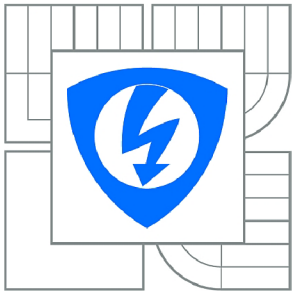
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

DESIGN OF MODULE FOR DEMONSTRATION AND TESTING OF SYSTEM BASIS CHIPS NCV7471

NÁVRH ZAŘÍZENÍ PRO DEMONSTRACI A TESTOVÁNÍ PRODUKTU NCV7471

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. MARTIN KRESTA

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. PETR FIEDLER, Ph.D.

SUPERVISOR

BRNO 2013

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student: Bc. Martin Kresta
Ročník: 2

ID: 115208
Akademický rok: 2012/13

NÁZEV TÉMATU:

Návrh zařízení pro demonstraci a testování produktu NCV7471

POKYNY PRO VYPRACOVÁNÍ:

Návrh zařízení pro demonstraci a testování produktu NCV7471 (SBC), který je určen pro elektronické řídící jednotky automobilů, převážně pro tzv. car body electronic.

DOPORUČENÁ LITERATURA:

1) Katalogové listy k NCV7471
2) Firemní dokumentace ON-Semi

Termín zadání: 11.2.2013 ***Termín odevzdání:*** 20.5.2013

Vedoucí práce: doc. Ing. Petr Fiedler, Ph.D.
Konzultanti diplomové práce:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ Fakulta elektrotechniky a komunikačních technologií
Ústav automatizace a měřicí techniky

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Abstrakt

Práce se zabývá návrhem automobilové elektronické řídicí jednotky (ECU) s funkcí partial networking definovanou normou ISO 11898-6. Cílem je navrhnout a vytvořit demonstrační ECU s použitím system basis chip NCV7471. Protože NCV7471 obsahuje standardní CAN transceiver, funkce partial networking je realizována pouze softwarem řídicí jednotky. Práce zvažuje možné způsoby realizace jak HW, tak SW části, tak aby byla zajištěna nízká spotřeba ECU v různých operačních módech, a snaží se sledovat současné trendy v automobilovém průmyslu.

Klíčová slova

Automobilové elektrotechnické systémy, Elektronická řídicí jednotka (ECU), Síť CAN, Partial Networking, Selective wake-up, Softwarové standardy v automobilovém průmyslu, AUTOSAR

Abstract

The thesis deals with the design of automotive ECU with partial networking (PN) functionality according to ISO 11898-6. Aim is to design and create evaluation electronic control unit (ECU) using system basis chip NCV7471. Since NCV7471 integrates standard CAN transceiver without HW PN support, the PN functionality is realized by ECU software. This thesis considers possible ways of realization in HW and SW domain to maintain low power consumption of the ECU in different operational modes in order to follow current trends in automotive industry.

Keywords

Automotive electronics, Electronic control unit (ECU), CAN network, Partial networking, Selective wake-up, Automotive software standards, AUTOSAR

Bibliografická citace:

KRESTA, M. *Návrh zařízení pro demonstraci a testování produktu NCV7471.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2013. 59 s. Vedoucí diplomové práce doc. Ing. Petr Fiedler, Ph.D.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma Návrh zařízení pro demonstraci a testování produktu NCV7471. jsem vypracoval samostatně pod vedením vedoucího semestrální práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené semestrální práce dále prohlašuji, že v souvislosti s vytvořením této semestrální práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: 1. května 2013

……

podpis autora

Poděkování

Děkuji společnosti ON Semiconductor za materiální pomoc a zaměstnancům Romanu Buzášovi a Ondřeji Kupčíkovi za velmi cenné rady při zpracování mé diplomové práce.

V Brně dne: 1. května 2013

……
podpis autora

Table of contents

1 INTRODUCION

Automotive electronic has undergone many changes in last decades. Nowadays electronic control units (ECUs) provide control or management in many domains such as power train control, safety systems and in competitive market still more important comfort features. With the raising complexity of in-vehicle electronic systems, the current consumption is no more negligible. The number of ECUs in modern car ranges from 40 up to 100 in high-end vehicles. Large part of these ECUs is dedicated to control systems, which may not be active all the time. Such systems are for example seat control modules, parking assistance or door control modules. The problem is that all ECUs in car are communicating together via shared bus. Nowadays, the most used in-vehicle communication bus is CAN (Controller Area Network), which requires that all connected ECUs must listen during pending bus communication. Therefore all ECUs must stay fully powered to be able to listen to the bus and respond to certain messages. This fact implies quite large waste of energy which can also increase a fuel consumption of vehicle with combustion engine or shorten a range of electric vehicles. Moreover, a legal regulations of CO2 emissions require reduction of the consumption whereas possible. Due to those reasons, car manufacturers are pushing to standardize functionality called partial networking (PN) in CAN standard ISO 11898. This feature is able to save significant amount of energy on currently unused ECUs. Partial networking requires a special CAN transceiver with selective wakeup functionality, therefore this became a challenge for semiconductor manufacturers.

The aim of this thesis is to explore in detail the possibilities of creation of ECU with selective wakeup functionality, using standard CAN transceiver or system basis chip with CAN transceiver. Since standard transceivers have no HW support for selective wakeup, it has to be realized by MCU firmware. Due to this fact, all power consumption requirements defined in preliminary PN standard are not expected to be met. The result of this work should show if the software realization of PN functionality brings reasonable benefit, and how much energy can be saved.

The second part of this thesis is focused on design and practical realization of demonstrative automotive ECU with software realization of PN functionality. The hardware of the ECU is designed with focus on low power consumption in different operational modes, while maintaining the concept similar to real ECUs used in modern cars. A system basis chip NCV7471, manufactured by ON Semiconductor, is used on the ECU in order to evaluate its power saving potential. The software is designed with focus to portability between the most of automotive MCUs. The possibilities of creating SW architecture partially compliant with AUTOSAR standard are thoroughly examined.

2 AUTOMOTIVE HW AND SW SOLUTIONS

This chapter describes hardware solutions and their software equipment used in modern cars. The principle of partial networking and selective wakeup functionality is described as well.

2.1 Electronic control unit (ECU)

Electronic control unit is the basic building block of in-vehicle electronic systems. In the modern car, there are tens of such units, and most of them are connected to central communication bus, most commonly CAN. Other ECUs can be connected to a local bus with certain function. For example LIN bus might be used for interior lighting. Such local buses are usually controlled by a bus master, which is connected to the central bus. The following text always deals with the ECUs connected to the central bus, which are generally more complex and always contain a microcontroller (MCU) or some programmable device.

Generally, the hardware of particular ECU is designed according to desired function(s) of the ECU. Therefore there are many types of different ECUs in vehicle with different level of complexity. However, almost all ECUs contain MCU, voltage regulator(s) and one or more bus transceivers. Other equipment is strongly dependent on purpose of the particular ECU. The estimated average current consumption of one ECU is 250mA in active mode [1].

2.2 In-vehicle communication bus (CAN)

The purpose of in-vehicle networks is to transport messages between ECUs. Special serial buses are preferred in automotive industry. They are designed to work safely in harsh environment of the vehicle. Nowadays, the most common automotive protocols are CAN, LIN, FlexRay and MOST. They vary in transfer speed and cost per node, as shown in figure 1. A High-Speed CAN, which can run at communication speed up to

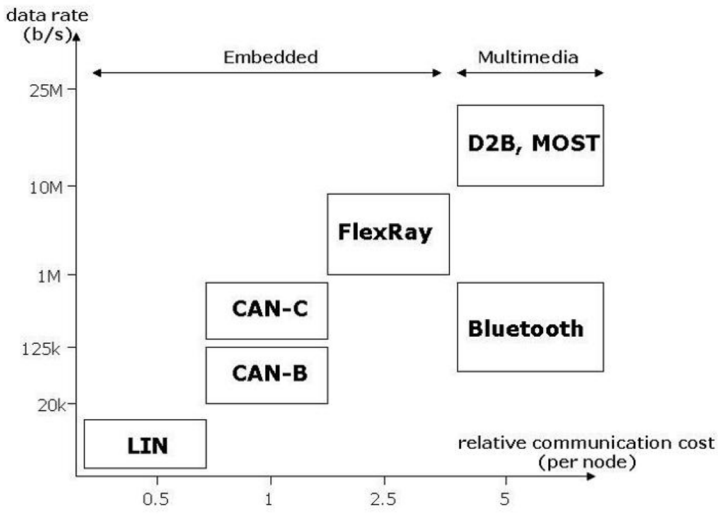


Figure 1 – Comparison of in-vehicle network protocols [2]

1 Mbps, is usually used as the central communication bus.

The CAN is standardized communication protocol (ISO 11898) [4]. This standard defines a physical layer (medium, bus levels, termination) as well as a transfer layer (message framing, arbitration, error detection, etc.). On the figure 2, a typical CAN network can be seen. Medium is a twisted-pair cable with lines named CANH and CANL and appropriate termination. Each node contains CAN transceiver and DSP or MCU with CAN controller. Binary data are represented by recessive (log. 1) and dominant (log. 0) bus states. The bus realizes so-called wired-AND, so if at least one node transmits dominant, the dominant "wins" and appears on the bus. If all nodes transmit recessive, recessive appears on the bus.

Intent of this chapter is not to describe principles of CAN protocol in details, but it is good to state the following facts for better understanding of this text. The CAN is multi-master serial bus with arbitration. All nodes (ECUs) connected to this bus are able to send and receive messages, but not simultaneously. The arbitration uses a priority, which is assigned to certain messages, not to physical nodes. When one node is sending a message, all other connected nodes have to listen to the bus and detect this message. As a consequence, when some communication is pending, all ECUs must stay powered in active mode, even if the message is not relevant for them. ECUs can switch to power-saving mode only if there is no communication on the bus for certain time (bus idle state).

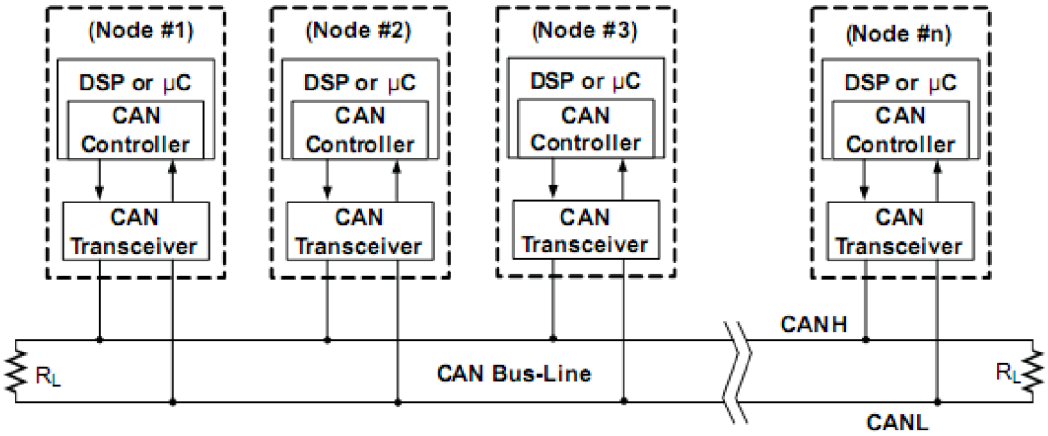


Figure 2 – Typical CAN bus with connected nodes (ECUs) [3]

2.3 Partial & Pretended Networking

Due to legal requirements for reduction of CO2 emissions given by governments around the world (US, EU, Japan) [1], car manufacturers are looking for ways how to decrease fuel consumption and emissions. Because electronic systems in combustion engine vehicles are powered from an engine-driven alternator, the savings in electrical energy consumption can significantly decrease fuel consumption and CO2 emissions. In modern cars, relatively a lot of electrical energy is wasted by ECUs that are rarely active, but have to stay powered all the time, because they are connected to CAN

network (see chapter 2.2). To handle this issue, an extension of CAN standard (ISO 11898-6) was created. It defines new functionality of CAN transceiver called “selective wakeup”.

The ECUs equipped with such transceiver can be switched to “selective sleep mode” while the other ECUs are communicating via bus. In this mode, the ECU’s circuits (include MCU) are unpowered, and only device which consumes power is CAN transceiver reducing the consumption below 500 µA in this mode [5]. The ECU can be woken-up to active mode by special predefined CAN message called wake-up frame (WUF). Since the PN CAN transceiver has to decode the messages on the bus and recognize valid WUF, it has to contain a precise internal oscillator and more complex logic then standard transceivers. Some semiconductor manufacturers (NXP, Infineon, ST Microelectronics) already developed and presented first stand-alone transceivers and system basis chips with selective wakeup functionality. The figure 3 shows an ECU with partial networking transceiver (presented by ST [7]). As can be seen, the partial networking allows power savings on rarely used ECUs. The amount of saved energy is dependent on number of ECUs which can be switched to the selective sleep mode. According to some calculations, the estimated saved energy in a typical EU vehicle is 33 W, which is equivalent to reduction of CO2 emissions by 0.88 g CO2/km. The details about the estimation can be found in [1].

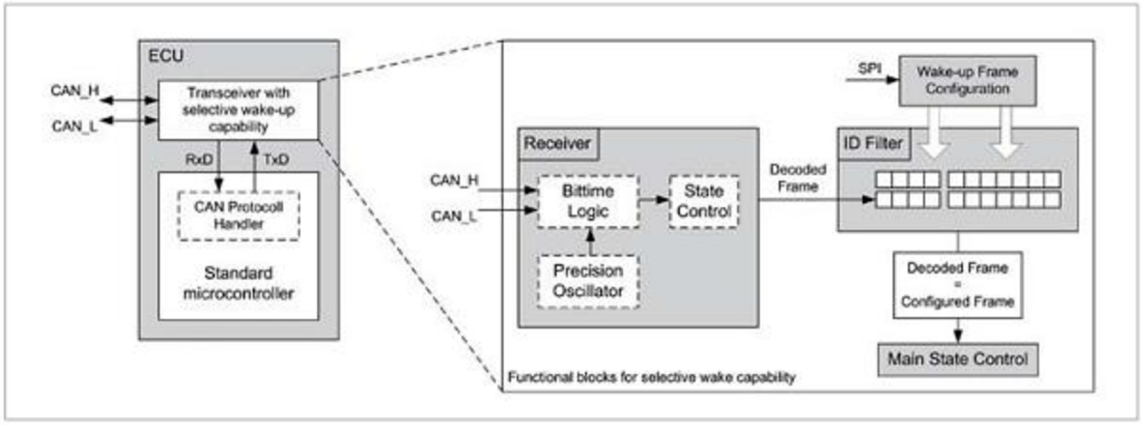


Figure 3 – ECU with partial networking transceiver [7]

Another power saving technique is so called Pretended Networking [6]. In contrast with Partial Networking, this approach does not require any special hardware. The basic principle of the Pretended Networking is to reduce the ECU power consumption by deactivation of unused HW peripherals, while bus communication is maintained. The Pretended Networking do not allows as significant power savings as Partial Networking, however the shorter ECU wakeup time and simpler network management [6] make it good alternative in some cases.

2.4 CAN Transceivers

The bus transceivers are semiconductor devices which translate signal levels on the bus to digital signal levels, which can be recognized by a microcontroller. The requirements on CAN transceivers are defined by ISO11898-2 and ISO11898-5 standards. CAN transceiver is connected to CANH, CANL lines on the bus side and to RXD, TXD signals on the microcontroller side. In addition, transceivers are able to detect various kinds of bus errors and signal it to the MCU by a single pin or by SPI. Transceiver also prevents bus from being continuously driven to dominant state that can be caused by malfunction of the MCU (HW or SW).

2.5 System basis chip NCV7471

The CAN bus transceiver can be realized as a stand-alone device (stand-alone transceiver), or as a part of so-called system basis chip (SBC). SBC is a device integrating some typical functions used in automotive ECUs in one IC. Some of the typical features integrated in SBC are voltage regulators and supervisors, bus transceivers and external watchdogs for application running on MCU. These devices are usually configured and controlled by the MCU via SPI. For the purposes of this thesis, a system basis chip NCV7471, developed by ON Semiconductor, is used.

NCV7471 integrates the following main features [8]:

- Control logic
  - Controls mode transitions including the power management and wakeup treatment (bus wakeups, local wakeup)
  - Generates reset and interrupt request for MCU
- Serial peripheral interface (SPI)
  - Mode settings, configuration
- 5V VOUT supply (DC/DC converter)
  - Can deliver up to 500mA with accuracy of ±2%
  - Supplies typically ECU's microcontroller
- 5V VOUT2 supply (low-drop output regulator)
  - Supplies typically external loads, e.g. sensors
  - Controlled by SPI
  - Protected against short to car battery
- A high-speed CAN transceiver
  - ISO 11898-2 and ISO 11898-5 compliant
  - TxD dominant time-out protection
- Two LIN transceivers
  - LIN2.1 and J2602 compliant
  - TxD dominant time-out protection

- Wakeup input (WU)
    - Edge-sensitive high-voltage input
- Protection and monitoring functions
    - Monitoring of the main (VBAT) supply
    - Monitoring of VOUT supply output
    - Diagnosis of VOUT2 supply output
    - Thermal warning and thermal shutdown protection
    - Programmable watchdog
    - FSO outputs to control a application under failure conditions

Block diagram of NCV7471 can be seen on figure 4. Detailed description of this device can be found in the datasheet [8]. The control logic of NCV7471 is represented as a state-machine, and offers various configurations of integrated features. The state-machine includes two low-power modes (standby mode and sleep mode). Independently of the device's modes, the integrated CAN transceiver can be switched to one of four operational modes. Thanks to wide configurability via SPI and included DC/DC voltage regulator, NCV7471 is a good choice for design of ECU with high efficiency and low power consumption.

2.6 Automotive SW standard AUTOSAR

In nowadays cars, almost all features are controlled by electronic systems, which also imply complex software systems. Furthermore, due to increasing legal and passenger requirements in aspects as safety, convenience or driver assistance, further grow of software complexity can be expected. Driven by these facts, automotive OEMs and their suppliers developed a software standard AUTOSAR to efficiently manage the software design.

AUTOSAR (AUTomotive Open System ARchitecture) is open and standardized automotive software architecture, developed by car manufacturers, suppliers and tool developers [9]. Key features of this standard are modularity, configurability, transferability and standardized interface of the software modules. Although the main intention of AUTOSAR is to define SW architecture in meaning of software modules and their interfaces (API) used in ECUs, standard also thoroughly defines a methodology of software design. Since that fact, AUTOSAR is not only SW, but also a technology standard.

Simplified software architecture of AUTOSAR ECU is depicted in figure 5. AUTOSAR is hierarchically layered architecture which can be basically separated to three main layer groups [10]. The lowest one is a Basic software layer (BSW). Modules in this layer can be dependent on the ECU hardware (MCU, ECU on-board devices, ECU network connections). The purpose of BSW layer is to abstract the ECU hardware

for higher layers and make them fully independent on the hardware. BSW layer is further divided into few layers with special meaning. All these layers are composed from so-called BSW modules with standardized functions and interfaces.

Application layer consists of software components (SWC), which implements particular functions of in-vehicle systems (air-conditioning, break systems, interior lighting, etc.). Software components are divided into three types (application, actuator and sensor SWC).

