

Timo Nikkanen

Controller for a pressure and humidity instrument in Martian atmosphere

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 4.12.2013

Thesis supervisor:

Prof. Martti Hallikainen

Thesis instructor:

Ph.D. Walter Schmidt



Aalto University
School of Electrical
Engineering

Author: Timo Nikkanen

Title: Controller for a pressure and humidity instrument in Martian atmosphere

Date: 4.12.2013

Language: English

Number of pages:9+87

Department of Radio Science and Technology

Professorship: Space Technology

Code: S-92

Supervisor: Prof. Martti Hallikainen

Instructor: Ph.D. Walter Schmidt

Finnish Meteorological Institute (FMI) is building a pressure and a humidity instrument for the ExoMars EDM Mars lander of European Space Agency, slated for launch in 2016. The instruments, part of a science package called DREAMS, build on the experience gained on previous meteorological Mars instrument missions participated by FMI. The instruments are based on capacitive Vaisala Barocap® and Humicap® sensor technologies.

Traditionally, the FMI pressure and humidity instruments have been controlled by Field-Programmable Gate Array (FPGA) chips. These components are reliable, but relatively expensive, power-hungry, slow to develop and lack the ability for an in-flight update.

This thesis presents a novel approach for controlling these instruments, by developing flight software for performing measurements with the DREAMS-P pressure and DREAMS-H humidity instruments. This design utilizes a commercial automotive microcontroller unit, resulting in a low-power, modifiable and scalable system. The successful design and testing of the instrument controller software and functionality is presented.

Keywords: Mars, pressure, humidity, space instrumentation, microcontroller, embedded software, software testing

Tekijä: Timo Nikkanen

Työn nimi: Kontrolleri paine- ja kosteusinstrumentille Marsin kaasukehään

Päivämäärä: 4.12.2013

Kieli: Englanti

Sivumäärä: 9+87

Radiotieteen ja -tekniikan laitos

Professuuri: Avaruustekniikka

Koodi: S-92

Valvoja: Prof. Martti Hallikainen

Ohjaaja: FT Walter Schmidt

Ilmatieteen laitos rakentaa DREAMS-P paine- ja DREAMS-H kosteusinstrumenttia osana Euroopan avaruusjärjestön 2016 laukaistavaa ExoMars EDM Mars-laskeutujaprojektia. Ilmatieteen laitoksen aiempiin Mars-mittalaiteprojekteihin ja Vaisalan Barocap® sekä Humicap® anturitekniologioihin pohjautuvat instrumentit ovat osa EDM-laskeutujan DREAMS-instrumenttipakettia.

Ilmatieteen laitoksen paine- ja kosteusinstrumentit ovat perinteisesti pohjautuneet FPGA-piireihin (Field-Programmable Gate Array). Nämä piirit ovat luotettavia, mutta verrattain kalliita ja omaavat suurehkon tehonkulutuksen. Ohjelmistokehitys FPGA-piireillä on myös verrattain hidasta, eivätkä ne pysty itsenäisiin avaruuslennon aikaisiin päivityksiin.

Tämä diplomityö esittelee uudenlaisen ohjelmistoratkaisun DREAMS-P paine- ja DREAMS-H kosteusinstrumenttien ohjaamiseen. Järjestelmä perustuu kaupalliseen, autoteollisuuden käyttöön suunniteltuun mikrokontrolleriin. Mikrokontrollerin käyttö mahdollistaa vähän tehoa kuluttavan, sekä helposti muokattavan ja laajennettavan järjestelmän rakentamisen. Työssä esitellään tämän DREAMS-P ja DREAMS-H -instrumentteja ohjaavan kontrollerin ohjelmiston suunnittelu ja testaus.

Avainsanat: Mars, paine, kosteus, mikrokontrolleri, avaruuslaitetekniikka, su-lautettu järjestelmä, ohjelmistotestaus

Preface

This thesis was written at the Finnish Meteorological Institute Radar and Space Technology group. The group is a part of the FMI Earth Observation program. Work leading to the thesis was started in fall 2012. The master's thesis is a part of a project building a pressure and humidity instrument for the ExoMars 2016 project of the European Space Agency.

I would like to thank my instructor Dr. Walter Schmidt for his unwavering and active support for my thesis, and the whole FMI DREAMS team for the opportunity to work in an amazingly interesting project. Also, I would like to thank Prof. Martti Hallikainen for supervising the thesis.

