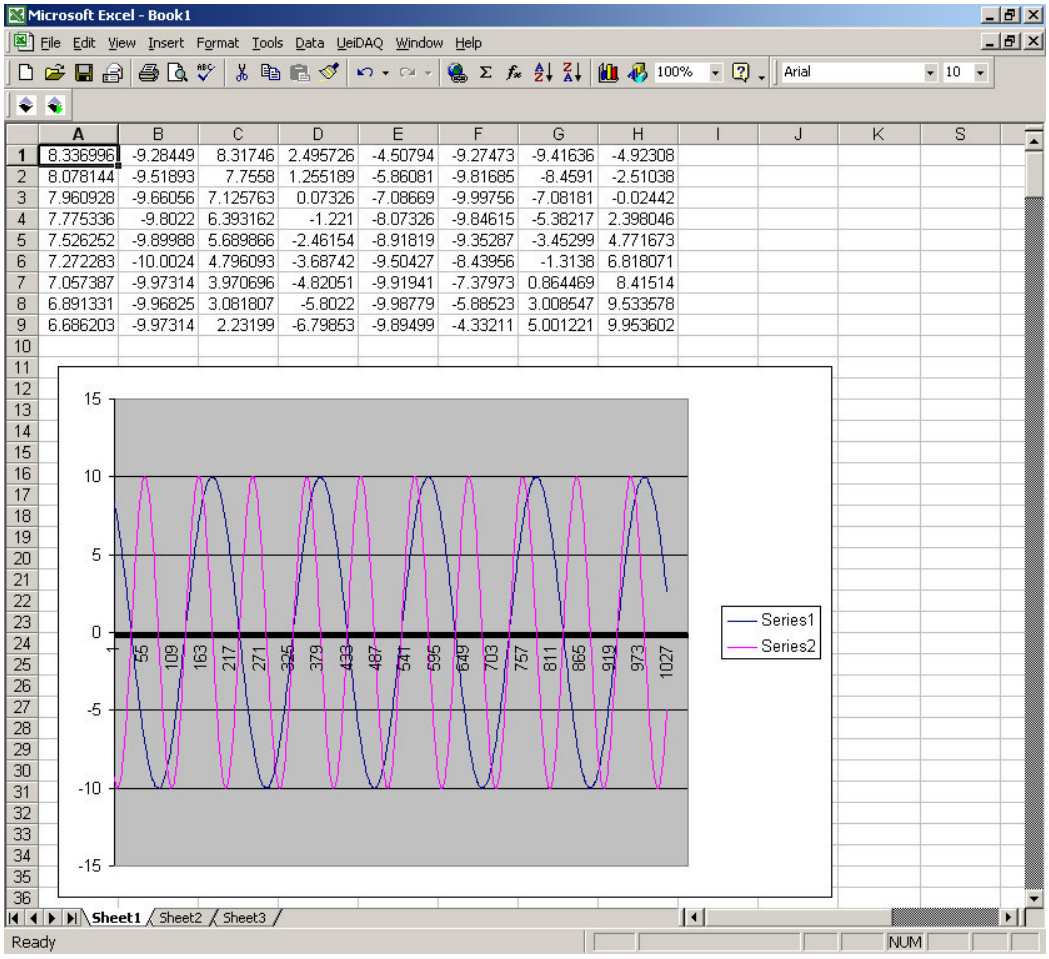

Figure 2.9: Data imported and displayed in a diagram Microsoft Excel spreadsheet.

3 Design methodology

3.1 Reverse engineering

The methodology used for analyzing the hardware for the flow meters computer interface with the purpose of gathering information about its design is called reverse engineering [25]. The goal of the method is through the use of experiments and studies of the hardware for the original computer interface gather information about the design. This information can later be used for further developments of the device.

The first step when performing reverse engineering of electronic equipment is to disassemble the device, study the Printed Circuit Board (PCB) and make notes of the manufacture ID: s of the circuits used in the design. After that a search is made in an internet-database containing datasheets for electronics components, such as the website Datasheetarchive [26]. Finally, the datasheets for the circuits of the device and interconnections between the components on the PCB is studied.

In some cases – datasheets for some or all of the components in the electronic device is not publicly available. One example is the case when the designer of the electronic device has utilized System on Chips implemented through the use of Application Specified Integrated Circuits (ASIC) [27]. In these cases it is very difficult or impossible to gain details of the design of the device.

If the product utilize known protocols such as USB, RS232 or I^2C it may still be possible to gather details about the protocol. One method when tracing the signals on the I^2C-interface is to find a known component, for example a microcomputer, on the PCB that is connected to the ASIC and use this as a point of origin when tracing the I^2C-signals.

To get details about the protocol and the packets of information, a measurement is done on the data and clock-signals using an oscilloscope. Further details about the protocol might be found by sending different commands while at the same time registering the response of the system.

Software can also be studied using reverse engineering by using a disassembler to convert the software machine language to assembler. When working with very large programs, a decompiler can then be used to convert the assembler-code to C-code.

Two hypotheses exist regarding what is sent from the flow meter to the computer.

The first hypothesis is that the time T_{up} and T_{down} are measured by the flow meter and then sent to the computer through the computer's USB-interface. The method used in the creation of this hypothesis was to save data dumps using the flow meters application software.

The second hypothesis is that the quotient $\frac{T_{up} - T_{down}}{T_{up} \cdot T_{down}}$ is calculated in the flow meter's DSP and sent to the computer where it is processed and displayed by the application software.

3.1.1 Hardware analysis

Reverse engineering of the flow meter and its computer interface included disassembly of the hardware and a study of their respective PCB:s. Notes of the manufacturer ID on the integrated circuits on the PCB for the ultrasonic flow meter and its computer interface was made. The numbers was looked up in the Internet-database Datasheetarchive and the components datasheet was downloaded and studied.

To get details about the pin-configuration, the connection between the pins in the HDMI-connector and corresponding pins on the flow meters DSP were traced with a multimeter. The same method was used when studying the hardware for the computer interface.

3.2 An analysis of the ultrasonic flow meters communications protocol

No details of the flow meters communication protocol will be revealed in this report due to a non-disclosure agreement with the SP Technical Research Institute of Sweden.

The equipment used in the analysis consisted of two personal computers, of which one acted as a simulator for the flow meter responses and the other for running the flow meters software. The computers were connected to each other using a null modem-cable.

To study the communication protocol of the flow meter, the software "Serial Port Monitor" from Eltima Software [28] was used. With the help of this software, the flow meter' responses to the application software's commands could be recorded. Different commands were tested manually and the reply sent from the flow meter was recorded each time. After recording of the communication between the application software and flow meter, a small program was written in VBA to simulate the flow meter's response to different commands.

The VBA-program waits for a command to be sent by the flow meters software through the computers serial port and null-modem and respond by transmitting a packet with the requested data.

The previously recorded responses sent from the flow meter were saved in a Microsoft Excel-spreadsheet and parameters were changed. Finally the flow meters software's response to the transmitted packets was registered.

This way it was possible to find the individual bytes in the packets sent by the ultrasonic flow meter containing the requested transit times t_{up} and t_{down}.