Runtime Environment (RTE) layer provides communication services for SWC in

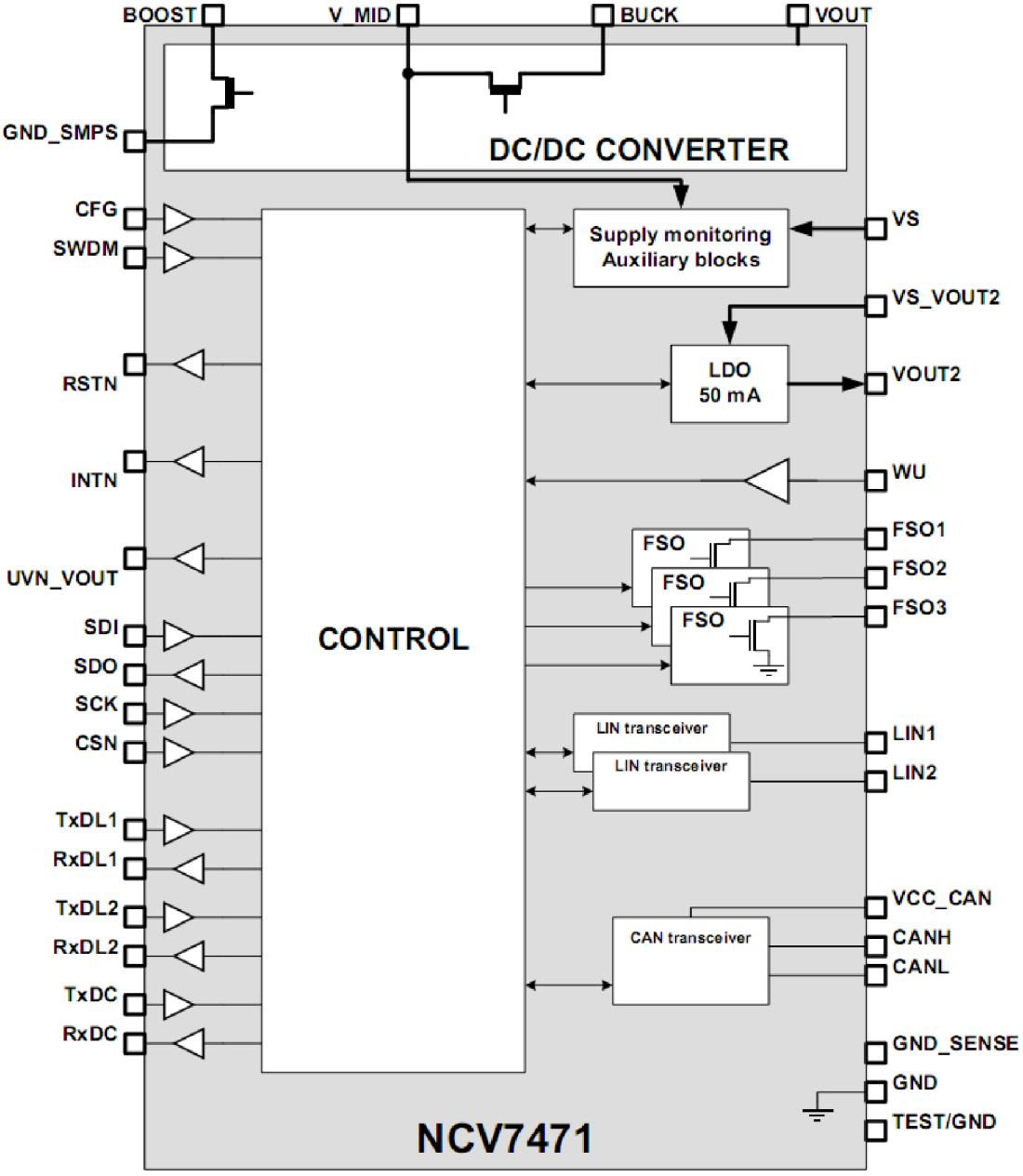


Figure 4 – Block diagram of NCV7471 [8]

application layer. Software components communicate together via RTE, which covers intra- and also inter-ECU communication. While implementation of RTE is ECU specific, the upper interface is completely ECU independent, which makes SWC independent from mapping to specific ECU [10].

The architecture allows creating a distributed system with effective usage of HW resources (memory, CPU, network). AUTOSAR assumes that particular BSW modules and SWCs will be implemented by different suppliers, for example drivers in Microcontroller Abstraction Layer (MCAL) can be provided by MCU manufacturer. A system of machine-readable XML description files was introduced in AUTOSAR to allow composition of final distributed system from individual modules. Since all AUTOSAR modules are provided with this description files, sophisticated tools allow automated analysis, configuration and composition of complex AUTOSAR ECUs and distributed systems. Without this design tools, provided for example by Elektrobit or Vector Informatics, it is practically impossible to create ECU SW fully compliant with AUTOSAR.

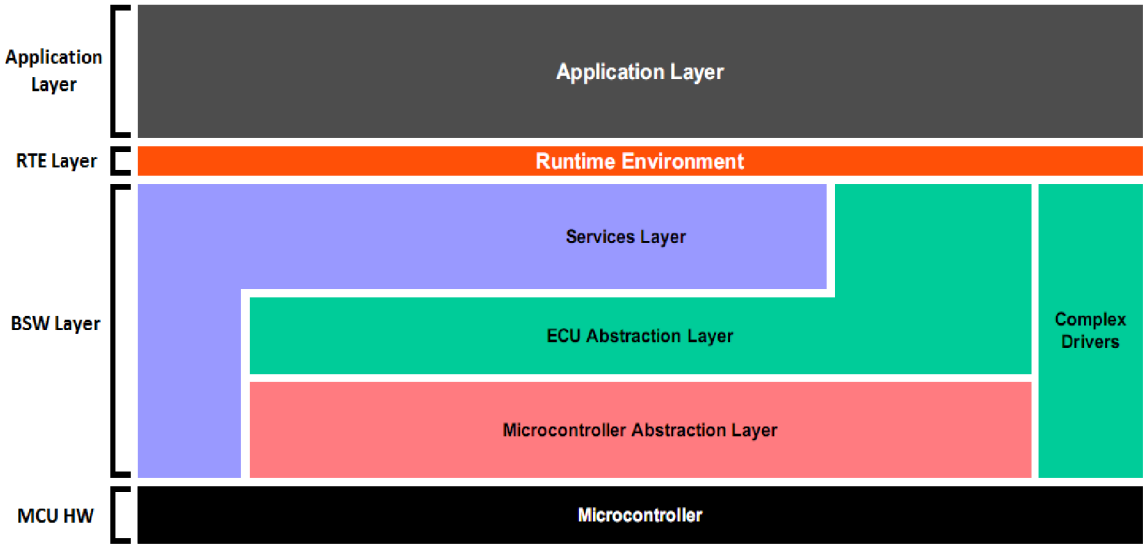


Figure 5 – Main layers of AUTOSAR software architecture [9]

3 DESIGN OF ECU WITH PARTIAL NETWORKING

This chapter introduces the main concept of designed ECU.

3.1 Partial networking without HW support

The main aim of the partial networking is to minimize unnecessary wasting of energy by the ECU. As defined by ISO 11898-6, the ECU has to support selective sleep mode and comply with the limits for wake-up time to normal mode. Furthermore, the current consumption of the ECU in this mode should be as low as possible. Because the designed ECU will not use a PN CAN transceiver, all CAN messages have to be processed by the MCU, also in selective sleep mode. This fact has large impact on selection of the MCU. Most of the automotive microcontrollers manufactured by renowned brands (Infineon, Freescale, Atmel, Renesas) offers wide range of different low power modes. Microcontroller families differ in wake-up times and current consumptions. S12X family manufactured by Freescale was selected for the design, as one of the more suitable choices. This MCU family integrates, besides other features, also automotive IVN controllers (CAN, LIN), low power oscillator circuit and internal PLL clock generator. Big advantage of this MCU family is also high configurability of clock source and clock distribution to individual MCU blocks. These features will be reviewed in chapter 3.5.

3.2 ECU HW description

The concept of designed ECU hardware is depicted in figure 6. Two main parts of the ECU are the MCU and SBC NCV7471. These devices communicate together via SPI and individual single-pin signals. Besides that, there are two LIN and one CAN interfaces. The ECU board can operate with supply voltage in the range of 3.5 to 26 V defined by operational ranges of used components [8][20]. For the evaluation of power savings, the ECU will be prepared for measurement of time-varying current consumption by an oscilloscope. This feature is depicted in figure 6 as Current sensing block. Two LIN and one CAN connectors allow connecting the ECU to three independent IVN networks. All these buses are equipped with particular termination and connected to NCV7471. For supplying external devices (nodes) connected to the LIN buses, the LIN connectors contain also VBAT pin. For the evaluation purposes, the ECU is equipped with ECU state diagnostic module, which consists of LED indicators and test points for connection of a logic analyzer.

As there is no requirement on specific ECU function (e.g. seat control module), the designed ECU does not contain any specific device such as motor driver, sensor, etc.

For the demonstration of practical ECU function, the ECU board is equipped with an expanding port (see figure 6). This port contains VBAT power supply line, MCU controlled 5 V power supply line, SPI interface and general purpose I/O pins. Various expanding boards designed for demonstration of different products of ON Semiconductor can be connected through this port. Board is also equipped with OSBDM connector for programming and debugging purposes.

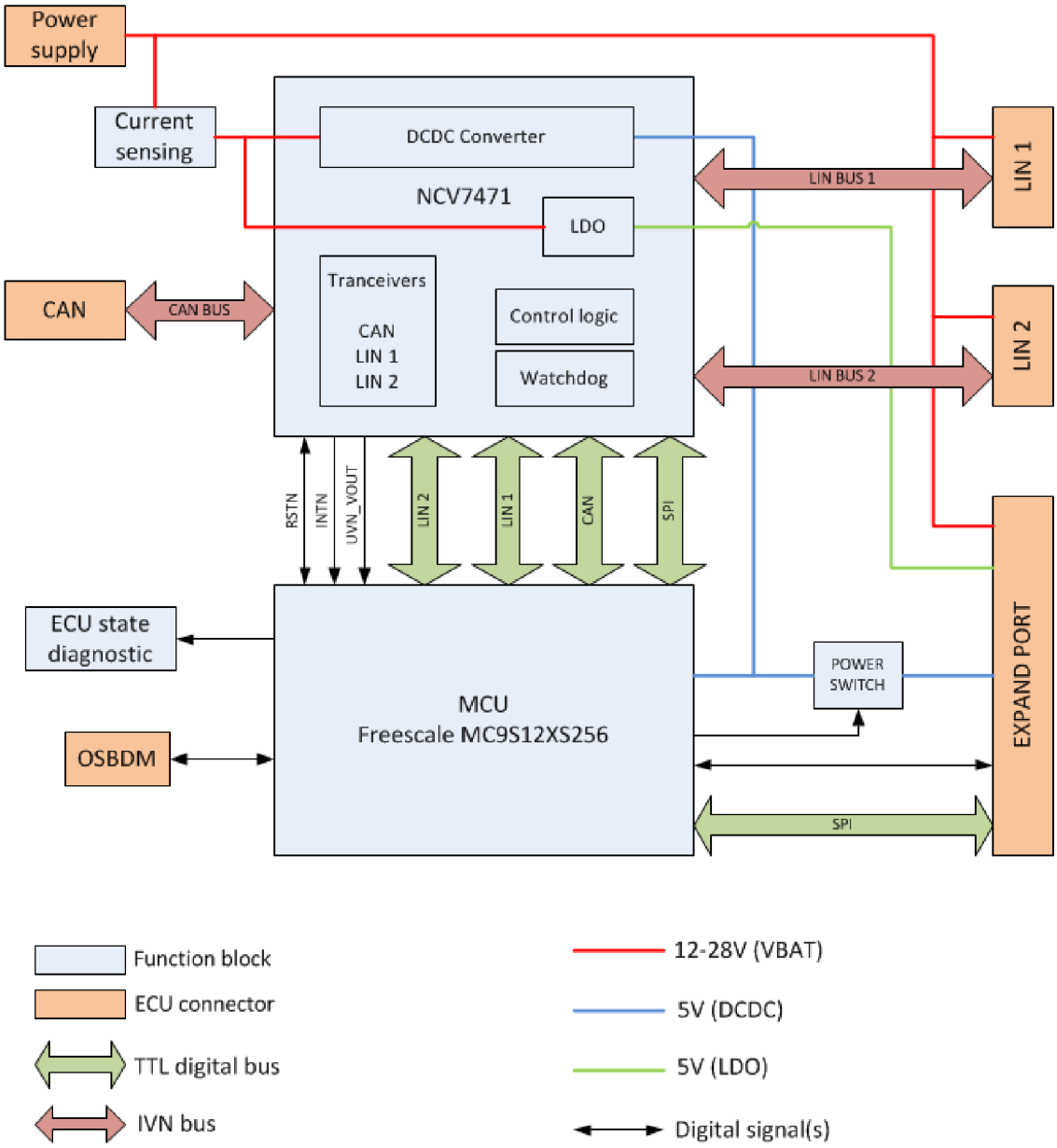


Figure 6 – Block diagram of designed ECU hardware

3.3 Impact of AUTOSAR on SW design

The most challenging part of this thesis is the design of the software architecture and implementation of executable code. One of the initial requirements was to create a driver for SBC NCV7471 which would be transferable between different automotive MCU platforms. This requirement strongly leads to use of AUTOSAR software architecture.

The best approach would be creation of BSW layer (see chapter 2.6) compliant with AUTOSAR, and non-AUTOSAR application software for demonstration of ECU functionality. This approach assumes to use BSW modules (mainly MCAL layer) and development tools provided by AUTOSAR members (Freescale, Elektrobit, Vector Informatic, etc.). However, after a small market survey, it seems these products are not available for the evaluation purposes or student projects. Without this equipment, it is not feasible in the frame of this thesis to make complete BSW layer. Instead of that, the BSW layer architecture will be simplified. Nevertheless the drivers for NCV7471 will be implemented according to AUTOSAR standard in relation to functions and interfaces (API). To maintain the transferability of NCV7471 drivers, underlying BSW modules (MCAL layer), which are directly used by these drivers, will be also implemented.

3.4 Concept of SW architecture

The software architecture for the evaluation ECU was designed with focus on simplicity and easy realization (figure 7).

As AUTOSAR doesn't define any SBC driver, the drivers for these devices are realized as separated modules. Each of these module covers one of the SBC functionalities; there will be CAN Transceiver driver and Watchdog driver in case of NCV7471. The CAN Transceiver driver is described in AUTOSAR standard [11]. Besides stand-alone transceivers, also transceivers integrated in SBCs are supported in release 4.0. Specification of this driver covers only standard CAN transceivers, since Partial Networking is not supported in release 4.0. The Watchdog driver is defined in AUTOSAR document [12] and specification of this driver is common for internal (MCU) and external (SBC) watchdog. Both mentioned drivers need an SPI handler/driver [13] for interaction with SBC. In addition, CAN Transceiver driver also requires a DIO driver [14] for sensing/driving single-pin signals. SPI and DIO drivers belong to the lowest layer of the AUTOSAR architecture and directly work with the MCU hardware. All four mentioned BSW modules will be implemented according to the AUTOSAR specifications.

Next part of SW architecture is the Evaluation software. This part won't be in conformance with AUTOSAR layered architecture [10]. Evaluation software will implement various features from covering functionalities of missing BSW modules,

mainly in MCAL and ECU abstraction layer, to realization of main evaluation functions (Application layer). Therefore the Evaluation software belongs to all four main layers of the AUTOSAR architecture.

The modules covering the functionality of the Partial Networking will be also realized in the Evaluation software. Because a CAN transceiver driver for standard CAN Transceiver is used, all PN functions, including RWUF detection, are realized in Evaluation software. The ECU and CAN state managers belong to imaginary services layer, while the MCU state manager belongs to MCAL layer. The module which implements mechanism for RWUF detection is not described in AUTOSAR, but because it is not dependent on ECU HW, it should be on the level of services layer.

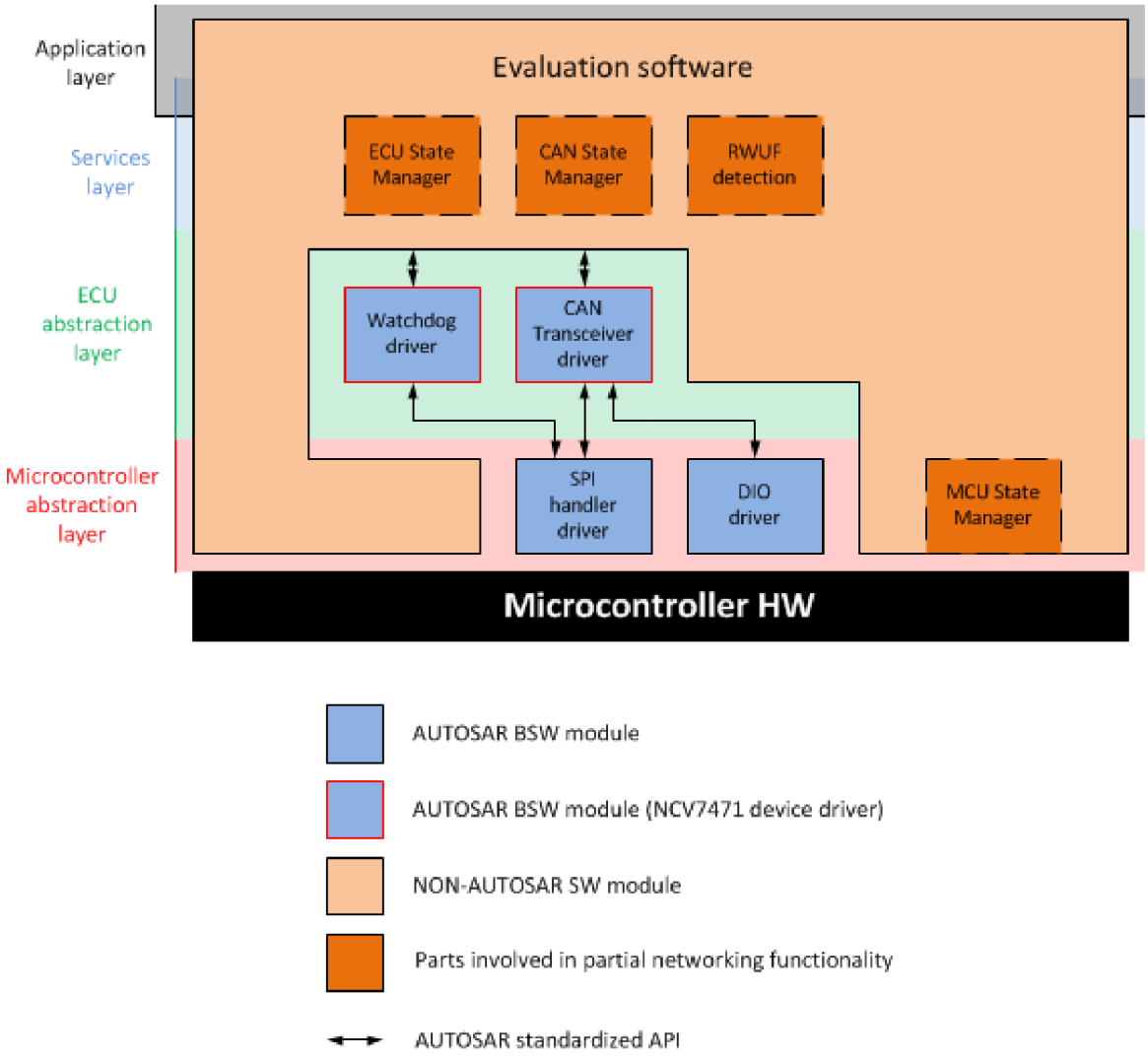


Figure 7 – Software architecture for designed ECU with Partial Networking functionality

3.5 ECU operational modes

For the energy saving, several operational modes of the ECU, including Partial networking modes, were designed. Expected current consumptions of MCU MC9S12XS and SBC NCV7471 were calculated for these operational modes. The operational modes are summarized in table 1. The last column of the table represents the required maximum current consumption of PN CAN Transceiver defined by ISO 11898-6.

