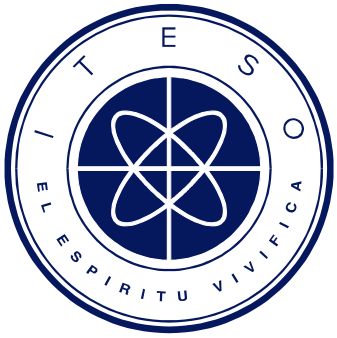
Instituto Tecnológico y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación el 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

ESPECIALIDAD EN SISTEMAS EMBEBIDOS



AUTOMOTIVE CONTROL CATALYZER TO SYNTHETIZE $CaCO_3$ FROM RESIDUAL CO_2 EMBEDDED CONTROL SYSTEM

Tesina para obtener el grado de:

ESPECIALISTA EN SISTEMAS EMBEBIDOS

Presentan: Ing. César Alejandro Hernández Navarro
Ing. Jesús Íñiguez Ramírez
Ing. Juan Pablo García Mares

Director: Héctor Antonio Rivas Silva

San Pedro Tlaquepaque, Jalisco. Julio del 2018

Dedications

First of all, we wish to thank CONACYT for the opportunity to increase our professional growth
Scholarship numbers 857100, 768811 and 452398

We expect to contribute to the society in the future with the development of new technologies and enhance our living standards.

Also, we would like to thank ITESO University, the coordinator Luis Rizo, our teachers, specially Francisco Martinez and Michele Brennan, our advisor Hector Rivas and our classmates for all the shared knowledge and experiences during this year, and most importantly our family members, for all their support, patience, and encouragement during this period.
Thank you.

Abstract

The pollution generated by the automotive sector has been an environmental issue in the latest years, and nowadays, different governmental and private sectors have taken actions on this matter. One of these efforts tries to provide a solution for contaminating gas emissions produced from different fuel combustion systems. The aim of the present project is to design a reliable and high-quality system that senses the environmental temperature, relative humidity, and calculates the dew point to control a catalyzer capable of transforming a considerable amount of exhaust gases into a new fuel component before they are released back into the environment. The control module was developed considering the requirements demanded by the automotive industry. The controller was based on an AUTOSAR architecture, it also included the standard CAN 2.0 communication protocol performed within the microcontroller validated as grade 2 by the Automotive Electronics Council, and the SHT11 sensor used was certified against RoHS. Equally important, the software architecture complied with the complexity inherent in AUTOSAR specifications. Therefore, different techniques were required for its solution, including, boundary diagram, requirement specifications, software interface, and module interaction definitions. Once these requirements where met, the code was implemented. As a result, this module could be categorized as an automotive-grade product that can be introduced in the automotive market.

Resumen

La contaminación generada por el sector automotriz ha sido un problema de medio ambiente en los últimos años, y hoy en día, diferentes sectores gubernamentales y privados han tomado acciones en la materia. Uno de esos esfuerzos procura de proveer una solución para la emisión de gases de producidos de diferentes sistemas de combustión. El objetivo del presente proyecto es diseñar un sistema de alta calidad y confiable, capaz de transformar una cantidad considerable de los gases emitidos en nuevo combustible antes de ser liberados de vuelta al entorno. El módulo de control fue desarrollado considerando los requerimientos demandados por la industria automotriz. El controlador fue basado sobre la arquitectura AUTOSAR, este también incluyo el protocolo de comunicación estándar CAN 2.0 desempeñado con el microcontrolador validado como grado 2 por el Consejo de Electrónica Automotriz, y el sensor SHT11 usado fue certificado contra RoHS. Igualmente importante, la arquitectura de software cumple con la complejidad inherente de las especificaciones de AUTOSAR, por consiguiente, diferentes técnicas fueron requeridas para su solución, incluyendo la definición de, diagramas de límites, especificación de requerimientos, interfaces de software e interacción de módulos. Una vez que los requerimientos fueron conocidos, el código fue implementado. Como resultado, este módulo puede ser categorizado como un producto de grado automotriz que puede ser introducido en el mercado automotriz.

List of Figures

Figure 3-1 Boundary Diagram 29
Figure 3-2 System P-Diagram 30
Figure 3-3 Software P-Diagram 31
Figure 3-4 Software Architecture 32
Figure 3-5 Dynamic Behavior from Operative System 41
Figure 4-1 SHT11 temperature request frame 44
Figure 4-2 Debug terminal 45
Figure 4-4 Comparison between obtained and sent data 45
Figure 4-5 Data read through CAN terminal 45

List of Tables

Table 2-1 SHT11 electrical characteristics.....22
Table 2-2 Temperature coefficients related to voltages 22
Table 2-3 Temperature coefficients related to resolution.....22
Table 2-4. Relative humidity coefficients related by resolution.....23
Table 2-5 SHT11 Dew points coefficients 23
Table 3-1. Memory map 42

List of Acronyms and abbreviations

AEC	Automotive Electronics Council
ARM	Acorn RISC Machine
AUTOSAR	AUTomotive Open System Architecture
CAN	Controller Area Network
CO	Carbon Monoxide
CO_2	Carbon Dioxide
CPU	Central Process Unit
DSP	Digital Signal Processor
ECU	Electronic Control Units
FPU	Floating-Point Unit
FR4	Flame Resistant grade 4
FreeRTOS	Free Real Time Operative System
ISO	International Standard Organization
LCP	Liquid Crystal Polymer
MAC	Multiple Accumulate Instructions
NO_2	Nitrogen Dioxide
OEM	Original Equipment Manufacturer
OS	Operating System
OEMs	Original Equipment Manufacturer
PBB	Polybrominated Biphenyl
PBDE	Polybrominated Diphenyl Ethers
PTAT	Proportional To Absolute Temperature
RoHS	Restriction of Hazardous Substances
RR	Round-Robin
SIMD	Single Instruction Multiple Data
SW	Software
UVB	Ultraviolet B
WEEE	Waste Electrical and Electronic Equipment

Table of Contents

Dedication iii

Abstract iv

Resumen v

List of Figures vi

List of Tables vii

List of Acronyms and abbreviations viii

Introduction 11

1. Background **13**

2. Automotive Catalyzer Control Module **15**

2.1. ARM ARCHITECTURE 15
2.1.1 Cortex-M4 15
2.2. AUTOSAR 16
2.2.1 AUTOSAR Specifications 17
2.3. CAN PROTOCOL 17
2.4. RTOS AND SCHEDULERS 18
2.4.1 Round-Robin Scheduling 19
2.5. ENVIRONMENT VARIABLES 19
2.5.1 Temperature 20
2.5.2 Relative Humidity 20
2.5.3 Dew Point 20
2.6. MEASUREMENT DEVICES 21
2.6.1 Temperature Measurement 22
2.6.2 Humidity measurement 23
2.6.3 Dew Point determination 23

3. Methodology **24**

3.1. SYSTEM REQUIREMENTS 25
3.1.1 Software Requirements 26
3.2. DIAGRAMS 29
3.2.1 Boundary Diagram 29
3.2.2 System P-Diagram 30
3.2.3 Software P- Diagram 31
3.3. SOFTWARE ARCHITECTURE 32
3.4. HARDWARE DESIGN 33
3.5. SOFTWARE DESIGN 33
3.5.1 Static Decomposition of SW components 33
3.5.1.1 Application/Sensor 33
3.5.1.1.1 SensEnv_Init 33
3.5.1.1.2 Process_Sensing_Env 34
3.5.1.2 EcuAL/SHT 34

3.5.1.2.1 SHT_Init 34
3.5.1.2.2 SHT_CalculateDP 35
3.5.1.2.3 SHT_ObtainTemp 35
3.5.1.2.4 SHT_ObtainRH 36
3.5.1.2.5 SHT_ReadTemperatureRaw 36
3.5.1.2.6 SHT_ReadHumidityRaw 37
3.5.1.3 McAL/Pio 37
3.5.1.3.1 PIO_Configure 37
3.5.1.3.2 PIO_Set 38
3.5.1.3.3 PIO_Clear 38
3.5.1.3.4 PIO_Get 39
3.5.1.4 Services/CanNm 39
3.5.1.4.1 CanNm_Tx 39
3.5.1.4.2 Can_SetSignal 40
3.5.2 Dynamic behavior of software components 41
3.6. MEMORY MAP 42
4. Results 43
Conclusions 46
References 47
Appendices 49

Introduction

Over the past century, there has been a dramatic temperature increase known as global warming caused by the greenhouse effect, aside from other outcomes produced by global pollution, such as extreme weather [1]. It has been discussed by governmental and social areas around the world that the deterioration of the ozone layer is one of the main causes of the greenhouse effect.

The ozone layer is located between 9.3 to 18.6 miles above the Earth and works as a shield from the harmful ultraviolet B (UVB) radiation emanated by the sun. The ozone molecule is constantly being formed and broken down in the high atmosphere, 10 to 50 kilometers above the Earth, in the region called the stratosphere. Today, there is concern that the ozone layer is being devastated due to the release of pollution, such as carbon dioxide, nitrogen dioxide, and particulate matter. Such deterioration allows large amounts of ultraviolet B rays to reach the Earth, which can cause skin cancer and cataracts in humans and harm animals as well, and it also inhibits the reproductive species of the last stage of the food chain [2].

The main challenge faced by many researchers in the Automotive Industry, as well as the European Commission, European Union legislations according to the Kyoto protocol and the Paris agreement [10], is to reduce the carbon dioxide (CO_2) and nitrogen dioxide (NO_2) emissions levels produced by road transport vehicles and thus, contribute to the improvement of air quality [2].

Several solutions have been developed to reduce or enhance the fuel system. For instance, a study demonstrated the process of treating CO_2 emissions to convert them into methanol or methane. This is an alternative option of CO_2 reduction to carbon monoxide (CO). When the hydrogen is obtained by renewable energy, this method is an efficient way to store the electricity generated from renewable sources [3]. A second discovery is a solution that converts CO_2 directly into gasoline-range hydrocarbons under industrial relevant conditions using a specific catalyst [4]. A last example and most related to the present research is a method for reducing pollution using calcium oxide (CaO) to capture CO_2 from exhaust gases or in the reactor generated by the automotive sector [8].

The Automotive Catalyzer Control described in this thesis is an adaptive module aimed to reduce the CO_2 and NO_2 emissions from fuel injection vehicle systems by means of gas condensation regarding on site conditions up to 3.5 times [9]. To achieve this, the system shall measure temperature and humidity of the environment, compute the data to obtain the relative humidity, transmit processed information through Controller Area Network (CAN) automotive communication protocol.

The embedded system developed under the present work is a dedicated computer system that focuses on one or two specific functions. This system includes hardware, such as electrical and mechanical components that perform only certain tasks. Design engineers can optimize size, cost, power consumption, reliability, and performance. In addition to the system incorporates sensors to measure the temperature, humidity, relative humidity, CO_2 quantity in the environment, among others; aside, it contains actuators to improve the catalyst reaction at its highest performance, such as a Peltier cell to ensure the temperature is always matching the dew point. The developed controller uses an ARM-based microcontroller and the programing language C has been used to develop the code under AUTOSAR-based program model according to the current automotive standard.

The present project describes the development of a controller (Automotive Catalyzer Control) based on an embedded system to manage, analyze, and control the catalyzer that converts CO_2 into $CaCO_3$ using water (H_2O) and CaO_2 as catalyst, to enhance the efficiency and thus, obtain the highest performance from the catalyzer.

1. Background

Several institutions and universities have investigated several methods to improve the pollution levels in the environment. Studies performed by the Swiss Federal Institute of Technology (ETH) [10] and the Madras Institute of Technology [11] report that the fossil fuel synthesis, specifically CO_2 emissions, can be designed to obtain a minor contaminating disposable substance, such as $CaCO_3$ which, can be used as fuel for the same vehicle right after it is produced. The findings demonstrate two methods for reducing CO_2 exhaust emissions, however these works do not aim to provide a practical solution in the automotive industry.

The previous research by M. Ammann entitled "*Adaptive Control of a Three-Way Catalytic Converter*", states the catalyzer efficiency by reducing exhaust emissions in automobiles, adding one control device and enhancing the function of oxygen storage. In addition, an adaptive system to control the oxygen level was considered as a strategy for reducing emissions by oxidizing unburned hydrocarbons, for instance, carbon monoxide and reducing nitrogen oxides. The oxygen level should be observed by a suturing integrator because it cannot be measured, and thus, it can be estimated by a recursive Markov's method and used as the control variable [10].

A third work entitled *"Energy analysis of $CaCO_3$ calcination with CO_2 capture"* describes a method to capture and "sequestrate" the CO_2 before it is released into the atmosphere using active lime (main component CaO) in the exhaust gas chamber. That is, the CaO absorbs CO_2 to yield calcium carbonate ($CaCO_3$), and then, $CaCO_3$ is thermally decomposed to CaO again and nearly pure CO_2 is released for sequestration, resulting in a number of joules per mole used as reusable fuel [11].

These previous reports did not include embedded control systems to monitor and control the variables, as the one described by Bo Qu and Daowei Fan. They developed an embedded control system that allowed the user to monitor via internet a temperature sensor using a 32-bit ARM microcontroller architecture [12] running Free Real Time Operative System (FreeRTOS) as an operating system.

The present work is based on a previous research paper by Manuel TéllezGirón-Enríquez who conceptualized an embedded catalyzer control prototype that synthetizes $CaCO_3$ by the chemical reaction between CO_2, H_2O (ambient condensed) and, $CaCO_2$ as a base reactive substance. This solution uses an 8-bit microcontroller running with a real-time operating system over the communications layer scheme OSI, and a humidity/temperature sensor [13]. The present work is improved by fulfilling an automotive standard like AUTOSAR, additionally implementing automotive communication protocol (CAN).

2. Automotive Catalyzer Control Module

2.1. ARM Architecture

System architecture is defined as the relationship between hardware and software. Both parts establish the rights and responsibilities between each other to define how hardware behaves with a correctly written software. The architecture also defines the basic instruction set and the exception and memory model that rely upon the operating system, in effect, the architecture defines what the central process unit (CPU) must do [17].

Among all the benefits that place ARM as the most popular architecture for embedded microprocessors, it is necessary to mention that ARM processors have the following features:

- Low power consumption devices
- Low heat emission devices
- Relative low cost
- Powerful design

These characteristics place ARM as the most suitable standard architecture in the embedded world and therefore, it has been considered an integral part of the present development.

2.1.1 Cortex-M4

ARM Cortex-M4 processor has been developed to attend digital signal control markets that demand an efficient, easy-to-use blend of control and signal processing capabilities. It has been designed to satisfy the emerging category of flexible solutions, specifically targeting the automotive and the embedded audio markets [18].

There are four key benefits that set the cortex-m4 family as one of the most popular among microprocessors. These benefits are also considered optimal characteristics when the project requires a correct, stable, and reliable performance:

- *"Gain the advantages of a microcontroller with integrated digital signal processor (DSP), single instruction multiple data (SIMD), and, multiple accumulate instructions (MAC) that simplify overall system design, software development, and debug.*
- *Accelerate single precision floating-point operations up to ten times over the equivalent integer software library with the optional Floating-Point Unit (FPU).*
- *Developed solutions for a large variety of markets with full-featured ARMv7-M instruction set that has been proven across a broad set of embedded applications.*
- *Achieve exceptional 32-bit performance with low dynamic power, delivering leading system energy efficiency due to integrated software-controlled sleep modes, extensive clock gating, and, optional state retention" [18].*

2.2. AUTOSAR

AUTOSAR (AUTomotive Open System ARchitecture) is a worldwide cooperative association that is integrated by automotive OEMs (Original Equipment Manufacturer), electronic semiconductors, and software industry. AUTOSAR provides a solution to overcome the complexity of developing software in vehicles by defining architecture specifications and explaining the interaction between software and hardware [19].

Since its foundation in 2003, AUTOSAR has defined four different partnership programs known as core, premium, development, and associate, each having their hierarchy and pre-defined roles for the cooperative association preservation goals defined below:

- Define a reference architecture for Electronic Control Units (ECU)
- Easy software (SW) upgrade/update

- SW maintainability
- Reduce SW complexity
- SW reusability

2.2.1 AUTOSAR Specifications

The AUTOSAR cooperative defines a set of specifications for different software application interfaces enhanced by a similar method within the embedded systems. These software architecture modules are designed as independent layers of code, defined by the interaction needed. Furthermore, these allows the software modules to be re-used by various manufacturers with similar purposes and by providing growth ability.

The catalyzer control module is based on AUTOSAR specifications, taking advantage of its capabilities and intended to fulfill the automotive-grade qualifications needed.

2.3. CAN Protocol

The CAN (Controller Area Network) communication protocol was first developed by Bosch in 1983 with the intention to solve the internal car communication gap in the automotive industry. In collaboration with other companies, the protocol was upgraded and standardized by the International Standard Organization (ISO) 11898 in 1993 [20], and it was embraced in the automotive industry because of its characteristics, such as:

- Noise resistant
- Error detection
- Traffic congestion avoidance

The protocol updates have produced different variations according to the market demands, and they have been implemented in non-automotive segments where the communication is crucial [21].

2.4. RTOS and Schedulers

RTOS is composed by two concepts, namely, "Real-Time" and "Operating System".

"Real-Time indicates an expectant response or reaction to a certain event or action on the instant of its evolution" [27]. The expectant response portrays the logical handling of the result produced. The moment of the event's evolution denotes the deadline for producing the result.

Operating System (OS) is a service program that provides an interface between hardware and application programs. "*OS is commonly equipped with features or resources like: Multitasking, Synchronization, Interrupt and Event Handling, Input/Output, Inter-task Communication, Timers and Clocks and Memory Management to fulfill its primary role of managing the hardware resources to meet the demands of application programs*"[27].

RTOS is therefore an operating system that supports real-time applications and embedded systems by supplying logically correct results within the deadline required. Such capabilities define its deterministic timing behavior and limited resource utilization nature [27]. Something like RTOS are the schedulers, these can also manage different tasks but in a simpler way and without many resources an RTOS needs.

There are many scheduler's algorithms, they differed in their characteristics and in how they prioritize the tasks. Some of the most popular algorithms are:

- Round-Robin scheduling
- First-in, First-out scheduling
- Preemptive priority scheduling

2.4.1 Round-Robin Scheduling

The Round-Robin (RR) scheduling is one of the most common and simplest non-preemptive scheduling algorithms. As William Stallings explains, it is composed of time slices (also known as time quanta) that are assigned to each process in equal portions and in circular order, and by handling all processes without priority (also known as cyclic executive) [28]. Round-robin scheduling is simple, easy to implement, and starvation-free.

In the present project, a Round-Robin scheduler has been implemented to manage threads (or tasks) with different periodicity. It works with a system tick (or time slices) of 500 microseconds and it defines 3 basic threads from which other 3 threads are derived.

2.5. Environment variables

Ambient temperature and humidity are known to have an impact in the automotive pollution process, and as a result, there are direct effects in three situations:

- Direct effect on emission rates via temperature adjustments.
- Direct effect for NOx factor via humidity corrections.
- Indirect effect through air conditioning adjustments.

The focused variables for the current work are based on the study by D. Choi entitled "*The Impacts of Temperature and Humidity on Emissions"* where the association between temperature and relative humidity demonstrates that both have a substantial impact on emission levels, especially for cold temperatures, emphasizing the importance of obtaining the local meteorological data accurately when the automobile is running [22]. For this reason, the temperature and humidity as the dew point are completely relevant for the approach of this project.

2.5.1 Temperature

The temperature is defined as the average of kinetic energy of the molecules or atoms in a body, and it is not dependent on the mass, material, or shape of the object. Temperature can be measured in Celsius, Fahrenheit, Rankin or Kelvin depending on the unit system.

Heat is described in physics as a measured of the total energy of the atoms or molecules in a body, this form of energy could be transmitted by [23]:

- **Conduction**: appears when the heat energy travels through a body, the particles begin to vibrate and transmit energy from one another.

- **Convection**: the particles move their positions allowing a flow of materials that transfer the heat evenly to the available space. This case occurs just for liquids or gases.

- **Radiation**: the radiant energy is absorbed by the bodies and is transmitted using electromagnetic waves, so it is not necessary to have a medium for transmission.

2.5.2 Relative Humidity

Oxford dictionary defines the humidity as "*A quantity representing the amount of water vapour in the atmosphere or in a gas*" [24]. The relative humidity is defined as a measure of the actual amount of moisture in the air against all the moisture that the air can hold [25].

2.5.3 Dew Point

Dew Point is defined as "*the temperature at which the partial pressure of the water vapor is equal to the saturated vapor pressure of water at that temperature"* [26]. This concept is a relationship between the changes of relative humidity and temperature.

2.6. Measurement devices

The SHT11 sensor provides a full calibrate digital output for humidity and temperature. This device uses a unique capacitive sensor element for measuring relative humidity while the temperature is measured by a band-gap sensor. The SHT11 sensor includes CMOSens® technology that guarantees excellent reliability and long-term stability, which results in a superior signal quality, with a fast response time and insensitivity to external disturbances.

The sensor housing consists of a liquid crystal polymer (LCP) cap with flame resistant grade 4 (FR4) epoxy glob top on. It is completely compliant with the *Restriction of Hazardous Substances* (RoHS) and Waste Electrical and Electronic Equipment (WEEE) regulations, and fully free of Lead, Cadmium, Mercury, Chromium, Polybrominated Biphenyl (PBB) and Polybrominated Diphenyl Ethers (PBDE).

The humidity sensor is not a normal electronic component, it requires handling with care, since the sensor may get an offset in its readings by the chemical vapor at high concentrations for a long exposure time.

The relative humidity also depends strongly on temperature. Therefore, it is essential to keep the humidity sensor at the same temperature of the air measured as well as prevent the undesired heat transference by adding ventilation or improving mounting positions.

The sensor could be connected to the development boar SAMV71 according with its electrical characteristics described in Table 2-1.

Table 2-1 SHT11 electrical characteristics

Parameter	Conditions	Min	Typ	Max	Units
Power supply DC		2.4	3.3	5.5	V
Supply current	Measuring		0.55	1	MA
	Average	2	28	0	uA uA
	Sleep		0.3	1.5	mA
	On			1	
Output current	Tri-stated		10	20	uA

2.6.1 Temperature Measurement

The SHT11 uses a proportional-to-absolute temperature (PTAT) band-gap as a linear sensor with the formula 2-1:

$$T = d_1 + d_2 * SO_T \qquad (2\text{-}1)$$

Where the parameter d_1 and d_2 depends of the resolution configured and the sensor's power supply, these parameters are shown in Tables 2-2 and 2-3.

Table 2-2. Temperature coefficients related to voltages

VDD	d^1(°C)	d^1 (°F)
5V	-40.1	-40.2
4A	-39.8	-39.6
3.5V	-39.7	-39.5
3V	-39.6	-39.3
2.5V	-39.4	-38.9

Table 2-3. Temperature coefficients related to resolution

SO_T	d_2(°C)	d_2(°F)
14 bit	0.01	0.018
12 bit	0.04	0.072

2.6.2 Humidity measurement

The SHT11 uses a non-lineal humidity sensor that must be compensated by the formula 2-2 and the coefficients from Table 2-4.

$$RH_{linear} = C_1 + C_2 * SO_{RH} + C_3 * SO_{RH}^2 \ (\%RH) \qquad (2\text{-}2)$$

Table 2-4. Relative humidity coefficients related by resolution

SO_{RH}	C_1	C_2	C_3
12 bit	-2.0468	0.0367	-1.5955E-6
8 bit	-2.0468	0.5872	-4.0845E-4

2.6.3 Dew Point determination

The SHT11 sensor does not measure the dew point directly but it can be derived from humidity and temperature data since both measurements are obtained from the same point at the same time, and thus, assuring accurate information.

For dew point calculations, the formula 2-3 is based on the temperature range and provides a good accuracy for the dew point approximation using the coefficients from Table 2-5.

$$T_d(RH,T) = T_n \frac{\ln\left(\frac{RH}{100\%}\right)+\frac{m*T}{T_n+T}}{\mathrm{m}-\ln\left(\frac{RH}{100\%}\right)-\frac{m*T}{T_n+T}} \qquad (2\text{-}3)$$

Table 2-5 SHT11 Dew points coefficients

Temperature Range	Tn (°C)	m
Above water, 0-50°C	243.12	17.62
Above ice, -40 - 0°C	272.62	22.46

3. Methodology

The current development system was performed in collaboration with electronics and chemistry engineers to focusing on the following requirements:

- Measurement of the temperature in Celsius every 500 milliseconds with a 12-bit resolution
- Dew Point calculation by using relative humidity and temperature measurement every 500 milliseconds
- CAN Communication Protocol embedded in the module
- Automotive grade quality

According to the previous system requirement, we used the advantage from the ready to development board, SAM V71 Xplained Ultra by Microchip®. This board uses one ATSAMV71Q21 [14] microcontroller, that not only satisfies the quality grade according to the Automotive Quality Normative from the International Organization for Standardization (ISO-TS-16949) but it is also validated as Automotive Electronics Council (AEC-Q100) grade 2 qualification, and additionally fulfills the necessary characteristics from the CAN Protocol Version 2.0 compliance according to the Automotive CAN controller specification (ISO 11898-1).

Additional advantages from the SAMV71 board is the incorporation of ATA6561 CAN Transceiver that allows communication with other ECUs without any external device [15].

The SHT11 Sensor by Sensirion® was selected not only by its sensing characteristics for temperature and humidity measurement, but also for its excellent long-term stability and low-cost, in addition to its 2-wire interface communication, accomplishing a fast integration within the system [16]. Equally important, each sensor was individually calibrated by the manufacturer with specific coefficients stored in their internal memory.

3.1. System Requirements

The following requirements were provided by the sponsor:

1. The system **shall** use a microcontroller that compliance with the Automotive Quality Normative.
2. The system **shall** use a microcontroller validated by Automotive Electronic Council.
3. The system **shall** use a 32 bits microcontroller architecture.
4. The system **shall** run over the SAM V71 Xplained Ultra Evaluation Kit.
5. The system **shall** be AUTOSAR 4.0.3 compliant.
6. The system **shall** use a RTOS or scheduler to manage different periodic tasks.
7. The system **shall** use an automotive grade temperature sensor.
8. The temperature sensor **shall** operate from -40°c to 120°C.
9. The system **shall** use an automotive grade humidity sensor.
10. The humidity sensor **shall** operate from 0% to 100% RH.
11. The system **shall** obtain the dew point from the catalyzer at least once each second.
12. The system **shall** communicate with other ECUs using the standard CAN 2.0.
13. The system **shall** provide the measurements to other ECUs through CAN frame each second.
14. The system **shall** use the standard CAN PINS at DB-9 Connector.
15. The system **shall** storage measurement buffers each 5 second.

3.1.1 Software Requirements

The system runs over an ATSAMV71Q21 microcontroller in which the following requirements were applied:

1. The microcontroller **shall** use 12.0 MHz external crystal.
2. The source clock **shall** be run at a speed of 30 MHz.
3. The scheduler **shall** be a cooperative scheduler.
4. The system tick of the scheduler **shall** operate at a 500 microseconds speed.
5. The scheduler **shall** be a cooperative non-preemptive scheduler.
6. Port PA03 **shall** be configured as input.
7. Port PA03 **shall** use internal pulldown configuration.
8. Port PA03 **shall** be connected to sensor data output.
9. Port PA04 **shall** be configured as output.
10. Port PA04 **shall** use internal pulldown configuration.
11. Port PA04 **shall** be connected to sensor clock input.
12. Port PA04 **shall** use falling edge interrupt configured.
13. The system **shall** configure the ports calling the SensEnv_Init function.
14. The SensEnv_Init function **shall** not have any parameter.
15. The SensEnv_Init function **shall** not return any value.
16. The system shall manage the interaction with the sensor calling the Process_Sensing_Env function.
17. The Process_Sensing_Env function shall not have any parameter.
18. The Process_Sensing_Env function shall not return any value.
19. The temperature measurement **shall** have 0.01°C resolution.
20. The temperature measurement **shall** have +- 0.4°C accuracy.
21. The system **shall** obtain the temperature calling the SHT_ObtainTemp function.
22. The SHT_ObtainTemp function **shall** not have any parameter.
23. The SHT_ObtainTemp function **shall** return the reading status E_OK or E_NOT_OK.

24. The SHT_ObtainTemp **shall** call to SHT_ReadTemperatureRaw function to request the temperature from the sensor.
25. The SHT_ReadTemperatureRaw function **shall** use a pointer as parameter to reference the data harvested address.
26. The SHT_ReadTemperatureRaw function **shall** return the sensor status with SHT_TASK_BUSY, SHT_TASK_OK or SHT_TASK_NOT_OK.
27. The humidity measurement **shall** have 0.05% HR resolution.
28. The humidity measurement **shall** have +- 3%HR accuracy.
29. The system **shall** obtain the humidity calling the SHT_ObtainRH function.
30. The SHT_ObtainRH function **shall** not have any parameter.
31. The SHT_ObtainRH function **shall** return the reading status E_OK or E_NOT_OK.
32. The SHT_ObtainRH **shall** call to SHT_ReadHumidityRaw function to request the humidity from the sensor.
33. The SHT_ReadHumidityRaw function **shall** use a pointer as parameter to reference the data harvested address.
34. The SHT_ReadHumidityRaw function **shall** return the sensor status with SHT_TASK_BUSY, SHT_TASK_OK or SHT_TASK_NOT_OK.
35. The system **shall** obtain the temperature measurement each 100 milliseconds.
36. To calculate the temperature, the system **shall** use the formulas provided by the OEM, these are shown on the data sheet of the sensor or in the section 2.5.1 from this document.
37. The system **shall** obtain the humidity measurement each 100 milliseconds.
38. To calculate the real relative humidity, the system **shall** use the formulas provided by the OEM, these are shown on the data sheet of the sensor or in the section 2.5.2 from this document.
39. The system **shall** calculate the dew point each 250-300 milliseconds.
40. The system **shall** calculate the dew point calling the SHT_CalculateDP function.
41. The SHT_CalculateDP function **shall** not have any parameter.
42. The SHT_CalculateDP function **shall** return the reading status E_OK or E_NOT_OK.