3.3 Reverse engineering of the ultrasonic flow meter and the original computer interface

Gathering details of the pin-configuration of the flow meter's computer interface-connector proved to be very difficult, primarily because of the multilayer PCB of the flow meter and difficulties in finding datasheets for some of the integrated circuits. Instead, a solution was chosen were the new and improved computer interface is connected between the USB to RS232-bridge and the RS232 to SSP-converter in the hardware for the computer interface. This solution has the benefit of offering the use of a standardized protocol for communication between the I^2C-interface and the RS232 to SSP-converter, as well as minimum amount of modification to the original computer interface's hardware.

The original computer interface for the ultrasonic flow meter consists of two PCB:s. One of the PCB:s is simply an USB to RS232-bridge and a driver for converting between the positive and negative voltage-levels on the RS232-interface and TTL-levels (+5 and 0 volts, respectively). The other PCB contains logic to convert the RS232-protocol to the synchronous serial protocol (SSP) used by the flow meter. The computer interface utilizes galvanic isolation using optocouplers between the RS232-interface and the RS232 to SSP-converter. Between the USB to RS232-bridge and the galvanic isolation is a complex network of transistors, diodes and resistors.

3.4 Results from the analysis of the communication protocol used by the ultrasonic flow meter

The analysis of the communication protocol used by the ultrasonic flow meter was successful. By monitoring the communication between the PC and the flow meter using the serial port monitor while simulating sending flow meter responses to the flow meters software, it was possible to find the individual bytes containing the requested Transit-times T_{up} and T_{down} in the packets sent from the ultrasonic flow meter.

3.5 Design options and decisions

A study of the ultrasonic flow meter's software reveals the use of a virtual COM-port through the use of the computer's USB-port. It is possible to access the virtual COM-port from Microsoft Excel; however after studying the communication protocol used by the ultrasonic flow meter the use of RS232-communication was considered to be too slow when working with Microsoft Excel. Also, the implementation of temperature measurement and the need for multiple ultrasonic flow meters require a new and more flexible computer interface. A simple protocol that solves this problem is I^2C.

The use of I^2C offers connection of multiple flow meters to a single bus. The mean value calculated from the measurements of the flow rate using multiple flow meters probably offers more reliable result compared to the output of a single flow meter. Each flow meter is identified with its own unique address on the I^2C -bus.

3.5.1 Hardware solutions

Two solutions exist for the development of the hardware for the improved computer interface.

The first alternative is using VHDL [29] and programmable logic (PLD) [30]. The advantage with this solution is that the computer interface may be developed entirely as a hardware device, which offers a very high-speed device designed as single chip. The disadvantage is that more work is required to develop the new hardware, compared to using existing hardware in the PIC microcontroller. Specialized development tools are also required for writing the VHDL-code and programming the PLD. The PLD: s is also more expensive, compared to a microcontroller, making this solution feasible only for mass produced devices.

The second alternative is to implement the computer interface using firmware and a microcontroller such as Microchip's PIC [31].

After discussion with SP, an agreement for a solution with a PIC microcontroller and software was reached to simplify future development of the computer interface. SP use Microchip's PIC microcontroller and lack the necessary tools required for VHDL and programmable logic-design.

To develop the new computer interface, information about the flow meters communication protocol was required. The method used to gather this information was to monitor the communication on the PC: s COM-port using a RS232-sniffer and register the flow meter's reply to commands sent by the flow meter's user application.

3.5.2 Microcontroller

Microchips PIC-microcontroller exists in several different versions. The different processors differ in memory, speed and peripheral functions.

Important considerations in this project are size, I/O-capabilities and the speed of the system to fulfill the real-time requirements. The microcontroller needs to be small and inexpensive and equipped with hardware for the I^2C -interface and an UART used for communicating with the ultrasonic flow meter. SP:s request of future development of the computer interface are factors important to consider in the selection of the microcontroller. At the same time, the cost of the project must be kept on a reasonable level.

The system needs to be fast enough to enable real-time data acquisition of the transit times measured by the ultrasonic flow meter. This is the primary reason behind the development of new hardware for the new computer interface.

One drawback with the PIC microcontroller is the limited amount of RAM.

The PIC16F877A used in this project have 386 registers that can be used as a general purpose RAM.

The use of the RS232-protocol in the original computer interface means that the Visual Basic application would have to read the information sent by the ultrasonic flow meter a single byte at a time in a loop. This is not a problem with compiled executable (.EXE) programs because of the speed of the computer. The interpretative nature of the Visual Basic language, however, combined with the overhead of the RS232-protocol would result in a system too slow to fulfill the requirement of near real-time data acquisition of the transit times.

The DLL-file that is included with the U2C-12 USB to I^2C-bridge handles all communication between the software and the U2C-12 hardware and stores received data in a memory buffer that can be accessed from the Visual Basic application.

4 Improved computer interface.

4.1 System overview

The new computer interface (Figure 4.1) consists of an USB to I^2C-bridge and a PIC microcontroller acting as a bridge between the I^2C- and the RS232-interface.

The DLL-file that is included with the USB to I^2C-bridge offers development of custom designed user applications for the flow meter through the use of the VBA programming language.

The primary motive behind the development of new hardware for the computer interface is based on the limitations and of the RS232-protocol. Compared to I^2C, RS232 is not well suited to handle large amounts of data sent in packets. The RS232-interface is difficult and slow when used from, for example, Visual Basic because the computer to read the packet sent by the ultrasonic flow meter byte by byte until the entire packet is received.

This limitation became apparent when working with the flow meter-simulator made in VBA. The big overhead associated with the computer's serial port is not a problem with compiled (.EXE)-programs because the execution time of the functions that reads data from the RS232-interface is minimal. The VBA-language, on the other hand, is an interpreting language that reads and interprets each line of code step-by-step, resulting in much slower execution of the program. As a result, it is important to be careful to avoid big loops, such as the ones used to read data from the computers serial port, in the program that could cause unnecessary delays.

The I^2C-interface on the other hand sends data in a packet, which is picked up by the DLL-file. The DLL-file stores the I^2C-packets in a memory buffer, which is easy and fast to access using programming languages such as VBA.

The flexibility of the I^2C-interface offers great possibilities for further expansion with multiple sensors and flow meters.

The computer-interface consists of a PCB with a PIC-microcontroller that works like a bridge between the I^2C and the RS232-interface. The Receive (RX)-wire from the RS232 to SSP-converter in the flow meter's computer interface is shared between the I^2C to RS232- and the USB to RS232-bridge in the original computer interface's-hardware. The rest of the wires are connected to the logic-PCB through a multiplexer. The multiplexer works as a switch and makes it possible to choose which interface (I^2C to RS232 or USB to RS232-bridge) should have control over the flow meter.

The design of the PCB for the I^2C to RS232-bridge, together with the use of the I^2C -bus, offers a high degree of flexibility. The system may in the future be expanded with several different modules, for example I^2C to CAN-bridge, analogue to digital and digital to analogue converters, etc, that is stacked on each other. Each new device added is given a unique address on the I^2C -bus, which simplifies the development of user applications on the PC.