Otaniemi, 4.12.2013

Timo Nikkanen

Contents

| | |
|---|-------------|
| Abstract | ii |
| Abstract (in Finnish) | iii |
| Preface | iv |
| Contents | v |
| Symbols and abbreviations | viii |
| 1 Introduction | 1 |
| 2 Background | 2 |
| 2.1 In-situ study of the Martian atmosphere | 2 |
| 2.1.1 Past and current studies | 2 |
| 2.1.2 Involvement of Finnish Meteorological Institute | 3 |
| 2.2 ExoMars mission | 3 |
| 2.2.1 General | 3 |
| 2.2.2 EDM lander | 4 |
| 2.2.3 DREAMS science package | 5 |
| 3 System Design of DREAMS-P/H instrument | 7 |
| 3.1 General description | 7 |
| 3.2 DREAMS-P pressure instrument | 7 |
| 3.3 DREAMS-H humidity instrument | 10 |
| 3.4 Transducer control | 11 |
| 3.5 Interfaces | 11 |
| 3.5.1 Electrical interfaces | 11 |
| 3.5.2 Telemetry and telecommand data interface | 14 |
| 3.5.3 Debugger interface | 14 |
| 3.6 System requirements | 15 |
| 4 DREAMS-P/H instrument controller unit | 17 |
| 4.1 Controller overview | 17 |
| 4.2 Microcontroller | 18 |
| 4.2.1 Overview | 18 |
| 4.2.2 Peripheral modules | 19 |
| 4.2.3 Space qualification | 20 |
| 4.3 Supporting components | 21 |
| 4.3.1 DREAMS-H heater power source | 21 |
| 4.3.2 Crystal oscillator | 22 |
| 4.4 Controller operational requirements | 23 |
| 4.4.1 General | 23 |
| 4.4.2 Instrument control | 25 |
| 4.4.3 Heater control | 27 |

| | | |
|----------|---|-----------|
| 4.4.4 | Pressure and humidity measurements | 27 |
| 4.4.5 | Configuration tables | 28 |
| 4.5 | Development tools | 29 |
| 4.5.1 | Evaluation board | 29 |
| 4.5.2 | Freescale CodeWarrior IDE software | 31 |
| 4.5.3 | DREAMS EGSE software | 33 |
| 5 | Controller operational software | 34 |
| 5.1 | Software system description | 34 |
| 5.2 | Operative principles | 34 |
| 5.2.1 | Non-volatile memory management | 34 |
| 5.2.2 | Error correction plan | 37 |
| 5.2.3 | Watchdog | 38 |
| 5.2.4 | Software consistency check and redundancy | 38 |
| 5.2.5 | In-flight updates | 39 |
| 5.3 | Software modules | 39 |
| 5.3.1 | Main program | 39 |
| 5.3.2 | Telecommand and telemetry | 41 |
| 5.3.3 | Instrument control | 43 |
| 5.3.4 | Heater control | 44 |
| 5.3.5 | Pulse counting | 45 |
| 5.3.6 | Memory management and manipulation | 46 |
| 6 | Test procedures and test results | 49 |
| 6.1 | Test philosophy | 49 |
| 6.2 | Verification test setup | 50 |
| 6.3 | Verification test procedure | 52 |
| 6.3.1 | Controller start-up | 52 |
| 6.3.2 | Power commands | 52 |
| 6.3.3 | DREAMS-H regeneration | 53 |
| 6.3.4 | Single test measurement commands | 53 |
| 6.3.5 | Configuration table handling | 54 |
| 6.3.6 | Automatic measurement commands | 55 |
| 6.3.7 | Memory control commands | 56 |
| 6.3.8 | Telecommand and telemetry handling | 56 |
| 6.3.9 | Power consumption | 57 |
| 6.4 | Verification test results | 57 |
| 6.4.1 | Controller start-up | 57 |
| 6.4.2 | Power commands | 58 |
| 6.4.3 | DREAMS-H regeneration | 58 |
| 6.4.4 | Single test measurement commands | 59 |
| 6.4.5 | Configuration table handling | 59 |
| 6.4.6 | Automatic measurement commands | 59 |
| 6.4.7 | Memory control commands | 60 |
| 6.4.8 | Telecommand and telemetry handling | 60 |

| | | |
|----------|--|-----------|
| 6.4.9 | Power consumption | 61 |
| 6.4.10 | Analysis | 62 |
| 7 | Conclusions | 63 |
| | References | 65 |
| | Annex A | 67 |
| A | Pulse counting software implementation | 67 |
| | Annex B | 69 |
| B | Software verification test stepwise procedure | 69 |