43. To calculate the real temperature, the system **shall** use the formulas provided by the OEM, these are shown on the data sheet of the sensor or in the section 2.5.3 from this document.
44. The microcontroller **shall** use the internal on-chip CAN peripheral module.
45. The microcontroller **shall** follow CAN 2.0A frame format.
46. The CAN module **shall** be configured to 100000 bps.
47. The microcontroller **shall** use 11 bits ID frames.
48. The microcontroller **shall** use CAN frame according UDS protocol.
49. The microcontroller **shall** use PC12 port as CAN RX to CAN transceiver (ATA6561).
50. The microcontroller **shall** use PC14 port as CAN TX to CAN transceiver (ATA6561).
51. The ID for the CAN message containing the measurements, also call "CatalyzerInfo" message **shall** be 0x0000150.
52. The temperature measurement **shall** be sent as a signal in the first byte (Byte0) with an 8-bit resolution in the CAN Message "CatalyzerInfo".
53. The temperature measurement **shall** be sent with a 0.1°C resolution.
54. The temperature measurement **shall** be represented in hexadecimal format.
55. The humidity measurement **shall** be sent as a signal in the second byte (Byte1) with an 8-bit resolution in the CAN Message "CatalyzerInfo".
56. The humidity measurement **shall** be sent with a 0.1% HR.
57. The humidity measurement **shall** be represented in hexadecimal format.
58. The value of the dew point **shall** be sent as a signal in the third byte (Byte2) with an 8-bit resolution in the CAN Message "CatalyzerInfo".
59. The dew point calculation **shall** be sent with a 0.1 HR resolution.
60. The dew point calculation **shall** be represented in hexadecimal format.
61. The function Can_SetSignal **shall** assigns the signal to be stored.
62. The frame **shall** be transmitted calling the CanNm_Tx function.
63. The CanNm_Tx function **shall** not have any parameter.
64. The CanNm_Tx function **shall** nor return any parameter.

3.2. Diagrams

3.2.1 Boundary Diagram

The following diagram (Figure 3-1) shows the internal modules from the microcontroller and ECU's components used by the system to obtain, process, and provide the measured data by the sensor.

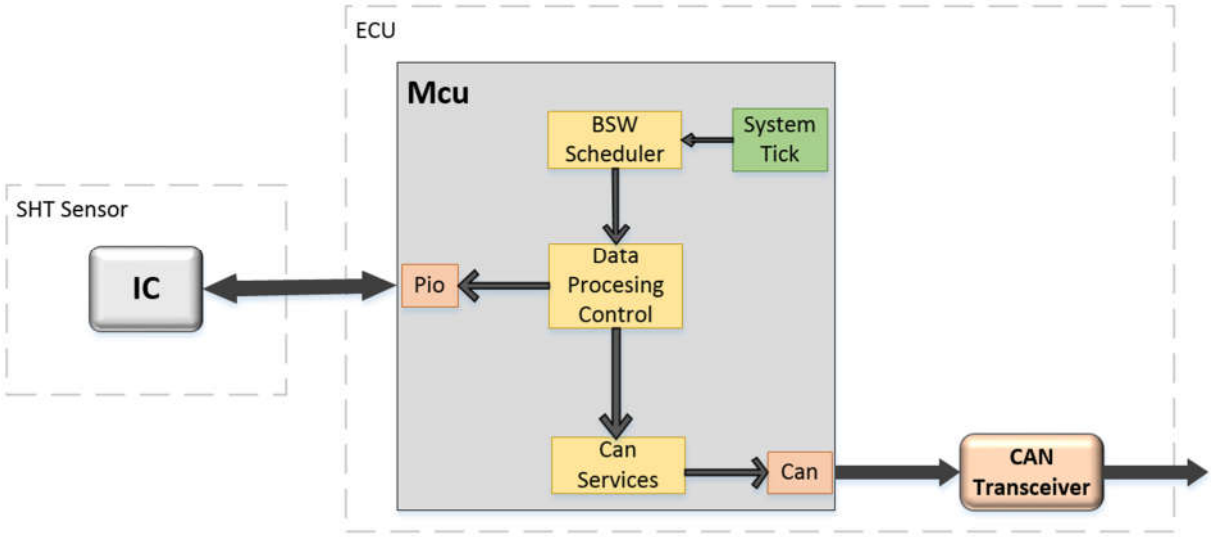


Figure 3-1. Boundary Diagram

3.2.2 System P-Diagram

The following Parameter Diagram (Figure 3-2) illustrates de actual inputs from the project by allowing a fast identification, including the different noise factors that shall be considered to reduce the possible risks and to achieve an optimal output.

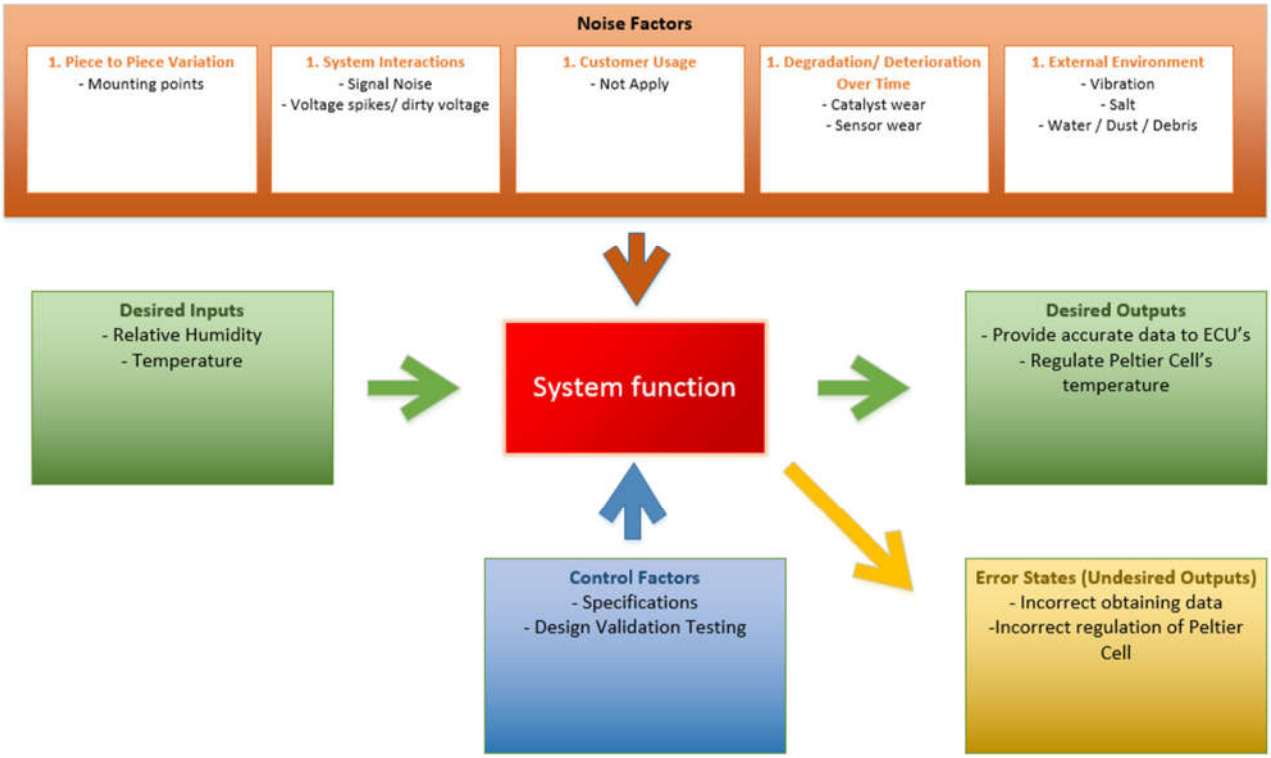


Figure 3-2. System P-Diagram.

3.2.3 Software P- Diagram

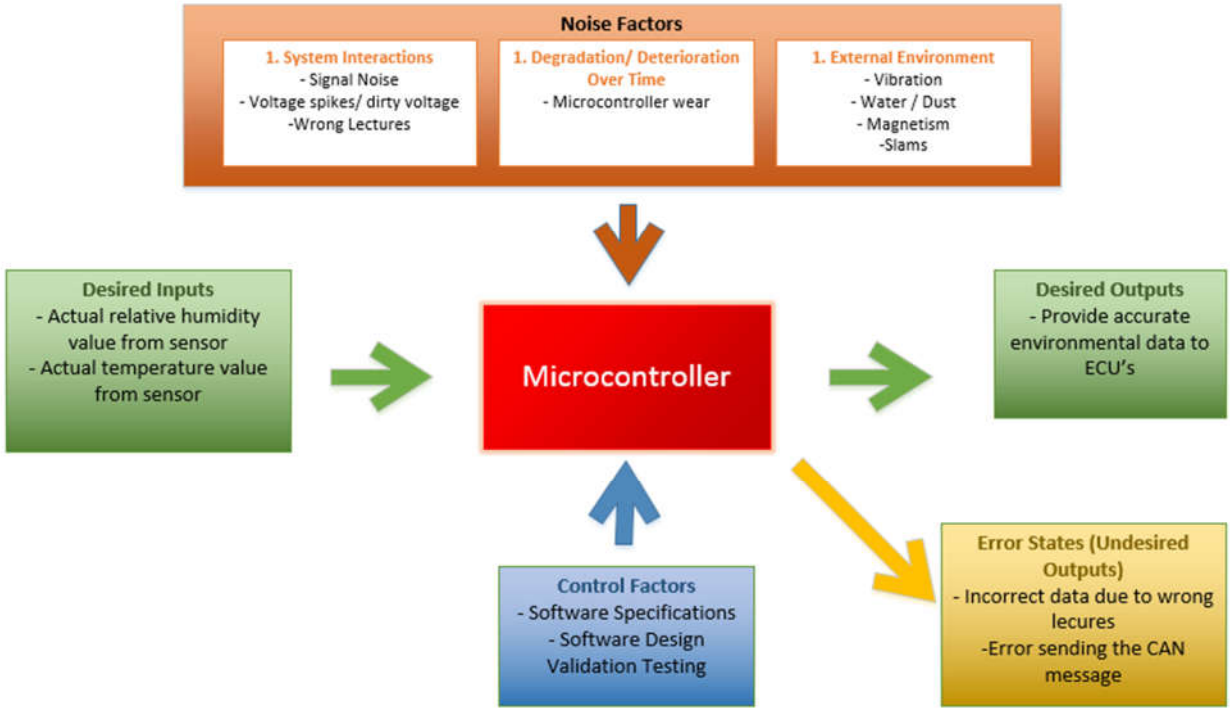


Figure 3-3. Software P-Diagram.

3.3. Software Architecture

The following picture (Figure 3-4) illustrates the software structure divided into subsystems, including components and modules defined from the AUTOSAR specifications segregated in layers as required by the system requirements.

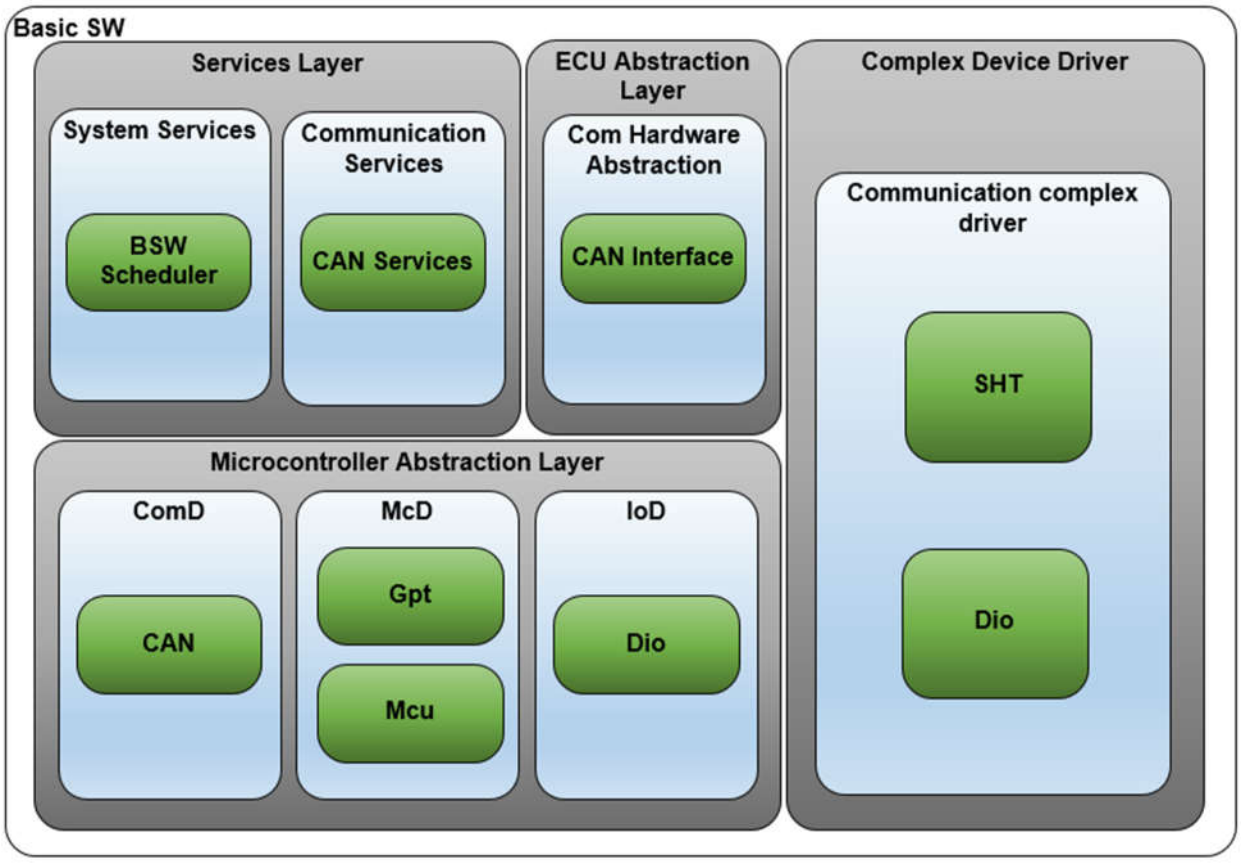


Figure 3-4. Software Architecture.

3.4. Hardware Design

Since an Xplained Ultra Evaluation Kit was implemented, no extra hardware is necessary to develop. It is only intended to wire the sensor connecting it with the board accordingly to the hardware needs.

3.5. Software Design

3.5.1 Static Decomposition of SW components

This section describes the principal functions and the different Application Program Interfaces (APIs) designed to support the requirements from the project and currently organized by component/unit.

3.5.1.1 Application/Sensor

3.5.1.1.1 SensEnv_Init

Service Syntax	**void SensEnv_Init (void)**
Description:	This function initializes variables used by the State Machine included in the sensor application witch internally calls initializing functions for lower-layer components.
Parameters:	N/A
Return Value:	N/A
File container:	SensingEnvironment.c

3.5.1.1.2 Process_Sensing_Env

Service Syntax	**void Process_Sensing_Env (void)**
Description:	This API handles a state machine that manages the sensor interaction, internally calling lower interfaces to communicate, get information from the sensor, and setting the required CAN signals.
Parameters:	N/A
Return Value:	N/A
File container:	SensingEnvironment.c

3.5.1.2 EcuAL/SHT

3.5.1.2.1 SHT_Init

Service Syntax	**void SHT_Init (void)**
Description:	This function initializes the sensor controller data and activate the required interrupts.
Parameters:	N/A
Return Value:	N/A
File container:	SHT.c

3.5.1.2.2 SHT_CalculateDP

Service Syntax	**Std_ReturnType SHT_CalculateDP (void)**
Description:	This function calculates the Dew Point
Parameters:	N/A
Return Value:	E_OK: The API succeeded in calculate the data. E_NOT_OK: An error is present while obtaining the result.
File container:	SHT.c

3.5.1.2.3 SHT_ObtainTemp

Service Syntax	**Std_ReturnType SHT_ObtainTemp (void)**
Description:	This function calculates the real temperature harvested from the sensor raw data.
Parameters:	N/A
Return Value:	E_OK: The reading of the data from the sensor has been succeeded. E_NOT_OK: The reading has not been achieved.
File container:	SHT.c

3.5.1.2.4 SHT_ObtainRH

Service Syntax	**Std_ReturnType SHT_ObtainRH (void)**
Description:	This function calculates the relative humidity with the raw data harvested from the sensor.
Parameters:	N/A
Return Value:	E_OK: The reading of the data from the sensor has been succeeded. E_NOT_OK: The reading has not been achieved.
File container:	SHT.c

3.5.1.2.5 SHT_ReadTemperatureRaw

Service Syntax	**static SHT_TaskStateType SHT_ReadTemperatureRaw(uint16_t *pData)**
Description:	This function calculates the temperature with the raw data harvested from the sensor.
Parameters:	pData: reference address to store the data harvested from the sensor
Return Value:	SHT_TASK_BUSY, SHT_TASK_OK, SHT_TASK_NOT_OK
File container:	SHT.c

3.5.1.2.6 SHT_ReadHumidityRaw

Service Syntax	**static SHT_TaskStateType SHT_ReadHumidityRaw(uint16_t *pData)**
Description:	This function calculates the relative humidity by using the raw data harvested from the sensor.
Parameters:	pData: reference address to store the data harvested from the sensor.
Return Value:	SHT_TASK_BUSY, SHT_TASK_OK, SHT_TASK_NOT_OK
File container:	SHT.c

3.5.1.3 McAL/Pio

3.5.1.3.1 PIO_Configure

Service Syntax	**uint8_t PIO_Configure(const Pin *list, uint32_t size)**
Description:	Configures a list of Pin instances, each list can either hold a single pin or a group of pins, depending on the mask value; all pins are configured by this function. The size of the array must also be provided and is easily computed using PIO_LISTSIZE whenever its length is not known in advance.
Parameters:	list: Pointer to a list of Pin instances size: Size of the Pin list (normally calculated using PIO_LISTSIZE)
Return Value:	1 if the pins have been configured properly; otherwise 0
File container:	pio.c

3.5.1.3.2 PIO_Set

Service Syntax	**void PIO_Set(const Pin *pin)**
Description:	Sets a high output level on all the PIOs defined in the given Pin instance. This has no immediate effects on PIOs that are not output, but the PIO controller will memorize the value they are changed to outputs.
Parameters:	Pointer to a Pin instance describing one or more pins
Return Value:	N/A
File container:	pio.c

3.5.1.3.3 PIO_Clear

Service Syntax	**void PIO_Clear(const Pin *pin)**
Description:	Sets a low output level on all the PIOs defined in the given Pin instance. This has no immediate effects on PIOs that are not output, but the PIO controller will memorize the value they are changed to outputs.
Parameters:	Pointer to a Pin instance describing one or more pins
Return Value:	N/A
File container:	pio.c

3.5.1.3.4 PIO_Get

Service Syntax	**uint8_t PIO_Get(const Pin *pin)**
Description:	This method returns the actual value that is being read at the pin. Returns 1 if one or more PIO of the given Pin instance, currently have a high level; otherwise returns 0.
Parameters:	Pointer to a Pin instance describing one or more pins
Return Value:	1 if the Pin instance contains at least one PIO that currently has a high level; otherwise 0.
File container:	pio.c

3.5.1.4 Services/CanNm

3.5.1.4.1 CanNm_Tx

Service Syntax	**void CanNm_Tx(void)**
Description:	This function manages the CAN transmission message to be sent every one second. This time is calibratable by the calibration variable "CAN_WAIT".
Parameters:	N/A
Return Value:	N/A
File container:	CanNm.c

3.5.1.4.2 Can_SetSignal

Service Syntax	**uint8_t Can_SetSignal(uint8_t Signal, uint8_t SignalValue)**
Description:	This function assigns a value to a CAN signal from the CAN message of the module.
Parameters:	- Signal: Specifies the signal in which it is desired to store the value - SignalValue: Value to assign.
Return Value:	N/A
File container:	CanNm.c

3.5.2 Dynamic behavior of software components

The following figure (Figure 3-5) represents the interaction between application and EcuAL layer during runtime. McAL has not been included to the flowchart due that there are several interfaces from this layer that would make the diagram uncomprehensive.

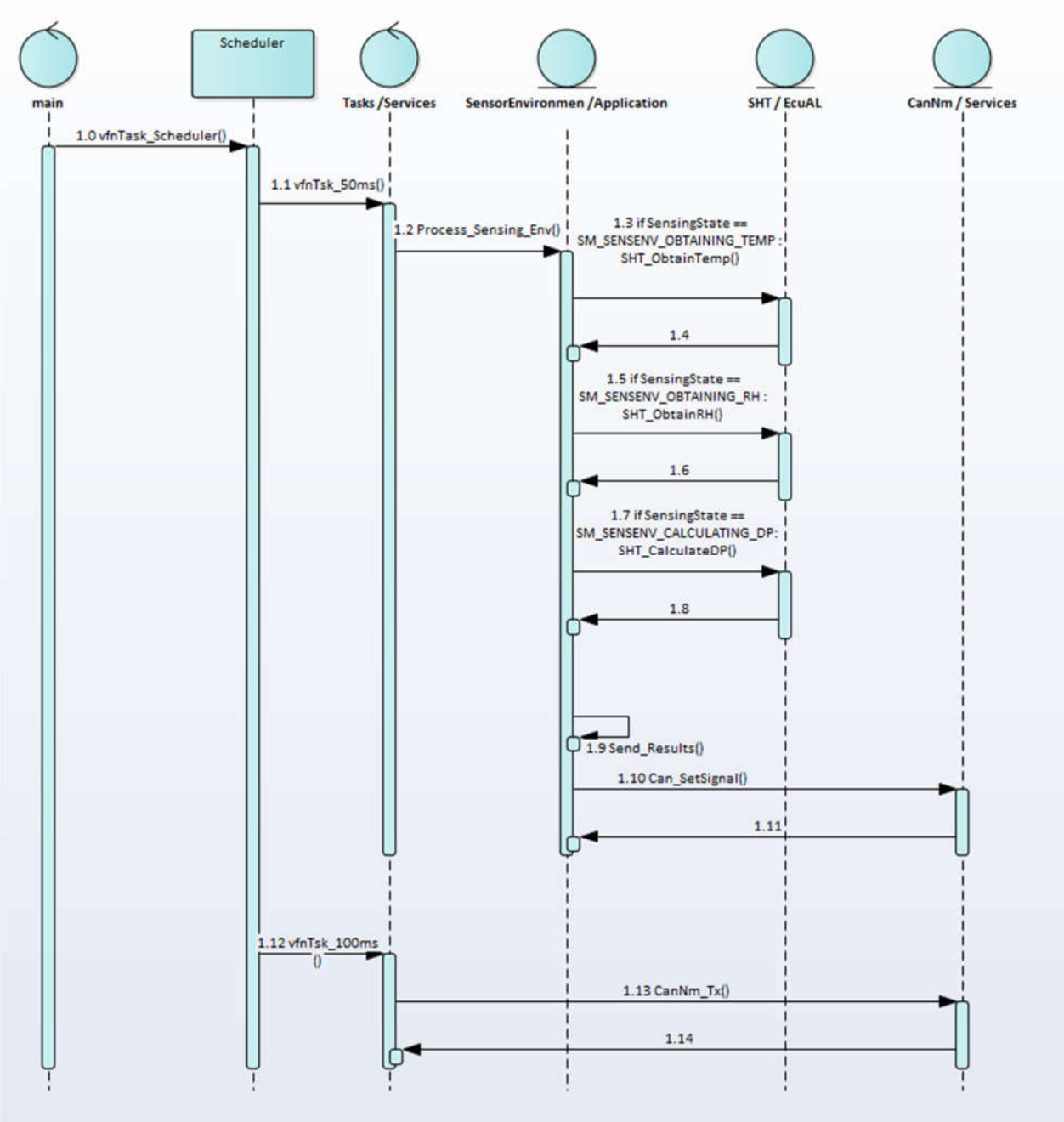


Figure 3-5. Dynamic Behavior from Operative System,

3.6. Memory Map

Table 3-1. Memory map

Memory region	Origin	End	Size
ROM (rx)	0x00400000	0x005FFFFF	0x00200000
RAM (rwx)	0x20400000	0x2044FFFF	0x00050000
RAM Heap	0x20450000	0x2045FFFF	0x00010000
SDRAM	0x70000000	0x701FFFFF	0x00200000

4. Results

The module was developed satisfying completely all the objectives required by the project´s sponsor, emphasizing the automotive grade requirements.

The hardware automotive grade was not only achieved by choosing the microcontroller and sensor according ISO-TS-16949, but also by using the software architecture complied according AUTOSAR specifications.

One of the challenges faced during the development phase was the implementation of the SHT11 communication's system. In this regard, the sensor did not follow a standardized communication protocol complicating the fast integration characteristic for an agile development mentioned by the manufacturer in the user's manual. The sensor´s behavior is described in the same manual, and the system must be AUTOSAR compliant, thus, the creation of one complex device driver to handle the communication with the sensor was mandatory.

Figure 4-1 shows one data frame sent by the sensor during one temperature request reading, the measurement simultaneously demonstrates the data (purple) and clock (yellow) pinouts from the microcontroller.

Figure 4-1. SHT11 temperature request frame

Once the first interaction was obtained, the analysis and interpretation of the information requested from the sensor was possible due to the formulas 2-1, 2-2 and 2-3, allowing to transform the raw data to readable information for the user or the system.

The system used the debug terminal from the SAMV71 to show the information obtained during runtime (Figure 4-2). This information is printed with the function "*Send_Results*" the SensingEnvironment.c file (Figure 4-3) producing a simple way to view the information continuously (Figure 4-4) before it is casted and sent through CAN bus as the system requirement requests (Figure 4-5).

Figure 4-2. Debug terminal

```
Data[0] = (uint8_t)SHTData.Temp;
Data[1] = (uint8_t)SHTData.RH;
Data[2] = (uint8_t)SHTData.DwPoint;
printf("%x  %x  %x\n\r",Data[0],Data[1],Data[2]);
printf("%f  %f  %f\n\r",SHTData.Temp,SHTData.RH,SHTData.DwPoint);
```

Figure 4-3. Data structures information obtained

Figure 4-4. Comparison between obtained and sent data

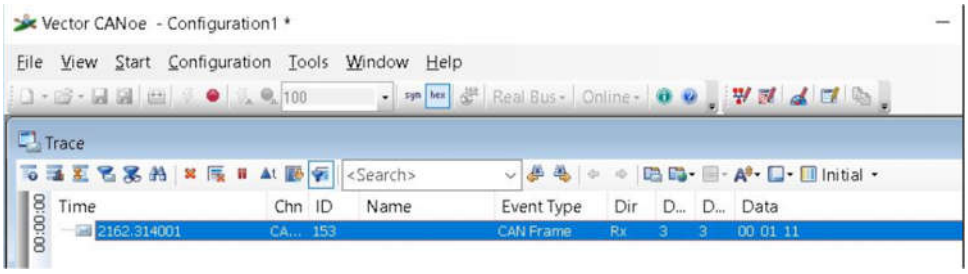


Figure 4-5. Data read through CAN terminal

In parallel, the CAN communication module acquires an inherit complexity by the AUTOSAR specification, granted that, implementing the transmission and reception functionalities was not an effortless task. Consequently, during the coding and requirements analysis, we determined that the information is only being sent and incoming messages are discarded, for this reason, we decided to remove and not waste any additional resources implementing the reception CAN message.

Conclusions

This project complied successfully with the CAN communication, RTOS, temperature, and humidity sensor specifications defined by our sponsor to achieve automotive grade quality.

One key factor to reach the specified requirements, was the implementation of the automotive SHT11 sensor, chosen due its electrical and physical characteristics, such as size, operating voltage, and building materials. For the usage of this sensor, a complex driver development was necessary, representing a challenge due to its complexity and unusual communication behavior. This driver follows the coding guidelines to fit the AUTOSAR architecture and can be integrated to any project for the data interchange, from application to basic software layers. By following these guidelines, the software increased the robustness, therefore, this provided a smooth maintenance and reduced sustainability. These features facilitate the integration of the module in a complete automotive system.

For the future research, new features are suggested to enhance the present module, such as implementing a Peltier-cell driver to control the temperature inside the catalyst, this will also include a PID controller based on the dew point, and a CO_2 sensor controller to harvest a major quantity of data to monitor from a greater scope, the way the catalyst behaves and improves the results. Furthermore, students and professionals that are encouraged to create a control module for an automotive-grade embedded system could use this module as a starting point since the present project intends to fulfill the automotive standard requirements, such as AUTOSAR, RTOS, and CAN.

References

[1] R. Jackson, "Global Climate Change: Effects," *Climate Change: Vital Signs of the Planet*. [Online]. Available: https://climate.nasa.gov/effects.

[2] "The Facts of Ozone Depletion," *National Geographic*, 09-Oct-2009. [Online]. Available: https://www.nationalgeographic.com/environment/global-warming/ozone-depletion/.

[3] "UNFCCC eHandbook." [Online]. Available: http://bigpicture.unfccc.int/.

[4] "Environmental aspects of the automotive industry - Crescita - European Commission," *Crescita*. [Online]. Available: /growth/sectors/automotive/environment-protection_it.

[5] "CO2 as Carbon Source for Fuel Synthesis - ScienceDirect." [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1876610214001398.

[6] "Directly converting CO2 into a gasoline fuel | Nature Communications." [Online]. Available: https://www.nature.com/articles/ncomms15174.

[7] "Air Pollution Causes, Effects, and Solutions," *National Geographic*, 09-Oct-2009. [Online]. Available: https://www.nationalgeographic.com/environment/global-warming/pollution/.

[8] S. Lin, T. Kiga, Y. Wang, and K. Nakayama, "Energy analysis of CaCO3 calcination with CO2 capture," *Energy Procedia*, vol. 4, pp. 356–361, Jan. 2011.

[9] "Frequently Asked Global Change Questions." [Online]. Available: http://cdiac.ess-dive.lbl.gov/faq.html.

[10] M. Ammann, H. P. Geering, C. H. Onder, C. A. Roduner, and E. Shafai, "Adaptive control of a three-way catalytic converter," in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*, 2000, vol. 3, pp. 1561–1566 vol.3.

[11] S. Prabhakar, M. Karthikeyan, K. Annamalai and V. N. Banugopan, "Control of emission characteristics by using Selective Catalytic Reduction (SCR) in D.I. diesel engine," *Frontiers in Automobile and Mechanical Engineering -2010*, Chennai, 2010, pp. 104-107.

[12] B. Qu and D. Fan, "Design of remote data monitoring and recording system based on ARM," in *2010 2nd International Conference on Industrial and Information Systems*, 2010, vol. 2, pp. 252–255.