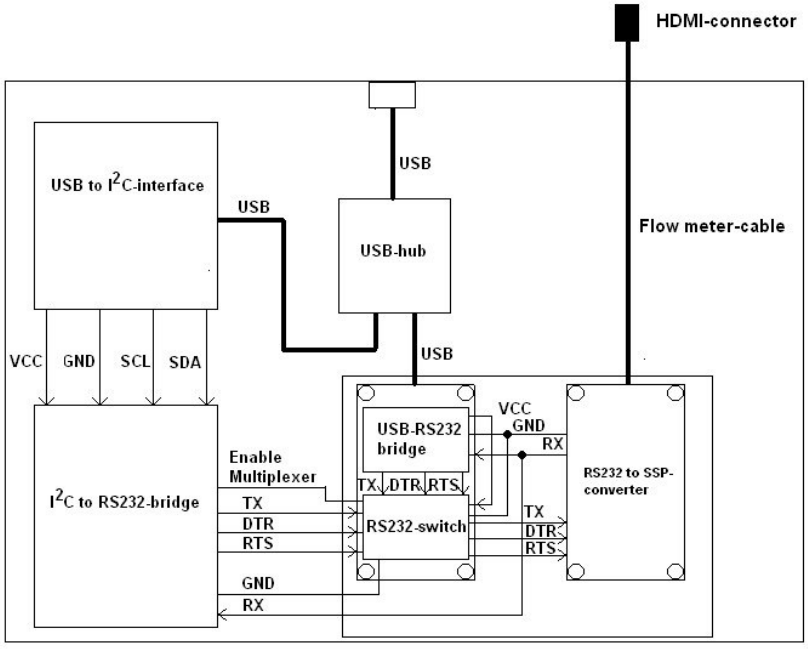

Figure 4.1: A block diagram of the new computer interface.

This solution has been chosen to simplify the use of the original computer interface as a stand-alone device. The RS232-Switch's "Enable"-signal for the multiplexer-IC is tied to ground through a 1 kOhm-resistor on the RS232-switch. When the I^2C to RS232-bridge is used, this signal is placed at +5 volts, which selects the RS232-inputs from the I^2C to RS232-bridge. The benefit with this solution is that the USB to RS232-bridge is automatically selected when original computer interface is being used as a stand-alone device.

4.1.1 USB to I²C-bridge

Today, there are several companies that offer complete USB to I²C -bridges.

The USB to I²C-bridge used in this project is named U2C-12 [32] (Figure 4.2) and manufactured by the Israeli company Dimax. The interface is built around the Cypress EZ-USB microcontroller.

Another benefit of the USB to I²C -bridge is the software-support in the form of a DLL with convenient functions that simplifies the use of the USB to I²C-bridge from for example Microsoft Excel.

The USB protocol is known for being complicated to work with. To simplify the development of the new computer interface and avoid having to bother handling the USB-protocol, the choice of using an USB to I²C-bridge was made.

The USB to I²C-bridge consists of a microcontroller with an USB and I²C-interface, respectively. A Dynamic Link Library (DLL)-file containing convenient functions for communication between the computer and the USB to I²C-bridge is included with the U2C-12 device. The functions offered by the DLL are simple to use from programming languages such as C and Visual Basic. A general purpose I/O (GPIO) and SPI-port is also integrated on the U2C-12 device.

The USB to I²C-bridge can perform read and write operations on byte as well as bit-level. In this project, only the byte-level functions are used.

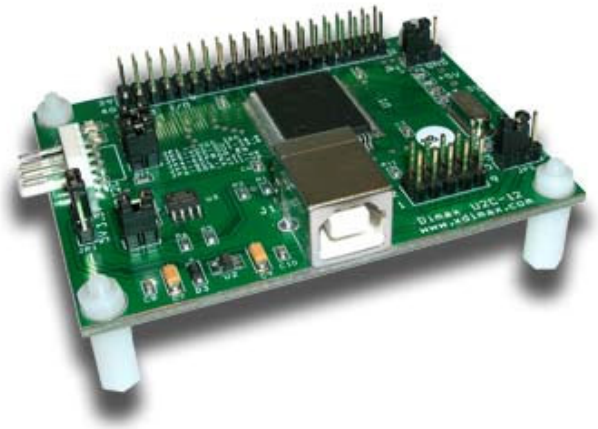

Figure 4.2: The USB to I²C-bridge from Dimax [32].

4.1.2 I^2C to RS232-bridge

The main component in the computer interface is the I^2C to RS232-bridge containing a PIC microcontroller with an I^2C and RS232-interface. The I^2C to RS232-interface is connected between the USB to RS232-bridge and the RS232 to SSP-converter in the original computer interface's-hardware through an RS232-switch consisting of a multiplexer and integrated RS232-driver. The Enable-signal of the multiplexer is connected to bit 0 of port B on the PIC microcontroller. This solution offers the selection of which interface (original computer interface- or I^2C-interface) should have control of the flow meter.

The task of the microcontroller is to wait for a command byte to be sent from the PC through the USB to I^2C-bridge and forward it to the ultrasonic flow meter. The microcontroller then reads the bytes sent from the ultrasonic flow meter and stores the information in a memory buffer to be sent to the PC using the I^2C-interface and the USB to I^2C-bridge.

In the early stages of development of the firmware for the PIC microcontroller a problem occurred when trying to communicate with the PIC using an USB to I^2C-interface from Elektor electronics. The reason for the problem was found to be the lack of support for detection of clock stretching in the TUSB3410-microcontroller used by Elektor Electronics USB to I^2C-interface.

Clock stretching is the ability for an I^2C-slave, in this case a PIC16F877A microcontroller, to pull the Serial Clock (SCL)-wire to ground to slow down fast I^2C-masters. The purpose of this function is to give the microcontroller time to load the transmission register SSPBUF with data to be sent. The serial clock must then be re-activated by setting the bit CKP in the status register SSPSTAT. In normal cases, the I^2C-master should detect when the SCL-wire is pulled low by the I^2C-slave and initiate wait-states. In this case, however, the I^2C-master failed to detect the clock stretching-event initiated by the I^2C-slave. The result was that the I^2C-masters continued trying to read from the I^2C-slave before the PIC microcontroller was ready to send. Texas Instruments explanation why TUSB3410 does not support clock stretching is that the main application for the I^2C-interface in the TUSB3410-device is communication with the I^2C-EEPROM used for storage of the firmware.

The clock stretching-problem disappeared when the speed of the PIC was increased. However, after a request from SP – a decision was made to switch to an USB to I^2C-bridge with built-in clock stretching-support. The choice was the U2C-12 USB to I^2C-bridge from Dimax.

The PIC16F877 microcontroller has a limited amount of RAM. For this reason, the choice was made only to store bytes that contain significant measurements, such as the transit times.

4.1.3 RS232-switch

The task of the RS232-switch is to offer the selection of which interface (original computer interface or I^2C) that should control the ultrasonic flow meter. The receive-signal from the original computer interface's-hardware does not need to be sent through the RS232-switch but is sent directly to the I^2C to RS232-bridge through a RS232 to TTL-converter located on the I^2C to RS232-bridge.

