TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND
Elektroenergeetika ja mehhatroonika instituut

PARALLEL MEASUREMENT SYSTEM FOR POWER ELECTRONICS CONVERTERS

JÕUELEKTROONIKA PARALLEELNE MÕÕTESÜSTEEM MUUNDURITELE

BACHELOR'S THESIS

Student: Irina Mironova

Student's code: 157284 AAAB

Supervisor: Oleksandr Matiushkin

Tallinn, 2020

Hereby I declare, that I have written this thesis independently.
No academic degree has been applied for based on this material. All works, major viewpoints and data of the other authors used in this thesis have been referenced.

".....” ..... 202…..

Author: .....
/signature /

Thesis is in accordance with terms and requirements

".....” ..... 202….

Supervisor: ….....
/signature/

Accepted for defence

".....”.....202… .

Chairman of theses defence commission: .....
/name and signature/

LÕPUTÖÖ LÜHIKOKKUVÕTE

Autor: Irina Mironova *Lõputöö liik:* Bakalaureusetöö

Töö pealkiri: Jõuelektroonika paralleelne mõõtesüsteem muunduritele

*Kuupäev:*17.12.2020 53 *lk*

Ülikool: Tallinna Tehnikaülikool

Teaduskond: Inseneriteaduskond

Instituut: Elektroenergeetika ja mehhatroonika instituut

Töö juhendaja(d): Doktorant Oleksandr Matiushkin

Sisu kirjeldus:

Töös kirjeldatakse mõõtesüsteemi arendamise üksikasju. Süsteem juurutati Texas Instruments mikrokontrollerile TMS320F28379D baasil. Transistorimaatriks-vedelkristallkuvar (TFT-LCD) oli vahend lõpptulemuste pealt järelduste tegemiseks. Süsteem suudab mõõta nii alalis- kui vahelduvvoolu omades samal ajal pinge mõõtmiseks sarnaseid funktsioone.

Märksõnad: mikrokontroller, digitaalne liides, kuvar, pinge ja voolu mõõtmine, puuteekraan

ABSTRACT

Author: Irina Mironova *Type of the work:* Bachelor Thesis

Title: Parallel measurement system for power electronics converters.

Date: 17.12.2020 53 *pages*

University: Tallinn University of Technology

School: School of Engineering

Department: Department of Electrical Power Engineering and Mechatronics

Supervisor(s) of the thesis: Doctoral student Oleksandr Matiushkin

Abstract:

The document describes details of the measurement system developing. The system was implemented based on the Texas Instruments microcontroller TMS320F28379D. The Thin-Film-Transistor Liquid-Crystal Display (TFT-LCD) was an instrument to output results. The system can measure as the Direct Current (DC) as the Alternative Current (AC), while it has similar functions for voltage measurement.

Keywords: microcontroller, digital interface, display, voltage and current measurement, touchscreen

THESIS TASK

Thesis title:	**Parallel measurement system for power electronics converters.**
	Jõuelektroonika paralleelne mõõtesüsteem muunduritele
Thesis title in Estonian:	
Student:	**Irina Mironova, 157284 AAAB**
Programme:	**Electrical engineering AAAB02**
Type of the work:	**Bachelor**
Supervisor of the thesis:	**Oleksandr Matiushkin**
Validity period of the thesis task:	**21.12.2020**
Submission deadline of the thesis:	**21.12.2020**

_____ _____ _____

Student (Signature) Supervisor (Signature) Head of programme(signature)

1. Reasons for choosing the topic

Today, microcontrollers are present in almost every part of electronics and a wide selection of communication interfaces provides different ways of connecting and interacting. For optimal and stable system operation, it is important to choose the right microcontroller and communication interface. That is why it is crucial to investigate pros and cons of different microcontrollers and interfaces. The measurement system is the important part of each power electronics converter. Simplified process of data collection, parallel data conversion and its visualization through display could help further work in power electronics

2. Thesis objective

The aim of this thesis is to develop system of data conversion and of output on display. Learn the general functions of the microcontroller TMS320F28379D and the features of the basic display.

3. List of sub-questions:

Overview of the existing interfaces.

Measure voltages and currents with TMS320F28379D.

Implement the data transfer between display and microcontroller in real time.

Compare different interfaces that can be used for data transferring to display.

4. Basic data:

During the research control board with the microcontroller will be used. For the theoretical part mainly, online sources will be used.

5. Research methods

General methods, that will be used in order to fulfill the desired goals are literature analysis, experimental tests and measurements. For practical part, an electronic system will be created, which is using different components such as power supply, control board based on microcontroller, display, measurement instruments, digital analyzer, code composer studio software and personal computer.

6. Graphical material

The work contains figures, tables and important parts of the programming codes. Full codes will be included in the extras. Also, program structure block diagram will be introduced

7. Thesis structure

1. Introduction (Overview the existing interfaces and microcontrollers)

2. Case study system divided in two parts: measurement part and control part

3. Proposed methods to measurement system.

4. Implementation (experimental results).

5. Conclusion

8. References.

The literature used consists mainly of online sources. Sources include specifications and descriptions for electronic components (datasheets) and scientific articles.

9. Thesis consultants

Main supervisor of the thesis is a doctoral student - Oleksandr Matiushkin. Because he is a foreigner thesis will be written in English.

10. Work stages and schedule

September – going through the literature, collecting basic data, writing the theoretical part.

October – work with Arduino board, doing experiments with sensors, connecting them through SPI; completing first and second part of the thesis

November – work with TMS320F28379D microcontroller; completing third and fourth part of the thesis

December – describing and formatting the results of the work process, making a conclusion

Thesis will be completed and submitted before 21.12.2020

Thesis defence will be in January

TABLE OF CONTENTS

LIST OF ABBREVIATIONS AND TERMS

SPI – Serial Peripheral Interface

MCU – Microcontroller Unit

FPGA – Field Programmable Gate Arrays

PWM – Pulse- Width Modulation

CAN – Controller Area Modulation

I2C – Inter-Integrated Circuit

UART – Universal Asynchronous Receiver-Transmitter

DAC – Digital to Analog Converter

ADC – Analog to Digital Convertor

TTL – Transistor-Transistor Logic

JTAG – Joint Test Action Group

SW – Software

HW – Hardware

CPOL – Clock Polarity

CPHA – Clock Phase

IC – Integrated Circuit

PCB – Printed Circuit Board

LCD – Liquid Crystal Display

TFT – Thin Film Transistor

RAM – Random Access Memory

CPU – Central Processing Unit

SOC – Start of Conversion

EOC – End of Conversion

PLL – Phase-Locked Loop

PROM – Programmable Read-Only Memory

EEPROM – Electrically Erasable Programmable Read-Only Memory

INTRODUCTION

Today, microcontrollers are presented almost in each part of electronics. A wide selection of communication interfaces provides different ways of connecting and interacting. It is important to choose the right microcontroller and communication interface for optimal and stable system operation. That is why it is crucial to investigate the pros and cons of different microcontrollers and interfaces. The measurement system is an important part of each power electronics converter. A simplified process of data collection, parallel data conversion and its visualization through display could help further work in power electronics. The main concept of this work is to implement measurement system with a control board based on TMS320F28379D microcontroller. The aim of this thesis is to develop a system of data conversion and of output on the display. General methods, that will be used in order to fulfill the desired goals are literature analysis, experimental tests and code writing for measurement system and display part. For the practical part, an electronic system will be created, which is using different components such as power supply, control board based on microcontroller, display, measurement instruments, digital analyzer, code composer studio software and personal computer.

1. OVERVIEW THE EXISTING INTERFACES AND MICROCONTROLLERS

1.1 Microcontroller unit and field programmable gate array description

Microcontroller Units (MCUs) – microcircuits that is designed to control the electronic devices. A typical MCU combines the functions of a processor and peripheral devices on a single chip and contains different types of memory such us RAM and / or ROM. Basically, it is a single-chip computer capable of performing relatively simple tasks. In our time the widespread use of microcontrollers has led to the development of a variety of microcontrollers, which have different architectures of the processor module, the memory size, a set of peripheral devices, and the type of package. An incomplete list of peripherals that microcontrollers may include [24]:

1) The universal digital ports that can be configured for both directions (input and output);
2) Various serial or parallel interfaces such as UART, I2C, SPI, CAN, USB, Ethernet;
3) Analog-to-digital and digital-to-analog converters (ADC and DACs);
4) Comparators;
5) Pulse width modulators (PWM controller) [27];
6) Timers;
7) Controllers of brushless motors;
8) Display and keypad interfaces;
9) Radio frequency receivers and transmitters;
10) Built-in flash memory arrays;
11) Built-in clock generator and watchdog timer;
12) Hardware trigonometric units;
13) Interrupt modules;
14) Phase-Locked Loop (PLL) and so on.

On the other hand, there are more reliable devices, but it requires more time for implementation. Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from Application Specific Integrated Circuits, which are custom manufactured for specific design tasks. Most FPGAs do not have built-in non-volatile memory, so re-enabling the device requires reloading the FPGA configuration or using an external Flash memory. FPGAs have a

remarkable role in the embedded system development due to capability to start system software development simultaneously with hardware, enable system performance simulations at very early phase of the development, allow various system partitioning (software and hardware) trials and iterations before final freezing of the system architecture. FPGAs provide benefits to designers of many types of electronic equipment, ranging from smart energy grids, aircraft navigation, automotive driver's assistance, medical ultrasounds, data centre search engines and others [26].

1.2 Programming FPGA and Microcontroller

FPGA programming is comparatively complex than that of a microcontroller. FPGA programming requires specialized software such a Xilinx, Intel Quartus. The language of FPGA programming is Verilog or VHDL. FPGA programming follows hardware design flow. FPGA programming has the following steps:

1) A creating a Verilog code or VHDL code;
2) Create a module in the software;
3) Complete pin assignments;
4) Create an SDC file. This file contains timing and design constraints;
5) Convert netlist into Binary Format;
6) Place and Route;
7) Compile the code – Generate a bit file;
8) Program the FPGA;
9) Analyze the report and reprogram.

Microcontroller is programmed using assembly language. High-level programming languages are also used such as JavaScript, Python, and C. A program loading to the microcontroller is the next:

1) A writing a program code (for example C);
2) Compile the code with using a compiler;
3) Upload the compiled version of the program to the microcontroller through digital interface.

1.3 Advantages and disadvantages of FPGA

An FPGA is integrated circuit that can be customized for a specific application and can be programmed any time after manufacturing. These features give the following advantages:

1) New hardware or logical functions can be programmed by altering the programmable blocks in the FPGA. This is possible when installing a new FPGA firmware;

2) FPGA does not have a fixed instruction set;
3) FPGA gives potential of faster and parallel processing of signals;
4) FPGAs have a possibility to use a trial-and-error method. This advantage allows a steep learning curve.

FPGA is a complicated device from this follow its disadvantages such as:

1) It is complex to configure an FPGA, because the designer needs to compile all the codes from scratch and then convert them into machine language;
2) FPGA has a high-power consumption. This could be crucial in projects where power consumption is limited;
3) FPGA can be bulky and costly for a simple application;
4) No internal oscillator: Clock for FPGA must be provided from an external source.

1.4 Advantages and disadvantages of microcontroller

Microcontrollers are widely spread in different types of electronic projects because of its benefits:

1) Microcontrollers are simple to program;
2) Microcontrollers are the better device for a simple and hardware specific application;
3) Microcontrollers are cost effective than FPGA.

But as every circuit, microcontroller have its own drawbacks:

1) Microcontrollers can perform limited tasks. Because they have a limited instruction set. The firmware that is loaded into the controller can perform only with the pre-loaded instructions;
2) Microcontroller relies on a sequential processing. That is one instruction at a time. Hence programming using interrupts becomes complex and hence microcontroller consumes considerable time in executing instructions;
3) Hardware is limited. Designer can utilize only the hardware available on the board.

Summarized information about FPGA and MCU is introduced in table 1.1. This is visual comparison about these two microcircuits.

