TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

JUHA JÄNTTI

THE EMBEDDED SOFTWARE OF A POWER MEASUREMENT CARD

Master of Science Thesis

Examiner: Prof. Karri Palovuori
Examiner and topic approved by the
Dean of the Faculty of
Computing and Electrical
Engineering
on 2nd of May 2018

ABSTRACT

JUHA JÄNTTI: The embedded software of a power measurement card
Tampere University of Technology
Master of Science Thesis, 48 pages, 5 Appendix pages
September 2018
Master's Degree Programme in Information Technology
Major: Embedded Systems
Examiner: Prof. Karri Palovuori

Keywords: measurement instrument, power and performance, data acquisition, embedded systems, firmware, software

Available commercial power measurement instruments and systems did not provide a solution meeting project requirement in terms of size, cost, performance, features and usability combination. Therefore, a new power and performance measurement card hardware that is tightly integrated with device under test and tailored for use with product enabling development boards was developed. The developed power measurement card is based on Atmel AVR32 microcontroller unit and a discrete analog front-end and is capable of measuring voltage and current of eight power rails and calculating average power. Measurement results are stored on a memory card or streamed to a PC measurement software via USB adapter.

Hardware design of the power measurement card and measurement accuracy characterization was done in collaboration with others and electronics design and results are briefly presented in this thesis. The embedded firmware of the power measurement card and a companion Java PC measurement application were developed as part of this thesis work. Accuracy and usability of the complete hardware and software solution met the project requirements. The developed power measurement card was successfully used to optimize power consumption of one product platform and enabled power measurements which otherwise would not have been practical to perform.

TIIVISTELMÄ

JUHA JÄNTTI: Tehonmittauskortin sulautettu ohjelmisto
Tampereen teknillinen yliopisto
Diplomityö, 48 sivua, 5 liitesivua
Syyskuu 2018
Tietotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Sulautetut järjestelmät
Tarkastaja: professori Karri Palovuori

Avainsanat: mittalaite, teho ja suorituskyky, sulautetut järjestelmät, laiteohjelmisto, ohjelmisto

Markkinoilla saatavilla olevat teolliset tehonmittauslaitteet ja -järjestelmät eivät täyttäneet projektin vaatimuksia koon, hinnan, suorituskyvyn, ominaisuuksien ja käytettävyyden kokonaisuuden osalta. Tämän ratkaisemiseksi kehitettiin uusi teho ja suorituskyky mittalaitekortti, joka integroituu tiiviisti testattavaan tuotekehitysalustaan. Kehitetty tehonmittauskortti rakentuu Atmelin AVR32-mikrokontrollerista sekä erilliskomponenteista kasatusta analogisesta mittapäästä. Mittaustulokset tallennetaan muistikortille tai lähetetään tietokoneella suoritettavalle mittausohjelmalle USB-adapterin välityksellä.

Tehonmittauskortin elektroniikkasuunnittelu ja mittaustarkkuuden karakterisointi tehtiin yhteistyössä muiden kanssa ja tulokset esitetään lyhyesti tässä diplomityössä. Tehonmittauskortin sulautettu ohjelmisto ja Java-mittausohjelmisto tietokoneelle kehitettiin kokonaan tämän diplomityön puitteissa. Kehitetyn elektroniikka- ja ohjelmistokokonaisuuden mittaustarkkuus ja käytettävyys täyttivät projektin vaatimukset. Kehitettyä tehonmittauskorttia käytettiin onnistuneesti yhden tuotekehitysalustan tehonkulutuksen optimoinnissa ja se mahdollisti sellaisten tehonmittausten tekemisen, mitkä olisivat muuten olleet epäkäytännöllisiä tehdä.

PREFACE

I would like to thank everyone who supported me during my studies and my co-workers at Intel Finland Oy for providing this thesis topic and opportunity to work with and learn from some of the most skilled engineers in the industry.

In Tampere, Finland, 23.9.2018

Juha Jäntti

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

A	Ampere, unit of current
ADC	Analog-to-Digital Converter
AFE	Analog Front-End
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASF	Atmel Software Framework
AVR	Atmel’s Microcontroller Architecture
B2B	Board to Board
C	A Programming Language
CSV	Comma Separated Values
DAQ	Data Acquisition
DC	Direct Current
DMA	Direct Memory Access
DMM	Digital Multimeter
DUT	Device Under Test
EEPROM	Electrically Erasable Programmable Read-Only Memory
FFD	Form Factor Device
FTDI	Future Technology Devices International Ltd.
FW	Firmware
GPIO	General Purpose Input/Output
GUI	Graphical User Interface
HW	Hardware
Hz	Hertz, unit of frequency
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IoT	Internet of Things
JTAG	Joint Test Action Group
LDO	Low Dropout Linear Regulator
LED	Light-Emitting Diode
Li-Ion	Lithium-Ion
LTS	Long Term Support
MCU	Microcontroller Unit
MUX	Multiplexer
OTG	On-The-Go
OSC	Oscillator
PC	Personal Computer
PCB	Printed Circuit Board
PLL	Phase-Locked Loop
PMIC	Power Management Integrated Circuit
PnP	Power and Performance
PoC	Proof of Concept
POR	Power-On Reset
RTC	Real-Time Clock
SD	Secure Digital
SDIO	Secure Digital Input Output
SMD	Surface-Mount Device
SoC	System-on-a-Chip
SSC	Synchronous Serial Controller

SW	Software
TDM	Time-Division Multiplexing
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
V	Volt, unit of voltage
VBUS	USB power rail
VREF	Voltage Reference
W	Watt, unit of power
XTAL	Crystal
°C	Degrees Celsius, unit of temperature
Ω	Ohm, unit of resistance
A_v	Amplifier voltage gain
I	Current
P	Power
R	Resistance
V	Voltage

1. INTRODUCTION

Increasing demand from customers is driving electronics industry towards smaller and more power-efficient devices for what is commonly known as the Internet of Things (IoT). Study by Gartner forecasts that the IoT market will continue its fast growth in the next years especially in the consumer appliances segment [3]. Some of the newer product segments include smart wearable devices and smart home appliances. Because of weight, size and cost requirements, these wearable devices are often powered from very low capacity lithium-ion (Li-Ion) batteries. To achieve the best possible battery life to meet customer demands in this very competitive business environment, it is important to optimize device power consumption in various use cases by hardware selection and software optimizations. Longer battery life is a way to gain market advantage over the competing solutions. But before actions for reducing device power consumption can be taken, one must first have accurate measurement data of the device and its subsystems' power consumption.

Traditional measurement systems designed for high sample rate data acquisition (DAQ) of up to hundreds of channels can cost tens of thousands of euros and be complex equipment to setup and use. They are not a practical solution for a high number of users with varying skill sets. Based on my own experience, setting up the required measurement software and hardware can be difficult and require deep understanding of both the measurement system and the device under test (DUT) to get accurate and repeatable measurement results. On the other hand, simple measurement instruments like digital multimeters (DMMs) do exist, but usually being limited to one measurement channel at a time and often lacking measurement software support makes them more suited for basic debug and measurement tasks than platform-level power measurements. Using them also requires good understanding of the DUT hardware to select appropriate measurement points and to perform the measurement safely.

There was a need for a power measurement system that is so easy to use that any engineer with a technical background, and without prior knowledge of the DUT hardware and the measurement system, could set it up with help of a light user manual in less than an hour. User should also be able to start doing meaningful measurements of the DUT right after setup is done. One key use case for the power measurement system would be software engineers doing power measurements when they make changes to a software build running on the DUT. Based on the measurement results, they would then know how their software changes affected device's power consumption in various use cases before committing software changes to others eliminating obvious mistakes in the earliest possible stage. It would be difficult to characterize power consumption in complex use cases that use multiple platform components and at the same time meeting performance goals unless you can know exactly how the power consumption is split between each subsystem component.

This thesis follows embedded system firmware development done while working at Intel Finland Oy showing some of the tasks and challenges a design engineer faces when developing software for a new proof-of-concept (PoC) embedded measurement solution. Developing embedded firmware and being part of a project team to accomplish customer requirements provided a great and rewarding opportunity for learning new skills in the field of electronics hardware, low-level C language programming and combined software and hardware project management. The new measurement solution was then completed by developing a measurement application for a personal computer (PC), which simplified the user experience and provided much needed development support during later stages of firmware development and hardware validation.

Scope of this thesis is developing the embedded firmware and the PC measurement software for a power and performance (PnP) measurement DAQ card. Figure 1 illustrates the thesis scope. The PnP DAQ card hardware was developed together with K. Ruoko and hardware development work is not part of this thesis. The hardware implementation is described in more detail in Ruoko's thesis [13]. Testing of the DAQ card was done in collaboration and relevant subset of the results are included in this thesis.

Chapter 1 lays the foundation for this thesis by giving background to the subject and explaining the measurement problem at hand. Chapter 2 outlines the project requirements and Chapter 3 describes the target hardware, the PnP DAQ card and a typical DUT system. In Chapter 4, development environment is setup. In chapter 5, software requirements are mapped to the target hardware components and firmware design is described. A short introduction to the PC measurement software follows in Chapter 6. Measurement accuracy test results are shown in Chapter 7 and Chapter 8 presents discovered software issues and solutions. Project execution and schedule is presented in Chapter 9. Finally, Chapter 10 summarizes how well targeted project requirements were met, and what were the key learnings from the thesis work.

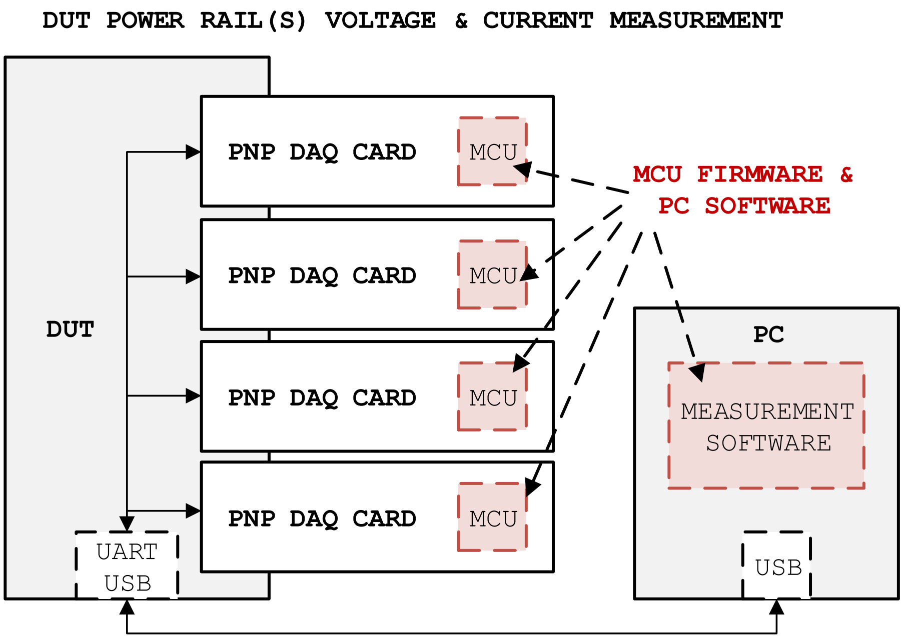


Figure 1. *Scope of this thesis is the firmware of the power measurement card and the PC measurement software. Hardware design of the DAQ card, which was done in collaboration with others [13], and an example DUT platform are briefly detailed and are not the key focus of this thesis.*

2. PROJECT REQUIREMENTS

At first, key project requirements for the new power measurement system were identified from perspective of the customer and end users. Requirements set approximate performance targets and other focus areas, which have a high priority in the implementation phase of the measurement card firmware and a companion PC measurement software. The key project requirements are listed in Table 1.

Table 1. *The key requirements of the power measurement system.*

# Requirement	Description
1.	Easy to setup and use
2.	Reliable operation
3.	Eight measurable power rails
4.	Sample rate of > 200 samples per second, data can be averaged
5.	Raw data storage to a micro SD memory card for cordless operation
6.	Real-time wired data streaming to a PC measurement application
7.	UART-to-USB connection and API for control and data streaming
8.	Good average DC voltage and current measurement accuracy with calibration support
9.	A graphical PC measurement software for real-time monitoring, control and data logging to .csv files
10.	The measurement system is used with product-enabling development boards and support for product form factor devices is not required

Because users of the new system may have little previous knowledge of the DUT or other measurement instruments, it was important to focus on usability and simple setup of the complete power measurement system. Expectation was set, that a new technically-minded engineering user would be able to setup and start measurements in less than 60 minutes by following a short user manual. This sets a high-quality standard for documentation, as well as companion PC measurement software implementation user experience. Once the power measurement system and documentation are ready, user testing would be performed to refine user manual and verify that requirements were met satisfactory.

From performance point of view, the system would focus on good average voltage of direct current (DC) and current measurement accuracy at a rate of modest 200 samples per second or higher. Based on early design estimations, performance targets of better than 10 mV and 10 mA range accuracy with typical milliohm (mΩ) range current shunt resistor values were set. Averaging of raw sample data by factor of 10 is acceptable to reduce data output rate, which would otherwise be significant from eight measurable channels. Since hardware architecture requires multiplexing four input channels at a time to an analog-to-digital converter (ADC), great care would have to be taken with firmware timings to prevent unwanted switching noise affecting measurements. To meet quality and performance standards of the end users, each power measurement card would have to be unit tested and possibly individually calibrated to meet the accuracy requirements. Software and firmware design should be built with support of at least basic offset and gain correction calibration for both voltage and current measurement for all channels. Simple calibration procedure would be made with the help from Maxim's application note [6]. Calibration coefficients and unique serial number would be stored on the measurement card's non-volatile electrically erasable programmable read-only memory (EEPROM) during production. Applying calibration coefficients may be offloaded from the measurement card's microcontroller unit (MCU) to a PC measurement software, which reads and applies the calibration coefficients and DUT's shunt resistor values to calculations during run-time. Because current and therefore also power calculations require knowledge of the DUT's shunt resistor values, they should be defined in user modifiable configuration file read by the measurement software.

To meet the usability requirements, and to provide a tool for development and debugging of the measurement system, a graphical PC measurement software with a simple enough user interface was required. Main supported operating systems would be 64-bit Microsoft Windows 7 and newer, but portability to Linux was preferred in the long run. Preferred Linux distributions would be Ubuntu 12.04 long term support (LTS) and 15.04 LTS. The PC measurement software should support at least minimum set of features to fulfill basic measurement tasks and provide means to log measurement results to a comma separated values (CSV) file.

The new power measurement system should be able to support portable use by logging data to a micro secure digital (SD) memory card, from which data can then be downloaded to a PC. However, the main use case for the measurement system would be doing wired measurements by streaming in data real-time from power measurement cards to a computer via universal serial bus (USB) virtual serial port.

To support basic data streaming and control features between a PC measurement software and the power measurement card firmware, a simple universal asynchronous receiver/transmitter (UART) application programming interface (API) is needed. This could optionally be used to integrate support of the developed power measurement cards to other software solutions, such as a factory automated unit test setup of a DUT hardware.

The developed power measurement system would primarily be used with product enabling system-on-a-chip (SoC) platform development boards such as macro-size debug versions of a product platform. Support for potentially very small final product form factor device (FFD) is not required of the system. Major issue supporting state-of-the-art mobile product FFDs that have a printed circuit board (PCB) the size of only a few square centimeters is that addition of current sense shunt resistors makes the layout routing less optimal and more difficult as you would not only need to route power rails to PCB top layer in order to add the shunt resistor but also route out the differential sense signals for external measurement card hardware. For the shunt resistors to be able to handle high currents, they would often need to be in larger surface-mount device (SMD) passive component packages such as 0805 or even 1206 in imperial units. Another difficulty would be connecting the current sense signals from the PCB to the measurement solution because there is no room for additional connectors on the product FFD. However, if the product form factor would not be as size constrained, the developed power measurement solution could be used even with FFDs.

3. HARDWARE DESCRIPTION

This chapter introduces the power measurement card hardware, for which firmware was developed as part of this thesis. Actual hardware design is not part of this thesis work, but some background is given because it is important to fully understand target device operation before developing firmware that supports it to make sure right things are done the correct and optimal way to get best system performance out of the hardware and firmware combination.

3.1 The PnP DAQ card

Basic operating principle of the power measurement card is detailed in Figure 2 simplified high-side current measurement circuit. Reasons for choosing high-side current sensing are detailed in an application note by Texas Instruments [15]. A high-side sensing circuit is preferred when application circuit cannot tolerate ground level disturbances [15, p. 3], which is the case with a modern complex SoC platform. Individual power rail inputs and outputs to and from a power management integrated circuit (PMIC) must be measurable instead of total ground current, which is another reason for using a high-side sensing architecture. On the developed PnP DAQ card, current sense circuit was designed for measuring low voltage DC power rails with focus on average accuracy. A low temperature drift, low resistance and high accuracy shunt resistor is placed on the high-side between a power rail supply and a load. Load current flowing through the shunt resistor causes a voltage drop proportional to the current. By measuring voltage drop over the shunt resistor and voltage on load side of the shunt, load current and finally power consumption can be calculated. Current *I* in amperes (A) is calculated with Ohm's law [4, p. 4] in Equation (1).

$$I = \frac{V}{R} , \tag{1}$$

where V is voltage in volts (V) and R is resistance in ohms (Ω). Then load power consumption *P* in watts (W) can be calculated with Equation (2) [4, p. 6].

$$P = V * I , \tag{2}$$

where V is voltage in volts and *I* is current in amperes. The ADC compares differential input signal to a reference voltage, and outputs digital value reflecting input signal level. This result can then be read by a microcontroller unit. If the ADC measures both the shunt resistor voltage drop, and the voltage delivered to load at the same time, instantaneous power can be accurately calculated. Amplifying the differential sense signal with an operational amplifier allows usage of lower shunt resistor values with less decrease to measurement accuracy at low currents by utilizing more of full-scale range of the ADC.

Amplification is needed because the DUT system cannot tolerate large voltage drops on already low voltage power rails that would happen with large shunt resistor values during load current spikes.