[13] M. TéllezGirón-Enríquez, "Sistema embebido de control para la síntesis de CACO3 a partir de CO2 residual," *CACO3 synthesis from residual exhaust CO2 embedded control system*, Nov. 2016.

[14] ATSAMV71Q21 - 32-bit SAM Microcontrollers - Microcontrollers and Processors. [Online].Available: https://www.microchip.com/wwwproducts/en/ATSAMV71Q21.

[15] "SAM V71 Xplained Ultra Evaluation Kit." [Online]. Available: http://www.microchip.com/DevelopmentTools/ProductDetails/atsamv71-xult.

[16] "SHT1x (RH/T) - Digital Humidity Sensor | Sensirion." [Online]. Available: https://www.sensirion.com/en/environmental-sensors/humidity-sensors/digital-humidity-sensors-for-accurate-measurements/.

[17] A. Ltd, "Architecture | Overview of the Arm Architecture," *ARM Developer*. [Online]. Available: https://developer.arm.com/products/architecture/learn-about-the-arm-

architecture. [18] A. Ltd, “Cortex-M4,” *ARM Developer*. [Online]. Available: https://developer.arm.com/products/processors/cortex-m/cortex-m4.
[19] “AUTOSAR - Enabling Innovation.” [Online]. Available: https://www.autosar.org/.
[20] “CAN in Automation (CiA): CAN knowledge.” [Online]. Available: https://www.can-cia.org/can-knowledge/.
[21] “CAN in Automation (CiA): History of the CAN technology.” [Online]. Available: https://www.can-cia.org/can-knowledge/can/can-history/.
[22] D. Choi, M. Beardsley, D. Brzezinski, J. Koupal, and J. Warila, “MOVES Sensitivity Analysis: The Impacts of Temperature and Humidity on Emissions,” p. 15.
[23] “Heat and Temperature- Concepts.” [Online]. Available: http://hop.concord.org/h1/phys/h1p.html.
[24] “humidity | Definition of humidity in English by Oxford Dictionaries,” *Oxford Dictionaries | English*. [Online]. Available: https://en.oxforddictionaries.com/definition/humidity.
[25] “What causes humidity?,” *Scientific American*. [Online]. Available: https://www.scientificamerican.com/article/what-causes-humidity/.
[26] E. W. Weisstein, “Dewpoint -- from Eric Weisstein’s World of Physics.” [Online]. Available: http://scienceworld.wolfram.com/physics/Dewpoint.html.
[27] Renesas Electronics Corporation, "R8C Family – General RTOS Concepts" [Online]. Available: https://www.renesas.com/en-in/doc/products/tool/apn/res05b0008_r8cap.pdf. Apr. 2010.
[28] Stallings, W. (2012). *Operating systems*. 7th ed. Englewood Cliffs, N.J.: Prentice-Hall.

Appendices

The project is organized as follows:

-Source
- Application
  - main.c
  - Sensor
- Basic Software
- Infrastructure
  - Std_Types.h
  - Template.c
  - Template.h

main.c

```
/*----------------------------------------------------------------------------
 *        Headers
 *----------------------------------------------------------------------------*/

#include "board.h"
#include "app_scheduler.h"
#include "Tasks.h"
#include "MemAlloc.h"
#include "mcan.h"
#include "canif.h"
#include "fpu.h"
#include <stdbool.h>
#include <stdio.h>

/*----------------------------------------------------------------------------
 *        Local definitions
 *----------------------------------------------------------------------------*/

TaskType Tasks[]={
/*  TaskPriority    TaskId   TaskFunctionPointer   */
  {     5,       TASK_1MS,      vfnTsk_1ms    },
  {     4,       TASK_2MSA,     vfnTsk_2msA   },
  {     4,       TASK_2MSB,     vfnTsk_2msB   },
  {     3,       TASK_10MS,     vfnTsk_10ms   },
```

```
  {     2,      TASK_50MS,     vfnTsk_50ms   },
  {     1,      TASK_100MS,    vfnTsk_100ms  }
};

/*----------------------------------------------------------------------------
 *        Local functions
 *----------------------------------------------------------------------------*/

/**
 *  \brief Configure LEDs
 *
 *  Configures LEDs \#1 and \#2 (cleared by default).
 */
static void _ConfigureLeds( void )
{
        LED_Configure( 0 ) ;
        LED_Configure( 1 ) ;
}

static void _ConfigureComPins( void )
{
        ComPins_Configure( 0 ) ;
        ComPins_Configure( 1 ) ;
}
/*----------------------------------------------------------------------------
 *        Exported functions
 *----------------------------------------------------------------------------*/
/**
 *  \brief getting-started Application entry point.
 *
 *  \return Unused (ANSI-C compatibility).
 */
extern int main( void )
{
        /* Disable watchdog */
        WDT_Disable( WDT ) ;

        /* Output example information */
        printf( "\n\r-- Getting Started Example Workspace Updated!!! %s --\n\r",
SOFTPACK_VERSION ) ;
        printf( "-- %s\n\r", BOARD_NAME ) ;
```

```
        printf( "-- Compiled: %s %s With %s--\n\r", __DATE__, __TIME__ ,
COMPILER_NAME);

        /* Enable I and D cache */
        SCB_EnableICache();
    SCB_EnableDCache();
    /* Enable Floating Point Unit */
    vfnFpu_enable();

        printf( "Configure LED PIOs.\n\r" ) ;
        _ConfigureLeds() ;
        _ConfigureComPins() ;

    vfnTsk_Init();
    SensEnv_Init();
    MemAllocInit();
    printf( "\n\r-- Memory Allocation Initialized!!! --\n\r" ) ;

    CanIf_Init(1u, CanMsgObj);  /* Initializate the CAN hardware */
        /* Initialize Task Scheduler */
        vfnScheduler_Init(&Tasks[0]);
        /* Start execution of task scheduler */
        vfnScheduler_Start();

        /*-- Loop through all the periodic tasks from Task Scheduler --*/
        for(;;)
        {
                /* Perform all scheduled tasks */
                vfnTask_Scheduler();
        }

}
```

SensingEnvironment.c

```
/*****************************************************************************
**********************/
/**
\file       SensingEnvironment.c
\brief      Application level - Sensing Relative Humidity and
*                       Temperature of catalyst environment
\author     Juan Pablo Garcia
\version    1.0
\project    Catalyst Controller
\date       8/March/2018
*/
/*****************************************************************************
**********************/

/*****************************************************************************
***
 *                      Include Files

*****************************************************************************
**/
#include "Std_Types.h"
#include "SensingEnvironment.h"
#include "SHT_Types.h"
#include "SHT.h"
#include "CanNm.h"
//#include "SD_Mem.h"



/*****************************************************************************
**
 *                      Macro Definitions

*****************************************************************************
**/
typedef enum
{
        SM_SENSENV_INIT = 0,
        SM_SENSENV_OBTAINING_TEMP,
        SM_SENSENV_OBTAINING_RH,
```

```
        SM_SENSENV_CALCULATING_DP,
        SM_SENSENV_SPREAD_RESULT

} SM_SensEnv_Type;

enum
{
 CatalystDP = 0,
 CatalystTemp,
 CatalystRH
}CanSignal;


/*******************************************************************************
**
 *                      Global Variable Definitions

*******************************************************************************
**/

SM_SensEnv_Type SensingState;

uint16_t RealTemp;
uint16_t RealRH;

/*******************************************************************************
***
*                       Static Function Declarations
*******************************************************************************
**/
Std_ReturnType get_Temp (void);
Std_ReturnType get_RH (void);
Std_ReturnType Calculate_DP (void);
Std_ReturnType Send_Results (void);
/*******************************************************************************
***
 *                      Global and Static Function Definitions

*******************************************************************************
**/

/*******************************************************************************
```

```
 *
 * Function:
 *
 * Description:
 *
 * Caveats:
 *   None
 *

*****************************************************************************/
void SensEnv_Init (void)
{
        SensingState = SM_SENSENV_INIT;
        SHT_Init();
}

void Process_Sensing_Env (void)
{
        switch (SensingState)
        {
                case SM_SENSENV_INIT:

                        SensingState = SM_SENSENV_OBTAINING_TEMP;
                break;
                case SM_SENSENV_OBTAINING_TEMP:
                        if(E_OK == get_Temp())
                        {
                                SensingState = SM_SENSENV_OBTAINING_RH;
                        }
                break;
                case SM_SENSENV_OBTAINING_RH:
                        if(E_OK == get_RH())
                        {
                                SensingState = SM_SENSENV_CALCULATING_DP;
                        }
                break;
                case SM_SENSENV_CALCULATING_DP:
                        if(E_OK == Calculate_DP())
                        {
                                SensingState = SM_SENSENV_SPREAD_RESULT;
                        }
                break;
```

```
                case SM_SENSENV_SPREAD_RESULT:
                        if(E_OK == Send_Results())
                        {
                                SensingState = SM_SENSENV_OBTAINING_TEMP;
                        }
    else
    {
     SensingState = SM_SENSENV_INIT;
    }
                break;
                default:
                        SensingState = SM_SENSENV_INIT;
                break;
        }
}


Std_ReturnType get_Temp (void)
{
        Std_ReturnType status = E_NOT_OK;
        if(E_OK == SHT_ObtainTemp())
        {
                status = E_OK;
        }

        return status;
}

Std_ReturnType get_RH (void)
{
        Std_ReturnType status = E_NOT_OK;
        if(E_OK == SHT_ObtainRH())
        {
                status = E_OK;
        }
        return status;
}

Std_ReturnType Calculate_DP (void)
{
        Std_ReturnType status = E_NOT_OK;
        if (E_OK == SHT_CalculateDP())
```

```
        {
                status = E_OK;
        }
        return status;
}

Std_ReturnType Send_Results (void)
{
        Std_ReturnType status = E_OK;
        SHTDataType SHTData;
        uint8_t volatile Data[3];
        GetData(SHTData);
        Data[0] = (uint8_t)SHTData.Temp;
        Data[1] = (uint8_t)SHTData.RH;
        Data[2] = (uint8_t)SHTData.DwPoint;
 printf("%x  %x  %x\n\r",Data[0],Data[1],Data[2]);
 printf("%f  %f  %f\n\r",SHTData.Temp,SHTData.RH,SHTData.DwPoint);
        if(Can_SetSignal(CatalystDP,   Data[0])
 || Can_SetSignal(CatalystTemp, Data[1])
 || Can_SetSignal(CatalystRH,   Data[2]))
        {
                status = E_NOT_OK;
        }
 /*
        if(SD_Store(Data, sizeof(Data)))
        {
                status = E_NOT_OK;
        }
        */
        return E_OK;
}
```

SensingEnvironment.h

```
/*****************************************************************************
**********************/
/**
\file       SensingEnvironment.h
\brief      Mcu Abstraction level - Floating Point Unit initialization and configuration
\author     Juan Pablo Garcia
\version    1.0
\project    Catalyst Controller
\date       8/March/2018
*/
/*****************************************************************************
**********************/
#ifndef SENSINGENVIRONMENT_H
#define SENSINGENVIRONMENT_H
/*****************************************************************************
***
 *                      Include Files

*****************************************************************************
**/

/*****************************************************************************
***
*                       Macro Definitions
*****************************************************************************
**/

/*****************************************************************************
***
*                       Type Definitions
*****************************************************************************
**/

/*****************************************************************************
***
 *                      Global Variable Definitions

*****************************************************************************
**/
```

```
/*****************************************************************************
***
*                        Function Declarations
*****************************************************************************
**/
extern void SensEnv_Init (void);
extern void Process_Sensing_Env (void);

#endif /* SENSINGENVIRONMENT_H */
```

CanIf.c

```
#include "CanIf.h"

CanIf_MessageConfigType *ptrCanIfMessageConfig;
uint8_t CanNumberOfMsgs;
CanIf_PduType CanIf_Pdu;

const MCan_ConfigType *LogChannelId[] =
{
 &mcan0Config,
 &mcan1Config
};

/*
 CAN Initialization:
 1) This function shall initialize the Can channel according to CanChannelId parameter.
 2) Additionally, the function shall configure the CAN TX Messages
        as per the message configuration structure shown below.
*/
void CanIf_Init(uint8_t CanChannelId, CanIf_MsgObjType CanIfMsgConfig)
{
 CanNumberOfMsgs = CanIfMsgConfig.CanNumberOfMsgs;
 ptrCanIfMessageConfig =  CanIfMsgConfig.CanIfMessageConfig;
 const MCan_ConfigType *lcConfig;
 uint8_t i = 0u;

 lcConfig = LogChannelId[CanChannelId];  /* Gets the CAN configuration of the CAN channel
to be used */

 MCAN_Init(lcConfig);       /* Initializes the MCAN hardware */
 MCAN_Enable(lcConfig);

 for( i = 0; i< CanNumberOfMsgs; i++ )   /* Get and set the RAM address for SDU's */
 {
  CanIf_PduType *CanPdu = &ptrCanIfMessageConfig[i].CanPdu;     /* Gets the address of
every CAN PDU */
  CanPdu->CanSdu = MCAN_ConfigTxDedBuffer(lcConfig,i, CanPdu->CanId, CanPdu-
>CanIdType, CanPdu->CanDlc );   /* Gets the RAM address for every CAN Tx SDU */
 }
 //MCAN_ConfigRxBufferFilter(lcConfig, RX_BUFFER_0, FILTER_0,
MSG_ID_0,CAN_STD_ID);
```

```
  //MCAN_ConfigRxClassicFilter(lcConfig, CAN_FIFO_0, FILTER_1,
MSG_ID_2,CAN_STD_ID, MSG_ID_2_MASK);
}

/*
  CAN Transmission:

   1) This function shall transmit the Can Message according to MsgId
            on the CAN Bus as per the CanChannelId parameter.
*/
void CanIf_Transmit(uint8_t CanChannelId, uint8_t MsgId)
{
  CanIf_PduType *Pdu;
  const MCan_ConfigType *lcConfig;
  uint8_t i = 0u;

  lcConfig = LogChannelId[CanChannelId];  /* Gets the CAN configuration of the CAN channel
to be used */

  for(i = 0; i < CanNumberOfMsgs; i++)
  {
    if(MsgId ==  ptrCanIfMessageConfig[i].CanMsgIdNumber)
    {
      MCAN_SendTxDedBuffer(lcConfig,i);
    }
  }
}

void CanIf_Receive(uint8_t CanChannelId, uint8_t MsgId)
{
  CanIf_PduType *Pdu;
  const MCan_ConfigType *lcConfig;
  uint8_t i = 0u;

  lcConfig = LogChannelId[CanChannelId];  /* Gets the CAN configuration of the CAN channel
to be used */

  /* Poll for new CAN messages in RX FIFO *
          for(i = 0; i < CanNumberOfMsgs; i++)
  {
                    fifo_entries = MCAN_GetRxFifoBuffer(&mcan0Config,
CAN_FIFO_0,(Mailbox64Type *) &rxMailbox2);
```

```
            if (fifo_entries > 0)
        rxdCntr++;
    } while (fifo_entries > 1);
    */
}
```

CanIf.h

```
#ifndef _CAN_IF_
#define _CAN_IF_

#include "CanIf_Cfg.h"
#include "CanIf_Types.h"

typedef struct
{
  uint32_t CanId;
  MCan_IdType CanIdType;
  MCan_DlcType CanDlc;
  uint8_t* CanSdu;
}CanIf_PduType;

typedef struct
{
   uint8_t CanMsgIdNumber;
   MCan_DirType MCanDir;
   CanIf_PduType CanPdu;
}CanIf_MessageConfigType;

typedef struct
{
   uint8_t CanNumberOfMsgs;
   CanIf_MessageConfigType *CanIfMessageConfig;
}CanIf_MsgObjType;

extern CanIf_MsgObjType CanMsgObj;

void CanIf_Init(uint8_t CanChannelId, CanIf_MsgObjType CanIfMsgConfig);
void CanIf_Transmit(uint8_t CanChannelId, uint8_t MsgId);

#endif /* _CAN_IF_ */
```

CanIf_Cfg.c

```
#include <stdio.h>
#include <stdint.h>

#include "CanIf.h"

CanIf_MessageConfigType CanTxMsg[] =
{
  { 0, CAN_TX, { 0x153,      CAN_STD_ID, CAN_DLC_3, (uint8_t *)0} }, /* CAN message */
  { 1, CAN_TX, { 0x123456A0, CAN_EXT_ID, CAN_DLC_6, (uint8_t *)0} },
};

CanIf_MsgObjType CanMsgObj =
{
 sizeof(CanTxMsg)/sizeof(CanIf_MessageConfigType),
 CanTxMsg
};
```

CanIf_Cfg.h

```
#ifndef _CAN_CFG_
#define _CAN_CFG_

#endif /* _CAN_CFG_ */
```

CanIf_Types.h

```
#ifndef _CANIF_TYPES_
#define _CANIF_TYPES_

#include "board.h"

typedef enum
{
  CAN_TX = 0,
  CAN_RX = 1,
}MCan_DirType;

typedef enum
{
  CAN_MSGID_0 = 0,
  CAN_MSGID_1 = 1,
  CAN_MSGID_2 = 2,
  CAN_MSGID_3 = 3
}MCan_MsgID;

typedef enum
{
  ControllerId_0 = 0,
  ControllerId_1 = 1,
}MCan_LogCanInt;

#endif /* _CANIF_TYPES_ */
```

SHT.c

```
/*****************************************************************************
**********************/
/**
\file       SHT.c
\brief      ECU Abstraction level - Sensirion Sensor Abstraction
\author     Jesus Iñiguez
\version    1.0
\project    Catalyst Controller
\date       8/March/2018
*/
/*****************************************************************************
**********************/

/*****************************************************************************
***
 *                      Include Files

*****************************************************************************
**/
#include "Std_Types.h"
#include "SHT_Types.h"
#include "SHT.h"
#include "board.h"
#include "math.h"
#include "pmc.h"
#include "sysclk.h"

//#define InterruptMode

/*****************************************************************************
**
 *                      Macro Definitions

*****************************************************************************
**/
static const Pin pinsSHTs[] = {PIN_TWI_TWD0, PIN_TWI_TWCK0};
#define TWIHS_CLK     100000

#define noACK 0
#define ACK 1
//adr command r/w
#define STATUS_REG_W  0x06 //000 0011 0
#define STATUS_REG_R  0x07 //000 0011 1
#define MEASURE_TEMP  0x03 //000 0001 1
```

```
#define MEASURE_HUMI  0x05 //000 0010 1
#define RESET         0x1e //000 1111 0
#define SDA           0x00
#define SCK           0x01
////////////////////////////////////////////////////////////////////////
// Command byte values        adr cmd  r/w
#define Reset         0x1e   // 000 1111  0
#define MeasureTemp   0x03   // 000 0001  1
#define MeasureHumi   0x05   // 000 0010  1
#define WrStatusReg   0x06   // 000 0011  0
#define RdStatusReg   0x07   // 000 0011  1

/*****************************************************************************
**
 *                      Prototypes Definitions

*****************************************************************************
**/
  uint8_t  s_write_byte(uint8_t value);
  uint16_t ReadValueFn();
  uint16_t ReadHumidity();
  uint16_t ReadTemperature();
  uint8_t  SwapData(uint8_t data);

static SHT_TaskStateType SHT_ReadTemperatureRaw(uint16_t *pData );
static SHT_TaskStateType SHT_ReadHumidityRaw(uint16_t *pData);
static float log_n(float x);
static float power( float dwX, uint32_t dwY );
extern void PIN_MakeOutputSDA( );
extern void PIN_MakeInputSDA( );
extern void PIN_Clear( uint32_t dwCom );
extern void PIOHandlesFunction(void);

/*****************************************************************************
**
 *                      Global Variable Definitions

*****************************************************************************
**/

  SHTDataType SHTData;
  uint8_t     TimeOut,err;
  uint8_t     AckConfirmVar=0u;
  uint8_t     SwitchInterrupt=0;
#ifdef InterruptMode
```

```
  uint8_t   RequestEnabled;
  uint16_t  MeasuredValues[2];//0 Temp 1 Humidity
  uint8_t   InterruptVar=0;
#endif
/*****************************************************************************
***
*                         Static Function Declarations
*****************************************************************************
**/

uint32_t ComPins_Configure( uint32_t dwCom )
{
        return ( PIO_Configure( &pinsSHTs[dwCom], 1 ) );
}




void PIOA_Handler()
{
  NVIC_DisableIRQ( PIOA_IRQn ) ;
  AckConfirmVar=1;
  PIOHandlesFunction();
#ifdef InterruptMode

  if(1u==InterruptVar)
  {
    if(MeasureTemp == RequestEnabled)
    {
      RequestEnabled = 0xff;
      MeasuredValues[0]= ReadValueFn();

    }
    else if(MeasureHumi == RequestEnabled)
    {
      RequestEnabled = 0xff;
      MeasuredValues[1]= ReadValueFn();
    }
    InterruptVar=0u;
  }
#endif
  /*PIOA->PIO_IER |= 0x08;     //PIO_IDR
  PIOA->PIO_ESR |= 0x08; //edge PIO_ESR
                  //LSR level
  PIOA->PIO_FELLSR |= 0x08; //falling edge
```

```
  PIOA->PIO_IFER |= 0x08;
  PIOA->PIO_IFSCER |= 0x04;
    */
  NVIC_EnableIRQ( PIOA_IRQn ) ;
}

/*****************************************************************************
 *
 * Function:        log_n
 *
 * Description:     Returns the natural logarithm of val
 *
 * Caveats:
 *   None
 *
*****************************************************************************/
static float log_n(float val)
{
   uint8_t n;
   float x, logNat;
   /*val = (1 + x)*/
   x = (val - 1.0);
   logNat = 0.0;
   if(val > 0 && val <= 1)
   {
      for(n = 0; n < 17; n++)
      {
         logNat += (power((-1),n) * (power(x,(n+1)))/(n+1));
      }
   }
   return logNat;
}


/*****************************************************************************
 *
 * Function:        power
 *
 * Description:     Returns the value of dwX powered to the exponential dwY
 *
 * Caveats:
 *   None
 *
```

```
*****************************************************************************/
static float power( float dwX, uint32_t dwY )
{
   float dwResult = 1 ;

   while ( dwY > 0 ) {
      dwResult *= dwX ;
      dwY-- ;
   }
   return dwResult ;
}
/*****************************************************************************
***
 *                          Global and Static Function Definitions

*****************************************************************************
**/


/*****************************************************************************
 *
 * Function:        SHT_Init
 *
 * Description:     Initialize the sensor controller
 *
 * Caveats:
 *   None
 *

*****************************************************************************/
 void SHT_Init (void)
{
        /*Initializing Variables*/
        SHTData.RH = 0.0;
        SHTData.Temp = 0.0;
        SHTData.DwPoint = 0.0;
        PMC_EnablePeripheral(ID_PIOA);
        NVIC_DisableIRQ( PIOA_IRQn ) ;
        NVIC_ClearPendingIRQ( PIOA_IRQn ) ;
        NVIC_SetPriority( PIOA_IRQn, 1 ) ;
        NVIC_EnableIRQ( PIOA_IRQn ) ;
}
```

```
/*****************************************************************************
 *
 * Function:       SHT_ObtainTemp
 *
 * Description:    This function calculates the Temperature obtained from the sensor
 *
 * Caveats:
 *   None
 *
*****************************************************************************/
Std_ReturnType SHT_ObtainTemp (void)
{
        uint16_t SO_T;
        Std_ReturnType status = E_NOT_OK;
        if(SHT_TASK_OK == SHT_ReadTemperatureRaw(&SO_T))
        {
                /*Given the Temp raw value we can now calculate the real temperature *
                * the temperature sensor is very linear by design, according to the  *
                * datasheet the formula is T = d1 + (d2 * SOt), where d1 and d2 are  *
                * coefficients and SOt the digital readout                          */
                SHTData.Temp = (SHT_SO_TEMP_D1 + (SHT_SO_TEMP_D2 * SO_T));
   printf("%f\n\r",SHTData.Temp );
   status = E_OK;
        }
        return status;
}


/*****************************************************************************
 *
 * Function:       SHT_ObtainRH
 *
 * Description:    This function calculates the Relation Humidity from the sensor
 *
 * Caveats:
 *   None
 *
*****************************************************************************/
Std_ReturnType SHT_ObtainRH (void)
{
        uint16_t SO_RH;
        Std_ReturnType status = E_NOT_OK;
```

```
        if(SHT_TASK_OK ==SHT_ReadHumidityRaw(&SO_RH))
        {
                float RH_Linear;
                /*Given the RH raw value and having calculated the real temperature  *
                * we can now calculate the real relative humidity. For compensating  *
                * non-linearity of the humidity sensor it is recommended to convert  *
                * the humidity readout (SOrh) wiht the following formula with        *
                * coefficients given by the datasheet:                               *
                *         RH_Linear = c1 + (c2 * SOrh) + (c3 * (SOrh)^2)             *
                * For temperatures significantly different from 25°C (~77°F)         *
                * the humidity signal requires temperature compensation.             *
                * The temperature correction corresponds roughly to                  *
                * 0.12%RH/°C @ 50%RH. Coefficients for the temperature               *
                * compensation are given by the datasheet:                           *
                *         RH = (T[°C] -25) * (t1 + (t2 * SO_RH)) + RH_Linear       */


   RH_Linear = (-2.0468 + (SHT_SO_HR_C2 * SO_RH) +
                        ((-0.0000015955) * SO_RH * SO_RH));

                SHTData.RH = ((SHTData.Temp - 25u) * (SHT_SO_HR_T1 +
                         (SHT_SO_HR_T2 * SO_RH)) + RH_Linear);
   printf("%f\n\r",SHTData.RH );
                status = E_OK;
        }
        return status;
}


/*****************************************************************************
 *
 * Function:       SHT_CalculateDP
 *
 * Description:    This function returns the calculated dew point from the sensor
 *
 * Caveats:
 *   None
 *

*****************************************************************************/
Std_ReturnType SHT_CalculateDP (void)
{
  float Tn, m, partial1;
  float  Ln_RH;
  int32_t number, result;
```

```
  char ln_result[50];
  /*For dew point (Td) calculations there are various formulas                    *
  * to be applied, most of them quite complicated. For the                      *
  * temperature range of -40 – 50°C the following                               *
  * approximation provides good accuracy with parameters given by the           *
  * datasheet.                                                                  *
  * Td(RH,T) = Tn * (Ln(RH/100%) + (m * T)/(Tn + T) / (m - Ln(RH/100%) - (m * T)/(Tn +
T))) */

  if(SHTData.Temp >= 0) /*Above water*/
  {
    Tn = SHT_DP_TN_ABV_WATER;
    m = SHT_DP_M_ABV_WATER;
  }
  else /*Above ice*/
  {
    Tn = SHT_DP_TN_ABV_ICE;
    m = SHT_DP_M_ABV_ICE;
  }

  Ln_RH = log_n(SHTData.RH / 100.0);
  partial1 =(float)((m * SHTData.Temp) / (Tn + SHTData.Temp));

  SHTData.DwPoint = (Tn * ((Ln_RH + partial1) / (m - Ln_RH - partial1)));
    printf("%f\n\r",SHTData.DwPoint);
  return E_OK;
}


/*****************************************************************************
 *
 * Function:        GetData
 *
 * Description:     Returns the value of the structure Data under the parameter
 *                  as reference
 *
 * Caveats:
 *   None
 *

*****************************************************************************/
void GetData(SHTDataType *Data)
{
        Data = &SHTData;
}
```

```
/*****************************************************************************
 *
 * Function:        SHT_ReadTemperatureRaw
 *
 * Description:     This function make a reading of the raw temperature over
 *                  the sensor
 *
 * Caveats:
 *   None
 *

*****************************************************************************/
static SHT_TaskStateType SHT_ReadTemperatureRaw(uint16_t *pData)
{
 uint16_t Data=0;
          SHT_TaskStateType status = SHT_TASK_NOT_OK;
 if(0u==SwitchInterrupt)
 {
  Data=ReadTemperature();
  if((Data&0x8000)==0x8000)
  {
    *pData=Data-0x8000;
    status = SHT_TASK_OK;
  }
  SwitchInterrupt++;
 }
 else
 {
  #ifdef InterruptMode
  RequestEnabled = MeasureTemp; //RequestEnabled = MeasureHumi;
  #endif
  SwitchInterrupt=0u;
 }
 return  status;
}


/*****************************************************************************
 *
 * Function:        SHT_ReadHumidityRaw
 *
 * Description:     This function make a reading of the raw humidity over
 *                  the sensor
```

```
 *
 * Caveats:
 *   None
 *

*****************************************************************************/
static SHT_TaskStateType SHT_ReadHumidityRaw(uint16_t *pData)
{
 uint16_t Data=0;
         SHT_TaskStateType status = SHT_TASK_NOT_OK;
 if(0u==SwitchInterrupt)
 {
  Data=ReadHumidity();
  if((Data&0x8000)==0x8000)
  {
    *pData=Data-0x8000;
    status = SHT_TASK_OK;
  }
  SwitchInterrupt++;
 }
 else
 {
  #ifdef InterruptMode
  RequestEnabled = MeasureTemp; //RequestEnabled = MeasureHumi;
  #endif
  SwitchInterrupt=0u;
 }
 return  status;
}

void s_delay_2_us()
{
 asm("NOP");
 asm("NOP");
}

void TransmitStart()
{
 PIN_Clear( SDA );
 PIN_Set(SCK);     s_delay_2_us();
 PIN_Clear(SCK);   s_delay_2_us();
 PIN_Set(SDA);     s_delay_2_us();
 PIN_Set(SCK);     s_delay_2_us();
 PIN_Clear(SCK);   s_delay_2_us();
 PIN_Clear(SDA);   s_delay_2_us();
```

```
  PIN_Set(SCK);     s_delay_2_us();
}

uint16_t ReadValueFn()
{
 uint8_t i, ByteHigh=0, ByteLow=0,SwpHigh=0,SwpLow=0;
 uint16_t Lx=0,ActualValue=0;

  for(i=1;i<=8;++i){                  // read high byte VALUE from SHT11
    PIN_Clear(SCK);
    s_delay_2_us();
    /*adquisition value*/
    ActualValue=(PIOA->PIO_PDSR&PIO_PDSR_P3)>>3;
    ByteHigh|= ((ActualValue)<<i);
    PIN_Set(SCK);
    s_delay_2_us();
  }
  PIN_MakeOutputSDA( );
  PIN_Set(SDA);
  s_delay_2_us();
  PIN_Clear(SCK);
  s_delay_2_us();
  PIN_Set(SCK);
  PIN_MakeInputSDA( );
  s_delay_2_us();
  ActualValue=0;
  for(i=1;i<=8;++i){                  // read low byte VALUE from SHT11
    PIN_Clear(SCK);
    s_delay_2_us();
    ActualValue=(PIOA->PIO_PDSR&PIO_PDSR_P3)>>3;
    ByteLow|= ((ActualValue)<<i);
    /*adquisition value*/
    PIN_Set(SCK);
    s_delay_2_us();
  }
  PIN_MakeOutputSDA( );
  PIN_Clear(SDA);
  s_delay_2_us();
  PIN_Clear(SCK);
  s_delay_2_us();
  PIN_Set(SCK);
  s_delay_2_us();
  ByteHigh=SwapData(ByteHigh);
  ByteLow=SwapData(ByteLow);
  Lx=(((uint16_t)ByteHigh<<8)|ByteLow)<<1;
```

```
  return(Lx);
}

uint8_t SwapData(uint8_t data)
{
  uint8_t tempData=0x00;
  tempData |= (data&0x80)>>7;
  tempData |= (data&0x40)>>5;
  tempData |= (data&0x20)>>3;
  tempData |= (data&0x10)>>1;
  tempData |= (data&0x08)<<1;
  tempData |= (data&0x04)<<3;
  tempData |= (data&0x02)<<5;
  tempData |= (data&0x01)<<7;
  return tempData;
}
//////////////////////////////////////////////////////////
void SendCommand(int Command){
 uint8_t i;

  for(i=128;i>0;i/=2){
     if(i & Command)
     {
       PIN_Clear(SDA);
     }
     else
     {
       PIN_Set(SDA);
     }
     PIN_Clear(SCK);
     s_delay_2_us();
     PIN_Set(SCK);
     s_delay_2_us();
  }
 /*Change pin mode to input*/
  PIN_MakeInputSDA();
  PIN_Clear(SCK);
  s_delay_2_us();
  PIN_Set(SCK);
  s_delay_2_us();

}

//////////////////////////////////////////////////////////
#ifdef InterruptMode
```

```
uint16_t ReadTemperature()
{
 uint8_t AcknowPINge;
 uint16_t Lx=0u,Value,DummyCounter;
 TransmitStart();
 SendCommand(MeasureTemp);
 DummyCounter=0x0000;
 while(0u==AckConfirmVar||DummyCounter<0xffff)
 {
  DummyCounter++;
 }/*wait for acknolwledge*/
 if(DummyCounter!=0xffff)
 {
  AckConfirmVar=0u;
  s_delay_2_us();
  PIN_MakeInputSDA();
  InterruptVar=1u;
 }

 //Value=CalcTempValues(Lx);
 return(Value);
}

uint16_t ReadHumidity()
{
 uint16_t Lx=0u,Value,DummyCounter;
 TransmitStart();
 SendCommand(MeasureHumi);

 while(0u==AckConfirmVar);      /*wait for acknolwledge*/
 AckConfirmVar=0u;
 s_delay_2_us();
 PIN_MakeInputSDA();
 DummyCounter=0;
 while(0u==AckConfirmVar||DummyCounter<0xffff)
 {
  DummyCounter++;
 }/*wait for acknolwledge*/
 if(DummyCounter!=0xffff)
 {
  AckConfirmVar=0u;
  s_delay_2_us();
  PIN_MakeInputSDA();
  InterruptVar=1u;
```

```
  }
  return(Value);
}

#else    //polling mode

uint16_t ReadTemperature()
{
  uint8_t AcknowPINge;
  uint16_t DummyCounter=0;
  long Lx=0u,Value;
  TransmitStart();
  SendCommand(MeasureTemp);
  while(0u==AckConfirmVar);/*wait for acknolwledge*/
  //while(0u==AckConfirmVar||DummyCounter<0xffff)
  //{
  //  DummyCounter++;
  //}/*wait for acknolwledge*/
  AckConfirmVar=0u;
  s_delay_2_us();
  PIN_MakeInputSDA();
  //while(0u==AckConfirmVar||DummyCounter<0xffff)
  //{
  //  DummyCounter++;
  //}/*wait for acknolwledge*/
  while(0u==AckConfirmVar);/*wait for acknolwledge*/
  AckConfirmVar=0u;
  Lx=ReadValueFn();
  //Value=CalcTempValues(Lx);
  return(Lx+0x8000);
}

uint16_t ReadHumidity(){
  uint8_t AcknowPINge;
  long Lx=0u,Value;
  TransmitStart();
  SendCommand(MeasureHumi);
  while(0==AckConfirmVar);/*wait for acknolwledge*/
  AckConfirmVar=0;
  s_delay_2_us();
  PIN_MakeInputSDA();
  while(0u==AckConfirmVar);/*wait for acknolwledge*/
  AckConfirmVar=0u;
  Lx=ReadValueFn();
  //Value=CalcTempValues(Lx);
```

```
  return(Lx+0x8000);
}
#endif
//////////////////////////////////////////////////////
//
//////////////////////////////////////////////////////

void RST_Connection(){
 int i;

  PIN_Clear(SDA);
  for(i=1;i<=10;++i) {
     PIN_Clear(SCK); s_delay_2_us();
     PIN_Set(SCK);  s_delay_2_us();
  }
  TransmitStart();
}
```