Table 1.1 Comparison between MCU and FPGA

Criteria	FPGA	Microcontroller
Flexibility	Hardware and firmware reprogrammable. Superior customization.	Reprogramming is possible in firmware only
Programming Firmware	Comparatively complex	Simple programming

Tools	No portability across tools.	Open-source tools available
Cost	Expensive	Cheap
Processing power/speed	Higher	Lower
Flash Memory	External	Internal

1.5 Company overview

The market of the MCUs and FPGAs is quite big and constantly evolving, because there is big competition between manufacturers.

Altera Corporation was a manufacturer of programmable logic devices (PLDs) headquartered in San Jose, California. On December 28, 2015, the company was acquired by Intel. The main product lines from Altera were the Stratix, Arria FPGAs and SoCs, lower-cost Cyclone series system on a chip FPGAs and others [23].

Xilinx is an American technology company that develops highly flexible and adaptive processing platforms. The company invented the field-programmable gate array (FPGA), programmable system-on-chips (SoCs), and the adaptive compute acceleration platform (ACAP) [3].

Atmel Corporation was a designer and manufacturer of semiconductors before being acquired by Microchip Technology in 2016 [11]. It was founded in 1984. The company focuses on embedded systems built around microcontrollers. Its products include microcontrollers (8-bit AVR, 32-bit AVR, 32-bit ARM-based, automotive grade, and 8-bit Intel 8051 derivatives), as it is described in [22].

Texas Instruments Incorporated is an American technology company headquartered in Dallas, Texas, that designs and manufactures semiconductors and various integrated circuits, which it sells to electronics designers and manufacturers globally, as it is demonstrated in [5].

STMicroelectronics is a Swiss-domiciled multinational electronics and semiconductor manufacturer headquartered in Geneva, Switzerland. It is commonly called ST, and it is Europe's largest semiconductor chip maker based on revenue .

Arduino is an open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices [6].

1.6 Microcontrollers overview

Today, we have wide choice of the microcontrollers. It depends on the requirements for it and in what conditions it will work. It could be the price, consumption, performance or other thing.

STM32 is a family of 32-bit microcontroller integrated circuits by STMicroelectronics. The STM32 chips are grouped into related series that are based around the same 32-bit ARM processor core, such as the Cortex-M33F, Cortex-M7F, Cortex-M4F, Cortex-M3, Cortex-M0+, or Cortex-M0. Internally, each microcontroller consists of the processor core, static RAM, flash memory, debugging interface, and various peripherals [10].

The MSP430 is a mixed-signal microcontroller family from Texas Instruments. Built around a 16-bit CPU, the MSP430 is designed for low cost and, specifically, low power consumption embedded applications [5].

AVR is a family of microcontrollers developed since 1996 by Atmel. These are modified Harvard architecture 8-bit RISC single-chip microcontrollers. AVR was one of the first microcontroller families to use on-chip flash memory for program storage, as opposed to one-time programmable ROM, EPROM, or EEPROM used by other microcontrollers at the time. In Table 2 we can see some examples of MCUs in the market [22].

Table 1.2 Comparison of different MCUs

Manufacturer	MCU Type	CPU	Flash, KB	RAM, KB	Interfaces	Voltage, V	Frequency, MHz	Price, $
STMicroelectronics	STM32F205RCT6	32-bit	256	96	USART, SPI, I2C, CAN, USB	1,8 to 3,6	120	20
Texas Instruments	MSP430F2274	16-bit	32	1	UART, SPI, I2C	1,8 to 3,6	16	5
Atmel	ATmega32	8-bit	32	2	UART, SPI, JTAG	4,5 to 5,5	16	2

1.7 Digital interfaces

Digital interfaces are created to give an opportunity to communicate between microcontrollers and different input/output devices. There are two types of digital interface: parallel and serial digital interface. The data is transmitted by several bits

simultaneously using multiply lines in parallel digital interface. The data bits are transmitted one at a time in sequential manner over the data bus or communication channel in serial digital interface. Serial digital interface is more commonly used in modern electronics. There are several different interfaces have been developed to solve the complex problems of balancing circuit design criteria such as features, cost, size, weight, reliability, availability and so on. exist. Below is the short review of the most popular serial digital interfaces.

1.8 Serial Peripheral Interface description

Serial Peripheral Interface (SPI) provides full duplex synchronous communication between a master device and a slave using four data lines [7]. SPI is a synchronous data bus, which means that it uses separate lines for data and a clock that keeps both sides in perfect sync. The clock is an oscillating signal that tells the receiver exactly when to sample the bits on the data line. This could be the rising (low to high) or falling (high to low) edge of the clock signal. When the receiver detects that edge, it will immediately look at the data line to read the next bit. Because the clock is sent along with the data, specifying the speed is not important, although devices will have a top speed at which they can operate. In SPI, only one side generates the clock signal and is called the "master", and the other side is called the "slave". There is always only one master, but there can be multiple slaves. There are two different way of how SPI could be connected, it could be a 3-wire or 4-wire type of connection. Traditional way of how SPI is connected is a 4-wire type of connection (figure 1.1). This connection has four types of signals: Serial Clock (CLK), Slave Select (SSN), Master Out Slave In (MOSI), and Master In Slave Out (MISO).

CLK: Serial Clock. Controlled by the master device. A new data bit is shifted out with each clock cycle.

SSN: Slave Select (the "N" identifies it as an active-low signal). Controlled by the master device. An active slave-select line indicates that the master is sending data to or requesting data from the corresponding slave device.

MOSI: Master Out Slave In. Data leaves the master device and enters the slave device. MOSI lines on chip A are connected to MOSI lines on chip B.

MISO: Master In Slave Out. Data leaves the slave device and enters the master device (or another slave, in a daisy-chain configuration; see the next section). MISO lines on chip A are connected to MISO lines on chip B.

When data is sent from the master to a slave, it is sent on a MOSI line. If the slave needs to send a response back to the master, the master will continue to generate a

prearranged number of clock cycles, and the slave will put the data onto a MISO line. Because the master always generates the clock signal, it must know in advance when a slave needs to return data and how much data will be returned. This is quite different than asynchronous serial, where random amounts of data can be sent in either direction at any time. In practice this is not a problem, as SPI is generally used to talk to sensors that have a specific command structure. For example, if MCU send the command for "read data" to a device, the device will always, for example, two bytes in return. (In cases where device might return a variable amount of data, there is a possibility to return one or two bytes specifying the length of the data and then have the master retrieve the full amount.) As SPI is full duplex, thus, it can transmit and receive data at the same time.

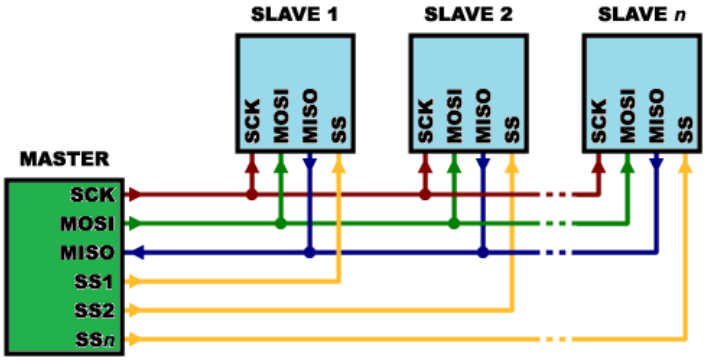

Figure 1.1 Traditional connection of SPI

SPI is widely spread across different projects because of its advantages:

1) It supports multiple slaves, for example one MCU can control several sensors at the same time;
2) It is faster than asynchronous serial;
3) The receive hardware can be a simple shift register.

Because of the specific connection of SPI, this digital interface has few disadvantages:

1) It requires more wires than other communications methods;
2) It usually requires separate SS lines to each slave, which can be problematic if numerous slaves are needed;
3) The communications must be well-defined in advance;
4) The master must control all communications (slaves cannot talk directly to each other).

In SPI, the master can select the Clock Polarity (CPOL) and Clock Phase (CPHA). The CPOL bit sets the polarity of the clock signal during the idle state. The idle state is defined as the period when Chip Select (CS) is high and transitioning to low at the start

of the transmission and when CS is low and transitioning to high at the end of the transmission. The CPHA bit selects the clock phase. Depending on the CPHA bit, the rising or falling clock edge is used to sample and/or shift the data. The master must select the clock polarity and clock phase, as per the requirement of the slave. Depending on the CPOL and CPHA bit selection, four SPI modes are available. Table 3 shows four types of SPI modes [25].

Table 1.3 SPI Modes with CPOL and CPHA

SPI Mode	CPOL	CPHA	Clock Polarity in Idle State	Clock Phase Used to Sample and/or Shift the Data
0	0	0	Logic low	Data sampled on rising edge and shifted out on the falling edge
1	0	1	Logic low	Data sampled on the falling edge and shifted out on the rising edge
2	1	0	Logic high	Data sampled on the rising edge and shifted out on the falling edge
3	1	1	Logic high	Data sampled on the falling edge and shifted out on the rising edge

1.9 Inter-Integrated Circuit

Inter-Integrated Circuit (I2C) is synchronous, 8-bit oriented, multi-master, multi-slave, packet switched, single-ended, serial communication bus. Typically used for communications between individual integrated circuits located on the same printed circuit board. Only two bus lines are required; a Serial Data Line (SDA) and a Serial Clock Line (SCL), regardless of how many devices are on the bus (Figure 1.2), as it is described in [1].

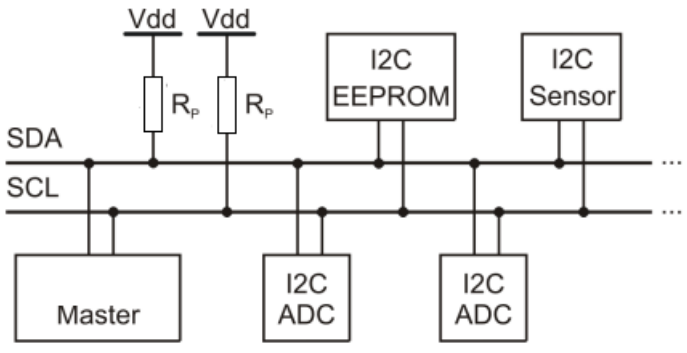

Figure 1.2 I2C connection example

Advantages, when using I2C digital interface are:

1) Simplicity – I2C protocol does not complicate the design. It requires only two bidirectional signal lines to establish communication among multiple devices;
2) The I2C protocol supports multi-master, multi-slave communication;
3) I2C protocol is using chip addressing way. It means that it is possible to add components to the bus without any complexity. It eliminates the necessity of chip select lines;
4) The I2C interface can work well with both slow integrated circuits and fast integrated circuits.

Disadvantages when using I2C digital interface are:

1) Imposes protocol overhead that reduces throughput;
2) requires pull-up resistors, which limit clock speed, consume valuable printed circuit board place and increase power dissipation;
3) Slower speeds bidirectional data transfers can be made at up to 100 kbit/s in the Standard-mode, up to 400 kbit/s in the Fast-mode, up to 1 Mbit/s in Fast-mode Plus, or up to 3.4 Mbit/s in the High-speed mode;
4) line capacity limitation is 400 pF;
5) Increases the complexity of firmware or low-level hardware.

1.10 1-Wire description

1-Wire is a device communications bus system designed by Dallas Semiconductor Corp, as it is shown in [2]. 1-Wire is half-duplex, asynchronous interface that provides low-speed (16.3 kbit/s) data, signaling, and power over a single conductor. 1-Wire is similar in concept to I2C, but with lower data rates and longer range. One distinctive feature of the bus is the possibility of using only two wires — data and ground. To accomplish this, 1-Wire devices include an 800-pF capacitor to store charge and power the device during periods when the data line is active. Popular devices that are using 1-wire technology: iButton, High-Precision 1-Wire Digital Thermometer and Addressable Switch.