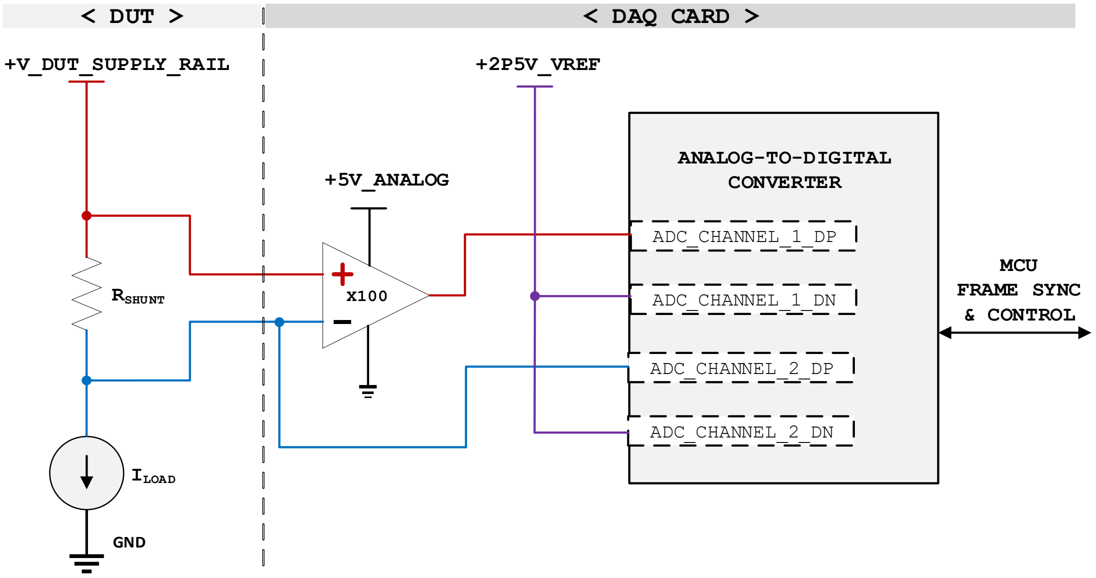


Figure 2. *A simplified high-side current and voltage sense circuit [15, p. 3] describing hardware division between a DUT and the DAQ card. Amplifier gain set resistors and negative feedback are omitted from the figure.*

Shunt resistor absolute value must be carefully selected by estimating maximum load current to meet DUT's load regulation requirements and not to exceed dynamic input range of the DAQ card's analog front-end (AFE) and the ADC, causing measurement to saturate and data to be invalid. Selection of the shunt resistor has significant effect on measurement accuracy, as too low value will result in a very small differential voltage over the shunt resistor and would not utilize full dynamic range of ADC and offset errors of amplifier and the ADC combined would be significant portion of measured result [15, p. 3]. Shunt resistor must also have a low temperature drift and good initial accuracy, preferably 0.5 % tolerance or less, to avoid inaccurate measurements without special shunt calibration process and when ambient temperature changes or the DUT is used in a thermal chamber for electrical validation purposes.

In addition to offset and gain errors, due to circuit limitations measurements at near-zero current are more inaccurate as even rail-to-rail operational amplifiers are typically unable to drive output all the way to ground level when used in a single positive supply configuration. This can be observed for example in the Analog Devices ADA4528 operational amplifier datasheet, which plots output low voltage versus output current [1, p. 10]. The ADA4528 was chosen for the DAQ card due to its low offset voltage error and output noise characteristics. Since necessary external negative supply rail biasing or use of true-zero type amplifiers that would overcome the limitation were not implemented on the power measurement card hardware, the circuit design is unable to measure zero current precisely. Another major design limitation of this circuit is ability to perform only unidirectional current measurement. Bidirectional current measurement would be useful when measuring battery charging or USB power rail (VBUS) in on-the-go (OTG) host mode but would require a more complex AFE circuitry. For those measurements, battery and USB VBUS power rails are usually easily accessible with other external measurement instruments and do not require integrated support from a DUT, mitigating the DAQ card hardware limitation.

A simplified hardware block diagram of the developed power measurement card is presented in Figure 3. The PnP DAQ card circuit is divided in three major functional groups. The AFE is responsible for differential input signal conditioning, input multiplexing and converting analog signals to digital sample values. Digital group handles circuit control signals, power management, communication and data processing. Both analog and digital groups have their own low noise supply voltage regulators consisting of a DC-to-DC boost converter stage followed by a low-dropout linear regulator (LDO). To further reduce switching noise coupled to supplies, more filtering is provided by ferrite beads on the individual LDO outputs. Used boost regulator input range accepts typical lithium battery chemistries with voltages from 2.7 V to 4.5 V, which means USB VBUS ranging from 4.75 V to 5.25 V or 5.5 V depending on USB implementation specification cannot be used directly [16]. The power measurement card can measure its own power consumption with a 16-bit integrated power monitor with a digital inter-integrated circuit (I2C) bus interface and a high-side current shunt resistor in series with the DAQ card power supply input.

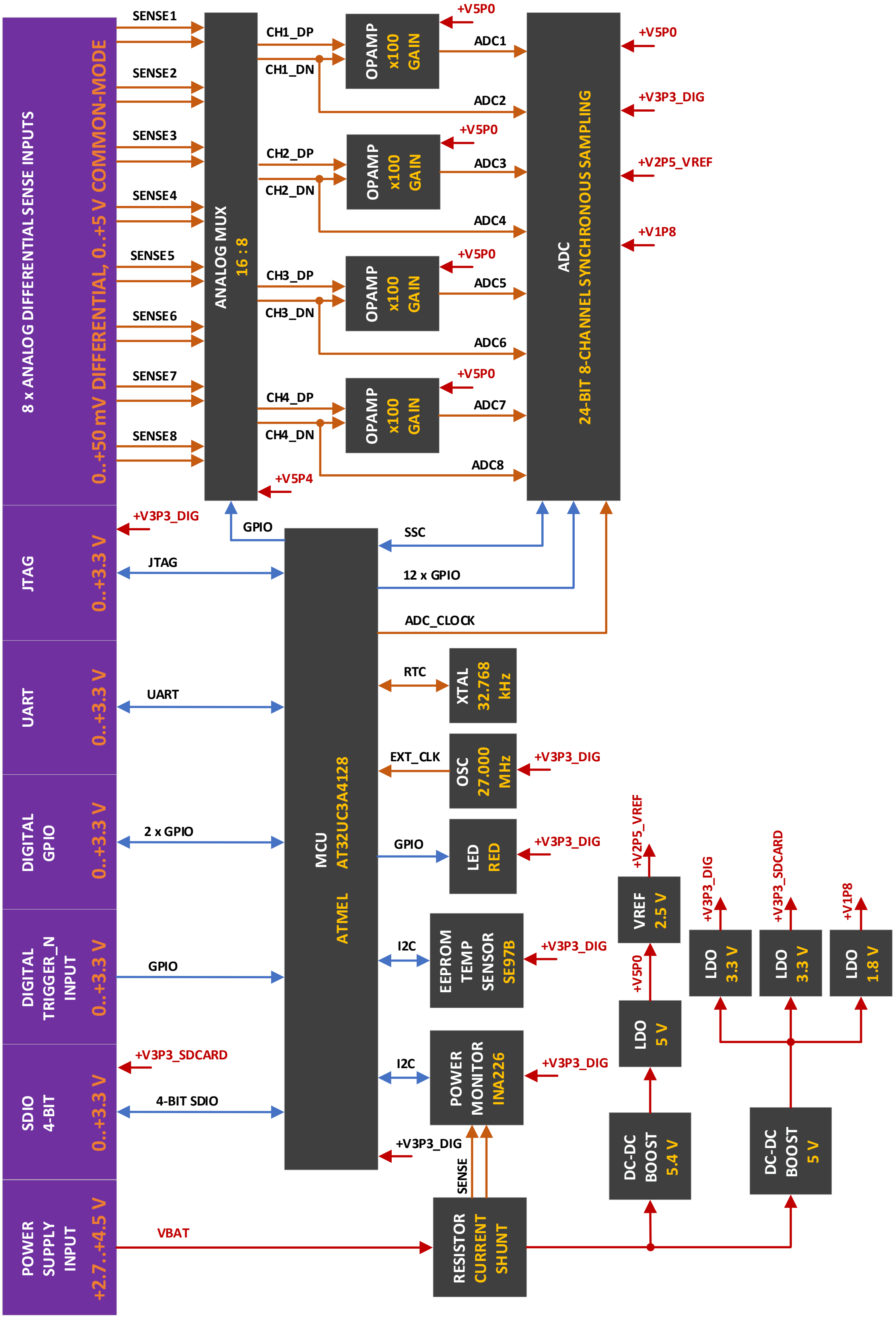


Figure 3. *A simplified hardware block diagram of the developed power measurement card. Level of detail shows all the major integrated circuits, power rails, digital and analog interfaces and clocks.*

Differential voltage from an external shunt resistor on a DUT is first going to an analog multiplexer component responsible of selecting group of eight signals from 16 inputs. Selection is done between two fixed groups of eight signals, effectively doubling available measurement rails from four to eight. After multiplexing, differential signals go to Analog Devices' ADA4528-2 operational amplifiers [1] working in differential amplifier configuration performing analog voltage gain of 100 and converting signal to single-ended, also acting as a buffer driving the ADC input. Load side of the differential sense signal also bypasses amplifier stage and is directly connected to one ADC channel for load voltage measurement. This way the undesirable voltage drop on the current sensing resistor is not affecting load power measurement. By not doing buffering on the voltage channels, circuit size could be optimized at the cost of lower input impedance of AFE, which was not critical in this application as input is always from a low impedance power rail. If a more generic DAQ operation was required, operational amplifiers could be added to the voltage channels too. A Texas Instruments' 24-bit ADS1278 ADC [14] was used on the power measurement card. It has eight differential inputs [14, p. 1], which are all referenced to a precision 2.5 V voltage reference (VREF) for input range of 0 V to 5 V, input signal being within +/- VREF from the reference level as required by the ADS1278 [14, p. 3]. At current shunt sensing gain of 100, this gives maximum of approximately 50 mV voltage drop over a shunt resistor and 5 V for maximum measurable load voltage. The ADC samples all the eight inputs simultaneously measuring DUT's power rail voltage and current synchronized in time. Four power rails are always measured at the same time and total number of available measurement channels is doubled by multiplexing input channels. This results in capability to measure up to eight power rails grouped in two alternating four rail time slots. Extract from the ADS1278 specifications [14] can be found in Appendix C.

The Atmel AVR32 architecture based AT32UC3A4 series microcontroller [7] and firmware running on it is responsible for controlling circuit components, data acquisition, writing and reading to a micro SD memory card with a 4-bit secure digital input output (SDIO) interface, I2C bus to an EEPROM and communicating with a PC via UART to USB connection. Block diagram and summary of the MCU specifications [7] can be found in Appendix B. A micro SD memory card stores measurement data in battery powered cordless use case. The MCU also operates a red indication light-emitting diode (LED) and three external general-purpose input/outputs (GPIOs), which are used for triggering data acquisition and two optional status LEDs on the DUT. The EEPROM chip has integrated I2C temperature sensor which is used for monitoring operational temperature of the DAQ card. The EEPROM is also used for storing calibration coefficients and unique serial number, which are remembered after power-loss due to non-volatile nature of EEPROM. The MCU is clocked from an external 27 megahertz (MHz) main oscillator (OSC) and a 32.768 kHz crystal (XTAL) for a real-time clock (RTC). The external 27 MHz clock is passed through the microcontroller's clock phase-locked loop (PLL) to the ADC’s sample clock input so that the MCU can change operating frequency of the ADC through firmware.

The power measurement card connects to a DUT with a 40-pin low profile Hirose DF40 series board to board (B2B) connector. The B2B connector passes through MCU programming joint test action group (JTAG) signals, UART signals, external digital trigger signal for measurement control, eight differential analog sense signals, two digital GPIOs and a few ground pins. Physical size of the power measurement card's 8-layer PCB is 40 x 15 millimeters and SMD components are placed on both sides. Small size and tight integration with the DUT hardware, coupled with reasonably low average power consumption of 400 mW per a DAQ card as listed in Appendix A specification of the PnP DAQ card, enables truly portable operation while allowing measurement of multiple power rails. A combination, which was not possible with existing readily available commercial solutions. Smallest SMD component footprint used is 0201 imperial for selection of resistors and decoupling capacitors. The used operational amplifiers, gain setting resistors, critical filtering capacitors, voltage reference and ADC were all selected with a focus on low temperature drift characteristics. Due to selection of high-quality components, unit cost of a single DAQ card was approximately 150 euros in low volume production with total solution cost of measuring 32 power rails summing up to approximately 600 euros. This is still very competitive compared to other commercial solutions especially when the feature set is considered and could be significantly reduced in volume production or with architectural design improvements. A photo of the final revision PnP DAQ card hardware is presented in Figure 4.

In total there were three hardware revisions, which were designed in parallel with firmware and PC measurement software development. First version PoC board design enabled validation of the design’s basic operation and firmware development, but extensive redesign of second revision boards added new features and changes, which had major impact on firmware design. Biggest changes were related to addition of EEPROM to store calibration information and redesign of analog front-end of the ADC. Third and final hardware revision was a minor correction to reduce unpowered input leakage and did not require any significant firmware changes.

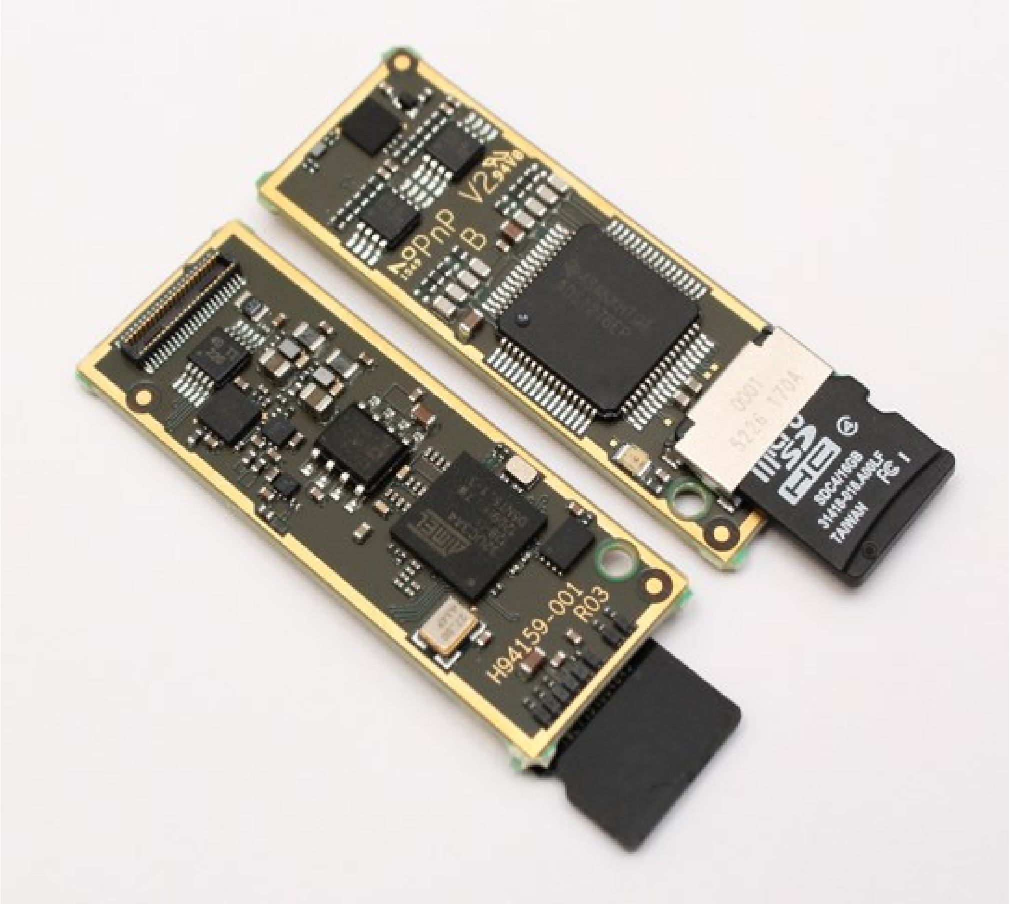


Figure 4. *Photo of the final revision PnP DAQ card hardware shows both sides of PCB fully assembled with components. The PCB has 8-layer any-layer micro via high-density stack-up to meet demanding size requirements.*

3.2 A DUT system

A simplified block diagram of the measurement system and a specific DUT platform is presented in Figure 5. Platform in this case means a product enabling development board used in product system design validation and not the final product FFD. Up to four DAQ cards are connect to the DUT with board to board connectors. In this specific case, the DUT has 32 current sensing shunt resistors in series with power rails. Measured rails could be for example a battery, SoC power rails supplied by PMIC's switching converters or LDOs going to platform peripherals. Differential sense voltages over the shunt resistors are connected to the DAQ cards' analog sensing inputs. The DAQ cards are powered separately from the DUT, through the DUT by connecting either an USB charging power bank or a wall charger capable of providing at least 1 ampere of peak current if four DAQ cards are used. This way power consumption of the DAQ cards does not affect DUT measurements in a significant way, although it should be noted that there is a few hundred µA current leakage to the DAQ card AFE. It is recommended that a low ripple switching DC-to-DC converter with a 4 V output and higher than 1 A peak current capability is used to create the DAQ card supply on the DUT. Significant inrush current could be a problem when four DAQ cards are connected in parallel to same power supply and therefore a soft-start circuit is recommended. Power supply input switching noise effects are greatly reduced by the DAQ card's two stage voltage regulation consisting of a boost switching converter followed by an LDO and ferrite bead filter.

The DAQ cards' UART interfaces go to a quad-channel UART-to-USB bridge chip and from there to a micro USB type B connector allowing data streaming and control with a PC measurement application. Start and stop of measurement can be triggered from the PC software over USB virtual serial port interface, a push button located on a DUT or from a platform SoC software controlled GPIO in active-low open-drain configuration. The UART connections should be power-gated with the DAQ card power supply preventing leakage to the DAQ card MCU pins when they are not powered. The DAQ cards limit input leakage current to analog sensing inputs when the DAQ card power supply is not present and do not require additional protection. Typical leakage current was measured to be below 100 µA per rail. While not optimal, this small leakage current poses no danger to either the DUT or the DAQ card hardware and allows normal operation of the DUT even without a DAQ card power supply.

Alternatively, the DAQ cards can be used as a standalone measurement system, if a DUT is not designed to support embedded DAQ cards. A standalone board needs to provide power to the DAQ cards and have an UART converter for data USB connection to a PC. In this case, the DUT system must still have current sensing resistors and a way to connect sense signals to external systems, such as the standalone measurement system via wires, cables or a flexible PCB.