Symbols and abbreviations

Abbreviations

| | |
|---------|---|
| AC | Alternating Current |
| AFT | Abbreviated Functional Test |
| ASIC | Application Specific Integrated Circuit |
| BDM | Background Debug Mode |
| CAD | Computer Aided Design |
| CAN | Controller Area Network |
| CEU | Common Electronics Unit |
| CISC | Complex Instruction Set Computer |
| COP | Computer Operating Properly |
| COTS | Commercial Off-The-Shelf |
| CRG | Clocks and Reset Generator |
| CVCM | Collected Volatile Condensable Materials |
| DC | Direct Current |
| D-Flash | Data Flash |
| DPA | Destructive Physical Analysis |
| DPU | Data Processing Unit |
| DREAMS | Dust characterization, Risk assessment, and Environment Analyzer on the Martian Surface |
| ECC | Error Correction Codes |
| ECT | Enhanced Capture Timer |
| EDM | Entry, descent, and landing Demonstrator Module |
| EGSE | Electronic Ground Support Equipment |
| EMC | ElectroMagnetic Compatibility |
| EMI | ElectroMagnetic Interference |
| ESA | European Space Agency |
| ExoMars | Exobiology on Mars |
| FFT | Full Functional Test |
| FMI | Finnish Meteorological Institute |
| FPGA | Field-Programmable Gate Array |
| IDE | Integrated Design Environment |
| INTA | Instituto Nacional de Técnica Aeroespacial |
| ISR | Interrupt Service Routine |
| ISRO | Indian Space Research Organization |
| LED | Light Emitting Diode |
| LIN | Local Interconnect Network |
| lsb | least significant bit |
| LQFP | Low profile Quad Flat Package |
| MAPBGA | Molded Array Process Ball Grid Array |
| MAVEN | Mars Atmosphere and Volatile EvolutionN |
| MCU | MicroController Unit |
| MMC | Memory Mapping Control |

| | |
|---------|---|
| MOM | Mars Orbiter Mission |
| MSL | Mars Science Laboratory |
| NASA | National Aeronautics and Space Administration |
| OBC | On-Board Computer |
| PC | Personal Computer |
| PCB | Printed Circuit Board |
| PE | Proximity Electronics |
| P-Flash | Program Flash |
| PIT | Periodic Interrupt Timer |
| PLL | Phase Locked Loop |
| QM | Qualification Model |
| RAM | Random Access Memory |
| REMS | Rover Environmental Monitoring Station |
| RISC | Reduced Instruction Set Computer |
| RML | Recovered Mass Loss |
| RTD | Resistance Temperature Detector |
| SAM | Scanning Acoustic Microscopy |
| SCI | Serial Communication Interface |
| SIS | Solar Intensity Sensor |
| SOHO | SOlar and Heliospheric Observatory |
| TCL | Tool Command Language |
| TGO | Trace Gas Orbiter |
| USB | Universal Serial Bus |

1 Introduction

The *Finnish Meteorological Institute* (FMI) has an extensive history in providing miniature high-performance pressure instruments for planetary lander missions. As a part of this ongoing research and development effort, FMI has expanded also to miniature humidity instruments, first of which landed on Mars aboard the *Curiosity* rover of NASA's *Mars Science Laboratory* (MSL) mission in August 2012.

Now, the Radar and Space Technology group of FMI is participating in the *ExoMars* project led by the *European Space Agency* (ESA). As a part of the *ExoMars* project, a spacecraft composed of two components, the *Trace Gas Orbiter* (TGO) and the *Entry, Descent and Landing Demonstrator Module* (EDM), will be launched towards Mars in 2016. If successful, the EDM, also known as *Schiaparelli*, will also deliver a limited science package called the *Dust characterization, Risk assessment, and Environment Analyzer on the Martian Surface* (DREAMS). FMI will design and build two instruments as a part of the DREAMS payload suite: a pressure instrument called *DREAMS-P* and a humidity instrument called *DREAMS-H*. The combination of these instruments shall be referred to as *DREAMS-P/H* in this thesis. [1]

Previous pressure and humidity instruments of FMI have utilized *Field-Programmable Gate Array* (FPGA) based controller units. The main purposes of such unit are to extract scientific data from the sensors of the instrument by converting frequencies into digital numbers, and the command and data interfacing between the instrument and the instrument package *Data Processing Unit* (DPU).