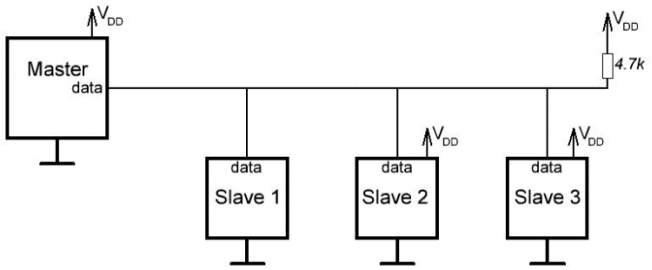

Figure 1.3 1-wire connection example

1.11 Universal Asynchronous Receiver-Transmitter description

Universal Asynchronous Receiver-Transmitter (UART) is an asynchronous serial communication protocol. The universal designation indicates that the data format and transmission speeds are configurable. UART is useful interface which can transmit signals without needing to synchronize with a clock signal. This method of transmission is extremely useful for reducing wires and In/Out pins. It provides a cost effective, simple and reliable communication between one controller to another controller or between a controller and host computer. UART can be thought of as a two-wire communication system (Figure 1.4), where one line is transmitting and the other is receiving. UART can be configured for Full Duplex, Half Duplex, RX only, or TX only version [28].

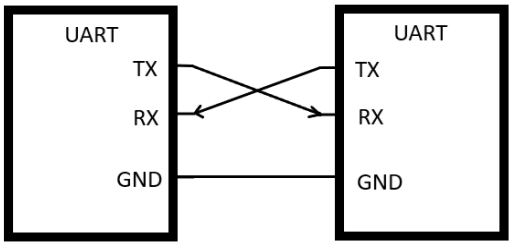

Figure 1.4 UART connection example

Advantages of using UART interface are the following:

1) It is only using two wires;
2) No clock signal is necessary;
3) UART has a parity bit to allow for error checking;
4) The structure of the data packet can be changed as long as both sides are set up for it;
5) Well documented and widely used method.

Disadvantages of using UART interface are:

1) The size of the data frame is limited to a maximum of 9 bits;
2) Does not support multiple slave or multiple master systems;
3) The baud rates of each UART must be within 10% of each other;

There is a method called UART Flow Control for slow and fast devices to communicate with each other over UART without the risk of losing data. Consider two microcontrollers are communicating over UART. One that is receiving data is slower than which are sending data and needs to tell to stop transmitting for a while. Flow control provides

extra signaling to inform the transmitter that it should stop (pause) or start (resume) the transmission. Hardware flow control uses extra wires, where the logic level on these wires define whether the transmitter should keep sending data or stop. With software flow control, special characters are sent over the normal data lines to start or stop the transmission.

1.12 Controller Area Network description

Controller Area Network (CAN) is a two-wire, half duplex, high-speed network system (Figure 1.5), that is far superior to conventional serial technologies such as RS232 in regard to functionality and reliability and yet CAN implementations are more cost effective. In the specific case of the CAN bus controller, it is needed a line driver (transceiver) to convert the controller's transistor–transistor logic signal to the actual CAN level, which is a differential voltage. The use of differential voltage contributes to the vast reliability of CAN. Though maximum number of nodes are not specified for the network. It supports up to 64 nodes due to electrical loading. Maximum length of wires is 40 meters [8].

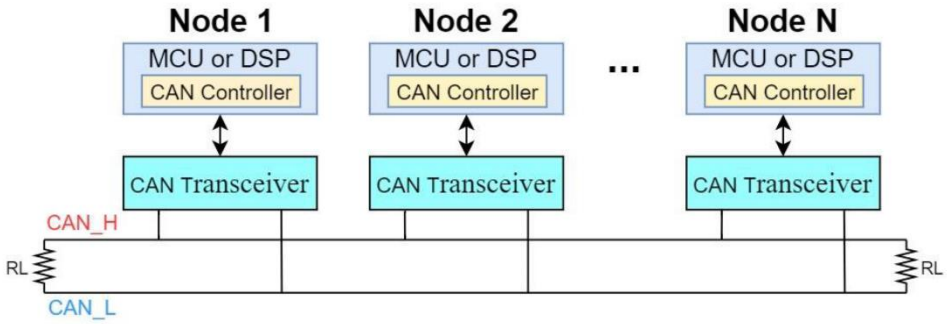

Figure 1.5 CAN simplified block scheme

Advantages of using CAN interface are:

1) Needs only two wires named CAN_H and CAN_L;
2) Operates at data rates of up to 1 Megabit per second;
3) Does not experience message collisions (as they can occur under other serial technologies);
4) Is not demanding in terms of cable requirements. Twisted-pair wiring is sufficient;
5) Does not support node IDs, only message IDs. One application can support multiple message IDs;
6) Supports message priority, i.e., the lower the message ID the higher its priority.

Disadvantages of using CAN interface are:

1) A small amount of data that can be transferred in one package (up to 8 bytes);
2) Large size of service data in the package (compared to useful data);

3) It could have undesirable interactions between nodes;
4) It incurs more expenditure for software development and maintenance.

1.13 Joint Test Action Group description

Joint Test Action Group (JTAG) is an industry standard for verifying designs and testing printed circuit boards after manufacture. Today JTAG is used for debugging, programming and testing on virtually all embedded devices. The JTAG interface gives a way to test the physical connections between pins on a chip. JTAG helps to determine that pin A on chip A is physically connected to pin B on chip B, and that all those pins are functioning correctly. Since JTAG gives direct hardware access to a device, it is also a fantastic tool for security research [4].

The official JTAG standard requires 4 standard pins (or signals) and defines an optional 5th (Figure 1.6). These signals, and the small bit of silicon logic that connects and controls them, are collectively referred to as the Test Access Port.

Test Clock (TCK): The drummer, or metronome that dictates the speed of the TAP controller. Voltage on this pin simply pulses up and down in a rhythmic, steady beat. On every "beat" of the clock, the TAP controller takes a single action. The actual clock speed is not specified in the JTAG standard. The TAP controller accepts its speed from the outside device controlling JTAG.

Test Mode Select (TMS): Voltages on the Mode Select pin control what action JTAG takes.

Test Data-In (TDI): The pin that feeds data into the chip. The JTAG standard does not define protocols for communication over this pin. That is left up to the manufacturer. As far as JTAG is concerned, this pin is simply an ingress method for 1s and 0s to get into the chip. What the chip does with them is irrelevant to JTAG.

Test Data-Out (TDO): The pin for data coming out of the chip. Like the Data-In pin, communication protocols are not defined by JTAG.

Test Reset (TRST): This optional signal is used to reset JTAG to a known good state.

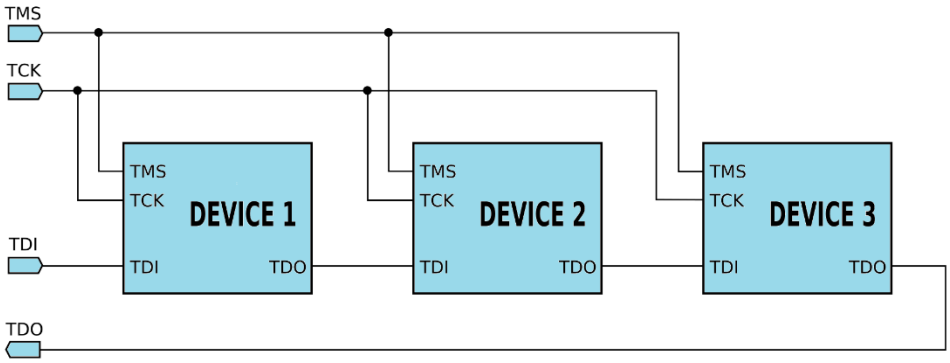

Figure 1.6 JTAG simplified block scheme

1.14 Bluetooth description

Bluetooth is a wireless technology standard used for exchanging data between fixed and mobile devices over short distances using short-wavelength UHF radio waves in the industrial, scientific and medical radio bands, from 2.402 GHz to 2.480 GHz, and building personal area networks (PANs). It was originally conceived as a wireless alternative to RS-232 data cables.

Bluetooth is a standard wire-replacement communications protocol primarily designed for low power consumption, with a short range based on low-cost transceiver microchips in each device, as it is shown in [9]. Because the devices use a radio (broadcast) communications system, they do not have to be in visual line of sight of each other; however, a quasi-optical wireless path must be viable. Range is power-class-dependent, but effective ranges vary in practice.

1.15 Wireless Fidelity description

Wireless Fidelity (Wi-Fi) is a universal wireless networking technology that utilizes radio frequencies to transfer data. Wi-Fi allows high-speed Internet connections without the use of cables. Wi-Fi is Half Duplex. Radio Signals are the keys, which make Wi-Fi networking possible. These radio signals transmitted from Wi-Fi antennas are picked up by Wi-Fi receivers, such as computers and cell phones that are equipped with Wi-Fi cards. Whenever, a computer receives any of the signals within the range of a Wi-Fi network, which is usually 90 — 150 meters for antennas, the Wi-Fi card reads the signals and thus creates an internet connection between the user and the network without the use of a cable.

2. STUDY CASE

2.1 Control board description

Control board based on the good-performance microcontroller TMS320F28379D was selected for project development. The control board was designed and prepared by the power electronics group at Tallinn University of Technology. The aim is to create a programming code for current and voltage measurement and visualize it on the display by using the serial peripheral interface. Control board requires 12 volts of the auxiliary power supply and consumes 0,438 amps of current (around 6W of the power). Board has up to 8 different inputs, where different power supplies could be connected. TFT LCD display with ILI9341 driver shows the results that were measured in real-time (Figure 2.1).

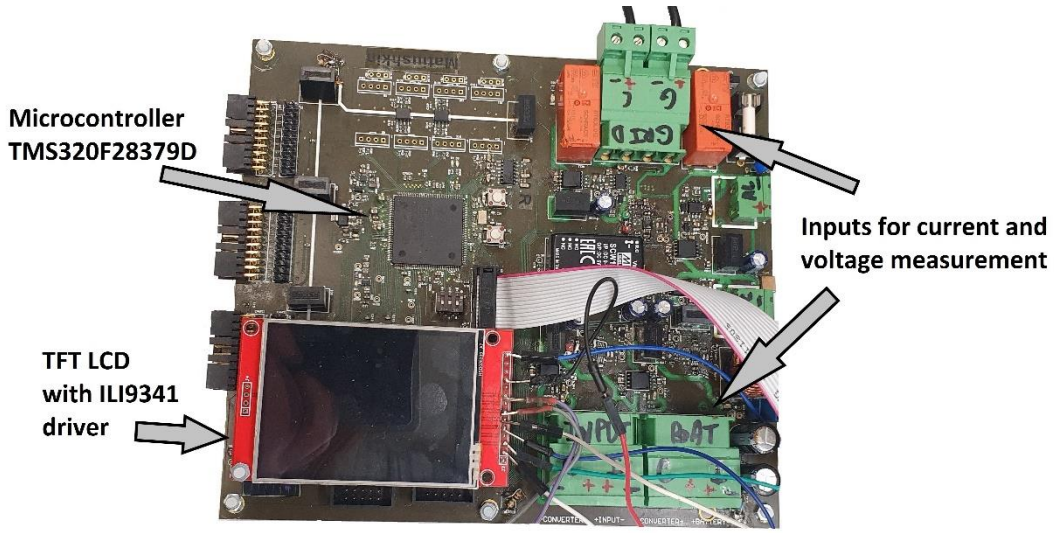

Figure 2.1 Control board overview

To protect the microcontroller from possible damages of high voltage, the control part and power part are isolated. An isolation is carried out using an optocoupler. An optocoupler in this case is integrated inside microcircuits. Alternating voltage, direct voltage and current requires different approach for measure, that is why there are different traces for current, direct voltage and alternating voltage reading on the control board, as it is shown in Figure 2.2. There are five voltage sensors and three current sensors on the control board. They measure voltage or current from five different power supplies simultaneously (but only three current measurement).