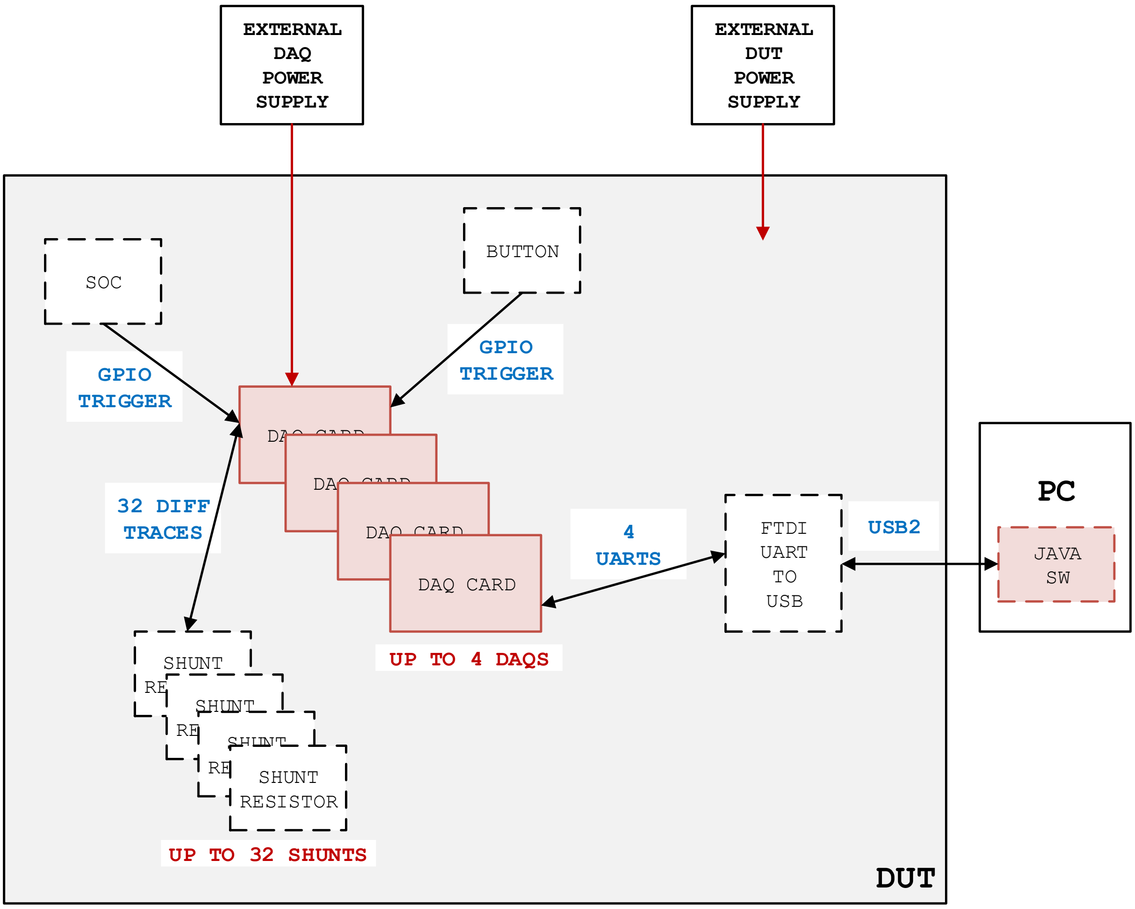


Figure 5. *A simplified block diagram of the developed power measurement system and how it is connected to a typical DUT. Each DUT is designed from the start to support the DAQ cards by having required supporting components and connectors.*

4. DEVELOPMENT ENVIRONMENT

This chapter introduces the used software development tools and the development environment. Hardware setup, such as a programmer and initial microcontroller evaluation board, is identified.

4.1 Firmware

Programming of the MCU firmware was done with an Atmel ICE JTAG programmer and software. Picture of the programmer is shown in Figure 6. [8] A DAQ card was installed on a purpose-built breakout board, which provided target device with power and allowed access to MCU's JTAG programming pins. At first, the factory default bootloader was erased and then firmware was programmed and verified on the target device.

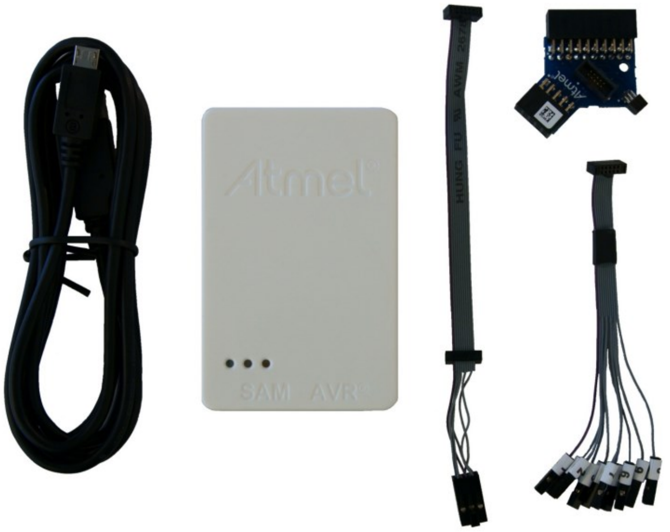


Figure 6. *Atmel ICE JTAG programmer and adapter cables [8, p. 7] were used for programming the target MCU and doing in-system debugging.*

An Atmel Xplained series evaluation board [11] was used as a starter platform, on which firmware basic structure with UART debugging could be built and tested before the first proof-of-concept DAQ card hardware was available. Picture of the development board is shown in Figure 7. I2C bus functions and external GPIO interrupts were also implemented early on and were for the most part immediately working with the target hardware proving value of using an evaluation board for early firmware development to reduce hardware bring-up time significantly.

Moving the project from A3 revision MCU on the evaluation board to A4 on the target hardware required some modifications on the Atmel libraries because existing driver libraries were not yet fully compatible with the new A4 series. For the most part, A3 and A4 series are firmware compatible with each other [9], so moving project from AT32UC3A3256 to AT32UC3A4128 device was easy and did not take longer than a day. Most work was due to target MCU having less integrated flash memory than the one on evaluation board and therefore having slightly different configuration header files.

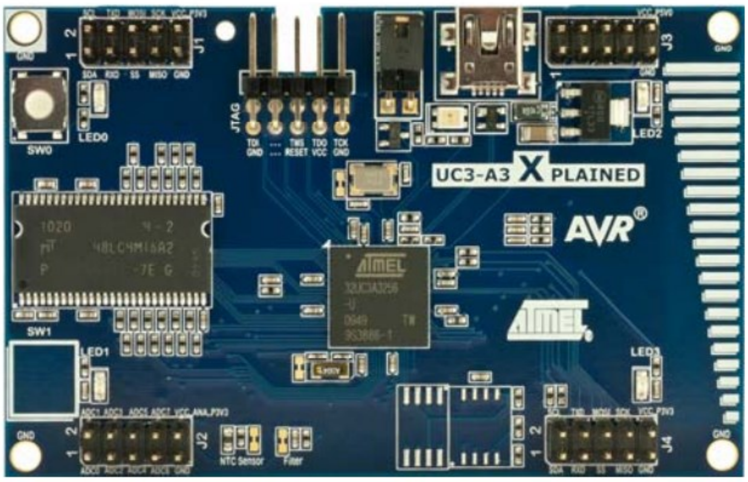


Figure 7. *Evaluation board for AT32UC3A3256 microcontroller [11, p. 1] was used to start firmware development before target hardware was available bringing forward project schedule by a few weeks.*

Atmel Studio 6 integrated development environment (IDE) was used for developing firmware code, compiling binaries and programming MCUs. The Atmel Studio supports all of Atmel’s AVR and SAM series microcontrollers. The Atmel Studio allowed easy integration of Atmel Software Framework’s (ASF) ready driver libraries [9] to the C language project. In the end, the firmware project was upgraded to a newer Atmel Studio 7 IDE version, which did not require any additional modifications. [10]

Atmel provides extensive driver libraries for their microcontrollers. Included are many example projects demonstrating use of hardware peripherals helping developers to get started with their own firmware designs. Integrated support on the Atmel Studio helped with integrating needed ASF software libraries and keeping them up-to-date. Full ASF support for new MCU models can take some time and may require some manual integration work from the developer as was the case in this project work. [9]

4.2 PC measurement software

Netbeans IDE was used for developing a graphical Java PC measurement software for the developed DAQ card. Netbeans' built-in graphical user interface (GUI) builder tool was used for making a simple Java Swing user interface, requiring less manual user interface programming and decreasing total development time. [12] Downsides of using the GUI editor were ugly machine generated code blocks and hard maintenance of the GUI design making interface structure more difficult to modify in the future. Advantages of the GUI builder outweighed disadvantages in a one-man agile project, being the only way to meet aggressive delivery deadline when starting software development from a scratch.

Pre-requisites for installing the developed software include virtual serial port drivers, which are needed for communication with a Future Technology Devices International Ltd. (FTDI) UART-to-USB bridge chip used between the DAQ cards and the PC [2]. RXTXcomm Java libraries [5] were used for adding serial port communication support to the measurement application. The RXTXcomm library supports both Windows and Linux operating systems [5], and since it was the only third-party Java library needed, the developed PC measurement software has no problem supporting the required operating systems.

5. EMBEDDED FIRMWARE

This chapter describes the power measurement card firmware design and functional implementation with the help of state and flow charts. Software resources are mapped to hardware requirements and Atmel ASF drivers [9] are utilized where applicable to reduce development time.

5.1 Mapping HW/SW resources

Firmware peripheral driver components from Atmel Software Framework libraries [9] were imported to project based on interfaces described in hardware block diagram in Figure 3 and microcontroller resources available [7, pp. 1-5], which can be found also in Appendix B extract from the MCU datasheet. These drivers implement functionality to utilize all the DAQ card's hardware resources. High-level peripheral driver content diagram of the developed firmware is shown in Figure 8.

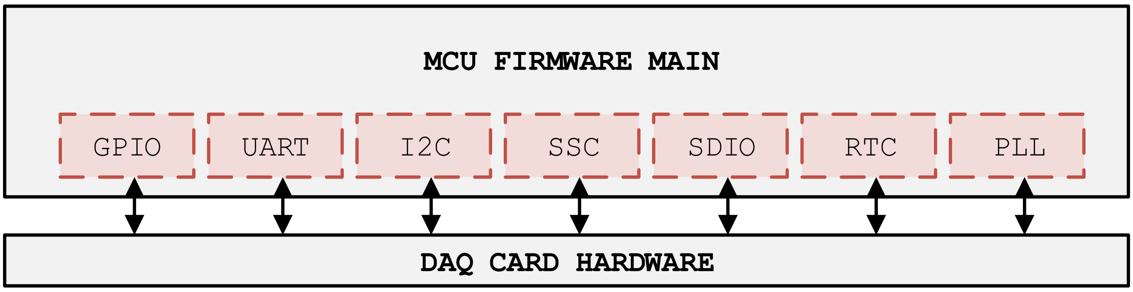


Figure 8. *Firmware functions were built on top of the Atmel ASF peripheral drivers [9].*

An UART interface handles serial communication with a PC measurement software over virtual USB serial port. PLL driver controls MCU main oscillator input and buffering of the clock output to the ADC. Measurement timestamp information is provided by RTC driver. SDIO driver utilizes a micro SD card in raw block data format for datalogging purposes. A synchronous serial interface is configured in a frame-sync mode to interface with the ADC data output. Some GPIOs were utilized for circuit control, and support for an external interrupt was added to trigger input signal. The firmware functions which used hardware peripherals were written on top of the ASF driver stack [9].

5.2 Functional description

A simple power management state machine of the developed MCU firmware is presented in Figure 9. The state machine has total of three states, which implement necessary functions controlling the hardware. After a power-on reset (POR) event, MCU enters *'init'* state in which hardware and digital interfaces are configured.

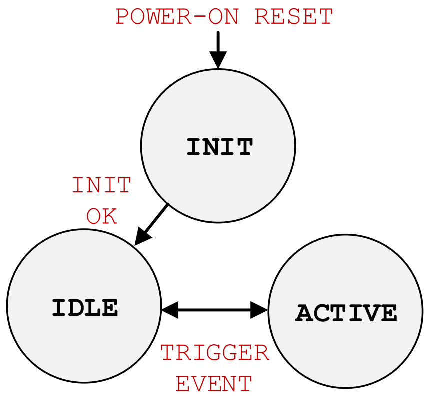


Figure 9. *Power management state machine of the developed firmware.*

State transitions between idle and active states are controlled by a trigger event, which can be either an external GPIO interrupt or an UART command modifying state variable. The GPIO interrupt implements a simple one-second signal de-bouncing preventing any unwanted glitches caused by a button press. Based on actual user experience, the button de-bouncing worked well.

Functional flowchart in Figure 10 shows how firmware executes *'init'* state after POR. At first, power rails are enabled to meet power sequencing requirements of the hardware. Then MCU clocks are switched to an external 27 MHz oscillator and a 32 kHz RTC XTAL. Next, UART communication is enabled and firmware begins to print out debug information regarding to power-up status. The MCU connects to several GPIOs, which are required to be in a pre-defined state. Firmware configures the pins from a default high impedance input mode [7, p. 19] to what is needed by the hardware implementation. The GPIO pins can have multiple different alternate functions multiplexed on them [7, pp. 8-12], and they need to be mapped to the specific driver functions in firmware code.

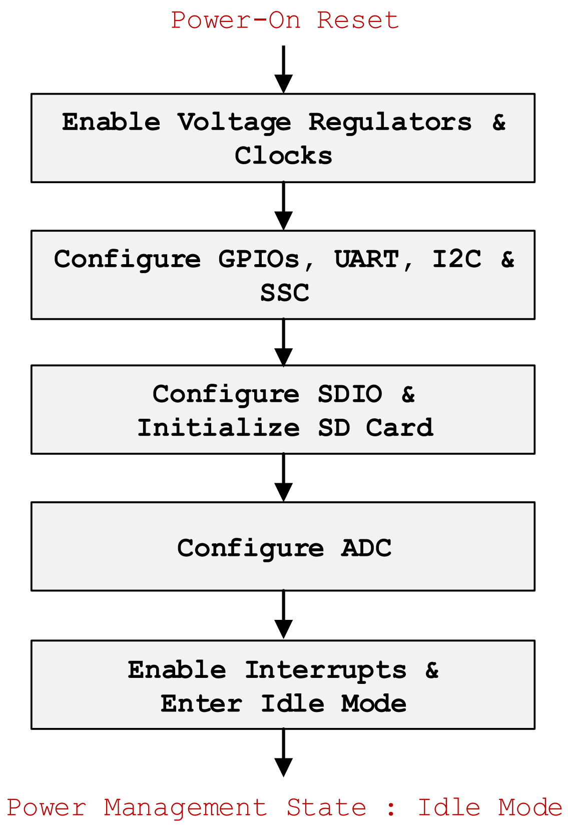

Power-On Reset
Enable Voltage Regulators & Clocks
Configure GPIOs, UART, I2C & SSC
Configure SDIO & Initialize SD Card
Configure ADC
Enable Interrupts & Enter Idle Mode
Power Management State : Idle Mode

Figure 10. *Functional flowchart of the init state.*

A red colored status LED is turned on before more complex hardware interfaces are initialized, indication configuration is in progress. Firmware enables I2C bus and performs a simple self-check by trying to communicate with all I2C components and receive acknowledgements from their slave addresses. However, the slave device register content is not read and verified. After that, firmware enables SDIO interface and tries to establish communication with a micro SD memory card. Since the SD card is needed by following firmware operations by default, firmware halts execution and waits until the SD card is successfully initialized. Firmware periodically tries to power cycle the SD card until the SD card is successfully detected. A red LED remains on indicating user the DAQ card failed to initialize peripherals, and the SD card can be hot-plug removed or inserted at this point. After inserting a functional SD card, the firmware continues forward. After successful peripheral initializations, GPIO interrupts are enabled for external triggering support. A synchronous serial controller (SSC) is configured as a frame-sync interface between MCU and ADC [7, pp. 508-545], and the ADC is initialized to a correct operating mode with control GPIOs. Firmware then automatically enters *'idle'* state and waits for interrupt by incoming UART commands or external trigger event.