This thesis presents the design and testing of the DREAMS-P/H controller unit. A novel approach to controlling these instruments is presented by utilizing a microcontroller. The design is based on a commercial automotive *Microcontroller Unit* (MCU), which has passed a space-qualification program conducted by FMI. Lower power consumption, better modifiability, shorter design time and lower cost are expected by using the versatile Freescale MC9S12XEP100 microcontroller, compared to the old FPGA designs.

The core of this study will concentrate on the design, implementation and testing of the prototype flight software for the aforementioned microcontroller. The electronic hardware or the science done with the instruments will not be discussed in detail. First, the paper shortly introduces the field of Martian atmospheric study and the Finnish contribution related to this field in the past and presently, as well as the high-level design of the *ExoMars* 2016 EDM mission and the DREAMS instrument suite. Secondly, the system design of DREAMS-P/H is described, focusing on the interfaces relevant for the controller unit. Then, the actual controller design is explained. After that, this thesis presents the operational software for the *Qualification Model* (QM) of DREAMS-P and DREAMS-H. Lastly, the software qualification test procedures for the space flight software are described and the test results are assessed.

2 Background

2.1 In-situ study of the Martian atmosphere

2.1.1 Past and current studies

Mars has been a priority target for space probes since the dawn of the space age. Only slightly more energy is needed to escape the Earth-Moon system compared to getting from the Earth to the Moon [2]. This means that same propulsion technologies can be used for reaching the lunar orbit or the Martian orbit. Nevertheless, landing a probe on Mars requires a lot more from other systems (guidance and navigation system, re-entry system etc.) and above all, great deal of autonomy on the spacecraft. Partly because of this, it was no earlier than in 1976, when the first two truly successful landers, *Viking 1* and *Viking 2* by NASA, touched down on the surface of Mars [3]. Before this, the Soviet Union had performed the first soft landing on Mars with its *Mars 3* probe in 1971, but the lander failed immediately after touchdown.

The Viking landers photographed their landing sites, took and analyzed surface samples, deployed seismometers and studied the atmospheric composition and meteorology. The Viking landers produced the only pressure time series to date, which cover a full Martian year. The knowledge on seasonal variation of pressure on Mars is still mainly based on these measurements. [4]

After the Viking landers, shorter pressure measurement series have been carried out by the *Mars Pathfinder* in 1997 and *Phoenix* in 2008. Currently, NASA's new Curiosity rover is performing pressure and humidity measurements in the Gale crater. Curiosity's design life time is one Martian year, meaning that scientists are now expecting long term meteorological data with seasonal variations for the first time after the Viking landers.

FMI is currently preparing pressure and humidity instruments for the ExoMars EDM project of ESA, which is described in greater detail in Section 2.2. The mission is planned to launch in 2016.

In addition to the in-situ measurements of Mars landers, there has been a multitude of atmospheric remote sensing measurements from the Martian orbit. Currently, the atmosphere is studied from orbit by ESA's *Mars Express* and NASA's *Mars Reconnaissance Orbiter*. These orbiters are to be joined by NASA's *Mars Atmosphere and Volatile Evolution* (MAVEN) probe and *Indian Space Research Organization's* (ISRO) *Mars Orbiter Mission* (MOM) launched in November 2013, and ESA's Trace Gas Orbiter launched in 2016.

The atmosphere of Mars has been observed to consist almost entirely (95 %) of Carbon dioxide (CO₂) and traces of other gases, such as Nitrogen and Argon [5]. The thin layer of gas is relatively cold compared to atmospheric temperatures encountered on Earth. For example, ambient gas temperature near the surface varied at the Mars Pathfinder landing site consistently from -76 °C to -10 °C. The pressure varied during this period from 6.5 hPa to 6.9 hPa. [6]

2.1.2 Involvement of Finnish Meteorological Institute

The scientific research conducted by the Finnish Meteorological Institute focuses on meteorology, air quality, climate change, remote sensing, marine research, arctic research and the interactions between solar wind and the near Earth environment, including space weather [7]. Planetary science studies of, for example Mars, are not directly part of the main objectives of FMI. Nevertheless, research on the Martian atmosphere can be utilized in studies of the Earth's atmosphere, and ultimately on meteorology. Traditionally atmospheric and climate models have been based on the observations of the atmosphere of the planet Earth. To ultimately understand the different phenomena and mechanisms acting on the Earth's atmosphere, studies on various types of atmospheres are needed also. Thus, FMI has been actively participating in the study of the atmospheres of Mars and the Saturn's largest moon, Titan.