SHT.h

```
/*****************************************************************************
**********************/
/**
\file      SHT.h
\brief     ECU Abstraction level - Sensirion Sensor Abstraction
\author    Juan Pablo Garcia
\version   1.0
\project   Catalyst Controller
\date      8/March/2018
*/
/*****************************************************************************
**********************/
#ifndef SHT_H
#define SHT_H
/*****************************************************************************
***
 *                      Include Files

*****************************************************************************
**/

/*****************************************************************************
***
*                      Macro Definitions
*****************************************************************************
**/
/*Coefficients given by table 9 from datasheet.*/
/*Temperature*/
#define SHT_SO_TEMP_D1     (-39.7f)            /*Using a Vdd = 5V*/
#define SHT_SO_TEMP_D2     (0.01f)             /*Using the 12bit configuration*/

/*Relative Humidity*/
#define SHT_SO_HR_C1       (-2.0468f)     /*Using the 8bit configuration*/
#define SHT_SO_HR_C2       (0.0367f)      /*Using the 8bit configuration*/
#define SHT_SO_HR_C3       (-1.5955E-6)  /*Using the 8bit configuration*/
#define SHT_SO_HR_T1       (0.01f)      /*Using the 8bit configuration*/
#define SHT_SO_HR_T2       (0.00008f)   /*Using the 8bit configuration*/

/*Dew Point*/
#define SHT_DP_TN_ABV_WATER          (243.12f) /*Temperature range 0 - 50 °C*/
#define SHT_DP_M_ABV_WATER           (17.62f)  /*Temperature range 0 - 50 °C*/
#define SHT_DP_TN_ABV_ICE           (272.62f) /*Temperature range -40 - 0 °C*/
#define SHT_DP_M_ABV_ICE            (22.46f)  /*Temperature range -40 - 0 °C*/
```

```
/*Status register Bits*/
#define SHT_STS_HTR_ON          (0x1u << 2) /*Set heater ON*/
#define SHT_STS_HTR_OFF (0x0u << 2) /*Set heater OFF*/
#define SHT_STS_RES_MIN (0x1u << 0) /*Set min resolution: 8bit RH / 12bit Temp*/
#define SHT_STS_RES_MAX (0x0u << 0) /*Set max resolution: 12bit RH / 14bit Temp*/

/*SHT Spi Commands*/
#define SHT_CMD_MEASURE_TEMP     (0x03u)
#define SHT_CMD_MEASURE_RH              (0x05u)
#define SHT_CMD_STS_WRITE        (0x06u)
#define SHT_CMD_STS_READ         (0x07u)
#define SHT_SFT_RST              (0x1Eu)

#define noACK 0
#define ACK   1
                    //adr  command  r/w
#define STATUS_REG_W 0x06   //000   0011    0
#define STATUS_REG_R 0x07   //000   0011    1
#define MEASURE_TEMP 0x03   //000   0001    1
#define MEASURE_HUMI 0x05   //000   0010    1
#define RESET       0x1e   //000   1111
/*****************************************************************************
***
*                       Type Definitions
*****************************************************************************
**/

/*****************************************************************************
***
 *                       Global Variable Definitions

*****************************************************************************
**/

/*****************************************************************************
***
*                       Function Declarations
*****************************************************************************
**/
extern void SHT_Init (void);
extern Std_ReturnType SHT_CalculateDP (void);
extern Std_ReturnType SHT_ObtainTemp (void);
extern Std_ReturnType SHT_ObtainRH (void);

#endif /* SHT_H */
```

SHT_Types.h

```
/*****************************************************************************
**********************/
/**
\file       Name.h
\brief      Mcu Abstraction level - Floating Point Unit initialization and configuration
\author     Juan Pablo Garcia
\version    1.0
\project    Catalyst Controller
\date       8/March/2018
*/
/*****************************************************************************
**********************/

/*****************************************************************************
***
 *                          Include Files

*****************************************************************************
**/
 #include <stdint.h>
/*****************************************************************************
***
*                          Macro Definitions
*****************************************************************************
**/


/*****************************************************************************
***
*                          Type Definitions
*****************************************************************************
**/
typedef enum
{
        SHT_TASK_BUSY = 0,
        SHT_TASK_OK,
        SHT_TASK_NOT_OK
} SHT_TaskStateType;

typedef struct
{
        float RH;
        float Temp;
        float DwPoint;
```

```
}SHTDataType;
```