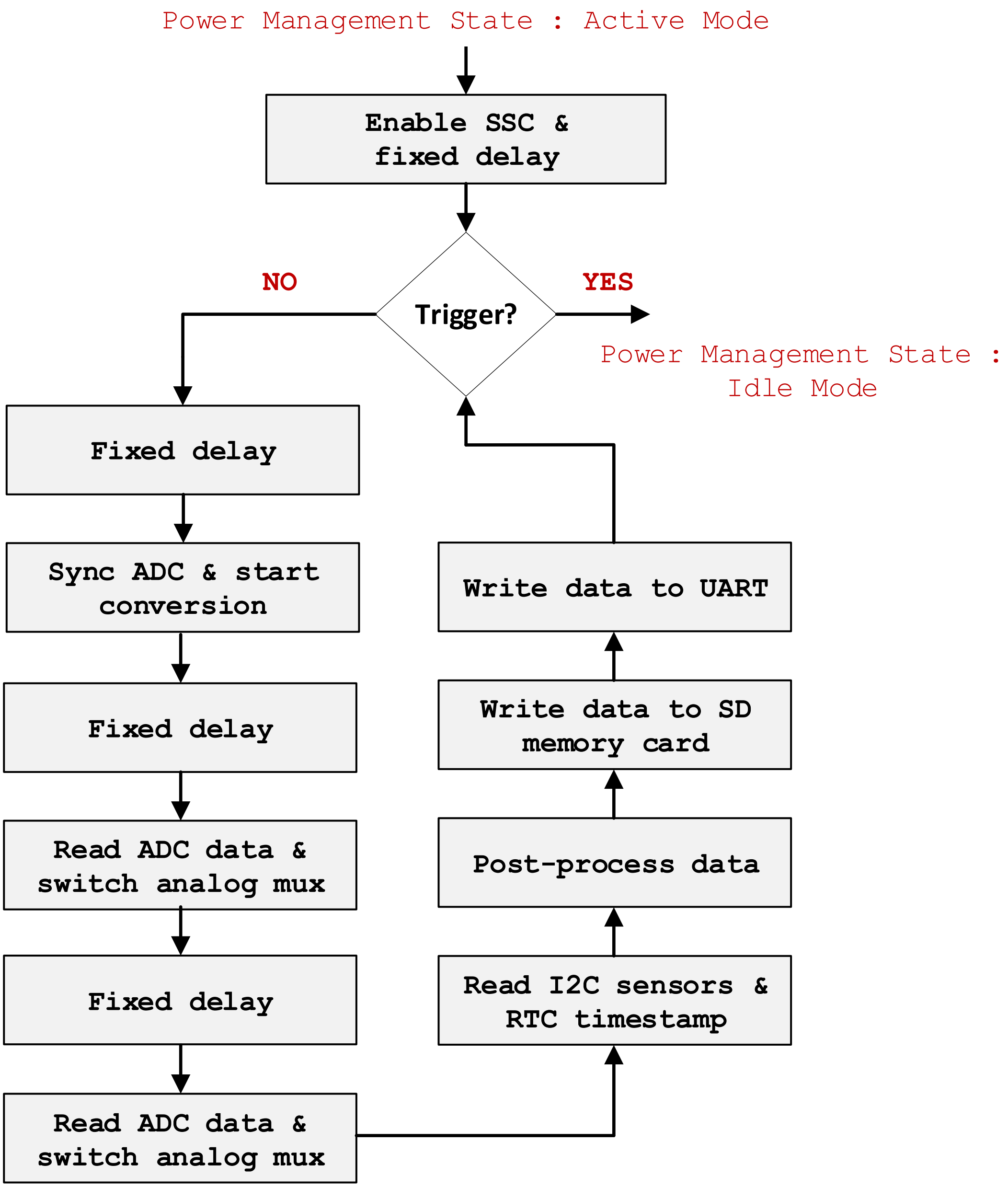

Power Management State : Active Mode
Enable SSC & fixed delay
Trigger?
NO
YES
Power Management State : Idle Mode
Fixed delay
Sync ADC & start conversion
Fixed delay
Read ADC data & switch analog mux
Fixed delay
Read ADC data & switch analog mux
Write data to UART
Write data to SD memory card
Post-process data
Read I2C sensors & RTC timestamp

Figure 11. *Functional flowchart of the active state.*

Functional flowchart in Figure 11 shows *'active'* state execution flow. When the firmware enters *'active'* state, all ADC channels and frame-sync interface is enabled. After a short delay allowing ADC time to exit shutdown mode, measurement loop is started. MCU sends a synchronization pulse to the ADC starting conversion [14, pp. 27-28], while frame-sync interface is ready to receive data. A fixed delay is waited until ADC output data is valid [14, pp. 27-28], and eight successive samples from all eight ADC channels are read to a software buffer by polling SSC receive data register when data ready bit is set [7, pp. 508-545]. This was by far the most timing critical part of firmware code and required a lot of work to get it right, as MCU's SSC block had trouble keeping up with ADC running at same 27 MHz clock as MCU without enabling hardware direct memory access (DMA), which was not practical due to project schedule. An analog multiplexer (MUX) is then switched, and a fixed delay is waited until AFE stabilizes and ADC outputs stable data. Eight samples are again read from all channels to a software buffer by polling SSC receive data register. Analog mux is switched again to connect previous channels to the AFE. After reading analog inputs with main ADC, I2C power and temperature sensors and RTC time counter values are read.

Raw ADC output data is post-processed by averaging eight consecutive samples to a single data value. Possible zero crossing in two's complement data is detected and taken in to account while making averaging by a three-bit shift-right operation. Data is then written to a micro SD card, if it has not been specifically disabled by UART command. Measurement data is always streamed to the UART interface while performing conversion from two's complement to one's complement on the fly for main ADC data. At start of each loop iteration, breakpoint checks if state variable has been changed by either external GPIO interrupt trigger event or UART command. If trigger is received, firmware exits '*active*' state and enters *'idle'* state after completing the ongoing measurement loop cycle. Interrupts are disabled during a measurement loop iteration to prevent timing critical parts of the firmware execution from being halted by UART or GPIO event.

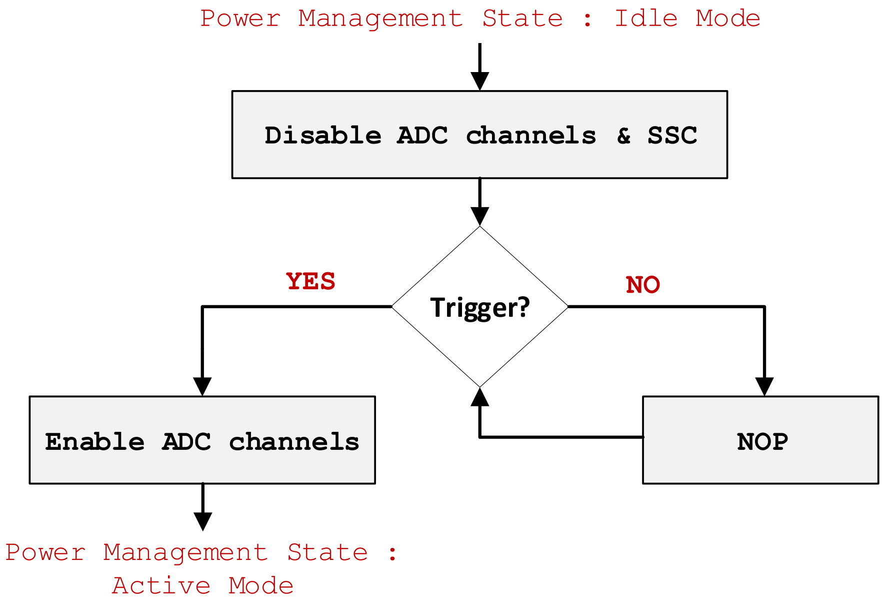

Power Management State : Idle Mode
Disable ADC channels & SSC
YES
Trigger?
NO
Enable ADC channels
NOP
Power Management State : Active Mode

Figure 12. *Functional flowchart of the idle state.*

Functional flowchart in Figure 12 shows *'idle'* state execution logic. When firmware enters *'idle'* state, status LED is turned off and all ADC channels are disabled for power savings with dedicated ADC channel shutdown control GPIOs [14, p. 29]. Firmware waits in a no operation loop until state variable change by trigger event is detected. After loop break, the ADC channels are enabled, status LED is turned on and firmware enters *'active'* state.

5.3 Configuration

After POR event occurs and MCU is powered up, firmware switches internal low-frequency oscillator to a 27 MHz external oscillator and configures RTC to produce time ticks at 1024 ticks per second rate using external 32.768 kHz XTAL as a reference. Firmware then performs MCU physical pin to function mapping and enables interrupt logic on UART receiver and external trigger GPIO. [7, pp. 1-19]

The Texas Instruments ADS1278 ADC is controlled with a selection of GPIOs, which are used for selecting operating mode, data format, enabling sampling channels and synchronization to the rest of the hardware. Data from the eight ADC channels is configured in time-division multiplexed (TDM) dynamic mode by setting format pins to "011" and received by the MCU via a Synchronous Serial Controller (SSC) interface in a frame-sync mode. A 27 MHz main sample clock frequency of the ADC is generated by buffering MCU's external 27 MHz oscillator with a PLL on the MCU. This allows MCU to synchronize precisely with the ADC and select desired sampling clock frequency depending on the ADC configuration used. Mode pins are set to "10" and clock divider to '1' to select ADC low-power mode and maximum of 27 MHz clock. All eight ADC channels are enabled by setting logic '1' on each active-low power-down pin. Configuration bits are explained in detail in the ADS1278 datasheet. [14, pp. 26, 30, 33]

The MCU's SSC frame-sync interface is configured by setting appropriate bits in the SSC configuration registers for a 24-bit data length and frame-sync signal active-high pulse polarity. These registers are explained in Atmel AVR32UC3A series datasheet. [7, pp. 508-545] The SSC is also configured to meet timing requirements of the ADC as described in ADS1278 datasheet. Active-low sync pulse allows MCU to synchronize ADC to switching event of an analog multiplexer in the ADC input path. This sync pulse can be asynchronous to the ADC clock if one clock cycle variance is allowed in delay to valid output data. After each sync pulse, a fixed delay of at least 128 clock cycles is waited before ADC sample data is valid and received by MCU. This requirement is detailed in ADS1278 datasheet and extract from it in Figure 13, which can also be found in Appendix C. [14]

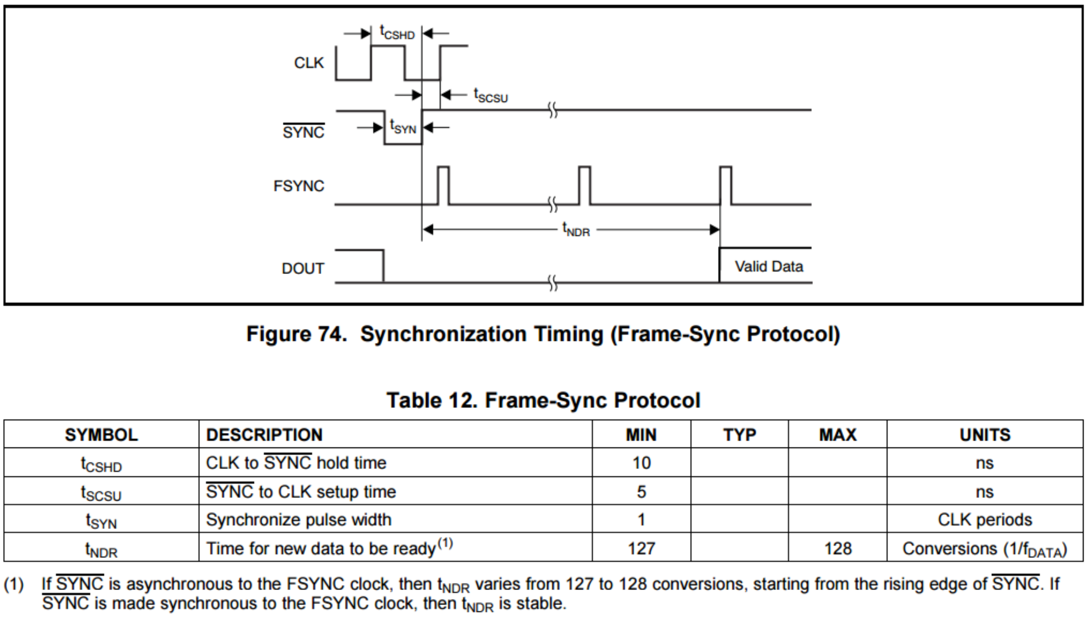


Figure 74. Synchronization Timing (Frame-Sync Protocol)

Table 12. Frame-Sync Protocol

SYMBOL	DESCRIPTION	MIN	TYP	MAX	UNITS
t_{CSHD}	CLK to $\overline{\text{SYNC}}$ hold time	10			ns
t_{SCSU}	$\overline{\text{SYNC}}$ to CLK setup time	5			ns
t_{SYN}	Synchronize pulse width	1			CLK periods
t_{NDR}	Time for new data to be ready(1)	127		128	Conversions ($1/f_{DATA}$)

(1) If $\overline{\text{SYNC}}$ is asynchronous to the FSYNC clock, then t_{NDR} varies from 127 to 128 conversions, starting from the rising edge of $\overline{\text{SYNC}}$. If $\overline{\text{SYNC}}$ is made synchronous to the FSYNC clock, then t_{NDR} is stable.

Figure 13. *Frame-sync protocol timing requirements of Texas Instruments ADS1278 analog-to-digital converter set timing constraints to firmware implementation [14, p. 28].*

Microcontroller's I2C master was configured to work at 400 kHz frequency. There were some minor differences in communication protocol required by the I2C power monitor and EEPROM ICs, which meant some modifications to the example ASF I2C source codes [9] had to be done taking in to account for example a repeated start condition.

The UART interface was configured at baud rate of 1.152 Mbps, which was tested to be the highest working speed when used with a FT4232H quad UART-to-USB bridge. Other parameters were set as no parity bits, normal polarity and eight data bits. The UART receive was handled by interrupt routine while transmit was done blocking way by polling.

Simple software de-bouncing was added on external trigger pin, which was configured as interrupt event when voltage level transitions from high to low. DUT system connects to the trigger pin via open-drain transistor, pull-up resistor being on the DAQ card side. Measurement is triggered on or off if MCU side pin is low for one second or more. Retriggering will not happen until pin state has returned to high and again low over one second de-bouncing period, preventing accidental multiple triggers by driving trigger signal low for long durations.

Microcontroller interfaces to a SD memory card via a 4-bit data width SDIO bus. SDIO driver automatically detects highest supported speed of the memory card. Physical hot-plug detect feature was not used and card detection was done by polling in initialization phase of firmware after POR event.

Voltage regulator power sequencing and timing for SD card and analog front-end proved challenging as having up to four cards in parallel powering up at the same time caused significant inrush current from DAQ card’s power supply seen in measurements performed with hardware. The DAQ card hardware provided minor soft-start functionality with a resistor and a capacitor delayed enables of voltage regulators, but additional measurements and firmware timing changes were done to minimize startup current. This improved functionality with some of the cheaper USB power bank models, which would otherwise trigger current limitation and the DAQ card’s boost voltage regulators would fail to start. When used with a wall charger or power supply capable of delivering at least one ampere of current, cards would always start reliably even before soft-start delays were added.

5.4 Arithmetic calculations

This section details various arithmetic calculations, which are performed by the developed DAQ card firmware and the PC measurement software. Firmware needs to convert raw ADC data, which is in two’s complement format, to one’s complement for other calculation. The ADC’s ideal output codes related to input voltage levels are detailed in ADS1278 datasheet and in Figure 14 . [14, p. 24]

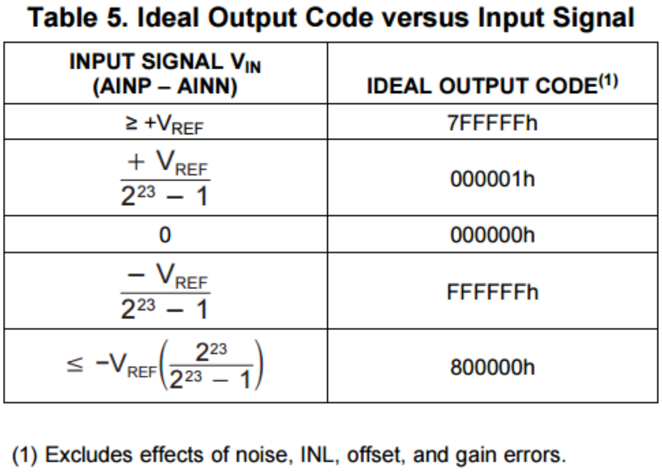


Table 5. Ideal Output Code versus Input Signal

INPUT SIGNAL V_{IN} (AINP – AINN)	IDEAL OUTPUT CODE[1]
$\geq +V_{REF}$	7FFFFFh
$\frac{+V_{REF}}{2^{23}-1}$	000001h
0	000000h
$\frac{-V_{REF}}{2^{23}-1}$	FFFFFFh
$\leq -V_{REF}\left(\frac{2^{23}}{2^{23}-1}\right)$	800000h

[1] Excludes effects of noise, INL, offset, and gain errors.

Figure 14. The ADS1278 ADC’s input signal ranges and corresponding ideal digital output codes assuming no error sources [14, p. 24].

Firmware converts raw ADC sample data before pushing it to the PC measurement software via UART. Because the 24-bit ADC inputs are referenced to a VREF instead of ground, the data range is from -8388608 to 8388607 zero point being at 2.5 V VREF input. Negative two’s complement values are detected by checking if they are higher than 0x7FFFFF in value meaning the most significant bit is one. If the two’s complement value is negative, the negative sign is printed to the UART connection and conversion to one’s complement is done by executing bitwise exclusive-or operation with 0xFFFFFF and the data and adding one. Positive values can be printed without conversion. UART message format used is signed decimal values represented in American Standard Code for Information Interchange (ASCII). The conversion of raw ADC sample data is done with C language code presented in Program 1.

```
// Get 24-bit ADC’s raw two’s complement data
uint32_t adc_data = adc_data_twosc;

// Detection of negative two’s complement value (MSB ‘1’)
if( adc_data > 0x7FFFFF )
{
      // Print negative sign to UART
      uart_print( “-“ );
      // Conversion to ones’ complement
      // Exclusive OR operation
      adc_data ^= 0xFFFFFF;
      // Add one
      adc_data += 1;
}
// Print to UART as ASCII string
uart_print( adc_data );
```

Program 1. *Conversion of 24-bit two’s complement ADC sample output data to ones’ complement and printing to UART interface in ASCII string format.*

When firmware performs averaging of eight ADC samples, a bit-shift right by three bits is executed instead of a division to potentially optimize performance. Program 2 shows the C language code for the shift operation. A bit-shift operation can be used here, and its performance does not rely on compiler utilizing hardware divider unit, if available.

```
adc_averaged_sample = adc_eight_samples_sum >> 3;
```

Program 2. *Divide by eight by doing a three-bit shift-right operation.*

The PC software receives ADC sample data ranging from -8388608 to 8388607. Digital output code is converted to voltage V with Equation (3), which is partially given in ADS1278 datasheet [14, p. 24] and Figure 14 and modified to include analog input reference offset of DAQ card hardware design.

$$V = ADC_{code} * \left(\frac{V_{ref}}{2^{23}-1}\right) + V_{airef} \, , \tag{3}$$

where ADC_{code} is a digital output code value from the ADC, V_{ref} is reference voltage of the ADC, and V_{airef} is analog input reference voltage connected to negative pin of differential input of the ADC channel. For example, when the ADC digital output code reads 2796000, V_{ref} is 2.5 V and V_{airef} is set to 2.5 V by the DAQ card hardware design, the Equation (3) gives voltage of approximately 3.33 V. Current I_{shunt} flowing through shunt resistor is calculated with Equation (4) based on Ohm's law Equation (1) and including voltage gain A_v of added amplifier in the signal path.

$$I_{shunt} = \frac{\frac{V_{shunt}}{R_{shunt}}}{A_v} \ , \tag{4}$$

where V_{shunt} is voltage over shunt resistance R_{shunt} and A_v is the ADC input amplifier voltage gain, which is set by the DAQ card hardware to a fixed value of 100. For example, when voltage over shunt is 2.5 V and the shunt resistance is 25 mΩ, the Equation (4) gives a current value of 1 ampere. Power rail voltage measurement error is removed and final calibrated rail voltage V_{cal} is calculated by applying gain and offset correction coefficients with Equation (5).

$$V_{cal} = V_{measured} * A_{cal} + V_{offset} \ , \tag{5}$$

where $V_{measured}$ is voltage measured by ADC, A_{cal} is gain correction coefficient and V_{offset} is offset error correction coefficient. Similarly, shunt current calibration is done by applying gain and offset correction coefficients by Equation (6).

$$I_{cal} = I_{measured} * A_{cal} + I_{offset} \ , \tag{6}$$

where $I_{meaasured}$ is measured shunt current, A_{cal} is gain correction coefficient and I_{offset} is offset error correction coefficient. Calibration procedure is further explained in Chapter 5.6.