Configuration of CAN Transceiver in different ECU modes is inspired by the operational modes of PN CAN Transceiver defined in [5].

ECU mode	SBC NCV7471	MCU Freescale MC9S12XS	ECU HW peripherals	11898-6 Standard
Standard sleep mode PN functionality disabled RWUP detection	Sleep mode CAN - wakeup VOUT - off Ic = 60µA	Unpowered	Unpowered	-
Fast-startup sleep mode PN functionality enabled RWUP detection	Standby mode CAN - wakeup VOUT - on Watchdog - off Ic = 100µA	Pseudo stop mode Oscillator 8 MHz Start-up time cca 3,5µs Ic = 200µA	Unpowered	30µA
RWUF detection mode PN functionality enabled RWUF detection	Standby mode CAN – receive-only VOUT - on Watchdog - off Ic = 10mA	Run mode SYSCLK- oscillator (8 MHz) CAN – listen-only PLL - disabled Ic = 7mA	Unpowered	500µA
Normal mode	Normal mode CAN - normal VOUT - on Watchdog - on VOUT2,LIN - by SPI Ic = 100mA	Run mode PLL can be used for SYSCLK CAN - normal Ic = 7-30mA (depends on SYSCLK)	Powered Ic = up to 450mA	-

Table 1 – Operational modes of designed ECU with PN functionality

In Standard sleep mode all ECU devices are unpowered, except SBC which is powered in Sleep mode with CAN wakeup detection enabled. This mode is equal to the sleep mode of standard CAN Transceiver, or PN CAN Transceiver with disabled PN functionality. The ECU can be woken up from this mode to normal mode by wakeup pattern (RWUP). RWUP is special pattern defined by ISO 11898-6 [5], which causes a wake up of all nodes in standard sleep mode connected to particular bus.

Fast-startup sleep mode is similar to the Standard sleep mode. The difference is that detected RWUP will not cause wakeup to normal mode, but switching to RWUF detection mode. The MCU is powered to maintain the requirement on detection of following frames in this mode. In other words, the MCU shall be able to detect correctly CAN frames after RWUP detection with low delay. For this reason, the MCU is in pseudo-stop mode [15] where the MCU oscillator is running, but clock signal is not distributed to ALU neither to MCU peripherals.

In RWUF detection mode, the MCU is configured to run mode, system clock SYSCLK is derived directly from an oscillator, PLL circuit is disabled. Only MCU core and CAN controller are enabled in this mode, all other MCU peripherals are disabled (disconnected from the system clock) to keep the consumption as low as possible. After switching to run mode (caused by INTN interrupt), the CAN transceiver is configured to receive-only mode via SPI. Now the MCU is able to detect possible wakeup frames (RWUF). The current consumptions in RWUF detection mode is shown in table 1. These values are calculated as the worst cases from the datasheets of the individual devices. The real consumption is expected to be significantly lower, but still higher then consumption of PN CAN Transceiver.

Finally, in Normal mode, full functionality of ECU is enabled. The MCU is in run mode and internal PLL circuit can be used to generate system clock. Different MCU peripherals can be enabled according to the application. The SBC NCV7471 is configured to normal mode and SBC on-chip watchdog is enabled. CAN Transceiver is in normal mode to allow reception and transmission of CAN messages. Other SBC features (LIN Transceivers, LDO regulator) can be configured via SPI. Also other on-board HW peripherals, associated with the function of particular ECU, are powered.

4 HARDWARE DESIGN OF ECU BOARD

Following chapter describes hardware realization of the designed ECU. This includes reasoning of the used components, description of the function blocks and the designed PCB layout. Component designators in the following text refer to the ECU schematic (see appendix 1).

4.1 Requirements on hardware design

Requirements like low cost, small dimensions and high reliability are obvious for all currently developed electronic devices. Because designed evaluation board should be similar to the automotive ECUs, also the EMC properties are important. This requirement mainly concerns PCB layout of the switched DC/DC converter and the CAN channel. Further, in contrast with the typical ECU, our board should be equipped with the easy accessible features for the evaluation purposes. This includes testpoints for measuring probes, buttons, switches and LED indicators. The design is further influenced by the internal standards used in ON Semiconductor's Brno Design Center, concerning mainly the used types of bus connectors. For the marketing and cost reasons it was preferred to use semiconductor components manufactured by ON Semiconductor.

4.2 DC/DC converter circuit

The main low voltage power supply of the ECU is provided by DC/DC converter. The control logic and switching transistors are integrated in the SBC device NCV7471. External part of converter circuit is designed with respect to the recommendations given in product datasheet [8] and application note [16] of NCV7471.

The DC/DC converter consists of boost and buck stages. Simplified schematic of the converter circuit is depicted in figure 8. In the figure, three important nodes are marked. The node *VS* represents filtered battery supply voltage and acts as an input of the boost stage. *Vmid* is a middle point between the boost and the buck stage and provide supply voltage for internal circuits of NCV7471 like voltage reference, internal regulator or wakeup detector of CAN bus [8]. Finally, the node *Vout* is output of the buck stage

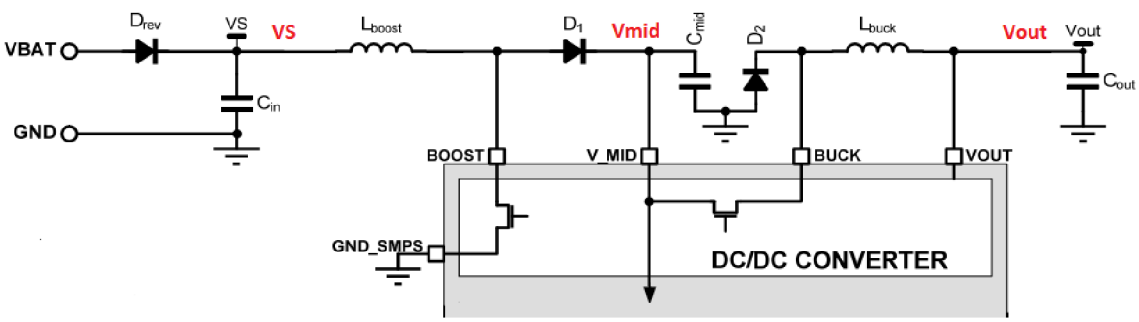


Figure 8 - Simplified circuit of DC/DC converter with marked nodes [15]

delivering regulated 5V output with the under-voltage detection feature. When the boost stage is enabled (by SPI command) and the *Vmid* voltage drops under approximately 6.5V, the boost stage becomes active. It allows to maintain functions of the NCV7471 logic and whole ECU even for input voltage *VS* dropping down to 2.5V. The buck stage then works as a step-down converter providing 5V output. The application note [16] proposes complete sets of external components (inductors, capacitors), calculated for different application needs. Because the goal of the thesis is to design typical ECU and demonstrate capabilities of NCV7471 device, the recommended components which utilize both stages of DC/DC converter and offers maximal output current of 500mA are used.

As can be seen in schematic (appendix 1), an input supply voltage *VS* is filtered by electrolytic capacitor C1, 220µF. Similarly output of the buck stage is filtered by a pair of ceramic capacitors C23 and C24, both of 10µF capacity.

The *Vmid* voltage is further used as a supply voltage for an internal 5V LDO regulator. The Voltage from *Vmid* node is leaded through RC low-pass filter (R20, C22) to pin VS_VOUT2 (see appendix 1). This allows that LDO can utilize the boost stage of DC/DC convertor for low battery voltage conditions.

4.3 Bus terminations and connectors

The ECU is equipped with one CAN and two independent LIN interfaces. Each of these interfaces is accessible via separate connector assembled on the ECU board.

CAN bus interface utilize standard 9-pin D-SUB connector with lines CANH, CANL, GND and VBAT. The VBAT pin can be optionally used as power supply input. Both CAN bus lines are protected by bidirectional ESD protection D2. Device NUP2105L is specially designed to protect the CAN transceivers in high-speed or fault tolerant networks [17]. The CANH and CANL lines are routed from the connector to NCV7471 pins through common mode choke L1. This choke with inductance of 100µH rejects common mode noise on differential bus and thus improves electromagnetic compatibility. Further, the board is equipped with slots for soldering of termination resistors. Termination is realized as so called split termination formed by two 62Ω resistors and one 4.7nF capacitor connecting their middle to ground. This type of CAN termination is defined and required by car manufacturers [18]. The termination circuit should be assembled only if the ECU is placed at the end of CAN bus.

The hardware of both LIN channels is identical. Each channel has its own 4-pin connector RJ11 contacting LIN, GND and VBAT lines. Here, the VBAT pin acts as power output allowing supplying of LIN nodes potentially connected to the ECU. The termination circuit consists of a reverse current protection diode D5, pull-up resistors R5, R7 and capacitor C5 (see appendix 1). The values of these components are given in [18]. Because maximal allowed size of 1kΩ pull-up resistor is 1206 package and the termination has to be able to handle power dissipation of at least 500mW, the pull-up

resistor is made by parallel combination of two 2kΩ resistors in 1206 package with maximal power dissipation of 250mW.

4.4 System Basis Chip peripherals

Except the circuitry of DC/DC converter and proper bus terminations, System Basis Chip NCV7471 requires few other components for correct function. The used components again come from recommendations given in [8].

The NCV7471 contains two digital configuration pins (CFG, SWDM). These are connected through 10kΩ resistor to middle pad of two-pole soldering jumper. This allows connecting these pins either to *VS* line (HIGH) or to ground (LOW). The selectable configuration is useful in software development phase; in final application the configuration stays fixed. The soldering jumpers were used instead of the classic through-hole ones because of their smaller footprint and higher reliability.

Further, a wake-up button with pull-up resistor R25 is connected to WU pin through 33kΩ resistor R29. The wake-up line is also connected to pin WUex of expanding port. Thanks to that, externally connected board can initiate wake-up event by driving this pin to ground. The on-board wake-up button together with the potential external switch creates so-called wired-or connection.

The output pin FSO2, internally realized as open collector, is used for signalization of Fail-safe mode operation. Red LED D17 connected to VBAT supply line through current limiting resistor R17 is used for signalization. The unregulated supply line VBAT (3.5-26V) is used, because in fail-safe mode, the DC/DC convertor is deactivated and regulated 5V line is unavailable. R17 resistance of 2kΩ maintains reasonable currents through LED in wide range of VBAT voltage level. SMD package 1206 is used for handling of higher power dissipation.

Finally, three low-voltage digital pins (RSTN, INTN, UVN_VOUT) are biased to HIGH level by 10kΩ pull-up resistors connected to 5V line (*Vout*).

4.5 Microcontroller peripherals

The microcontroller MC9S12XS256 (U1) is a heart of the ECU and uses various interfaces for interaction with other components.

MCU and SBC are interconnected by 4 independent serial interfaces and 3 single digital lines. The SPI bus is used for communication between MCU and SBC. Other 3 buses are Rx-Tx pairs for CAN and two LIN transceivers integrated in SBC. All signals of these serial buses are realized by simple pin to pin connection without any additional components. The single digital lines, namely reset (RSTN), interrupt request (INTN) and under voltage detection (UVN) are equipped with pull-up resistors (see chapter 4.4 System Basis Chip peripherals). The RSTN pin of SBC can be disconnected from MCU reset pin by removing jumper RST. This is necessary to ensure that potential resets

generated by SBC cannot overdrive reset level generated by serial programmer during software download phase. The INTN signal is connected to the MCU through two-pole soldering jumper, allowing use of either maskable (IRQ) or non-maskable (XIRQ) external interrupt [15].

The expanding port is universal interface for connection of various external modules. The 20-pin socket-type connector contains 6 general-purpose digital I/O lines directly connected to MCU GPIO pins and 3 analog inputs connected to ADC inputs of MCU. All these analog inputs are prepared for potential assembly of RC low-pass filter. Further, the connector offers SPI bus with dedicated chip select signal CSN2. Also one LIN channel is available directly on the expanding port. The circuits connected to the expanding port can be supplied by one of three available power supply pins. The battery voltage VBAT is available on the first pin. The 18^{th} pin provides output of 5V LDO regulator (VOUT2) and the 20^{th} pin can provide 5V power supply from DC/DC converter. The last mentioned supply pin can by disconnected from VOUT line by high-side switch controlled from MCU. The high-side switch is realized by the P-channel MOSFET T14 controlled by digital signal EXPEN. The used MOSFET SFT1314 is dedicated for switching applications and offers low R_{dsON} even with low V_{GS} voltage. Transistor is switched to conductive state by LOW level on EXPEN signal. This power switch is beneficial for power consumption control in the low-power operational modes.

Further, the ECU contains 4 switches and 7 diagnostic LEDs for the evaluation purposes. The switches are assembled in one 8-pin SMD package connected directly between the MCU pins and the ground node. This simple connection utilizes pull-up resistors integrated in MCU. The diagnostic LEDs are supplied by VOUT voltage (5V) connected in series with current limiting resistors 1.5kΩ, resulting in current approximately 2mA. First three LEDs (D10-D12) are dedicated for signalization of current ECU power mode. Other 4 LEDs (D13-D16) can be used for visualization of bus traffic (CAN, LIN1 and LIN2). All these 7 LEDs can be disconnected from supply voltage by jumper LEDS. That allows more precise measurement of power consumption of the ECU in low-power operational modes.

The debug interface consists of 6-pin header compatible with serial programmer/debugger PEmicro [19]. Only four of six pins are used. Except the 5V supply pins, there is reset signal MCU_RST and serial data line BKGD. Both these digital lines are biased by pull-up resistor 10kΩ. The reset signal is further equipped with button switch RESET for manual reset of the ECU.

A standard oscillator circuit created by 8MHz SMD crystal and two ceramic capacitors C10, C11 is connected to dedicated pins (EXTAL, XTAL).

Each supply pin of the MCU (8 in total) has own ceramic blocking capacitor with capacity either 100nF or 220nF. The supply pin of integrated ADC VDDA is further filtered by RL filter, as recommended in [15].

4.6 Current sensing circuit

The current sensing block is part which is normally not present in automotive ECUs. The current sensing feature is added to allow more advanced measurement of current consumption in low power modes related to subject of this work. The main idea is to monitor the current flowing from the battery supply (VBAT) to input of the DC/DC convertor (VS). This monitored current automatically reflects the power consumption of all the devices supplied by one of the SBC's voltage regulators (VOUT, VOUT2), mainly the MCU. The consumption of the externally connected devices supplied directly from the battery (VBAT), as well as consumption of diagnostic LED D17 is not monitored.

The current sensing circuit utilizes integrated current shunt monitor INA210 manufactured by Texas Instruments [20]. The device is connected in typical application scheme for high-side monitoring as presented in figure 9 with supply pin V+ connected directly to VBAT voltage. Because the reference input REF is in our application connected to ground, the current monitor circuit amplifies the voltage drop on sensing resistor R30 (1Ω) and provides the voltage output referenced to ground node. The device INA210 is used because it works with supply voltages ranging from 2.7 up to 26V and offers very low offset voltage, typically 0.55µV [20]. Because the monitored current is drawn by DC/DC discontinuously, an additional electrolytic capacitor C30 is used to filter the input signal. With the described connection utilizing 1Ω sensing resistor and voltage amplification of 200V/V, it is possible to measure currents from tens of microamperes with reasonable precision. The upper measurement limit depends on battery voltage. For the VBAT voltage 12V it is almost 60mA. The voltage output is further prepared for assembling of RC low-pass filter (R35, C35) in case of a need.

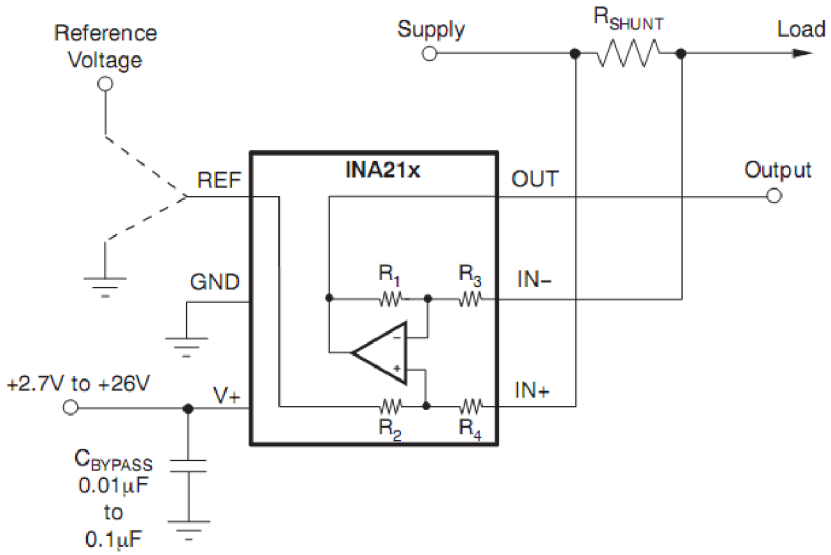


Figure 9 - Typical application of INA21x as high-side current monitor [20]

Whole current sensing circuit is designed for measurements of current consumption only in low power modes. Since in normal mode the consumed current can rise up to hundreds of mA, the output of current monitor could go to saturation. More importantly, the 1Ω resistance between filtered battery voltage VBAT and input of the DC/DC converter VS could result in unstable operation of SMPS. This problem is solved by bypassing of sensing resistor by MCU controlled MOSFET switch. The gate voltage of P-channel MOSFET T30 is given by voltage divider R31, R32 supplied by VBAT voltage. The current through this divider is switched on/off by digital transistor T31. The digital transistor MMUN2213L does not require any external resistors and allows to be controlled directly from MCU pin. The HIGH level on signal RSBP turns transistor T31 to conductive state and activates T30. The designed divider ratio 470:33 results in sufficient gate-to-source voltage for turning the channel on, even for battery voltage going down to 3.5V[1]. The maximal gate-to-source voltage is then limited by Zener diode D30 to 8.2V. This ensures safe operational conditions even for high battery voltage.

4.7 PCB layout design

The whole ECU is made on one two-layer PCB with compact rectangular dimensions of 65x69mm. The photograph of assembled board is in figure 10. More detailed drawings of top and bottom layers together with assembly drawings are then presented in appendix 2.

Most of the used components are SMD devices. The through-hole components are only connectors, jumpers and testpoints. The preferred housing of resistors and capacitors is package 0603. In cases where bigger package is necessary due to the power dissipation of resistors or higher capacity of capacitors, the package 1206 is used. For easy accessibility, all connectors, jumpers, switches, LEDs and testpoints are assembled on the top side of PCB. On the bottom side, there are four support feet, one in each corner.