mcan.c

```
/* ----------------------------------------------------------------------------
 *         SAM Software Package License
 * ----------------------------------------------------------------------------
 * Copyright (c) 2012, Atmel Corporation
 *
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * - Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the disclaimer below.
 *
 * Atmel's name may not be used to endorse or promote products derived from
 * this software without specific prior written permission.
 *
 * DISCLAIMER: THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY
EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-
INFRINGEMENT ARE
 * DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT,
INDIRECT,
 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA,
 * OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * ----------------------------------------------------------------------------
 */

/** \file
 *  Implements functions for Controller Area Network (CAN)
 *  peripheral operations.
 */
/** \addtogroup can_module
 *@{*/
```

```
/*----------------------------------------------------------------------------
 *        Headers
 *----------------------------------------------------------------------------*/
#include "board.h"
#include "chip.h"
#include "mcan_config.h"
#include <assert.h>
/*----------------------------------------------------------------------------
 *      Definitions
 *----------------------------------------------------------------------------*/
#define MAILBOX_ADDRESS(address) ( 0xFFFC & (address) )

#define CAN_CLK_FREQ_HZ            MCAN_PROG_CLK_FREQ_HZ

#define MCAN0_TSEG1                ( MCAN0_PROP_SEG + MCAN0_PHASE_SEG1 )
#define MCAN0_TSEG2                ( MCAN0_PHASE_SEG2 )
#define MCAN0_BRP                  ((uint32_t) (( (float) CAN_CLK_FREQ_HZ / \
                                                  ((float)( MCAN0_TSEG1 +
MCAN0_TSEG2 + 3 ) *\
                                                  (float)
MCAN0_BIT_RATE_BPS )) - 1 ))
#define MCAN0_SJW                  ( MCAN0_SYNC_JUMP - 1 )
#define MCAN0_FTSEG1               ( MCAN0_FAST_PROP_SEG +
MCAN0_FAST_PHASE_SEG1 )
#define MCAN0_FTSEG2               ( MCAN0_FAST_PHASE_SEG2 )
#define MCAN0_FBRP                 ((uint32_t) (( (float) CAN_CLK_FREQ_HZ / \
                                                  ((float)( MCAN0_FTSEG1 +
MCAN0_FTSEG2 + 3 ) * \
                                                  (float)
MCAN0_FAST_BIT_RATE_BPS )) - 1 ))
#define MCAN0_FSJW                 ( MCAN0_FAST_SYNC_JUMP - 1 )

#define MCAN0_STD_FLTS_WRDS        (MCAN0_NMBR_STD_FLTS)
                                                  /* 128 max filters */
#define MCAN0_EXT_FLTS_WRDS        (MCAN0_NMBR_EXT_FLTS * 2)
                                                  /* 64 max filters */
#define MCAN0_RX_FIFO0_WRDS        (MCAN0_NMBR_RX_FIFO0_ELMTS * \

        ((MCAN0_RX_FIFO0_ELMT_SZ/4) + 2))
                                                  /* 64 elements max */
#define MCAN0_RX_FIFO1_WRDS        (MCAN0_NMBR_RX_FIFO1_ELMTS *\

        ((MCAN0_RX_FIFO1_ELMT_SZ/4) + 2))
                                                  /* 64 elements max */
```

```
#define MCAN0_RX_DED_BUFS_WRDS          (MCAN0_NMBR_RX_DED_BUF_ELMTS
* \

        ((MCAN0_RX_BUF_ELMT_SZ/4) + 2))
                                                        /* 64 elements max */
#define MCAN0_TX_EVT_FIFO_WRDS      (MCAN0_NMBR_TX_EVT_FIFO_ELMTS * 2)
                                                        /* 32 elements max */
#define MCAN0_TX_DED_BUF_WRDS       (MCAN0_NMBR_TX_DED_BUF_ELMTS * \

        ((MCAN0_TX_BUF_ELMT_SZ/4) + 2))
                                                        /* 32 elements max */
#define MCAN0_TX_FIFO_Q_WRDS        (MCAN0_NMBR_TX_FIFO_Q_ELMTS *\

        ((MCAN0_TX_BUF_ELMT_SZ/4) + 2))
                                                        /* 32 elements max */

#define MCAN1_TSEG1                 ( MCAN1_PROP_SEG + MCAN1_PHASE_SEG1 )
#define MCAN1_TSEG2                 ( MCAN1_PHASE_SEG2 )
#define MCAN1_BRP                  ((uint32_t) (( (float) CAN_CLK_FREQ_HZ / \
                                                        ((float)( MCAN1_TSEG1 +
MCAN1_TSEG2 + 3 ) *\
                                                        (float)
MCAN1_BIT_RATE_BPS )) - 1 ))
#define MCAN1_SJW                  ( MCAN1_SYNC_JUMP - 1 )
#define MCAN1_FTSEG1                ( MCAN1_FAST_PROP_SEG +
MCAN1_FAST_PHASE_SEG1 )
#define MCAN1_FTSEG2                ( MCAN1_FAST_PHASE_SEG2 )
#define MCAN1_FBRP                 ((uint32_t) (( (float) CAN_CLK_FREQ_HZ /\
                                                        ((float)( MCAN1_FTSEG1 +
MCAN1_FTSEG2 + 3 ) *\
                                                        (float)
MCAN1_FAST_BIT_RATE_BPS )) - 1 ))
#define MCAN1_FSJW                  ( MCAN1_FAST_SYNC_JUMP - 1 )

#define MCAN1_STD_FLTS_WRDS         (MCAN1_NMBR_STD_FLTS)
                                                        /* 128 max filters */
#define MCAN1_EXT_FLTS_WRDS         (MCAN1_NMBR_EXT_FLTS * 2)
                                                        /* 64 max filters */
#define MCAN1_RX_FIFO0_WRDS         (MCAN1_NMBR_RX_FIFO0_ELMTS * \

        ((MCAN1_RX_FIFO0_ELMT_SZ/4) + 2))
                                                        /* 64 elements max */
#define MCAN1_RX_FIFO1_WRDS         (MCAN1_NMBR_RX_FIFO1_ELMTS *\

        ((MCAN1_RX_FIFO1_ELMT_SZ/4) + 2))
```

```
                                                        /* 64 elements max */
#define MCAN1_RX_DED_BUFS_WRDS          (MCAN1_NMBR_RX_DED_BUF_ELMTS
* \

        ((MCAN1_RX_BUF_ELMT_SZ/4) + 2))
                                                        /* 64 elements max */
#define MCAN1_TX_EVT_FIFO_WRDS      (MCAN1_NMBR_TX_EVT_FIFO_ELMTS * 2)
/* 32 elements max */
#define MCAN1_TX_DED_BUF_WRDS       (MCAN1_NMBR_TX_DED_BUF_ELMTS * \

        ((MCAN1_TX_BUF_ELMT_SZ/4) + 2))
                                                        /* 32 elements max */
#define MCAN1_TX_FIFO_Q_WRDS        (MCAN1_NMBR_TX_FIFO_Q_ELMTS * \

        ((MCAN1_TX_BUF_ELMT_SZ/4) + 2))
                                                        /* 32 elements max */

/* validate CAN0 entries */
#if ( MCAN0_TSEG1 > 63 )
#error "Invalid CAN0 TSEG1"
#endif
#if (  MCAN0_TSEG2 > 15 )
#error "Invalid CAN0 TSEG2"
#endif
#if ( MCAN0_SJW > 15 )
#error "Invalid CAN0 SJW"
#endif
#if ( MCAN0_FTSEG1 > 15 )
#error "Invalid CAN0 FTSEG1"
#endif
#if (  MCAN0_FTSEG2 > 7 )
#error "Invalid CAN0 FTSEG2"
#endif
#if ( MCAN0_FSJW > 3 )
#error "Invalid CAN0 FSJW"
#endif

#if ( MCAN0_NMBR_STD_FLTS > 128 )
#error "Invalid CAN0 # of Standard Filters"
#endif
#if ( MCAN0_NMBR_EXT_FLTS > 64 )
#error "Invalid CAN0 # of Extended Filters"
#endif
#if ( MCAN0_NMBR_RX_FIFO0_ELMTS > 64 )
#error "Invalid CAN0 # RX FIFO 0 ELEMENTS"
```

```
#endif
#if ( MCAN0_NMBR_RX_FIFO1_ELMTS > 64 )
#error "Invalid CAN0 # RX FIFO 0 ELEMENTS"
#endif
#if ( MCAN0_NMBR_RX_DED_BUF_ELMTS > 64 )
#error "Invalid CAN0 # RX BUFFER ELEMENTS"
#endif
#if ( MCAN0_NMBR_TX_EVT_FIFO_ELMTS > 32 )
#error "Invalid CAN0 # TX EVENT FIFO ELEMENTS"
#endif
#if ( (MCAN0_NMBR_TX_DED_BUF_ELMTS + MCAN0_NMBR_TX_FIFO_Q_ELMTS)  >
32 )
#error "Invalid CAN0 # TX BUFFER ELEMENTS"
#endif

#if   ( 8 == MCAN0_RX_FIFO0_ELMT_SZ )
#define MCAN0_RX_FIFO0_DATA_SIZE  (0u)
#elif ( 12 == MCAN0_RX_FIFO0_ELMT_SZ )
#define MCAN0_RX_FIFO0_DATA_SIZE  (1u)
#elif ( 16 == MCAN0_RX_FIFO0_ELMT_SZ )
#define MCAN0_RX_FIFO0_DATA_SIZE  (2u)
#elif ( 20 == MCAN0_RX_FIFO0_ELMT_SZ )
#define MCAN0_RX_FIFO0_DATA_SIZE  (3u)
#elif ( 24 == MCAN0_RX_FIFO0_ELMT_SZ )
#define MCAN0_RX_FIFO0_DATA_SIZE  (4u)
#elif ( 32 == MCAN0_RX_FIFO0_ELMT_SZ )
#define MCAN0_RX_FIFO0_DATA_SIZE  (5u)
#elif ( 48 == MCAN0_RX_FIFO0_ELMT_SZ )
#define MCAN0_RX_FIFO0_DATA_SIZE  (6u)
#elif ( 64 == MCAN0_RX_FIFO0_ELMT_SZ )
#define MCAN0_RX_FIFO0_DATA_SIZE  (7u)
#else
#error "Invalid CAN0 RX FIFO0 ELEMENT SIZE"
#endif

#if   ( 8 == MCAN0_RX_FIFO1_ELMT_SZ )
#define MCAN0_RX_FIFO1_DATA_SIZE  (0u)
#elif ( 12 == MCAN0_RX_FIFO1_ELMT_SZ )
#define MCAN0_RX_FIFO1_DATA_SIZE  (1u)
#elif ( 16 == MCAN0_RX_FIFO1_ELMT_SZ )
#define MCAN0_RX_FIFO1_DATA_SIZE  (2u)
#elif ( 20 == MCAN0_RX_FIFO1_ELMT_SZ )
#define MCAN0_RX_FIFO1_DATA_SIZE  (3u)
#elif ( 24 == MCAN0_RX_FIFO1_ELMT_SZ )
#define MCAN0_RX_FIFO1_DATA_SIZE  (4u)
```

```
#elif ( 32 == MCAN0_RX_FIFO1_ELMT_SZ )
#define MCAN0_RX_FIFO1_DATA_SIZE  (5u)
#elif ( 48 == MCAN0_RX_FIFO1_ELMT_SZ )
#define MCAN0_RX_FIFO1_DATA_SIZE  (6u)
#elif ( 64 == MCAN0_RX_FIFO1_ELMT_SZ )
#define MCAN0_RX_FIFO1_DATA_SIZE  (7u)
#else
#error "Invalid CAN0 RX FIFO1 ELEMENT SIZE"
#endif

#if   ( 8 == MCAN0_RX_BUF_ELMT_SZ )
#define MCAN0_RX_BUF_DATA_SIZE  (0u)
#elif ( 12 == MCAN0_RX_BUF_ELMT_SZ )
#define MCAN0_RX_BUF_DATA_SIZE  (1u)
#elif ( 16 == MCAN0_RX_BUF_ELMT_SZ )
#define MCAN0_RX_BUF_DATA_SIZE  (2u)
#elif ( 20 == MCAN0_RX_BUF_ELMT_SZ )
#define MCAN0_RX_BUF_DATA_SIZE  (3u)
#elif ( 24 == MCAN0_RX_BUF_ELMT_SZ )
#define MCAN0_RX_BUF_DATA_SIZE  (4u)
#elif ( 32 == MCAN0_RX_BUF_ELMT_SZ )
#define MCAN0_RX_BUF_DATA_SIZE  (5u)
#elif ( 48 == MCAN0_RX_BUF_ELMT_SZ )
#define MCAN0_RX_BUF_DATA_SIZE  (6u)
#elif ( 64 == MCAN0_RX_BUF_ELMT_SZ )
#define MCAN0_RX_BUF_DATA_SIZE  (7u)
#else
#error "Invalid CAN0 RX BUFFER ELEMENT SIZE"
#endif

#if   ( 8 == MCAN0_TX_BUF_ELMT_SZ )
#define MCAN0_TX_BUF_DATA_SIZE  (0u)
#elif ( 12 == MCAN0_TX_BUF_ELMT_SZ )
#define MCAN0_TX_BUF_DATA_SIZE  (1u)
#elif ( 16 == MCAN0_TX_BUF_ELMT_SZ )
#define MCAN0_TX_BUF_DATA_SIZE  (2u)
#elif ( 20 == MCAN0_TX_BUF_ELMT_SZ )
#define MCAN0_TX_BUF_DATA_SIZE  (3u)
#elif ( 24 == MCAN0_TX_BUF_ELMT_SZ )
#define MCAN0_TX_BUF_DATA_SIZE  (4u)
#elif ( 32 == MCAN0_TX_BUF_ELMT_SZ )
#define MCAN0_TX_BUF_DATA_SIZE  (5u)
#elif ( 48 == MCAN0_TX_BUF_ELMT_SZ )
#define MCAN0_TX_BUF_DATA_SIZE  (6u)
#elif ( 64 == MCAN0_TX_BUF_ELMT_SZ )
```

```
#define MCAN0_TX_BUF_DATA_SIZE  (7u)
#else
#error "Invalid CAN0 TX BUFFER ELEMENT SIZE"
#endif

/* validate CAN1 entries */
#if ( MCAN1_TSEG1 > 63 )
#error "Invalid CAN1 TSEG1"
#endif
#if (  MCAN1_TSEG2 > 15 )
#error "Invalid CAN1 TSEG2"
#endif
#if ( MCAN1_SJW > 15 )
#error "Invalid CAN1 SJW"
#endif
#if ( MCAN1_FTSEG1 > 15 )
#error "Invalid CAN1 FTSEG1"
#endif
#if (  MCAN1_FTSEG2 > 7 )
#error "Invalid CAN1 FTSEG2"
#endif
#if ( MCAN1_FSJW > 3 )
#error "Invalid CAN1 FSJW"
#endif

#if ( MCAN1_NMBR_STD_FLTS > 128 )
#error "Invalid CAN1 # of Standard Filters"
#endif
#if ( MCAN1_NMBR_EXT_FLTS > 64 )
#error "Invalid CAN1 # of Extended Filters"
#endif
#if ( MCAN1_NMBR_RX_FIFO0_ELMTS > 64 )
#error "Invalid CAN1 # RX FIFO 0 ELEMENTS"
#endif
#if ( MCAN1_NMBR_RX_FIFO1_ELMTS > 64 )
#error "Invalid CAN1 # RX FIFO 0 ELEMENTS"
#endif
#if ( MCAN1_NMBR_RX_DED_BUF_ELMTS > 64 )
#error "Invalid CAN1 # RX BUFFER ELEMENTS"
#endif
#if ( MCAN1_NMBR_TX_EVT_FIFO_ELMTS > 32 )
#error "Invalid CAN1 # TX EVENT FIFO ELEMENTS"
#endif
#if ( (MCAN1_NMBR_TX_DED_BUF_ELMTS + MCAN1_NMBR_TX_FIFO_Q_ELMTS)  >
32 )
```

```
#error "Invalid CAN1 # TX BUFFER ELEMENTS"
#endif

#if   ( 8 == MCAN1_RX_FIFO0_ELMT_SZ )
#define MCAN1_RX_FIFO0_DATA_SIZE  (0u)
#elif ( 12 == MCAN1_RX_FIFO0_ELMT_SZ )
#define MCAN1_RX_FIFO0_DATA_SIZE  (1u)
#elif ( 16 == MCAN1_RX_FIFO0_ELMT_SZ )
#define MCAN1_RX_FIFO0_DATA_SIZE  (2u)
#elif ( 20 == MCAN1_RX_FIFO0_ELMT_SZ )
#define MCAN1_RX_FIFO0_DATA_SIZE  (3u)
#elif ( 24 == MCAN1_RX_FIFO0_ELMT_SZ )
#define MCAN1_RX_FIFO0_DATA_SIZE  (4u)
#elif ( 32 == MCAN1_RX_FIFO0_ELMT_SZ )
#define MCAN1_RX_FIFO0_DATA_SIZE  (5u)
#elif ( 48 == MCAN1_RX_FIFO0_ELMT_SZ )
#define MCAN1_RX_FIFO0_DATA_SIZE  (6u)
#elif ( 64 == MCAN1_RX_FIFO0_ELMT_SZ )
#define MCAN1_RX_FIFO0_DATA_SIZE  (7u)
#else
#error "Invalid CAN1 RX FIFO0 ELEMENT SIZE"
#endif

#if   ( 8 == MCAN1_RX_FIFO1_ELMT_SZ )
#define MCAN1_RX_FIFO1_DATA_SIZE  (0u)
#elif ( 12 == MCAN1_RX_FIFO1_ELMT_SZ )
#define MCAN1_RX_FIFO1_DATA_SIZE  (1u)
#elif ( 16 == MCAN1_RX_FIFO1_ELMT_SZ )
#define MCAN1_RX_FIFO1_DATA_SIZE  (2u)
#elif ( 20 == MCAN1_RX_FIFO1_ELMT_SZ )
#define MCAN1_RX_FIFO1_DATA_SIZE  (3u)
#elif ( 24 == MCAN1_RX_FIFO1_ELMT_SZ )
#define MCAN1_RX_FIFO1_DATA_SIZE  (4u)
#elif ( 32 == MCAN1_RX_FIFO1_ELMT_SZ )
#define MCAN1_RX_FIFO1_DATA_SIZE  (5u)
#elif ( 48 == MCAN1_RX_FIFO1_ELMT_SZ )
#define MCAN1_RX_FIFO1_DATA_SIZE  (6u)
#elif ( 64 == MCAN1_RX_FIFO1_ELMT_SZ )
#define MCAN1_RX_FIFO1_DATA_SIZE  (7u)
#else
#error "Invalid CAN1 RX FIFO1 ELEMENT SIZE"
#endif

#if   ( 8 == MCAN1_RX_BUF_ELMT_SZ )
#define MCAN1_RX_BUF_DATA_SIZE  (0u)
```

```
#elif ( 12 == MCAN1_RX_BUF_ELMT_SZ )
#define MCAN1_RX_BUF_DATA_SIZE  (1u)
#elif ( 16 == MCAN1_RX_BUF_ELMT_SZ )
#define MCAN1_RX_BUF_DATA_SIZE  (2u)
#elif ( 20 == MCAN1_RX_BUF_ELMT_SZ )
#define MCAN1_RX_BUF_DATA_SIZE  (3u)
#elif ( 24 == MCAN1_RX_BUF_ELMT_SZ )
#define MCAN1_RX_BUF_DATA_SIZE  (4u)
#elif ( 32 == MCAN1_RX_BUF_ELMT_SZ )
#define MCAN1_RX_BUF_DATA_SIZE  (5u)
#elif ( 48 == MCAN1_RX_BUF_ELMT_SZ )
#define MCAN1_RX_BUF_DATA_SIZE  (6u)
#elif ( 64 == MCAN1_RX_BUF_ELMT_SZ )
#define MCAN1_RX_BUF_DATA_SIZE  (7u)
#else
#error "Invalid CAN1 RX BUFFER ELEMENT SIZE"
#endif

#if   ( 8 == MCAN1_TX_BUF_ELMT_SZ )
#define MCAN1_TX_BUF_DATA_SIZE  (0u)
#elif ( 12 == MCAN1_TX_BUF_ELMT_SZ )
#define MCAN1_TX_BUF_DATA_SIZE  (1u)
#elif ( 16 == MCAN1_TX_BUF_ELMT_SZ )
#define MCAN1_TX_BUF_DATA_SIZE  (2u)
#elif ( 20 == MCAN1_TX_BUF_ELMT_SZ )
#define MCAN1_TX_BUF_DATA_SIZE  (3u)
#elif ( 24 == MCAN1_TX_BUF_ELMT_SZ )
#define MCAN1_TX_BUF_DATA_SIZE  (4u)
#elif ( 32 == MCAN1_TX_BUF_ELMT_SZ )
#define MCAN1_TX_BUF_DATA_SIZE  (5u)
#elif ( 48 == MCAN1_TX_BUF_ELMT_SZ )
#define MCAN1_TX_BUF_DATA_SIZE  (6u)
#elif ( 64 == MCAN1_TX_BUF_ELMT_SZ )
#define MCAN1_TX_BUF_DATA_SIZE  (7u)
#else
#error "Invalid CAN1 TX BUFFER ELEMENT SIZE"
#endif

#define CAN_11_BIT_ID_MASK              (0x7FF)
#define CAN_29_BIT_ID_MASK              (0x1FFFFFFF)
#define ELMT_SIZE_MASK              (0x1F)
/* max element size is 18 words, fits in 5 bits */

#define BUFFER_XTD_MASK             (0x40000000)
#define BUFFER_EXT_ID_MASK          (0x1FFFFFFF)
```

```
#define BUFFER_STD_ID_MASK          (0x1FFC0000)
#define BUFFER_DLC_MASK             (0x000F0000)
#define BUFFER_RXTS_MASK            (0x0000FFFF)

#define STD_FILT_SFT_MASK           (3 << 30)
#define STD_FILT_SFT_RANGE          (0 << 30)
#define STD_FILT_SFT_DUAL           (1 << 30)
#define STD_FILT_SFT_CLASSIC        (2 << 30)
#define STD_FILT_SFEC_MASK          (7 << 27)
#define STD_FILT_SFEC_DISABLE       (0 << 27)
#define STD_FILT_SFEC_FIFO0         (1 << 27)
#define STD_FILT_SFEC_FIFO1         (2 << 27)
#define STD_FILT_SFEC_REJECT        (3 << 27)
#define STD_FILT_SFEC_PRIORITY      (4 << 27)
#define STD_FILT_SFEC_PRIORITY_FIFO0  (5 << 27)
#define STD_FILT_SFEC_PRIORITY_FIFO1  (6 << 27)
#define STD_FILT_SFEC_BUFFER        (7 << 27)
#define STD_FILT_SFID1_MASK         (0x03FF << 16)
#define STD_FILT_SFID2_MASK         (0x3FF << 0)
#define STD_FILT_SFID2_RX_BUFFER    (0 << 9)
#define STD_FILT_SFID2_DEBUG_A      (1 << 9)
#define STD_FILT_SFID2_DEBUG_B      (2 << 9)
#define STD_FILT_SFID2_DEBUG_C      (3 << 9)
#define STD_FILT_SFID2_BUFFER(nmbr)   (nmbr & 0x3F)

#define EXT_FILT_EFEC_MASK          (7 << 29)
#define EXT_FILT_EFEC_DISABLE       (0 << 29)
#define EXT_FILT_EFEC_FIFO0         (1 << 29)
#define EXT_FILT_EFEC_FIFO1         (2 << 29)
#define EXT_FILT_EFEC_REJECT        (3 << 29)
#define EXT_FILT_EFEC_PRIORITY      (4 << 29)
#define EXT_FILT_EFEC_PRIORITY_FIFO0  (5 << 29)
#define EXT_FILT_EFEC_PRIORITY_FIFO1  (6 << 29)
#define EXT_FILT_EFEC_BUFFER        (7 << 29)
#define EXT_FILT_EFID1_MASK         (0x1FFFFFFF)
#define EXT_FILT_EFT_MASK           (3 << 30)
#define EXT_FILT_EFT_RANGE          (0 << 30)
#define EXT_FILT_EFT_DUAL           (1 << 30)
#define EXT_FILT_EFT_CLASSIC        (2 << 30)
#define EXT_FILT_EFT_RANGE_NO_XIDAM   (3 << 30)
#define EXT_FILT_EFID2_MASK         (0x1FFFFFFF)
#define EXT_FILT_EFID2_RX_BUFFER    (0 << 9)
#define EXT_FILT_EFID2_DEBUG_A      (1 << 9)
#define EXT_FILT_EFID2_DEBUG_B      (2 << 9)
#define EXT_FILT_EFID2_DEBUG_C      (3 << 9)
```

```
#define EXT_FILT_EFID2_BUFFER(nmbr)   (nmbr & 0x3F)


/*----------------------------------------------------------------------------
 *      Internal variables
 *----------------------------------------------------------------------------*/

static const Pin pinsMcan0[] =  {PIN_MCAN0_TXD, PIN_MCAN0_RXD };
static const Pin pinsMcan1[] =  {PIN_MCAN1_TXD, PIN_MCAN1_RXD };

static uint32_t can0MsgRam[MCAN0_STD_FLTS_WRDS +
                                                MCAN0_EXT_FLTS_WRDS +
                                                MCAN0_RX_FIFO0_WRDS +
                                                MCAN0_RX_FIFO1_WRDS +
                                                MCAN0_RX_DED_BUFS_WRDS +
                                                MCAN0_TX_EVT_FIFO_WRDS +
                                                MCAN0_TX_DED_BUF_WRDS +
                                                MCAN0_TX_FIFO_Q_WRDS];

static uint32_t can1MsgRam[MCAN1_STD_FLTS_WRDS +
                                                MCAN1_EXT_FLTS_WRDS +
                                                MCAN1_RX_FIFO0_WRDS +
                                                MCAN1_RX_FIFO1_WRDS +
                                                MCAN1_RX_DED_BUFS_WRDS +
                                                MCAN1_TX_EVT_FIFO_WRDS +
                                                MCAN1_TX_DED_BUF_WRDS +
                                                MCAN1_TX_FIFO_Q_WRDS];

const MCan_ConfigType mcan0Config =
{
        MCAN0,
        MCAN_BTP_BRP(MCAN0_BRP) | MCAN_BTP_TSEG1(MCAN0_TSEG1) |
        MCAN_BTP_TSEG2(MCAN0_TSEG2) | MCAN_BTP_SJW(MCAN0_SJW),
        MCAN_FBTP_FBRP(MCAN0_FBRP) | MCAN_FBTP_FTSEG1(MCAN0_FTSEG1) |
        MCAN_FBTP_FTSEG2(MCAN0_FTSEG2) | MCAN_FBTP_FSJW(MCAN0_FSJW),
        MCAN0_NMBR_STD_FLTS,
        MCAN0_NMBR_EXT_FLTS,
        MCAN0_NMBR_RX_FIFO0_ELMTS,
        MCAN0_NMBR_RX_FIFO1_ELMTS,
        MCAN0_NMBR_RX_DED_BUF_ELMTS,
        MCAN0_NMBR_TX_EVT_FIFO_ELMTS,
        MCAN0_NMBR_TX_DED_BUF_ELMTS,
        MCAN0_NMBR_TX_FIFO_Q_ELMTS,
        (MCAN0_RX_FIFO0_DATA_SIZE << 29) | ((MCAN0_RX_FIFO0_ELMT_SZ/4)+2),
        /* element size in WORDS */
```

```
        (MCAN0_RX_FIFO1_DATA_SIZE << 29) | ((MCAN0_RX_FIFO1_ELMT_SZ/4)+2),
        /* element size in WORDS */
        (MCAN0_RX_BUF_DATA_SIZE << 29) | ((MCAN0_RX_BUF_ELMT_SZ/4)+2),
        /* element size in WORDS */
        (MCAN0_TX_BUF_DATA_SIZE << 29) | ((MCAN0_TX_BUF_ELMT_SZ/4)+2),
        /* element size in WORDS */
        {
                &can0MsgRam[0],
                &can0MsgRam[MCAN0_STD_FLTS_WRDS],
                &can0MsgRam[MCAN0_STD_FLTS_WRDS + MCAN0_EXT_FLTS_WRDS],
                &can0MsgRam[MCAN0_STD_FLTS_WRDS + MCAN0_EXT_FLTS_WRDS +
MCAN0_RX_FIFO0_WRDS],
                &can0MsgRam[MCAN0_STD_FLTS_WRDS + MCAN0_EXT_FLTS_WRDS +
MCAN0_RX_FIFO0_WRDS +
                MCAN0_RX_FIFO1_WRDS],
                &can0MsgRam[MCAN0_STD_FLTS_WRDS + MCAN0_EXT_FLTS_WRDS +
MCAN0_RX_FIFO0_WRDS +
                MCAN0_RX_FIFO1_WRDS + MCAN0_RX_DED_BUFS_WRDS],
                &can0MsgRam[MCAN0_STD_FLTS_WRDS + MCAN0_EXT_FLTS_WRDS +
MCAN0_RX_FIFO0_WRDS +
                MCAN0_RX_FIFO1_WRDS + MCAN0_RX_DED_BUFS_WRDS +
MCAN0_TX_EVT_FIFO_WRDS],
                &can0MsgRam[MCAN0_STD_FLTS_WRDS + MCAN0_EXT_FLTS_WRDS +
MCAN0_RX_FIFO0_WRDS +
                MCAN0_RX_FIFO1_WRDS + MCAN0_RX_DED_BUFS_WRDS +
MCAN0_TX_EVT_FIFO_WRDS +
                MCAN0_TX_DED_BUF_WRDS]
  },
};

const MCan_ConfigType mcan1Config =
{
        MCAN1,
        MCAN_BTP_BRP(MCAN1_BRP) | MCAN_BTP_TSEG1(MCAN1_TSEG1) |
        MCAN_BTP_TSEG2(MCAN1_TSEG2) | MCAN_BTP_SJW(MCAN1_SJW),
        MCAN_FBTP_FBRP(MCAN1_FBRP) | MCAN_FBTP_FTSEG1(MCAN1_FTSEG1) |
        MCAN_FBTP_FTSEG2(MCAN1_FTSEG2) | MCAN_FBTP_FSJW(MCAN1_FSJW),
        MCAN1_NMBR_STD_FLTS,
        MCAN1_NMBR_EXT_FLTS,
        MCAN1_NMBR_RX_FIFO0_ELMTS,
        MCAN1_NMBR_RX_FIFO1_ELMTS,
        MCAN0_NMBR_RX_DED_BUF_ELMTS,
        MCAN1_NMBR_TX_EVT_FIFO_ELMTS,
        MCAN1_NMBR_TX_DED_BUF_ELMTS,
        MCAN1_NMBR_TX_FIFO_Q_ELMTS,
```

```
        (MCAN1_RX_FIFO0_DATA_SIZE << 29) | ((MCAN1_RX_FIFO0_ELMT_SZ/4)+2),
        /* element size in WORDS */
        (MCAN1_RX_FIFO1_DATA_SIZE << 29) | ((MCAN1_RX_FIFO1_ELMT_SZ/4)+2),
        /* element size in WORDS */
        (MCAN1_RX_BUF_DATA_SIZE << 29) | ((MCAN1_RX_BUF_ELMT_SZ/4)+2),
        /* element size in WORDS */
        (MCAN1_TX_BUF_DATA_SIZE << 29) | ((MCAN1_TX_BUF_ELMT_SZ/4)+2),
        /* element size in WORDS */
        {
                &can1MsgRam[0],
                &can1MsgRam[MCAN1_STD_FLTS_WRDS],
                &can1MsgRam[MCAN1_STD_FLTS_WRDS + MCAN1_EXT_FLTS_WRDS],
                &can1MsgRam[MCAN1_STD_FLTS_WRDS + MCAN1_EXT_FLTS_WRDS +
MCAN1_RX_FIFO0_WRDS],
                &can1MsgRam[MCAN1_STD_FLTS_WRDS + MCAN1_EXT_FLTS_WRDS +
MCAN1_RX_FIFO0_WRDS
                + MCAN1_RX_FIFO1_WRDS],
                &can1MsgRam[MCAN1_STD_FLTS_WRDS + MCAN1_EXT_FLTS_WRDS +
MCAN1_RX_FIFO0_WRDS
                + MCAN1_RX_FIFO1_WRDS + MCAN1_RX_DED_BUFS_WRDS],
                &can1MsgRam[MCAN1_STD_FLTS_WRDS + MCAN1_EXT_FLTS_WRDS +
MCAN1_RX_FIFO0_WRDS
                + MCAN1_RX_FIFO1_WRDS + MCAN1_RX_DED_BUFS_WRDS +
MCAN1_TX_EVT_FIFO_WRDS],
                &can1MsgRam[MCAN1_STD_FLTS_WRDS + MCAN1_EXT_FLTS_WRDS +
MCAN1_RX_FIFO0_WRDS
                + MCAN1_RX_FIFO1_WRDS + MCAN1_RX_DED_BUFS_WRDS +
MCAN1_TX_EVT_FIFO_WRDS
                + MCAN1_TX_DED_BUF_WRDS]
        },
};


/*----------------------------------------------------------------------------
 *      Exported Functions
 *----------------------------------------------------------------------------*/
 /**
 * \brief Initializes the MCAN hardware for giving peripheral.
 * Default: Mixed mode TX Buffer + FIFO.
 *
 * \param mcanConfig  Pointer to a MCAN instance.
 */
void MCAN_Init( const MCan_ConfigType * mcanConfig )
{
        Mcan      * mcan = mcanConfig->pMCan;
```

```
	uint32_t    regVal32;
	uint32_t  * pMsgRam;
	uint32_t    cntr;
	IRQn_Type   mCanLine0Irq;

	/* Both MCAN controllers use programmable clock 5 to derive bit rate */
	// select MCK divided by 1 as programmable clock 5 output
	PMC->PMC_PCK[5] = PMC_PCK_PRES(MCAN_PROG_CLK_PRESCALER - 1) |
MCAN_PROG_CLK_SELECT;
	PMC->PMC_SCER = PMC_SCER_PCK5;

	if( MCAN0 ==  mcan ) {
		PIO_Configure(pinsMcan0, PIO_LISTSIZE(pinsMcan0));
		// Enable MCAN peripheral clock
		PMC_EnablePeripheral( ID_MCAN0 );
		// Configure Message RAM Base Address
		regVal32 = MATRIX->CCFG_CAN0 & 0x000001FF;
		MATRIX->CCFG_CAN0 = regVal32 |
			( (uint32_t) mcanConfig->msgRam.pStdFilts & 0xFFFF0000 );
		mCanLine0Irq = MCAN0_IRQn;
	} else if ( MCAN1 ==  mcan ) {
		PIO_Configure(pinsMcan1, PIO_LISTSIZE(pinsMcan1));
		// Enable MCAN peripheral clock
		PMC_EnablePeripheral( ID_MCAN1 );
		// Configure Message RAM Base Address
		regVal32 = MATRIX->CCFG_SYSIO & 0x0000FFFF;
		MATRIX->CCFG_SYSIO = regVal32 | ( (uint32_t) mcanConfig-
>msgRam.pStdFilts & 0xFFFF0000 );
		mCanLine0Irq = MCAN1_IRQn;
	} else {
		return;
	}

	/* Indicates Initialization state */
	mcan->MCAN_CCCR = MCAN_CCCR_INIT_ENABLED;
	do { regVal32 = mcan->MCAN_CCCR; }
		while(0u == (regVal32 & MCAN_CCCR_INIT_ENABLED));

	/* Enable writing to configuration registers */
	mcan->MCAN_CCCR = MCAN_CCCR_INIT_ENABLED |
MCAN_CCCR_CCE_CONFIGURABLE;

	/* Global Filter Configuration: Reject remote frames, reject non-matching frames */
	mcan->MCAN_GFC = MCAN_GFC_RRFE_REJECT | MCAN_GFC_RRFS_REJECT
		| MCAN_GFC_ANFE(2) | MCAN_GFC_ANFS(2);
```

```
        // Extended ID Filter AND mask
        mcan->MCAN_XIDAM = 0x1FFFFFFF;

        /* Interrupt configuration - leave initialization with all interrupts off */
        // Disable all interrupts
        mcan->MCAN_IE =  0;
        mcan->MCAN_TXBTIE = 0x00000000;
        // All interrupts directed to Line 0
        mcan->MCAN_ILS =  0x00000000;
        // Disable both interrupt LINE 0 & LINE 1
        mcan->MCAN_ILE = 0x00;
        // Clear all interrupt flags
        mcan->MCAN_IR = 0xFFCFFFFF;
        /* Enable NVIC - but no interrupts will happen since all sources are
                disabled in MCAN_IE */
        NVIC_ClearPendingIRQ(mCanLine0Irq);
        NVIC_EnableIRQ(mCanLine0Irq);
        NVIC_ClearPendingIRQ((IRQn_Type) (mCanLine0Irq+1));
        NVIC_EnableIRQ((IRQn_Type) (mCanLine0Irq+1));

        /* Configure CAN bit timing */
        mcan->MCAN_BTP = mcanConfig->bitTiming;
        mcan->MCAN_FBTP = mcanConfig->fastBitTiming;

        /* Configure message RAM starting addresses & sizes */
        mcan->MCAN_SIDFC = MAILBOX_ADDRESS( (uint32_t) mcanConfig-
>msgRam.pStdFilts )
                        | MCAN_SIDFC_LSS(mcanConfig->nmbrStdFilts);
        mcan->MCAN_XIDFC = MAILBOX_ADDRESS( (uint32_t) mcanConfig-
>msgRam.pExtFilts )
                        | MCAN_XIDFC_LSE(mcanConfig->nmbrExtFilts);
        mcan->MCAN_RXF0C = MAILBOX_ADDRESS( (uint32_t) mcanConfig-
>msgRam.pRxFifo0 )
                        | MCAN_RXF0C_F0S(mcanConfig->nmbrFifo0Elmts);
        // watermark interrupt off, blocking mode
        mcan->MCAN_RXF1C = MAILBOX_ADDRESS( (uint32_t) mcanConfig-
>msgRam.pRxFifo1 )
                        | MCAN_RXF1C_F1S(mcanConfig->nmbrFifo1Elmts);
        // watermark interrupt off, blocking mode
        mcan->MCAN_RXBC = MAILBOX_ADDRESS( (uint32_t) mcanConfig-
>msgRam.pRxDedBuf );
        mcan->MCAN_TXEFC = MAILBOX_ADDRESS( (uint32_t) mcanConfig-
>msgRam.pTxEvtFifo )
                        | MCAN_TXEFC_EFS(mcanConfig->nmbrTxEvtFifoElmts);
```

```
        // watermark interrupt off
        mcan->MCAN_TXBC = MAILBOX_ADDRESS( (uint32_t) mcanConfig-
>msgRam.pTxDedBuf )
                        | MCAN_TXBC_NDTB(mcanConfig->nmbrTxDedBufElmts)
                        | MCAN_TXBC_TFQS(mcanConfig->nmbrTxFifoQElmts);
        mcan->MCAN_RXESC = ((mcanConfig->rxBufElmtSize >> (29-
MCAN_RXESC_RBDS_Pos)) &
                                                MCAN_RXESC_RBDS_Msk) |
                                                ((mcanConfig->rxFifo1ElmtSize >> (29-
MCAN_RXESC_F1DS_Pos)) &
                                                MCAN_RXESC_F1DS_Msk) |
                                                ((mcanConfig->rxFifo0ElmtSize >> (29-
MCAN_RXESC_F0DS_Pos)) &
                                        MCAN_RXESC_F0DS_Msk);
        mcan->MCAN_TXESC = ((mcanConfig->txBufElmtSize >> (29-
MCAN_TXESC_TBDS_Pos)) &
                                MCAN_TXESC_TBDS_Msk);

        /* Configure Message Filters */
        // ...Disable all standard filters
        pMsgRam = mcanConfig->msgRam.pStdFilts;
        cntr = mcanConfig->nmbrStdFilts;
        while ( cntr > 0 ) {
          *pMsgRam++ = STD_FILT_SFEC_DISABLE;
          cntr--;
        }
        // ...Disable all extended filters
        pMsgRam = mcanConfig->msgRam.pExtFilts;
        cntr = mcanConfig->nmbrExtFilts;
        while ( cntr > 0 ) {
          *pMsgRam = EXT_FILT_EFEC_DISABLE;
          pMsgRam = pMsgRam + 2;
          cntr--;
        }

        mcan->MCAN_NDAT1 = 0xFFFFFFFF;  // clear new (rx) data flags
        mcan->MCAN_NDAT2 = 0xFFFFFFFF;  // clear new (rx) data flags

        regVal32 =  mcan->MCAN_CCCR & ~(MCAN_CCCR_CME_Msk |
MCAN_CCCR_CMR_Msk);
        mcan->MCAN_CCCR = regVal32 | MCAN_CCCR_CME_ISO11898_1;
        mcan->MCAN_CCCR = regVal32 | (MCAN_CCCR_CMR_ISO11898_1 |
MCAN_CCCR_CME_ISO11898_1);

        __DSB();
```

```
        __ISB();
}

/**
 * \brief Enables a FUTURE switch to FD mode (tx & rx payloads up to 64 bytes)
 * but transmits WITHOUT bit rate switching
 * INIT must be set - so this should be called between MCAN_Init() and
 * MCAN_Enable()
 * \param mcanConfig  Pointer to a MCAN instance.
 */
void MCAN_InitFdEnable( const MCan_ConfigType * mcanConfig )
{
        Mcan     * mcan = mcanConfig->pMCan;
        uint32_t   regVal32;

        regVal32 =  mcan->MCAN_CCCR & ~MCAN_CCCR_CME_Msk;
        mcan->MCAN_CCCR = regVal32 | MCAN_CCCR_CME(1);
}

/**
 * \brief Enables a FUTURE switch to FD mode (tx & rx payloads up to 64 bytes) and transmits
 * WITH bit rate switching
 * INIT must be set - so this should be called between MCAN_Init() and MCAN_Enable()
 * \param mcanConfig  Pointer to a MCAN instance.
 */
void MCAN_InitFdBitRateSwitchEnable( const MCan_ConfigType * mcanConfig )
{
        Mcan     * mcan = mcanConfig->pMCan;
        uint32_t   regVal32;

        regVal32 =  mcan->MCAN_CCCR & ~MCAN_CCCR_CME_Msk;
        mcan->MCAN_CCCR = regVal32 | MCAN_CCCR_CME(2);
}

/**
 * \brief Initializes the MCAN in loop back mode.
 * INIT must be set - so this should be called between MCAN_Init() and
 * MCAN_Enable()
 * \param mcanConfig  Pointer to a MCAN instance.
 */
void MCAN_InitLoopback( const MCan_ConfigType * mcanConfig )
{
        Mcan * mcan = mcanConfig->pMCan;

        mcan->MCAN_CCCR |= MCAN_CCCR_TEST_ENABLED;
```

```
        //mcan->MCAN_CCCR |= MCAN_CCCR_MON_ENABLED;  // for internal loop back
        mcan->MCAN_TEST |= MCAN_TEST_LBCK_ENABLED;
}

/**
 * \brief Initializes MCAN queue for TX
 * INIT must be set - so this should be called between MCAN_Init() and
 * MCAN_Enable()
 * \param mcanConfig  Pointer to a MCAN instance.
 */
void MCAN_InitTxQueue( const MCan_ConfigType * mcanConfig )
{
        Mcan * mcan = mcanConfig->pMCan;
        mcan->MCAN_TXBC |= MCAN_TXBC_TFQM;
}

/**
 * \brief Enable MCAN peripheral.
 * INIT must be set - so this should be called between MCAN_Init()
 * \param mcanConfig  Pointer to a MCAN instance.
 */
void MCAN_Enable( const MCan_ConfigType * mcanConfig )
{
        Mcan * mcan = mcanConfig->pMCan;
        mcan->MCAN_CCCR &= ~MCAN_CCCR_INIT_ENABLED;
}

/**
 * \brief Requests switch to Iso11898-1 (standard / classic) mode (tx & rx
 * payloads up to 8 bytes).
 * \param mcanConfig  Pointer to a MCAN instance.
 */
void MCAN_RequestIso11898_1( const MCan_ConfigType * mcanConfig )
{
        Mcan     * mcan = mcanConfig->pMCan;
        uint32_t   regVal32;

        regVal32 =  mcan->MCAN_CCCR & ~MCAN_CCCR_CMR_Msk;
        mcan->MCAN_CCCR = regVal32 | MCAN_CCCR_CMR_ISO11898_1;
        while ( (mcan->MCAN_CCCR & ( MCAN_CCCR_FDBS | MCAN_CCCR_FDO )) != 0
)
                { /* wait */ }
}

/**
```

```
 * \brief Requests switch to FD mode (tx & rx payloads up to 64 bytes) but
 * transmits WITHOUT bit
 * rate switching. requested mode should have been enabled at initialization
 * \param mcanConfig  Pointer to a MCAN instance.
 */
void MCAN_RequestFd( const MCan_ConfigType * mcanConfig )
{
        Mcan     * mcan = mcanConfig->pMCan;
        uint32_t   regVal32;

        if (( mcan->MCAN_CCCR & MCAN_CCCR_CME_Msk ) == MCAN_CCCR_CME(1)
) {
                regVal32 =  mcan->MCAN_CCCR & ~MCAN_CCCR_CMR_Msk;
                mcan->MCAN_CCCR = regVal32 | MCAN_CCCR_CMR_FD;
                while ( (mcan->MCAN_CCCR & MCAN_CCCR_FDO) == 0 ) { /* wait */ }
        }
}

/**
 * \brief Request switch to FD mode (tx & rx payloads up to 64 bytes) and
 * transmits WITH bit rate switching.
 * requested mode should have been enabled at initialization
 * \param mcanConfig  Pointer to a MCAN instance.
 */
void MCAN_RequestFdBitRateSwitch( const MCan_ConfigType * mcanConfig )
{
        Mcan     * mcan = mcanConfig->pMCan;
        uint32_t   regVal32;

        if (( mcan->MCAN_CCCR & MCAN_CCCR_CME_Msk ) == MCAN_CCCR_CME(2)
) {
          regVal32 =  mcan->MCAN_CCCR & ~MCAN_CCCR_CMR_Msk;
          mcan->MCAN_CCCR = regVal32 | MCAN_CCCR_CMR_FD_BITRATE_SWITCH;
          while ( (mcan->MCAN_CCCR & ( MCAN_CCCR_FDBS | MCAN_CCCR_FDO )) !=
                        ( MCAN_CCCR_FDBS | MCAN_CCCR_FDO ) ) { /* wait */ }
        }
}

/**
 * \brief Switch on loop back mode.
 * TEST must be set in MCAN_CCCR - e.g. by a prior call to MCAN_InitLoopback()
 * \param mcanConfig  Pointer to a MCAN instance.
 */
void MCAN_LoopbackOn( const MCan_ConfigType * mcanConfig )
{
```

```
        Mcan * mcan = mcanConfig->pMCan;
        mcan->MCAN_TEST |= MCAN_TEST_LBCK_ENABLED;
}

/**
 * \brief Switch off loop back mode.
 * \param mcanConfig  Pointer to a MCAN instance.
 */
void MCAN_LoopbackOff( const MCan_ConfigType * mcanConfig )
{
        Mcan * mcan = mcanConfig->pMCan;
        mcan->MCAN_TEST &= ~MCAN_TEST_LBCK_ENABLED;
}

/**
 * \brief Enable message line and message stored to Dedicated Receive Buffer
 * Interrupt Line.
 * \param mcanConfig  Pointer to a MCAN instance.
 * \param line  Message line.
 */
void MCAN_IEnableMessageStoredToRxDedBuffer( const MCan_ConfigType * mcanConfig,
                MCan_IntrLineType line )
{
        Mcan * mcan = mcanConfig->pMCan;
        if ( line == CAN_INTR_LINE_0 ) {
                mcan->MCAN_ILS &= ~MCAN_ILS_DRXL;
                mcan->MCAN_ILE |= MCAN_ILE_EINT0;
        } else  {
        // Interrupt Line 1
                mcan->MCAN_ILS |= MCAN_ILS_DRXL;
                mcan->MCAN_ILE |= MCAN_ILE_EINT1;
        }
        mcan->MCAN_IR = MCAN_IR_DRX;  // clear previous flag
        mcan->MCAN_IE |= MCAN_IE_DRXE;  // enable it
}

/**
 * \brief Configures a Dedicated TX Buffer.
 * \param mcanConfig  Pointer to a MCAN instance.
 * \param buffer  Pointer to buffer.
 * \param id  Message ID.
 * \param idType  Type of ID
 * \param dlc  Type of dlc.
 */
uint8_t  * MCAN_ConfigTxDedBuffer( const MCan_ConfigType * mcanConfig,
```

```
                uint8_t buffer, uint32_t id, MCan_IdType idType, MCan_DlcType dlc )
{
 Mcan * mcan = mcanConfig->pMCan;
 uint32_t * pThisTxBuf = 0;

 if ( buffer < mcanConfig->nmbrTxDedBufElmts )
 {
        pThisTxBuf = mcanConfig->msgRam.pTxDedBuf + (buffer *
                        (mcanConfig->txBufElmtSize & ELMT_SIZE_MASK));
              if ( idType == CAN_STD_ID )
          *pThisTxBuf++ = (( id << 18 ) & ( CAN_11_BIT_ID_MASK << 18 ));
        else
                        *pThisTxBuf++ = BUFFER_XTD_MASK | ( id &
CAN_29_BIT_ID_MASK );

   *pThisTxBuf++ = (uint32_t) dlc << 16;
              /* enable transmit from buffer to set TC interrupt bit in IR, but
              interrupt will not happen unless TC interrupt is enabled*/
              mcan->MCAN_TXBTIE = ( 1 << buffer) ;
        }
        SCB_CleanInvalidateDCache();
        return (uint8_t *) pThisTxBuf;  // now it points to the data field
}

/**
 * \brief Send Tx buffer.
 * \param mcanConfig  Pointer to a MCAN instance.
 * \param buffer  Pointer to buffer.
 */
void MCAN_SendTxDedBuffer( const MCan_ConfigType * mcanConfig, uint8_t buffer )
{
        Mcan * mcan = mcanConfig->pMCan;

        if ( buffer < mcanConfig->nmbrTxDedBufElmts ) {
          mcan->MCAN_TXBAR = ( 1 << buffer );
        }
}

/**
 * \brief Adds Message to TX Fifo / Queue
 * \param mcanConfig  Pointer to a MCAN instance.
 * \param id  Message ID.
 * \param idType  Type of ID
 * \param dlc  Type of dlc.
 * \param data  Pointer to data.
```

```
 */
uint32_t MCAN_AddToTxFifoQ( const MCan_ConfigType * mcanConfig,
                uint32_t id, MCan_IdType idType, MCan_DlcType dlc, uint8_t * data )
{
        Mcan * mcan = mcanConfig->pMCan;
        uint32_t   putIdx = 255;
        uint32_t * pThisTxBuf = 0;
        uint8_t  * pTxData;
        uint8_t    cnt;

        // Configured for FifoQ and FifoQ not full?
        if (( mcanConfig->nmbrTxFifoQElmts > 0 ) &&
                        (( mcan->MCAN_TXFQS & MCAN_TXFQS_TFQF ) == 0 )) {
                putIdx = ( mcan->MCAN_TXFQS & MCAN_TXFQS_TFQPI_Msk ) >>
MCAN_TXFQS_TFQPI_Pos;
                pThisTxBuf = mcanConfig->msgRam.pTxDedBuf + (putIdx *
                        (mcanConfig->txBufElmtSize & ELMT_SIZE_MASK));
                if ( idType == CAN_STD_ID )
                        *pThisTxBuf++ = (( id << 18 ) & ( CAN_11_BIT_ID_MASK << 18 ));
                else
                        *pThisTxBuf++ = BUFFER_XTD_MASK | ( id &
CAN_29_BIT_ID_MASK );
                *pThisTxBuf++ = (uint32_t) dlc << 16;
                pTxData = (uint8_t *) pThisTxBuf;
                for ( cnt = 0; cnt < dlc ; cnt++ ) {
                        *pTxData++ = *data++;
                }
                /* enable transmit from buffer to set TC interrupt bit in IR, but
                interrupt will not happen unless TC interrupt is enabled */
                mcan->MCAN_TXBTIE = ( 1 << putIdx);
                // request to send
                mcan->MCAN_TXBAR = ( 1 << putIdx );
        }
        SCB_CleanInvalidateDCache();
        return putIdx;  // now it points to the data field
}

/**
 * \brief Check if data transmitted from buffer/fifo/queue
 * \param mcanConfig  Pointer to a MCAN instance.
 * \param buffer  Pointer to data buffer.
 */
uint8_t MCAN_IsBufferTxd( const MCan_ConfigType * mcanConfig, uint8_t buffer )
{
        Mcan * mcan = mcanConfig->pMCan;
```

```
        return ( mcan->MCAN_TXBTO & ( 1 << buffer ) );
}

/**
 * \brief Configure RX Buffer Filter
 * ID must match exactly for a RX Buffer Filter
 * \param mcanConfig  Pointer to a MCAN instance.
 * \param buffer  Pointer to data buffer.
 * \param filter  data of filter.
 * \param idType  Type of ID
 */
void MCAN_ConfigRxBufferFilter( const MCan_ConfigType * mcanConfig,
                uint32_t buffer, uint32_t filter, uint32_t id, MCan_IdType idType)
{
        uint32_t * pThisRxFilt = 0;

        if ( buffer < mcanConfig->nmbrRxDedBufElmts ) {
                if ( idType == CAN_STD_ID ) {
                        if (( filter < mcanConfig->nmbrStdFilts )
                                        && ( id <= CAN_11_BIT_ID_MASK )) {
                                pThisRxFilt = mcanConfig->msgRam.pStdFilts + filter;
                                // 1 word per filter
                                *pThisRxFilt = STD_FILT_SFEC_BUFFER | (id << 16) |
                                                STD_FILT_SFID2_RX_BUFFER | buffer;
                        }
                } else {
                // extended ID
                        if (( filter < mcanConfig->nmbrExtFilts ) &&
                                        ( id <= CAN_29_BIT_ID_MASK )) {
                                pThisRxFilt = mcanConfig->msgRam.pExtFilts + (2 * filter);
                                // 2 words per filter
                                *pThisRxFilt++ = (uint32_t) EXT_FILT_EFEC_BUFFER | id;
                                *pThisRxFilt = EXT_FILT_EFID2_RX_BUFFER | buffer;
                        }
                }
        }
        SCB_CleanInvalidateDCache();
}

/**
 * \brief Configure Classic Filter
 * Classic Filters direct accepted messages to a FIFO & include both a ID and
 * a ID mask
 * \param mcanConfig  Pointer to a MCAN instance.
```

```
 * \param buffer  Pointer to data buffer.
 * \param fifo   fifo Number.
 * \param filter  data of filter.
 * \param idType  Type of ID
 * \param mask  Mask to be match
 */
void MCAN_ConfigRxClassicFilter( const MCan_ConfigType * mcanConfig,
                MCan_FifoType fifo, uint8_t filter, uint32_t id,
                MCan_IdType idType, uint32_t mask )
{
        uint32_t * pThisRxFilt = 0;
        uint32_t   filterTemp;

        if ( idType == CAN_STD_ID ) {
                if (( filter < mcanConfig->nmbrStdFilts ) && ( id <= CAN_11_BIT_ID_MASK )
                                && ( mask <= CAN_11_BIT_ID_MASK )) {
                        pThisRxFilt = mcanConfig->msgRam.pStdFilts + filter;
                        // 1 word per filter
                        filterTemp = (uint32_t) STD_FILT_SFT_CLASSIC | (id << 16) | mask;
                        if ( fifo == CAN_FIFO_0 ) {
                                *pThisRxFilt = STD_FILT_SFEC_FIFO0 | filterTemp;
                        } else if ( fifo == CAN_FIFO_1 ) {
                                *pThisRxFilt = STD_FILT_SFEC_FIFO1 | filterTemp;
                        }
                } else {
                        // extended ID
                        if (( filter < mcanConfig->nmbrExtFilts )
                                        && ( id <= CAN_29_BIT_ID_MASK )
                                        && ( mask <= CAN_29_BIT_ID_MASK )) {
                                pThisRxFilt = mcanConfig->msgRam.pExtFilts + (2 * filter);
                                // 2 words per filter
                                if ( fifo == CAN_FIFO_0 ) {
                                        *pThisRxFilt++ = EXT_FILT_EFEC_FIFO0 | id;
                                } else if ( fifo == CAN_FIFO_0 ) {
                                        *pThisRxFilt++ = EXT_FILT_EFEC_FIFO1 | id;
                                }
                                *pThisRxFilt = (uint32_t) EXT_FILT_EFT_CLASSIC | mask;
                        }
                }
        }
        SCB_CleanInvalidateDCache();
}

/**
 * \brief check if data received into buffer
```

```
 * \param mcanConfig  Pointer to a MCAN instance.
 * \param buffer  Pointer to data buffer.
 */
uint8_t MCAN_IsNewDataInRxDedBuffer( const MCan_ConfigType * mcanConfig,
                                                                uint8_t buffer )
{
        Mcan * mcan = mcanConfig->pMCan;

        SCB_CleanInvalidateDCache();

        if ( buffer < 32 ) {
          return ( mcan->MCAN_NDAT1 & ( 1 << buffer ));
        } else if ( buffer < 64 ) {
          return ( mcan->MCAN_NDAT1 & ( 1 << (buffer - 32 )));
        }
        else
          return 0;
}

/**
 * \brief Get Rx buffer
 * \param mcanConfig  Pointer to a MCAN instance.
 * \param buffer  Pointer to data buffer.
 * \param pRxMailbox  Pointer to rx Mailbox.
 */
void MCAN_GetRxDedBuffer( const MCan_ConfigType * mcanConfig,
                                        uint8_t buffer, Mailbox64Type * pRxMailbox )
{
        Mcan     * mcan = mcanConfig->pMCan;
        uint32_t * pThisRxBuf = 0;
        uint32_t   tempRy;  // temp copy of RX buffer word
        uint8_t  * pRxData;
        uint8_t    idx;

        SCB_CleanInvalidateDCache();

        if ( buffer < mcanConfig->nmbrRxDedBufElmts ) {
                pThisRxBuf = mcanConfig->msgRam.pRxDedBuf
                        + (buffer * (mcanConfig->rxBufElmtSize & ELMT_SIZE_MASK));
                tempRy = *pThisRxBuf++;  // word R0 contains ID
                if ( tempRy & BUFFER_XTD_MASK ) {
                        // extended ID?
                        pRxMailbox->info.id = tempRy & BUFFER_EXT_ID_MASK;
                } else {
                        // standard ID
```

```
                pRxMailbox->info.id = ( tempRy & BUFFER_STD_ID_MASK) >> 18;
            }
            tempRy = *pThisRxBuf++;  // word R1 contains DLC & time stamp
            pRxMailbox->info.length = (tempRy & BUFFER_DLC_MASK) >> 16;
            pRxMailbox->info.timestamp = tempRy & BUFFER_RXTS_MASK;
            // copy the data from the buffer to the mailbox
            pRxData = (uint8_t *) pThisRxBuf;
            for ( idx = 0; idx < pRxMailbox->info.length; idx++ )
                pRxMailbox->data[idx] = *pRxData++;
            /* clear the new data flag for the buffer */
            if ( buffer < 32 ) {
                mcan->MCAN_NDAT1 = ( 1 << buffer );
            } else {
                mcan->MCAN_NDAT1 = ( 1 << (buffer - 32 ));
            }
        }
}

/**
 * \brief Get from the receive FIFO and place in a application mailbox
 * \param mcanConfig  Pointer to a MCAN instance.
 * \param fifo  Fifo Number
 * \param pRxMailbox  Pointer to rx Mailbox.
 * \return: # of fifo entries at the start of the function
 *      0 -> FIFO was empty at start
 *      1 -> FIFO had 1 entry at start, but is empty at finish
 *      2 -> FIFO had 2 entries at start, has 1 entry at finish
 */
uint32_t MCAN_GetRxFifoBuffer( const MCan_ConfigType * mcanConfig,
            MCan_FifoType fifo, Mailbox64Type * pRxMailbox )
{
        Mcan     * mcan = mcanConfig->pMCan;
        uint32_t * pThisRxBuf = 0;
        uint32_t   tempRy;  // temp copy of RX buffer word
        uint8_t  * pRxData;
        uint8_t    idx;
        uint32_t * fifo_ack_reg;
        uint32_t   get_index;
        uint32_t   fill_level;
        uint32_t   element_size;

        SCB_CleanInvalidateDCache();

        // default: fifo empty
        fill_level = 0;
```

```
        if ( fifo == CAN_FIFO_0 ) {
                get_index = ( mcan->MCAN_RXF0S & MCAN_RXF0S_F0GI_Msk ) >>
MCAN_RXF0S_F0GI_Pos;
                fill_level = ( mcan->MCAN_RXF0S & MCAN_RXF0S_F0FL_Msk ) >>
MCAN_RXF0S_F0FL_Pos;
                pThisRxBuf = mcanConfig->msgRam.pRxFifo0;
                element_size = mcanConfig->rxFifo0ElmtSize & ELMT_SIZE_MASK;
                fifo_ack_reg = (uint32_t *) &mcan->MCAN_RXF0A;
        } else if ( fifo == CAN_FIFO_1 ) {
                get_index = ( mcan->MCAN_RXF1S & MCAN_RXF1S_F1GI_Msk ) >>
MCAN_RXF1S_F1GI_Pos;
                fill_level = ( mcan->MCAN_RXF1S & MCAN_RXF1S_F1FL_Msk ) >>
MCAN_RXF1S_F1FL_Pos;
                pThisRxBuf = mcanConfig->msgRam.pRxFifo1;
                element_size = mcanConfig->rxFifo1ElmtSize & ELMT_SIZE_MASK;
                fifo_ack_reg = (uint32_t *) &mcan->MCAN_RXF1A;
        }

        if ( fill_level > 0 ) {
                pThisRxBuf = pThisRxBuf + (get_index * element_size);
                tempRy = *pThisRxBuf++;  // word R0 contains ID
                if ( tempRy & BUFFER_XTD_MASK ) {
                        // extended ID?
                        pRxMailbox->info.id = tempRy & BUFFER_EXT_ID_MASK;
                } else {
                        // standard ID
                        pRxMailbox->info.id = ( tempRy & BUFFER_STD_ID_MASK) >> 18;
                }
                tempRy = *pThisRxBuf++;  // word R1 contains DLC & timestamps
                pRxMailbox->info.length = (tempRy & BUFFER_DLC_MASK) >> 16;
                pRxMailbox->info.timestamp = tempRy & BUFFER_RXTS_MASK;
                /* copy the data from the buffer to the mailbox */
                pRxData = (uint8_t *) pThisRxBuf;
                for ( idx = 0; idx < pRxMailbox->info.length; idx++ )
                        pRxMailbox->data[idx] = *pRxData++;
                // acknowledge reading the fifo entry
                *fifo_ack_reg = get_index;
                /* return entries remaining in FIFO */
        }
        return ( fill_level );
}

/**@}*/
```