5.5 UART API

To support command and control of the DAQ card from a PC measurement application, a simple UART API was implemented. The API contains 17 basic commands, each meant to enable required base functionality for targeted use cases. More commands are easy to add to firmware UART receive parser if additional features are needed in the future. The supported firmware UART API commands are listed in Table 2.

Table 2. *Supported UART API commands of the developed power measurement card.*

# Command	Description
1.	Software system reset
2.	Trigger (toogle)
3.	Trigger (start)
4.	Trigger (stop)
5.	Read SD card content
6.	Reset RTC counter
7.	SD card writing (disable)
8.	SD card writing (enable)
9.	Read EEPROM content (all)
10.	Clear SD card data
10.	Read SD card type and size
12.	Force slow SD card read speed
13.	Read system status register
14.	LED (on)
15.	LED (off)
16.	Read EEPROM register
17.	Write EEPROM register

Each ASCII format string command packet consists of three parts. First is the command code, then the parameter code and finally the data. The commands are terminated with a '\r' carriage return escape sequence. When firmware receives a command, a corresponding reply is sent back to the host application. All reply packets start with a string "!R" for easy parsing on the receiving PC software side. A return packet specifies received command code, and if the command was a data query, corresponding data is also sent. Table 3 describes command and Table 4 reply packet format. During active measurement, data packets are always pushed to UART by firmware. Data packet format is different from commands, and message is surrounded with starting '[', and ending ']' brackets and has fixed positions for voltage and current measurement results from all eight channels separated by dots as delimiters. Packets being pushed to UART without a request command from the PC software is by design to reduce downstream traffic from the PC host and maximize the upstream data rate, which is crucial to performance of the DAQ card. Streaming data packages are identified with brackets and have a fixed format with a specific amount of separator characters for easy parsing and basic error detection. There is no error detection or checksum calculation built-in to the firmware, so this falls entirely to the PC application's responsibility. To make communication more robust, error checking could be added to the commands in future. This is not strictly required though, as the system proved to work without glitches by relying on basic sanity checks of received data format and value ranges by the PC application.

Table 3. *ASCII string format of the DAQ card UART command packets. All command packets are terminated with a carriage return escape sequence. For certain command packets, the data field may be longer than a single byte.*

START	CMD ID	DELIMETER	PARAM	DELIMETER	DATA	STOP
C	12	< DOT >	34	< DOT >	56	\r
	HEX		*HEX*		*HEX*	*CARRIAGE*
CHAR	*1 BYTE*	*CHAR*	*1 BYTE*	*CHAR*	*1 … n BYTES*	*RETURN*

Table 4. *ASCII string format of the DAQ card UART reply packets. All reply packets are terminated with a carriage return escape sequence. For certain reply packets, the data field may be longer than a single byte.*

START	DELIMETER	CMD ID	DELIMETER	DATA	STOP
!R	< DOT >	12	< DOT >	34	\r
		HEX		*HEX*	*CARRIAGE*
2 CHARS	*CHAR*	*1 BYTE*	*CHAR*	*1 … n BYTES*	*RETURN*

5.6 Calibration

The DAQ card firmware was built to support calibration of both voltage and current measurement during manufacturing by storing calibration coefficients to I2C EEPROM. Calibration is divided in two main components, the offset and the gain error. Maxim's application notes detail how these error sources affect unipolar ADC measurements, which is the configuration used in the designed DAQ card. Figure 15 illustrates how negative and positive offset errors reduce dynamic range of the ADC. [6]

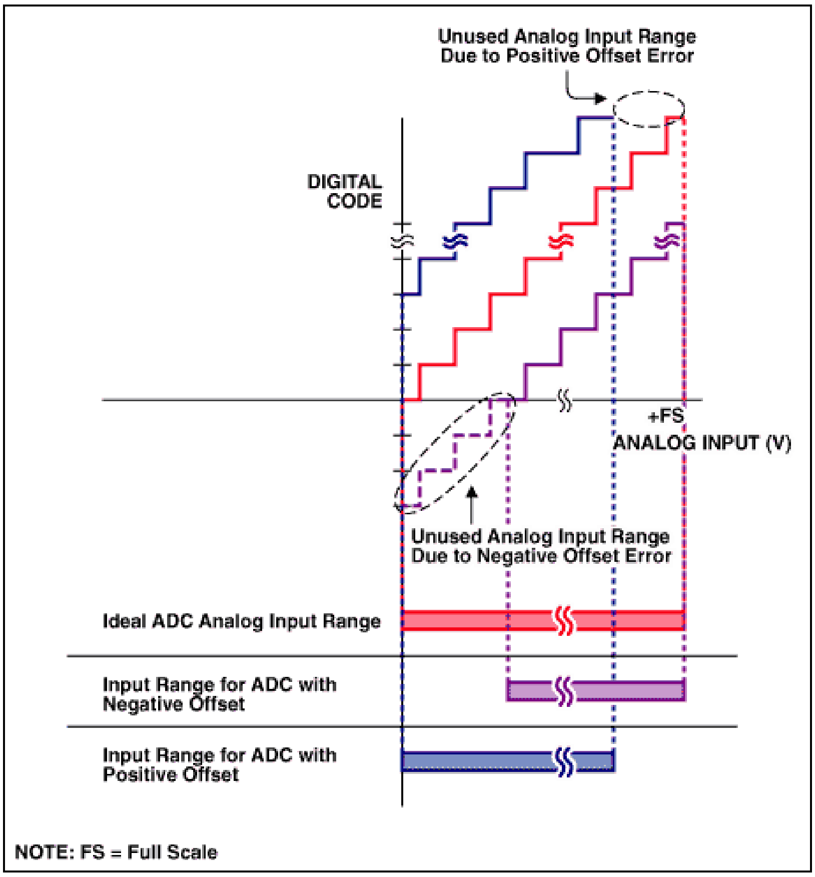

Unused Analog Input Range
Due to Positive Offset Error
DIGITAL
CODE
+FS
ANALOG INPUT (V)
Unused Analog Input Range
Due to Negative Offset Error
Ideal ADC Analog Input Range
Input Range for ADC with
Negative Offset
Input Range for ADC with
Positive Offset
NOTE: FS = Full Scale

Figure 15. *Negative and positive offset errors reduce unipolar ADC full-scale input voltage range as explained by Maxim Integrated in an application note [6].*

To calculate calibration coefficients for offset and gain, each DAQ card had to be individually tested against a reference instrument. By measuring voltage at two or more steps, coefficients can be calculated to first remove the offset and then adjust gain to match more closely ideal ADC. Figure 16 illustrates this process of eliminating offset and gain errors. In the case of the developed DAQ card, measurement error is driven by initial offset error when ADC input is low, and gain error dominates as input increases closer to maximum. On all tested DAQ cards, voltage measurement channels had a small positive offset error and current measurement channels had a small negative offset error. Initial gain errors were typically below 0.5 % of ADC full scale range. [6]

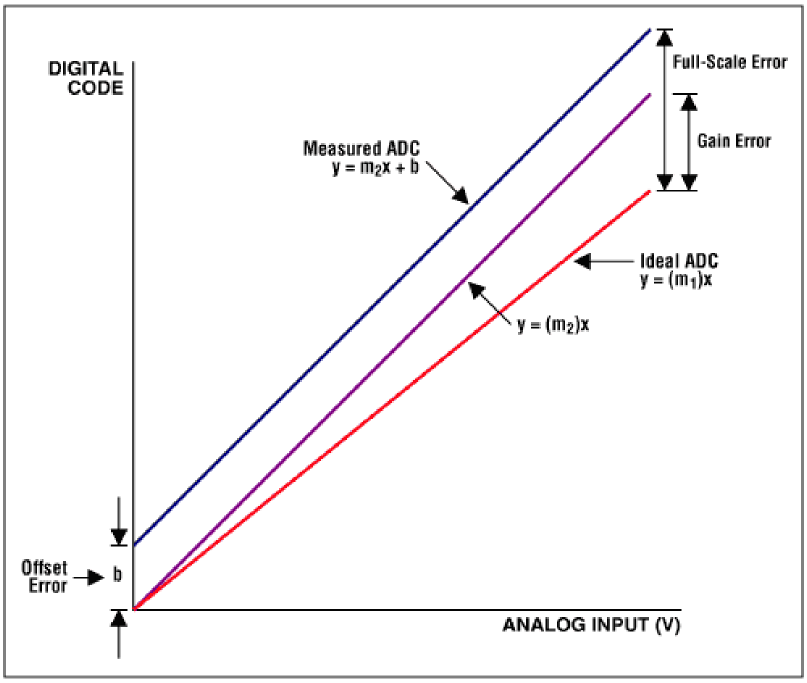


Figure 16. *Gain and offset errors of an ADC can be measured, and software calibrated to improve measurement accuracy [6].*

It is important to note that depending on hardware component thermal drift characteristics, the measurement instrument output may drift over ambient temperature change. For this reason, it may be required to make calibration measurements at a few different ambient temperatures. The DAQ cards were calibrated in a typical operating temperature of approximately 40 degrees Celsius (°C) and low thermal drift components guaranteed staying within specifications at operating temperature range from -20 to 60 °C. This was validated by thermal chamber testing. When correct coefficient values are known, they are written to EEPROM memory chip by sending EEPROM data write commands from PC software to DAQ card firmware. The firmware automatically verifies that EEPROM register value is successfully written by reading it back and comparing to given data input. Each of the eight measurement channels has its own gain and offset coefficients. Similarly, current measurement offset and gain errors can be found by measuring two different test currents at opposite ends of input range, calculating coefficient values and writing them to EEPROM. Calibration coefficients for all voltage and current measurement channels are read by PC software when it is started, and coefficients are applied to all data received from the DAQ card. This means the responsibility of applying calibration is of the host PC application, and DAQ card only sends raw ADC data out. The coefficients are applied according to Equation (5) and Equation (6) in Chapter 5.4. In the developed measurement system, only one gain and offset coefficient was used for each voltage and current channel. This could be improved in future designs with multiple coefficients covering different parts of the input range and operating temperature for greatly enhanced measurement accuracy.

6. PC MEASUREMENT SOFTWARE

Power measurement card PC measurement software was written with Java programming language utilizing graphical Swing libraries. Use of Java enabled easy portability of software to both Linux and Windows operating systems. Focus with PC software was to keep it simple and provide only necessary features for planned measurement use cases.

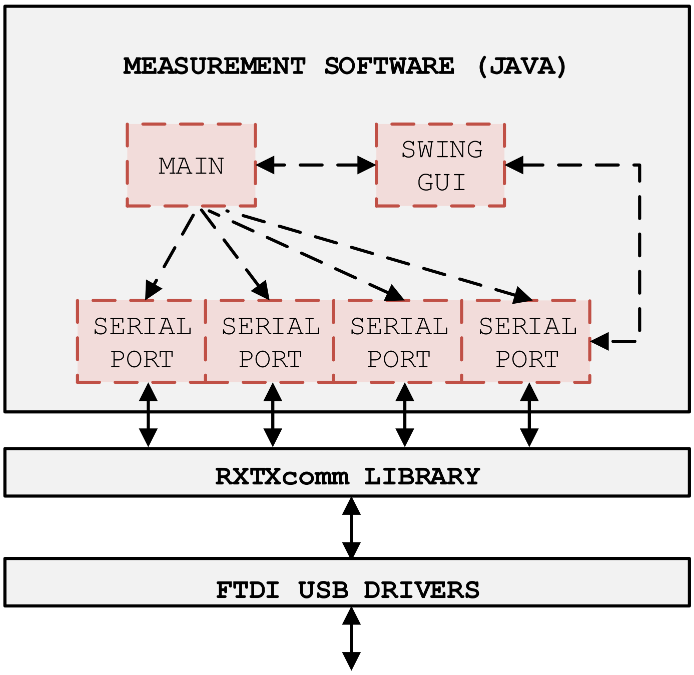


Figure 17. *Instance diagram of the developed PC measurement application.*

High-level instance diagram of the developed Java PC application is shown in Figure 17. Application consists of main class, Swing GUI and four serial port class instances which interact with external RXTXcomm library [5] enabling communication with up to four DAQ cards. Each of the serial ports handles received data writing to a CSV format log file and pushes data to the Swing GUI updated four times a second to keep it human readable. Updating the shown values too fast would make it hard to read. Serial port instances perform 32 consecutive samples moving averaging of real-time measurement data shown to user on Swing GUI to further improve human readability. Main class instantiates the GUI and serial port classes, controls GUI updating and has functions for automating serial port writing to support production automation tasks. Swing GUI also handles user interaction by detecting mouse presses of buttons.

6.1 Features

Priority in development of the PC application was first to enable critical debugging functionality, and later to add support for enough features to perform simple monitoring, control and measurement tasks with a clean looking graphical user interface. The PC application also proved vital for unit testing and calibration, not just debugging work during firmware development. Secondary benefit of the PC application was ability to show off more interesting live demos to promote the new DAQ system for interested parties. They would not have to make their own user interface, if they chose to start using the DAQ cards. Main features of the first PC software release are listed in Table 5.

Table 5. *Features of the developed PC measurement software*

# Feature	Description
1.	Graphical user interface
2.	Data logging to CSV files with µV/µA resolution
3.	User configurable power rail names, shunt values and serial ports with a text file
4.	Built-in serial terminal and command shortcuts
5.	Production scripts for writing EEPROM registers
6.	Support of applying DAQ card calibration coefficients
7.	Real-time monitoring of voltage, current and power measurements
8.	Windows and Linux operating systems support

Power rail names and shunt resistor values are configurable by user with a simple text file. Serial ports are also configured with a text file of their own. Configuring the serial port order had to be done manually by checking DAQ card serial numbers printed on PC software's serial terminal. This was the most difficult step in setup process and had to be explained very well in user manual. Automating serial port setup could be a good addition in future software versions.

Application provides raw data logging capabilities to CSV files. Real-time measurement data can be monitored for all available channels. Built-in serial terminal prints out serial messages from four DAQ cards and it is possible to control DAQ cards with UART API commands directly from terminal or with UI button macros.

To speed up production of DAQ cards, support for automated EEPROM register writing was implemented on the PC software. This enabled for example writing calibration coefficients and serial number to EEPROM automatically from a text file. PC software supports reading calibration coefficients from DAQ cards and applying them to data, producing ready calibrated output for users.

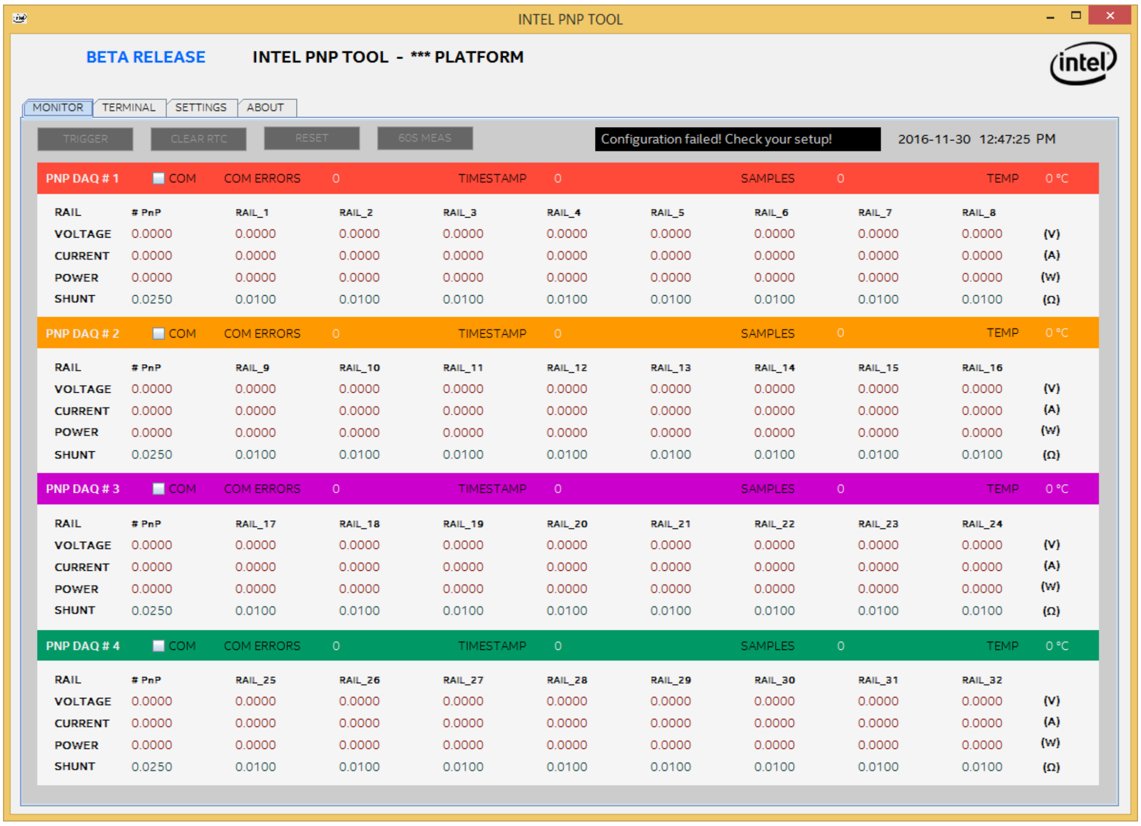
The developed PC application was successfully tested on 64-bit Windows 7, Windows 8.1, Ubuntu 12.04 LTS and Ubuntu 15.04 LTS. Other operating systems would likely work as well if Java JRE7 or newer and serial port adapter drivers are available.