[1] Note that in used application drain and source electrodes have always almost same potential. Thus $V_{GS} \approx V_{GD}$

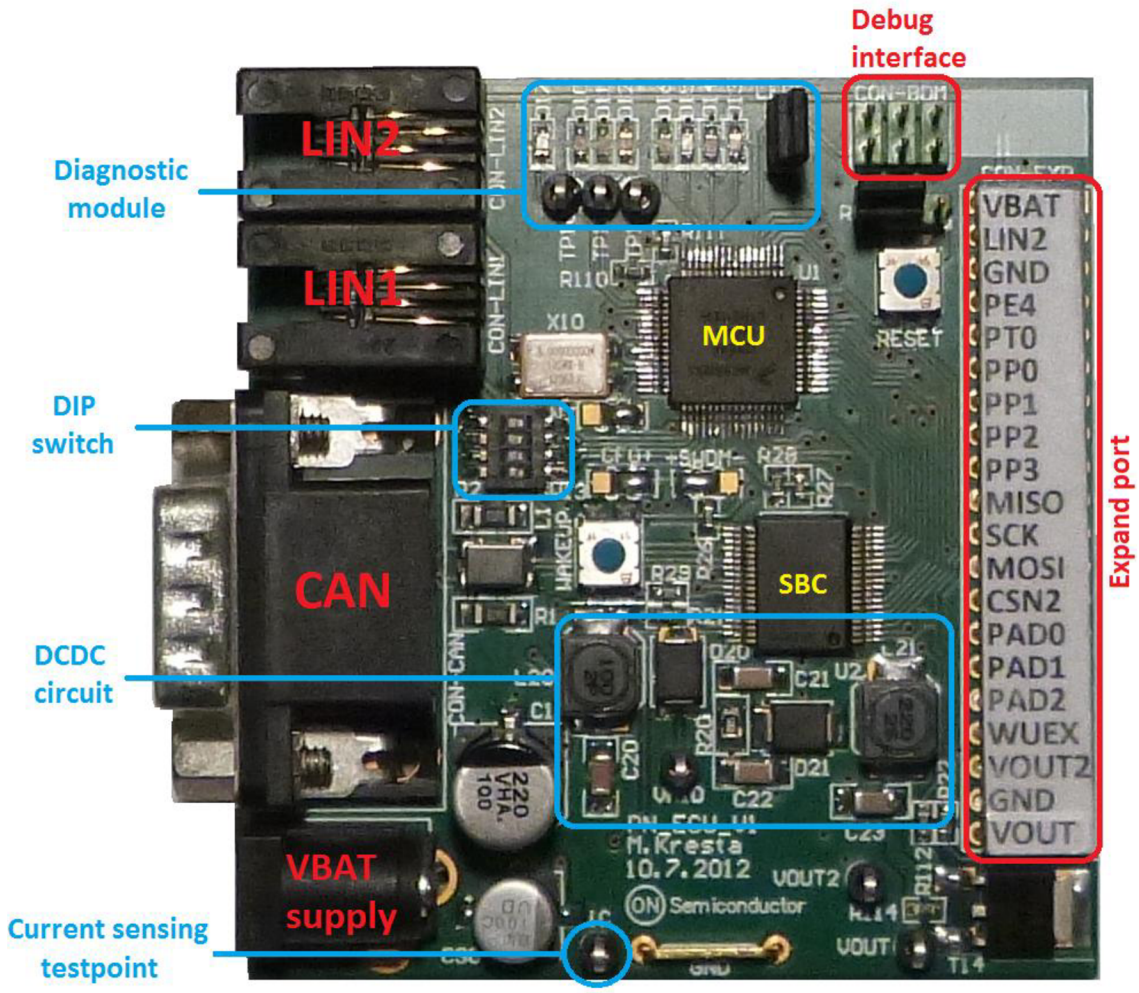


Figure 10 - Photography of assembled ECU board with marked main parts

Placement of the main components is obvious from figure 10. Along the left edge, there are two LIN connectors (RJ11), the CAN connector (D-SUB) and the power supply connector (power jack). On the opposite, right edge, there is the expanding port socket. The diagnostic LEDs are placed on the top side of PCB together with the jumper for their deactivation and three testpoints. These testpoints are dedicated for connection of logic analyzer probes. This allows evaluating the timing of transitions between different operational modes. The current sensing testpoint on the bottom edge of the board is then dedicated for oscilloscopic probe. The blocking capacitors of MCU supply pins are assembled on the bottom side. This allows placing them directly under the particular pins which are blocked.

With respect to the EMC, there are two crucial parts regarding the layout design. Firstly, the CANH and CANL tracks connecting the CAN connector and CAN transceiver integrated in SBC should be symmetrical, routed close together and have the same length if possible. By adjusting position of CAN connector and SBC these requirements were met (see appendix 2, top layer layout). The second crucial part is

layout of DC/DC converter and associated tracks. The designed layout is based on recommendation given in [16] (see figure 11). The most critical tracks conducting high currents are artificially extended by copper polygons. The vias placed under the SBC package helps to sink the device heat. Two additional testpoints added to nodes VMID and VOUT allow evaluation of DC/DC converter function.

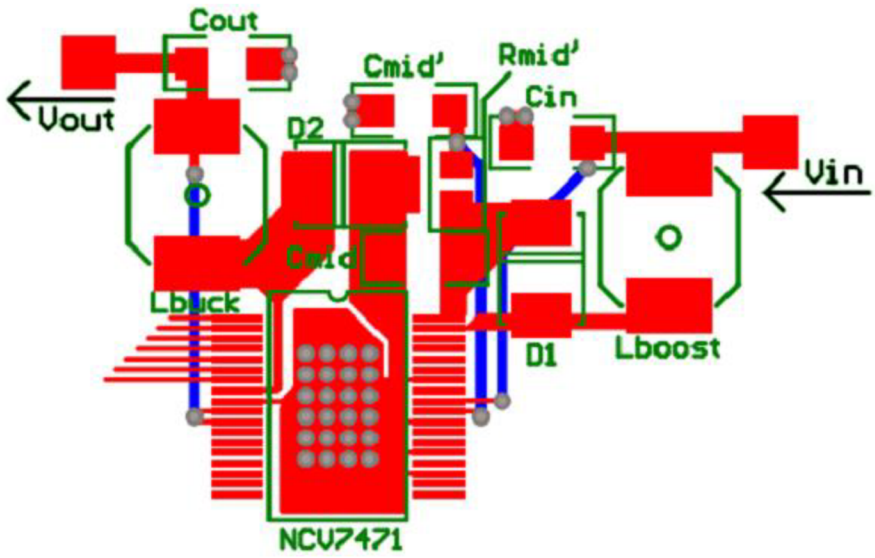


Figure 11 - Recommended PCB layout of DCDC converter [16]

5 SOFTWARE

This chapter describes individual software modules, their functionality as well as respective location of these modules in AUTOSAR architecture. The details about implementation can be found directly in commented source codes. Further, the used development tools are briefly introduced.

5.1 Software development process

The software modules belonging to lower layers of AUTOSAR Basic Software are implemented according to available AUTOSAR specifications, release version 4.0. Since no AUTOSAR tools were available for development of these drivers, all the code is written manually. Individual drivers offer different levels of functionality and configurability defined by AUTOSAR. Especially link-time configurability[2] is limited and available only for selected driver modules.

Other modules belonging to the Services and Application layer are implemented with respect to needs of actual application, without any impact of AUTOSAR standard.

All the software modules were created in IDE CodeWarrior Development Studio version 5.9.0 for the S12(X) core MCUs. For code downloading and on-chip debugging, P&E USB BDM MULTILINK [19] serial programmer was used.

5.2 Final software architecture

The original concept of software architecture as presented in the chapter 3.4 was further modified during the software development phase. In the original concept, only BSW modules necessary for the abstraction of NCV 7471 device were designed, and all other functionality was covered by an Evaluation software (see figure 7). However, for better code readability and maintainability it is an advantage to implement some additional BSW modules, especially those belonging to Microcontroller Abstraction Layer. This enhancement of the MCAL layer makes Evaluation Application fully independent on MCU hardware. Further a module of Selective Wakeup Manager was added. This module covers all functionality regarding the Partial Networking and selective wakeup. The Selective Wakeup Manager stretches from Services layer down to MCAL, as the direct access to MCU's hardware is necessary due to timing requirements. This kind of multilayer BSW module is allowed in the AUTOSAR architecture as so called Complex driver [10]. The final architecture of developed software is depicted in figure 12.

[2] An AUTOSAR configuration class; the data can be configured after the software module has been compiled [9].

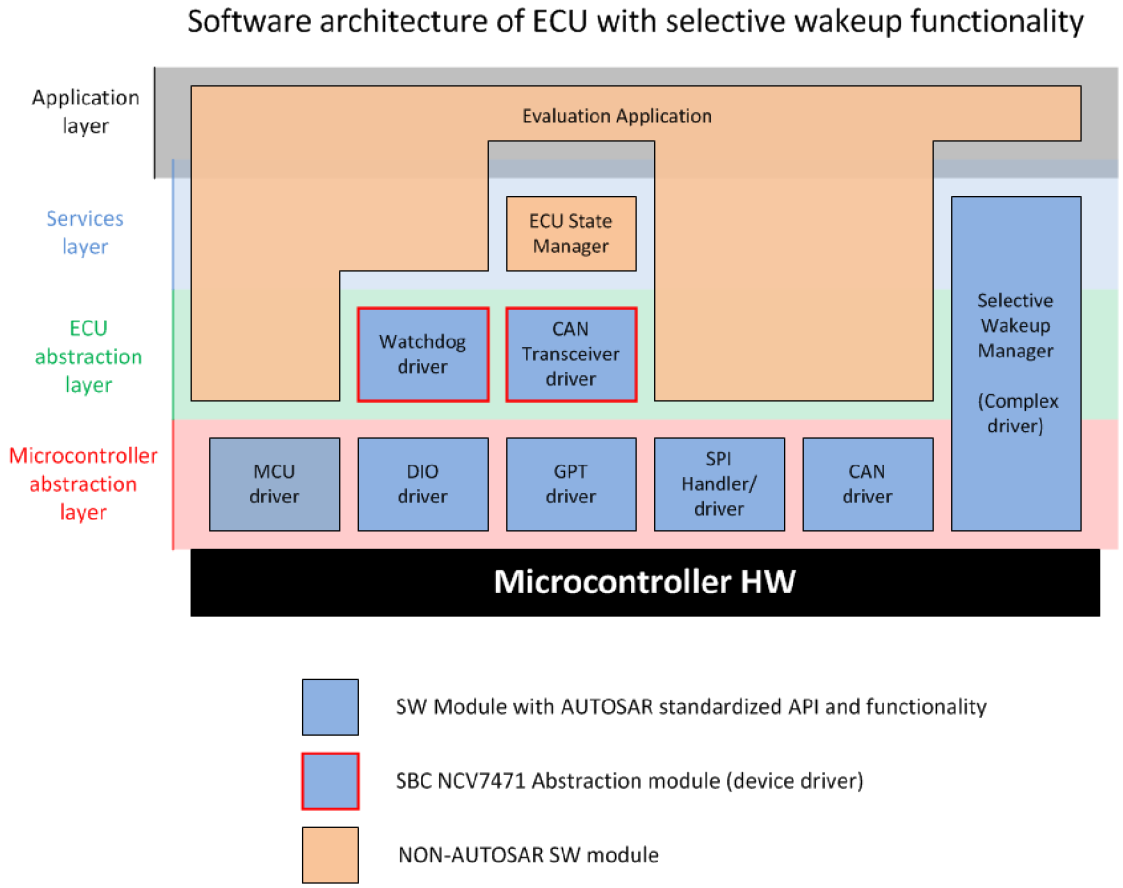


Figure 12 - The architecture of realized software

5.3 MCU Driver

The MCU Driver belonging to the lowest layer of the software architecture is freely inspired by AUTOSAR specification [21] and adjusted to specific application. The driver's main purpose is to provide an API for setting MCU operational mode as well as for complete system clock configuration. Further, the driver takes care about necessary POR configuration. Additionally, the driver is also responsible for basic IO ports configuration[3] and thus partially covers functionality of not implemented PORT Driver. The whole module consists of *Mcu.c* and *Mcu.h* files, thus no AUTOSAR configurability is supported.

The *Mcu_SetMode* API supports all operational modes of MC9S12XS microcontroller (run, wait, stop and pseudo stop). Before changing operational mode, the driver automatically configures also system clocks and IO ports, depending on the requested operational mode.

[3] The IO ports are configured for concrete application and no API for IO control is provided.

The function for configuration of the system clock *Mcu_InitClock* then allows select external oscillator or internal PLL circuit as a source of the system clock. In the first case, the bus clock *FBUS* is equal to half of a crystal frequency. In second case, IPLL module is configured to generate 20MHz FBUS clock.

5.4 DIO Driver

Another BSW module in MCAL layer is the DIO Driver. The implementation is made according to AUTOSAR specification [14] and provides APIs for read/write operations of MCU inputs/outputs. The six different functions allow read/ write access to the single pins, to the whole 8-bit ports or to the groups of adjacent pins within one port. In the AUTOSAR terminology those three access types are called channels, ports and channel groups respectively. The module is created by standard source and header file *Dio.c*, *Dio.h* and by configuration files *Dio_Cfg.h* and *Dio_Lcfg.c*. Those configuration files allow assigning pin's symbolic names and defining Channel groups with respect to specific application hardware.

As required by specification [14], all implemented APIs use numerical IDs for identification of individual channels and ports. Channel groups are then defined by structure containing ID of port to which particular group belongs, bit mask and offset of the group within the port. This approach using numerical IDs is necessary for efficient abstraction of IO access for higher layers.

5.5 GPT Driver

The General Purpose Timer Driver for control of MCU's hardware timers is located in MCAL layer and abstracts access to PIT (Periodic Interrupt Timer) hardware module. The module is based on AUTOSAR specification [22], nevertheless only functionality required by application is implemented. The driver consisting of *Gpt.c*, *Gpt.h*, and *Gpt_Cfg.h* offers no link-time configurability. All necessary settings of PIT module are fixed or run-time configurable. File *Gpt_Cfg.h* is used for assigning of symbolic names to individual timer channels. The provided API allows starting/stopping particular timer channel, configure its period and enable interrupts independently for each channel. Further the interrupt callback function can be configured in run-time.

The function *Gpt_Init* turns on the PIT module, and configures time bases for clocking of incremental timer registers. Channels PIT0 and PIT1 have fixed time base of 1us and channels PIT2 and PIT3 of 10us.

Calling of *Gpt_StartTimer* then configures period of 16-bit timer as a multiple of its time base and starts particular timer channel. The timer then periodically generates interrupts if enabled, until the channel is stopped by *Gpt_StopTimer* API, or whole PIT module is disabled by *Gpt_DeInit*.

The non-standardized API *Gpt_SetCbkPointer* then allows assigning of a function pointer to each timer channel. This callback function is then called from respective interrupt routine.

5.6 SPI Handler/Driver

The SPI Handler/Driver is one of the most complex modules within the MCAL layer. The aim of this driver is to abstract the MCU's SPI hardware for higher software layers. In our application, the API of this driver is used by CAN Transceiver Driver and Watchdog Driver. The AUTOSAR specification [13] defines three different levels of scalable functionality to meet different application needs. Because asynchronous transmission and prioritization is not required by mentioned higher-layer drivers, the Simple Synchronous SPI Handler/Driver (LEVEL0) is implemented. For buffering of incoming and outcoming messages, so-called External Buffers are used, which is a suggested approach for controlling of complex HW chips [13]. Beside the standard source and header files (*Spi.c*, *Spi.h*), module uses two configuration files *Spi_Cfg.h* and *Spi_Lcfg.c* which offers full link-time configurability. This allows setting different parameters of SPI transmission for multiple external devices as well as grouping of related SPI frames for their atomic transmission. For the configuration and grouping, concept of SPI Sequences, Jobs and Channels is used. This concept is well described in [13]. The key APIs are *Spi_SetupEB* and *Spi_SynchTransmit*.

The *Spi_SetupEB* sets External Buffer for particular SPI channel. The source and destination pointers as well as length of the buffer are passed as function arguments. The function stores these values into global variables of SPI module which are then used during the transmission.

The *Spi_SynchTransmit* then performs transmission of one configured SPI Sequence whose ID is passed as only function argument. Depending on configuration, the function sequentially sends all SPI Jobs contained within the sequence. Because every Job can be targeted to different external device, *Spi_SynchTransmit* function should configure SPI hardware every time before the start of new Job transmission. To improve the performance, this configuration is done only if new Job is targeted to different external device then the previous one, or if the MCU system clock was changed since last SPI transmission[4]. Within the Job are sequentially sent individual SPI Channels which may consist of one or more SPI frame, given by the length of particular external buffer. Handling of CS signal depends on configuration of actual external device and is done either automatically by SPI hardware or manually by software control of particular digital IO pin.

[4] When the MCU system clock frequency is changed, the SPI baudrate generator has to be reconfigured to keep desired SPI baudrate.

The driver is configured by the Spi_Lcfg.c file for the purposes of specific application. The only external device accessed via SPI is the NCV7471 which is configured to communication baudrate of 1Mbps and automatic CS signal handling. Further one SPI Sequence, whose symbolic name is used by higher layers for accessing NCV 7471 device. This sequence contains only one SPI Job containing one SPI Channel. This configuration is the most time-efficient and simple solution. Since the configured Channel utilizes External Buffer with variable length, one synchronous atomic SPI transmission can consist of many SPI frames, efficiently configuring all SBC functionalities, as well as of just one frame for triggering of NCV7471 watchdog.

5.7 CAN Driver

The last module within the MCAL layer is CAN Driver which plays important role in implementation of the selective wakeup feature. Module provides reduced set of API's standardized within AUTOSAR specification [23] for controlling of MSCAN HW. Extra, non-standardized functions are added to improve the performance of the RWUF detection process. The CAN Driver module consists of three source files (*Can.c*, *Can.h*, *Can_Lcfg.c*) offering a link-time configurability of selected parameters. Beside the initialization of MSCAN hardware, the driver allows sending and receiving of CAN messages with standard or extended identifiers. For effective handling of CAN messages, the driver utilizes hardware buffers of MSCAN module and Identifier Acceptance Filters.

The API *Can_Init* sets the baudrate and bit-timing registers as well as default Identifier Acceptance filters with respect to linked *Can_Lcfg.c* configuration file. Those settings are made in MSCAN Initialization mode. After leaving this Initialization mode, MSCAN interrupts are disabled by default.

The receiver interrupts can be enabled or disabled by *Can_SetInterrrupts* API call, independently on actual CAN Driver operational mode.

The *Can_SetOpMode* API allows transitions between four operational modes of the CAN driver. Whereas *CAN_RUN* and *CAN_SLEEP* modes represent their respective modes of MSCAN hardware, the other two modes, *CAN_LISTENONLY* and *CAN_RWUF_DETECTION* use exactly the same setting of MSCAN hardware, the listen-only mode. The sense of those two modes is only to distinguish whether the listen-only mode was requested from ECU normal mode by the application software, or whether the listen-only mode is needed by RWUF detection process.

To enable run-time configuration of the MSCAN's Identifier Acceptance Filters, the CAN driver provides a *Can_SetIdFilter* API. Because the Identifier Acceptance Filters are used for validation of RWUF frames, the run-time configurability is necessary to allow configuring RWUF frame run-time, as required by ISO 11898-6 [5].

When the MSCAN receiver is active and a received CAN message passes through the configured Identifier Acceptance Filters, a receive interrupt is generated, if enabled.