The USB to RS232-bridge used in the original computer interface's-hardware uses the signals Transmit (TX), Receive (RX), Data Terminal Ready (DTR) and Request To Send (RTS) of the RS232-protocol. The signals get converted to RS232 (+ and –10 volt)-levels using an integrated RS232-driver. These voltage levels are not suitable to be used with TTL-logic and needs to be converted back to TTL-levels. The solution chosen to limit the number of components on the RS232-switch was to use the ESD-protection diodes in the 74-series logic in combination with 1-kohm current limiting-resistors on the inputs to convert the RS232-levels back to TTL. After sending the signals through the multiplexer, the signals need to be converted back to RS232-levels again using an integrated RS232-driver. The reason for this is that galvanic isolation is utilized through the use of optocouplers between the TTL-logic and the USB to RS232-bridge in the hardware.for the original computer interface

Because a logic "1" is represented by a voltage of –5 to –10 volts on the RS232-interface, the TX, DTR and RTS-signals sent from the USB to RS232-bridge in the original computer interface will be inverted when the negative voltage is removed using the ESD-protection diodes in the multiplexer. To correct this, the output of the multiplexer, alternatively all of its inputs on the channel used by the USB to RS232-bridge, used in the RS232-switch needs to be inverted. To minimize the number of components and the size of the RS232-bridge the choice was made to use a multiplexer with an inverted output. This in turn means that the RS232-signals sent from the I^2C to RS232-bridge to the other channel on the multiplexer on the RS232-switch also needs to be inverted to give the correct output from the RS232-switch. This is done using TTL-inverters on the transmit (TX), Request To Send (RTS) and Data Terminal Ready (DTR)-signals located on the I^2C to RS232-bridge.

4.2 PIC Firmware

The PIC firmware is written in C and consist of a I^2C -interface, a decoder for the commands for the ultrasonic flow meter as well as functions to send commands and receive flow meter-data using the PIC:s Asynchronous Synchronous Receiver Transmitter (USART).

Figure 4.3 and Figure 4.4 on the next page show flow charts of the main function and interrupt service routine (ISR) for the PIC firmware.

The program starts by performing an initialization of the PIC's I^2C-interface and UART. It then goes into an eternal loop and waits for an interrupt request from the I^2C-hardware or incoming data in the UART's recieve-register.

The USART is configured for asynchronous mode with the speed of 19200 kbps, 8 bits of data and one stop bit.

A choice was made to enable interrupt only for the I^2C-interface. The reason is the risk that interrupt-requests from the USART-hardware will interfere with the I^2C-communication. Instead the USART:s receiver-interrupt flag RCIF is polled using a IF-statement in an eternal while-loop in the main-function.

When a command is received from the I^2C interface an interrupt occurs. This causes the PIC to jump to the Interrupt Service Routine (ISR) and check if the interrupt-request is created by the I^2C-hardware by checking the bit SSPIF. A choice was made to enable interrupt only for the I^2C-interface. The reason is the risk that interrupt-requests from the USART-hardware will interfere with the I^2C-communication. Instead the receiver-function waits for data to arrive in the USART:s receive-register. The bytes received from the ultrasonic flow meter are stored in a memory buffer using a pointer to index the next location. It is first decoded to decide how many bytes the PIC should expect to receive from the flow meter. Then the command is stored in the UART: s transmitter-buffer and sent to the flow meter. A command not used by the flow meter is also implemented to offer the option of switching to the original computer interface using the VBA-software and the I^2C-bus. Selection of the interface, I^2C or the USB to RS232-bridge in the original interface, that should be used to control the ultrasonic flow meter is done by setting or clearing bit 0 on the PIC processor's port B.

The PIC's memory buffer holding packets received from the flow meter can be read at any time using the PIC: s I^2C-interface.

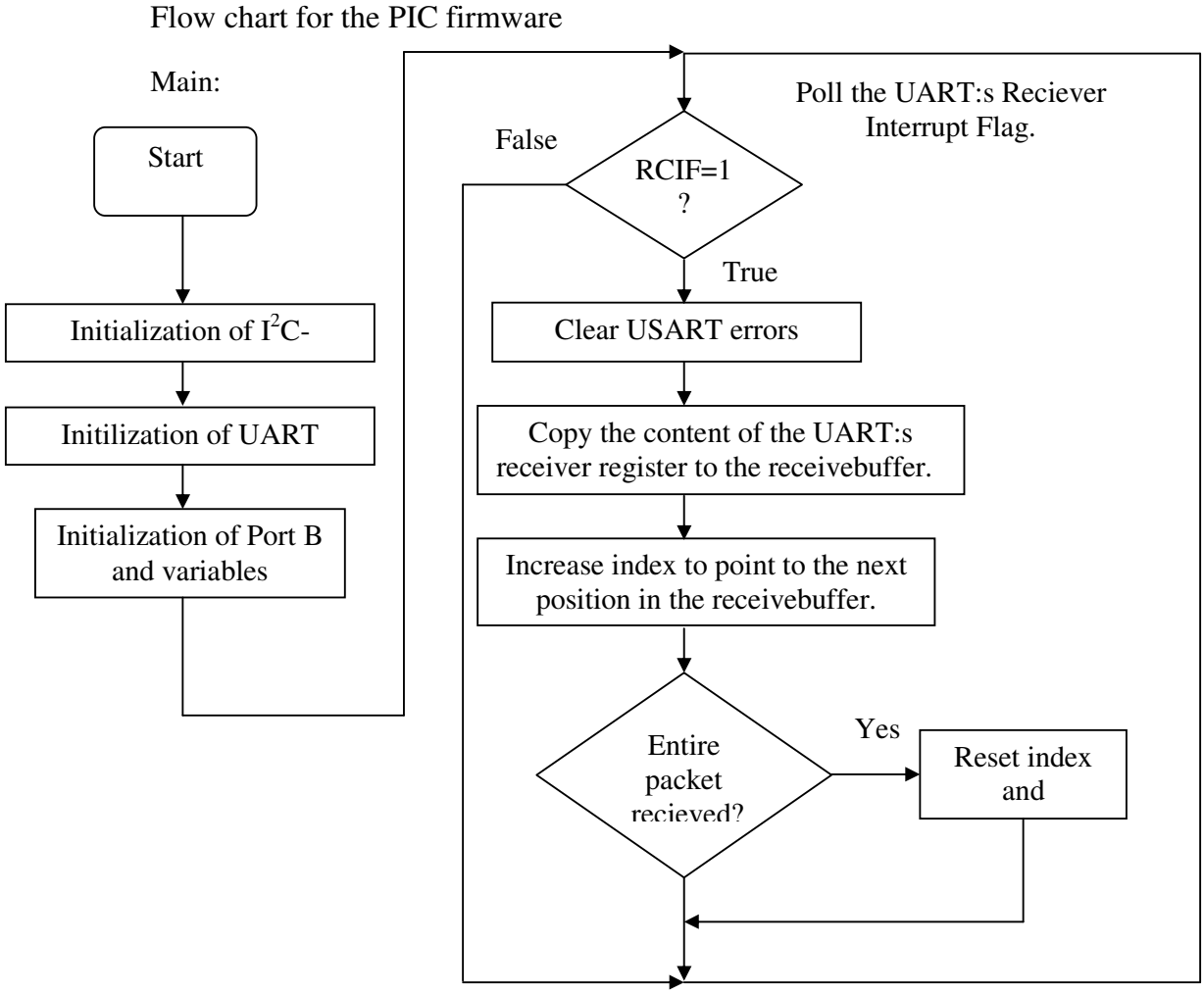

Figure 4.3: Flowchart for the main function and the eternal loop-structure.

