VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

DEVELOPMENT OF ELECTRONICS FOR TRACTION CONTROL OF EXPERIMENTAL VEHICLE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JOSEF VEJLUPEK

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

FACULTY OF MECHANICAL ENGINEERING

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

DEVELOPMENT OF ELECTRONICS FOR TRACTION CONTROL OF EXPERIMENTAL VEHICLE

VÝVOJ ELEKTRONIKY PRO ŘÍZENÍ TRAKCE EXPERIMENTÁLNÍHO VOZIDLA

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. JOSEF VEJLUPEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. ROBERT GREPL, Ph.D.

SUPERVISOR

BRNO 2010

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav mechaniky těles, mechatroniky a biomechaniky
Akademický rok: 2009/2010

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Josef Vejlupek

který/která studuje v **magisterském navazujícím studijním programu**

obor: **Mechatronika (3906T001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Vývoj elektroniky pro řízení trakce experimentálního vozidla

v anglickém jazyce:

Development of electronics for traction control of experimental vehicle

Stručná charakteristika problematiky úkolu:

Práce se bude zabývat kompletním vývojem elektronické části vozidla se 4 plně řiditelnými koly pro účely testování trakčních algoritmů a dalších elektronických systémů jízdní bezpečnosti a jízdního komfortu.
Úkol spočívá predevším v nalezení vhodného rozdělení jednotlivých výpočetních úloh mezi mikrokontroléry a jejich vzájemné komunikace. Jedná se především o časově kritické úlohy.

Cíle diplomové práce:

1) Návrh a sestavení výkonového stupně pro řízení DC motoru (napětí 35V, maximální proud 20A).
2) Návrh architektury sítě mikrokontrolérů pro řízení vozidla s podporou systému jízdní stability.
3) Vývoj jednotlivých jednotek pro řízení pohonů a zpracování sensorické informace, komunikace po sběrnici CAN.
4) Vývoj dálkového řízení vozidla.
5) Využití technik HIL při testování řídicí elektroniky.

Seznam odborné literatury:

- Valášek, M.: Mechatronika, Vydavatelství CVUT 1995
- Noskievic: Modelování a identifikace systému
- Vlk, F.: Dynamika motorových vozidel. Brno 2000
- Vlk, F.: Asistenční a informační systémy motorových vozidel. Automobilová elektronika 1. Brno 2006
- Vlk, F.: Systémy řízení podvozku a komfortní systémy. Automobilová elektronika 2. Brno 2006
- Grepl, R.: Modelování mechatronických systému v Matlab/SimMechanics, BEN, 2007

Vedoucí diplomové práce: Ing. Robert Grepl, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2009/10.

V Brně, dne

L.S.

prof. Ing. Jindřich Petruška, CSc.
Ředitel ústavu

doc. RNDr. Miroslav Doupovec, CSc.
Děkan fakulty

Abstrakt

Tato práce se zabývá návrhem a realizací palubní elektroniky experimentálního vozidla Car4, dále pak základní programovou výbavou řídicí jednotky a Hardware In the Loop simulačním ověřením funkčnosti řídicí jednotky.

Abstract

This thesis deals with design of on-board electronics of experimental vehicle Car4, with basic firmware of control unit, and Hardware In the Loop simulation for verification of Control Unit function.

Thanks

I would like to thank to my family for their support and trust in me. To my supervisor Ing. Robert Grepl, Ph.D. for his patience and guidance, and to my friends and colleagues from MechLab, who worked with me on the same project: Michal Jasanský, Vojtěch Lamberský, and Matěj Šimurda.

Declaration on word of honour

I statutory declare, that I wrote this paper by myself with usage of stated literature and under supervision of my instructor.
Josef VEJLUPEK, Brno, 2010

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Project goals

This thesis deals with my work on project Car4. This project is supervised under the MechLab: Mechatronics laboratory on FME VUT Brno.[1] Primary goal of the project Car4 is to prepare a small vehicle with all wheel drive (AWD), that would be used as a tool for education, a demonstration and experimental purposes. There are many tasks available on a such complex project:

- Design of the mechanical construction of the car itself, including motor drives and steering mechanism.
- Design of the electronic components: DC motor power drive, control unit, power supply, sensor array.
- System modelling and identification, parameter estimation.
- Control algorithms development and testing: This is one of the most interesting goals: To develop and test various traction control algorithms such as ABS[2], DSTC[3], etc.
- Software implementation onto selected platform. Algorithm testing and tuning.

[1] Faculty of Mechanical Engineering - Brno University of Technology.
[2] Anti-lock brakes
[3] Dynamic Stability and Traction Control

1.2 Goals defined for this thesis

As this project is quite complex, there are multiple people working on different tasks. My job was to develop and build electronic hardware including main on-board control unit, power electronic for motors and communication subsystem for remote control. Goals given for this diploma thesis:

1. Design and making of Power unit capable of driving DC motor with up to 35 Volts and 20 Amps.
2. Design and making of array of Control units as we have assumed that one micro-controller would not be enough.
3. Implementation of Control units, communication between them, and communication with sensory array.
4. Remote control development and implementation.
5. HIL[4] testing.

Additional work was done based on the natural needs of the project to be completed: We had to create basic Firm-Ware for the control units, some mechanical modifications on the Car4, and also test all parts together for reliability and durability.

1.3 Goals defined for my colleagues thesis

In this section are individual project goals as they have been divided between my colleagues described in their own words.[5]

1.3.1 Vojtěch Lamberský - Development of algorithms state estimation of experimental vehicle

> Development of simple algorithms for dsPIC microcontrollers using MATLAB embedded coder tools.
> Connecting PhyTec development board with MPC555 to other on-board control units using ECAN.
> Testing and tuning of advanced algorithms implemented in Car4 on-board control units.

Thesis citation: [39]

[4] Hardware In the Loop.
[5] Following text is translation of their original.

1.3.2 Michal Jasanský - Design of dynamic models for traction control of experimental vehicle

Thesis is dealing with design of dynamic models for state estimation and traction control of a vehicle with all wheel drive and steer. From the many options described in literature we have picked and modified models suitable in terms of dynamic description, and also suitable for implementation in terms of computational power, for our real experimental vehicle Car4.

Thesis citation: [40]

1.3.3 Matěj Šimurda - Design and implementation of complementary sensory system for experimental car

Thesis is dealing with design and realization of complementary sensor array for Car4 project. These sensors tasks are: measurements of dampers position, motor temperature monitoring, and range detection sensors for collision avoidance. Also the part of control unit firmware, which deals with motor overheat protection and collision avoidance has been designed and implemented.

Thesis citation: [38]

1.3.4 Filip Valdejch - Design of experimental vehicle undercarriage with four wheel steering

This bachelor thesis covers design and manufacture of experimental four-wheeled vehicle undercarriage, with all-wheel steering and all-wheel drive. The vehicle is supposed to serve as a teaching aid or prototype for new system development. The vehicle is able to test all-wheel steering concept and systems for steering, drive and braking control. The thesis includes several concepts of vehicle design and also technical details, such as wheel mount and driving power distribution.

Thesis citation: [41]

Chapter 2

Vehicle control design

This chapter is describing the development of the Car4 vehicle control scheme including all the peripherals and development of the basic control firmware.

2.1 Control Unit Concept

From the project specification, we have assembled following requirements for the *Control Unit*:

- Microcontroller based platform (cheap, not based on PC.)
- Peripherals and/or communication ports for external sensors (PWM, QEI, SPI, I2C, ...)
- Easy to develop, debug, and use. (With technology already available in our laboratory [1])
- Enough computing power to implement control algorithms, such as Anti-lock brakes, digital stability and traction control, ...
- Remote control for driver's command, but all control algorithms running on-board.
- Data from vehicle telemetry available for post-processing and evaluation.

Some of these requirements were quite abstract, as we did not have enough experience i.e. with the computing power assessment. As we will show later on, this did not prove to be a big problem, as there are very few platforms, that provide enough computing power, and meet all the peripheral requirements.

2.1.1 Available options

We have been limited by our goals to a very narrow spectrum of controllers from a few companies known to us:

- Microchip - PIC, dsPIC: 8-bit[1] and 16-bit microcontrollers.
- Freescale - 8-bit microcontrollers and DSPs[2].
- Atmel - more specialized in stepper motors and BLDC[3] motors.

As we wanted to implement PWM motor control, CAN bus, SPI, I2C, and UART communication, and also to utilize rapid prototyping method in software development (Kerhuel toolbox 6.1.2) we have been lead to use the dsPIC from Microchip.
Other option would be to use some final product, that enables motor control. Such modules are available i.e. from Maxon, or various hobby-robotics shops. Maxon ones are quite expensive, the other do not provide required features or do not meet the power specifications we initially had.

There are only few controllers that satisfy all of our requirements. Most limiting was the criteria of having two QEI modules, and also motor PWM (stand alone and therefore accurate peripheral). This left us with the 16-bit dsPIC family. We have decided to use two microcontrollers, one for each axle as there would not be enough pins on one microcontroller to drive both axles and other devices on board. One of our goals was to implement CAN bus, and this choice provides ideal opportunity to do so. This decision lead to the first concept of the Car4 on-board controller structure.
Next concept was based on demanding computing power requirements, we have decided to use *MPC555* from Freescale, which gave us comfort of FPU[4]. This unit should be able to run the Car4 on it's own, as it has enough peripherals. However it was not our intention from the beginning, and such approach would be probably too easy.
Another option was to use a 32-bit microcontroller from Microchip, that would read the data from the axle control units and sensors, run more complicated algorithms, and send the low level control commands back to the axle control units.
At the beginning, we have also considered using more distributed scheme:

[1] 8-bit microcontrollers are not available with CAN.
[2] Digital Signal Processors
[3] Brush-Less Direct Current
[4] Floating Point Unit

One controller for communication with Car4 operator, one for each axle motion control, one or two for the sensory array. This would be more real-car-like, but would mean to use more components, and careful selection of each individual controller to suit the needs. So we have decided for more universal solution, using two or three basic concepts: ACU28, ACU44, and optionally one other, i.e.: MPC555.

2.1.2 Realized options

We have started with the simple scheme of two axle control units based on dsPIC33FJ128MC804 from Microchip [13], which is 16-bit microcontroller. For more details, about the hardware itself, please refer to subsection 3.2.3. Connection scheme is shown in figure 2.1 in green.
With this layout, we have run the basic control of the Car4 vehicle: steering, speed control of motors (with PID regulator), and realized basic sensor readings (quadrature encoder, current, and data from ADIS[5]). Later demands for easy implementation lead to use of *MPC555* from phyCORE [26], shown in figure 2.1 in blue.
Last part, shown in figure 2.1 in red / orange, is *Secondary sensor array*, which has been designed by Matěj Šimurda in his bachelor thesis: [38].

2.2 Realization

This section describes our realization of the final on-board electronics, and reasons for this scheme.

2.2.1 Specifications

In this subsection, we present the main features and specifications for the Control Units and Sensors. These specifications have been based upon the project specifications discussed in 1.1, and 2.1.

Control Units

We are using two *ACU44* (Axle Control Unit based on dsPIC 3.2.3), each *ACU44* is driving one axle (front - master / rear - slave). Each axle is pieced together with two PD4266 DC motors from Transmotec [27], to power them, there is a *ISL Power Board* 3.1, which creates the power link between *ACU44* and motor. Another board is used to drive the steering servos from Scanner

[5]Please refer to subsection 2.2.1 for more informations about used sensors.