In the receive interrupt routine, all message data (ID, DLC, data, time stamp) are copied to the auxiliary SW buffer at first to release MSCAN receive buffer. Further processing of received data is done by private function *Can_FrameReceived* and depends on actual operational mode of CAN Driver. If the driver is not in *CAN_RWUF_DETECTION* mode, the data are transformed to the standardized format of structure data type *Can_PDUType* and configured callback function is called. If the driver is in *CAN_RWUF_DETECTION* mode, complete PDU doesn't have to be reconstructed from received data, which significantly shortens processing time. Only the message data (payload) and the DLC code are passed as parameters to the dedicated function of the Selective Wakeup Manager module (see figure 12).

The *Can_Write* API initiates CAN message transmissions. The API accepts the PDU and its internal priority as function parameters. The PDU is transformed to a format of MSCAN transmit message buffer and copied into one of the available transmit buffers. The internal priority is set and the buffer is scheduled for transmission. Finally, the transmitter empty interrupt of particular buffer is enabled.

5.8 CAN Transceiver Driver

As stated in AUTOSAR specification of CAN Transceiver Driver [11], the driver is responsible for abstraction of external CAN transceiver or of SBC containing CAN transceiver. In our application, this driver controls external SBC NCV7471 connected to the MCU by SPI and digital lines. The implemented driver takes control over all NCV7471 features, except watchdog triggering, which is assured by Watchdog Driver (chapter 5.9). For access to the physical interfaces, SPI Handler/Driver and DIO driver from underlying SW layer are used. The CAN Transceiver Driver consisting of four source files (*CanTrcv.c*, *CanTrcv.h*, *CanTrcv_Lcfg.c*, *CanTrcv_Cfg.h*) implements most of the APIs defined by AUTOSAR specification [11] and allows basic link-time configurability. The whole driver uses only the AUTOSAR standardized APIs of the underlying MCAL drivers which allows easy portability to various MCU platforms.

Since the CAN Transceiver Driver uses API *Spi_SyncTransmit* of underlying SPI driver, it is also responsible for allocation and setting of particular SPI External Buffer by *Spi_SetupEB* API (see chapter 5.6). This External Buffer, realized as one-dimensional array of 16-bit words, is used for write/read operations to/from NCV7471 SPI registers. The length of the buffer determines the number of consecutively sent SPI frames. For different operations, buffers of length 1 to 3 SPI frames are used.

The key functionality of the CAN Transceiver Driver is the *CanTrcv_SetOpMode* API. This function sets the NCV7471 into one of four different configurations, by accessing SPI registers Control0, Control1 and Status1. Particular configurations of those four modes are presented in table 2.

CanTrcv operational mode	Normal	Standby	Sleep	PN RWUF
NCV7471 mode	Normal	Standby	Sleep	Standby
Watchdog mode (period)	Time-out (256ms)	Off	Off	Off
LDO VOUT2	On	Off	Off	Off
CAN	Normal	Wakeup	Wakeup	Receive-only
LIN1	On	Off	Off	Wakeup
LIN2	On	Off	Off	Off
Local wakeup	Off	Falling edge	Falling edge	Falling edge

Table 2 – Configurations of NCV7471 device supported by *CanTrcv_SetOpMode* API

5.9 Watchdog driver

Watchdog Driver is second BSW module in the ECU Abstraction Layer. The task of this simple driver is to configure and trigger the external watchdog as a part of system basis chip NCV7471. The module implements APIs described in AUTOSAR specification [12] and for the sake of simplicity is the configuration of whole module fixed. Thus this MSW module consists of only two source files *Wdg.c* and *Wdg.h*. The NCV7471 device is accessed via SPI interface in the same way as by CAN Transceiver Driver.

The *Wdg_SetMode* API currently supports three different configurations of external watchdog. *WDG_OFF_MODE* represents, as name indicates, deactivated watchdog. This mode is used in all ECU low power modes. In ECU normal mode, then watchdog is set to *WDG_SLOW_MODE*, which configures watchdog to the time-out mode with period of 256ms. Beside the configuration of watchdog itself, the function also starts a dedicated GPT channel *WDG_GPT*. The interrupt of this periodic timer with 100ms period is then used for watchdog triggering. Last supported mode is *WDG_TIMER_MODE*, which utilizes external watchdog as a low power external timer, generating interrupts with 256ms period. This interrupts are signalized to the MCU via INTN pin.

If the application software runs correctly, it has to periodically set so-called trigger condition by call of *Wdg_SetTriggerCondition* API. The trigger condition determines whether the watchdog should be triggered or not. After a call of mentioned API, trigger condition is valid for certain time interval, which is given as API's parameter. It is the responsibility of the application software to decide how long this interval is.

Wdg_Cbk_GptNotification is a callback function called by the interrupt routine of GPT channel *WDG_GPT* interrupt. The function checks the trigger condition, and if this is valid, the watchdog is triggered by SPI sequence.

5.10 ECU State Manager

The ECU State Manager is the only software module belonging into the Services Layer. The main task of this module is overall configuration of particular ECU features in dependence on actual ECU's operational mode. The module is dedicated to specific application (PN ECU) and covers configuration of MCU, NCV7471 (including CAN, LIN, DC/DC, LDO, Watchdog) and other ECU hardware (powering of expanding port). The ECU State Manager, which is not based on any AUTOSAR specification, is realized by *EcuM.c* and *EcuM.h* source files. Basically, the APIs of the ECU State manager are provided for application software, however when the Partial Networking feature is enabled, the module also interacts with the Selective Wakeup Manager.

The first of two most important functions is *EcuSetMode*. This API takes a required ECU operational mode as an argument, and base on that, configures the ECU, using APIs of the underlying BSW modules. The *EcuSetMode* API supports only two ECU modes which, can be requested by application software. The ECU_NORMAL mode represents full ECU functionality, while ECU_SLEEP mode is the low-power mode. When the ECU_SLEEP mode is requested, current configuration of the ECU depends on the Partial Networking setting. If the PN is disabled, the ECU enters Standard Sleep mode (NCV7471 is configured to sleep mode and MCU became unpowered). If the PN feature is enabled, then the new operational mode is Fast-startup sleep mode (MCU stays powered). For detail configuration in those modes see table Table 4.

The second important function is *EcuMain*, which have to be periodically called during run-time. In our simple application, this function is called within every cycle of the main program loop. However, in the more complex applications it can be also called as a scheduled task of the real-time operational system. The *EcuMain* function checks software flags, which can be set by various asynchronous events, or are valid during certain operational modes. Depending on the state of those binary software flags, appropriate actions are performed afterwards.

5.11 Selective Wakeup Manager

The Selective Wakeup Manager module implements selective wakeup functionality defined by ISO 11898-6 and provides APIs for its configuration. In order to meet timing requirements of selective wakeup process, the module requires direct access to MCU HW, especially to MSCAN and SPI modules. Based on that, the Selective Wakeup Manager should belong into MCAL layer. However, the module is also dependent on the external CAN Transceiver (ECU Abstraction Layer), and provides configuration APIs belonging to Services Layer. To handle this extended functionality, the module is realized as multi-layer Complex Driver reaching from the MCAL up to the Services layer. Since the software realization of selective wakeup capability is special, not standardized function, the Selective Wakeup Manager is not based on any

AUTOSAR specification. Implementation is highly dependent on used MCU (MC9S12XS) and SBC (NCV7471).

In order to mimic the behavior of ISO 11898-6 CAN Transceiver, the module defines global variables which substitute its SPI registers, and APIs substituting SPI write/read operations. Those global variables store configurable data (RWUF, RWUF ID mask, CFGSWU) and read-only flags (RWUP, RWUF, LWU, CFGERR, CANTO, FDERR) defined in ISO 11898-6 [5]. APIs *Pn_SetCFGSWU*, *Pn_SetRwuf*, *Pn_SetIDMask* and *Pn_SetRWUFPayload* are dedicated for use by application software, as well as other simple APIs for reading flags. Those APIs only interfaces the mentioned global variables but do not change current setting of HW modules.

The other functions are involved in the process of selective wakeup itself. For better understanding of further text, this process is described by state/flow diagram in figure 13.

Pn_GoToRwupDetection API is called by ECU Manager, before the Fast-startup sleep mode is entered[5]. The function checks whether the RWUF is correctly configured and if so, it sets the ID acceptance filters of MSCAN module. The acceptance filters are configured as two 32-bit filters, allowing usage of extended identifiers. The filter 0 is set to accept only IDs matching ID of configured RWUF, while the filter 1 accepts all other IDs. After setting ID acceptance filters, is the MSCAN module configured to sleep mode and WUP detection is disabled. This means, that just configured ID acceptance filters are not effectively used. However the setting of filters is preserved during sleep mode and used later in RWUF detection mode.

When CAN wakeup pattern is detected (blue arrow in figure 13) The ECU Manager configures ECU into RWUF detection mode and calls *Pn_EcuWakeup* API. This function sets MSCAN to listen-only mode and starts “Delay-to-wait” timer (black arrow)[6]. *Pre-Wait mode delay* optimizes overall wakeup time, in case that RWUP is immediately followed by a stream of RWUFs, which is the fastest possible wakeup scenario.

When “Delay-to-wait” timer elapses (gray arrow), The *RWUF detection wait mode A* is entered and external watchdog timer is started (256ms). In this mode, the MCU is in wait mode and MSCAN accepts all valid frames (through filter 0 + filter 1).

If MSCAN receives non-RWUF valid CAN frame (brown arrow), the CAN driver calls API *Pn_Cbk_ValidFrameDetected*. The *RWUF detection wait mode B* is entered and external watchdog timer is restarted. Further, the function closes the ID acceptance filter 1 which means that further are accepted only IDs matching configured RWUF by filter 0. This comprehensive usage of acceptance filters in RWUF detection mode

[5] The Fast-startup sleep mode, is low power mode with enabled selective wakeup functionality (see table 3)

[6] The “Delay-to-wait” timer is ca 1ms long software delay.

minimizes the time when MCU is in the run mode, which reduces average current consumption.

The ISO 11898-6 requires that if there is no CAN communication for longer than t_{TOCAN}, the RWUF detection mode is left and Fast-startup sleep mode is entered again. In our application, this t_{TOCAN} interval is defined by external watchdog timer (part of NCV7471) with 256ms period. When this timer expires (purple arrows), the *Pn_TimeoutInt* API is called from INTN ISR. Depending on current state, ECU goes back to the Fast-startup sleep mode, or enters *RWUF detection wait mode A* (the acceptance filter 1 is opened to accept all IDs). This realization implies that effective

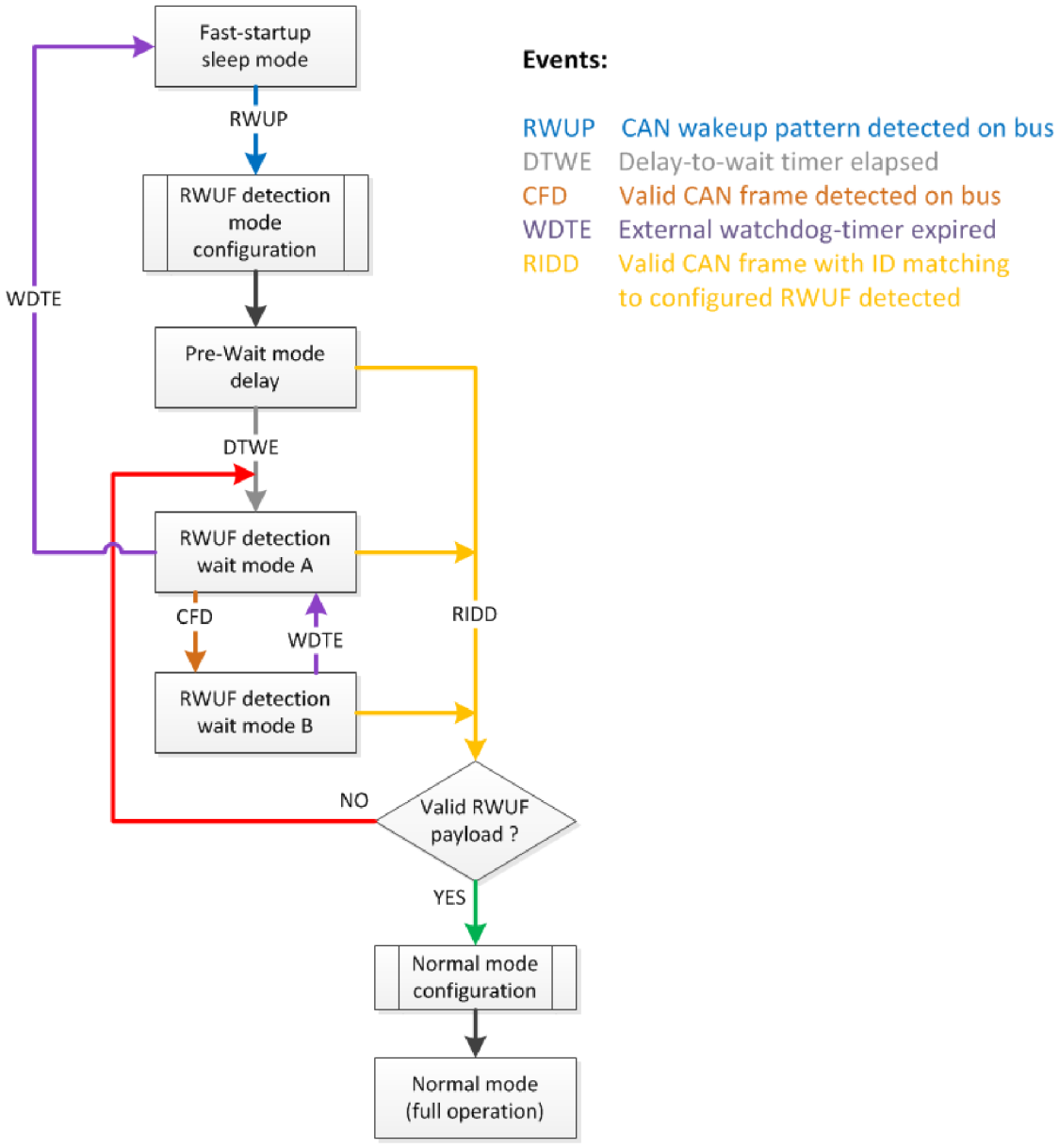


Figure 13 - State/flow diagram of selective wakeup process

t_{TOCAN} delay can vary in range from 256ms to 512ms, which is compliant with ISO 11898-6 requirement t_{TOCAN} < 750ms [5].

Anytime, when in the RWUF detection mode (including *Pre-Wait mode delay, RWUF detection wait mode A* and *RWUF detection wait mode B*), the ID acceptance filter 0 accepts ID of configured RWUF, the API *Pn_Cbk_RwufIdDetected* is called by the CAN driver. This function compares the data field of received frame and the payload of configured RWUF frame. If received frame is evaluated as a valid RWUF, the ECU is woken up and enters Normal mode. Otherwise the process stays in the RWUF detection mode.

The process of evaluation of potential RWUF frames is thoroughly described in standard specification [5]. On each received valid CAN frame, three evaluation checks concerning ID, DLC and data payload are sequentially applied. If all those checks match the configured RWUF, the ECU is woken up. The ID is concerned as matching if all ID bits exactly matches configured ID, except those bits which are configured as "do not care" by ID mask register (see figure 14). This check is done by MSCAN hardware utilizing ID acceptance filter 0. The other two checks are done by software within

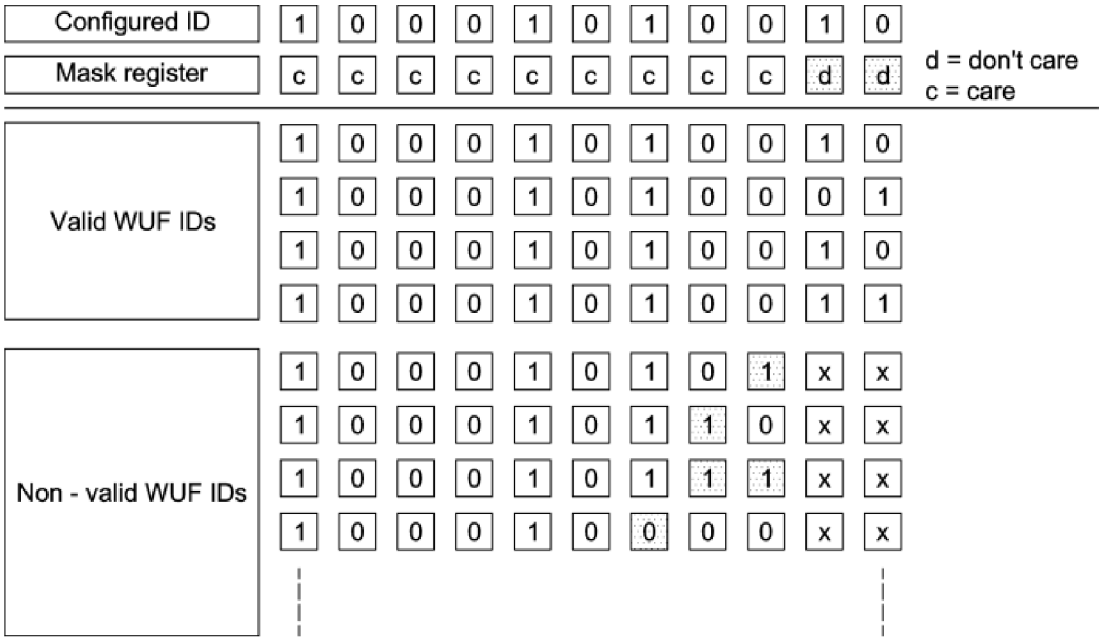


Figure 14 - Example of ID masking mechanism[25]

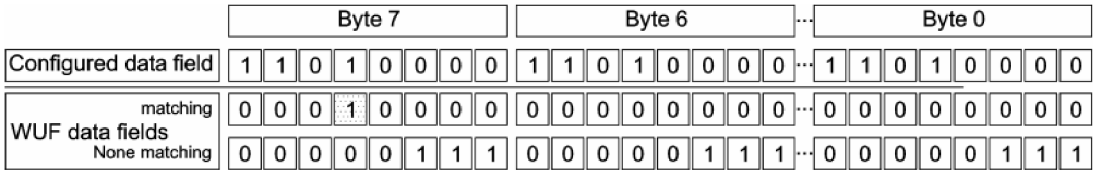


Figure 15 - Example of data field evaluation mechanism [25]

function *Pn_Cbk_RwufIdDetected*. The received frame is considered as valid RWUF if its DLC equals to DLC of configured RWUF and if at least one log 1 bit within the data field matches to configured RWUF payload (see figure 15).

5.12 Evaluation Application

The uppermost module in the AUTOSAR hierarchy is the Evaluation Application. With respect to the AUTOSAR SW architecture, this module should belong only to the Application layer and use only APIs provided by modules of Services Layer. However, because of case of presented SW solution some modules of standardized BSW stack are omitted, the Evaluation Application is realized as multilayer module with direct access to lower layers (see Figure 12). The Evaluation Application module coded in the source files *App.c* and *App.h* is not based on any AUTOSAR standard.

Function *App_Init* is called after the power-on-reset event (POR) of the MCU. This function initializes the network node address to value defined by 4-pin DIP switch. Those hardware switches allow assigning unique address for up to 16 ECUs without any need for change of software. Further, the initialization function configures default RWUF frame and enables selective wakeup functionality. The default RWUF frame have predefined ID (*02x* hex)[7] and two-byte long data field. The data field contains just one bit set to one (log 1). Position of this bit is given by node address.