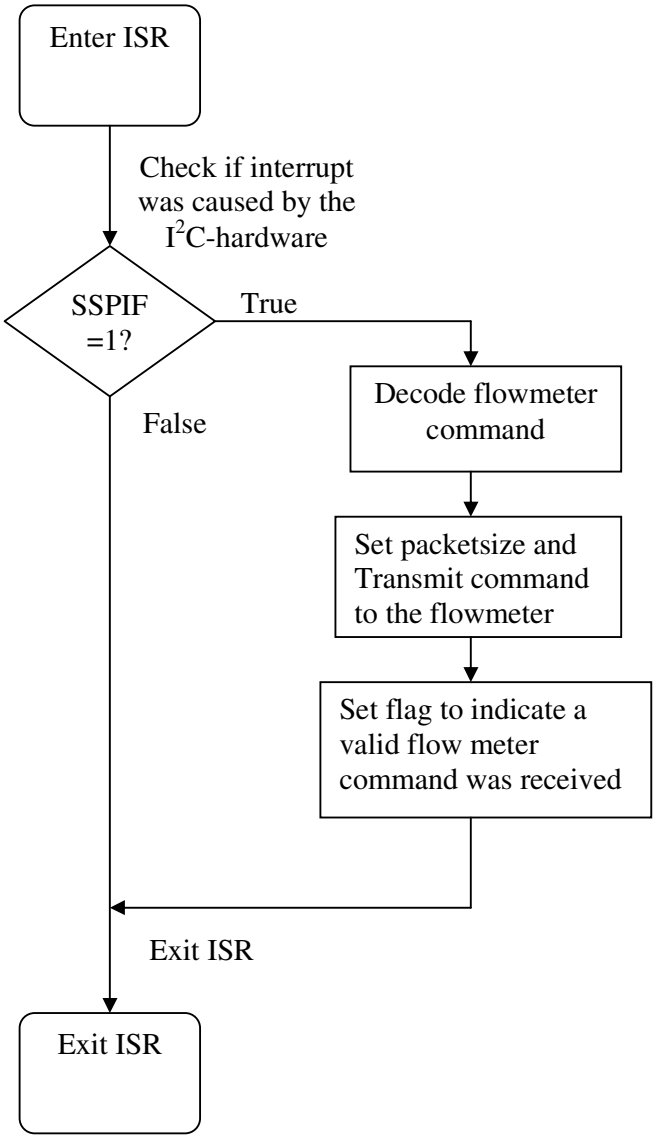

Figure 4.4: Flowchart for the Interrupt Service Routine (ISR)

4.3 User interface

The user interface (Figure 4.5) for the ultrasonic flow meter is written using VBA and Microsoft Excel.

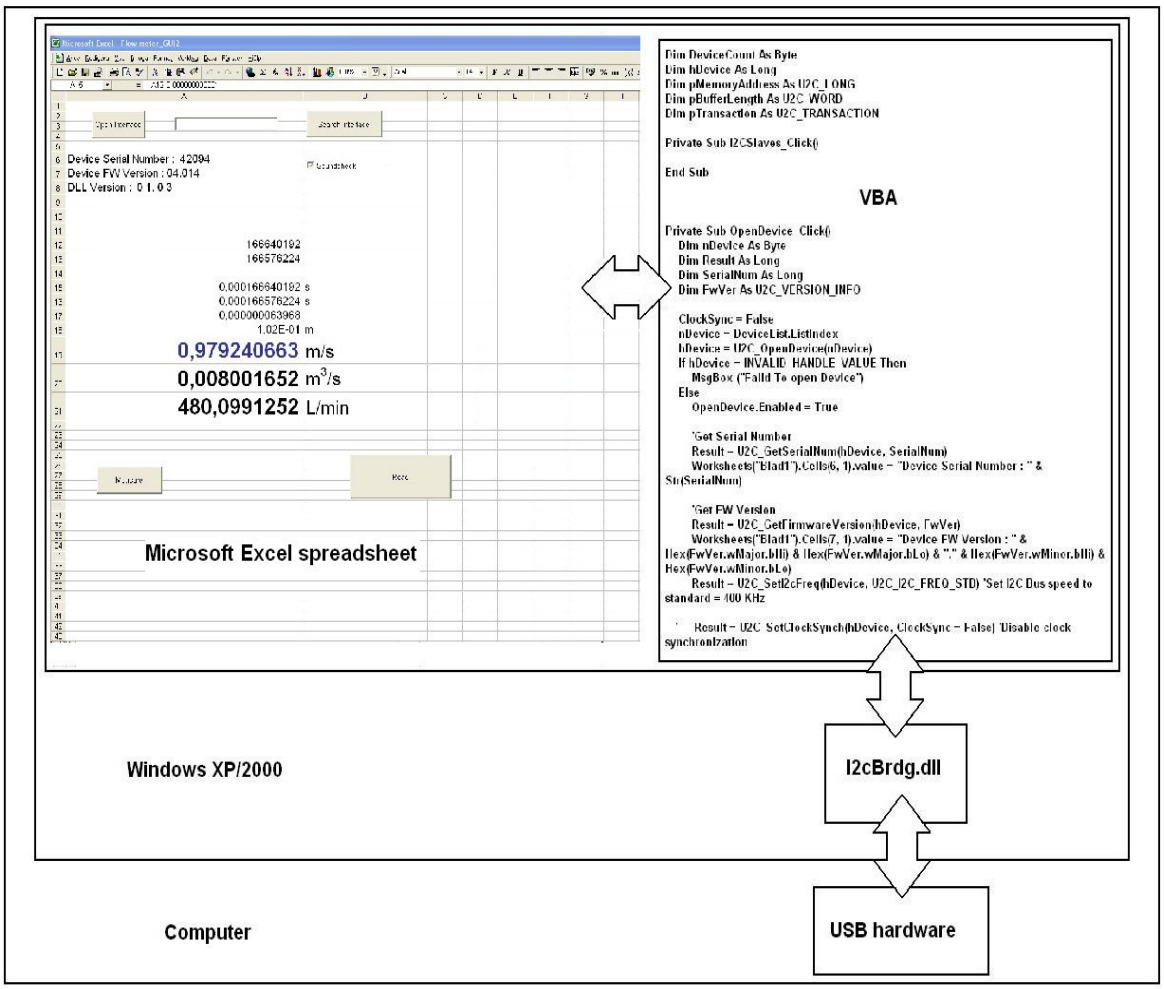

Figure 4.5: A block diagram showing the connection between the different components of the user interface and the computer.