6.2 Graphical user interface

Graphical user interface of the developed PC software was designed to be clean and simple looking supporting only the minimum features required for measurement and debug tasks. To keep the system easy to use, measurement process was made very simple requiring minimal user interaction. To perform measurements, user only needs to start the software, press a button on GUI couple times and close the software. First time installation of software requires installing FTDI drivers [2], Java runtime and copying RXTXcomm library file [5], and setting up virtual serial ports to match the FTDI bridge on DUT hardware.

After starting the software by running a batch file or directly from terminal on Linux, user is greeted with GUI which defaults to monitor tab. On this tab, moving average real-time data can be monitored from all available measurement channels. Update interval of the GUI was set as four times a second and averaging to 32 samples, resulting in moving average of roughly one second at the typical sample rate of the DAQ cards. Faster update rate was tried through iterating various speeds, but more than four times a second made it hard for human to read and provided no benefits as data is anyway stored at full sample rate in the background.

On the monitor tab, user can see measured voltage and current with calculated power value. Each power rail is identified by a name corresponding to hardware schematic net name, defined in a separate configuration text file. Other miscellaneous information such as how many samples are received from each DAQ card, what is the DAQ card temperature and current RTC timestamp value and detected communication error count by PC software are also shown on this tab. Monitor tab has the mostly used command shortcut buttons for triggering measurement, clearing RTC values and resetting DAQ cards from software for user convenience. Based on user feedback, a button for automatic 60 second measurement was also added to this tab. Each of the four DAQ cards are color coded with red, orange, purple and green to match terminal text colors on terminal tab. Monitor tab of the developed PC software graphical user interface is shown in Figure 18.



Configuration failed! Check your setup!

2016-11-30 12:47:25 PM

PNP DAQ # 1 □ COM COM ERRORS 0 TIMESTAMP 0 SAMPLES 0 TEMP 0 °C

RAIL	# PnP	RAIL_1	RAIL_2	RAIL_3	RAIL_4	RAIL_5	RAIL_6	RAIL_7	RAIL_8	
VOLTAGE	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	(V)
CURRENT	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	(A)
POWER	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	(W)
SHUNT	0.0250	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100	(Ω)

PNP DAQ # 2 □ COM COM ERRORS 0 TIMESTAMP 0 SAMPLES 0 TEMP 0 °C

RAIL	# PnP	RAIL_9	RAIL_10	RAIL_11	RAIL_12	RAIL_13	RAIL_14	RAIL_15	RAIL_16	
VOLTAGE	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	(V)
CURRENT	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	(A)
POWER	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	(W)
SHUNT	0.0250	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100	(Ω)

PNP DAQ # 3 □ COM COM ERRORS 0 TIMESTAMP 0 SAMPLES 0 TEMP 0 °C

RAIL	# PnP	RAIL_17	RAIL_18	RAIL_19	RAIL_20	RAIL_21	RAIL_22	RAIL_23	RAIL_24	
VOLTAGE	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	(V)
CURRENT	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	(A)
POWER	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	(W)
SHUNT	0.0250	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100	(Ω)

PNP DAQ # 4 □ COM COM ERRORS 0 TIMESTAMP 0 SAMPLES 0 TEMP 0 °C

RAIL	# PnP	RAIL_25	RAIL_26	RAIL_27	RAIL_28	RAIL_29	RAIL_30	RAIL_31	RAIL_32	
VOLTAGE	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	(V)
CURRENT	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	(A)
POWER	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	(W)
SHUNT	0.0250	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100	0.0100	(Ω)

Figure 18. *Monitor tab of the developed PC application shows user latest averaged measurement results and ability to control measurement hardware using command shortcut buttons.*

Integrating terminal to the PC software was mainly driven by debug use requirements during firmware development. In the end, it proved to be valuable to end users as well, being able to see the UART responses coming from DAQ cards. Button shortcuts for more advanced UART API commands were also placed on this tab for both the developer's and more advanced user's convenience. Terminal tab is shown in Figure 19.

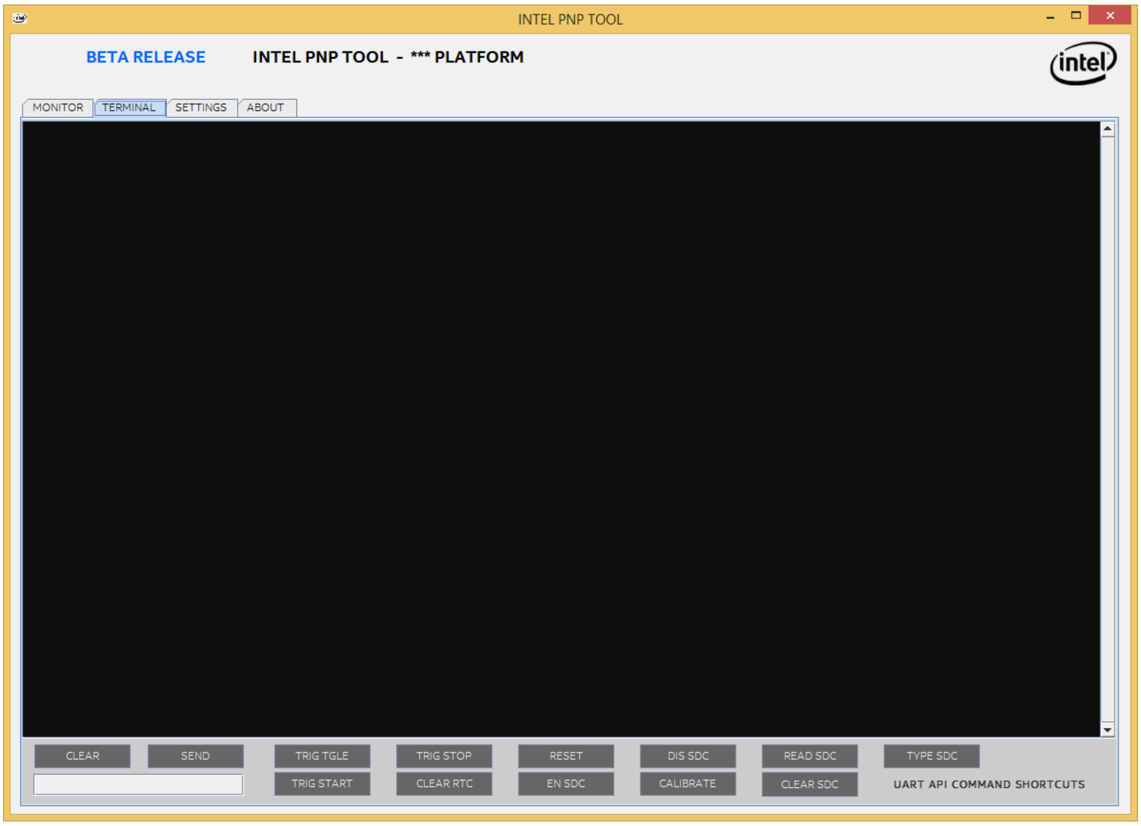


Figure 19. *Terminal tab of the developed PC application shows serial terminal outputs from measurement hardware and allows manual sending of commands mainly for development purposes.*

The settings tab was included mostly for future use, enabling more advanced features and settings configuration by user. Currently it only has controls for enabling and disabling printing of raw data packets to terminal from each DAQ card, which was useful during development. About tab included contact and version information of the PC software. Addition of more tabs to the developed software is very easy, if new features are needed in the future.

7. MEASUREMENT RESULTS

Characterization measurements were done to validate accuracy and operation of complete power measurement card hardware, firmware and software design. Laboratory instruments such as Keysight 34401A 6.5-digit digital multimeter and a Keysight DC power supply were used as reference instruments. A breakout board supplying power and exposing DAQ card analog inputs was used as a validation vehicle, to which external shunt resistor between power supply and load was wired. Key results from characterization measurements are shown in Appendix A power measurement card specifications. Only the most relevant characterization measurements regarding firmware and software design are detailed in this chapter, because focus of this thesis work is not the hardware implementation.

7.1 Voltage measurement accuracy

A Keysight 34401A 6.5-digit digital multimeter was used as a reference instrument. A unit-tested and calibrated DAQ card was installed on a breakout board powered from an USB power bank and connected to a PC with USB for real-time measurement data streaming. A few minutes was waited for the DAQ card to reach typical operating temperature of 40 °C. Test voltage supplied from a DC power supply unit was applied on the DAQ card voltage measurement channel inputs, while measuring it with the Keysight DMM connected in parallel. Test voltage was swept from 0 mV to 4096 mV in increasing steps, and the difference in voltage measured by the DAQ card and the DMM was recorded. Measurement setup used in the DAQ card voltage measurement accuracy characterization is presented in Figure 20.

Figure 21 shows that DAQ card voltage channels measured test voltage with better than ±300 µV accuracy. Other four channels are analog multiplexed to same inputs and have practically same measurement accuracy based on characterization measurements. Multiplexing does not have significant effect on the accuracy when switching timing gives enough time for output voltage to stabilize. All DAQ cards are unit tested to meet ±1 mV accuracy on all channels, so this unit was within specifications with a good margin. This result meets project requirements and is within design expectations.

Additional testing revealed there was no significant drift in measurements when operating temperature was swept from -20 °C to 60 °C in a thermal chamber, validating DAQ card low temperature drift component selection performance. The low thermal drift allows use of embedded DAQ cards even when the whole DUT is placed in a thermal chamber for electrical validation work.

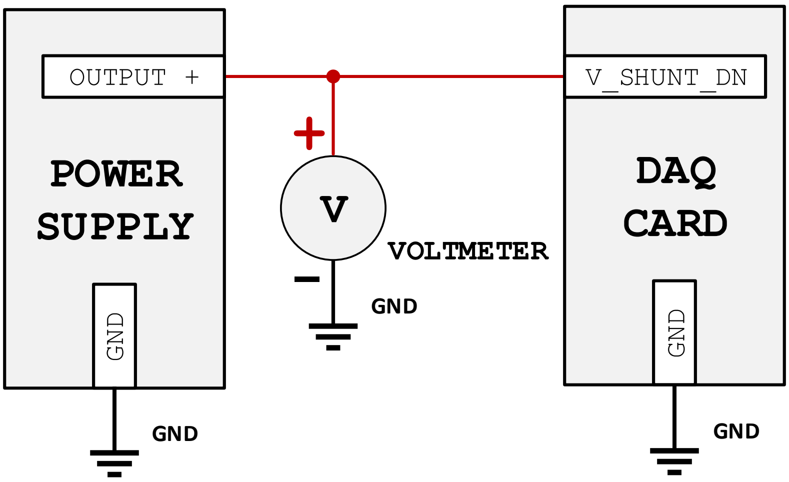


Figure 20. Measurement setup used in the DAQ card voltage measurement accuracy characterization consisted of a power supply unit and a voltmeter.

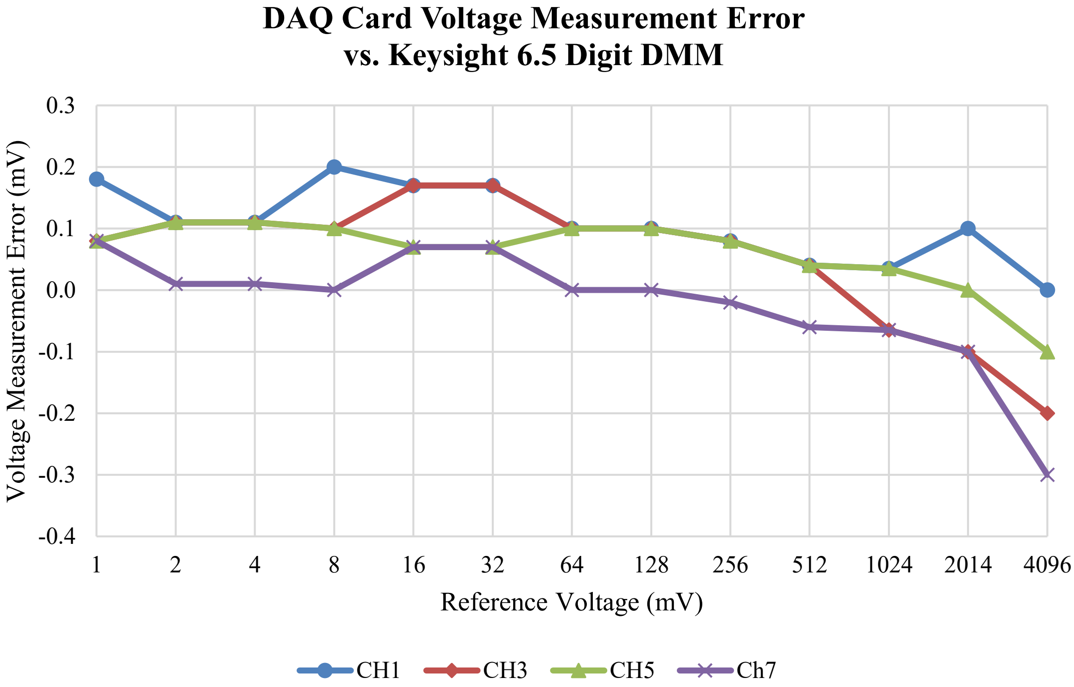


Figure 21. The DAQ card voltage measurement accuracy characterization results from all four of the ADC channels used for voltage measurements.

7.2 Current measurement accuracy

Measurement setup used in the DAQ card current measurement accuracy characterization is presented in Figure 22. A Keysight 34401A 6.5-digit DMM was used as a reference ammeter instrument. A unit-tested and calibrated DAQ card was installed on a breakout board powered from a USB power bank and connected to a PC with USB for real-time measurement data streaming. A few minutes was waited for the DAQ card to reach typical operating temperature of 40 °C. Test current supplied from a DC power supply unit was applied on a test load, which was a programmable current sink. Positive and negative side of an external 20 mΩ 0.1% tolerance precision shunt resistor were connected to the DAQ card measurement inputs. The DC power supply provided adjustable test current flowing through test load and shunt resistor connected in series. The Keysight DMM was connected in series with the load current path and used as a more precise reference instrument. Load current was swept from 0 mA to 1638.4 mA in increasing steps and difference measured by the DAQ card and the DMM was recorded.

Selection of current shunt resistor value has significant effect on minimum accurately measurable current. With a 20 mΩ shunt resistor, the DAQ card typically has better than 1% absolute measurement accuracy after 50 mA of current as is seen in Figure 23, when precision shunt resistor is used. This is within design expectations and could be further improved with better calibration, although this was enough to meet the project requirements in this specific case. All produced DAQ card units were tested to pass the ±3 mA current measurement accuracy specification with a 20 mΩ shunt over the full-scale input range. Common-mode performance was also validated by sweeping input voltage over the whole input range and comparing current measurement to a precision DMM. Results showed similarly good correlation with both instruments over the whole input voltage range.

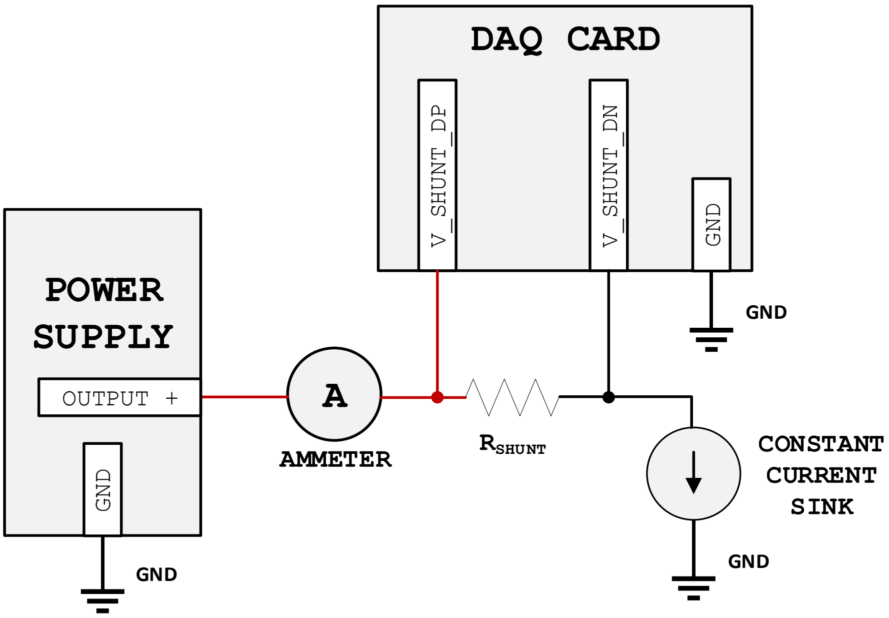


Figure 22. *Measurement setup used in the DAQ card current measurement accuracy characterization consisted of a power supply unit, an ammeter, a precision shunt resistor and an adjustable constant current sink.*

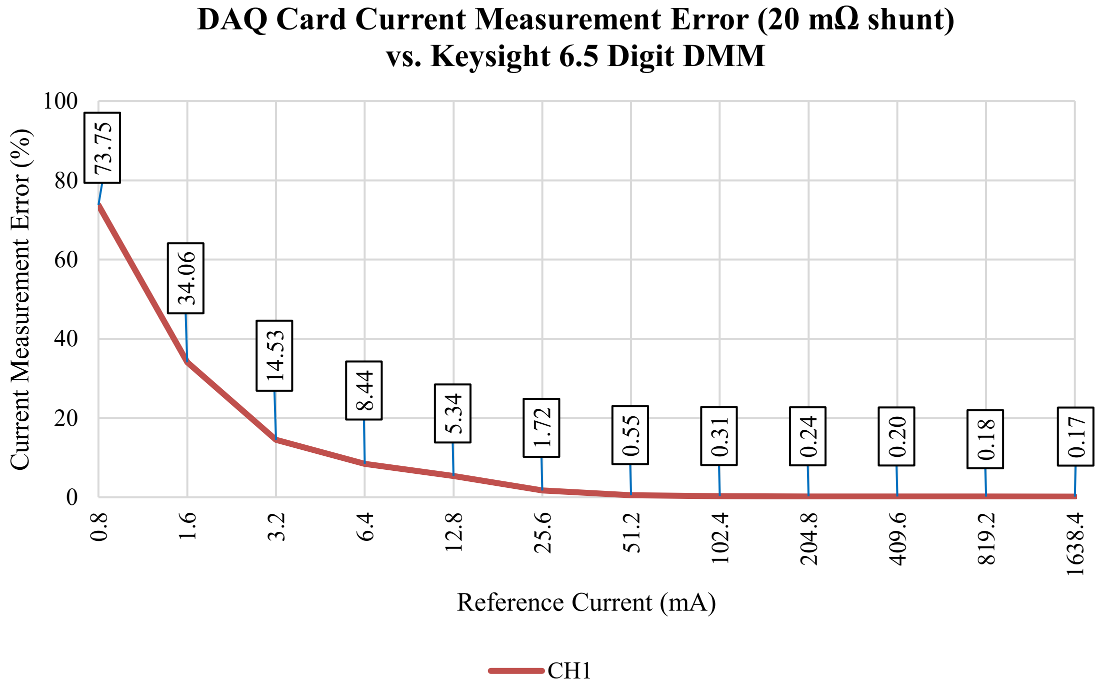


Figure 23. *The DAQ card current measurement accuracy characterization results with a 20 mΩ shunt resistor.*

8. SOFTWARE ISSUES AND SOLUTIONS

Various programming mistakes were found, root causes were identified, and appropriate solutions were implemented during the DAQ card firmware and PC software development and testing. The most critical firmware problem was discovered at a very late stage of doing final unit testing. In the measurement loop, firmware adds up eight consecutive ADC samples to a single 32-bit integer variable. The sum is then divided by eight to perform averaging. Because raw ADC sample data is in two's complement format, and when there was a zero-crossing at middle point of the ADC input range during the eight-sample period, averaging by eight by shifting data right three bits produced unexpected results. In a typical use, this situation is unlikely to show up as averaging period is short, and the zero-crossing needs to happen during it. However, one of the unit testing voltages was selected at this strategic 2.5 V value, which is the reference voltage of ADC and zero-crossing point of two's complement data, and some cards started to show odd results. More debugging revealed this happened only when the ADC input was precisely half of the full-scale input range, so that there is a good change to have a zero-crossing event below and over the middle point. As a quick workaround, firmware performed a zero-crossing detection before doing averaging and took appropriate measures to prevent data corruption. This method proved successful eliminating the problem and was validated by checking raw input and averaged data on zero-crossing events in both directions.