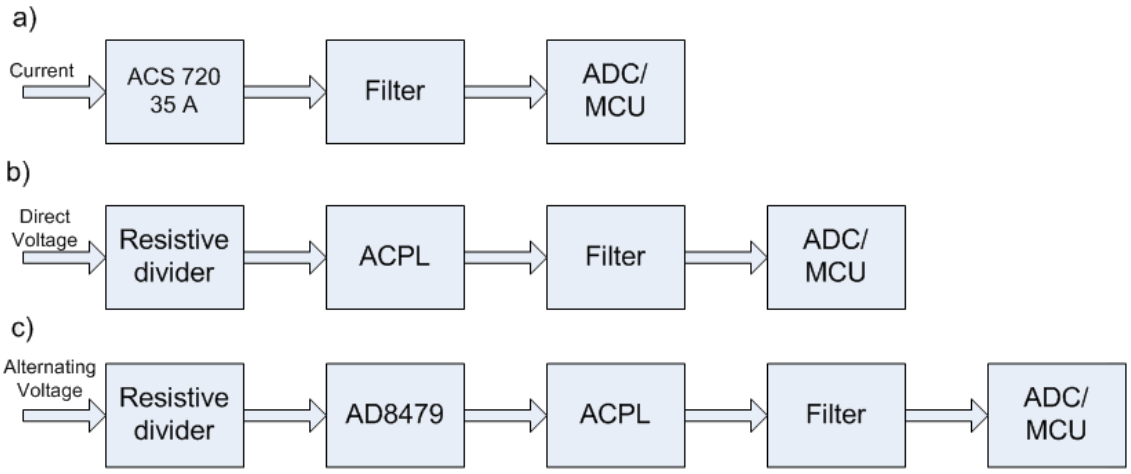

Figure 2.2 Simplified block diagram for a) current reading b) direct voltage reading c) alternating voltage reading

For the current reading simplified diagram is shown in Figure 2.2.a. Current from power supply or from any other source goes to the ACS 720. The ACS 720 is a high accuracy current sensor integrated circuit with the sensing range of 35 amps [19]. One of the key benefits of this circuit is high isolation with the digital output. At zero amps, current sensor gives an output of 1.5 volts. After current sensor, signal goes through the electronic filter and then to the ADC in the MCU.

The DC voltage path is shown in Figure 2.2.b. DC voltage from power supply or from the other source goes first to the resistive. Resistive divider gives a maximum voltage of 0.7 volts, which then goes to the ACPL. The ACPL is automotive high precision DC voltage sensor with isolation [20]. After voltage sensor, signal goes through the electronic filter to the ADC in the MCU.

Alternating voltage measurement requires additional processing. The simplified diagram is shown in Figure 2.2.c. Microcontroller can detect only positive logic from 0 to 3.3 volts, this condition considered when a negative voltage or alternating voltage will be measured. For that reason, after resistive divider that will give voltage range from -0.7 to 0.7 volts depends on the input voltage, the AD8479 amplifier will be used. The AD8479 is a difference amplifier with a very high input common-mode voltage range (up to ±600 V), as it is shown in [21]. This amplifier will enhance the input signal, and in case of -0.7 input voltage, output voltage equals 0.52 volts. After amplifier, the signal goes to ACPL voltage sensor, and through electric filters to the ADC in the MCU.

Code Composer Studio is the software, where the main programming code is created. The Texas Instruments created a set of software to minimize development time. It includes device-specific drivers and libraries, also every function and parameter in these

libraries have a detailed explanation of how they work. These libraries will be used to minimize time while configuring the microcontroller and to concentrate attention to the code itself.

The XDS100V2(Figure 11) is the program debugger which is connected to the computer through isolated USB connection. The communication interface between personal computer and microcontroller is used for safety condition.

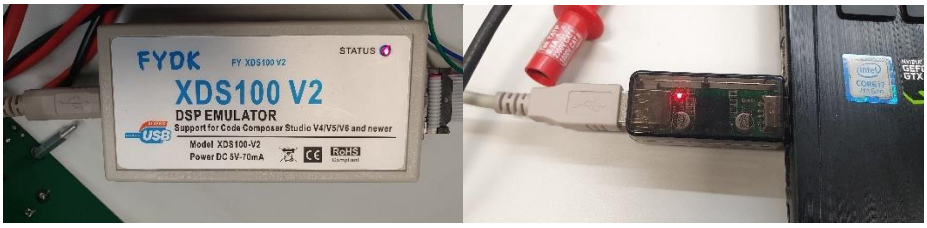

Figure 2.3 Programmer and isolated USB connection

The Programmer XDS100V2 is a piece of the electronic equipment that allows to load the code directly to flash of the microcontroller, but the target devices also include PROM, EPROM, EEPROM, Flash memory, FPGAs and others.

The SPI communication between the microcontroller and the display was tested with the digital analyzer. The SPI has a lot of digital signals that are working simultaneously. To ease signal processing - Saleae Logic Analyzer (Figure 3) will be used. This device can record and display up to 8 digital signals simultaneously. It is not suitable for analog signals but can help with debugging digital protocols such as SPI, I2C, CAN and others.

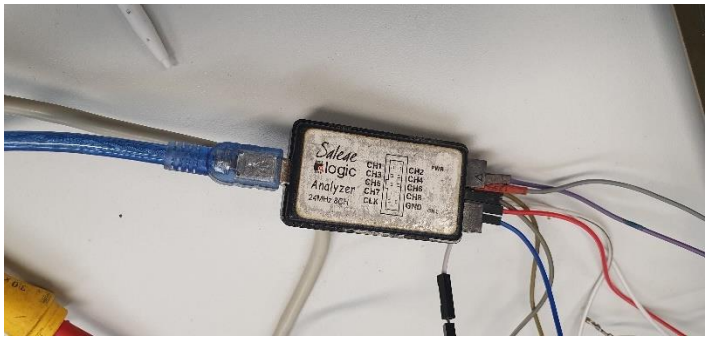

Figure 2.4 Saleae Logic Analyzer

2.2 Microcontroller TMS320F28379D description

The TMS320F28379D is a powerful 32-bit floating-point MCU designed for advanced closed-loop control applications such as industrial motor drives; solar inverters and digital power; electrical vehicles and transportation; sensing and signal processing. The F2837xD series supports a dual-core C28x architecture that significantly boosts system

performance. The integrated analog and control peripherals also allow consolidate control architectures and eliminate multiprocessor use in high-end systems. The dual real-time control subsystems are based on TI's 32-bit C28x floating-point CPUs, which provide 200 MHz of signal processing performance in each core. The C28x CPUs are further boosted by the new TMU (Trigonometric Math Unit) accelerator, which enables fast execution of algorithms with trigonometric operations common in transforms and torque loop calculations; and the VCU (Viterbi/ Complex Math Unit) accelerator, which reduces the time for complex math operations common in encoded applications [12]-[13].

TMS320F28379D specification:

1) Operating Voltage 3.3V;
2) Digital In/Out pins – 97;
3) Flash Memory – 1MB (512 KB per CPU);
4) RAM – 204KB;
5) Clock Speed – 200 MHz;
6) PWM Digital In/Out Pins – 24;
7) Communication Peripherals: 2 – CAN, 2 – I2C, 3 – SPI, 4 – UART, 1 – USB and others;
8) Two TMS320C28x 32-bit CPUs;
9) Price around 10 euro (chip only).

2.3 TFT LCD touch description

2.8-inch thin-film transistor liquid crystal display - TFT LCD (Figure 2.5) with touch ability will be used for measurement visualization. The display is controlled by the ILI9341 driver. It has 240 x 320 pixels of the resolution. The display uses a serial interface, and it is controlled by 5 wires (CS, RS, SCL, SDA, RST). Important to remember that the display SPI module's logic pin can only give a 3.3V of high level, while some microcontrollers' output can have a high level of 5V [14].

Display basic specification:

1) Driver IC: ILI9341;
2) Input voltage: 5V/ 3.3V;
3) Resolution: 240X320;
4) Color: 65K color;
5) Module Interface: 4-wire SPI;
6) Screen size 2.8 inch;
7) Touch function (XPT2046 - a 4-wire resistive touch screen controller);

8) Low cost.

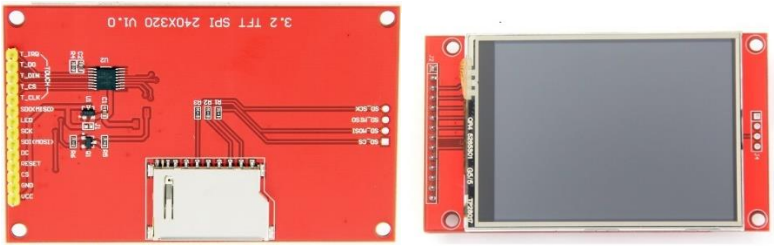

Figure 2.5 SPI TFT Display

2.4 Configuration of SPI connection

Driver ILI9341 is using the SPI interface to communicate with the microcontroller. Logic voltage should not be higher than 3.3V. For proper display functioning, it requires the following connections (figure 2.6): input voltage, ground, chip select for display, chip select for touch screen, data/command pin, touchscreen interrupt pin(T_IRQ), MISO/MOSI/CLCK pins that a shared between display and touch function.

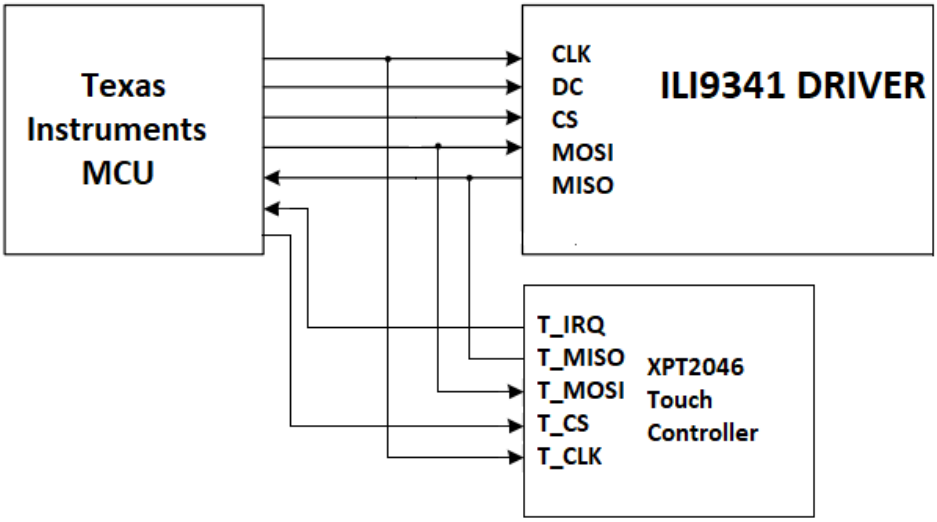

Figure 2.6 SPI connections between microcontroller and display

TMS320F28379D has three SPI ports. SPI configuration must be set in the programming code: which SPI to activate, mark that microcontroller will be master, choose SPI mode depending on the clock polarity and phase, choose SPI bit rate and mark number of bits transferred per frame. As a result, the programming code has the next view:

```
SPI_disableModule(SPIA_BASE);
SPI_setConfig(SPIA_BASE, DEVICE_LSPCLK_FREQ, SPI_PROT_POL0PHA1,
              SPI_MODE_MASTER, 2000000,     8);
SPI_disableFIFO(SPIA_BASE);
SPI_disableLoopback(SPIA_BASE);

SPI_enableInterrupt(SPIA_BASE, SPI_INT_RX_DATA_TX_EMPTY);
SPI_enableModule(SPIA_BASE);
```

Where SPIA_BASE is one of the SPI ports of the microcontroller that will be activated, before making some changes, SPI module should be disabled. After changes are made, it should be enabled again.