mcan.h

```
/* ----------------------------------------------------------------------------
 *         SAM Software Package License
 * ----------------------------------------------------------------------------
 * Copyright (c) 2011, Atmel Corporation
 *
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * - Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the disclaimer below.
 *
 * Atmel's name may not be used to endorse or promote products derived from
 * this software without specific prior written permission.
 *
 * DISCLAIMER: THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY
EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-
INFRINGEMENT ARE
 * DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT,
INDIRECT,
 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA,
 * OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * ----------------------------------------------------------------------------
 */

/**
 *  \file
 *
 *  \section Purpose
 *
 *  Interface for configuring and using Timer Counter (TC) peripherals.
 *
```

```
 *  \section Usage
 *  -# Optionally, use TC_FindMckDivisor() to let the program find the best
 *     TCCLKS field value automatically.
 *  -# Configure a Timer Counter in the desired mode using TC_Configure().
 *  -# Start or stop the timer clock using TC_Start() and TC_Stop().
 */

#ifndef _MCAN_
#define _MCAN_

/*------------------------------------------------------------------------------
 *         Headers
 *------------------------------------------------------------------------------*/

#include "chip.h"

#include <stdint.h>

/*------------------------------------------------------------------------------
 *         Global functions
 *------------------------------------------------------------------------------*/

#ifdef __cplusplus
 extern "C" {
#endif

typedef enum
{
        CAN_STD_ID = 0,
        CAN_EXT_ID = 1
} MCan_IdType;

typedef enum
{
        CAN_DLC_0 = 0,
        CAN_DLC_1 = 1,
        CAN_DLC_2 = 2,
        CAN_DLC_3 = 3,
        CAN_DLC_4 = 4,
        CAN_DLC_5 = 5,
        CAN_DLC_6 = 6,
        CAN_DLC_7 = 7,
        CAN_DLC_8 = 8,
        CAN_DLC_12 = 9,
        CAN_DLC_16 = 10,
```

```
        CAN_DLC_20 = 11,
        CAN_DLC_24 = 12,
        CAN_DLC_32 = 13,
        CAN_DLC_48 = 14,
        CAN_DLC_64 = 15
} MCan_DlcType;

typedef enum
{
        CAN_FIFO_0 = 0,
        CAN_FIFO_1 = 1
} MCan_FifoType;

typedef enum
{
        CAN_INTR_LINE_0 = 0,
        CAN_INTR_LINE_1 = 1
} MCan_IntrLineType;

typedef struct MailboxInfoTag
{
        uint32_t   id;
        uint32_t   length;
        uint32_t   timestamp;
} MailboxInfoType;


typedef struct MailBox8Tag
{
        MailboxInfoType info;
        uint8_t        data[8];
} Mailbox8Type;

typedef struct MailBox12Tag
{
        MailboxInfoType info;
        uint8_t        data[12];
} Mailbox12Type;

typedef struct MailBox16Tag
{
        MailboxInfoType info;
        uint8_t        data[16];
} Mailbox16Type;
```

```
typedef struct MailBox20Tag
{
        MailboxInfoType info;
        uint8_t        data[20];
} Mailbox20Type;

typedef struct MailBox24Tag
{
        MailboxInfoType info;
        uint8_t        data[24];
} Mailbox24Type;

typedef struct MailBox32Tag
{
        MailboxInfoType info;
        uint8_t        data[32];
} Mailbox32ype;

typedef struct MailBox48Tag
{
        MailboxInfoType info;
        uint8_t        data[48];
} Mailbox48Type;

typedef struct MailBox64Tag
{
        MailboxInfoType info;
        uint8_t        data[64];
} Mailbox64Type;



typedef struct MCan_MsgRamPntrsTag
{
        uint32_t * pStdFilts;
        uint32_t * pExtFilts;
        uint32_t * pRxFifo0;
        uint32_t * pRxFifo1;
        uint32_t * pRxDedBuf;
        uint32_t * pTxEvtFifo;
        uint32_t * pTxDedBuf;
        uint32_t * pTxFifoQ;
} MCan_MsgRamPntrs;

typedef struct MCan_ConfigTag
```

```
{
        Mcan          * pMCan;
        uint32_t        bitTiming;
        uint32_t        fastBitTiming;
        uint32_t        nmbrStdFilts;
        uint32_t        nmbrExtFilts;
        uint32_t        nmbrFifo0Elmts;
        uint32_t        nmbrFifo1Elmts;
        uint32_t        nmbrRxDedBufElmts;
        uint32_t        nmbrTxEvtFifoElmts;
        uint32_t        nmbrTxDedBufElmts;
        uint32_t        nmbrTxFifoQElmts;
        uint32_t        rxFifo0ElmtSize;
        uint32_t        rxFifo1ElmtSize;
        uint32_t        rxBufElmtSize;
        // Element sizes and data sizes (encoded element size)
        uint32_t        txBufElmtSize;
        // Element size and data size (encoded element size)
        MCan_MsgRamPntrs  msgRam;
} MCan_ConfigType;

extern const MCan_ConfigType mcan0Config;
extern const MCan_ConfigType mcan1Config;

__STATIC_INLINE uint32_t MCAN_IsTxComplete(
        const MCan_ConfigType * mcanConfig )
{
        Mcan * mcan = mcanConfig->pMCan;
        return ( mcan->MCAN_IR & MCAN_IR_TC );
}

__STATIC_INLINE void MCAN_ClearTxComplete(
        const MCan_ConfigType * mcanConfig )
{
        Mcan * mcan = mcanConfig->pMCan;
        mcan->MCAN_IR = MCAN_IR_TC;
}

__STATIC_INLINE uint32_t MCAN_IsMessageStoredToRxDedBuffer(
        const MCan_ConfigType * mcanConfig )
{
  Mcan * mcan = mcanConfig->pMCan;

  return ( mcan->MCAN_IR & MCAN_IR_DRX );
}
```

```
__STATIC_INLINE void MCAN_ClearMessageStoredToRxBuffer(
        const MCan_ConfigType * mcanConfig )
{
        Mcan * mcan = mcanConfig->pMCan;
        mcan->MCAN_IR = MCAN_IR_DRX;
}

__STATIC_INLINE uint32_t MCAN_IsMessageStoredToRxFifo0(
        const MCan_ConfigType * mcanConfig )
{
        Mcan * mcan = mcanConfig->pMCan;
        return ( mcan->MCAN_IR & MCAN_IR_RF0N );
}

__STATIC_INLINE void MCAN_ClearMessageStoredToRxFifo0(
        const MCan_ConfigType * mcanConfig )
{
        Mcan * mcan = mcanConfig->pMCan;
        mcan->MCAN_IR = MCAN_IR_RF0N;
}

__STATIC_INLINE uint32_t MCAN_IsMessageStoredToRxFifo1(
        const MCan_ConfigType * mcanConfig )
{
        Mcan * mcan = mcanConfig->pMCan;
        return ( mcan->MCAN_IR & MCAN_IR_RF1N );
}

__STATIC_INLINE void MCAN_ClearMessageStoredToRxFifo1(
        const MCan_ConfigType * mcanConfig )
{
        Mcan * mcan = mcanConfig->pMCan;
        mcan->MCAN_IR = MCAN_IR_RF1N;
}

void MCAN_Init(
        const MCan_ConfigType * mcanConfig );

void MCAN_InitFdEnable(
        const MCan_ConfigType * mcanConfig );

void MCAN_InitFdBitRateSwitchEnable(
        const MCan_ConfigType * mcanConfig );
```

```
void MCAN_InitTxQueue(
        const MCan_ConfigType * mcanConfig );

void MCAN_InitLoopback(
        const MCan_ConfigType * mcanConfig );

void MCAN_Enable(
        const MCan_ConfigType * mcanConfig );

void MCAN_RequestIso11898_1(
        const MCan_ConfigType * mcanConfig );

void MCAN_RequestFd(
        const MCan_ConfigType * mcanConfig );

void MCAN_RequestFdBitRateSwitch(
        const MCan_ConfigType * mcanConfig );

void MCAN_LoopbackOn(
        const MCan_ConfigType * mcanConfig );

void MCAN_LoopbackOff(
        const MCan_ConfigType * mcanConfig );

void MCAN_IEnableMessageStoredToRxDedBuffer(
        const MCan_ConfigType * mcanConfig,
        MCan_IntrLineType line );

uint8_t  * MCAN_ConfigTxDedBuffer(
        const MCan_ConfigType * mcanConfig,
        uint8_t buffer,
        uint32_t id,
        MCan_IdType idType,
        MCan_DlcType dlc );

void MCAN_SendTxDedBuffer(
        const MCan_ConfigType * mcanConfig,
        uint8_t buffer );

uint32_t MCAN_AddToTxFifoQ(
        const MCan_ConfigType * mcanConfig,
        uint32_t id, MCan_IdType idType,
        MCan_DlcType dlc, uint8_t * data );

uint8_t MCAN_IsBufferTxd(
```

```
        const MCan_ConfigType * mcanConfig,
        uint8_t buffer );

void MCAN_ConfigRxBufferFilter(
        const MCan_ConfigType * mcanConfig,
        uint32_t buffer,
        uint32_t filter,
        uint32_t id,
        MCan_IdType idType);

void MCAN_ConfigRxClassicFilter(
        const MCan_ConfigType * mcanConfig,
        MCan_FifoType fifo,
        uint8_t filter,
        uint32_t id,
        MCan_IdType idType,
        uint32_t mask );

uint8_t MCAN_IsNewDataInRxDedBuffer(
        const MCan_ConfigType * mcanConfig,
        uint8_t buffer );

void MCAN_GetRxDedBuffer(
        const MCan_ConfigType * mcanConfig,
        uint8_t buffer,
        Mailbox64Type * pRxMailbox );

uint32_t MCAN_GetRxFifoBuffer(
        const MCan_ConfigType * mcanConfig,
        MCan_FifoType fifo,
        Mailbox64Type * pRxMailbox );

#ifdef __cplusplus
}
#endif

#endif /* #ifndef _MCAN_ */
```

mcan_config.h

```
/* ----------------------------------------------------------------------------
 *         SAM Software Package License
 * ----------------------------------------------------------------------------
 * Copyright (c) 2011, Atmel Corporation
 *
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * - Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the disclaimer below.
 *
 * Atmel's name may not be used to endorse or promote products derived from
 * this software without specific prior written permission.
 *
 * DISCLAIMER: THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY
EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-
INFRINGEMENT ARE
 * DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT,
INDIRECT,
 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA,
 * OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * ----------------------------------------------------------------------------
 */

/**
 *  \file
 *
 *  \section Purpose
 *
 *  Interface for configuring and using Timer Counter (TC) peripherals.
 *
```

```
 *  \section Usage
 *  -# Optionally, use TC_FindMckDivisor() to let the program find the best
 *     TCCLKS field value automatically.
 *  -# Configure a Timer Counter in the desired mode using TC_Configure().
 *  -# Start or stop the timer clock using TC_Start() and TC_Stop().
 */

#ifndef _MCAN_CONFIG_
#define _MCAN_CONFIG_

/*------------------------------------------------------------------------------
 *         Headers
 *------------------------------------------------------------------------------*/


/*------------------------------------------------------------------------------
 *         Global functions
 *------------------------------------------------------------------------------*/

#ifdef __cplusplus
 extern "C" {
#endif

/* Programmable Clock Source for Baud Rate is Common To Both MCAN Controllers */
#define MCAN_PROG_CLK_PRESCALER      1   /* /1 to /256 */
// select one of the following for the programmable clock source
//#define MCAN_PROG_CLK_SELECT        PMC_PCK_CSS_SLOW_CLK
//#define MCAN_PROG_CLK_SELECT        PMC_PCK_CSS_MAIN_CLK
//#define MCAN_PROG_CLK_SELECT        PMC_PCK_CSS_PLLA_CLK
//#define MCAN_PROG_CLK_SELECT        PMC_PCK_CSS_UPLL_CLK
#define MCAN_PROG_CLK_SELECT        PMC_PCK_CSS_MCK
#define MCAN_PROG_CLK_FREQ_HZ \
                ( (float) 150000000 / (float) MCAN_PROG_CLK_PRESCALER )

#define MCAN0_BIT_RATE_BPS          100000
#define MCAN0_PROP_SEG              2
#define MCAN0_PHASE_SEG1            11
#define MCAN0_PHASE_SEG2            11
#define MCAN0_SYNC_JUMP             4

#define MCAN0_FAST_BIT_RATE_BPS     2000000
#define MCAN0_FAST_PROP_SEG         2
#define MCAN0_FAST_PHASE_SEG1       4
#define MCAN0_FAST_PHASE_SEG2       4
#define MCAN0_FAST_SYNC_JUMP        2
```

```
#define MCAN0_NMBR_STD_FLTS          8  /* 128 max filters */
#define MCAN0_NMBR_EXT_FLTS          8  /* 64 max filters */
#define MCAN0_NMBR_RX_FIFO0_ELMTS    0  /* # of elements, 64 elements max */
#define MCAN0_NMBR_RX_FIFO1_ELMTS    0  /* # of elements, 64 elements max */
#define MCAN0_NMBR_RX_DED_BUF_ELMTS  16 /* # of elements, 64 elements max */
#define MCAN0_NMBR_TX_EVT_FIFO_ELMTS 0  /* # of elements, 32 elements max */
#define MCAN0_NMBR_TX_DED_BUF_ELMTS  4  /* # of elements, 32 elements max */
#define MCAN0_NMBR_TX_FIFO_Q_ELMTS   0  /* # of elements, 32 elements max */
#define MCAN0_RX_FIFO0_ELMT_SZ       8  /* 8, 12, 16, 20, 24, 32, 48, 64 bytes */
#define MCAN0_RX_FIFO1_ELMT_SZ       8  /* 8, 12, 16, 20, 24, 32, 48, 64 bytes */
#define MCAN0_RX_BUF_ELMT_SZ         8  /* 8, 12, 16, 20, 24, 32, 48, 64 bytes */
#define MCAN0_TX_BUF_ELMT_SZ         8  /* 8, 12, 16, 20, 24, 32, 48, 64 bytes */

#define MCAN1_BIT_RATE_BPS           100000
#define MCAN1_PROP_SEG               2
#define MCAN1_PHASE_SEG1             11
#define MCAN1_PHASE_SEG2             11
#define MCAN1_SYNC_JUMP              4

#define MCAN1_FAST_BIT_RATE_BPS      2000000
#define MCAN1_FAST_PROP_SEG          2
#define MCAN1_FAST_PHASE_SEG1        4
#define MCAN1_FAST_PHASE_SEG2        4
#define MCAN1_FAST_SYNC_JUMP         2

#define MCAN1_NMBR_STD_FLTS          8   /* 128 max filters */
#define MCAN1_NMBR_EXT_FLTS          8   /* 64 max filters */
#define MCAN1_NMBR_RX_FIFO0_ELMTS    12  /* # of elements, 64 elements max */
#define MCAN1_NMBR_RX_FIFO1_ELMTS    0   /* # of elements, 64 elements max */
#define MCAN1_NMBR_RX_DED_BUF_ELMTS  4   /* # of elements, 64 elements max */
#define MCAN1_NMBR_TX_EVT_FIFO_ELMTS 0   /* # of elements, 32 elements max */
#define MCAN1_NMBR_TX_DED_BUF_ELMTS  4   /* # of elements, 32 elements max */
#define MCAN1_NMBR_TX_FIFO_Q_ELMTS   4   /* # of elements, 32 elements max */
#define MCAN1_RX_FIFO0_ELMT_SZ       8   /* 8, 12, 16, 20, 24, 32, 48, 64 bytes */
#define MCAN1_RX_FIFO1_ELMT_SZ       8   /* 8, 12, 16, 20, 24, 32, 48, 64 bytes */
#define MCAN1_RX_BUF_ELMT_SZ         64  /* 8, 12, 16, 20, 24, 32, 48, 64 bytes */
#define MCAN1_TX_BUF_ELMT_SZ         32  /* 8, 12, 16, 20, 24, 32, 48, 64 bytes */

#ifdef __cplusplus
}
#endif

#endif /* #ifndef _MCAN_CONFIG_ */
```