The other functions of the Evaluation Application module decodes received CAN messages and eventually, based on frame ID, perform set of predefined actions. The collection of all supported IDs is in table 3, other IDs are ignored. All 12 supported frames use standard identifier. The first 7 most significant bits (11..4) of 11-bit identifier identifies the type of message. The remaining four least significant bits (3..0) are reserved for node address of transmitting node. This ensures that two or more nodes

ID (hex)	DLC	D0	D1	D2	D3	D4	D5	D6	D7	Description
01x	1	WUR								Confirmation of wake up + wakeup reason
02x	2	wake mask								Default RWUF frame
04x	2	CNAD	SWE							Enable/disable selective wakeup feature
05x	2	sleep mask								Go to sleep/selective sleep
12x	6	CNAD	EIF	ID (uint 32)						Configure RWUF ID
13x	6	CNAD	EIF	ID mask (uint 32)						Configure RWUF ID mask
14x	0-8	RWUF payload								Configure payload of RWUF frame
15x	1	CNAD								Select Node for RWUF payload configuration
17x	3	CNAD	NUT	NOR						Perform evaluation test A (wakeup time)
18x	3	CNAD	NUT	NOR						Perform evaluation test B (100% busload)
19x	3	CNAD	NUT	NOR						Perform evaluation test C (100% busload with RWUF IDs)
20x	3	CNAD	NUT	NOR						Perform evaluation test D (WU with extended ID RWUF)

Table 3 - Collection of supported CAN IDs

[7] The small letter "x" in ID number represents "don't care" value.

can never transmit message with same ID, which is necessary for correct arbitration process [24]. The messages which are targeted to specific single node (consumer) contain the consumer's node address (CNAD) in first byte of data field[8]. The remote sleep request message (ID *05x*) is a broadcast message. This message can switch one or multiple nodes into sleep mode in a same way as RWUF frame can wake up multiple nodes.

The messages with IDs *12x* up *15x* are dedicated for remote configuration of CNAD's RWUF frame including ID, ID mask, DLC and payload. The configuration data are passed in data field, only the RWUF's DLC parameter is derived from DLC of *14x* message. The Extended Identifier Flag (EIF) determines whether the configured RWUF uses standard or extended ID.

The next four messages (*17x*, *18x*, *19x*, *20x*) perform special evaluation tests for effective measuring of wakeup time and current consumption during certain bus traffic. The CNAD is here node address of node which will perform the test, while the abbreviation NUT stands for Node Under Test. The node performing the test firstly remotely configures RWUF on node under test and remotely puts this node to sleep mode. After two seconds delay, the node performing the test trigs the oscilloscope and starts to consecutively send certain CAN frames. The number of sent frames is given by parameter NOR (Number Of Repetitions).

5.13 Main

The source file *main.c* is very simple piece of code which after POR calls APIs for MCU and ECU initialization and than in main program loop alternately calls *Ecu_Main* and *App_Main* functions. In the AUTOSAR SW architecture, those tasks are covered by scheduler of real-time operational system, but for evaluation purposes is realized solution absolutely sufficient.

[8] Since the message with ID 14x cannot contain CNAD byte, the special message with ID 15x have to be sent prior to ID 14x, to select consumer node.

6 EVALUATION PROCESS AND RESULTS

The chapter describes different evaluation tests performed on realized ECU and presents their results. The achieved level of compliancy with ISO 11898-6 is examined and designed solution is compared with Partial Networking fully compliant to ISO 11898-6 standard [25] and with Pretended Networking method.

6.1 PC Application CAN Interface

For easy testing of developed ECUs connected in CAN network, PC application CAN Interface and special hardware creating physical interface between USB and CAN protocol were used. Both, the PC application and HW interface are part of laboratory equipment in ON semiconductor Brno Design Centrum. The figure 16 shows the CAN Interface application with example of transmitted and received CAN messages.

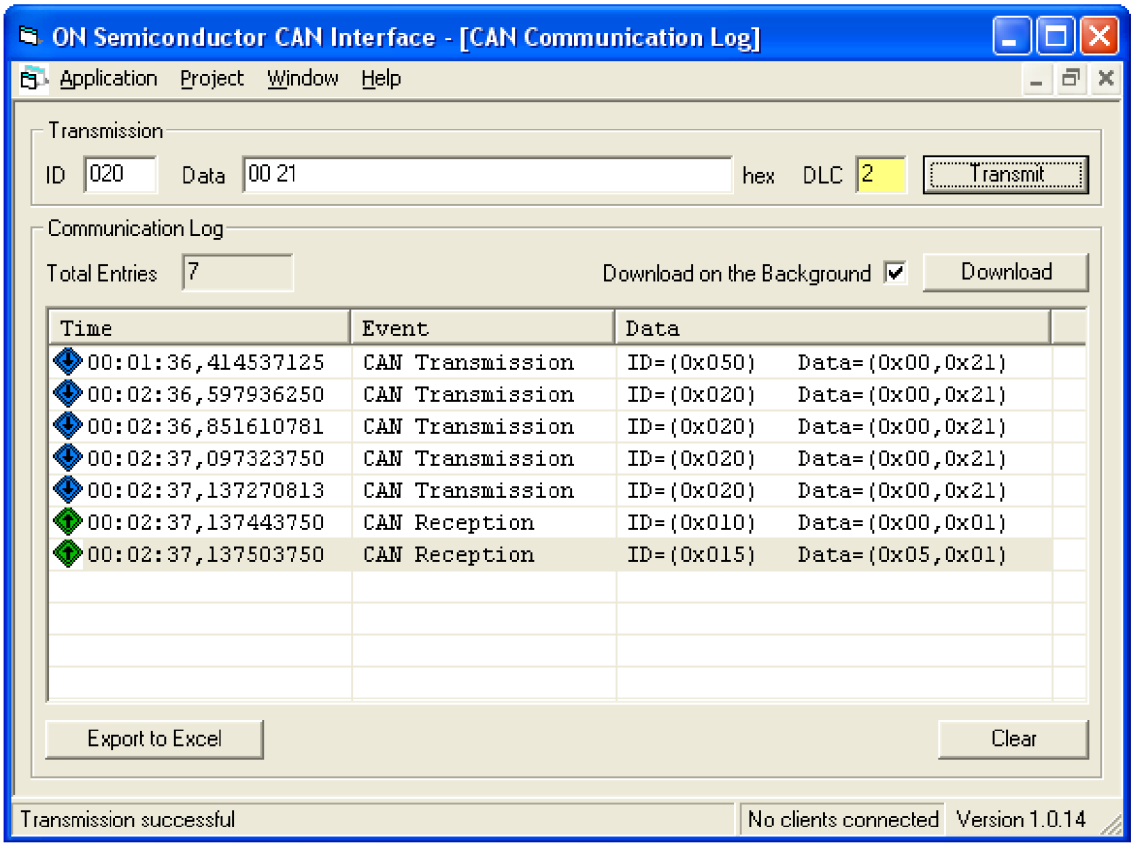


Figure 16 - PC application CAN Interface used for evaluation of developed ECUs

6.2 Static current measurements

The current consumption in individual ECU operational modes was measured by the laboratory source-meter. Using of precise laboratory equipment for measuring of steady currents offers better accuracy than on-board current sensing circuit. The source-meter, supplying whole ECU, is connected to VBAT connector and measures the current flowing through this connector. Because the current consumption of current sensing circuit is out of our interest, the integrated circuit INA210 was disassembled during those static measurements. Thus, the measured current reflects consumption of SBC NCV7471 and all components supplied by its 5V VOUT voltage (including MCU), which emulates real ECU. The flow of measured current through the current sensing circuit is visualized by a green arrow in figure 17. The current consumption in all operational modes is reasonably low and 1Ω sensing resistor will not affect the stability of the DC/DC converter. Thus the bypassing MOSFET transistor is in the off state during all the measurements. The high-voltage[9] inputs of NCV7471 (CFG and SWDM) are connected to ground to avoid pull-down currents.

The table 4 presents the achieved current consumptions in four ECU operational modes. Those values represent currents drawn from 12V supply, which is the typical car battery voltage. The frequency of used MCU oscillator is 4MHz, instead of originally proposed 8MHz (see table 1), which further reduces overall current consumption. The oscillator frequency 4MHz is minimal value to support 500kbps CAN baudrate, which is required by ISO 11898-6 standard. The original setup with 8MHz oscillator, allowing maximal CAN baudrate of 1Mbps, was also evaluated and results can be found in appendix 3.

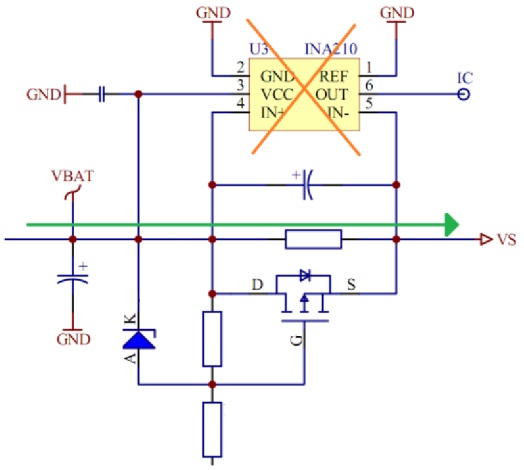


Figure 17 - Current sensing circuit in configuration for static measurements

[9] Digital inputs which tolerate battery voltage level (max 28V). Those pins are equipped with internal pull-down resistors.

Table 4 - ECU Operational modes with corresponding static current consumptions

ECU mode Designed ECU power mode	**SBC NCV7471** Configuration of NCV 7471	**MCU Freescale MC9S12XS** Configuration of MCU	**HW peripherals** power status of ECU functional peripherals (not assembled on evaluation ECU)	**ISO 11898-6** Current consumption of PN CAN transceiver in corresponding mode (required by ISO 11898-6)	**Current cons.** Current consumption from VBAT (12V), measured on evaluation ECU (NCV7471 + MCU)
Standard sleep mode PN functionality disabled RWUP detection	Sleep mode CAN - wakeup VOUT - off	Unpowered	Unpowered	-	**60µA** (T_A = 25°C)
Fast-startup sleep mode PN functionality enabled RWUP detection	Standby mode CAN - wakeup VOUT - on Watchdog - off	Pseudo stop mode Oscillator 4 MHz Start-up time about 3,5µs	Unpowered	30µA[1)]	**235µA** NCV7471 ≈ 100µA MCU ≈ 135µA (T_A = 25°C)
RWUF detection mode PN functionality enabled RWUF detection	Standby mode CAN - receiveonly VOUT - on Watchdog - off	Wait (Run) mode SYSCLK from oscillator (4 MHz) CAN - listenonly PLL - disabled	Unpowered	500µA[1)]	**3.4mA (4.2mA)**[2)] NCV7471 ≈ 2mA MCU ≈ 1.4mA
Normal mode Full functionality of ECU	Normal mode CAN - normal VOUT - on Watchdog - on VOUT2,LIN - by SPI	Run mode PPL used for SYSCLK (20 MHz) CAN - normal	Powered (Current consumption up to 450mA)	-	**12mA**[3)] + consumption of ECU function hardware

Note 1) : At this moment there is not known any CAN transceiver which meets this requirement (1/2013)

Note 2) : MCU in Run mode (4.2mA) only when RWUF with matching ID is detected. After decoding of RWUF data, MCU goes back to Wait mode (3.4mA)

Note 3) : Measured with CAN bus idle

6.3 Dynamic current measurements

Besides the static current consumptions, time-varying consumption during transitions between individual operational modes was evaluated as well. CAN bus events needed for those tests are generated by one of the nodes connected in the network, further called as test master. The oscilloscope then captures current consumption and other digital signals measured on another node connected in network, called as Node Under Test (NUT). The fixed test scenarios are implemented as part of Evaluation Application software (see chapter 5.12). The individual tests are initiated manually using PC application CAN Interface. To allow oscilloscopic measurements of the current consumed by the ECU, on-board current sensing circuit is used. The IC INA210 amplifies voltage drop on the sensing resistor, as shown in figure 18. The finite input resistance of input pin IN- causes additive measurement error by increasing of current flowing through the sensing resistor (red arrow). Nevertheless, this additive error (ca 30µA) is not critical for evaluation of dynamic behavior.

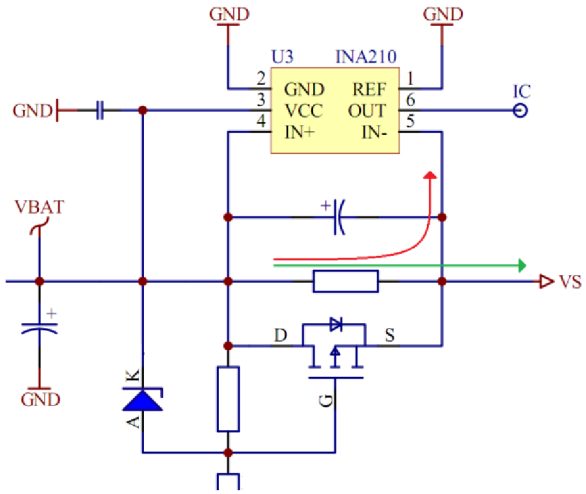


Figure 18 - Current sensing circuit in configuration for dynamic measurements

The first test simulates the fastest possible selective wakeup scenario. The test also proves that ECU detects fourth RWUF correctly. ISO 11898-6 specification requires that maximally first five CAN frames, following RWUP, can be ignored [5]. The result of this test is shown in figure 19. Initially, the NUT is in Fast-startup sleep mode. At time t = 0, test master starts transmitting of four identical RWUF frames, with minimal inter-frame space. Those RWUFs use standard 11bit identifier and data field is empty (DLC = 0), which represent the shortest valid CAN frame. At time-point A (see figure 18), the CAN Transceiver (part of NCV7471) detects wake-up pattern (RWUP) and generates INTN interrupt for MCU. The MCU is woken by this interrupt and immediately configures CAN Transceiver and CAN controller to receive only mode. This configuration is completed at time-point B, when CAN controller starts synchronizing to the CAN bit stream. At time-point C, the CAN controller is synchronized and ready to accept CAN messages. Thus the fourth frame is received and detected as valid RWUF. The MCU sets the ECU into Normal mode configuration, which is completed at time-point D. The peak of current consumption (Ic), rising at time of 1ms, is a result of rapid change of power needs caused by switching MCU clocks to PLL (4MHz →20MHz) and by setting NCV7471 into Normal mode.

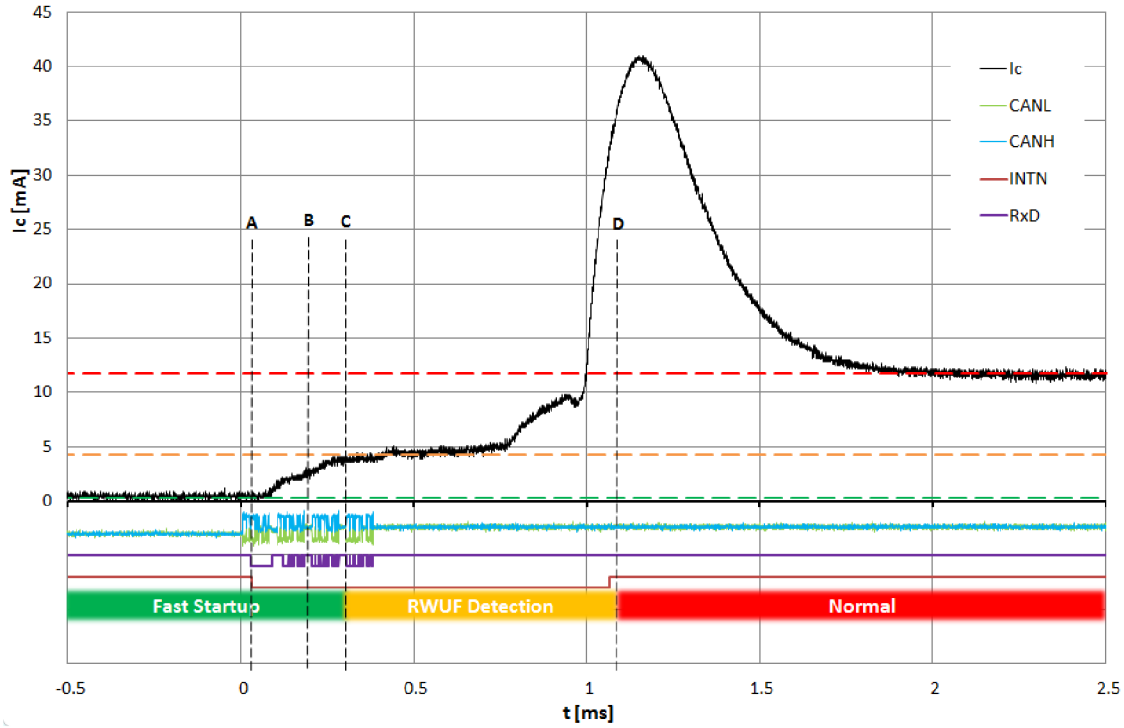


Figure 19 - The fastest possible wakeup scenario

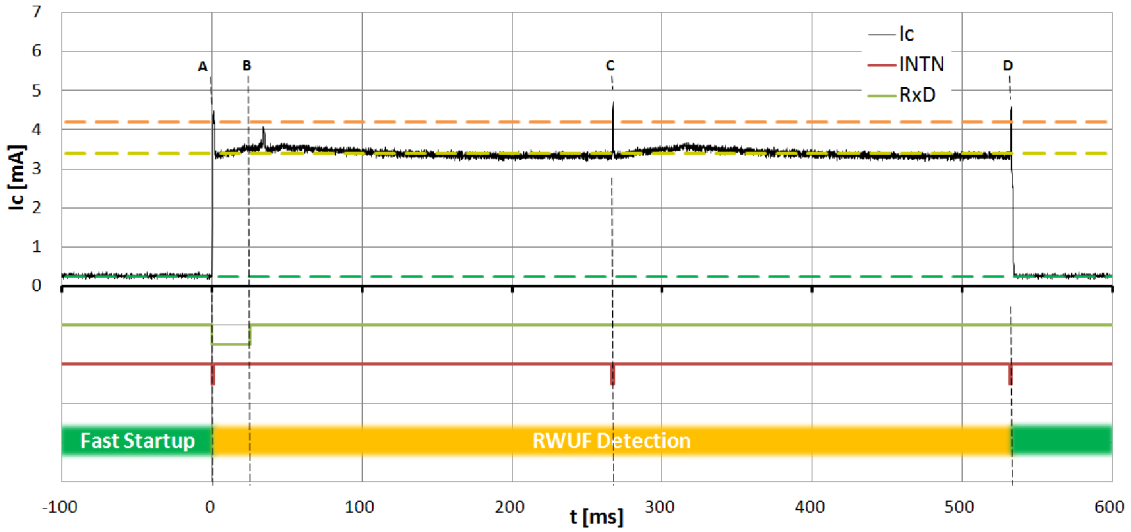


Figure 20 - The consumption in RWUF detection mode with 100% bus load (non-RWUF ID messages)

Next evaluation test captures current consumption of NUT during ongoing bus communication. The aim of this test is to show power saving efficiency in situation when some nodes are in low power selective sleep mode, while other nodes are using CAN bus for normal communication. Also the functionality of t_{TOCAN} delay is demonstrated by this test. The captured data are plotted in figure 20. As well as in

previous test, the NUT is initially in Fast-startup sleep mode. The test master starts transmitting stream of CAN messages, simulating 100% bus load. The first transmitted message is treated as a wakeup pattern (RWUP) by sleeping node (NUT). As a consequence, the NUT enters RWUF Detection mode (time-point A) and its consumption increases to 4.2mA. Since the stream of CAN messages consists of general CAN frames with ID not matching to configured RWUF, the MCU of NUT immediately goes to wait mode, reducing current consumption to 3.4mA. The wait mode is then preserved during pending communication, because CAN messages do not match with ID acceptance filter of MSCAN module. At time-point B, the bus communication ends and bus become idle. The NUT stays in RWUF Detection mode, ready to detect potential RWUF. In the state diagram in figure 13, this state is represented as RWUF detection wait mode B. After 256ms (measured from time-point A) is MCU woken up by external watchdog timer and NUT goes to RWUF detection wait mode A (see figure 13). After next 256 ms, where the bus is still idle, ECU goes back to Fast-startup sleep mode (time-point D). The effective t_{TOCAN} delay is in this example ca. 505ms (measured from time-point B to D).