2.5 LCD initialization

After SPI configuration the next important step is an LCD initialization. Initialization is the process that activates, prepares display for work and defines parameters. Before initialization, need to split transimittion data or command to LCD. One data/command packet has size of 8 bit. When DATA/COMMAND pin is low, the display interprets the information as a command, when high interprets the information as data. In according to transmit function, it was written the next code

```
void lcd_com(int value)

{

    GPIO_writePin(DC_PIN, 0);
    SPI_writeDataBlockingNonFIFO(SPIA_BASE, (value<< 8));

    while(!(SPI_getInterruptStatus(SPIA_BASE)==SPI_INT_RX_DATA_TX_EMPTY)){}
    SPI_clearInterruptStatus(SPIA_BASE, SPI_INT_RX_DATA_TX_EMPTY);
}

void lcd_data(int value)

{
     GPIO_writePin(DC_PIN, 1);
     SPI_writeDataBlockingNonFIFO(SPIA_BASE, (value<< 8));
     while(!(SPI_getInterruptStatus(SPIA_BASE)==SPI_INT_RX_DATA_TX_EMPTY)){}
     SPI_clearInterruptStatus(SPIA_BASE, SPI_INT_RX_DATA_TX_EMPTY);
}
```

where command GPIO_writePin is changing status of the pin, it is high when value is one and low when value is zero. Small delay is implemented through the SPI interrupt options. It means that the MCU will not send next information until the previous packet is finished. Between information packets must exist small delay, without it display will not understand what information was sent to it and initialization will fail.

Next step is to configure main display parameters such as power, driver timing control, power on sequence control, memory access control, pixel format and others. Hex addresses of commands and parameters are taken from the driver datasheet to properly initialize the LCD display.

MCU is sending information to the display when the chip select pin has low level. On the other hand, communication between MCU and display is disabled, when the chip select pin has high-status. The most significant bit is transmitted first; it is important to specify

this condition in the programming code. In Figure 2.7 is shown an example of digital signals between the MCU and display.

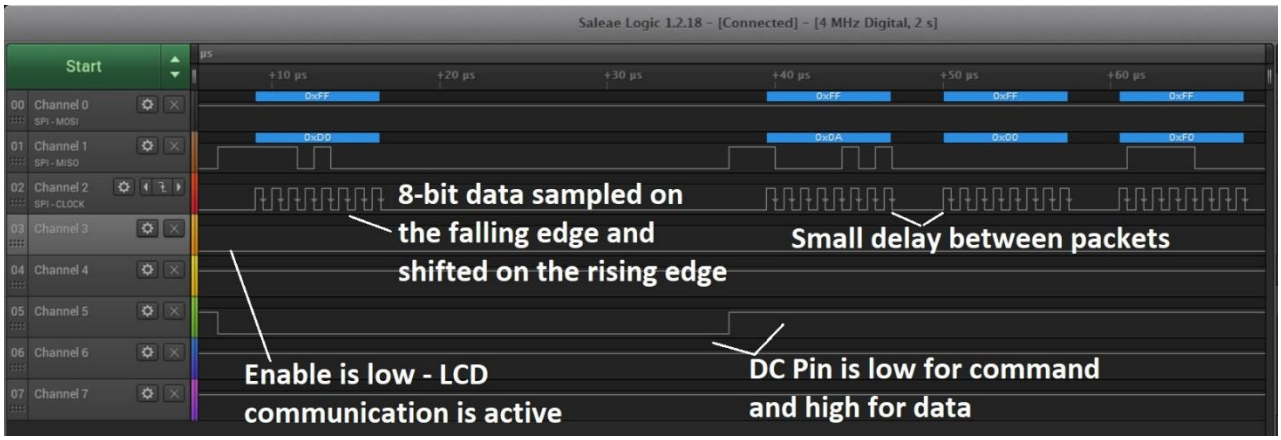

Figure 2.7 SPI signals example between the MCU and display

For convenient use of display, it is good to add a possibility to rotate the screen, for example, the display could have either portrait or landscape orientation. For this reason, X- and Y-axis can have either display width or height values, below is the programming code example of this setting:

```
void ili9341_setRotation(uint8_t m)
{
    GPIO_writePin(CS_PIN, 0);

    uint8_t rotation;

    lcd_com (0x36);
    rotation = m% 4;

    switch (rotation)
    {
        case 0:
            lcd_data (0x40|0x08);
            LCD_W = 240;
            LCD_H = 320;
            break;
        case 1:
            lcd_data (0x20|0x08);
            LCD_W = 320;
            LCD_H = 240;
            break;
        case 2:
            lcd_data (0x80|0x08);
            LCD_W = 240;
            LCD_H = 320;
            break;
        case 3:
            lcd_data (0x40|0x80|0x20|0x08);
            LCD_W = 320;
            LCD_H = 240;
            break;
    }
```

```
    GPIO_writePin(CS_PIN, 1);
}
```

It is good to remember, that this parameter is changing zero point of the display. The image will be rotated, when changing this parameter, but all X- and Y- coordinates will remain the same.

2.6 LCD Graphic Library creation

To start GUI creation, that will visualize result on display, it is important to create a basic graphics library where all functions are collected from a simple draw a pixel function:

```
void ili9341_drawpixel(uint16_t x3,uint16_t y3,uint16_t colour1)
{
    GPIO_writePin(CS_PIN, 0);
    if ((x3 < 0) ||(x3 >= LCD_W) || (y3 < 0) || (y3 >= LCD_H))
    {
        return;
    }

    address_set(x3, y3, x3 + 1, y3 + 1);
    ili9341_pushcolour (colour1);
    GPIO_writePin(CS_PIN, 1);
}
```

The complex function was created that can display float number for the values that were calculated by the ADC module:

```
void ili9341_fdig_string (float value)
{
    int temp = (int)(value*100.0f);
    uint8_t sign_flag = 0;
    uint8_t amount_flag = 0;
    if(temp < 0)
    {
        sign_flag = 1;
        temp = -temp;
    }
    if(temp < 10)
    {
        amount_flag = 1;
    }
    else if(temp < 100)
    {
        amount_flag = 2;
    }
    if(temp == 0)
    {
        ili9341_drawchar(cursor_x , cursor_y, '+', textcolour, textbgcolour,
textsize);
        cursor_x = cursor_x + textsize * 6;
        ili9341_drawchar(cursor_x , cursor_y, '0', textcolour, textbgcolour,
textsize);
        cursor_x = cursor_x + textsize * 6;
```

```
        ili9341_drawchar(cursor_x , cursor_y, '.', textcolour, textbgcolour,
textsize);
        cursor_x = cursor_x + textsize * 6;
        ili9341_drawchar(cursor_x , cursor_y, '0', textcolour, textbgcolour,
textsize);
        cursor_x = cursor_x + textsize * 6;
        ili9341_drawchar(cursor_x , cursor_y, '0', textcolour, textbgcolour,
textsize);
        cursor_x = cursor_x + textsize * 6;
    }
    else
    {
        unsigned char str[10] = "";
        int i = 0;

        while(temp != 0)
        {
            str[i] = temp % 10 + '0';
            temp /= 10;
            i++;
        }

        if(amount_flag == 1)
        {
            str[i] =  '0';
            i++;
            str[i] =  '0';
            i++;
        }

        if(amount_flag == 2)
        {
            str[i] =  '0';
            i++;
        }
        int k = i;
        for(i = k-1; i >= 0; i--)
        {
            if(i == 1)
            {
                ili9341_drawchar(cursor_x , cursor_y, '.', textcolour,
textbgcolour, textsize);
                cursor_x = cursor_x + textsize * 6;
            }
            if(i == k-1 )
            {
                if(sign_flag)
                {
                    ili9341_drawchar(cursor_x , cursor_y, '-', textcolour,
textbgcolour, textsize);
                    cursor_x = cursor_x + textsize * 6;
                }
                else
                {
                    ili9341_drawchar(cursor_x , cursor_y, '+', textcolour,
textbgcolour, textsize);
                    cursor_x = cursor_x + textsize * 6;

                }
```

```
            }
            ili9341_drawchar(cursor_x , cursor_y, str[i], textcolour,
textbgcolour, textsize);
            cursor_x = cursor_x + textsize * 6;
        }
    }
}
```

This "ili9341_fdig_string" function calculates different statements of the result, for example current or voltage measurements could be decimal or integer, can have either a negative or positive sign.

Every time image appeared on the display, it meant that the MCU had sent coordinates (x and y) for each pixel, using the logic that is presented in Figure 2.8.

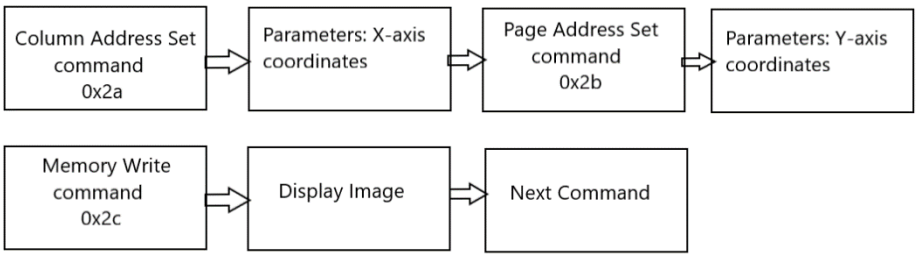

Figure 2.8 Address set command sequence

The display image can be presented with using different colors. All colors on a display are made up by combining the light from three colors (red, blue, and green). The max value of each of the colors is 255. The minimum value is 0. The bit assignment of the 16-bit color values is R5G6B5, which make up one 16-bit word as follows: RRRRRGGGGGGBBBBB (most significant bit is leftmost). The formula to get a 16-bit color value from the red/green/blue parts is:

$$color16bit = (red \ll 11) + (green \ll 5) + blue, \quad (1)$$

Where shifting red left by 11, shifting green left by 5 and adding the shifted red and green value and blue together delivers the 16-bit color value. (Table 2.1)

Table 2.1 Color binary and hex values example

Color	Binary Value	Hex Value
BLACK	0000000000000000	0000
BLUE	0000000000011111	001F
WHITE	1111111111111111	FFFF
RED	1111100000000000	F800

2.7 Touchscreen initialization

The display has resistive touchscreen functionality that is controlled by the XPT2046 touch controller. Resistive touchscreens are characterized using two layers, made of electrically resistive material. These layers are separated by a layer of air or inert gas. When the surface of a resistive touchscreen is tapped, the two layers make contact — this is how resistive touchscreens can identify where, exactly touch happened [15]. During datasheet studying – main commands that should be sent to the touch controller are the next:

1) CMD_RDX – 0xD0 Reads X coordinate;
2) CMD_RDY – 0x90 Reads Y coordinate;
3) TP_PRES_DOWN 0x80 tells that touchscreen is pressed;
4) TP_CATH_PRES 0x40 pressure of the touch.