RC [28].
We are using term Master for the front *ACU44*, and Slave for the rear *ACU44*. This is based on assumption, that the front *ACU44* is where the advanced control algorithms (such as ABS / ASR) are running. In the implementation without *MPC555*, the Master unit also receive operator's commands and transmits telemetry using UART port. This link is realized using wireless UART realized by *RS232 Remote Unit* based on HW86010, unit is described in section 3.4.
Interconnection of the *ACU44*s is realized using CAN bus. Connection to *ISL Power Board*s is realized through digital logic lines: PWM, DIR, and DIS, those being outputs from *ACU44*, and one analog voltage signal, which is input for *ACU44* ADC. *Servo interface board* uses two digital lines with PWM, one for each servo. Lines are wired together in one cable assembly with the second of the *ISL Power Board*s[6].

ACU44 specification:

- based on dsPIC33FJ128MC804 from Microchip
- controls two DC motors via ISL Power Boards (speed / position or current - momentum drive)
- controls two servomotors
- communication buses: CAN bus 2.0A and 2.0B capable, I2C, SPI, UART
- several additional digital input/output lines are available
- $3.3V$ operation, $5V$ input tolerant[7]

Sensors

On-board sensors are essential for the Car4 operation and control. There are several level of sophistication determined by the complexity of implemented control algorithms.

On-board Sensors: Most basic sensors are encoders on the DC motors. We can determine the distance and also the speed[8] of each wheel. Another information, that we have about the motors state is the current read-out. It

[6] Connector labelled ISL2 on the ACU44 board.
[7] Please refer to the dsPIC33FJ128MC804 datasheet.
[8] Proper filtration and derivation need to be implemented.

is realized through current transducer LTS-25NP 3.1.2. This sensor gives us analog voltage signal which is proportional to the current that goes through the motor, than the signal is measured with ADC on the *ACU44*.

For the more advanced and precise control algorithms, there is also ADIS 16405 [29] from Analog Devices. This complex high precision sensor is incorporating tri-axis gyroscope, accelerometer and magnetometer. It uses serial SPI interface to send data and to receive configuration commands. It also incorporates temperature sensor.
During the development, we have come to conclusion, that the datasheet for the device is incorrect: when we have tried to read from the addresses stated in the datasheet[9], the data did not match. We have reached the conclusion, that there is a mistake in the datasheet in Table 8: "User Register Memory Map". We have successfully used our table 2.1, these corrections are quite obvious. For the algorithms implemented so far, we make use of the accelerometer's and gyroscope's readouts. We have also briefly tested the magnetometer readouts, and the data came out quite nicely. It could be used as a compass or possibly even as an artificial horizon in combination with accelerometer. Furthermore it can be used in advanced filtering techniques such as Kalman filter to obtain more precise results.
ADIS 16405 has default settings of sample rate set to 819.2 SPS[10], this is

Register name	Address from datasheet	Address used
XGYRO_OUT	0x04	0x0200
YGYRO_OUT	0x06	0x0300
ZGYRO_OUT	0x08	0x0400
XACCL_OUT	0x0A	0x0500
YACCL_OUT	0x0C	0x0600
ZACCL_OUT	0x0E	0x0700
XMAGN_OUT	0x10	0x0800
YMAGN_OUT	0x12	0x0900
ZMAGN_OUT	0x14	0x0A00

Table 2.1: SPI Memory map correction

according to datasheet set for optimal performance. However, it can be set for 1638.4 SPS (which is twice as fast as the default value.) It would lower the bandwidth. As we are currently running the fastest loop at $200Hz$, we

[9] Rev B 07/2009
[10] Samples Per Second

can go down to 409.6 or even 204.8 SPS. Precision could be also increased by selecting lower bandwidth / sensitivity. ADIS also provides four configurable digital I/O lines, one 12-bit Analog Input (ADC), and one 12-bit Analog Output (DAC), but we are not using those.
We find important to comment on fragility of ADIS connector: it is very sensitive to proper manipulation, if not handled carefully, it can be damaged quite easily: wires tends to rip off the cable.

Additional Sensors: Additional sensor array, designed and realized by Matěj Šimurda [38], features motor temperature monitoring, collision avoidance system, and shock absorber position monitoring.

2.2.2 Used Hardware

Hardware (HW) which has been developed in this project is described in following chapter 3. So we will briefly describe here only MPC555, which we also used, but as a final product we did not modify it in any way.

MPC555

MPC555 manufactured by Freescale [20] is PowerPC™32-bit microcontroller with FPU[11] featuring:[12]

- *40MHz Core with Floating Point Unit*
- 26 Kbytes of Static RAM
- 448 Kbytes Flash EEPROM Memory with 5-V programming (CMF)
- Flexible Memory Protection Unit
- General-Purpose I/O Support
- Two Time Processor Units (TPU3)
- 18-Channel Modular I/O System (MIOS1)
- Two Queued Analog-to-Digital Converter Modules (QADC)
- *Two CAN 2.0B Controller Modules (TouCANs)*
- *Queued Serial Multi-Channel Module (QSMCM)*

[11] Floating Point Unit
[12] List has been adopted from Freescale web page.

- U-Bus System Interface Unit (USIU)

We have emphasized features, that are most important for us. As we have stated earlier, we have decided to use MPC555 because need of powerful platform to compute advanced algorithms without having to tailor them into fixed point. From talking to people at BOSCH, we know, that this exact microcontroller is being used in automotive industry as an ECU.[13] We are using whole development board phyCORE [26] made by PHYTEC.
To create and compile the code, we are using MATLAB Simulink RTT, FM5 toolbox, and Code-Warrior. Simulink Real-Time-Toolbox generates the C-code, FM5 toolbox adds the target specific code, and Code-Warrior then compiles it into machine code and loads it to the target. Code development for MPC555 have been done mainly by Vojtěch Lamberský and Michal Jasanský, please refer to their thesis for more details: [39] and [40].

[13]Engine Control Unit also known as PCM: Power-train Control Module

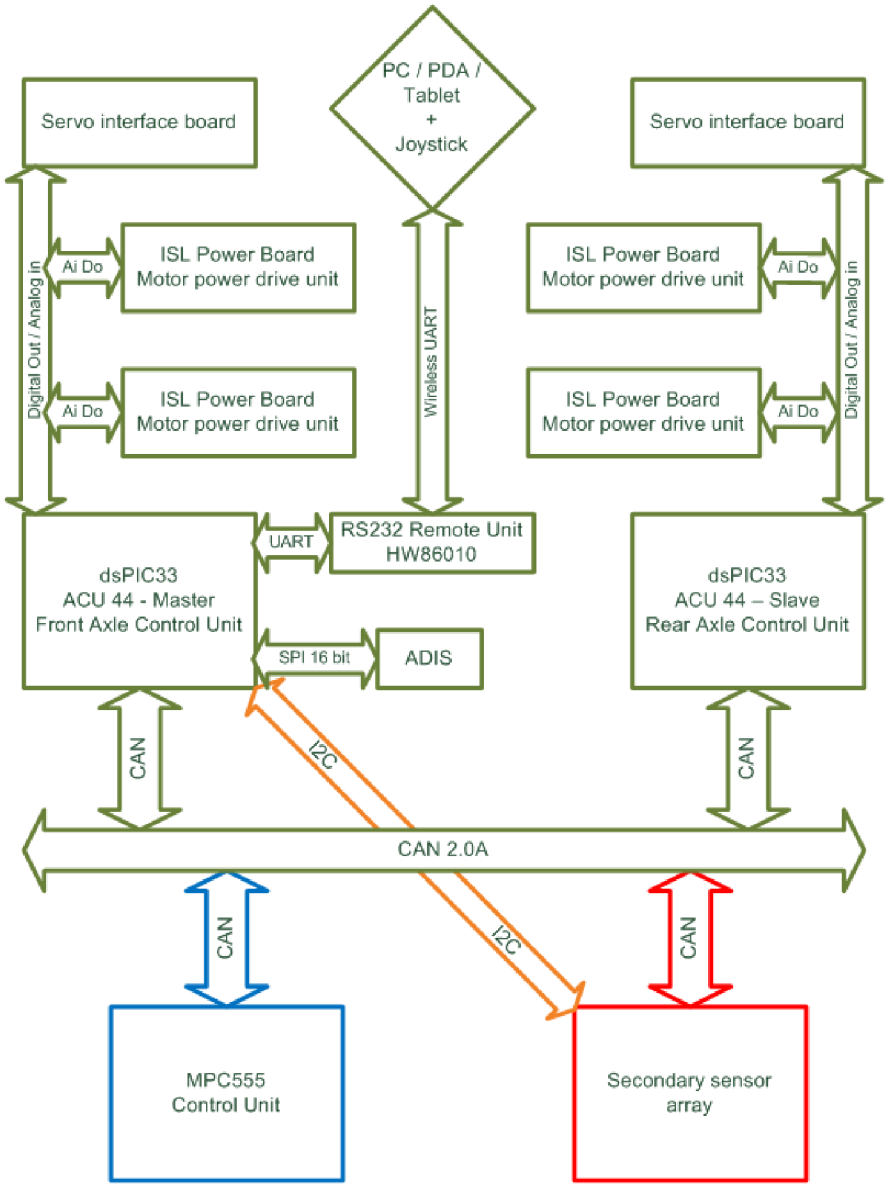

Figure 2.1: On-Board devices (Power Sources are not included)

Chapter 3

Development of On-Board Hardware and Remote Control

3.1 ISL Power board

ISL Power Board is a H-bridge designed for unipolar PWM[1] drive of a small DC motor. We are using four ISL Power Board units to power four drive motors in the Car4 project. This section describes how to use the unit, and gives some advice on it's operation.

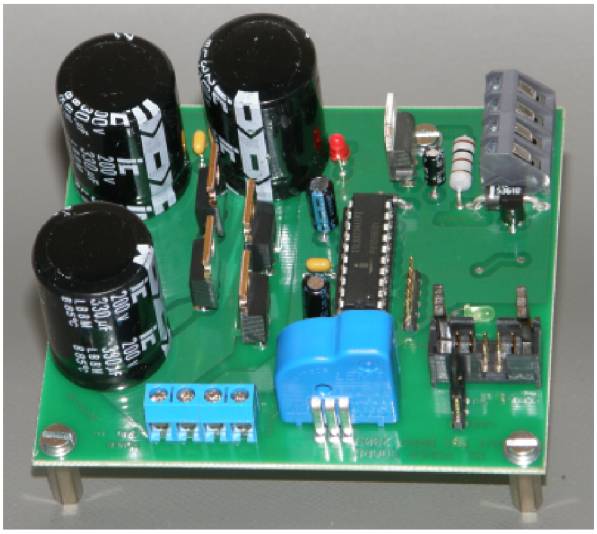

Figure 3.1: ISL Power Board, without heat-sink installed

[1]Pulse Width Modulation

3.1.1 Requirements - Specification

Requirements given by the project Car4 specification:

- Be able to supply DC motor operating on 35 Volts at 20 Amps. (Up to $700W$.)[2]
- Galvanic isolated signal inputs: PWM, DIR[3], (DIS[4])
- PWM should be able to run at $50kHz$
- Galvanic isolated analog output: current via motor
- Compact size

Final product specification:

- Capable of supplying DC motor operating up to 60 Volts at 105 Amps (theoretical maximum)
- PWM should be able to run at $10 \sim 100kHz$
- Tested maximum rating: $30V$ $30A$ peak, $60V$ $3A$, and $30V$ $8A$ continuously at $10kHZ$.[5]
- Galvanic isolated signal inputs: PWM, DIR, DIS
- Galvanic isolated analog (voltage) output: current via motor up to $\pm 38A$[6]
- Size: 94mm x 82mm x 40mm (height without cooler)

[2]This was preliminary specification, which now can seem over-designed, use of the unit was not intended only for Car4 project, but also for other projects supervised by MechLab[1]

[3]Direction

[4]Recommended: Disable

[5]There is limitation due to thickness of the copper layer on the PCB. It can be strengthened by additional wires connected in parallel to the power lines, or by adding some solder on pathways.