One hard to debug problem occurred when firmware was streaming contents of SD memory card over UART interface. During this operation, microcontroller responded to none of the UART interrupt commands it received. This was a minor issue with impact of causing possible inconvenience to some users in rare situations as operation resumed normally when data streaming from the SD card was over. Therefore, it was decided not to spend time fixing it and instead document the bug clearly in user manual. It is still the only known bug in the shipping firmware version more than half a year later.

Second SD memory card related problem was related to the speed the content was being streamed over UART. Since hardware used no flow control because of limited connector pin count, it was possible to lose big chunks of data due to buffer overflow occurring on the UART bridge chip. Simply slowing down the data streaming by adding more loop delay solved the issue. If a different UART to USB bridge chip with a smaller receive and transmit data buffer is used, the SD card reading might have to be slowed down more. As a backup solution for future, a new UART API command was added to reduce SD card reading speed if problems were to occur with the default speed. The SD memory cards also had unexpectedly wildly varying write latencies, sometimes a write operation taking longer than multiple sample periods. This unpredictable behavior was a significant issue especially with some of the cheaper SD memory card models. By picking the best SD memory card within reasonable cost, with least write latency variation, was good enough for this application but could prove to be architectural pitfall if not prepared for. Other memory storage options with fixed or less varying write latencies should be used if faster and more predictable measurement sample rates are needed, or at least provide a larger external buffer memory allowing larger block writes to the SD memory card mitigating some of the latency variations.

Most of the PC measurement application issues were related to graphical elements acting unexpectedly on certain operating systems. Especially Ubuntu Linux proved problematic by overriding interface text elements with its own larger default font causing visual problems to occur on a software build, which looked as intended on a Windows system. Another issue related to serial data receiving was never fixed, as sometimes a single received data packet would not get passed to the PC measurement software until next one was received. This may be a library issue with the RXTXcomm or odd behavior by the FTDI serial driver. Thus, it may cause minor inconvenience at times when using UART API commands manually but is otherwise insignificant issue as no received data is lost. As a workaround, a flush command could be added to the UART API sending a dummy packet and pushing the last valid data packet through.

9. PROJECT EXECUTION

Preparation for firmware development started before project requirements were fully known and a proof-of-concept target hardware was not yet available. Evolving system requirements were implemented in agile manner, selecting options which were sometimes faster but not necessarily better is terms of system complexity and cost. Project schedule is outlined in Table 6.

Table 6. *Actualized project execution timeline.*

# Month	Project milestone
1.	Firmware development on a development board
2.	Proof-of-concept hardware available
3.	PC measurement software development
4.	Combined HW/FW/SW validation with a new HW revision
5.	First stable FW/SW release
6.	Measurement accuracy characterization completed with final HW revision
7.	Production and delivering measurement systems to customers

First few weeks of firmware development was done on a development kit of previous revision AVR32UC3A3 microcontroller, which was the closest available option and software compatible for the most parts with AVR32UC3A4 device on the target hardware. [7] During this time, all required Atmel Software Framework driver libraries [9] were imported and basic structure of firmware state machine was prepared. Out of the serial interfaces, UART and I2C could be fully enabled in this phase. When target hardware was available, project was converted to the A4 device. The Atmel Xplained evaluation board used in initial development [11] had the MCU in a different package. Because of this, and a custom pinout of the target board, almost all GPIO pin to function mappings had to be redone when moving to target hardware. Thanks to abstracting GPIO pin mapping and code variable names on a header file early on, this was easy task and required no changes on bulk firmware code. In total, there were three hardware revisions starting from early proof-of-concept work, an extensive redesign and final issue solving round. Major changes in firmware code needed to be done for the redesigned board revision.

Debugging target hardware issues same time as enabling more complex firmware functions such as SDIO interface proved challenging. Especially SSC frame-sync interface with ADC was problematic, losing sync after only a few samples, and implementation was changing almost every week trying to find a way to overcome this.

Luckily the less than optimal SSC implementation proved stable and working even in a small-scale volume production. In total, firmware development, testing and issue solving took approximately three months.

Development of PC software started when firmware was close to feature complete stage. If a basic application template would have been ready earlier, development of the firmware would likely have been faster as PC measurement software proved crucial for testing of the whole measurement system. Development of the PC software took approximately two months from start to finish and final touches to it were done as late as production unit testing of the DAQ cards. The PC software continued to improve after shipping of hardware. Firmware development was prioritized first, as doing firmware updates after shipping hardware would not be a trivial task. Development of the Java Swing based PC application proved relatively easy task compared to the DAQ card firmware and there were no significant issues at any phase of the development.

When complete measurement kit with user manual was nearing completion, some kits were given to users with similar technical background as the expected user base but had no previous experience with neither the DUT hardware nor the developed measurement system. This refined documentation further and proved the system met usability requirements, as people succeeded in installing the kit and started first measurements within the first hour. The DAQ card measurement accuracy was verified by other users and it met their expectations, replicating some of the characterization measurement results done during the DAQ card development.

Initially planned project schedule of five months could not be met due to combination of a needed new hardware revision and overly-optimistic initial expectations resulting in a two-month extra delay before whole measurement kit was ready. This delay however significantly improved quality of both the hardware and software deliverables. Good quality deliverables resulted in positive feedback from users and no significant issues were found during the project. Effort of documentation, characterization and production unit testing was also underestimated, and the amount of work surprised as documentation and testing took two months in total for two persons. This was a valuable learning and measures can be taken to prepare better in the future projects by starting documentation in parallel to design work.

Probably the biggest challenge in the whole project was how to make users comfortable with the new measurement instrument and trust its results. Often people would prefer using older, more limited and even worse-performing instruments, because they were used to it and grown to trust it. With time and sharing extensive characterization results, and helping users to replicate them, users started to build trust on the new measurement system. Having more detailed data from platform subsystems than was possible before allowed to identify new problems with DUT’s power management software code and some would not like the new measurement system because the results were not what they wanted to see. Some people would rather question the new measurement system instead of being open-minded to solving the real issues found, but thankfully they were only the loud minority. The benefits of the measurement system were validated when it was used to optimize power consumption of a real product development board, which ultimately met its challenging power consumption targets. The measurement system also received good feedback and few improvement ideas to better accommodate platform software engineer’s needs.

10. CONCLUSIONS

Overall, the project was a challenging experience, but for the most part it was completed without major problems during development. At times, solving a certain issue took a few days, but in the end a working solution was always found, and the project could continue forward towards shipping to users. Some of the more notable problems during the project were debugging target hardware during firmware development, unit testing in small-scale volume production and microcontroller's frame-sync functionality at high speeds required by the specific ADC chip. The amount of needed design collateral documentation for a measurement instrument, which had to be completed before any measurement kits could be delivered to users, was also a surprise and can be better planned for in the future projects. In total, the project took seven months to complete from start to finish. Most of the time was spent in firmware development, but a few months was spent also developing the PC measurement software, characterization testing and documentation to provide customers a ready and validated measurement kit. Table 7 summarizes the project works, development time, challenges and results.

Table 7. *Summary of the project goals, schedule, results and notable challenges.*

Project goals	Create a firmware and a PC measurement software for the PnP DAQ card and meet project requirements regarding functionality, measurement accuracy and usability.
Schedule	Firmware: 3 months PC measurement software: 2 months Testing: 1 month Documentation: 1 month
Results	The PnP DAQ card functionality, measurement accuracy and usability met project requirements, but the project was delayed by two months to accommodate a new hardware revision with design fixes.
Notable challenges	Atmel AVR32's SSC functionality and configuration Voltage regulator power sequencing and inrush current SD memory card read/write latency Firmware timings and how they affect ADC measurement accuracy Firmware development started without target hardware Debugging target hardware during firmware development Initial project schedule was too optimistic Division of work between developing and documenting Unit testing and calibration in production

Project tasks included developing firmware and a PC measurement software for the DAQ card while meeting performance and usability requirements. Detailed characterization of final hardware, firmware and software combination proved that performance of the system met the project requirements. User testing provided much needed feedback about usability of the system, and validated ease of use meeting expectations for a typical technical-minded user base. Based on feedback from users, more features could be added to the PC measurement software to make automated testing easier and further improve usability. Makings software more generic in terms of supported DUT platforms could prove useful, if DAQ cards would be used in multiple projects, reducing amount of software support needed from developer of the power measurement system.

The developed PnP DAQ system was successfully used in a hardware project optimizing a DUT's power consumption to meet its design targets. This proved that the developed one, and other similar DAQ systems, are not just useful tools to have around for debugging purposes but are a valuable addition to any development board's capabilities. The developed power measurement solution brings advanced power measurement capabilities to potentially every software developers' desk giving them a powerful new tool to analyze hardware and software behavior in real-time. To justify additional cost of deploying power measurement solutions to a larger developer base, already low total cost of ownership of the new power measurement solution is further reduced by reusability in multiple projects.

A key learning from the project would be the importance of design for testability and design for manufacturability. If not built to the design from the start, it is very difficult to ramp up production volumes as process is quickly slowed down by unit testing and electrical validation. The simple things like placement of test points underside of a PCB where they cannot be probed with an oscilloscope without soldering wires or lack of a breakout board to enable easier access to signals during test and development can add up and cause significant delays in the long run. In addition, one learned the hard way to read component manufacturer datasheets with a healthy skepticism as some of the fine print or even completely omitted details can reveal unwanted surprises with far-reaching consequences later in the project.

The plans for future include development of a new proof-of-concept system which would add more features at lower total system cost while using a different microcontroller series better suited for future needs and design reuse without the need for advanced PCB manufacturing solutions. C++/Qt based approach to developing the PC software is also planned with greatly improved GUI and USB virtual serial port API without the need for additional UART-to-USB adapters.

REFERENCES

[1] ADA4528-2 Precision, Ultralow Noise, RRIO, Zero-Drift Op Amp Datasheet, Analog Devices, Inc., Rev. F, 2017, p. 10. Available (accessed 01.09.2018): http://www.analog.com/media/en/technical-documentation/data-sheets/ADA4528-1_4528-2.pdf

[2] D2XX Drivers, Future Technology Devices International, Ltd., 2018, web page. Available (accessed 15.07.2018): http://www.ftdichip.com/Drivers/D2XX.htm

[3] Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016, Gartner, Inc., Feb 2017, web page. Available (accessed 14.07.2018): http://www.gartner.com/newsroom/id/3598917

[4] P. Horowitz, W. Hill, The Art of Electronics, Cambridge University Press, 3rd ed. 2015, pp. 4-6.

[5] T. Jarvi and contributors, RXTX project wiki, qbang.org, web page. Available (accessed 15.07.2018): http://rxtx.qbang.org/wiki/index.php/Main_Page

[6] Tutorial 748 The ABCs of ADCs: Understanding How ADC Errors Affect System Performance, Maxim Integrated Products, Inc., Jul 2002, web page. Available (accessed 25.08.2018): https://www.maximintegrated.com/en/app-notes/index.mvp/id/748

[7] Atmel AT32UC3A3/A4 Series Complete Datasheet, Microchip Technology, Inc., Oct 2012, pp. 1-19, 508-545. Available (accessed 14.07.2018): http://ww1.microchip.com/downloads/en/DeviceDoc/doc32072.pdf

[8] Atmel-ICE Debugger User Guide, Microchip Technology, Inc., Oct 2016, p. 8. Available (accessed 15.07.2018): http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42330-Atmel-ICE_UserGuide.pdf

[9] Atmel Software Framework (ASF), Microchip Technology, Inc., 2018, web page. Available (accessed 01.09.2018): https://www.microchip.com/mplab/avr-support/advanced-software-framework

[10] Atmel Studio 7, Microchip Technology, Inc., 2018, web page. Available (accessed 15.07.2018): http://www.microchip.com/mplab/avr-support/atmel-studio-7

[11] Atmel AVR32918: UC3-A3 Xplained Hardware User's Guide, Microchip Technology, Inc., 2018, p. 1. Available (accessed 02.09.2018): http://ww1.microchip.com/downloads/en/AppNotes/doc32159.pdf

[12] NetBeans IDE, Oracle Corporation, 2018, web page. Available (accessed 15.07.2018): https://netbeans.org/

[13] K. Ruoko, Power consumption measurement hardware for portable platforms, Tampere University of Technology, M.Sc. Thesis, 2016.

[14] ADS1278 Octal, Simultaneous Sampling, 24-Bit Analog-to-Digital Converter Datasheet, Texas Instruments, Inc., Feb 2011, pp. 1-33. Available (accessed 02.09.2018): http://www.ti.com/lit/ds/sbas367f/sbas367f.pdf

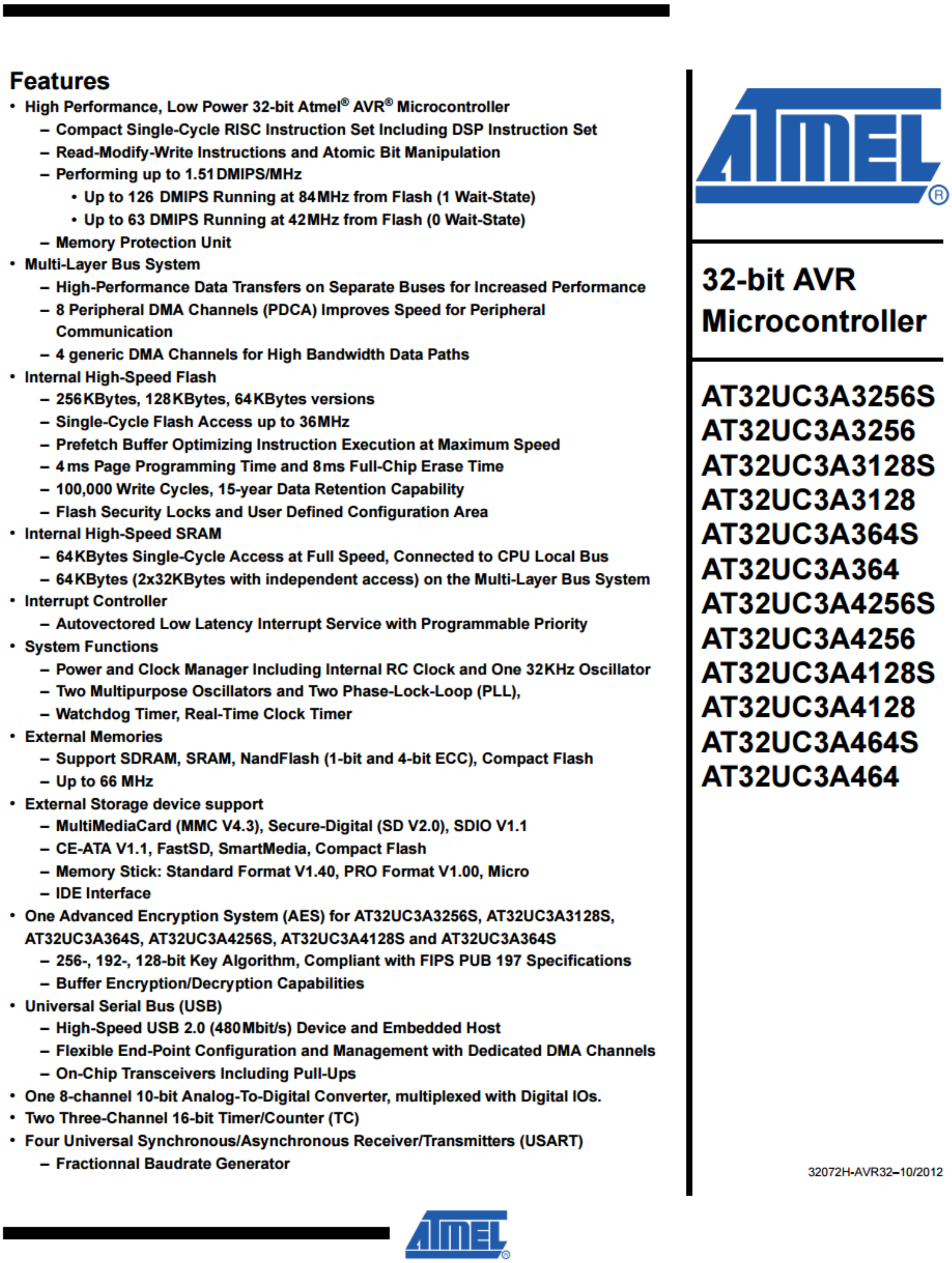
[15] Current Sense Amplifiers Guide, Texas Instruments, Inc., 2018, pp. 2-3. Available (accessed 01.09.2018): http://www.ti.com/lit/sg/slyb194d/slyb194d.pdf