pio.c

```
/* ----------------------------------------------------------------------------
 *         SAM Software Package License
 * ----------------------------------------------------------------------------
 * Copyright (c) 2014, Atmel Corporation
 *
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * - Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the disclaimer below.
 *
 * Atmel's name may not be used to endorse or promote products derived from
 * this software without specific prior written permission.
 *
 * DISCLAIMER: THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY
EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-
INFRINGEMENT ARE
 * DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT,
INDIRECT,
 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA,
 * OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * ----------------------------------------------------------------------------
 */

/** \file */

/*----------------------------------------------------------------------------
 *        Headers
 *----------------------------------------------------------------------------*/
#include "chip.h"
```

```
/*----------------------------------------------------------------------------
 *        Local functions
 *----------------------------------------------------------------------------*/

/**
 * \brief Configures one or more pin(s) of a PIO controller as being controlled by
 * peripheral A. Optionally, the corresponding internal pull-up(s) can be enabled.
 *
 * \param pio  Pointer to a PIO controller.
 * \param mask  Bitmask of one or more pin(s) to configure.
 * \param enablePullUp  Indicates if the pin(s) internal pull-up shall be
 *                      configured.
 */
static void PIO_SetPeripheralA(
                Pio *pio,
                unsigned int mask,
                unsigned char enablePullUp)
{
        unsigned int abcdsr;
        /* Disable interrupts on the pin(s) */
        pio->PIO_IDR = mask;

        /* Enable the pull-up(s) if necessary */
        if (enablePullUp) {
                pio->PIO_PUER = mask;
        } else {
                pio->PIO_PUDR = mask;
        }

        abcdsr = pio->PIO_ABCDSR[0];
        pio->PIO_ABCDSR[0] &= (~mask & abcdsr);
        abcdsr = pio->PIO_ABCDSR[1];
        pio->PIO_ABCDSR[1] &= (~mask & abcdsr);
        pio->PIO_PDR = mask;
}

/**
 * \brief Configures one or more pin(s) of a PIO controller as being controlled
 * by peripheral B. Optionally, the corresponding internal pull-up(s) can be
 * enabled.
 *
 * \param pio  Pointer to a PIO controller.
 * \param mask  Bitmask of one or more pin(s) to configure.
 * \param enablePullUp  Indicates if the pin(s) internal pull-up shall be
 *                      configured.
```

```
 */
static void PIO_SetPeripheralB(
                Pio *pio,
                unsigned int mask,
                unsigned char enablePullUp)
{
        unsigned int abcdsr;
        /* Disable interrupts on the pin(s) */
        pio->PIO_IDR = mask;

        /* Enable the pull-up(s) if necessary */
        if (enablePullUp) {
                pio->PIO_PUER = mask;
        } else {
                pio->PIO_PUDR = mask;
        }

        abcdsr = pio->PIO_ABCDSR[0];
        pio->PIO_ABCDSR[0] = (mask | abcdsr);
        abcdsr = pio->PIO_ABCDSR[1];
        pio->PIO_ABCDSR[1] &= (~mask & abcdsr);

        pio->PIO_PDR = mask;
}

/**
 * \brief Configures one or more pin(s) of a PIO controller as being controlled
 * by peripheral C. Optionally, the corresponding internal pull-up(s) can be
 * enabled.
 *
 * \param pio  Pointer to a PIO controller.
 * \param mask  Bitmask of one or more pin(s) to configure.
 * \param enablePullUp  Indicates if the pin(s) internal pull-up shall be
 *                  configured.
 */
static void PIO_SetPeripheralC(
                Pio *pio,
                unsigned int mask,
                unsigned char enablePullUp)
{
        unsigned int abcdsr;
        /* Disable interrupts on the pin(s) */
        pio->PIO_IDR = mask;

        /* Enable the pull-up(s) if necessary */
```

```
        if (enablePullUp) {
                pio->PIO_PUER = mask;
        } else {
                pio->PIO_PUDR = mask;
        }

        abcdsr = pio->PIO_ABCDSR[0];
        pio->PIO_ABCDSR[0] &= (~mask & abcdsr);
        abcdsr = pio->PIO_ABCDSR[1];
        pio->PIO_ABCDSR[1] = (mask | abcdsr);

        pio->PIO_PDR = mask;
}

/**
 * \brief Configures one or more pin(s) of a PIO controller as being controlled
 * by peripheral D. Optionally, the corresponding internal pull-up(s) can be
 * enabled.
 *
 * \param pio  Pointer to a PIO controller.
 * \param mask  Bitmask of one or more pin(s) to configure.
 * \param enablePullUp  Indicates if the pin(s) internal pull-up shall be
 *                      configured.
 */
static void PIO_SetPeripheralD(
                Pio *pio,
                unsigned int mask,
                unsigned char enablePullUp)
{
        unsigned int abcdsr;
        /* Disable interrupts on the pin(s) */
        pio->PIO_IDR = mask;

        /* Enable the pull-up(s) if necessary */
        if (enablePullUp) {
                pio->PIO_PUER = mask;
        } else {
                pio->PIO_PUDR = mask;
        }

        abcdsr = pio->PIO_ABCDSR[0];
        pio->PIO_ABCDSR[0] = (mask | abcdsr);
        abcdsr = pio->PIO_ABCDSR[1];
        pio->PIO_ABCDSR[1] = (mask | abcdsr);
```

```
	pio->PIO_PDR = mask;
}

/**
 * \brief Configures one or more pin(s) or a PIO controller as inputs. Optionally,
 * the corresponding internal pull-up(s) and glitch filter(s) can be enabled.
 *
 * \param pio  Pointer to a PIO controller.
 * \param mask  Bitmask indicating which pin(s) to configure as input(s).
 * \param enablePullUp  Indicates if the internal pull-up(s) must be enabled.
 * \param enableFilter  Indicates if the glitch filter(s) must be enabled.
 */
static void PIO_SetInput(
		Pio *pio,
		unsigned int mask,
		unsigned char attribute)
{
	/* Disable interrupts */
	pio->PIO_IDR = mask;

	/* Enable pull-up(s) if necessary */
	if (attribute & PIO_PULLUP)
		pio->PIO_PUER = mask;
	else
		pio->PIO_PUDR = mask;

	/* Enable Input Filter if necessary */
	if (attribute & (PIO_DEGLITCH | PIO_DEBOUNCE))
		pio->PIO_IFER = mask;
	else
		pio->PIO_IFDR = mask;

	/* Enable de-glitch or de-bounce if necessary */
	if (attribute & PIO_DEGLITCH) {
		pio->PIO_IFSCDR = mask;
	} else {
		if (attribute & PIO_DEBOUNCE) {
			pio->PIO_IFSCER = mask;
		}
	}
	/* Configure pin as input */
	pio->PIO_ODR = mask;
	pio->PIO_PER = mask;
}
```

```
/**
 * \brief Configures one or more pin(s) of a PIO controller as outputs, with the
 * given default value. Optionally, the multi-drive feature can be enabled
 * on the pin(s).
 *
 * \param pio  Pointer to a PIO controller.
 * \param mask  Bitmask indicating which pin(s) to configure.
 * \param defaultValue  Default level on the pin(s).
 * \param enableMultiDrive  Indicates if the pin(s) shall be configured as
 *                      open-drain.
 * \param enablePullUp  Indicates if the pin shall have its pull-up activated.
 */
static void PIO_SetOutput(
                Pio *pio,
                unsigned int mask,
                unsigned char defaultValue,
                unsigned char enableMultiDrive,
                unsigned char enablePullUp)
{
        /* Disable interrupts */
        pio->PIO_IDR = mask;

        /* Enable pull-up(s) if necessary */
        if (enablePullUp) {
                pio->PIO_PUER = mask;
        } else {
                pio->PIO_PUDR = mask;
        }

        /* Enable multi-drive if necessary */
        if (enableMultiDrive) {
                pio->PIO_MDER = mask;
        } else {
                pio->PIO_MDDR = mask;
        }

        /* Set default value */
        if (defaultValue) {
                pio->PIO_SODR = mask;
        } else {
                pio->PIO_CODR = mask;
        }
        /* Configure pin(s) as output(s) */
        pio->PIO_OER = mask;
        pio->PIO_PER = mask;
```

```
}

/*----------------------------------------------------------------------------
 *        Global functions
 *----------------------------------------------------------------------------*/

/**
 * \brief Configures a list of Pin instances, each of which can either hold a
 * single pin or a group of pins, depending on the mask value; all pins are
 * configured by this function. The size of the array must also be provided and
 *  is easily computed using PIO_LISTSIZE whenever its length is not known in
 * advance.
 *
 * \param list  Pointer to a list of Pin instances.
 * \param size  Size of the Pin list (calculated using PIO_LISTSIZE).
 *
 * \return 1 if the pins have been configured properly; otherwise 0.
 */
uint8_t PIO_Configure( const Pin *list, uint32_t size )
{
        /* Configure pins */
        while ( size > 0 ) {
                switch ( list->type ) {
                case PIO_PERIPH_A:
                        PIO_SetPeripheralA(list->pio,
                                           list->mask,
                                           (list->attribute & PIO_PULLUP) ? 1 : 0);
                        break;

                case PIO_PERIPH_B:
                        PIO_SetPeripheralB(list->pio,
                                           list->mask,
                                           (list->attribute & PIO_PULLUP) ? 1 : 0);
                        break;

                case PIO_PERIPH_C:
                        PIO_SetPeripheralC(list->pio,
                                           list->mask,
                                           (list->attribute & PIO_PULLUP) ? 1 : 0);
                        break;

                case PIO_PERIPH_D:
                                PIO_SetPeripheralD(list->pio,
                                                   list->mask,
```

```
                                                        (list->attribute &
PIO_PULLUP) ? 1 : 0);
                                break;
                        case PIO_INPUT:
#ifndef __FPGA
                                PMC_EnablePeripheral(list->id);
#endif
                                PIO_SetInput(list->pio,
                                                list->mask,
                                                list->attribute);
                                break;

                        case PIO_OUTPUT_0:
                        case PIO_OUTPUT_1:
                                PIO_SetOutput(list->pio,
                                                list->mask,
                                                (list->type == PIO_OUTPUT_1),
                                                (list->attribute & PIO_OPENDRAIN) ? 1 : 0,
                                                (list->attribute & PIO_PULLUP) ? 1 : 0);
                                break;

                        default: return 0;
                        }
                        list++;
                        size--;
                }
                return 1;
}

/**
 * \brief Sets a high output level on all the PIOs defined in the given Pin
 * instance.
 * This has no immediate effects on PIOs that are not output, but the PIO
 * controller will memorize the value they are changed to outputs.
 *
 * \param pin  Pointer to a Pin instance describing one or more pins.
 */
void PIO_Set(const Pin *pin)
{
        pin->pio->PIO_SODR = pin->mask;
}

/**
 * \brief Sets a low output level on all the PIOs defined in the given Pin
 * instance.
```

```
 * This has no immediate effects on PIOs that are not output, but the PIO
 * controller will memorize the value they are changed to outputs.
 *
 * \param pin  Pointer to a Pin instance describing one or more pins.
 */
void PIO_Clear(const Pin *pin)
{
        pin->pio->PIO_CODR = pin->mask;
}

/**
 * \brief Returns 1 if one or more PIO of the given Pin instance currently have
 * a high level; otherwise returns 0. This method returns the actual value that
 * is being read on the pin. To return the supposed output value of a pin, use
 * PIO_GetOutputDataStatus() instead.
 *
 * \param pin  Pointer to a Pin instance describing one or more pins.
 *
 * \return 1 if the Pin instance contains at least one PIO that currently has
 * a high level; otherwise 0.
 */
unsigned char PIO_Get( const Pin *pin )
{
        unsigned int reg ;

        if ( (pin->type == PIO_OUTPUT_0) || (pin->type == PIO_OUTPUT_1) ) {
                reg = pin->pio->PIO_ODSR ;
        } else {
                reg = pin->pio->PIO_PDSR ;
        }

        if ( (reg & pin->mask) == 0 ) {
                return 0 ;
        } else {
                return 1 ;
        }
}

/**
 * \brief Returns 1 if one or more PIO of the given Pin are configured to output
 * a high level (even if they are not output).
 * To get the actual value of the pin, use PIO_Get() instead.
 *
 * \param pin  Pointer to a Pin instance describing one or more pins.
 *
```

```
 * \return 1 if the Pin instance contains at least one PIO that is configured
 * to output a high level; otherwise 0.
 */
unsigned char PIO_GetOutputDataStatus(const Pin *pin)
{
	if ((pin->pio->PIO_ODSR & pin->mask) == 0) {
		return 0;
	} else {
		return 1;
	}
}

/**
 * \brief Configures Glitch or Denouncing filter for input.
 *
 * \param pin  Pointer to a Pin instance describing one or more pins.
 * \param cuttoff  Cut off frequency for denounce filter.
 */
void PIO_SetDebounceFilter( const Pin *pin, uint32_t cuttoff )
{
	Pio *pio = pin->pio;

	pio->PIO_IFSCER = pin->mask; /* set Denouncing, 0 bit field no effect */
	pio->PIO_SCDR = ((32678/(2*(cuttoff))) - 1) & 0x3FFF;
	/* the lowest 14 bits work */
}

/**
 * \brief Enable write protect.
 *
 * \param pin  Pointer to a Pin instance describing one or more pins.
 */
void PIO_EnableWriteProtect( const Pin *pin )
{
	Pio *pio = pin->pio;

	pio->PIO_WPMR = ( PIO_WPMR_WPKEY_VALID | PIO_WPMR_WPEN_EN );
}

/**
 * \brief Disable write protect.
 *
 * \param pin  Pointer to a Pin instance describing one or more pins.
 */
```

```
void PIO_DisableWriteProtect( const Pin *pin )
{
        Pio *pio = pin->pio;

        pio->PIO_WPMR = ( PIO_WPMR_WPKEY_VALID | PIO_WPMR_WPEN_DIS );
}

/**
 * \brief Get write protect violation information.
 *
 * \param pin  Pointer to a Pin instance describing one or more pins.
 */
uint32_t PIO_GetWriteProtectViolationInfo( const Pin * pin )
{
        Pio *pio = pin->pio;
        return (pio->PIO_WPSR);
}

/**
 *  \brief Set pin type
 * the pin is controlled by the corresponding peripheral (A, B, C, D,E)
 * \param pin      Pointer to a Pin instance describing one or more pins.
 * \param pinType  PIO_PERIPH_A, PIO_PERIPH_B, ...
 */
void PIO_SetPinType( Pin * pin, uint8_t pinType)
{
        pin->type = pinType;
}

void PIN_Set( uint32_t dwCom )
{
 if(dwCom)
 {
   PIOA->PIO_WPMR &= 0x00;
   PIOA->PIO_PER   = 0x10;
   PIOA->PIO_OER  |= 0x10;
   PIOA->PIO_CODR |= 0x10;
   PIOA->PIO_OWER |= 0x10;
 }
 else
 {
   PIOA->PIO_WPMR &= 0x00;
   PIOA->PIO_PER   = 0x08;
   PIOA->PIO_OER  |= 0x08;
   PIOA->PIO_CODR |= 0x08;
```

```
    PIOA->PIO_OWER |= 0x08;
  }
}

void PIN_Clear( uint32_t dwCom )
{
  if(dwCom)
  {
    PIOA->PIO_WPMR &= 0x00;
    PIOA->PIO_PER   = 0x10;
    PIOA->PIO_OER  |= 0x10;
    PIOA->PIO_SODR |= 0x10;
    PIOA->PIO_OWER |= 0x10;
  }
  else
  {
    PIOA->PIO_WPMR &= 0x00;
    PIOA->PIO_PER   = 0x08;
    PIOA->PIO_OER  |= 0x08;
    PIOA->PIO_SODR |= 0x08;
    PIOA->PIO_OWER |= 0x08;
  }
}

void PIOHandlesFunction(void)
{
  PIOA->PIO_IDR = 0x18;
}

void PIN_MakeInputSDA( )
{
    PIOA->PIO_WPMR&= 0x00;
    PIOA->PIO_OWDR|= 0x08;
    PIOA->PIO_ODR |= 0x08;
    PIOA->PIO_IDR |= ~0x08;
    PIOA->PIO_ISR ;
    PIOA->PIO_IER |= 0x08;
    PIOA->PIO_ESR |= 0x08;

    PIOA->PIO_FELLSR  |= 0x08;
    PIOA->PIO_IFER    |= 0x08;
    PIOA->PIO_IFSCER  |= 0x04;
}

void PIN_MakeOutputSDA( )
```

```
{
   PIOA->PIO_WPMR &= 0;
   //enable IT
   PIOA->PIO_IDR  |= 0x08;
   //PIOA->PIO_PDR = 0x08;
   //PIOA->PIO_PER = 0x08;
   PIOA->PIO_IFDR |= 0x08;
   //PIOA->PIO_DIFSR |= 0x08;
   //PIOA->PIO_SCDR |= 0x04
   PIOA->PIO_OER  |= 0x08;
   PIOA->PIO_OWER |= 0x08;

}
```

pio.h

```
/* ----------------------------------------------------------------------------
 *         SAM Software Package License
 * ----------------------------------------------------------------------------
 * Copyright (c) 2012, Atmel Corporation
 *
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * - Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the disclaimer below.
 *
 * Atmel's name may not be used to endorse or promote products derived from
 * this software without specific prior written permission.
 *
 * DISCLAIMER: THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY
EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-
INFRINGEMENT ARE
 * DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT,
INDIRECT,
 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA,
 * OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * ----------------------------------------------------------------------------
 */

/**
 *  \file
 *
 *  \section Purpose
 *
 *  This file provides a basic API for PIO configuration and usage of
 *  user-controlled pins. Please refer to the board.h file for a list of
```

```
 *  available pin definitions.
 *
 *  \section Usage
 *
 *  -# Define a constant pin description array such as the following one, using
 *     the existing definitions provided by the board.h file if possible:
 *     \code
 *        const Pin pPins[] = {PIN_USART0_TXD, PIN_USART0_RXD};
 *     \endcode
 *     Alternatively, it is possible to add new pins by provided the full Pin
 *     structure:
 *     \code
 *     // Pin instance to configure PA10 & PA11 as inputs with the internal
 *     // pull-up enabled.
 *     const Pin pPins = {
 *          (1 << 10) | (1 << 11),
 *          REG_PIOA,
 *          ID_PIOA,
 *          PIO_INPUT,
 *          PIO_PULLUP
 *     };
 *     \endcode
 *  -# Configure a pin array by calling PIO_Configure() with a pointer to the
 *     array and its size (which is computed using the PIO_LISTSIZE macro).
 *  -# Change and get the value of a user-controlled pin using the PIO_Set,
 *     PIO_Clear and PIO_Get methods.
 *  -# Get the level being currently output by a user-controlled pin configured
 *     as an output using PIO_GetOutputDataStatus().
 */

#ifndef _PIO_
#define _PIO_

/*
 *         Headers
 */

#include "chip.h"

#include <stdint.h>

/*
 *         Global Definitions
 */
```

```
/**  The pin is controlled by the associated signal of peripheral A. */
#define PIO_PERIPH_A              0
/**  The pin is controlled by the associated signal of peripheral B. */
#define PIO_PERIPH_B              1
/**  The pin is controlled by the associated signal of peripheral C. */
#define PIO_PERIPH_C              2
/**  The pin is controlled by the associated signal of peripheral D. */
#define PIO_PERIPH_D              3
/**  The pin is an input. */
#define PIO_INPUT                 4
/**  The pin is an output and has a default level of 0. */
#define PIO_OUTPUT_0              5
/**  The pin is an output and has a default level of 1. */
#define PIO_OUTPUT_1              6

/**  Default pin configuration (no attribute). */
#define PIO_DEFAULT               (0 << 0)
/**  The internal pin pull-up is active. */
#define PIO_PULLUP                (1 << 0)
/**  The internal glitch filter is active. */
#define PIO_DEGLITCH              (1 << 1)
/**  The pin is open-drain. */
#define PIO_OPENDRAIN             (1 << 2)

/**  The internal debouncing filter is active. */
#define PIO_DEBOUNCE              (1 << 3)

/**  Enable additional interrupt modes. */
#define PIO_IT_AIME               (1 << 4)

/**  Interrupt High Level/Rising Edge detection is active. */
#define PIO_IT_RE_OR_HL           (1 << 5)
/**  Interrupt Edge detection is active. */
#define PIO_IT_EDGE               (1 << 6)

/**  Low level interrupt is active */
#define PIO_IT_LOW_LEVEL          (0              | 0 | PIO_IT_AIME)
/**  High level interrupt is active */
#define PIO_IT_HIGH_LEVEL         (PIO_IT_RE_OR_HL | 0 | PIO_IT_AIME)
/**  Falling edge interrupt is active */
#define PIO_IT_FALL_EDGE          (0              | PIO_IT_EDGE | PIO_IT_AIME)
/**  Rising edge interrupt is active */
#define PIO_IT_RISE_EDGE          (PIO_IT_RE_OR_HL | PIO_IT_EDGE | PIO_IT_AIME)
/** The WP is enable */
#define PIO_WPMR_WPEN_EN          ( 0x01     << 0 )
```

```
/** The WP is disable */
#define PIO_WPMR_WPEN_DIS        ( 0x00     << 0 )
/** Valid WP key */
#define PIO_WPMR_WPKEY_VALID     ( 0x50494F << 8 )
#ifdef __cplusplus
 extern "C" {
#endif

/*
 *         Global Macros
 */

/**
 *  Calculates the size of an array of Pin instances. The array must be defined
 *  locally (i.e. not a pointer), otherwise the computation will not be correct.
 *  \param pPins  Local array of Pin instances.
 *  \return Number of elements in array.
 */
#define PIO_LISTSIZE(pPins)    (sizeof(pPins) / sizeof(Pin))

/*
 *        Global Types
 */


/*
 *  Describes the type and attribute of one PIO pin or a group of similar pins.
 *  The #type# field can have the following values:
 *    - PIO_PERIPH_A
 *    - PIO_PERIPH_B
 *    - PIO_OUTPUT_0
 *    - PIO_OUTPUT_1
 *    - PIO_INPUT
 *
 *  The #attribute# field is a bitmask that can either be set to PIO_DEFAULt,
 *  or combine (using bitwise OR '|') any number of the following constants:
 *    - PIO_PULLUP
 *    - PIO_DEGLITCH
 *    - PIO_DEBOUNCE
 *    - PIO_OPENDRAIN
 *    - PIO_IT_LOW_LEVEL
 *    - PIO_IT_HIGH_LEVEL
 *    - PIO_IT_FALL_EDGE
 *    - PIO_IT_RISE_EDGE
 */
```

```
typedef struct _Pin
{
        /*  Bitmask indicating which pin(s) to configure. */
        uint32_t mask;
        /*  Pointer to the PIO controller which has the pin(s). */
        Pio    *pio;
        /*  Peripheral ID of the PIO controller which has the pin(s). */
        uint8_t id;
        /*  Pin type. */
        uint8_t type;
        /*  Pin attribute. */
        uint8_t attribute;
} Pin ;

/*
 *        Global Access Macros
 */

/*
 *        Global Functions
 */

extern uint8_t PIO_Configure( const Pin *list, uint32_t size ) ;

extern void PIO_Set( const Pin *pin ) ;

extern void PIO_Clear( const Pin *pin ) ;

extern uint8_t PIO_Get( const Pin *pin ) ;

extern uint8_t PIO_GetOutputDataStatus( const Pin *pin ) ;

extern void PIO_SetDebounceFilter( const Pin *pin, uint32_t cuttoff );

extern void PIO_EnableWriteProtect( const Pin *pin );

extern void PIO_DisableWriteProtect( const Pin *pin );

extern void PIO_SetPinType( Pin * pin, uint8_t pinType);

extern uint32_t PIO_GetWriteProtectViolationInfo( const Pin * pin );
#ifdef __cplusplus
}
#endif
#endif /* #ifndef _PIO_ */
```

pio_capture.c

```
/* ----------------------------------------------------------------------------
 *         SAM Software Package License
 * ----------------------------------------------------------------------------
 * Copyright (c) 2014, Atmel Corporation
 *
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * - Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the disclaimer below.
 *
 * Atmel's name may not be used to endorse or promote products derived from
 * this software without specific prior written permission.
 *
 * DISCLAIMER: THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY
EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-
INFRINGEMENT ARE
 * DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT,
INDIRECT,
 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA,
 * OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * ----------------------------------------------------------------------------
 */

/** \addtogroup pio_capture_module Working with PIO Parallel Capture Mode
 *  \ingroup peripherals_module
 * The PIO Parallel Capture Mode driver provides the interface to configure
 * and use the PIO Parallel Capture Mode peripheral.\n
 *
 * The PIO Controller integrates an interface able to read data from a CMOS digital
 * image sensor, a high-speed parallel ADC, a DSP synchronous port in synchronous
```

```
* mode, etc.... For better understanding and to ease reading, the following
* description uses an example with a CMOS digital image sensor
*
* To use the PIO Parallel Capture, the user has to follow these few steps:
* <ul>
*   <li> Enable PIOA peripheral clock </li>
*   <li> Configure the PDC </li>
*   <li> Configure the PIO Capture interrupt </li>
*   <li> Enable the PDC </li>
*   <li> Enable the PIO Capture </li>
*   <li> Wait for interrupt </li>
*   <li> Disable the interrupt </li>
*   <li> Read the DATA </li>
* </ul>
*
* For more accurate information, please look at the PIO Parallel Capture Mode
* section of the Datasheet.
*
* <b>API Usage:</b>
*
*  -# Configurate the interrupt for PIOA, can be done by
* PIO_InitializeInterrupts()
*  -# Initialize the PIO Parallel Capture API by filing the SpioCaptureInit
*     structure
* options:
*       - alwaysSampling: for sample data with or without take in account
*         ENABLE pins.
*       - halfSampling: for sample all data or only one time out of two
*  -# Call PIO_CaptureInit() for init and enable the PDC, init the PIO capture.
*  -# Call PIO_CaptureEnable() for enable the PIO Parallel Capture.
*  -# When an interrupt is received, the PIO_CaptureHandler() is call and the
*     respective callback is launch.
*  -# When the transfer is complete, the user need to disable interrupt with
*     PIO_CaptureDisableIt(). Otherwise, the PDC will send an interrupt.
*  -# The data receive by the PIO Parallel Capture is inside the buffer passed
*     in the PIO_CaptureInit().
*
* Related files :\n
* \ref pio_capture.c\n
* \ref pio_capture.h\n
*/
/*@{*/
/*@}*/
/**
 * \file
```

```
 *
 * Implementation of PIO Parallel Capture.
 *
 */
/*----------------------------------------------------------------------------
 *        Headers
 *----------------------------------------------------------------------------*/

#include "chip.h"

#include <assert.h>

#define PIO_PCISR_RXBUFF (0x1u<<3)
#define PIO_PCISR_ENDRX  (0x1u<<2)
/*----------------------------------------------------------------------------
 *        Local Functions
 *----------------------------------------------------------------------------*/
/** Copy the API structure for interrupt handler */
static SpioCaptureInit* _PioCaptureCopy;

/*----------------------------------------------------------------------------
 *        Global Functions
 *----------------------------------------------------------------------------*/

/*----------------------------------------------------------------------------*/
/**
 * \brief The PIO_CaptureHandler must be called by the PIO Capture Interrupt
 * Service Routine with the corresponding PIO Capture instance.
 */
/*----------------------------------------------------------------------------*/
extern void PIO_CaptureHandler( void )
{
        volatile uint32_t pio_captureSr;
        uint32_t k;

        /* Read the status register*/
        pio_captureSr = PIOA->PIO_PCISR ;
        k = pio_captureSr;
        pio_captureSr = k & PIOA->PIO_PCIMR ;

        if (pio_captureSr & PIO_PCISR_DRDY) {
                /* Parallel Capture Mode Data Ready */
                if ( _PioCaptureCopy->CbkDataReady != NULL ) {
                        _PioCaptureCopy->CbkDataReady( _PioCaptureCopy );
                } else {
```

```
                        TRACE_DEBUG("IT PIO Capture Data Ready received (no
callback)\n\r");
                }
        }

        if (pio_captureSr & PIO_PCISR_OVRE) {
                /* Parallel Capture Mode Overrun Error */
                if ( _PioCaptureCopy->CbkOverrun != NULL )      {
                        _PioCaptureCopy->CbkOverrun( _PioCaptureCopy );
                } else {
                        TRACE_DEBUG("IT PIO Capture Overrun Error received (no
callback)\n\r");
                }
        }

        if (pio_captureSr & PIO_PCISR_RXBUFF) {
                /* Reception Buffer Full */
                if ( _PioCaptureCopy->CbkBuffFull != NULL ) {
                        _PioCaptureCopy->CbkBuffFull( _PioCaptureCopy );
                } else {
                        TRACE_DEBUG("IT PIO Capture Reception Buffer Full received \
                                (no callback)\n\r");
                }
        }

        if (pio_captureSr & PIO_PCISR_ENDRX) {
                /* End of Reception Transfer */
                if ( _PioCaptureCopy->CbkEndReception != NULL ) {
                        _PioCaptureCopy->CbkEndReception( _PioCaptureCopy );
                } else {
                        TRACE_DEBUG("IT PIO Capture End of Reception Transfer \
                                received (no callback)\n\r");
                }
        }
}

/*----------------------------------------------------------------------------*/
/**
 * \brief Disable Interrupt of the PIO Capture
 * \param itToDisable : Interrupt to disable
 */
/*----------------------------------------------------------------------------*/
void PIO_CaptureDisableIt( uint32_t itToDisable )
{
        /* Parallel capture mode is enabled */
```

```
        PIOA->PIO_PCIDR = itToDisable;
}

/*----------------------------------------------------------------------------*/
/**
 * \brief Enable Interrupt of the PIO Capture
 * \param itToEnable : Interrupt to enable
 */
/*----------------------------------------------------------------------------*/
void PIO_CaptureEnableIt( uint32_t itToEnable )
{
        /* Parallel capture mode is enabled */
        PIOA->PIO_PCIER = itToEnable;
}

/*----------------------------------------------------------------------------*/
/**
 * \brief Enable the PIO Capture
 */
/*----------------------------------------------------------------------------*/
void PIO_CaptureEnable( void )
{
        /* PDC: Receive Pointer Register */
        //    PIOA->PIO_RPR = (uint32_t)_PioCaptureCopy->pData ;
        //    /* PDC: Receive Counter Register */
        //    /* Starts peripheral data transfer if corresponding channel is active */
        //    PIOA->PIO_RCR = PIO_RCR_RXCTR(_PioCaptureCopy->dPDCsize) ;

        /* Parallel capture mode is enabled */
        PIOA->PIO_PCMR |= PIO_PCMR_PCEN ;
}

/*----------------------------------------------------------------------------*/
/**
 * \brief Disable the PIO Capture
 */
/*----------------------------------------------------------------------------*/
void PIO_CaptureDisable( void )
{
        /* Parallel capture mode is disabled */
        PIOA->PIO_PCMR &= (uint32_t)(~PIO_PCMR_PCEN) ;
}

/*----------------------------------------------------------------------------*/
/**
```

```
 * \brief Initialize the PIO Capture
 * \param dsize :
 *  0 = The reception data in the PIO_PCRHR register is a BYTE (8-bit).
 *  1 = The reception data in the PIO_PCRHR register is a HALF-WORD (16-bit).
 * 2/3 = The reception data in the PIO_PCRHR register is a WORD (32-bit).
 * \param alwaysSampling: ALWYS: Parallel Capture Mode Always Sampling
 * 0 = The parallel capture mode samples the data when both data enables are active.
 * 1 = The parallel capture mode samples the data whatever the data enables are.
 * \param halfSampling: HALFS: Parallel Capture Mode Half Sampling
 * 0 = The parallel capture mode samples all the data.
 * 1 = The parallel capture mode samples the data only one time out of two.
 * \param modeFirstSample: FRSTS: Parallel Capture Mode First Sample
 * This bit is useful only if the HALFS bit is set to 1. If data are numbered
 * in the order that they are received with an index from 0 to n:
 * 0 = Only data with an even index are sampled.
 * 1 = Only data with an odd index are sampled.
 */
/*----------------------------------------------------------------------------*/
void PIO_CaptureInit( SpioCaptureInit *pInit )
{
        PMC_EnablePeripheral( ID_PIOA );

        assert( (pInit->dsize < 0x4) ) ;
        assert( (pInit->alwaysSampling < 2)  );
        assert( (pInit->halfSampling < 2) );
        assert( (pInit->modeFirstSample < 2) );
        /* Copy the API structure for interrupt handler */
        _PioCaptureCopy = pInit;

        if ( pInit->CbkDataReady != NULL ) {
                PIOA->PIO_PCIER = PIO_PCISR_DRDY;
        }

        if ( pInit->CbkOverrun != NULL ) {
                PIOA->PIO_PCIER = PIO_PCISR_OVRE;
        }

        if ( pInit->CbkEndReception != NULL ) {
                PIOA->PIO_PCIER = PIO_PCISR_ENDRX;
        }

        if ( pInit->CbkBuffFull != NULL ) {
                PIOA->PIO_PCIER = PIO_PCISR_RXBUFF;
        }
}
```