4.3.1 The Application Programming Interface used by the USB to I^2C-bridge

The DLL for the U2C-12's-device's Application Programming Interface (API) contains a number of functions for communication with the USB to I^2C-bridge. In this project, the following functions have been used.
U2C_GetDllVersion ()
U2C_GetDeviceCount ()
U2C_OpenDevice (nDevice)
U2C_GetSerialNum (hDevice, SerialNum)
U2C_Write (hDevice, pTransaction)
U2C_Read (hDevice, pTransaction)

A driver for Visual Basic, named I2cBridge.bas, containing calls to the DLL is included with the U2C-12-device. This file also contains type-definitions for all of the parameters used by the USB to I^2C-bridge.

At startup, the USB to I^2C -bridge needs to be initialized.
When the user presses the "Search"-button in the VBA-application, the function U2C_GetDeviceCount () is called to perform a search for USB to I^2C-bridges connected to the computer. The function returns the number of U2C-12 devices found on the computer and stores the result in a list box-item in the VBA-application. These numbers is later used by the U2C_OpenDevice function to specify which U2C-12 device should be opened. The Search Device-subroutine also checks and reports the DLL-version using a call to the function U2C_GetDllVersion ().

Next the user needs to press the "Open"-button. The subroutine connected to this button uses the U2C_OpenDevice to open communication with the U2C-12-device selected from the previously mentioned list box. The function requires an argument containing the number of the U2C-12 device selected from the list box. If a communication channel with selected U2C-12 device was successfully opened, U2C_OpenDevice returns a handle to the selected device. This handle is later used for all subsequent communication with the USB to I2C-bridge. If the opening of the selected U2C-12 device fails, the string "INVALID_HANDLE_VALUE " is returned.

The read and write-operations on the USB to I2C-bridge is performed using the functions U2C_Read and U2C_Write respectively. The functions require a valid handle to the previously opened U2C-12 device and a pointer to a structure. Before accessing the U2C-12 device, the structure needs to be filled with the following information.

- Address of the I^2C-device to be accessed
- Memory address and length (only needed when accessing I^2C EEPROM: s)
- A buffer containing data to be written to or read from the selected I^2C-device
- The number of bytes to be written to or read from the selected I^2C-device.

In the Visual Basic example that is included with the U2C-12 device, the structure U2C_TRANSACTION is already predefined in the following way.

```
Public Type U2C_TRANSACTION
   nSlaveDeviceAddress As Byte
   nMemoryAddressLength As Byte    'can be from 0 up to 4 bytes
   nMemoryAddress As U2C_LONG
   nBufferLength As U2C_WORD       'can be from 1 up to 256
   Buffer(255) As Byte
End Type
```

The structure is easy to use from VBA by declaring a variable of this type.
Once the variable has been declared, Visual Basic will automatically show the defined members of the structure whenever the variable is being used.

Example:

```
Dim pMemoryAddress As U2C_LONG
Dim pBufferLength As U2C_WORD
Dim pTransaction As U2C_TRANSACTION

pMemoryAddress.b0 = 0
pMemoryAddress.b1 = 0
pMemoryAddress.b2 = 0
pMemoryAddress.b3 = 0

pBufferLength.bHi = 0
pBufferLength.bLo = 1

pTransaction.nSlaveDeviceAddress = 50
pTransaction.nMemoryAddressLength = 0
pTransaction.nMemoryAddress = pMemoryAddress
pTransaction.nBufferLength = pBufferLength
pTransaction.Buffer(0) = I2CWriteData
```

The types U2C_WORD and U2C_LONG used in the example above are also predefined in the I2cBridge.bas-file.

The acquired data for the transit times consists of four bytes and needs to be converted into a 32-bit word before it can be used in the calculation of the flow rate.

The conversion (Figure 4.7) is done through shifting and addition of the four bytes. Unfortunately, Visual Basic has no built-in shift functions. Instead, the solution is multiplying each byte with 256, 256^2 and 256^3, respectively and adding the products to form a 32-bit word

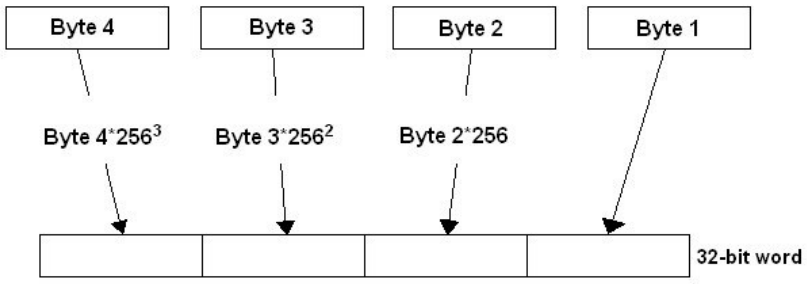

Figure 4.7: Construction of a 32-bit word from four bytes.

After the conversion is done, the word is multiplied with a constant to produce the transit times T_{up} and T_{down}

Visual Basic example showing the conversion of four bytes to a single precision 32-bit integer:

```
tup_1 = Worksheets("Sheet2").Cells(1, 1).value
tup_2 = Worksheets("Sheet2").Cells(1, 2).value
tup_3 = Worksheets("Sheet2").Cells(1, 3).value
tup_4 = Worksheets("Sheet").Cells(1, 4).value

tdown_1 = Worksheets("Sheet2").Cells(1, 5).value
tdown_2 = Worksheets("Sheet2").Cells(1, 6).value
tdown_3 = Worksheets("Sheet2").Cells(1, 7).value
tdown_4 = Worksheets("Sheet2").Cells(1, 8).value
'Convert the eight bytes to doubles.
tdown = 0
tdown = tdown_4
tdown = ((tdown * 256 + tdown_3) * 256 + tdown_2) * 256 + tdown_1

tup = 0
tup = tup_4
tup = ((tup * 256 + tup_3) * 256 + tup_2) * 256 + tup_1
```

4.4 Power supply

The new computer interface needs a power supply of +5 volts. Two alternatives exist for supplying the device.

1. Using an external power supply.

2. Supply the project using the USB-port's 5-voltage power supply.

The benefit with the first alternative is that the computer is protected in case of short circuit or overload of the computer's power supply. The drawback is extra hardware in the form of a Power Supply Unit (PSU).

The benefit with the second alternative is that it does not require any extra hardware. The new computer interface is ready to use as soon as it is plugged into the computer's USB-port. The drawback is that the maximum current on the computers' USB-port is limited to 500mA. An overload or short circuit of the computer's USB-port risk damage the computers power supply.

To find the total power consumption of the project, the completed project was connected to a 5-voltage power supply and the current was measured using a multimeter. The total power consumption of the new computer interface was measured to be 230 mA.

After a discussion with my supervisors at SP, the decision was made to supply the project using the computer's USB-port. The reason for the decision was SP: s need of using the flow meter and the new computer interface in the field and minimize the number of external cables.

4.5 Design for EMC

To fulfill the EMC-requirements in the EU-directive 2004/22/EC, the choice of a robust metal-encapsulation was made. EMC has also been taken into account the designing the PCB: s for the I^2C to RS232-bridge and the RS232-switch by keeping the traces on the PCB as short as possible.

The sockets on the rear panel has also been done with good EMC in mind by being careful to avoid large slits that might cause leakage of electromagnetic interference.

5 Testing and results

5.1 Evaluation of the ultrasonic flow meter.

The evaluation of the ultrasonic flow meter and analysis of the systems hardware, communications protocol and software revealed limitations and several ways for further development of the existing system.