[16] USB 2.0 Specification: USB 2.0 ECN VBUS Max Limit, USB Implementers Forum, Inc., Aug 2014. Available (accessed 01.09.2018): http://www.usb.org/developers/docs/usb20_docs/

APPENDIX A: PNP DAQ CARD SPECIFICATIONS

Parameter	Test Conditions	Min	Typical	Max	Unit
Operating supply voltage		2.7	4	4.5	V
Operating supply current	Supply @ 4.0V, active state		100		mA
Operating supply power	Supply @ 4.0V, active state		400		mW
Measurable rails / PnP card			8	8	
Measurable rails / DUT	4 x PnP DAQ cards		32	32	
Data rate	µSD card enabled		33		sps
Sample rate			8 x Data rate		sps
Oversampling ratio			8		
ADC resolution			24		bits
Rail current shunt voltage (differential)		0		50	mV
Rail voltage		0		5	V
Shunt voltage gain		100	100	100	
Temperature sensor resolution			0.125		°C
Temperature sensor accuracy			± 2		°C
Rail DC voltage accuracy			± 1		mV
Rail DC current accuracy	20mΩ shunt		± 3		mA
UART baud rate			1152000		bits/s
µSD memory card	SDHC, Class 4		16		GB

APPENDIX B: EXTRACT FROM ATMEL AT32UC3A DATASHEET



Features

- High Performance, Low Power 32-bit Atmel® AVR® Microcontroller
  - Compact Single-Cycle RISC Instruction Set Including DSP Instruction Set
  - Read-Modify-Write Instructions and Atomic Bit Manipulation
  - Performing up to 1.51 DMIPS/MHz
    - Up to 126 DMIPS Running at 84 MHz from Flash (1 Wait-State)
    - Up to 63 DMIPS Running at 42 MHz from Flash (0 Wait-State)
  - Memory Protection Unit
- Multi-Layer Bus System
  - High-Performance Data Transfers on Separate Buses for Increased Performance
  - 8 Peripheral DMA Channels (PDCA) Improves Speed for Peripheral Communication
  - 4 generic DMA Channels for High Bandwidth Data Paths
- Internal High-Speed Flash
  - 256 KBytes, 128 KBytes, 64 KBytes versions
  - Single-Cycle Flash Access up to 36 MHz
  - Prefetch Buffer Optimizing Instruction Execution at Maximum Speed
  - 4 ms Page Programming Time and 8 ms Full-Chip Erase Time
  - 100,000 Write Cycles, 15-year Data Retention Capability
  - Flash Security Locks and User Defined Configuration Area
- Internal High-Speed SRAM
  - 64 KBytes Single-Cycle Access at Full Speed, Connected to CPU Local Bus
  - 64 KBytes (2x32KBytes with independent access) on the Multi-Layer Bus System
- Interrupt Controller
  - Autovectored Low Latency Interrupt Service with Programmable Priority
- System Functions
  - Power and Clock Manager Including Internal RC Clock and One 32KHz Oscillator
  - Two Multipurpose Oscillators and Two Phase-Lock-Loop (PLL),
  - Watchdog Timer, Real-Time Clock Timer
- External Memories
  - Support SDRAM, SRAM, NandFlash (1-bit and 4-bit ECC), Compact Flash
  - Up to 66 MHz
- External Storage device support
  - MultiMediaCard (MMC V4.3), Secure-Digital (SD V2.0), SDIO V1.1
  - CE-ATA V1.1, FastSD, SmartMedia, Compact Flash
  - Memory Stick: Standard Format V1.40, PRO Format V1.00, Micro
  - IDE Interface
- One Advanced Encryption System (AES) for AT32UC3A3256S, AT32UC3A3128S, AT32UC3A364S, AT32UC3A4256S, AT32UC3A4128S and AT32UC3A364S
  - 256-, 192-, 128-bit Key Algorithm, Compliant with FIPS PUB 197 Specifications
  - Buffer Encryption/Decryption Capabilities
- Universal Serial Bus (USB)
  - High-Speed USB 2.0 (480 Mbit/s) Device and Embedded Host
  - Flexible End-Point Configuration and Management with Dedicated DMA Channels
  - On-Chip Transceivers Including Pull-Ups
- One 8-channel 10-bit Analog-To-Digital Converter, multiplexed with Digital IOs.
- Two Three-Channel 16-bit Timer/Counter (TC)
- Four Universal Synchronous/Asynchronous Receiver/Transmitters (USART)
  - Fractionnal Baudrate Generator

32-bit AVR Microcontroller

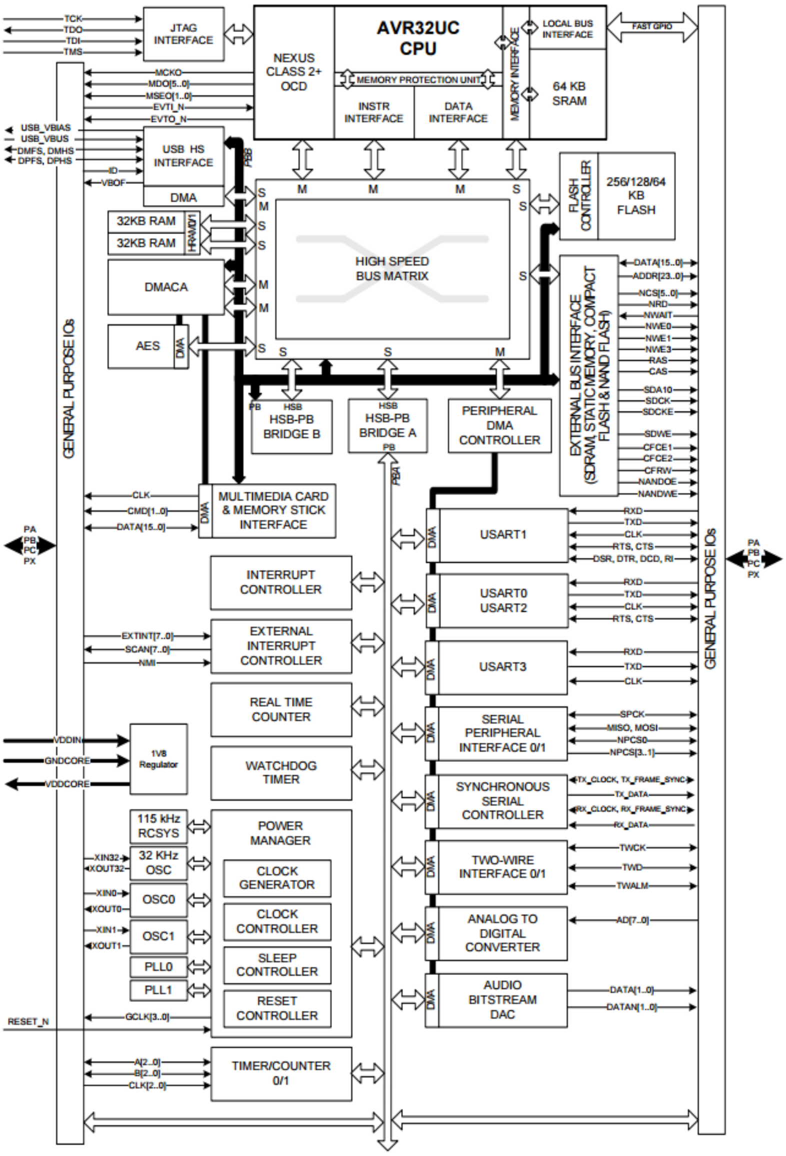
AT32UC3A3256S
AT32UC3A3256
AT32UC3A3128S
AT32UC3A3128
AT32UC3A364S
AT32UC3A364
AT32UC3A4256S
AT32UC3A4256
AT32UC3A4128S
AT32UC3A4128
AT32UC3A464S
AT32UC3A464

32072H-AVR32–10/2012

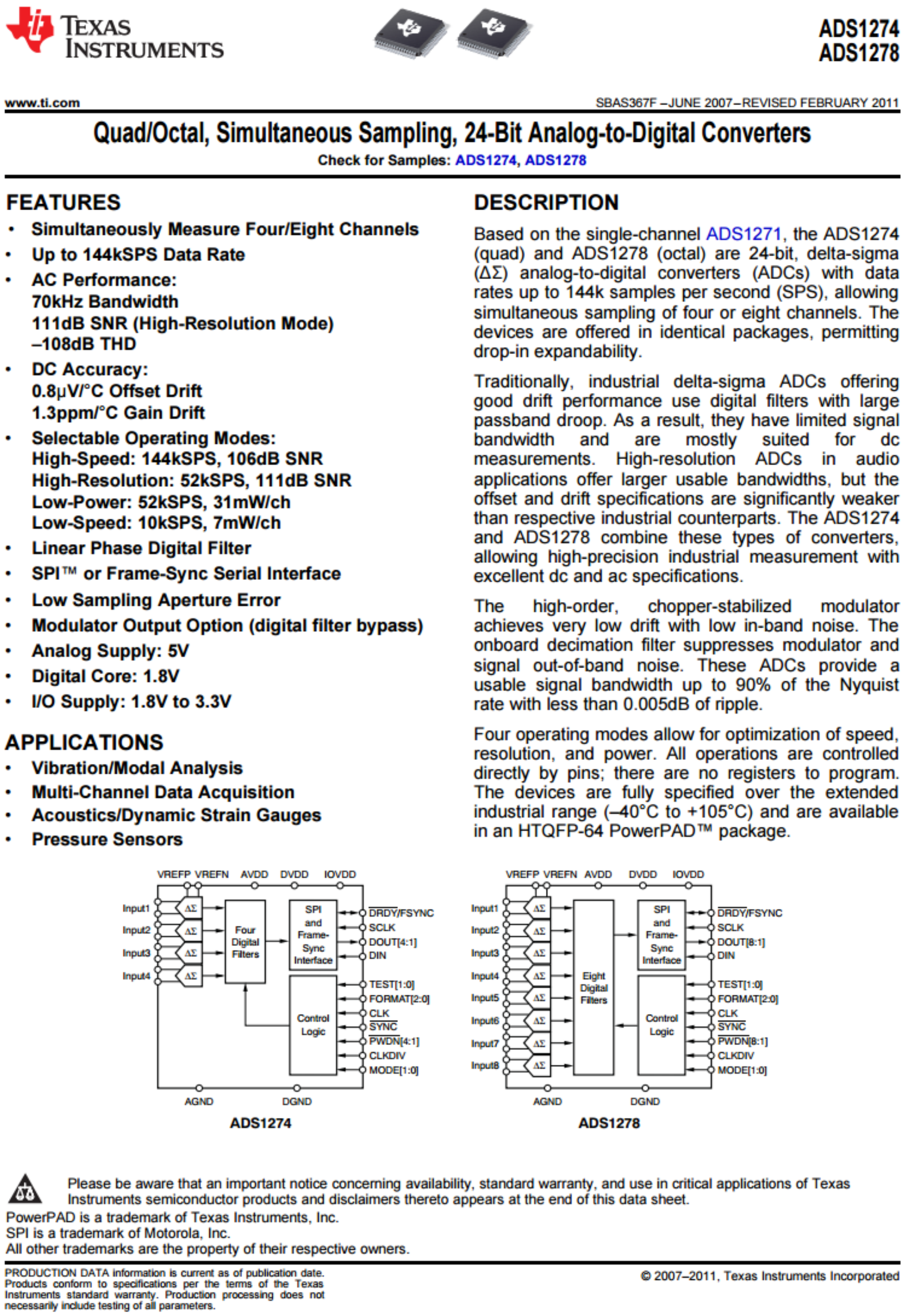
2. Overview

2.1 Block Diagram

Figure 2-1. Block Diagram



APPENDIX C: EXTRACT FROM TEXAS INSTRUMENTS ADS1278 DATASHEET



TEXAS INSTRUMENTS

ADS1274
ADS1278

www.ti.com

SBAS367F – JUNE 2007 – REVISED FEBRUARY 2011

Quad/Octal, Simultaneous Sampling, 24-Bit Analog-to-Digital Converters

Check for Samples: ADS1274, ADS1278

FEATURES

- **Simultaneously Measure Four/Eight Channels**
- **Up to 144kSPS Data Rate**
- **AC Performance:**
  70kHz Bandwidth
  111dB SNR (High-Resolution Mode)
  –108dB THD
- **DC Accuracy:**
  0.8μV/°C Offset Drift
  1.3ppm/°C Gain Drift
- **Selectable Operating Modes:**
  High-Speed: 144kSPS, 106dB SNR
  High-Resolution: 52kSPS, 111dB SNR
  Low-Power: 52kSPS, 31mW/ch
  Low-Speed: 10kSPS, 7mW/ch
- **Linear Phase Digital Filter**
- **SPI™ or Frame-Sync Serial Interface**
- **Low Sampling Aperture Error**
- **Modulator Output Option (digital filter bypass)**
- **Analog Supply: 5V**
- **Digital Core: 1.8V**
- **I/O Supply: 1.8V to 3.3V**

APPLICATIONS

- **Vibration/Modal Analysis**
- **Multi-Channel Data Acquisition**
- **Acoustics/Dynamic Strain Gauges**
- **Pressure Sensors**

DESCRIPTION

Based on the single-channel ADS1271, the ADS1274 (quad) and ADS1278 (octal) are 24-bit, delta-sigma (ΔΣ) analog-to-digital converters (ADCs) with data rates up to 144k samples per second (SPS), allowing simultaneous sampling of four or eight channels. The devices are offered in identical packages, permitting drop-in expandability.

Traditionally, industrial delta-sigma ADCs offering good drift performance use digital filters with large passband droop. As a result, they have limited signal bandwidth and are mostly suited for dc measurements. High-resolution ADCs in audio applications offer larger usable bandwidths, but the offset and drift specifications are significantly weaker than respective industrial counterparts. The ADS1274 and ADS1278 combine these types of converters, allowing high-precision industrial measurement with excellent dc and ac specifications.

The high-order, chopper-stabilized modulator achieves very low drift with low in-band noise. The onboard decimation filter suppresses modulator and signal out-of-band noise. These ADCs provide a usable signal bandwidth up to 90% of the Nyquist rate with less than 0.005dB of ripple.

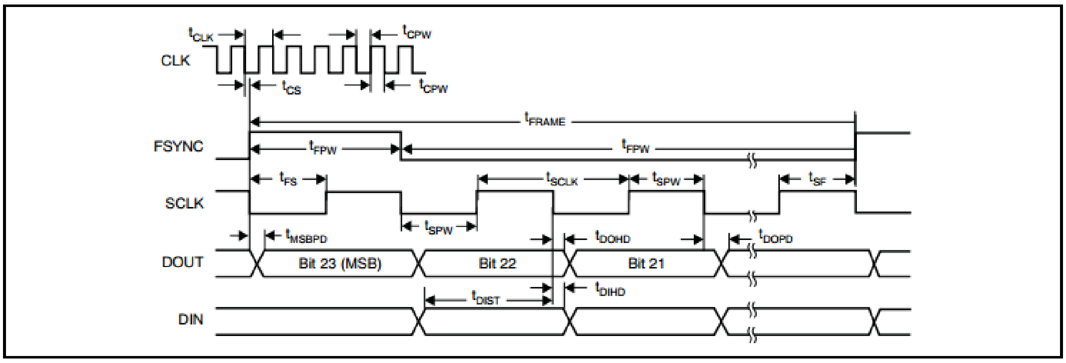
Four operating modes allow for optimization of speed, resolution, and power. All operations are controlled directly by pins; there are no registers to program. The devices are fully specified over the extended industrial range (–40°C to +105°C) and are available in an HTQFP-64 PowerPAD™ package.

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

PowerPAD is a trademark of Texas Instruments, Inc.
SPI is a trademark of Motorola, Inc.
All other trademarks are the property of their respective owners.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of the Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

FRAME-SYNC FORMAT TIMING



FRAME-SYNC FORMAT TIMING SPECIFICATION

For T_A = –40°C to +105°C, IOVDD = 1.65V to 3.6V, and DVDD = 1.65V to 2.2V, unless otherwise noted.

SYMBOL	PARAMETER		MIN	TYP	MAX	UNIT
t_{CLK}	CLK period ($1/f_{CLK}$) (see Table 7)	High-Speed mode	27		10,000	ns
		Other modes	37		10,000	ns
t_{CPW}	CLK positive or negative pulse width		11			ns
t_{CS}	Falling edge of CLK to falling edge of SCLK		–0.25		0.25	t_{CLK}
t_{FRAME}	Frame period ($1/f_{DATA}$)(1)		256		2560	t_{CLK}
t_{FPW}	FSYNC positive or negative pulse width		1			t_{SCLK}
t_{FS}	Rising edge of FSYNC to rising edge of SCLK		5			ns
t_{SF}	Rising edge of SCLK to rising edge of FSYNC		5			ns
t_{SCLK}	SCLK period(2)		1			t_{CLK}
t_{SPW}	SCLK positive or negative pulse width		0.4			t_{CLK}
t_{DOHD}(3)(4)	SCLK falling edge to old DOUT invalid (hold time)		10			ns
t_{DOPD}(4)	SCLK falling edge to new DOUT valid (propagation delay)				31	ns
					21	ns(5)
					25	ns(6)
t_{MSBPD}	FSYNC rising edge to DOUT MSB valid (propagation delay)				31	ns
					21	ns(5)
					25	ns(6)
t_{DIST}	New DIN valid to falling edge of SCLK (setup time)		6			ns
t_{DIHD}(3)	Old DIN valid to falling edge of SCLK (hold time)		6			ns

(1) Depends on MODE[1:0] and CLKDIV selection. See Table 8 (f_{CLK}/f_{DATA}).

(2) SCLK must be continuously running and limited to ratios of 1, 1/2, 1/4, and 1/8 of f_{CLK}.

(3) t_{DOHD} (DOUT hold time) and t_{DIHD} (DIN hold time) are specified under opposite worst-case conditions (digital supply voltage and ambient temperature). Under equal conditions, with DOUT connected directly to DIN, the timing margin is > 4ns.

(4) Load on DOUT = 20pF.

(5) DOUT1, TDM mode, IOVDD = 3.15V to 3.45V, and DVDD = 2V to 2.2V.

(6) DOUT1, TDM mode, IOVDD = 3.15V to 3.45V, and DVDD = 1.7V to 1.9V.