[6]$\pm 80A$ with minor change in board layout

3.1.2 Main components used in design

ISL83204A

ISL83204A[2] from Intersil is a high frequency, medium voltage full bridge N-Channel FET driver IC. According to it's data-sheet, it can drive $1000pF$ Load at $1MHz$, with $60V$ and $2.5A$. Device is available in DIP20 and SOIC package, we have used the DIP variant. Typical applications for this circuit are medium/large voice coil motors drive, full bridge power supplies, switching power amplifiers, UPS, high performance motor controls, noise cancellation systems, and battery powered vehicles. Circuit features user-programmable dead time, under-voltage protection and several other functions that make it really simple to use with just a few external components. Key features:

- Drives N-Channel FET Full Bridge
- Bootstrap Supply Max Voltage to $75V$
- Programmable dead time
- Input Logic Thresholds Compatible with $5V$ to $15V$ Logic levels[7]
- No DIR pin: has two input comparator.

Table 3.1 shows the logic operation of the H-bridge. It's function is shown in figure 3.2. As mentioned above, ISL83204A does not have DIR pin, to determine the direction, it compares analog values of voltage on pin IN+ and IN-. We have designed the board so that on IN- is set constant voltage (approx $2.4V$), and IN+ is set to low state ($0V$) or high state ($5V$) via DIR input on the board. HEN pin is fed by PWM signal. DIS (disable) pin is enabled by DIS input pin, and also turned high if the controller side of ISO7241C is powered down.

IRFB4115

To fulfil the power transfer criteria, we need suitable transistor. Main requirements are:

- Voltage - determined by specification in 3.1.1, to be safe we should at least double.
- Current - determined by specification in 3.1.1, to be safe we should add at least 50%.

[7] High state from $2.5V$

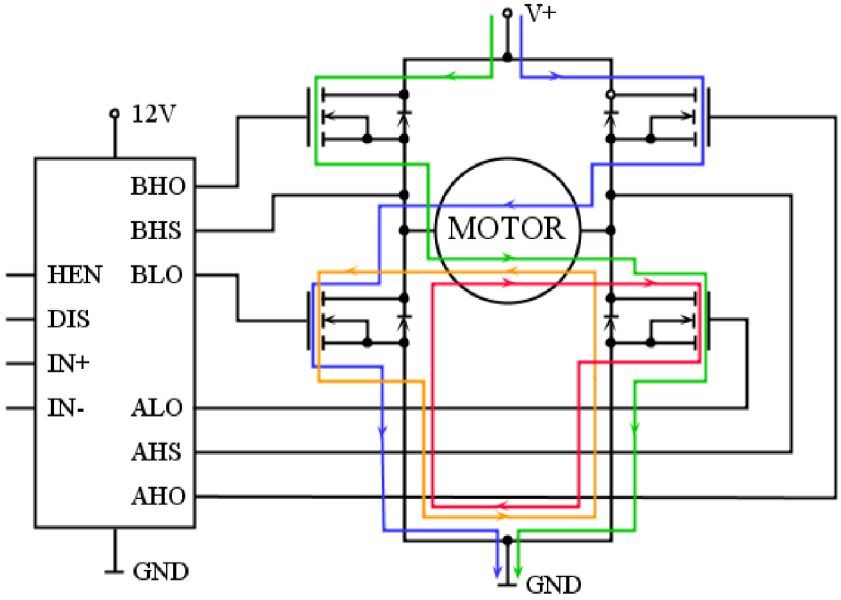

Figure 3.2: Current Flow through an H-Bridge

- As short dead-time[8] as possible.
- As low $R_{DS(on)}$ as possible.
- N-channel FET

When we started to design first version, we looked for device that could be easily bought on Czech market[9], that mostly limited us to products of International Rectifier. In the last version, we have decided to use IRFB4115[4][10]. It's parameters are:

- $V_{DSS} = 150V$
- $I_D = 104A$
- dead-time of $41 + 39 = 80ns$
- very low $R_{DS(on)} = 11m\Omega$
- Build-in body diode

[8] determined by sum of "Turn-Off Delay Time" and "Fall Time"
[9] In GM Electronic, GES, ...
[10] Not available at GME, but order is possible from Farnell[3]

INPUT				OUTPUT				color
IN+>IN-	**HEN**	**U/V**	**DIS**	**ALO**	**AHO**	**BLO**	**BHO**	in Fig 3.2
X	X	X	1	0	0	0	0	NA
0	1	0	0	1	0	0	1	green
0	0	0	0	1	0	0	0	red
1	1	0	0	0	1	1	0	blue
1	0	0	0	0	0	1	0	orange
X	X	1	X	0	0	0	0	NA

Table 3.1: ISL83204A operation - Truth table

ISO7421C

To meet the galvanic isolation requirement, we have used ISO7421C: High speed quad digital isolator[5]. Essential features of our design are:

- four digital channels in one package, and pin configuration
- signalling rate up to $25Mbit$
- full galvanic isolation
- High Electromagnetic Immunity
- $3.3V$ and $5V$ Supply and logic operation
- noise-filter

Unfortunately it has one feature, that was not really desirable, luckily it does not impact our design seriously: When one side is powered down, the other side's outputs are turned high (enable pin has to be high on that side). So when there is no connection on the controller side, the HEN (PWM) pin is turned high, but DIS (Disable) pin is high as well. On the other side, if the ISL Power Board is powered down, we can read high state on "ISL VOUT" pin, which was left unconnected on the ISL side, due to the mistake in initial design.

LTS 25-NP

To gauge the current through the motor, we have decided to use current transducer from LEM[7] LTS 25-NP[8]. We have used it in configuration that sets "primary nominal current rms" to $I_{PN} = \pm 12A$. This allows us to measure currents in range: $I_P = \pm 38A$ via corresponding DC voltage $U_{IP} = \langle 0.5, 4.5 \rangle V$. As we are using it with dsPIC33, which operates on $3.3V$

logic, we are limited with this setting to measure $I_P = \langle -38, +16 \rangle\, A$. This does not meet our initial requirements and leaves us with several options:

- To rescale the voltage signal with proper amplifier or resistor divider.
- To measure the voltage with some external ADC and send digital signal to dsPIC.
- To modify the board layout[11] for different setting on LTS 25-NP to change I_{PN}, this would decrease the resolution.

Under the current layout design, LTS25-NP is able to measure current up to $38A$. In order to get better resolution, LTS15-NP can be used. It would be able to measure up to $24A$.

Conversion of the voltage signal to current readout is done by simple linear equation:

$$I_{motor} = x = \frac{y - q}{k} = \frac{U_{LEM} - 2.5}{\frac{3.125-2.5}{12}} \tag{3.1}$$

Where $3.125V$ is the voltage readout on LEM at nominal current of $12A$, $2.5V$ is the voltage readout for zero current. This is not exactly true due to some offset of the zero current value: it is not exactly $2.5V$ and the value may vary a bit (tenths of mV) from piece to piece. Compensation due to this error has to be done when the device is being used. It could be done by measuring the "zero value" when the motor is disconnected or powered down.

3.1.3 How to use ISL Power Board

Goal of this section familiarize reader with details of using the ISL Power Board: How to connect it to power supply, control device, and how to design and install cooler.

Power Supply

ISL Power Board requires at least two power supplies:

- to provide power for the motor (up to $60V$), ISL83204A ($12V$) and ISO7421C (board side) ($5V$)
- to provide power for ISO7421C (controller side) ($3.15 \sim 5.5V$) and LTS 25-NP ($5V$)

[11] This could actually be done very easily, see LTS 25-NP data-sheet: [8]

On the board, there is $12V$ positive voltage regulator L7812 [9] which is powering ISL83204A and also through another $5V$ positive voltage regulator 78L05 [10] powering board side of ISO7421C. There is a terminal named SWITCH, that can be used to feed the L7812 with voltage from the motor supply. If the SWITCH terminal is disconnected, $12V$ has to be provided on the "12V IN/OUT" terminal, to power up the ISL83204A and board side of ISO7421C.

WARNING: *L7812 should be supplied with no more than* $35V$*!*

If you want to use higher voltage, you should consider connecting a power resistor to the SWITCH terminal.
Please note, that the $12V$ supply for ISL controller has common ground with the power side. Make sure, that there are no ground loops in your circuits. If you are powering the $12V$ input, do not connect the ground wire to the terminal, instead make sure that the ground (or minus terminals) are connected together as close as possible (i.e: If you are using one battery pack to power the motor, and another to power the $12V$ appliances make sure, that the minus terminals are connected together in one place.

Terminal named POWER is used to supply the power part of the board - drive the "high" voltage and current through FETs to the motor. It can be powered with DC voltage within range: $U_{pow} = \langle 0, 60 \rangle\, V$. In case you are using it via SWITCH terminal to power the ISL83204A, the supply voltage should be at least $14.5V$ to cover for L7812 voltage drop-down!
WARNING: *The device is not protected against polarity reversal nor overvoltage! Polarity reversal may damage it, and cause serious injuries! There are electrolytic capacitors.*
WARNING: *When connecting the POWER terminal to a power supply, please use R or RL in series, there is a big capacitance in the electrolytic capacitors!*

Current transducer LTS 25-NP expects $5V$ power supply, current consumption shall not exceed several tenths of mA. It is powered through the signal connector. There is jumper to select pin to power LTS 25-NP (can select pin 1, which is common with ISO7241C, or pin 2), that is in case the ISO7241C shall be supplied with $3.3V$ (i.e: if it is operating with $3.3V$ micro-controller).

Input and Output signals

Output pins of the ISL Power Board:

- X5 - MOTOR terminal - provides power for the motor, voltage is given by input voltage on POWER terminal and by PWM duty cycle.

- X7 - 12V IN/OUT terminal - if SWITCH terminal is "ON", it can be used to drain stabilized 12V to power some small appliance.

- pin 8 on SV1: LTS 25-NP analog voltage signal: voltage proportional to the current going through the MOTOR terminal.

- pin 9 on SV1: ISL output of the input comparator / Not used, the resistor in the scheme should be left disconnected. (There has been an error in my original design.)

Input pins:

- X6 - POWER terminal - used to provide voltage for the motor. Works from 12 V up to 60 V safely.

- X8 - SWITCH terminal - if pins are connected together, ISL83204A is powered through 12V stabilizer from the POWER terminal, and X7 could be used to provide 12V for other components.[12]

- pin 3 - DIR on SV1: Direction of rotation of the motor (Logical low means one direction, high the other).