5.2 The improved computer interface.

An improved and fully functional computer interface for the ultrasonic flow meter was developed.

The new computer interface offers extended software support on multiple computer-platforms such as Windows XP/2000 and Linux, real-time data acquisition and measurements of the liquids flow rate and several possibilities for future expansion of the system. By utilizing the existing hardware in the original computer interface it was possible to implement the design with a minimum amount of modification to the existing computer interface's hardware. The project could be considered as an extension of the functionality of the ultrasonic flow meter and its existing computer interface.

A test of the new computer interface shows a substantial increase in performance of the data acquisition of the transit times compared when using the RS232-interface. The increased performance is mainly the result of the I^2C-bus, the new hardware and the DLL used by Dimax's USB to I^2C-bridge, U2C-12, reducing the big overhead of the RS232-protocol on the computer.

The new computer interface offers SP the choice of using Microsoft Excel, the original equipment manufacturers software, LabView or any programming language of their choice that supports the use of DLL: s for data acquisition applications.

The RS232-switch used in the new computer interface offers remote selection of the interface (I^2C- or USB to RS232-bridge) to be used to control the ultrasonic flow meter.

5.2.1 Test and verification of the new computer interface hardware and VBA-application.

Figure 5.1 shows a test of the Microsoft Excel-application developed for the improved computer interface for the ultrasonic flow meter. The application was tested using previously recorded packets sent by the ultrasonic flow meter. A packet was entered into the Microsoft Excel spreadsheet and the flow meter-simulator developed with the purpose of analyzing the packets sent by the ultrasonic flow meter. A comparison was then done between the transit-times presented in Microsoft Excel and the transit times presented in the flow meters original software. From the results of the comparison it could be concluded that the individual bytes in the packet sent from the ultrasonic flow meter, discovered using the flow meter-simulator, contained the transit-times t_{up} and t_{down}. Figure 5.2 on the next page shows a test of the new and improved computer interface at SP in Borås.

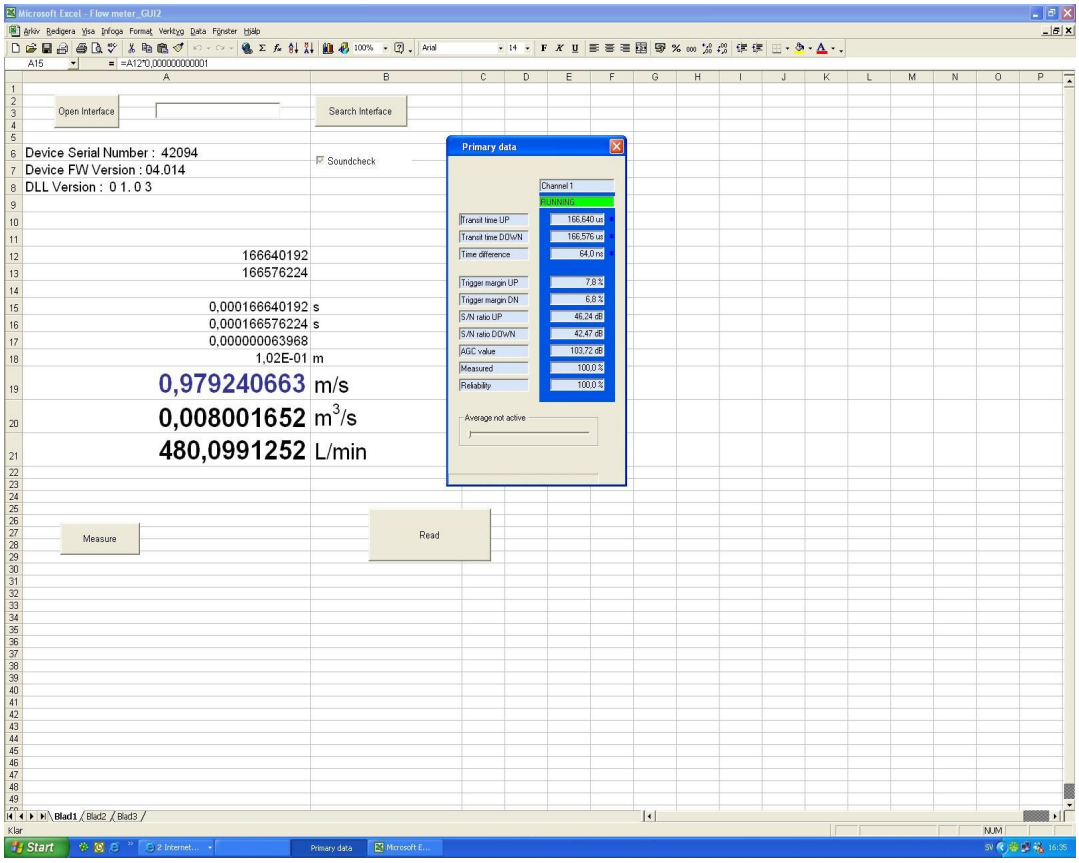

Figure 5.1: A test of the Microsoft Excel-application developed for the new computer interface, using a previously recorded packet containing the transit times for a known flow rate measured by the ultrasonic flow meter. The transit-times presented in the Microsoft Excel spreadsheet are found to be consistent with the transit-times, t_{up} and t_{down} (166.640 µs and 166.576 µs, respectively), presented in the ultrasonic flow meter's original software.