When touchscreen interrupt pin(T_IRQ) is low, the touch function is activated. Touch controller has a different chip select pin, thus for touchscreen functionality, it is needed to create their own commands for reading the information, and for sending the information. Touchscreen initialization includes setting up display width, height, and rotation. Touch-rotation should match the LCD-rotation. Touch driver is sending 16 bit command, that is why in the programming code information is written in first byte, then is second byte, and the program will return the command to microcontroller as a sum of two bytes (most significant bit first). The result of this command looks like that:

```
uint16_t TP_Read_ADC(int value) // Touch Command
{
    uint16_t num;
    unsigned char byte1, byte2;

    SPI_writeDataBlockingNonFIFO(SPIA_BASE, (value<< 8));

    while(!(SPI_getInterruptStatus(SPIA_BASE)==SPI_INT_RX_DATA_TX_EMPTY)){}
    SPI_clearInterruptStatus(SPIA_BASE, SPI_INT_RX_DATA_TX_EMPTY);

    SPI_writeDataBlockingNonFIFO(SPIA_BASE, 0);
    byte1 = SPI_readDataBlockingNonFIFO(SPIA_BASE);
    DEVICE_DELAY_US(40);

    SPI_writeDataBlockingNonFIFO(SPIA_BASE, 0);
    byte2 = SPI_readDataBlockingNonFIFO(SPIA_BASE);
    DEVICE_DELAY_US(40);

    num = (byte1<<8) | byte2;

    return num;
}
```

For touchscreen data created an additional command, microcontroller can read data from the touchscreen with using this command:

```
void TP_Write_Byte(int value) // Touch Data
{
    GPIO_writePin(CS_PIN, 1);
    GPIO_writePin(DC_PIN, 1);
    SPI_writeDataBlockingNonFIFO(SPIA_BASE, (value<< 8));
    while(!(SPI_getInterruptStatus(SPIA_BASE)==SPI_INT_RX_DATA_TX_EMPTY)){}
    SPI_clearInterruptStatus(SPIA_BASE, SPI_INT_RX_DATA_TX_EMPTY);
}
```

2.8 Touchscreen calibration

For proper touchscreen functionality. The touchscreen has been calibrated. To do that few samples was taken from each corner of the display and from the center of the display. Calibration formulas that were used are:

$$x = \frac{x - OFFSET}{GAIN}, \tag{2}$$

$$y = \frac{y - OFFSET}{GAIN}, \tag{3}$$

where OFFSET is the value taken from the corners of the display, where coordinates for x- or y-axis equal to zero. Gain is obtained from the result where the coordinates have its maximum value. Three samples were taken for each axis. Calculation results are presented in table 2.2 and 2.1.

Table 2.2 Touchscreen „x" coordinates calculation

	Coordinate position (x; y)				
	(0;0)	(0;320)	(240;0)	(240;320)	(120;160)
x1	1600	1760	29023	28976	14116
x2	1640	1550	29075	29692	14022
x3	1520	1739	29380	30182	14590
Average	1586,667	1683	29159	29616	14242
-	-	-	-	OFFSET X	1634,833
-	-	-	-	GAIN X	121,1963

Table 2.3 Touchscreen „y" coordinates calculation

	Coordinate position (x; y)				
	(0;0)	(240;0)	(0;320)	(240;320)	(120;160)
y1	3323	3614	31107	30940	16190
y2	3497	3472	32760	30222	16107

y3	3019	3508	31137	30969	15952
Average	3279,667	3531,333	31668	30710	16083
-	-	-	-	OFFSET Y	3405,5
-	-	-	-	GAIN Y	88,483

2.9 ADC configuration

Analog-to-digital converter (ADC) is a system that converts an analog signal into a digital signal. An ADC may also provide an isolated measurement such as an electronic device that converts an input analog voltage or current to a digital number representing the magnitude of the voltage or current [17]. The TMS320F28379D has 4 ADC built inside. They can have either 12- or 16-bits resolution. As ADC will be used in differential mode, according to the datasheet it should be configured at 16-bit resolution [13]. Differential mode allows ADC to measure not only direct but also alternative currents and voltages. The input voltage of the converter is sampled on a pair of inputs, a positive and a negative. Below is the sample of programming code, that will configure all four ADC modules:

```
ADC_setMode(ADCA_BASE, ADC_RESOLUTION_16BIT, ADC_MODE_DIFFERENTIAL);
ADC_setMode(ADCB_BASE, ADC_RESOLUTION_16BIT, ADC_MODE_DIFFERENTIAL);
ADC_setMode(ADCD_BASE, ADC_RESOLUTION_16BIT, ADC_MODE_DIFFERENTIAL);
ADC_setMode(ADCC_BASE, ADC_RESOLUTION_16BIT, ADC_MODE_DIFFERENTIAL);
```

Which means that all four ADC modules of the microcontroller will be configured with the same principle.

A prescaler is an electronic counting circuit that is used to reduce a high-frequency electrical signal that goes to the ADC module. The prescaler takes the basic timer clock frequency and divides it by some value before feeding to the timer. The configuration is set for each ADC module separately. Programming code looks like that:

```
ADC_setPrescaler(ADCA_BASE, ADC_CLK_DIV_4_0);
ADC_setPrescaler(ADCB_BASE, ADC_CLK_DIV_4_0);
ADC_setPrescaler(ADCC_BASE, ADC_CLK_DIV_4_0);
ADC_setPrescaler(ADCD_BASE, ADC_CLK_DIV_4_0);
```

Where ADC_CLK_DIV_4_0 means that the input clock will be divided by 4. The ADC triggering and conversion sequencing is accomplished through configurable start-of-conversions (SOCs). It should be configured for each measurement that will be performed, in this case for “Ian”, “Van” and “Vinn” value:

```
ADC_setupSOC(ADC_MODULE_IAN, ADC_SOC_IAN, ADC_TRIGGER_EPWM4_SOCA,
ADC_DIFF_CH_IAN, 64);
```

```
ADC_setupSOC(ADC_MODULE_VAN, ADC_SOC_VAN, ADC_TRIGGER_EPWM4_SOCA,
ADC_DIFF_CH_VAN, 64);
ADC_setupSOC(ADC_MODULE_VINN, ADC_SOC_VINN, ADC_TRIGGER_EPWM4_SOCA,
ADC_DIFF_CH_VINN, 64);
```

Where each setting has its explanation:

1) ADC_MODULE_IAN - is the base address of the ADC module;
2) ADC_SOC_IAN – which start-of-conversion will be used;
3) ADC_TRIGGER_EPWM4_SOCA - trigger the source that will cause the SOC;
4) ADC_DIFF_CH_IAN - channel is the number associated with the input signal;
5) 64 - is the acquisition window duration.

Below are the settings that will configure the end-of-conversion (EOC) pulse generated by ADC:

```
ADC_setInterruptPulseMode(ADCA_BASE, ADC_PULSE_END_OF_CONV);
ADC_setInterruptPulseMode(ADCB_BASE, ADC_PULSE_END_OF_CONV);
ADC_setInterruptPulseMode(ADCC_BASE, ADC_PULSE_END_OF_CONV);
ADC_setInterruptPulseMode(ADCD_BASE, ADC_PULSE_END_OF_CONV);
```

The pulse will be generated at the end of the voltage conversion, one cycle prior to the ADC result latching into its result register (Figure 2.10).

Figure 2.10 ADC working sequence

Interrupt will occur and process will start over again after ADC finished. The interrupt function will be performed by Enhance Pulse-Width Modulation (EPWM). PWM - is a method of reducing the average power delivered by an electrical signal, by effectively chopping it up into discrete parts. PWM and microcontroller clocks must be synchronized for accurate values [12]. PWM has a triangle waveform (Figure 2.11). PWM frequency will be set to 50 000 HZ.

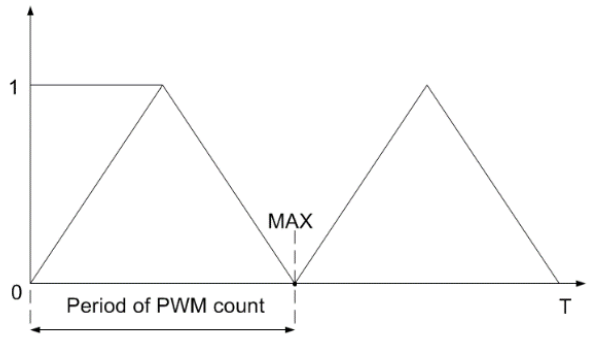

Figure 2.11 PWM triangle waveform

The calculation of the PWM period can be seen in equation (4):

$$T_{PWMOUT} = \frac{1}{f_{EPWM}} * 2 * MAX, \quad (4)$$

Where f_{EPWM} is the frequency of PWM, MAX is the period of PWM count.

The PWM frequency is inversely proportional to the period:

$$f_{PWMOUT} = \frac{1}{T_{pwmout}} \quad (5)$$

EPWM frequency is obtained from microcontroller 200 MHz frequency divided by 4. Period of EPWM at MAX point equals to 500 microseconds.

```
SysCtl_setEPWMClockDivider(SYSCTL_EPWMCLK_DIV_2);
SysCtl_enablePeripheral(SYSCTL_PERIPH_CLK_EPWM4);
EPWM_setClockPrescaler(EPWM_MODULE, EPWM_CLOCK_DIVIDER_2,
                            EPWM_HSCLOCK_DIVIDER_1);
EPWM_setTimeBasePeriod(EPWM_MODULE, 500U);
EPWM_setPhaseShift(EPWM_MODULE, 0);
EPWM_setTimeBaseCounter(EPWM_MODULE, 0);
EPWM_setTimeBaseCounterMode(EPWM_MODULE, EPWM_COUNTER_MODE_UP_DOWN);
EPWM_disablePhaseShiftLoad (EPWM_MODULE);

EPWM_setCounterCompareValue(EPWM_MODULE, EPWM_COUNTER_COMPARE_A, 0);
EPWM_setCounterCompareValue(EPWM_MODULE, EPWM_COUNTER_COMPARE_B, 0);
```

The microcontroller is measuring amplitude of the voltage in case of the alternating voltage measurement. For the programming code Phase-Locked Loop (PLL) block is used. A phase-locked loop is a feedback circuit designed to allow one circuit board to synchronize the phase of its on-board clock with an external timing signal. PLL circuits operate by comparing the phase of an external signal to the phase of a clock signal produced by a voltage-controlled crystal oscillator. The circuit then adjusts the phase of the oscillator's clock signal to match the phase of the reference signal. Thus, the original reference signal and the new signal are precisely in phase with each other [16].

Simplified diagram is shown in Figure 2.12, where k_p is proportional part of PI, $\frac{k_i}{s}$ is integral part of PI, and θ is the angle of the amplitude change

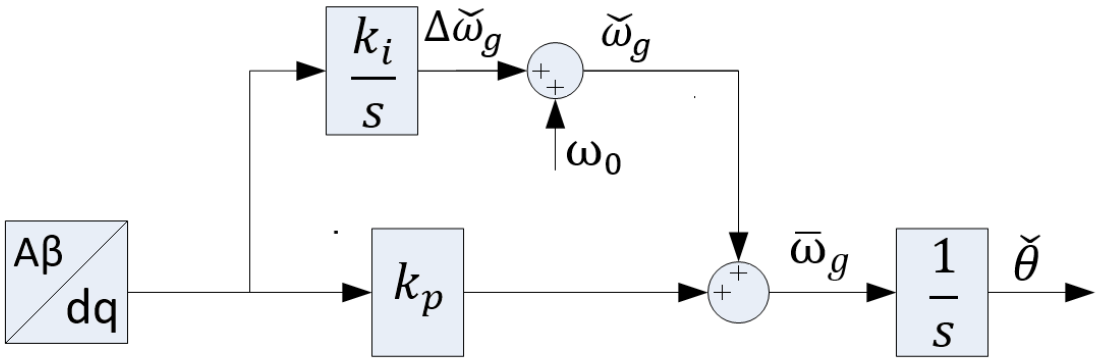

Figure 2.12 PLL block simplified diagram

As with the touchscreen calibration, final results for current and voltage were calibrated before transmitting it to the display. Calibration values are obtained experimentally. For example, several different known current or voltage values were fed into the ADC, and then VA_ZERO and VA_FACTOR was calculated. ADC, first, gives code numbers (for 16-bit ADC code numbers are from zero to 65535). These values need to be transformed to real voltage and current reading [18]. Final function for current and voltage measurement:

```
__interrupt void isr_func(void)
{
    Interrupt_clearACKGroup(INTERRUPT_ACK_GROUP3);
    EPWM_clearEventTriggerInterruptFlag(EPWM_MODULE);

    va = (float)ADC_readResult(ADCDRESULT_BASE, ADC_SOC_NUMBER1);
    ia = (float)ADC_readResult(ADCDRESULT_BASE, ADC_SOC_NUMBER0);
    vin = (float)ADC_readResult(ADCCRESULT_BASE, ADC_SOC_NUMBER1);

    Va = (va - VA_ZERO) * VA_FACTOR;
    Ia = (ia - IA_ZERO) * IA_FACTOR;
    Vin = (vin - VIN_ZERO) * VIN_FACTOR;

    adc_finish++;
    SOGI_PLL_phase_A();

}
```

Where ADC_readResult reads the result from the ADC, which will be additionally calibrated. To the display variables Va, Ia, Vin (capital letters) will be sent.