- pin 5 - PWM on SV1: Pulse Width Modulated signal is fed to HEN input on ISL83204A.

- pin 7 - DISABLE on SV1: Disable signal: If turned high, all FET Gates are pulled down.

- jumper SV2 - selects supply pin for LTS 25-NP, use position 1+2 to provide $5V$ from pin 1 on SV1, use position 3+2 to provide $5V$ from pin 2 on SV1. For details see 3.1.3

Cooler design and installation

There are two coolers to be considered: One for the L7812, and another one have to be in place for FETs.

[12] This should be taken into consideration, when designing cooler.

Pin #	Name	Description
1	ISO	Supply from controller side +3.3V or 5V
2	LEM	Supply +5V [13]
3	DIR	Direction for the motor
4	NC	Not connected
5	PWM	Pulse Width Modulation for the motor
6	NC	Not connected
7	DIS	If hi, ISL will disable shut down all FETs
8	LEM	Analog signal from LEM, 0.5-4.5 V see 3.1.2 for details
9	ISL VOUT	Should be left disconnected due to a mistake error in design
10	DGND	Digital and analog ground, for future designs should be separated

Table 3.2: SV1 Connector Description

Cooler for L7812 We can assume that the current driven from L7812 will be sum of current for ISL83204A $I_{ISL} = 50mA$), and current for ISO7241C $I_{ISO} = 28mA$. Maximal power dissipation is then:

$$P_{L7812} = (I_{ISL} + I_{ISO}) \cdot (U_{IN} - I_{OUT}) \tag{3.2}$$

In the extreme case, this would mean:

$$P_{L7812} = (50mA + 28mA) \cdot (35V - 12V) \leq 1.794W \tag{3.3}$$

Such power dissipation should be covered by the TO-220 package itself. To take some load of the voltage regulator, there is a 100Ω $3W$ power resistor placed in series. It is possible to replace it directly on board to change for a smaller value, or you can also add another one to the switch terminal.

Cooler for IRFB4115

To get total power dissipation on the IRFB4115, we have to consider losses on the transistor and as well on the catch diode. Catch diode forward voltage is $V_{SD} = 1.8V$. This gives us at the maximum rating:

$$P_{IRF-diode} = V_{SD} \cdot I_{IRF-max} = 1.8V \cdot 20A = 36W \tag{3.4}$$

This is of course actually extreme rating, as the current is not running through the catch diode all the time, and also it is not constant, but it is usually dropping exponentially. We can safely assume, that the dissipation on the catch diode would not be greater than $18W$.

Power losses from resistance of the IRFB4115 itself are determined by following equation:

$$W_{Rdson} = R_{DS(on)} \cdot I^2_{IRF-max} = 11m\Omega \cdot 20A^2 = 4.4W \tag{3.5}$$

Switching power losses are determined by following equations:

$$W_{off} \cong \frac{1}{4} U_D \cdot I_z \cdot t_{off} \tag{3.6}$$

$$W_{on} \cong \frac{1}{4} U_D \cdot I_z \cdot t_{on} \tag{3.7}$$

$$P_{switching} = f \cdot (W_{on} + W_{off}) \tag{3.8}$$

This gives us at $20kHz$, $36V$, and $20A$ switching times for IRFB4115 of $t_{d(on)} = 18ns$, $t_r = 73ns$, $t_{d(off)} = 41ns$, and $t_f = 39ns$:

$$P_{switching} = 20 \cdot 10^3 \cdot (\frac{1}{2} \cdot 36 \cdot 20 \cdot ((18 + 37 + 41 + 39) \cdot 10^{-9})) = 1.23W \tag{3.9}$$

Total dissipation on all FETs when running at full power (35V and 20A) in one direction and 50 % duty cycle would be sum of switching power losses on one FET (lets say on BHO), one catch diode of another one (on BLO), and looses from resistance (on ALO and BHO):

$$W_{total} = f \cdot (W_{on} + W_{off}) + 0.5 \cdot V_{SD} \cdot I_{IRF-\max} + (1 + 0.5) \cdot R_{DS_{on}} \cdot I^2_{IRF-\max} \tag{3.10}$$

$$W_{total} = 1.23W + 18W + 6.6W \cong 26W \tag{3.11}$$

WARNING: *All four FETs are to be electrically isolated from the cooler! Theoretically only Q1 and Q3 have the same potential on drain pin, which is common with the casing, but to stay safe it is highly recommended to isolate them as well!*

These ratings are exaggerated, and if we would use cooler designed to handle them, it would be quite large. We have used smaller but sufficient cooler, as the demands for power on Car4 vehicle are not that high. Used coolers have proven to be sufficient.

3.2 On-board control unit

3.2.1 Requirements - Specification

We have determined, from the requirements of the whole concept of the vehicle, and from technologies known to us at the time, that we are going to

need at least two control units: one for each axle, and possibly other one for communication and additional sensor array.

We have decided to use dsPIC from Microchip [11], mainly because there is a support package for MATLAB Real Time Workshop: Kerhuel Toolbox 6.1.2. This complies with requirement of rapid prototyping and development. Main requirements for the *Axle Control Unit* are:

- 2x Motor PWM (2 pins)
- 2x PWM for servo control (2 pins)
- 2x Analog to Digital Converter (2 pins)
- 1x UART (2 pins)
- 1x SPI (4 pins[14])
- 1x I2C (2 pins)
- 2x GPIO[15] for DIR[16] and DIS[17] (4 pins)
- 2x 2-channel QEI[18] (4 pins)
- 1x CAN (2 pins)

From these requirements that we have assembled over time, we have decided first for dsPIC33FJ128MC802 [12], however very soon we have switched to dsPIC33FJ128MC804 [13], which is the same controller, but in a different package: 802 is in a DIP package, and has total of 28 pins; 804 is in TQFP package, and has 44 pins.
We are not going to describe the microcontroller features here, as they are easily accessible in the devices datasheet.

3.2.2 ACU28 board description

Axle Control Unit 28 - ACU28 is equipped with the dsPIC33FJ128MC802 microcontroller. This unit has been used for initial testing and acquaintance with microcontroller usage and programming. It can be used to drive two DC motors via *ISL Boards* 3.1, and two model servo motors via *Servo interface*

[14] 4 pins are minimum for one device, for each device, there may be additional pin needed
[15] General Purpose Input Output
[16] Motor Direction
[17] Motor Disable
[18] Quadrature Encoder Input

3.3. Unit is composed from two double-layer PCB: Smaller one is a basic plug-in module designed to be connected to a suitable board, that can be designed to the requirements. On board of this PCB are:

- Microcontroller dsPIC33FJ128MC802
- Quartz crystal 20 MHz (can be changed)
- Programming and debugging port for ICD2 or ICD3. (Described in section 7.1)
- Reset button
- Optional power connector X1 (recommended input voltage range: $5 \sim 9V$)

Second PCB is adding additional external parts, that are needed for some peripherals to work, and also connectors to other boards and devices on board Car4.

- $5V$ positive voltage regulator 7805
- MAX3232 - RS232 transceiver
- SN65HVD232Q - CAN bus transceiver
- two ML10 connectors for *ISL Power Board* and *Servo interface*
- two PSH02-04PG connectors for motor encoders.

Only concerns in this design were in selecting pull-up resistors for encoders used on the motors, that are used in Car4 project. As there is not a sufficient info in datasheet, we had to determine suitable values experimentally: we have used $2k7\Omega$ resistors. Second concern was to make sure, that the voltage signal from LEM sensor on *ISL Power Board* 3.1, which can range from $0.5V$ to $4.5V$ is limited to $3.3V$ which is maximum for the dsPIC internal AD converter. We have decided to use a resistive divider. Upper resistor value has been chosen $1k2\Omega$ and lower $3k3\Omega$. This is based on following calculation: 3.12

$$U_{out} = U_{in} \cdot \frac{R_{low}}{R_{hi} + R_{low}} = 4.5 \cdot \frac{3k3}{1k2 + 3k3} = 3.3V \tag{3.12}$$

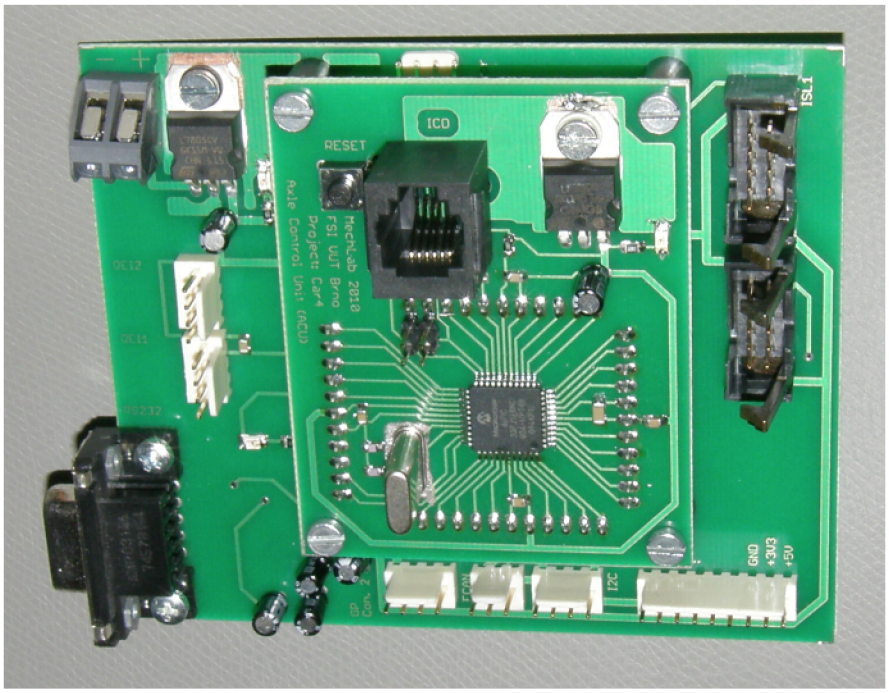

Figure 3.3: assembled ACU44 - top view

3.2.3 ACU44 board description

Axle Control Unit 44 - ACU44 is equipped with the dsPIC33FJ128MC804 microcontroller. Which is basically the same as dsPIC33FJ128MC802, that we have used in ACU28, just in a different package - with more pins.
ACU44 is also composed from two double-layer PCBs: Smaller one is designed as a plug-in module, which can be connected to a proper board. On board of this PCB are:

- Microcontroller dsPIC33FJ128MC804
- Quartz crystal 20 MHz (can be changed)
- Programming and debugging port for ICD2 or ICD3. (Described in section 7.1)
- Reset button
- 3.3V positive voltage regulator
- Two jumpers marked as SV3, they can be used to disconnect microcontroller from the board-to-board connector, when the programming / debugging port is needed.

Second PCB is, as it was with ACU28, adding just some peripheral functionality:

- Resistive divider for analog input - see description above: 3.2.2.
- $5V$ positive voltage regulator 7805
- MAX3232 - RS232 transceiver
- SN65HVD232Q - CAN bus transceiver
- Two ML10 connectors for *ISL Power Board* and *Servo interface*
- Two PSH02-04PG connectors for motor encoders.
- One PSH02-09PG connector for SPI communication, GND, $3.3V$, and $5V$ source pins.
- Two additional PSH02-04PG connectors for digital I/O, and/or ADC input.
- Configurable function LED. i.e.: Can be used to check that the program is running, or as a flag for overload.
- Power connector.