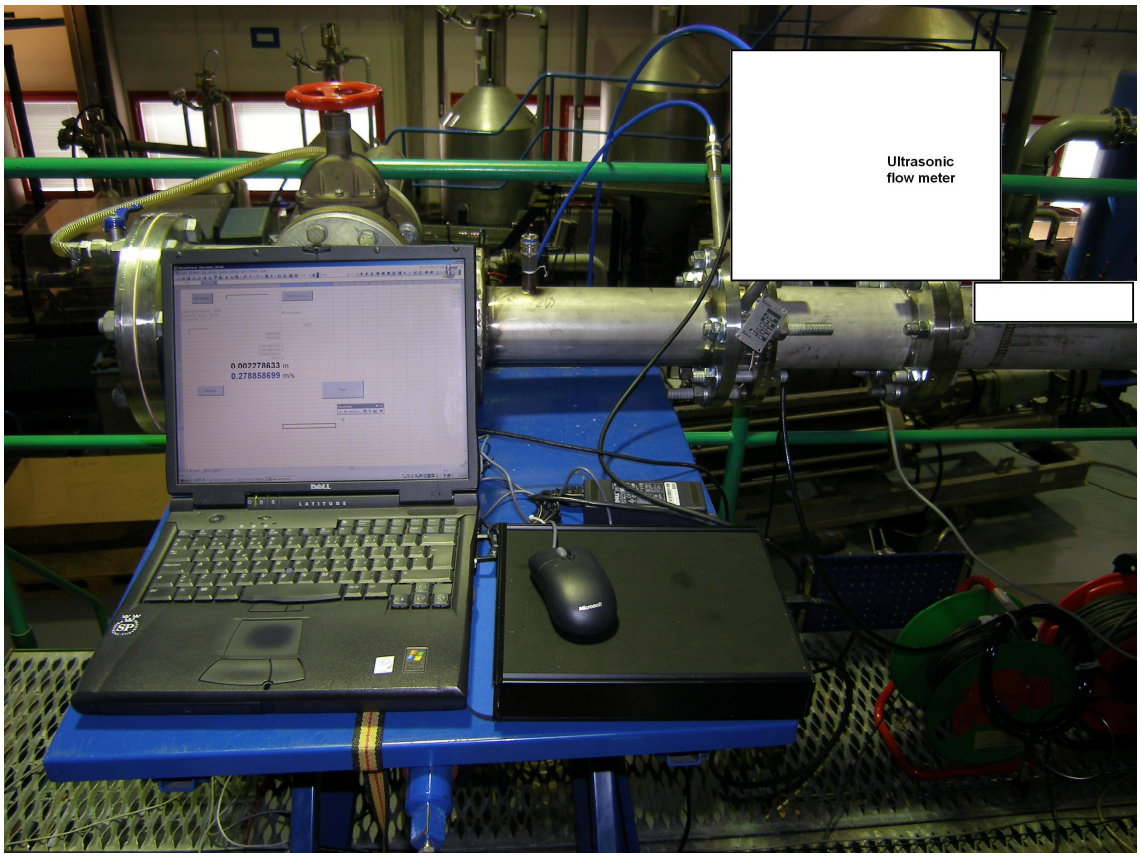

Figure 5.2: Test of the new computer interface (the black box to the right of the laptop) for the clamp-on ultrasonic flow meter at SP Technical Research Institute of Sweden in Borås.

6 Conclusions

6.1 Summary of achievements

A new and improved hardware for a computer interface for the ultrasonic flow meter was successfully developed. The new computer interface offers a greatly increased flexibility of the ultrasonic flow measurement-system compared to the previous interface. With the new computer interface, SP have option of using applications developed using any programming language of their choice that supports the use of the API offered by the USB to I^2C-bridge's DLL. In addition to flow measurement, the new computer interface supports easy expansion with measurement of temperature, pressure and other variables through the use of the I^2C-bus.

6.2 Limitations of the new interface.

6.2.1 Limitations of the I^2C to RS232-bridge

One of the limitations with the new computer interface is that it still relies on the use of the RS232-protocol for the communication between the I^2C-interface and the RS232 to SSP-converter used in the original computer interface.

A brief analysis of the RS232 to SSP-converter showed that it would be possibly to reverse engineer the device, however it was considered to be too time consuming to fit the scope of this project. As a result, in this project the RS232 to SSP-converter have been considered to be a black box. Further analysis and reverse engineering of the RS232 to SSP-bridge could eliminate the need for the use of a separate RS232 to SSP-bridge and, as a result, the use of the RS232-protocol by transferring the management of the ultrasonic flow meters SSP-interface to the PIC microcontroller. Because SSP is a synchronous communications protocol, it is theoretically possible to communicate with the ultrasonic flow meter close to the speed used by the I^2C–bus. The PIC16F877A has built- in hardware for the SSP-interface.

6.2.2 Limitations of the PIC microcontroller firmware

The PICC Lite C-compiler only allows access to two of the PIC-processors four memory banks resulting in a maximum amount of RAM of 80 bytes for storage of packets received from the ultrasonic flow meter. For this reason the decision was made to store only parts of the entire packets sent by the ultrasonic flow meter that contain the requested data, such as the transit times. The limitation in the PIC firmware can be easily corrected using a full version of the PIC C-compiler.

The PIC firmware contains no function for reporting to the PC that new flow meter-data is available. Instead, the firmware is built with the assumption that the communication with the ultrasonic flow meter is fast enough for data to arrive from the ultrasonic flow meter before the software running on the PC makes the next "poll" of the computer interface. One solution would be to reserve one of the bytes in the receive-buffer to be used as a flag for indication of completed acquisition of new data from the ultrasonic flow meter.

6.3 Future work

6.3.1 Expansion

The new computer interface can be expanded in a number of ways by connecting additional devices to the I^2C-bus. Devices that require higher communication speed than the speed offered by the I^2C–bus can be connected to the USB to I^2C-bridge's parallel General Purpose Input/Output (GPIO)-port.

7 References

[1] http://www.krohne.com/html/dlc/HB_ULTRASONIC_e_72.pdf (2007-06-08)

[2] http://www.krohne-mar.com/Measuring_Principle__Ultrasonic_Flowmeters__en.1346.0.html (2007-06-08)

[3] http://en.wikipedia.org/wiki/Digital_signal_processing (2007-06-08)

[4] http://en.wikipedia.org/wiki/Electromagnetic_Compatibility (2007-06-08)

[5] http://europa.eu.int/eur-lex/pri/en/oj/dat/2004/l_135/l_13520040430en00010080.pdf (2007-06-08)

[6] http://en.wikipedia.org/wiki/Coriolis_effect (2007-06-20)

[7] http://en.wikipedia.org/wiki/Doppler_shift (2007-06-24)

[8] http://en.wikipedia.org/wiki/Ultrasound (2007-06-24)

[9] http://en.wikipedia.org/wiki/Piezoelectric (2007-06-24)

[10] http://en.wikipedia.org/wiki/Ultrasonography (2007-06-24)

[11] http://en.wikipedia.org/wiki/SONAR (2007-06-24)

[12] http://en.wikipedia.org/wiki/Data_acquisition (2007-06-08)

[13] http://en.wikipedia.org/wiki/PCI_bus (2007-06-08)

[14] http://zone.ni.com/devzone/cda/tut/p/id/4811 (2007-06-08)

[15] http://zone.ni.com/devzone/cda/tut/p/id/3388 (2007-06-08)

[16] http://en.wikipedia.org/wiki/HART_Protocol (2007-06-24)

[17] http://www.usb.org/developers/docs/ (2007-06-08)

[18] http://www.interfacebus.com/Design_Connector_CPCI.html (2007-06-08)

[19] http://en.wikipedia.org/wiki/Eurocard (2007-06-08)

[20] http://www.semiconductors.bosch.de/pdf/can2spec.pdf (2007-06-08)

[21] http://en.wikipedia.org/wiki/Differential_signaling (2007-06-08)

[22] http://en.wikipedia.org/wiki/Bit_stuffing (2007-06-08)

[23] http://www.embedded.com/story/OEG20010718S0073 (2007-06-08)

[24] http://www.engineering.usu.edu/cee/faculty/gurro/VBA&Excel.htm (2007-06-08)

[25] http://www.jenkins-ip.com/serv/serv_6.htm (2007-06-08)

[26] http://www.datasheetarchive.com (2007-06-08)

[27] http://en.wikipedia.org/wiki/Application-specific_integrated_circuit (2007-06-08)

[28] http://www.eltima.com/products/serial-port-monitor/ (2007-06-08)

[29] http://en.wikipedia.org/wiki/VHDL (2007-06-08)

[30] http://en.wikipedia.org/wiki/Programmable_logic (2007-06-08)

[31] http://www.microchip.com (2007-06-08)

[32] http://www.dimax.com.ua/i2c_usb/u2c12.shtml (2007-05-28)