3. PROPOSED METHODS TO MEASUREMENT SYSTEM

As TMS320F28379D has two cores, it gives opportunity to divide tasks between cores. There are many benefits from using multicore microcontrollers such as balance performance and energy consumption, separation of concerns and workload distribution. In this thesis project one core is used to collect and process data from ADC, second core manages the LCD display (Figure 3.1). This parallel working principle gives stability to the system and increases speed of data processing (speed of ADC data collection is not influencing SPI data transferring to display. This feature expands capability to measure several parameters simultaneously without delay.

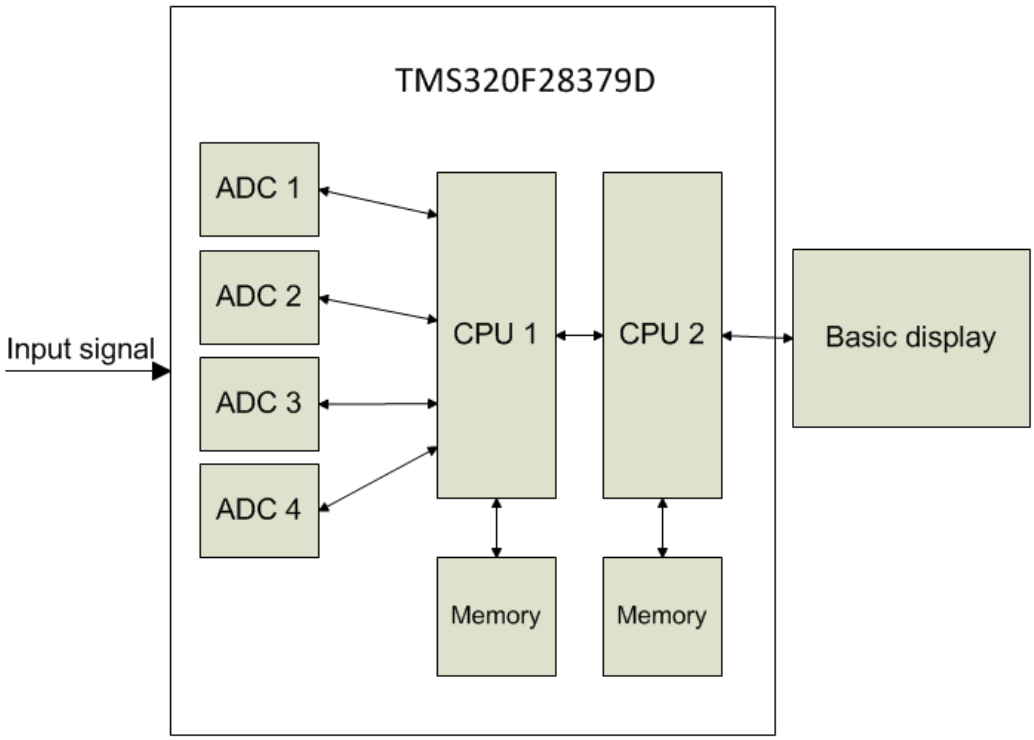

Figure 3.1 Project block-diagram

4. IMPLEMENTATION RESULTS

To ensure that everything is working properly, practical tests were made. The known value of the current and voltage from the power supply was applied to the control board. After that, values from the power supply were compared with the values that were calculated by the microcontroller. Some calculating mistake is still present, but it depends on power supply error, accuracy of the passive elements and ADC operations. Values are measured in the real time it means when changing values on the power supply, simultaneously values are changing on the display screen. Current, direct, and alternating voltage values on the display screen are switched using touchscreen. In figure 30 upper picture is showing the direct current value from power supply that is given to the microcontroller, below is calculated by microcontroller result that is shown on display. Results 1.18 A and 1.17 A are relatively close to each other, it means that calculation mistake is small.

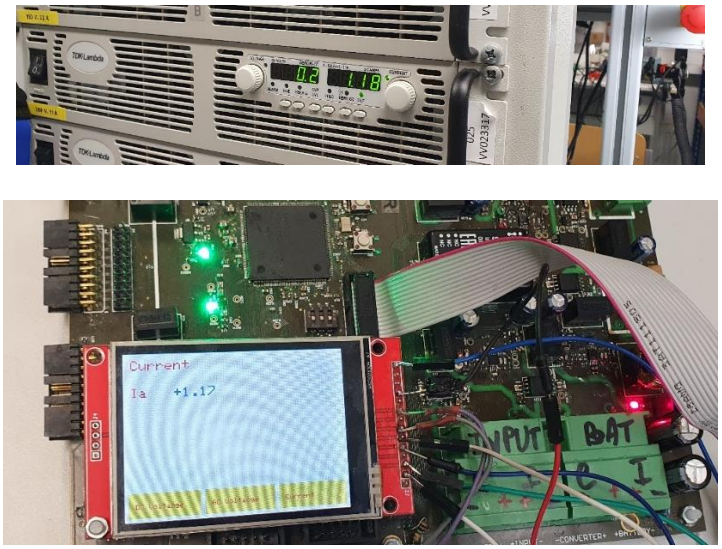

Figure 4.1 Current measurement

Figure 4.2 shows results of direct voltage measurement. Upper picture is showing voltage value from power supply, below is the calculated result from the microcontroller. Value are changing in real time. Small mistake is still existing but is relatively small: power supply value equals 69.9 V, while the microcontroller result is 70 V.

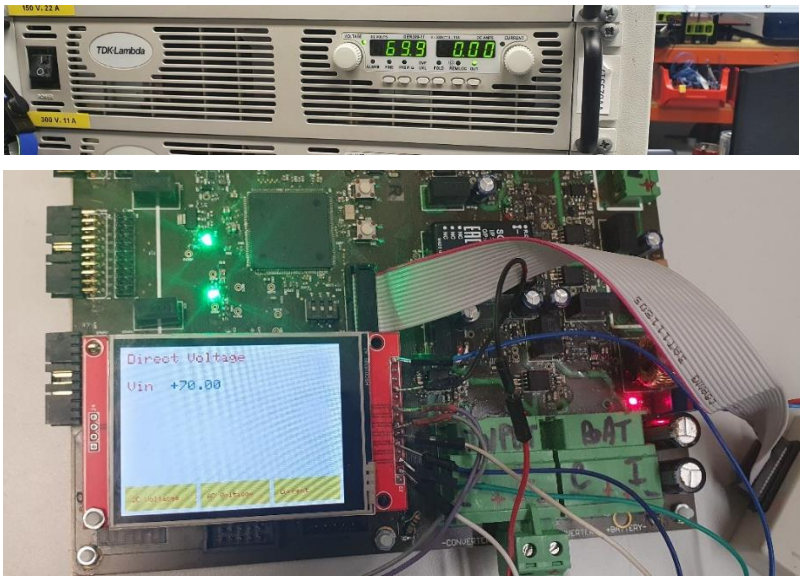

Figure 4.2 Direct Voltage Measurement

One of the parts of work was to measure voltage amplitude generated by the autotransformer and show amplitude peak value on the display. For that purpose, PLL block was created. In figure 4.3 upper picture shows real value measured by oscilloscope, below is the value which was calculated by microcontroller. Amplitude value is 69 V, value that is measured by microcontroller is 70,32 V. Mistake is existing but is relatively small.

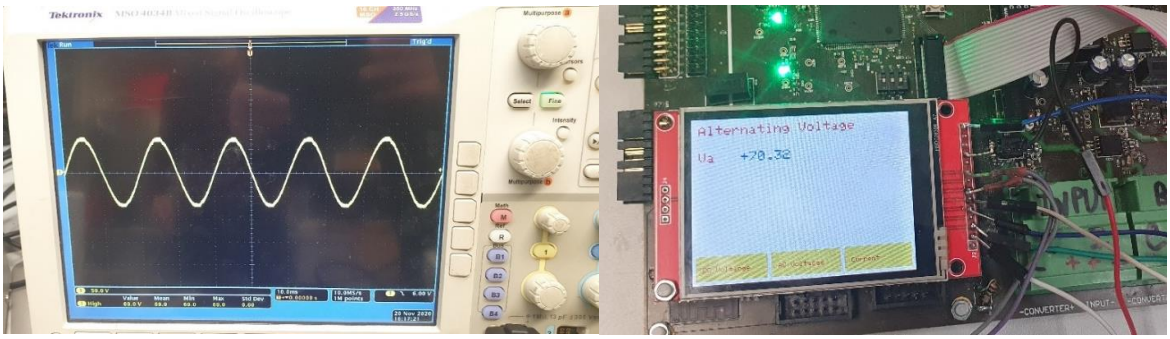

Figure 4.3 Autotransformer amplitude measurement

Touchscreen functionality was also controlled. Measurement are divided in three groups: Direct voltage, Alternating voltage and current. Using touchscreen, it is possible to switch between them. This helps to group measurement in a way that suits the user. All practical tests were successful.

SUMMARY

Microcontrollers are the microcircuits that are designed to control the electronic devices. There are a lot of MCUs from different manufactures were developed, which have different properties and used for the different tasks. Different digital interfaces are used to establish connection between MCU and different output/input devices, such as SPI, UART, CAN, Wi-Fi, Bluetooth, 1-wire, JTAG and I2C. One particular microcontroller (TMS320F28379D) from the Texas Instruments were studied. Four ADC modules in the TMS320F28379D were configured to measure current and voltage. The SPI connection between the microcontroller and the display with ILI9341 driver was successfully programmed and basic graphics library was created. It is possible to create voltage or current graphs in the future projects. This library will help to configure SPI communication of another microcontroller from Texas instruments with the ILI9341 display much faster. Touch functionality of the display is calibrated and working properly. Four ADC modules of the TMS320F28379D microcontroller are configured and working properly. Control board can measure current and voltage from five different power supplies simultaneously. Practical tests, that were made to ensure that ADC modules are working correctly, and the display output is correct, were successful. TMS320F28379D microcontroller has two cores and both are working at the same time. One core is controlling ADC modules, second core is processing an SPI connection between the MCU and the display. This combination gives a relatively cheap and reliable way to watch and control changes in power electronics circuits in real time. Thesis tasks of parallel measurement with the TMS320F28379D microcontroller was successfully accomplished.

KOKKUVÕTE

Mikrokontrollerid (edaspidi MCU) on mikrolülitused, mis on loodud elektroonikaseadmete juhtimiseks. Erinevate tootjate poolt on arendatud palju MCU-sid, millel on erinevad omadused ja mida kasutatakse erinevate ülesannete lahendamiseks. Erinevaid digitaalseid liideseid kasutatakse MCU ja erinevate väljund/sisend seadmete vahel ühenduse loomiseks, nt SPI, UART, CAN, Wi-Fi, Bluetooth, 1-wire, JTAG ja I1C. Uuriti ühte konkreetset Texas Instrumentsi mikrokontrollerit (TMS320F28379D). TMS320F28379D-i nelja ADC moodulit seadistati pinge ja voolu mõõtmiseks. ILI9341 draiveriga õnnestus programmeerida edukas SPI ühendus mikrokontrolleri ja kuvari vahel ning seeläbi loodi algne graafikakogu. On võimalik luua pinge ja voolu graafikuid tulevikuprojektide jaoks. See kogu aitab konfigureerida SPI suhtlust ühe teise Texas Instruments-i mikrokontrolleri ja ILI9341 kuvari vahel palju kiiremini. Kuvari puutetundlikkuse funktsionaalsus on kalibreeritud ja töötab korralikult. Juhtpaneel suudab mõõta voolu ja pinget samaaegselt viiest erinevast vooluallikast. Praktilised testid, mis viidi läbi tagamaks, et ADC moodulid töötavad korralikult ja kuvari väljund on õige, olid edukad. TMS320F28379D mikrokontrolleril on kaks tuuma ja mõlemad töötavad samal ajal. Üks tuum juhib ADC mooduleid, teine tuum töötleb SPI-ühendust MCU ja kuvari vahel. Selline kombinatsioon annab suhteliselt soodsa ja töökindla võimaluse jõuelektroonika ahelate jälgimiseks ja muudatuste kontrollimiseks reaalajas. Lõputöö ülesanded TMS320F28379D mikrokontrolleriga paralleelseteks mõõtmisteks täideti edukalt.