Third test simulates almost same situation as previous one. The only difference is that ID of CAN messages transmitted by the test master matches RWUF ID configured in the NUT. This case may occur when there are two or more nodes in selective sleep mode and some other active node wants to wake up only a subset of these nodes, which is determined by RWUF payload. In this situation, CAN frames pass through ID acceptance filters and the MCU have to parse data payload, which results in increased current consumption in time period A-B (see figure 21). From time-point B, when the bus becomes idle, is the behavior of the NUT the same as in previous test case.

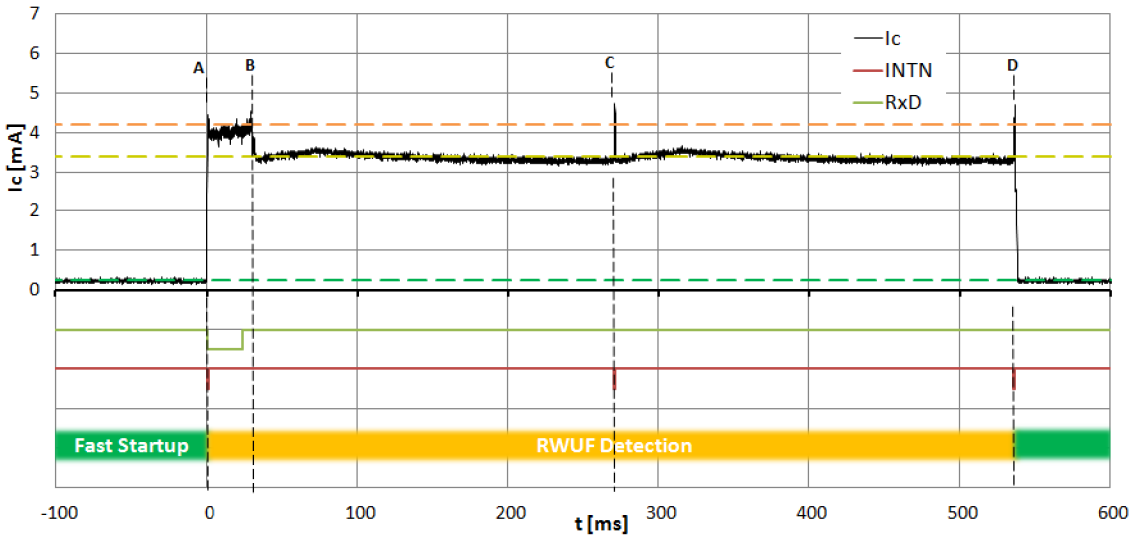


Figure 21 - The consumption in RWUF detection mode with 100% bus load (RWUF ID messages)

6.4 Deviations from ISO 11898-6 standard

One of the goals of the PN ECU project was to ultimately meet requirements specified in Hardware Requirements for Partial Networking [5] which is the document on which the ISO standard 11898-6 [25] is based on. This chapter lists the requirements with differences between presented solution and ISO11898-6 target dedicated chip design solution.

Firstly, overall ECU static current consumption in low power modes is higher than required for transceivers dedicated for partial networking (see table 4). This deviation, caused by usage of standard MCU for RWUF detection, was expected from early stages of the development process. Currently, in April of 2013, there is still no commercially available ISO 11898-6 CAN Transceiver which meets hard consumption limits during ongoing bus communication. The NXP, which is the PN CAN transceiver technology leader, claims consumption under 1mA on their PN CAN transceiver TJA1145 [26]. Consumption required by the specification is 500µA.

Other deviation concerns the processing of CAN messages in RWUF Detection mode. The MSCAN module integrated in MCU, accepts only fully valid CAN frames and all frames containing some type of frame error are ignored. However, dedicated PN CAN transceiver should ignore acknowledgement (ACK) and end-of-frame (EOF) errors, when deciding about frame validity. Effectively, this deviation may cause problems when there is only one active node in CAN network which transmits RWUF frames to wake up other nodes. Since this active node will not receive ACK bit in dedicated ACK slot, its transceiver automatically transmits EOF error, which makes the frame invalid. This corrupted RWUF frame than cannot cause remote wakeup of other nodes.

The last discrepancy with ISO 11898-6 standard is missing frame error counter mechanism [25]. This mechanism automatically wakes up the ECU when certain number of erroneous CAN frames was detected. Since the MSCAN module does not report erroneous (invalid) CAN frames to the MCU, these events cannot be handled by software and thus frame error counter cannot be implemented as required by the standard. In order to partially cover this functionality, a receive error counter (REC) is utilized in MSCAN module. When the REC reaches the warning level (REC > 96) it generates an interrupt and MCU can take appropriate action. The behavior of receive error counter is described in BOSCH CAN Specification 2.0 [24].

6.5 Pros and cons of designed solution

The software realization of ECU with selective wakeup functionality was designed as an alternative solution to dedicated PN CAN Transceivers. This chapter lists main advantages and disadvantages of designed solution in comparison with those dedicated HW chips. Further, the comparison with new power saving method, called Pretended Networking[10], is considered.

Advantages:

+ The overall ECU wakeup time from selective sleep mode is faster than when ISO 11898-6 CAN Transceiver is used, because the MCU is already running and some modules are initialized when RWUF is being detected.
+ The solution utilizes standard CAN Transceiver IP with robust EMC behavior and lower cost than ISO 11898-6 compatible CAN Transceiver.
+ The power saving potential is higher than with Pretended Networking, due to selective wakeup functionality.

Disadvantages:

- Designed approach is not standardized in AUTOSAR
- Proposed solution requires SW adjusted to used MCU and CAN Transceiver (SBC) for maximal power saving potential.
- The power consumption in RWUF Detection mode depends on frequency of MCU oscillator.

The most serious disadvantage of PN ECU approach seems to be the lack of standardization. Complex software development can be one of the major reasons preventing successful commercial usage.

[10] The AUTOSAR consortium plans to support this method in next release of AUTOSAR standard (4.0.4) [26].

6.6 Proposed modifications for next generation of NCV7471

The measurements shown that system basis chip NCV7471 is ideal choice for low power applications, especially thanks to integrated DC/DC converter. In order to further improve NCV7471 performance, few modifications were proposed. , These modifications can be considered to implement in next generation of NCV7471 system basis chip, and they can significantly simplify software realization of partial networking and thus further increase the power savings potential.

First two proposals are related to design of analog part of NCV7471 device. The effective current consumption of the CAN receiver[11] is currently about 2mA (current drawn from VBAT). With most recent analog design IP available, it can be reduced down to 1mA or less, as claimed by ON Semiconductor's designers. The second modification concerns internal biasing of high-voltage input pin CFG. Currently, the CFG pin is equipped with an internal pull-down resistor of typical resistance of 100kΩ, which results in current about 120μA, when this pin is connected to VBAT voltage (High level). This unnecessary waste of energy can be solved by smart internal biasing of CFG pin, similar to WU pin realization [8].

Next proposed modification is enhancing of device functionality by adding one extra state to state machine, called RWUF Detection mode. This digital part enhancement of NCV7471 device requires some additional bits in SPI registers. The proposed behavior in new RWUF Detection mode is summarized in following points.

- The RWUF Detection mode can be enabled/disabled by dedicated flag (bit) within SPI control registers
- The RWUF Detection mode is entered from Standby mode when CAN wakeup pattern (RWUP) is detected, if enabled.
- When RWUF Detection mode is entered, CAN controller is autonomously configured to receive-only mode and INTN pin is asserted to wakeup MCU.
- The watchdog timer is used to define t_{TOCAN} delay. This timer is started when RWUF detection mode is entered, and restarted every time, when bus activity is detected.
- When the t_{TOCAN} delay (watchdog timer) elapses, device sets dedicated SPI status flag and asserts INTN to inform MCU. The device goes back to Standby mode and autonomously configures CAN controller to wakeup mode.
- The MCU can request a transition to the Normal mode via SPI, when valid RWUF frame is detected.

[11] Consumption of CAN controller in receive-only mode (part of NCV7471)

This last modification significantly simplifies MCU software and makes whole process of selective wakeup more robust.

7 CONCLUSION

The conceptual design of automotive ECU with PN functionality, principles of Partial Networking and AUTOSAR software architecture were explored and briefly summarized in chapter 2. This chapter also introduces a system basis chip NCV7471 and basic solutions used in automotive electronic systems. In the second part of this thesis, the first concept of hardware and software of the evaluation ECU was created. Selected components are devices typically used in automotive industry with respect to a low current consumption. Designed software architecture represents a compromise between complexity and desired transferability of particular software parts. Expected power consumptions in different operational modes were also estimated for chosen hardware.

The fourth chapter brings detailed description of ECU hardware. After summarizing of the most important requirements, the final circuits were designed. Whole schematic, as well as PCB layout, is designed in manners common for automotive devices, with emphasis to EMC performance.

Further, the entire software architecture was developed, debugged and tested. The final realization of software architecture reflects challenging requirements of selective wakeup functionality, while keeping NCV7471 software drivers as transferrable as possible. AUTOSAR inspired SW architecture also improves code readability and maintainability.

The most important sixth chapter brings results of different evaluation tests and allows comparing of designed solution with other, standardized approaches. Achieved current consumptions are lower than originally expected, especially thanks to DC/DC converter of NCV7471 SBC and tailored software. Although the power saving potential is lower than with special ISO 11898-6 CAN Transceiver, the approach is still competitive, especially when cost and the time-proven EMC robustness is taken into account.

Furthermore, by applying of proposed modifications in next generation of NCV7471 system basis chip, partial networking software realization can be even more advantageous.

References

[1] Holger, H.: *Partial Networking: In-Vehicle Networks Can Reduce Costs and CO2 Emissions* [online]. 2010-3-4 [quote: 2012-25-3]. Available at: http://www.ecnmag.com/articles/2010/03/partial-networking-vehicle-networks-can-reduce-costs-and-co2-emissions

[2] Keskin, U.: In-*vehicle communication networks: a literature survey* [online]. Eindhoven: 2009-7-28. Available at: http://alexandria.tue.nl/repository/books/652514.pdf

[3] Corrigan, S.: *Controller Area Network Physical Layer Requirements* [online]. 2008-7-1 [quote: 2012-25-3]. Available at: http://www.ti.com/general/docs/lit/getliterature.tsp?baseLiteratureNumber=slla270&track=no

[4] ISO 11898-1. *Road vehicles - Controller area network (CAN) - Part 1: Data link layer and physical signalling*. Geneva: International Organization for Standardization, 2003.

[5] Audi, BMW, Daimler, Porsche and Volkswagen: *Hardware Requirements for Partial Networking.* 2010-28-5, Release 1.1. Internal document.

[6] Wille, M.: *Support of energy efficient technologies by AUTOSAR – Partial Networking & Co* [online]. Fellbach: Autosar, 2011. Available at: http://www.autosar.org/download/papersandpresentations/AUTOSAR_support%20of%20energy%20efficient%20technologies.pdf

[7] Burkhardt, F.: Partial deactivation of CAN nodes. *CAN Newsletter.*1/2012. CAN in Automation, March 2013. pg. 70-73. B25361

[8] ON Semiconductor: *NCV7471, System Basis Chip with a High-Speed CAN, Two LINs and a Boost-Buck DC/DC Converter*. 2012-1-10. Rev. 0. Available at: www.onsemi.com/pub_link/Collateral/NCV7471-D.PDF

[9] Autosar: *Autosar, Automotive Open System Architecture, Basics* [online]. [quote: 2012-25-3]. Available at: http://autosar.org/index.php?p=1&up=0&uup=0&uuup=0

[10] Autosar: *Layered Software Architecture* [online]. 2010-24-11.Version: 3.1.0. Available at: http://autosar.org/download/R4.0/AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[11] Autosar: *Specification of CAN Transceiver Driver* [online]. 2010-21-10. Version: 2.1.0. Available at: http://autosar.org/download/R4.0/AUTOSAR_SWS_CANTransceiverDriver.pdf

[12] Autosar: *Specification of Watchdog Driver* [online]. 2010-13-10. Version: 2.4.0. Available at: http://autosar.org/download/R4.0/AUTOSAR_SWS_WatchdogDriver.pdf

[13] Autosar: *Specification of SPI Handler/Driver* [online]. 2010-12-11. Version: 3.1.0. Available at: http://autosar.org/download/R4.0/AUTOSAR_SWS_SPIHandlerDriver.pdf

[14] Autosar: *Specification of DIO Driver* [online]. 2010-15-10. Version 2.4.0. Available at: http://autosar.org/download/R4.0/AUTOSAR_SWS_DIODriver.pdf

[15] Freescale: *MC9S12XS256 Reference Manual* [online]. 2011-7 Rev. 1.12. Available at: http://cache.freescale.com/files/microcontrollers/doc/ref_manual/MC9S12XS256RMV1.pdf

[16] ON Semiconductor: *NCV7471 System Basis Chip, Application Note.* 2012-22-5. Available upon request at: <http://www.onsemi.com/PowerSolutions/supportTechnical.do>

[17] ON Semiconductor: *NUP2105L/D, Dual Line CAN Bus Protector* [online]. 2012-1-1. *Rev.5.* Available at: www.onsemi.com/pub_link/Collateral/NUP2105L-D.PDF

[18] Audi, BMW, Daimler, Porsche and Volkswagen: *Hardware Requirements for LIN, CAN and FlexRay Interfaces in Automotive Applications*. 2012-5-4. Release 1.3. Internal document.

[19] PE Microcomputer Systems, Inc.: *Technical Summary For USB BDM MULTILINK* [online]. Rev. C. [quote: 2012-7-7] Available at: http://www.pemicro.com/downloads/download_file.cfm?download_id=208

[20] Texas Instruments: *INA210, Voltage Output, High or Low Side Measurement, Di-Directional Zero-Drift Series Current Shunt Monitor* [online], 2012 Available at: http://www.ti.com/lit/ds/symlink/ina210.pdf

[21] Autosar: *Specification of MCU Driver* [online]. 2010-13-10. Version: 3.1.0. Available at: http://www.autosar.org/download/R4.0/AUTOSAR_SWS_MCUDriver.pdf

[22] Autosar: *Specification of GPT Driver* [online]. 2011-13-10. Version: 3.2.0. Available at: http://www.autosar.org/download/R4.0/AUTOSAR_SWS_GPTDriver.pdf

[23] Autosar: *Specification of CAN Driver* [online]. 2011-02-11. Version: 4.0.0. Available at: http://www.autosar.org/download/R4.0/AUTOSAR_SWS_CANDriver.pdf

[24] Robert Bosch GmbH: *CAN Specification* [online]. Stuttgart: 1991, Version: 2.0. Available at: <http://esd.cs.ucr.edu/webres/can20.pdf

[25] ISO 11898-6. *Road vehicles - Controller area network (CAN) - Part 6: High-speed medium access unit with selective wake-up functionality*. Geneva: International Organization for Standardization, 2012-20-11.

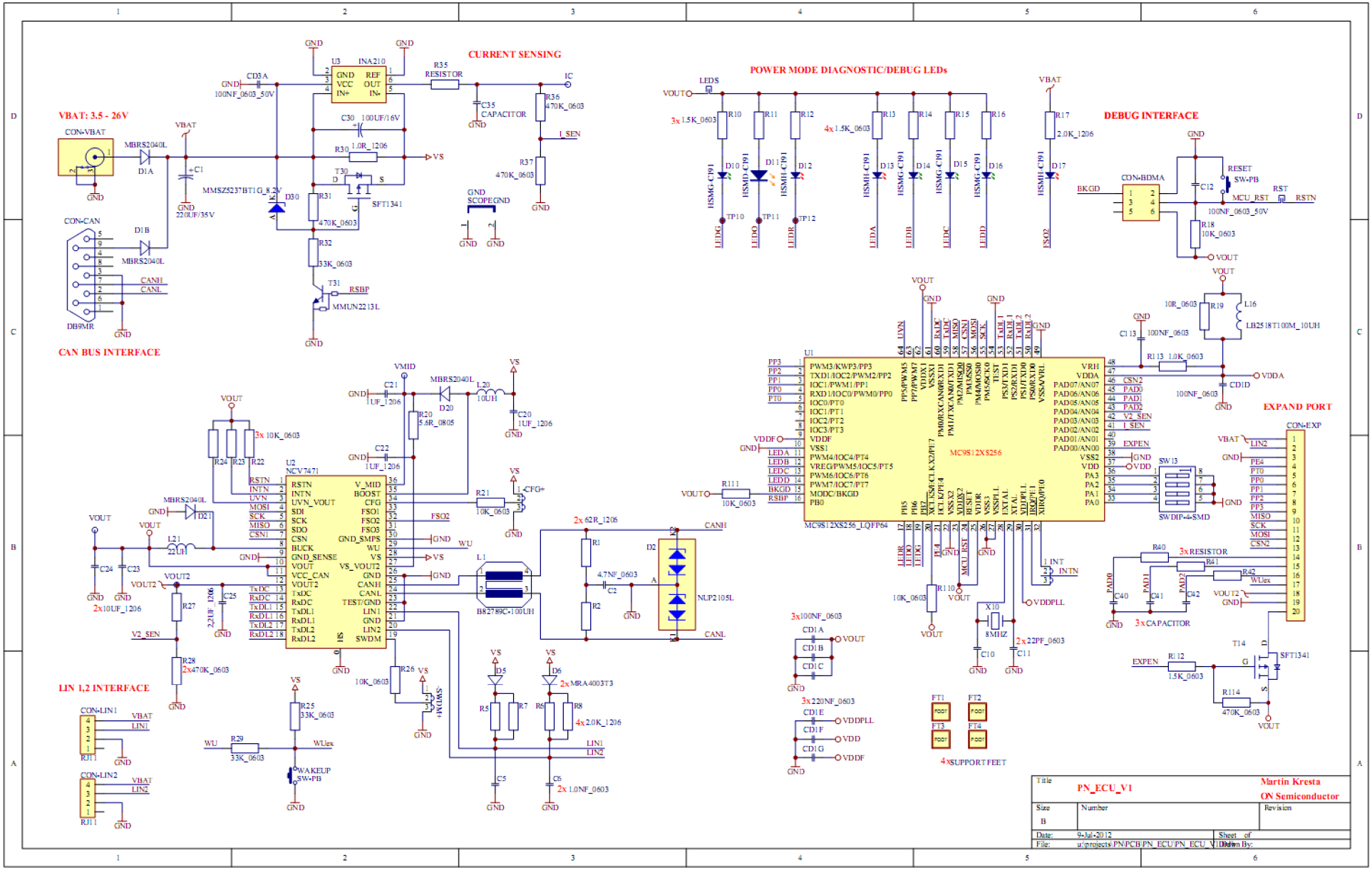
[26] NXP: *NXP HS-CAN transceiver TJA1145 & SBC UJA1168 for partial networking* [online]. 8-2011. Available at: http://www.nxp.com/documents/leaflet/75017167.pdf