ACU44 needs a low DC voltage power supply, at least $5.5V$ and depending on connected peripherals and used crystal, around $100 \sim 150mA$. In theory, the maximal input voltage is $35V$, but it would lead to great power looses, as there is just a positive voltage regulator, which would transfer the additional power to heat. Acceptable power dissipation for TO-220 is around $1.5W$, so the input voltage should not exceed $15V$. Another supply recommendation is to use separate power supply for the controller from power electronics (ISL Power Board, Servo Interface Board power side) to prevent brown-out resets.[19]
From the experience gained in the development, application deployment and testing, we have found some space for improvements in design:

- Add power and signal jumpers for MAX3232 to be able to use directly the UART line.

[19]Microcontroller will reset the program, if supply voltage drops under certain threshold ($2.5V$)- this can happen when the batteries are running low and can not provide sufficient response for peak current drain

- Use different type of connectors for power supply connection: non-screwable / plug-in.
- Add more Analog inputs, use shielded cables.
- Better ground wiring, especially for the analog input.
- Add connectors for oscilloscope for debugging purposes. (Especially if the unit would be used for educational purposes.)

Although there are some imperfections, the ACU44 works well and meets or exceeds all initial requirements.

3.3 Servo interface

Servo Interface Unit was designed to provide isolated interface for model servos, that are used for steering the wheels. Insulation was a safety requirement given by the option, where the Car4 is operated via sensitive devices like are cards MF624 and dSCPACE. *Servo Interface Unit* is operating with high current: peak value for one servo is $3A$, there are two servos on each *Servo Interface Unit*.

3.3.1 Requirements - Specification

Requirements for the servo interface are:

- Input lines should be galvanically isolated from the output.
- Unit should have 2 servo Inputs / Outputs.
- Unit will be powered directly with $6V$.
- Unit should sustain peak current of $6A$.

3.3.2 Final product

We have designed *Servo Interface Unit* using ISO7220C [6], which is same principal as ISO7421C described in 3.1.2, but has two channels oriented in same direction, to insulate the input. Another component is positive voltage regulator 7805, few capacitors, and connectors. Unit is designed to be connected to *ACU28*, *ACU44*, or *MF624* via the *ISL2* connector, which is able to provide the PWM signal from OC[20] peripheral.

[20]Output Compare - *ACU28* and *ACU44*, *MF624* has also PWM output.

WARNING: *Board is not protected against short-circuit, reversed polarity of the power source, nor over-voltage!*
PCB was created as a one-sided, without any metalized vias, therefore we were able to make the final version in home-lab conditions using photo method.

3.4 Remote Control module

For gathering undistorted measurements the vehicle should have significant freedom of movement. Using long cable would be quite problematic and cable itself would influence the vehicle behaviour. Therefore we decided that *Remote Control Module* - RCM is needed.

3.4.1 Requirements - Specification

From project goals we derived following requirements for RCM:

- Should communicate on UART or RS232 protocol - we have UART on the microcontroller we are using 2.1.2 and mainly it is common port on PC. Other option would be something, that could be converted to USB,[21] but still have to be available on our microcontroller (SPI, I2C, CAN).
- Sufficient range - at least hundreds of meters.
- Resistant towards interference.
- Operate within licence-free band.
- Duplex communication.
- Sufficient bit-rate and latency. (determined by the amount of data we need to read - measure and sample rate, and also the drivers commands)

We have decided for an easy solution: final product - wireless UART transceiver from Höft & Wessel: HW 86010.[14] We have obtained two of these modules, one for the computer and one for the Car4 vehicle.

[21] Universal Serial Bus

3.4.2 Main components used in design

HW 86010

HW86010 is a *DECT*[22] based module, which offers few interesting functions for voice and data transfer. It contains several types of interfaces: UART, PCM[23], I^2C, digital and analog inputs and outputs. We needed just the UART interface, which provided us with 115.2 kBd[24] rate. We are using UART line to send commands to the Car4, and we are transmitting the telemetry back to the PC. With $200Hz$ sampling rate, we can theoretically transmit up to 72 Bytes in each direction in one iteration. $(115200/(200 * 8) = 72Bytes)$

MAX3232E / TRS3232E

MAX3232E (interchangeable with *TRS3232E* and other MAX 232 circuits) is $3V$ to $5.5V$ Multichannel RS-232 Line Driver. It is used to create interface between PC RS-232 line and UART peripheral on dsPIC or *HW86010*. It incorporates two drivers and two receivers, operates up to 250 kbps, and works with $3.3V$ and/or $5V$ power supply. As for external components, it needs one input blocking ceramic capacitor, and four electrolytic capacitors as charge pumps. Datasheet is available from [32]. Device is manufactured by MAXIM, Texas Instruments, and others.

3.5 Power Unit - ISL Boards

Power Unit - ISL Board was designed to provide $12V$ supply from up to $36V$ source. It is switching power supply, based on TPS5450 [16] step-down swift^{TM}converter from Texas Instruments. It can deliver current of up to $5A$. That is more than enough to power all four of *ISL Power Boards* 3.1 logic control sides. According to *SwitcherPro Desktop* 6.4 the efficiency should be at worst around 75%, and around 90% at best.[25]

3.5.1 Requirements - Specification

Requirements are given by the project specification:

[22] Digital Enhanced Cordless Telecommunications

[23] Pulse-code modulation

[24] kilo Baud - rate of 1kBd means 1024 bit per second.

[25] Efficiency is dependant on the input voltage and load: lower input voltage and higher load is better.

- Input voltage range: $24 \sim 36V$.
- Output voltage: given by ISL83204A (described in: 3.1.2) $9.5 \sim 15V$.
- Current output: should not exceed $600mA$.
- Efficient operation.

3.5.2 ISL PU board description

The design of the board schematics was completely done using the *Switcher-Pro Desktop* software, which also gives some suggestions about the layout. We have used suggested design, redrawn the schematics into *Eagle*, and then designed corresponding PCB layout.

Power Unit - ISL Board can be also used to provide power for other on-board $12V$-supplied electronic.

Chapter 4

Development of On-Board basic firmware for ACU44

In this chapter, we are describing the basic firmware (FW) implementation used in ACU44. This firmware is designed to give control to the advanced algorithms used for vehicle stability, or just simple movement commands. It is also used for sensor data acquisition and communication with driver and other on-board units.

4.1 Overview of Kerhuel toolbox blockset

Kerhuel toolbox, which is also described in section 6.1.2 provides us with control over most of the dsPIC peripherals. It can be divided into a few categories:

PIC Configuration *Master block*, and for controllers with remappable peripherals *Peripheral PIN MAPPING* block.

Simulink Configuration Block that is used to set all paths and compiler options. Configures several options under the *Configuration Parameters* dialogue.

Communication *SPI Port*, *BUS I²C*, *BUS CAN*, *Serial PORT - UART*

I/O Functions *Digital I/O functions* read and write on selected pin, *Peripheral I/O Functions* Input Capture, Output Compare, Motor PWM, ADC,...

Others C function call, SW reset, Chronograph, Nop,...

Beside Kerhuel toolbox, we have also used some custom C-code which has already been created in the community around Kerhuels toolbox. First we needed to add QEI handling, which was not implemented in the toolbox, we had to modify the code (which was already created) for our controller and add second channel handling. Another code we are using is for the ECAN communication: Kerhuels toolbox implements only CAN standard, but we are using controllers with ECAN (CAN 2.0A / CAN 2.0B). This code was created by Bryant Mairs: [34]. In the version we have used was a small bug which gave us some problems from time to time. In the version available from May 2010 this bug was removed. For future work, this version should be used.

4.2 Communication implementation

Fast and reliable communication is crucial for the function of the Car4 vehicle. We had to implement remote control communication which would also send vehicle telemetry back to the operator. Another communication channel has to be established on board between both ACU44 and also with MPC555.

4.2.1 UART Data Transfer - Drivers commands and telemetry

Physical layer implementation has been described in section 3.4. As for the FW part: We have designed a packet for vehicle operator commands. UART implementation is able to send one Byte (8-bits) at the time, this would not give us enough resolution for operators commands. Therefore we have designed following sequence:

1. Start Byte: 255
2. Lower 8 bits of the first 12-bit number
3. 4 Upper bits of the first, 4 Lower bits of second 12-bit number
4. 8 Upper bits of the second number
5. CRC[1] from the Bytes 2, 3, and 4.[2]
6. Lower 8 bits of the third 12-bit number

[1] Cyclic Redundancy Check
[2] CRC is implemented as Bitwise XOR.

7. 4 Upper bits of the third, 4 Lower bits of fourth 12-bit number
8. 8 Upper bits of the fourth number
9. CRC from the Bytes 6, 7, and 8.

These numbers can basically represent any data, we are using them as:

1. Motor momentum
2. Front axle steering angle
3. Rear axle steering angle
4. twelve buttons

PC side implementation is easy, there is no problem sending more bytes in one time step using MATLAB or any other software (We have used simple Delphi application for initial testing and debugging.) We did not manage to receive more bytes in one time step on the ACU44 using the basic tools in Kerhuel toolbox, but this can be done using custom C code. But for our purposes it was sufficient to receive one Byte in each time step. We run the UART Rx handle loop at 200 Hz. We are using 9 Byte long packet, this give us rate of 22.2 packets per second for sending operators commands. In each iteration, we have shifted the bytes in our internal buffer, and checked for the valid packet. Valid packet means, that the first Byte in the buffer is equal to Start Byte (255), and that both CRC positions match the proper values. When the valid packet is received, re-assembled data are saved into memory as new values. Demonstration files on how this works are enclosed, see: B.3.

4.2.2 ECAN protocol - on-board communication

For the communication between the control units, we needed something more reliable, faster and expandable than UART, also ECAN implementation was one of the initial requirements for this work. ECAN protocol is Enhanced version of the CAN protocol. ECAN is also referred to as a CAN 2.0A and CAN 2.0B. Great description can be found on the web pages of the CAN creator: Robert Bosch GmbH: [36]. Another possibly even better description is to be found on Kvasers web page: [35]. We have used implementation designed to work with Kerhuel toolbox created by Bryant Mairs: [34]. As we have a quite simple on-board network, we are using the 2.0A version of protocol, which is basically the CAN implementation. ECAN allows us to

send all data we need to send in each time step (our fastest loop is running at $200Hz$.) Only concern is with proper labelling the data, as the label of the data decides the priority on the bus. We have assigned higher priority to the command codes and global disable code, and lower priority to the sensory data. This could be of course easily modified by changing the priority which is determined by an identifier. For the older CAN and CAN 2.0A, there are 2032 (or 2048 - for most CAN controllers, but the 2032 through 2047 are considered illegal. [35]) possible labels, for the 2.0B there are 532676608 possible labels.
ECAN is designed to be extremely robust and reliable and that we can confirm. Once we got it working, it worked greatly. Only problems were with the implementation due to some compilation difficulties and few bugs in the tools we have used, but now, these should be resolved and for further implementations they should work well.
ECAN implementation on MPC555 was done by Vojtěch Lamberský, and it is described in his thesis: [39]. As the command structure has changed for this variant (2x ACU44 + MPC555) we had to change the identifiers and also add some more data on the bus.