pio _capture.h

```
/* ----------------------------------------------------------------------------
 *         SAM Software Package License
 * ----------------------------------------------------------------------------
 * Copyright (c) 2011, Atmel Corporation
 *
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * - Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the disclaimer below.
 *
 * Atmel's name may not be used to endorse or promote products derived from
 * this software without specific prior written permission.
 *
 * DISCLAIMER: THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY
EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-
INFRINGEMENT ARE
 * DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT,
INDIRECT,
 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA,
 * OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * ----------------------------------------------------------------------------
 */

#ifndef PIO_CAPTURE_H
#define PIO_CAPTURE_H

/*----------------------------------------------------------------------------
 *        Types
 *----------------------------------------------------------------------------*/
```

```
/** \brief PIO Parallel Capture structure for initialize.
 *
 * At the end of the transfer, the callback is invoked by the interrupt handler.
 */
typedef struct _SpioCaptureInit {

        /** PIO_PCRHR register is a BYTE, HALF-WORD or WORD */
        uint8_t dsize;
        /** PDC size, data to be received */
        uint16_t dPDCsize;
        /** Data to be received */
        uint32_t *pData;
        /** Parallel Capture Mode Always Sampling */
        uint8_t alwaysSampling;
        /** Parallel Capture Mode Half Sampling */
        uint8_t halfSampling;
        /** Parallel Capture Mode First Sample */
        uint8_t modeFirstSample;
        /** Callback function invoked at Mode Data Ready */
        void (*CbkDataReady)( struct _SpioCaptureInit* );
        /** Callback function invoked at Mode Overrun Error */
        void (*CbkOverrun)( struct _SpioCaptureInit* );
        /** Callback function invoked at End of Reception Transfer */
        void (*CbkEndReception)( struct _SpioCaptureInit* );
        /** Callback function invoked at Reception Buffer Full */
        void (*CbkBuffFull)( struct _SpioCaptureInit* );
        /** Callback arguments.*/
        void *pParam;

} SpioCaptureInit ;


/*----------------------------------------------------------------------------
 *       Global Functions
 *----------------------------------------------------------------------------*/
extern void PIO_CaptureDisableIt( uint32_t itToDisable ) ;
extern void PIO_CaptureEnableIt( uint32_t itToEnable ) ;
extern void PIO_CaptureEnable( void ) ;
extern void PIO_CaptureDisable( void ) ;
extern void PIO_CaptureInit( SpioCaptureInit* pInit ) ;

#endif /* #ifndef PIO_CAPTURE_H */
```

pio_it.c

```
/* ----------------------------------------------------------------------------
 *         SAM Software Package License
 * ----------------------------------------------------------------------------
 * Copyright (c) 2014, Atmel Corporation
 *
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * - Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the disclaimer below.
 *
 * Atmel's name may not be used to endorse or promote products derived from
 * this software without specific prior written permission.
 *
 * DISCLAIMER: THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY
EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-
INFRINGEMENT ARE
 * DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT,
INDIRECT,
 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA,
 * OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * ----------------------------------------------------------------------------
 */

/*
 * \file
 */

/*------------------------------------------------------------------------------
 *         Headers
 *------------------------------------------------------------------------------*/
```

```
#include "chip.h"

#include <assert.h>

/*----------------------------------------------------------------------------
 *        Local definitions
 *----------------------------------------------------------------------------*/

/* Maximum number of interrupt sources that can be defined. This
 * constant can be increased, but the current value is the smallest possible
 * that will be compatible with all existing projects. */
#define MAX_INTERRUPT_SOURCES       7

/*----------------------------------------------------------------------------
 *        Local types
 *----------------------------------------------------------------------------*/

/**
 * Describes a PIO interrupt source, including the PIO instance triggering the
 * interrupt and the associated interrupt handler.
 */
typedef struct _InterruptSource
{
        /* Pointer to the source pin instance. */
        const Pin *pPin;

        /* Interrupt handler. */
        void (*handler)( const Pin* ) ;
} InterruptSource ;

/*----------------------------------------------------------------------------
 *        Local variables
 *----------------------------------------------------------------------------*/

/* List of interrupt sources. */
static InterruptSource _aIntSources[MAX_INTERRUPT_SOURCES] ;

/* Number of currently defined interrupt sources. */
static uint32_t _dwNumSources = 0;

/*----------------------------------------------------------------------------
 *        Local Functions
 *----------------------------------------------------------------------------*/
```

```
/**
 * \brief Handles all interrupts on the given PIO controller.
 * \param id  PIO controller ID.
 * \param pPio  PIO controller base address.
 */
extern void PioInterruptHandler( uint32_t id, Pio *pPio )
{
	uint32_t status;
	uint32_t i;

	/* Read PIO controller status */
	status = pPio->PIO_ISR;
	status &= pPio->PIO_IMR;

	/* Check pending events */
	if ( status != 0 ) {
		TRACE_DEBUG( "PIO interrupt on PIO controller #%d\n\r", id ) ;

		/* Find triggering source */
		i = 0;
		while ( status != 0 ) {
			/* There cannot be an un-configured source enabled. */
			assert(i < _dwNumSources);

			/* Source is configured on the same controller */
			if (_aIntSources[i].pPin->id == id) {
				/* Source has PIOs whose statuses have changed */
				if ( (status & _aIntSources[i].pPin->mask) != 0 ) {
					TRACE_DEBUG( "Interrupt source #%d triggered\n\r", i ) ;
					_aIntSources[i].handler(_aIntSources[i].pPin);
					status &= ~(_aIntSources[i].pPin->mask);
				}
			}
			i++;
		}
	}
}

/*----------------------------------------------------------------------------
 *        Global Functions
 *----------------------------------------------------------------------------*/

/**
 * \brief Parallel IO Controller A interrupt handler
 * \Redefined PIOA interrupt handler for NVIC interrupt table.
```

```
 */
extern void PIOA_Handler_0( void )
{
        PioInterruptHandler( ID_PIOA, PIOA ) ;
}

/**
 * \brief Parallel IO Controller B interrupt handler
 * \Redefined PIOB interrupt handler for NVIC interrupt table.
 */
extern void PIOB_Handler( void )
{
        PioInterruptHandler( ID_PIOB, PIOB ) ;
}

/**
 * \brief Parallel IO Controller C interrupt handler
 * \Redefined PIOC interrupt handler for NVIC interrupt table.
 */
extern void PIOC_Handler( void )
{
        PioInterruptHandler( ID_PIOC, PIOC ) ;
}


/**
 * \brief Parallel IO Controller D interrupt handler
 * \Redefined PIOD interrupt handler for NVIC interrupt table.
 */
extern void PIOD_Handler( void )
{
        PioInterruptHandler( ID_PIOD, PIOD ) ;
}

/**
 * \brief Parallel IO Controller E interrupt handler
 * \Redefined PIOE interrupt handler for NVIC interrupt table.
 */
extern void PIOE_Handler( void )
{
        PioInterruptHandler( ID_PIOE, PIOE ) ;
}

/**
 * \brief Initializes the PIO interrupt management logic
```

```
 *
 * The desired priority of PIO interrupts must be provided.
 * Calling this function multiple times result in the reset of currently
 * configured interrupts.
 *
 * \param priority  PIO controller interrupts priority.
 */
extern void PIO_InitializeInterrupts( uint32_t dwPriority )
{
        TRACE_DEBUG( "PIO_Initialize()\n\r" ) ;

        /* Reset sources */
        _dwNumSources = 0 ;

        /* Configure PIO interrupt sources */
        TRACE_DEBUG( "PIO_Initialize: Configuring PIOA\n\r" ) ;
        PMC_EnablePeripheral( ID_PIOA ) ;
        PIOA->PIO_ISR ;
        PIOA->PIO_IDR = 0xFFFFFFFF ;
        NVIC_DisableIRQ( PIOA_IRQn ) ;
        NVIC_ClearPendingIRQ( PIOA_IRQn ) ;
        NVIC_SetPriority( PIOA_IRQn, dwPriority ) ;
        NVIC_EnableIRQ( PIOA_IRQn ) ;

        TRACE_DEBUG( "PIO_Initialize: Configuring PIOB\n\r" ) ;
        PMC_EnablePeripheral( ID_PIOB ) ;
        PIOB->PIO_ISR ;
        PIOB->PIO_IDR = 0xFFFFFFFF ;
        NVIC_DisableIRQ( PIOB_IRQn ) ;
        NVIC_ClearPendingIRQ( PIOB_IRQn ) ;
        NVIC_SetPriority( PIOB_IRQn, dwPriority ) ;
        NVIC_EnableIRQ( PIOB_IRQn ) ;

        TRACE_DEBUG( "PIO_Initialize: Configuring PIOC\n\r" ) ;
        PMC_EnablePeripheral( ID_PIOC ) ;
        PIOC->PIO_ISR ;
        PIOC->PIO_IDR = 0xFFFFFFFF ;
        NVIC_DisableIRQ( PIOC_IRQn ) ;
        NVIC_ClearPendingIRQ( PIOC_IRQn ) ;
        NVIC_SetPriority( PIOC_IRQn, dwPriority ) ;
        NVIC_EnableIRQ( PIOC_IRQn ) ;

        TRACE_DEBUG( "PIO_Initialize: Configuring PIOD\n\r" ) ;
        PMC_EnablePeripheral( ID_PIOD ) ;
        PIOD->PIO_ISR ;
```

```
        PIOD->PIO_IDR = 0xFFFFFFFF ;
        NVIC_DisableIRQ( PIOD_IRQn ) ;
        NVIC_ClearPendingIRQ( PIOD_IRQn ) ;
        NVIC_SetPriority( PIOD_IRQn, dwPriority ) ;
        NVIC_EnableIRQ( PIOD_IRQn ) ;

        TRACE_DEBUG( "PIO_Initialize: Configuring PIOE\n\r" ) ;
        PMC_EnablePeripheral( ID_PIOE ) ;
        PIOE->PIO_ISR ;
        PIOE->PIO_IDR = 0xFFFFFFFF ;
        NVIC_DisableIRQ( PIOE_IRQn ) ;
        NVIC_ClearPendingIRQ( PIOE_IRQn ) ;
        NVIC_SetPriority( PIOE_IRQn, dwPriority ) ;
        NVIC_EnableIRQ( PIOE_IRQn ) ;
}

/**
 * Configures a PIO or a group of PIO to generate an interrupt on status
 * change. The provided interrupt handler will be called with the triggering
 * pin as its parameter (enabling different pin instances to share the same
 * handler).
 * \param pPin  Pointer to a Pin instance.
 * \param handler  Interrupt handler function pointer.
 */
extern void PIO_ConfigureIt( const Pin *pPin, void (*handler)( const Pin* ) )
{
        Pio* pio ;
        InterruptSource* pSource ;

        TRACE_DEBUG( "PIO_ConfigureIt()\n\r" ) ;

        assert( pPin ) ;
        pio = pPin->pio ;
        assert( _dwNumSources < MAX_INTERRUPT_SOURCES ) ;

        /* Define new source */
        TRACE_DEBUG( "PIO_ConfigureIt: Defining new source #%d.\n\r", _dwNumSources )
;

        pSource = &(_aIntSources[_dwNumSources]) ;
        pSource->pPin = pPin ;
        pSource->handler = handler ;
        _dwNumSources++ ;

        /* PIO3 with additional interrupt support
```

```
         * Configure additional interrupt mode registers */
        if ( pPin->attribute & PIO_IT_AIME ) {
                // enable additional interrupt mode
                pio->PIO_AIMER  = pPin->mask ;

                // if bit field of selected pin is 1, set as Rising Edge/High level detection event
                if ( pPin->attribute & PIO_IT_RE_OR_HL ) {
                        pio->PIO_REHLSR    = pPin->mask ;
                } else {
                        pio->PIO_FELLSR     = pPin->mask;
                }

                /* if bit field of selected pin is 1, set as edge detection source */
                if (pPin->attribute & PIO_IT_EDGE)
                        pio->PIO_ESR     = pPin->mask;
                else
                        pio->PIO_LSR     = pPin->mask;
        } else {
                /* disable additional interrupt mode */
                pio->PIO_AIMDR       = pPin->mask;
        }
}

/**
 * Enables the given interrupt source if it has been configured. The status
 * register of the corresponding PIO controller is cleared prior to enabling
 * the interrupt.
 * \param pPin  Interrupt source to enable.
 */
extern void PIO_EnableIt( const Pin *pPin )
{
                uint32_t i = 0;
        uint32_t dwFound = 0;

        TRACE_DEBUG( "PIO_EnableIt()\n\r" ) ;

        assert( pPin != NULL ) ;

#ifndef NOASSERT

        while ( (i < _dwNumSources) && !dwFound ) {
                if ( _aIntSources[i].pPin == pPin ) {
                        dwFound = 1 ;
                }
                i++ ;
```

```
        }
        assert( dwFound != 0 ) ;
#endif

        pPin->pio->PIO_ISR;
        pPin->pio->PIO_IER = pPin->mask ;
}

/**
 * Disables a given interrupt source, with no added side effects.
 *
 * \param pPin  Interrupt source to disable.
 */
extern void PIO_DisableIt( const Pin *pPin )
{
        assert( pPin != NULL ) ;

        TRACE_DEBUG( "PIO_DisableIt()\n\r" ) ;

        pPin->pio->PIO_IDR = pPin->mask;
}
```

pio_it.h

```
/* ----------------------------------------------------------------------------
 *         SAM Software Package License
 * ----------------------------------------------------------------------------
 * Copyright (c) 2011, Atmel Corporation
 *
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * - Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the disclaimer below.
 *
 * Atmel's name may not be used to endorse or promote products derived from
 * this software without specific prior written permission.
 *
 * DISCLAIMER: THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY
EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-
INFRINGEMENT ARE
 * DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT,
INDIRECT,
 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA,
 * OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * ----------------------------------------------------------------------------
 */

/**
 *  \file
 *
 *  \par Purpose
 *
 *  Configuration and handling of interrupts on PIO status changes. The API
 *  provided here have several advantages over the traditional PIO interrupt
```

```
 *  configuration approach:
 *    - It is highly portable
 *    - It automatically demultiplexes interrupts when multiples pins have been
 *      configured on a single PIO controller
 *    - It allows a group of pins to share the same interrupt
 *
 *  However, it also has several minor drawbacks that may prevent from using it
 *  in particular applications:
 *    - It enables the clocks of all PIO controllers
 *    - PIO controllers all share the same interrupt handler, which does the
 *      demultiplexing and can be slower than direct configuration
 *    - It reserves space for a fixed number of interrupts, which can be
 *      increased by modifying the appropriate constant in pio_it.c.
 *
 *  \par Usage
 *
 *  -# Initialize the PIO interrupt mechanism using PIO_InitializeInterrupts()
 *    with the desired priority (0 ... 7).
 *  -# Configure a status change interrupt on one or more pin(s) with
 *    PIO_ConfigureIt().
 *  -# Enable & disable interrupts on pins using PIO_EnableIt() and
 *    PIO_DisableIt().
 */

#ifndef _PIO_IT_
#define _PIO_IT_

/*
 *        Headers
 */

#include "pio.h"

#ifdef __cplusplus
 extern "C" {
#endif

/*
 *        Global functions
 */

extern void PIO_InitializeInterrupts( uint32_t dwPriority ) ;

extern void PIO_ConfigureIt( const Pin *pPin, void (*handler)( const Pin* ) ) ;
```

```
extern void PIO_EnableIt( const Pin *pPin ) ;

extern void PIO_DisableIt( const Pin *pPin ) ;

extern void PIO_IT_InterruptHandler( void ) ;

extern void PioInterruptHandler( uint32_t id, Pio *pPio ) ;

extern void PIO_CaptureHandler( void ) ;

#ifdef __cplusplus
}
#endif

#endif /* #ifndef _PIO_IT_ */
```

CanNm.c

```
#include "CanNm.h"

/*
  CAN Network manager
  1) Additional function to execute the CAN messages
*/

uint8_t CAN_MSG_Counter = 0;

void CanNm_Tx(void)
{
  if(CAN_MSG_Counter >= CAN_WAIT)
  {
    CAN_MSG_Counter = 0;
    CanIf_Transmit(ControllerId_1 ,CAN_MSGID_0);              /* Send CAN tx message */
  }
  else
  {
    CAN_MSG_Counter++;
  }
}

uint8_t Can_SetSignal(uint8_t Signal, uint8_t SignalValue)
{
  CanMsgObj.CanIfMessageConfig[0].CanPdu.CanSdu[Signal] = SignalValue;  /* Update SDU
*/
  return 0; /* OK = 0 */
}
```

CanNm.h

```
#ifndef _CAN_NTW_
#define _CAN_NTW_

#include "CanIf.h"

/* How many times shall the CanNm_Tx be called to send the CAN MSG */
#define CAN_WAIT    9

extern void CanNm_Tx(void);
extern uint8_t Can_SetSignal(uint8_t Signal, uint8_t SignalValue);

#endif /* _CAN_NTW_ */
```

app_scheduler.c

```
/*****************************************************************************
**/
/**
\file       app_scheduler.c
\brief      Multi-thread Task scheduler.
\author     Abraham Tezmol
\version    0.1
\date       09/09/2008
\author     Francisco Matinez
\version    0.2
\date       06/21/2016
\author     Francisco Matinez
\version    0.2
\date       09/04/2017
*/

/** Variable types and common definitions */

/** Scheduler function prototypes definitions */
#include "app_scheduler.h"
#include "chip.h"

typedef struct
{
  uint8_t runTask;
  uint8_t taskOverload;
  uint8_t tickValue;
  TaskIdType taskId;
  FuncPtr tskFcnPtr;
}TaskCtrlType;

/* -- Global Variables ------------------------------------------------------*/
uint8_t gu8Scheduler_Status;
uint8_t gu8Scheduler_Counter;
uint8_t u8_10ms_Counter;
uint8_t u8_50ms_Counter;
uint8_t u8_100ms_Counter;
TaskIdType gu8Scheduler_Thread_ID;
TaskIdType gu8Scheduler_Thread_ID_Backup;

TaskCtrlType task_ctrl_array[TASK_MAXNUM];

/*****************************************************************************
**/
```

```
/**
* \brief    Periodic Interrupt Timer Service routine.                          \n
          This interrupt is the core of the task scheduler.                  \n
          It is executed every 500us                                         \n
          It defines 3 basic threads from which other 3 threads are derived:  \n
          a) 1ms thread (basic) ->  100ms thread (derived)                   \n
          b) 2ms A thread (basic)-> 50ms thread (derived)                    \n
          c) 2ms B thread (basic)-> 10ms thread (derived)                    \n
          It partitions core execution time into time slices (500us each one).\n
          This arrangement assures core will have equal task loading across time.\n
          For more information on how time slice is assigned to each thread,  \n
          refer to file "S12X Task Scheduler Layout.xls"
* \author   Abraham Tezmol
* \param    void
* \return   void
* \todo
*/

void vfnScheduler_Callback(void)
{

          /*-- Update scheduler control variable --*/
          gu8Scheduler_Counter++;

          /*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~*/
          /*  1ms execution thread - used to derive two execution threads:          */
          /*  a) 1ms thread (highest priority tasks)                                */
          /*  b) 100ms thread (lowest priority tasks)                               */
          /*  As any other thread on this scheduling scheme,                        */
          /*  all tasks must be executed in <= 500us                                */
          /*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~*/
          if ((gu8Scheduler_Counter & 0x01) == 0x01)
          {
                    u8_100ms_Counter++;
                    /*-- Allow 100 ms periodic tasks to be executed --*/
                    if (u8_100ms_Counter >= 100)
                    {
                              gu8Scheduler_Thread_ID = TASK_100MS;
                              u8_100ms_Counter = 0;
                              task_ctrl_array[(uint8_t)TASK_100MS].runTask=1;
                              task_ctrl_array[(uint8_t)TASK_100MS].tickValue =
gu8Scheduler_Counter;
                    }
```

```
            /*-- Allow 1 ms periodic tasks to be executed --*/
            else
            {
                    gu8Scheduler_Thread_ID = TASK_1MS;
            }
            task_ctrl_array[(uint8_t)TASK_1MS].runTask=1;
            task_ctrl_array[(uint8_t)TASK_1MS].tickValue = gu8Scheduler_Counter;
        }
        else
        {

        /*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~*/
            /*  2ms execution thread - used to derive two execution threads:          */
            /*  a) 2ms group A thread (high priority tasks)                          */
            /*  b) 50ms thread (second lowest priority tasks)                        */
            /*  As any other thread on this scheduling scheme,                       */
            /*  all tasks must be executed in <= 500us                               */

        /*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~*/
            if ((gu8Scheduler_Counter & 0x02) == 0x02)
            {
                    u8_50ms_Counter++;
                    /*-- Allow 50 ms periodic tasks to be executed --*/
                    if (u8_50ms_Counter >= 25)
                    {
                            gu8Scheduler_Thread_ID = TASK_50MS;
                            u8_50ms_Counter = 0;
                            task_ctrl_array[(uint8_t)TASK_50MS].runTask=1;
                            task_ctrl_array[(uint8_t)TASK_50MS].tickValue =
gu8Scheduler_Counter;
                    }
                    /*-- Allow 2 ms group A periodic tasks to be executed --*/
                    else
                    {
                            gu8Scheduler_Thread_ID = TASK_2MSA;
                    }
                    task_ctrl_array[(uint8_t)TASK_2MSA].runTask=1;
                    task_ctrl_array[(uint8_t)TASK_2MSA].tickValue =
gu8Scheduler_Counter;
            }
            else
            {
```

```
	/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~*/
			/*  2ms execution thread - used to derive two execution threads:
*/
			/*  a) 2ms group B thread (high priority tasks)			*/
			/*  b) 10ms thread (medium priority tasks)			*/
			/*  As any other thread on this scheduling scheme,			*/
			/*  all tasks must be executed in <= 500us			*/

	/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~*/
			if ((gu8Scheduler_Counter & 0x03) == 0x00)
			{
				u8_10ms_Counter++;
				/*-- Allow 10 ms periodic tasks to be executed --*/
				if (u8_10ms_Counter >= 5)
				{
					gu8Scheduler_Thread_ID = TASK_10MS;
					u8_10ms_Counter = 0;
					task_ctrl_array[(uint8_t)TASK_10MS].runTask=1;
					task_ctrl_array[(uint8_t)TASK_10MS].tickValue =
gu8Scheduler_Counter;
				}
				/*-- Allow 2 ms group B periodic tasks to be executed --*/
				else
				{
					gu8Scheduler_Thread_ID = TASK_2MSB;
				}
				task_ctrl_array[(uint8_t)TASK_2MSB].runTask=1;
				task_ctrl_array[(uint8_t)TASK_2MSB].tickValue =
gu8Scheduler_Counter;
			}
		}
	}
}

/****************************************************************************
**/
/**
* \brief    Scheduler - Periodic Interrup Timer Initialization
* \author   Abraham Tezmol
* \param    void
* \return   void
* \todo
```

```
*/
void vfnScheduler_Init(TaskType *TaskArray)
{
 TaskIdType task_idx;
 /* Init Global and local Task Scheduler variables */
 gu8Scheduler_Counter = 0;
 u8_10ms_Counter = 0;
 u8_50ms_Counter = 0;
 u8_100ms_Counter = 0;
 gu8Scheduler_Status = TASK_SCHEDULER_INIT;
        for (task_idx = 0; task_idx < (uint8_t)TASK_MAXNUM; task_idx++)
        {
                task_ctrl_array[task_idx].tskFcnPtr = TaskArray[task_idx].tskFcnPtr;
                task_ctrl_array[task_idx].taskId = TaskArray[task_idx].taskId;
        }
}

/*****************************************************************************
**/
/**
* \brief    Scheduler Start - Once time base is armed, start execution of   \n
        Multi-thread Round Robin scheduling scheme.                  \n
        This function requires prior execution of "vfnScheduler_Init"
* \author   Abraham Tezmol
* \param    void
* \return   void
* \todo
*/
void vfnScheduler_Start(void)
{
   TimeTick_Configure(vfnScheduler_Callback);
   gu8Scheduler_Status = TASK_SCHEDULER_RUNNING;
}

/*****************************************************************************
**/
/**
* \brief    Scheduler Stop - stop execution of Multi-thread Round Robin scheduling scheme.
* \author   Abraham Tezmol
* \param    void
* \return   void
* \todo
*/
void vfnScheduler_Stop(void)
{
```

```
   /* Update scheduler status accordingly */
   gu8Scheduler_Status = TASK_SCHEDULER_HALTED;
}

/*****************************************************************************
**/
/**
* \brief    Multi-thread round robin task Scheduler  (non-preemtive)       \n
        It calls the different tasks based on the status of           \n
        "gu8Scheduler_Thread_ID". This variable is modified by        \n
        ISR "vfnScheduler_PIT_Isr"                                    \n
        List of tasks shall be defined @ "tasks.h" file
* \author   Abraham Tezmol
* \param    void
* \return   void
* \todo
*/
void vfnTask_Scheduler(void)
{
        TaskIdType task_idx;

        for (task_idx = 0;task_idx < (uint8_t)TASK_MAXNUM; task_idx++)
        {
                if( 1 == task_ctrl_array[task_idx].runTask )
                {
                        task_ctrl_array[task_idx].runTask = 0;
                        if ( NULL != task_ctrl_array[task_idx].tskFcnPtr )
                        {
                                task_ctrl_array[task_idx].tskFcnPtr();
                        }
                        if ( gu8Scheduler_Counter != task_ctrl_array[task_idx].tickValue )
                        {
                                task_ctrl_array[task_idx].taskOverload = 1;
                                gu8Scheduler_Status = TASK_SCHEDULER_OVERLOAD;
                        }
                        else{
                                gu8Scheduler_Status = TASK_SCHEDULER_RUNNING;
                        }
                }
        }

}
```

app_scheduler.h

```
/*****************************************************************************
**/
/**
\file       app_scheduler.h
\brief      Task scheduler function prototypes
\author     Abraham Tezmol
\version    0.1
\date       09/09/2008
\author     Francisco Matinez
\version    0.2
\date       06/21/2016
\author     Francisco Matinez
\version    0.2
\date       09/04/2017
*/

#ifndef APP_SCHEDULER_H        /*prevent duplicated includes*/
#define APP_SCHEDULER_H

/*-- Includes ------------------------------------------------------------*/
#include <stdint.h>
/*-- Types Definitions ---------------------------------------------------*/
typedef void (*FuncPtr)(void);
typedef uint8_t TaskIdType;

typedef struct
{
  uint8_t taskPriority;
  TaskIdType taskId;
  FuncPtr tskFcnPtr;
}TaskType;

/*-- Defines -------------------------------------------------------------*/


/* Global Task Scheduler Status definitions */

#define  TASK_1MS     0x00u
#define  TASK_2MSA    0x01u
#define  TASK_2MSB    0x02u
#define  TASK_10MS    0x03u
#define  TASK_50MS    0x04u
#define  TASK_100MS   0x05u
#define  TASK_MAXNUM  0x06u
```

```
#define TASK_SCHEDULER_INIT           0x00u
#define TASK_SCHEDULER_RUNNING        0x01u
#define TASK_SCHEDULER_OVERLOAD_1MS   0x02u
#define TASK_SCHEDULER_OVERLOAD_2MS_A 0x03u
#define TASK_SCHEDULER_OVERLOAD_2MS_B 0x04u
#define TASK_SCHEDULER_HALTED         0xAAu

#define TASK_SCHEDULER_OVERLOAD                    (0x55u)

/*-- Macros ------------------------------------------------------------*/


/*-- Function Prototypes -----------------------------------------------*/

/** Sheduler Initalization (arming) */
void vfnScheduler_Init(TaskType *TaskArray);

/** Scheduler kick-off function */
void vfnScheduler_Start(void);

/** Scheduler stop function */
void vfnScheduler_Stop(void);

/** Multi-thread round robin task scheduler */
void vfnTask_Scheduler(void);

/*****************************************************************************
**/

#endif /* APP_SCHEDULER_H */
```

Tasks.c

```
/*
 * Tasks.c
 *
 * Created: 6/21/2016 7:25:43 PM
 *  Author: Francisco Martinez
 */

#include "Tasks.h"
#include "led.h"
#include "CanNm.h"

uint8_t u8100ms_Ctr=0;
uint8_t u8100ms_Ctr2=0;

void vfnTsk_Init(void)
{

}


void vfnTsk_1ms(void)
{

}

void vfnTsk_2msA(void)
{

}

void vfnTsk_2msB(void)
{

}

void vfnTsk_10ms(void)
{
	static uint8_t u8500ms_Ctr=0;

	u8500ms_Ctr++;

	if (25 <= u8500ms_Ctr)
	{
		u8500ms_Ctr = 0;
```

```
                LED_Toggle( 1 );
        }
}

void vfnTsk_50ms(void)
{
        Process_Sensing_Env();
}

void vfnTsk_100ms(void)
{
        CanNm_Tx();
        u8100ms_Ctr++;
   u8100ms_Ctr2++;

        if (5 <= u8100ms_Ctr)
        {
                u8100ms_Ctr = 0;
                LED_Toggle( 0 );
        }
   if (10 <= u8100ms_Ctr2)
        {
                u8100ms_Ctr2 = 0;
        }
}
```

Tasks.h

```
/*****************************************************************************
**/
/**
\file       Tasks.h
\author     Francisco Matinez
\version    0.2
\date       06/21/2016
*/

#ifndef __APP_TASKS
#define __APP_TASKS

/*-- Includes ------------------------------------------------------------*/

/*-- Variables -----------------------------------------------------------*/

/*-- Defines -------------------------------------------------------------*/

/*-- Macros --------------------------------------------------------------*/

extern void vfnTsk_Init(void);
extern void vfnTsk_1ms(void);
extern void vfnTsk_2msA(void);
extern void vfnTsk_2msB(void);
extern void vfnTsk_10ms(void);
extern void vfnTsk_50ms(void);
extern void vfnTsk_100ms(void);


/*=============================================================================
=========*/
#endif /*__APP_TASKS */
```