List of abbreviations

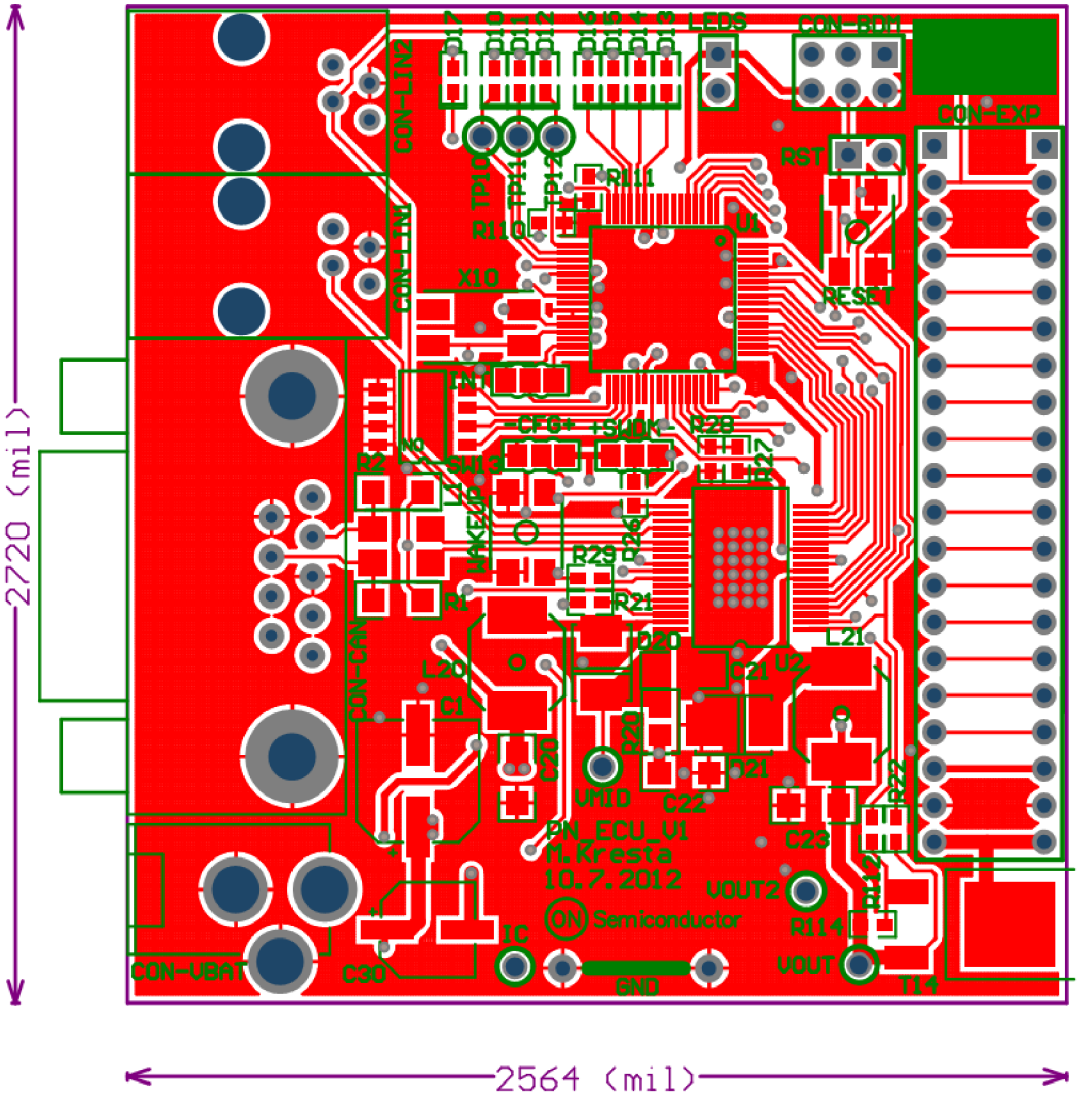
abbreviation	meaning
ECU	Electronic Control Unit
MCU	Microcontroller
CAN	Controller Area Network
LIN	Local Interconnect Network
PN	Partial Networking
DSP	Digital Signal Processor
SBC	System Basis Chip
FSO	Fail-Safe Output
SPI	Serial Peripheral Interface
OSBDM	Open Source Background Debug Mode interface
BSW	Basic Software (Autosar)
MCAL	Microcontroller Abstraction Layer (Autosar)
SWC	Software Component (Autosar)
RTE	Run-Time Environment (Autosar)
API	Application Interface function
DIO	Digital Input/Output
RWUP	Remote Wakeup Pattern (Partial Networking)
RWUF	Remote Wakeup Frame (Partial Networking)
SYSCLK	Microcontroller System Clock
ALU	Arithmetic Logic Unit
PLL	Phase Locked Loop
LDO	Linear Low Dropout regulator
SMPS	Switched Mode Power Supply
GPIO	General Purpose Input/Output
PCB	Printed Circuit Board
PDU	Packet Data Unit
ISR	Interrupt Subroutine
DLC	Data Length Code
POR	Power-on Reset
IVN	In-Vehicle Network
REC	Receive Error Counter

List of appendixes

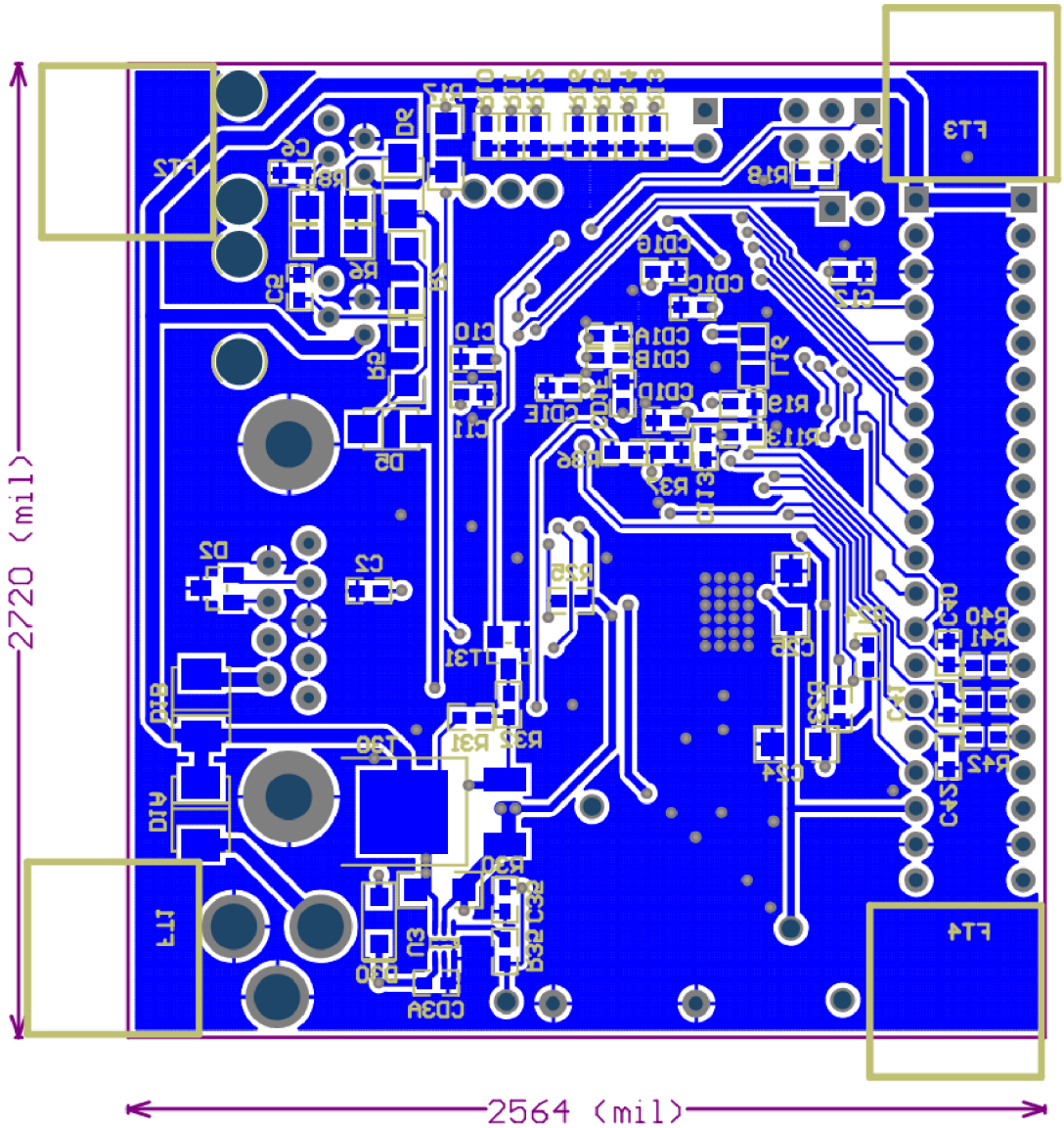
Appendix 1


CURRENT SENSING
VBAT: 3.5 - 26V
CAN BUS INTERFACE
LIN 1,2 INTERFACE
POWER MODE DIAGNOSTIC/DEBUG LEDs
DEBUG INTERFACE
EXPAND PORT
MC9S12XS256_LQFP64
SUPPORT FEET
Title
PN_ECU_V1
Martin Kresta
ON Semiconductor
Size
B
Number
Revision
Date: 9-Jul-2012
File: u:\projects\PN\PCB\PN_ECU\PN_ECU_V1.DDB
Sheet of
Drawn By:

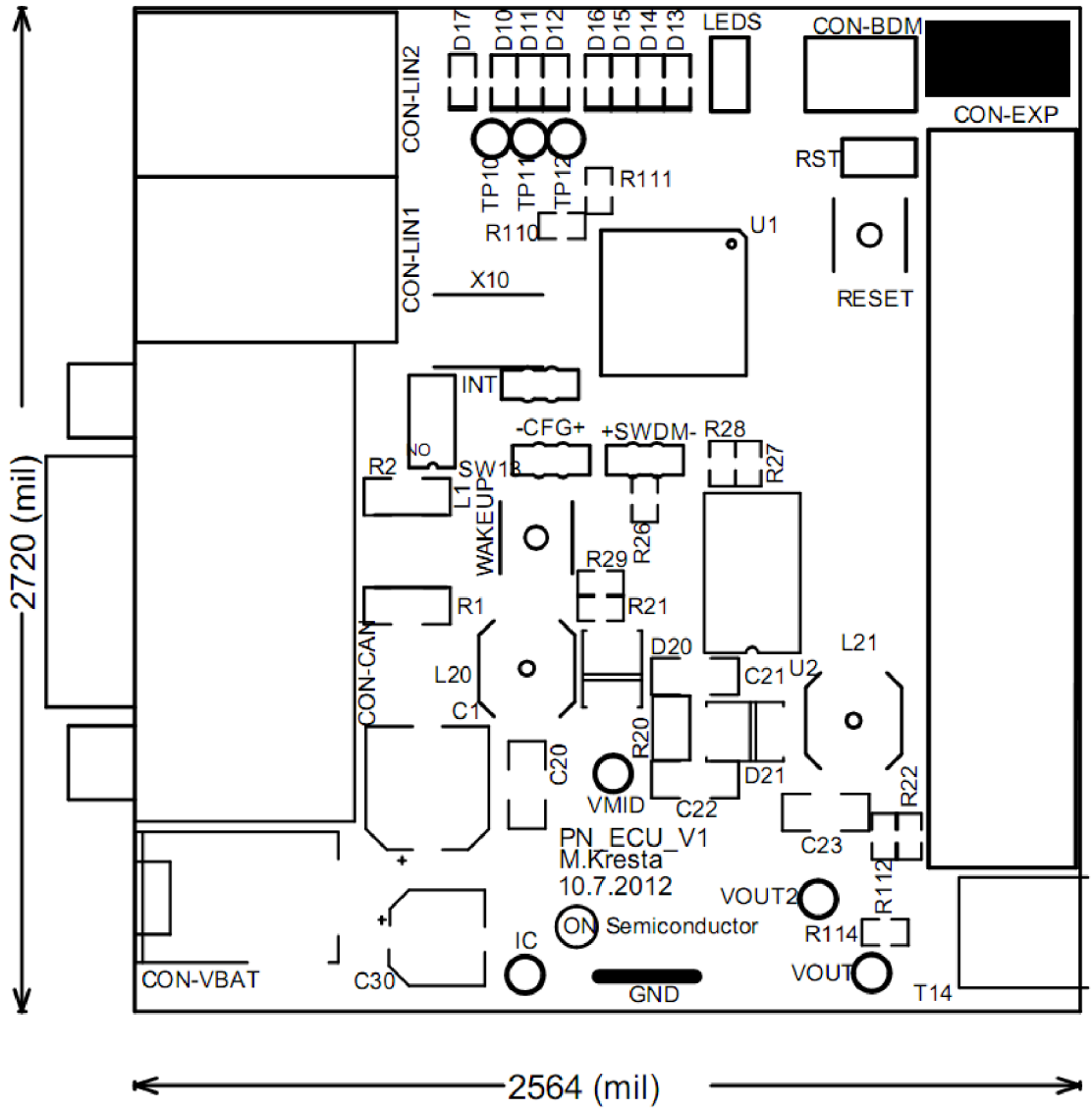
Appendix 2



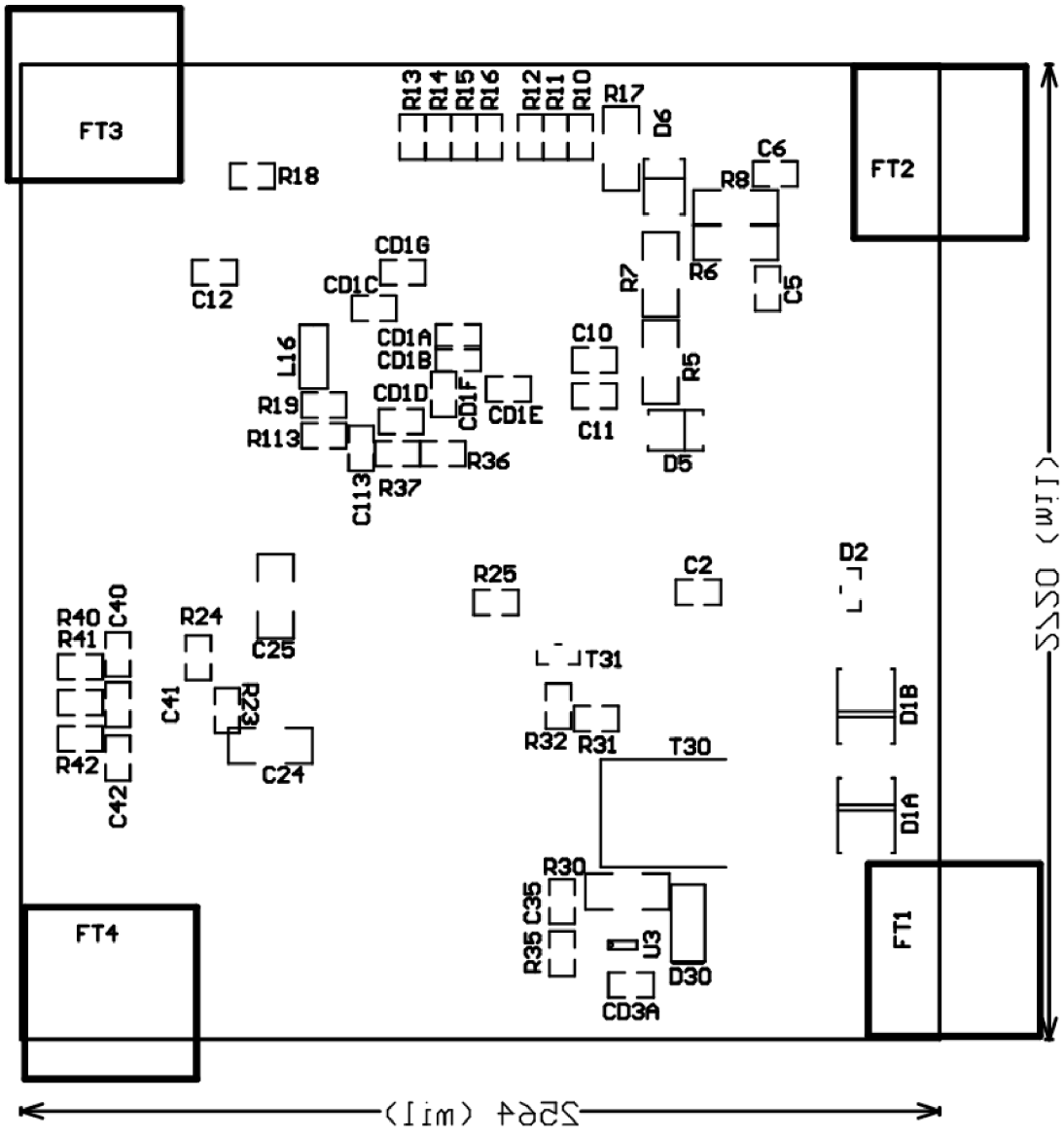
A - Copper top layer + components on top side



B - Copper bottom layer + components on bottom side



C - Assembly drawing, top side



D - Assembly drawing, bottom side

ECU operational modes with corresponding static current consumptions

8MHz Oscillator (maximal CAN baudrate 1Mbps)

ECU mode Designed ECU power mode	**SBC NCV7471** Configuration of NCV 7471	**MCU Freescale MC9S12XS** Configuration of MCU	**ECU HW peripherals** power status of ECU functional peripherals (not assembled on evaluation ECU)	**ISO 11898-6** Current consumption of PN CAN transceiver in corresponding mode (required by ISO 11898-6)	**Current cons.** Current consumption from VBAT (12V), measured on evaluation ECU (NCV7471 + MCU)
Standard sleep mode PN functionality disabled RWUP detection	Sleep mode CAN - wakeup VOUT - off	Unpowered	Unpowered	-	**60µA** (T_A = 25°C)
Fast-startup sleep mode PN functionality enabled RWUP detection	Standby mode CAN - wakeup VOUT - on Watchdog - off	Pseudo stop mode Oscillator 8 MHz Start-up time about 3,5µs CAN - sleep	Unpowered	30µA[1)]	**255µA** NCV7471 ≈ 100µA MCU ≈ 155µA (T_A = 25°C)
RWUF detection mode PN functionality enabled RWUF detection	Standby mode CAN - receiveonly VOUT - on Watchdog - off	Wait (Run) mode SYSCLK from oscillator (8 MHz) CAN - listenonly PLL - disabled	Unpowered	500µA[1)]	**3.8mA (4.9mA)**[2)] NCV7471 ≈ 2mA MCU ≈ 1.8mA
Normal mode Full functionality of ECU	Normal mode CAN - normal VOUT - on Watchdog - on VOUT2,LIN - by SPI	Run mode PPL used for SYSCLK (20 MHz) CAN - normal	Powered (Current consumption up to 450mA)	-	**12mA**[3)] + consumption of ECU function hardware

Note 1) : At this moment there is not known any CAN transceiver which meets this requirement (1/2013)

Note 2) : MCU goes to Run mode (4.9mA) only when RWUF with matching ID is detected. After decoding of RWUF data, MCU goes back to Wait mode (3.8mA)

Note 3) : Measured with CAN bus idle