4.3 Sensor data acquisition

On board the Car4 vehicle, there are various sensors:

1. Motor encoders
2. ADIS - for further details see 2.2.1
3. Complementary sensory system created by Matěj Šimurda - for further details see 2.2.1 and [38].

We had to implement the Motor Encoder readouts which is done by a dedicated peripheral of the dsPIC. In the program, we can read out the position of the motor, however, this is implemented as a 16-bit integer number which increases or decreases by one on each edge[3]. We can use derivation to compute the angular speed of the motor, but there are two problems:

Counter overflow this can be easily solved by using simple condition: if the derivation is higher than certain threshold, ignore the value and take the last one. Threshold can be determined from the maximal revs of the motor: At $36V$ we would get about 11400 tics per second. As

[3]Depends on the peripheral settings.

we are using much faster sampling time, we can decrease this value accordingly. At sample time $Ts = 0.05s$ it would be 570 tics. So the threshold could be set to 700. The overflow of 16-bit counter would generate number much higher[4].

Oscillations around one value especially at low speeds and when using fast sampling time we are getting just a few tics at each iteration. This can cause problems with the derivations, therefore we have to implement some sort of filtration. Simple but functional is to use the mean of few last values: We have used the mean of two to four last values, and it worked quite well. More advanced filtering methods are described by Vojtěch Lamberský in his thesis: [39].

4.4 PWM Motor Control

We are using PWM to control all motors on board the Car4 vehicle. There are two DC motors and two model servo motors on each axle. Each axle has its own Axle Control Unit, these are described in section: [**?**]. PWM for the DC motors is generated using the motor PWM dedicated peripheral of the dsPIC, PWM for the model servos is generated using OC (Output Compare) module, which is more accurate than the motor PWM option.

4.4.1 Motor Control - Driving ISL Power Board

We are using just one output of each PWM channel[5]. Direction of the motor is set using the DIR (direction) pin. As the internal implementation is done as signed integer, and the PWM module accepts only unsigned value, we are using this to determine the direction: positive values are one, negative the other. Absolute value is used to determine the duty cycle.
We are running the PWM on $10kHz$. We have experienced problem when both ACU44 have been set to the same frequency, there was a certain disturbance, which caused the motors to "shake". We have eliminated this behaviour by setting different frequencies for both ACU44:

Master unit is running at $10kHz$

Slave unit is running at $8kHz$

[4] Close to 2^{16}.
[5] Complementary output is not used due to our use of the ISL power board.

Figure 4.1 shows the voltage on motor terminals of the ISL Power Board running at 75% of duty cycle. The smaller signal is the PWM output after being passed through ISO7421C 3.1.2 from dsPIC. Figure 4.2 shows the same for duty cycle of 50%, there is nice display of the transitional process.

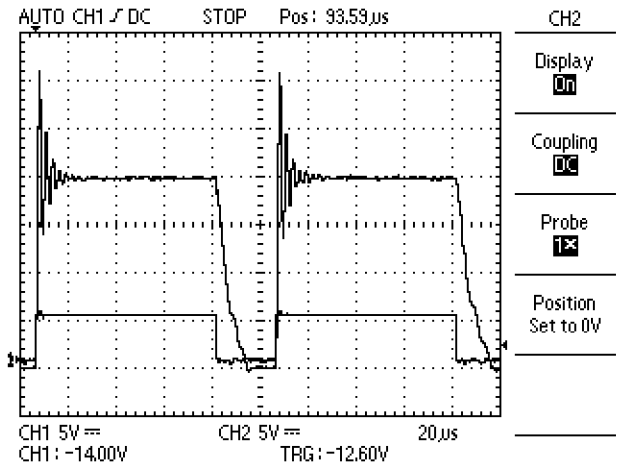

Figure 4.1: Voltage on motor out terminals of the ISL Power Board running at 75% of duty cycle.

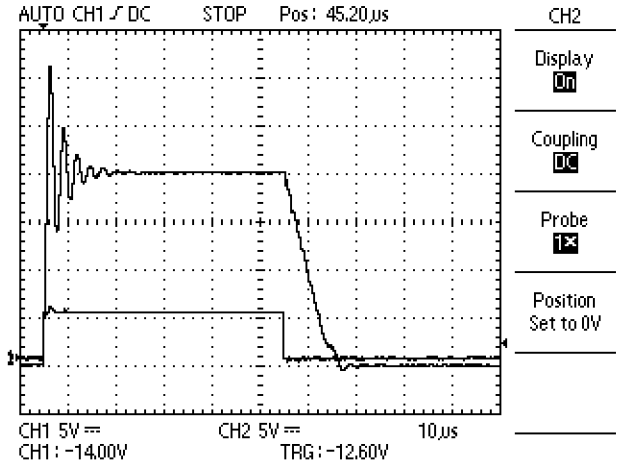

Figure 4.2: Voltage on motor out terminals of the ISL Power Board running at 50% of duty cycle.

4.4.2 Motor Control - Driving Servos

We are using model servos [28] which already have their own control electronics including some position regulator. Position is set by certain duty cycle of PWM signal with $50Hz$ period. We have used OC (Output Compare) peripheral to generate this signal. We had to set the zero steering angle value and both boundary values for wheels experimentally, as each servo had a bit different zero position. This was quite laborious as we had to do the calibration quite often as the servos are quite fragile and sensitive to careful manipulation and handling. This is a slight issue as these components are under significant stress.

4.5 Conclusion

We have quite successfully assembled and tested the basic firmware for the on-board axle control units (ACU44), including the communication with operator and prepared the interface for the advanced control algorithms implementation. All source files and models are enclosed in this thesis as an annexe on a CD, please refer to annexe section: B.3.

Chapter 5

Product testing and HIL

5.1 Development Testing

As a lot of the practical work was new to us, we had to go by small steps towards our goal - a working electronics for Car4 platform. We have started with ISL Power Board development 3.1, which was actually quite simple but with practically no experience with electronics design it took a lot of time. At the beginning, we have used to test the circuitry only digital logic signal (0V from GND and 5V from the power supply), and without signal generator, it was quite a problem to simulate a PWM, later we have generated PWM with Atmega8 microcontroller.

For the testing of ACU44 we had already at our disposal full Car4 mechanical construction with motors, servos, and finished ISL Power Board. Testing and debugging is done very easily with Kerhuel Toolbox using the rs232gui. This GUI is used to run code on data received over RS232 line, usually to plot them. In example: we have used this to display the PID components, sensor data, or other computational data. To send data from the controller, there is a block for multiplexed Tx Output, which accepts 16-bit unsigned integer.

We have also created a small application in Delphi, for sending packets via RS232 line, mainly to set motor speed or momentum, and steering angles. Later requirements leaded to creation of a control application in MATLAB - Simulink, created by Vojtěch Lamberský, which was easier and faster to use as we needed to read and to write, and from time to time also reconfigure the structure of the packets. In further plans, we are considering using LabVIEW from National Instruments, which we have found to be suited for this

type of application.

Another set of tools used in the development and testing phase was just the usual laboratory equipment such as DC power supply, oscilloscope and multimeter. We did not have any special requirements or needs to use something else, although logic analyser would have come handy some times.

5.2 HIL - Hardware In the Loop

HIL - Hardware In the Loop is method used for testing, validation and verification of the Firm-Ware code implemented in controllers (or processor) in a special way, based on emulation of the hardware peripherals connected to the controller. Common setup is that the tested hardware (usually a controller) is connected to the software model[1] of the rest of the system. Interface between the software and hardware is usually realized by various Input and Output devices such as *ADC*[2], *DAC*[3], Digital Input and Output ports. *HIL* simulation is broadly used in aerospace, automotive, robotics and power electronic applications during the product development phase. Main motivation to use *HIL*, according to [24] is:

- Cost - cost of all tools and effort, cost of prototyping
- Duration - tight development schedules, high-burden-rate plant, early process human factors development.
- Safety - i.e.: Flight-by-wire, jet engine control, ...
- Feasibility - i.e.: testing of electronics systems for renewable power sources: i.e.: variable speed wind turbines, power electronics inverters for photovoltaic power systems.
- Education - simulators: Nuclear Power Plant operator, flight simulators[4],...
- Repeatability - tests can be run under exactly the same conditions.

Another motivation is to use *HIL* to test in conditions, that are usually hard to achieve, but may set in. Also long time reliability and stability can be tested.

[1] Also referred as "plant simulation"
[2] Analog to Digital Converter
[3] Digital to Analog Converter
[4] These are not usually considered as HIL techniques, but the principals are usually the same.

5.2.1 Motivation

Our motivation to use *HIL* techniques is obviously not really one of the above stated. Although we have gotten close to *HIL*, when we worked on various algorithms at home without the actual Car4 vehicle, but just with ACU44. So our main motivation was to learn something about *HIL* methods, that are being widely used in automotive industry as we have mentioned before, we are trying to get familiar with and close to them.

5.2.2 Realization

As we had a little time left for this part of the project, we did use the technique just on a simple problem: We have decided to test ACU44 with simple firmware implementing serial communication for commands and PID speed regulator.

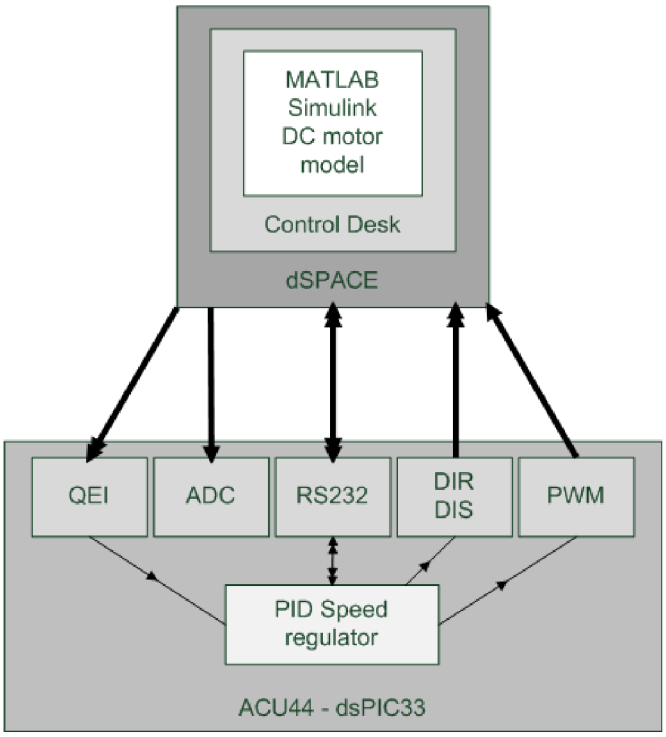

Figure 5.1: Schematics of the software and wiring connection

Problem definition

We have defined following tasks for the ACU44 testing with *HIL* techniques:

1. Test what happens when one encoder wire is disconnected.
2. Test the PID speed controller while changing motor power source voltage.
3. Test the PID speed controller while changing load on the shaft.
4. Test the ACU44 for the serial communication "overload".