REFERENCES

[1] UM10204 - I2C-bus specification and user manual [Online]. Available: https://www.nxp.com/docs/en/user-guide/UM10204.pdf [Accessed 10.09.2020].

[2] 1 wire description [Online]. Available: http://avr.ru/beginer/understand/1wire [Accessed 10.09.2020].

[3] Xilinx official webpage [Online]. Available: https://www.xilinx.com/about/company-overview.html [Accessed 15.09.2020].

[4] JTAG description [Online] Available: https://en.wikipedia.org/wiki/JTAG [Accessed 15.09.2020].

[5] Texas Instruments official webpage [Online]. Available: https://www.ti.com/ [Accessed 20.09.2020].

[6] Arduino official webpage [Online]. Available: https://www.arduino.cc/ [Accessed 10.09.2020].

[7] SPI brief introduction [Online]. Available: https://www.arduino.cc/en/reference/SPI [Accessed 10.09.2020].

[8] CAN bus introduction [Online] https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en [Accessed 10.09.2020].

[9] Bluetooth description [Online]. Available: https://www.techopedia.com/definition/26198/bluetooth [Accessed 10.09.2020].

[10] STMicroelectronics official webpage [Online]. Available: https://www.st.com/content/st_com/en.html [Accessed 10.09.2020].

[11] Microchip official webpage [Online]. Available: https://www.microchip.com/ [Accessed 10.09.2020].

[12] TMS320F28379D datasheet [Online]. Available: https://www.ti.com/lit/ds/symlink/tms320f28379d.pdf?ts=1606462776181&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTMS320F28379D [Accessed 10.10.2020].

[13] TMS320F2837xD Technical Reference Manual (Rev. I) [Online]. Available: https://www.ti.com/lit/ug/spruhm8i/spruhm8i.pdf?ts=1606456599989 [Accessed 1.11.2020].

[14] ILI9341 driver datasheet [Online]. Available: http://www.lcdwiki.com/res/MSP2202/ILI9341_Datasheet.pdf [Accessed 1.11.2020].

[15] XPT2046 control driver datasheet [Online]. Available: https://ldm-systems.ru/f/doc/catalog/HY-TFT-2,8/XPT2046.pdf [Accessed 1.11.2020].

[16] Phase-Locks Loop description [Online]. Available: https://www.allaboutcircuits.com/technical- articles/what-exactly-is-a-phase-locked-loop-anyways/ [Accessed 10.11.2020].

[17] Training materials about ADC [Online]. Available: https://training.ti.com/ti-precision-labs-adcs-understanding-and-calibrating-offset-and-gain-adc-systems [Accessed 10.11.2020].

[18] ADC calibration [Online]. Available: https://www.ti.com/europe/downloads/f2810_12_calibration_10.pdf [Accessed 10.11.2020].

[19] ACS720 datasheet [Online]. Available: https://www.allegromicro.com/~/media/Files/Datasheets/ACS720-Datasheet.ashx [Accessed 1.12.2020].

[20] ACPL datasheet [Online]. Available: https://datasheetspdf.com/pdf-file/1251203/AVAGO/ACPL-C87AT/1 [Accessed 1.12.2020].

[21] AD8479 datasheet [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/AD8479.PDF [Accessed 1.12.2020].

[22] Atmel corporation description [Online]. Available: https://en.wikipedia.org/wiki/Atmel [Accessed 12.09.2020].

[23] Altera corporation description [Online]. Available: https://en.wikipedia.org/wiki/Altera [Accessed 12.09.2020].

[24] Microcontroller overview [Online]. Available: https://www.allaboutcircuits.com/technical-articles/what-is-a-microcontroller-introduction-component-characteristics-component/ [Accessed 12.09.2020].

[25] SPI description [Online]. Available: https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html [Accessed 12.09.2020].

[26] FPGA description [Online]. Available: https://circuitdigest.com/tutorial/what-is-fpga-introduction-and-programming-tools [Accessed 25.09.2020].

[27] PWM description [Online]. Available: https://en.wikipedia.org/wiki/Pulse-width_modulation [Accessed 10.12.2020].

[28] UART description [Online]. Available: https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter [Accessed 12.09.2020].

5. EXTRAS

5.1 Workplace

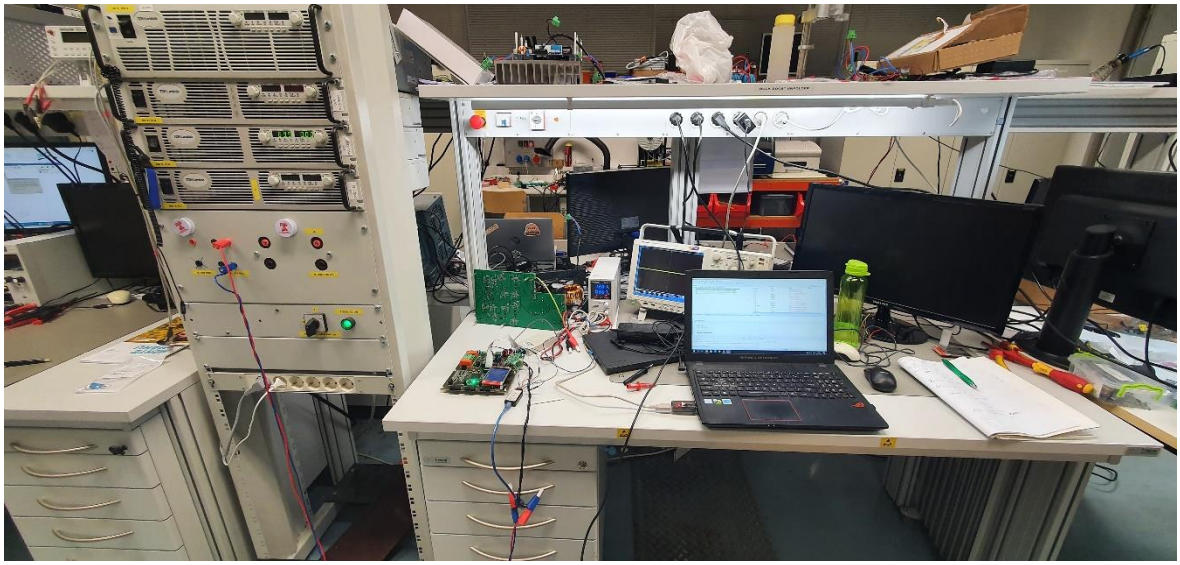

Figure 5.1 Workplace

5.2 Programming Code

Main body

```
#include "LCD_LIB.h"
#include "measuremnt.h"
#include "adc_config.h"
int16_t adc1, adc2, adc3;
uint16_t px,py;
uint32_t adc_finish;
uint32_t pll_flag;
__interrupt void isr_func(void);
void main(void)
{
  Device_init();
  Device_initGPIO();
  Interrupt_initModule();
  Interrupt_initVectorTable();
```

```
init_EPWM();
initADC();
initADCSOC();
Interrupt_register(INT_EPWM4,  &isr_func);
Interrupt_enable(INT_EPWM4);
lcd_spi();
EINT;
ERTM;
lcd_init();
ili9341_setRotation(1);
TP_Init(1, LCD_W, LCD_H);
// main menu
ili9341_clear(WHITE);
ili9341_fillrect(5,200,100,35,YELLOW);
ili9341_fillrect(110,200,100,35,YELLOW);
ili9341_fillrect(215,200,100,35,YELLOW);
ili9341_setcursor(15, 215);
ili9341_settextcolour(RED, YELLOW);
ili9341_settextsize(1);
ili9341_string("DC Voltage");
ili9341_setcursor(120, 215);
ili9341_string("AC Voltage");
ili9341_setcursor(225, 215);
ili9341_string("Current");
            adc1 = 0;
            adc2 = 0;
            adc3 = 0;
while(1)
{
```

```
TP_Scan(0);
  if (TP_Get_State() & TP_PRES_DOWN)
   {
      px = x;
      py = y;
   }
     if (TP_Get_State() & TP_PRES_DOWN)
     {
     // ADC Menu 1
      if(((px > 5) && (px < 100)) & ((py > 200) && (py < 235)))
        {
            ili9341_fillrect(5, 5, 310,190, WHITE);
            ili9341_setcursor(10, 10);
            ili9341_settextcolour(RED, WHITE);
            ili9341_settextsize(2);
            ili9341_string("Direct Voltage");
            ili9341_setcursor(10, 50);
            ili9341_string("Vin");
            adc1 = 1;
            adc2 = 0;
            adc3 = 0;
        }
     //ADC Menu 2
      if(((px > 110) && (px < 210)) & ((py > 200) && (py < 235)))
        {
            ili9341_fillrect(5, 5, 310,190, WHITE);
            ili9341_setcursor(10, 10);
            ili9341_settextcolour(RED, WHITE);
            ili9341_settextsize(2);
```

```
            ili9341_string("Alternating Voltage");
            ili9341_setcursor(10, 50);
            ili9341_string("Va");
            adc1 = 0;
            adc2 = 1;
            adc3 = 0;            }
       //ADC Menu 2
        if(((px > 215) && (px < 315)) & ((py > 200) && (py < 235)))
            {
              ili9341_fillrect(5, 5, 310,190, WHITE);
              ili9341_setcursor(10, 10);
              ili9341_settextcolour(RED, WHITE);
              ili9341_settextsize(2);
              ili9341_string("Current");
              ili9341_setcursor(10, 50);
              ili9341_string("Ia");
              adc1 = 0;
              adc2 = 0;
              adc3 = 1;
             }
     }
       if(adc_finish > 5000)
           {
adc_finish = 0;
     //For Vin
   if (adc1 &&! adc2 &&! adc3)
     {
        ili9341_setcursor(70, 50);
        ili9341_settextcolour(BLUE, WHITE);
```

```
            ili9341_settextsize(2);
            ili9341_fdig_string(Vin);
          }
         //For Va
      if (adc2 &&! adc1 &&! adc3)
         {
            ili9341_setcursor(70, 50);
            ili9341_settextcolour(BLUE, WHITE);
            ili9341_settextsize(2);
            ili9341_fdig_string(Vgrid_A);
          }
         //For Iin
      if (adc3 &&! adc2 &&! adc1)
          {
            ili9341_setcursor(70, 50);
            ili9341_settextcolour(BLUE, WHITE);
            ili9341_settextsize(2);
            ili9341_fdig_string(Ia);
          }
      }
  }
}
__interrupt void isr_func(void)
{
   Interrupt_clearACKGroup(INTERRUPT_ACK_GROUP3);
   EPWM_clearEventTriggerInterruptFlag(EPWM_MODULE);
 //   ADC_clearInterruptStatus(ADCD_BASE, adcIntNum)
   va = (float)ADC_readResult(ADCDRESULT_BASE, ADC_SOC_NUMBER1);
   ia = (float)ADC_readResult(ADCDRESULT_BASE, ADC_SOC_NUMBER0);
```

```
    vin = (float)ADC_readResult(ADCCRESULT_BASE, ADC_SOC_NUMBER1);
    Va = (va - VA_ZERO) * VA_FACTOR;
    Ia = (ia - IA_ZERO) * IA_FACTOR;
    Vin = (vin - VIN_ZERO) * VIN_FACTOR;
    adc_finish++;
    SOGI_PLL_phase_A();
}
```