Tools and resources requirements

From the nature of the problem, we obviously needed suitable hardware to run the simulation on in real time, and with corresponding interface (analog output, digital input and output, PWM capture input.) In MechLab, we have at our disposal dSPACE DS1103[25] which meets our requirements on hardware.
We also needed to create connection between dSPACE terminal block and ACU44, this was quite simple as dSPACE provides quite good description of the pinout. We required ADC output to generate voltage signal simulation current readout from LEM (3.1.2), digital inputs for DIR and DIS (3.1.3), digital outputs for Encoder phases, and PWM capture input to measure duty and frequency of the PWM. Schematics is shown on figure: 5.1. Another requirement to do *HIL* simulation is to have the model of the plant as good as possible.

Main problem is to create a model of the plant which is as accurate as possible, but still suitable to run in real time on our target hardware. We needed to emulate a DC motor, which is not really demanding on sample time, as the time constants of our motor are:
Mechanical time constant:

$$\tau_{mechanical} = \frac{R \cdot J_T}{K_T \cdot K_{Phi}} = \frac{1.6 \cdot 150.86e^{-6}}{78.127e^{-3} \cdot 0.03} = 0.103s \tag{5.1}$$

Electrical time constant:

$$\tau_{electrical} = \frac{L}{R} = \frac{132.58e^{-3}}{1.6} = 8.3e^{-3}s \tag{5.2}$$

Where R is motor winding resistance (Ω), L is motor winding inductance (H), J_T is total inertia on the motor shaft ($N \cdot m \cdot s^2$), K_T is motor torque constant ($N \cdot m \cdot A^{-1}$) and K_{Phi} is motor voltage constant ($V \cdot s \cdot rad^{-1}$). We have used values from estimation done by colleague Michal Jasanský. Time step is usually selected as 1/10 of the fastest time constant which is in our case the electrical. We have decided to run the simulation on a discrete time model, as there were problems with real time on the continuous model. Sample time for the motor was selected to $1e^{-3}s$, which is not exactly 1/10 of $\tau_{electrical}$ but we are limited to use integer multiples of the fastest sample time.
Most demanding on the sample time was generating signal for *QEI*. This is determined from the number of edges at the maximal motor speed. At no-load speed, the motor gives us about 130 rps[5], which generate in total 9880 edges[6]. According to Nyquist theorem, we should sample at least twice as fast, in this case, we have to generate the pulses using two digital outputs. We have used $5e^{-5}s$ for the encoder signal generation, which is almost boundary value to safely run the simulation. As we have mentioned, we have used discrete model of the DC motor. We have created state space model and then discretized it using zero-order hold method in MATLAB.

Experiment realization

In 5.2.2 we have defined tasks for the *HIL*. The results were:

Encoder channel malfunction PID controller fails to operate, as would be expected: signal on one channel being interpreted as if the motor is oscillating around one position with the algorithms and filters we are using to determine speed, such situation would be interpreted as zero or very slow speed with changing direction, leading to saturation of the I component of the PID controller,[7] and consequently full speed of the motor. We did not treat for this problem, as we do not expect the cable assembly to fail in this way. Also it would be probably impossible with used hardware to do anything else than immediate stop of the vehicle.

Motor power supply voltage changes, motor load changes As we are using simple PID controller, it can compensate for both. It is only limited by the motor power and of course, with voltage drop the maximal reachable speed drops as well.

[5] Revolutions per second.
[6] Encoder generates 38 edges per revolution per channel.
[7] If the requested speed is non-zero.

Serial communication overload We are using UART (RS232 line) to send a packet which includes desired speed. When valid packet is received, new speed requirement is given on the input of the PID regulator. When we have overloaded the line, it lead to invalid packets and / or big latency[8]. Communication speed (baud rate) has to be set carefully, to achieve error-free operation.

5.2.3 Conclusion

We have tested several scenarios, most challenging was the application set-up to achieve real-time simulation. We have encountered few problems with stability of the PWM frequency, but it is more likely, that this was caused by the dSPACE, then dsPIC - we have changed the PWM frequency from $10kHz$ down to $5kHz$, and even then it occasionally gave a readout of $2.5kHz$ or even $0Hz$. We did test only on a simple task, which is usually not the point of *HIL* simulation, but our motivation was to learn the basics of the *HIL* approach and techniques, and this we did quite successfully.

[8] Given by the buffer size.

Chapter 6

Software tools used in development

In this section, we will shortly introduce available software tools for rapid prototyping, as well as other tools used when working on this project.

6.1 Matlab

We expect that to the reader of this thesis is *MATLAB* known tool, therefore we are not going to describe it here. Reason, why we are mentioning *MATLAB* is, that we are using several toolboxes for *MATLAB*, that the reader may not be familiar with.
Version used: 2009a

6.1.1 Real Time Toolbox

Real Time Toolbox (RTT) is tool allowing *MATLAB - Simulink* to communicate with the real world via additional hardware for data acquisition, signal processing, and various digital and analog outputs. More information, about the RTT can be found on Humusoft webpage: [17]. RTT is useful forgiving C code representation of various Simulink blocks, that can be further compiled for our target microcontroller, and its suitability for Real time code.

What does Real-time mean? At this point we would like to explain the meaning of the therm "Real-time": common mistaken view is, that real time means quick. Accurate meaning is "in time": RT system should ensure, that the responses occur in time, or on time. More information about RT terminology and principals can be found here: [33].

6.1.2 Kerhuel toolbox

Kerhuel toolbox is embedded toolbox for *MATLAB - Simulink*, that allows code generation for various microcontrollers from Microchip.[1] Together with *Real Time Toolbox*, it produces C code, that is ready to be compiled for target microcontroller. It allows you, to set the microcontroller, and use it's peripherals directly via Simulink blocks.
Blockset is still under development, so there are some minor bugs, and some functions, that could be added. But it features most of the functions we needed in our project.
Installation note: It is better to have MPLAB 6.2, and C30 compiler, installed before running the *Kerhuel toolbox* instalation script. And it is also essential, that you install the toolbox in directory, which is different from your *MATLAB* path.
Version used: 3.2b

6.1.3 Target support package FM5

Target support package FM5 is product similar to *Kerhuel toolbox*, described in 6.1.2. This toolbox is made to work with processors from MPC5xx family by Freescale.[20] We have at our disposal *phyCORE®- MPC555 Rapid Development Kit* which uses MPC555 processor. This kit should be able to take place of both ACU44 units as it has enough peripherals on its own, furthermore, it has computing power, including FPU.[2] We did not use this kit to directly control the Car4 vehicle, as our ambition was to use low cost solution (dsPIC microcontrollers from Microchip cost around 3 to 6 USD, where the whole kit from phyCORE starts from 539 USD.) We have worked on option, where the MPC555 is used as the main computing unit, getting informations through CAN bus to and from ACU44 units.

6.2 MPLAB IDE

MPLAB Integrated Development Environment (IDE) [21] is a development set of tools for Microchips PIC^{TM}and dsPIC^{TM}microcontrollers. We have used it mainly to compile generated C code, and to download the program into the dsPIC^{TM}microcontroller on board ACU28 and ACU44. It can be also used for debugging, we have used this option as well.

[1] There is also toolbox developed directly by Microchip, but it is (spring of 2010) still a bit far from being finished, and usable for our purposes.
[2] Floating Point Unit

To download the code generated by using Kerhuel Toolbox 6.1.2, following steps need to be done:

1. Generate code using Kerhuel toolbox for MATLAB - click *Incremental build*. This will create C-code and then a hex file.
2. Open MPLAB
3. Configure for proper device: Configure → Select Device: Device (i.e.: dsPIC33FJ128MC804)
4. Connect ICD2 to your PC.
5. Select Programmer: Programmer → Select Programmer → MPLAB ICD2.
6. Load HEX file: File → Import...
7. Program: Programmer → Program
8. To run your program disconnect ICD2 or press the *Release from Reset* button.

To debug the program (or compile it on you own), generated C-code needs to be loaded into the MPLAB as a project. To do this, follow these steps:

1. Follow steps 1-3 from the previous list.
2. Create new project: Project → Project Wizard; click Next; select proper device (i.e.: dsPIC33FJ128MC804); click Next.
3. Select a language tool-suite: Active Toolsuite drop-down select: Microchip C30 Toolsuite; as a Toolsuite Contents: MPLAB C30 C Compiler (pic30-gcc.exe); click Next; select Path for your project; click Next twice, then Finish.
4. Add all Source and Header Files into *Files* tree of your project: You have to add all .c and .h files generated by Kerhuel toolbox into your MPLAB project.
5. Add linker Script: this is .gld file, typical location would be: *c:\Program Files\Microchip\MPLAB C30\support\dsPIC33F\gld\p33FJ128MC804.gld*
6. Select Debug / Release option for your build based on what you want do do.

7. Build project: Project → Build all

8. Select Debugger / Programmer via Debugger / Programmer menu

9. Run (Debugger menu) / Program (Programmer menu) your microcontroller.

We have come across several events, that we had to compile the code using the above described method, because the code produced by compilation called by Kerhuel Toolbox did not work properly[3]. This was the at least the case using the CAN bus. We have consulted this with Lubin Kerhuel on his forum [19] designated for the blockset support, but did not reach (as of April 2010) the solution yet. Some problems are also probably with the MPLAB compilation, but this is at the moment beyond our level of understanding.
Version used: 8.36

6.3 Eagle

For all electronic schematics and PCB[4] layout designs, we have used Eagle 4.16 from CadSoft. For beginners, we recommend [37]. For the board production, we have used several options:

- Prototyping in home conditions: ironing and later the photographic method, both for one sided and two sided PCBs.
- APAMA [22]
- PragoBoard [23]

Making PCBs at home (or laboratory, but without professional tools) is good for prototyping and debugging purposes, especially for beginners. Most of the circuits could be also tested on solderless breadboard. We have decided to make our PCBs directly, as we have often used parts, that would be problem to put into solderless breadboard, such as SMDs, or high power circuits (LEM, IRFB4115, ...)
As for the professionals: APAMA is a small company based in Brno, we had previously good experience with them. Unfortunately they did not keep declared delivery time.
Pragoboard offers special service, that is suitable for small orders: Pool Service. This service is designed to produce cheaply low-number of pieces of

[3] Exact cause have not been determined yet.
[4] Printed Circuit Board

PCBs. We have used their services only once for the Car4 project: Matěj Šimurda [38] had some of his PCBs made there; quality was good, and delivery time was as declared - one week.
Version used: 4.16

6.4 SwitcherPro Desktop

SwitcherPro™Desktop is Switching Power Supply Design Tool from Texas Instruments. We have used this tool to design a switching $12V$ power supply 3.5 for *ISL Power Board* 3.1 control side, and to design $6V$ power supply for *servo interface*. This program can design switching power supply using TPS40K™controllers from Texas Instruments, and it also recommends parts, that are needed from other manufacturers. We have used the designed schematics to draw schematics in Eagle and then we have designed PCB layout accordingly.
Version used: 3.6.0.26406

Chapter 7

Hardware tools used in development

In this section, we will briefly describe tools, that we have used in the project.

7.1 Microchip ICD2

We have used *ICD2* - In-Circuit Debugger from Microchip [30] to program and debug code on dsPIC microcontrollers from Microchip.[1] ICD2 works with MPLAB, usage is described in section 6.2.
There are no special requirements for this hardware. Only requirement is to have the programming / debugging terminals available on the target microcontroller. On some microcontrollers, there are more combinations. We have used default set-up on ACU44. Connector can be realized with 6-wire "phone" cable, or any other 6-wire connector. We have used the phone cable, as it is default connector used by Microchip, it is practical, cheap and quite reliable.

7.2 UV Nail Lamp

For prototyping new PCBs, we have used the photo method. For exposure, we have used modified UV Nail Lamp - we have rearranged the lamps inside the device to get as regular exposure as possible. Also a small stand has been made to raise the distance between PCB and lamps. To create the positive image, we have printed the motive on to a foil for laser printers. For double

[1] Actually a non-original version - clone, was acquired for a fraction of the original price, for authors personal use.

layer PCBs we have pieced up both motives and then fixed the PCB between them using two glass desks.
Exposure time, using this approach is around 50 seconds. We have successfully made paths with $1.27mm$ pitch and $0.61mm$ width. Also with enough care, two-sided PCBs are easily produced. Metallized vias for through-hole mounted parts can be achieved using cable tubes, but easier way is to take this into account when designing the layout and not to use vias for through-hole parts.
Inspiration for this approach was drawn from [31]

Chapter 8

Conclusion

During last 14 months, we have as a team developed and created a vehicle, that meets more or less all initial requirements. As an individual I have gained quite a lot of experience in circuit design, rapid prototyping and development methods, some programming experience and also some teamwork based experience. I have found this project as a very challenging and contributive for my education process. Hopefully, I have met all the tasks initially given for this thesis, and also in addition done some other tasks needed for the successful completion of the Car4 vehicle.

In this thesis I have described my work done in past 14 months, which included all initial requirements specified for this thesis and also some additional tasks that came up during the project[1].
I have designed the on-board control unit network based on the requirements given in the project specification. I have also designed on-board circuitry including *axle control unit* for control of the vehicle motion, *ISL Power board* to drive the motors, remote control power and interconnection module, and servo interconnection module. In addition I have created one switching power supply for more economical power distribution. (Another design was unfortunately unsuccessful so far.)
Another task I have successfully carried out was the design, assembly and testing of the basic firmware implementation, which makes the Car4 ready for use of advanced control algorithm's.
As a last task, I have used HIL techniques to test the ACU44. At this point, there was not much to test, but the main goal was to explore and get familiar with the techniques used these days. This is a very complex area, so I had had just started to comprehend it's full potential. However, some HIL

[1] Some of the additional tasks are not even described here.

techniques were tested and used successfully.

Main outcome of my work is my part on the successful project, the large quantum of knowledges from different areas gained and shared among the team, and also the electronics that can be also used for other projects supervised under the MechLab.

Bibliography

[1] MechLab - Mechatronics Laboratory website:, http://www.umt.fme.vutbr.cz/mechlab, 20.11.2009

[2] ISL83204A - 60V/2.5A Peak, High Frequency Full Bridge FET Driver:, http://www.intersil.com/products/deviceinfo.asp?pn=ISL83204A, 20.11.2009

[3] Farnell - Electronic Components Distributor and Supplier:, http://www.farnell.com/, 21.11.2009

[4] IRFB4115 - HEXFET®Power MOSFET:, https://ec.irf.com/v6/en/US/adirect/ir?cmd=catProductDetailFrame&productID=IRFB4115PBF, 21.11.2009

[5] ISO7241C - high speed quad digital isolator from Texas Instruments:, http://focus.ti.com/docs/prod/folders/print/iso7241c.html, 21.11.2009

[6] ISO7220C - high speed dual digital isolator from Texas Instruments:, http://focus.ti.com/docs/prod/folders/print/iso7220c.html, 24.04.2010

[7] LEM - current transducer, voltage transducer, sensor, power measurement:, http://www.lem.com/, 22.11.2009

[8] LTS 25-NP data-sheet:, http://www.lem.com/docs/products/lts25-npe.pdf, 22.11.2009

[9] L7812 data-sheet:, http://www.st.com/stonline/products/literature/ds/2144/l7812ab.pdf, 22.11.2009

[10] 78L05 data-sheet:, http://www.st.com/stonline/products/literature/ds/2145/l78l05ab.pdf, 22.11.2009

[11] Microchip web page:, http://www.microchip.com/, 19.04.2010

[12] dsPIC33FJ128MC802 product info:, http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en532302, 21.04.2010

[13] dsPIC33FJ128MC804 product info:, http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en532303, 21.04.2010

[14] Höft & Wessel webpage:, http://www.hoeft-wessel.com/de/home.htm, 23.04.2010

[15] Höft & Wessel webpage: HW86010 module:, http://www.hoeft-wessel.com/en/products/HW86010.htm, 23.04.2010

[16] Texas Instruments webpage: TPS5450 module:, http://focus.ti.com/docs/prod/folders/print/tps5450.html, 24.04.2010

[17] Humusoft webpage: Real Time Toolbox:, http://www.humusoft.cz/produkty/rtt/, 25.04.2010

[18] Lubin Kerhuel webpage: about Kerhuels Toolbox:, http://www.kerhuel.eu/wiki/Simulink_-_Embedded_Target_for_PIC, 25.04.2010

[19] Lubin Kerhuel forum: Toolbox support:, http://www.kerhuel.eu/forum/index.php, 25.04.2010

[20] Freescale webpage: MPC555 Product Summary Page:, http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MPC555, 25.04.2010

[21] Microchip webpage: MPLAB IDE:, http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002, 25.04.2010

[22] Apama Brno webpage:, http://apama.cz/, 25.04.2010

[23] Pragoboard s.r.o. webpage:, http://www.pragoboard.cz/, 25.04.2010

[24] Hardware-in-the-loop simulation - Wikipedia, the free encyclopedia:, http://en.wikipedia.org/wiki/Hardware-in-the-loop_simulation, 09.05.2010

[25] dSPACE International:, http://www.dspaceinc.com/ww/en/inc/home.cfm, 10.05.2010

[26] MPC555 - PhyCORE rapid development kit:, http://www.phytec.com/products/rdk/PowerPC/phyCORE-MPC555.html, 15.05.2010

[27] Transmotec - PD42 series DC motors:, http://www.transmotec.com/dc-motors/planetary-gear/PD42-Series.aspx, 16.05.2010

[28] Scanner RC - SSV-9960MG High Performance Large Servo:, http://www.scanner-rc.com/specification/SSV-9960MG.pdf, 16.05.2010

[29] ADIS 16405 - High Precision Tri-Axis Gyroscope, Accelerometer, Magnetometer:, http://www.analog.com/en/sensors/inertial-sensors/adis16405/products/product.html, 16.05.2010

[30] ICD2 - In-Circuit Debugger, Microchip:, http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en010046, 19.05.2010

[31] fotocesta: domácí výroba desek plošných spojů (DPS):, http://www.elweb.cz/clanky.php?clanek=101, 19.05.2010

[32] MAX3232 - RS-232 line driver and receiver:, http://focus.ti.com/lit/ds/symlink/max3232e.pdf, 20.05.2010

[33] What is a Real-Time Operating System (RTOS) - Developer Zone - National Instruments:, http://zone.ni.com/devzone/cda/tut/p/id/3938, 24.05.2010

[34] ECAN driver for the dsPIC by Bryant Mairs:, http://overtux.cse.ucsc.edu/gitweb/?p=malife/ecan.git;a=summary, 24.05.2010

[35] Kvaser - advanced CAN solutions: The CAN Bus - Introduction:, http://www.kvaser.com/can/intro/index.htm, 24.05.2010

[36] Robert Bosch GmbH - What is CAN?:, http://www.semiconductors.bosch.de/en/20/can/1-about.asp, 24.05.2010

[37] Ing. Jiří Mišurec, CSc., Ing. Václav Zeman, Ing. Miroslav Štěpén: *Konstrukce elektronických zařízení - návrh plošných spojů*, Fakulta elektrotechniky a komunikačních technologií VUT v Brně, 2002

[38] Šimurda M.: *Návrh a realizace doplňkového sensorického systému pro experimentální vozidlo.*, Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2010

[39] Bc. Lamberský V.: *Vývoj algoritmů pro odhad stavu experimentálního vozidla.*, Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2010

[40] Bc. Jasanský M.: *Návrh dynamických modelů pro řízení trakce experimentálního vozidla.*, Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2010

[41] Valdejch F.: *Design of experimental vehicle undercarriage with four wheel steering.*, Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2010

Appendix A

Used abbreviations

4WD 4 Wheel Drive

ABS Anti-lock Braking System

ACUxx Axle Control Unit (28-pin or 44-pin version)

ADC Analog to Digital Converter

AWD All Wheel Drive

AWS All Wheel Steered

BLDC Brush-Less Direct Current (motor)

BOR Brown-Out Reset

CAN Controller Area Network

CD Compact Disk

CRC Cyclic Redundancy Check

DAC Digital to Analog Converter

DAQ Data Acquisition

DC Direct Current

DECT Digital Enhanced Cordless Telecommunications

DIP Dual In-line package

DIR DIRection

DIS DISable

DMA Direct Memory Access

DSP Digital Signal Processor

DSTC Dynamic Stability and Traction Control

ECAN Enhanced Controller Area Network (revision 2.0A and 2.0B)

ESC Electronic Stability Control

FPU Floating Point Unit

FW Firm-Ware

GPIO General Purpose Input/Output

GUI Graphical User Interface

HIL Hardware In the Loop

HW Hard-Ware

I^2C Inter-Integrated Circuit

IC Integrated Circuit

LED Light Emitting Diode

OC Output Compare

PCB Printed Circuit Board

PCM Pulse-Code Modulation

PLL Phase-Locked Loop

PWM Pulse Width Modulation

QEI Quadrature Encoder Interface

RT Real Time

RTOS Real Time Operating System

SOIC Small Outline Integrated Circuit

SPI Serial Peripheral Interface bus

TQFP Thin Quad Flat-Pack

UART Universal Asynchronous Receiver/Transmitter

UPS Uninterruptible Power Supply

USB Universal Serial Bus

Appendix B

List of enclosed files

B.1 Delphi - tool for initial testing and debugging

Folder contains several archives with different versions of the tool that could be used to drive the Car4 vehicle with certain on-board FW, through RS232 line.

B.2 Eagle - PCB layouts and schematics

DIR: Car4_dsPIC33_AxleControlUnit_28PIN contains schematics and layouts for ACU28.

DIR: Car4_dsPIC33_AxleControlUnit_44PIN contains schematics and layouts for ACU44.

DIR: Car4_H_Bridge_with_ISL83204 contains schematics and layouts for ISL Power Board.

DIR: Car4_HW86010_-_RC_RS232_Remote_Control contains schematics and layouts for Remote control unit.

DIR: Car4_PowerUnit contains schematics and layouts for Power Unit for ISL PB.

DIR: Car4_servo_interface contains schematics and layouts for Servo Interface Board.

file: BobesovaKnihovna.lbr contains custom eagle library with all used parts that are not in standard eagle distribution.

B.3 MATLAB - Firmware, models

DIR: 100321_comunication These files show how the packets are handled. We are transmitting 12-bit and 16-bit values over UART which uses 8-bit values.

DIR: 100403_BasicFW This directory contains files with the implementation of the basic firmware including the customary C-code. Also projects for MPLAB are placed there, however, the paths are not relative, so the files in the projects would have to be re-linked.

B.4 related datasheets

Folder contains related documentation including various datasheets and applications notes related to the components used in design and during the development.