The FMI involvement in extraterrestrial atmospheric instruments began with the development of a pressure instrument for the Russian led *Mars 96* project. Unfortunately, this instrument never reached Mars, due to a launch vehicle failure at the start of the mission in 1996. FMI delivered similar instruments also for the NASA led *Mars Polar Lander* in 1998 and the British *Beagle 2* in 2002 [4, page 17]. Again, these missions faced the same ill fate as the Mars 96 mission. Contact was never established with neither of the landers after their expected landing.

The first pressure instrument by FMI on a successful mission was actually not on a mission to Mars. That particular instrument was aboard the Huygens probe of the Cassini-Huygens mission to Saturn. The Cassini-Huygens combination was launched in 1997 and entered Saturn's orbit in 2004. Later, the Huygens atmospheric probe separated and successfully entered the atmosphere of Saturn's largest moon, Titan in 2005. The electronics design for Huygens pressure instrument was similar to the designs done for the Mars landers, although Titan's mean surface pressure is around 1500 hPa, whereas Mars has mean surface pressure around 6 hPa [8]. The largest differences between the designs were concerning the instrument sensor selection.

Finally, in 2008 the Mars Phoenix lander delivered successfully the first FMI pressure instrument onto the Martian soil, onto the high northern latitudes of Mars. Following this, another pressure instrument and the first direct humidity instrument was brought close to the equator of Mars aboard the NASA's Curiosity rover in 2012.

Besides contributions to atmospheric planetary instruments, Finnish Meteorological Institute has also participated in various other space projects like the *Solar and Heliospheric Observatory* (SOHO), *Rosetta*, *SMART-1*, *Mars Express* and *Venus Express*. These contributions have been mainly related to designing and building space weather instruments and instrument data processing units.

2.2 ExoMars mission

2.2.1 General

The ExoMars project has been in considerable turmoil several times during its history. After the start of the ESA's high priority project in early 2000s, the co-

operation partners and the configuration of the mission have changed considerably. At present, the ExoMars project is done in co-operation with the Russian space agency Roscosmos. The plan consists of a 2016 launch of the Trace Gas Orbiter and the Entry, descent and landing Demonstrator Module. A second launch including a higher amount of Russian participation is planned in 2018. This launch would deliver a rover with instruments primarily focused on exobiology studies. Both launches will use the Russian Proton launch vehicle. [1]

The Trace Gas Orbiter will benefit Martian atmospheric sciences, as well as exobiology studies. TGO is designed to detect and locate the sources of low concentration atmospheric gases, such as methane and water vapor. The results will help in determining whether the sources of methane could be related to biological or geological activity. The orbiter will also act as a data relay satellite for its accompanying EDM lander, and the ExoMars rover to be launched in 2018.

2.2.2 EDM lander

The Entry, descent and landing Demonstrator Module is primarily a technology demonstration mission, which tests various technologies needed for a safe, high precision landing for future European Mars missions.



Figure 1: ExoMars EDM powered landing onto the surface of Mars. (Credit: ESA)

The ExoMars EDM will be the first European Mars lander performing a powered

precision landing, as seen in Figure 1. The entry, descent and landing sequence of ExoMars EDM starts with the entry to the Martian atmosphere under the protection of a heat shield. When the spacecraft velocity has lowered down to Mach 2, a parachute is opened, which further slows down the lander to Mach 1. After jettisoning the parachute, liquid thrusters are fired to perform an autonomous controlled landing. The thrusters are shut down slightly before the touchdown to protect the lander from debris sent flying by the rocket jets. The final excess velocity is absorbed by shock absorbing material installed under the lander.

2.2.3 DREAMS science package

In addition to the primary technology demonstration purposes, the EDM will include a small instrument suite called the Dust characterization, Risk assessment, and Environment Analyzer on the Martian Surface, or DREAMS. The science package, seen in Figure 2, is an autonomous battery powered system, and utilizes the EDM lander systems only for communications. DREAMS consists of six instruments, most of which are aimed at atmospheric research.

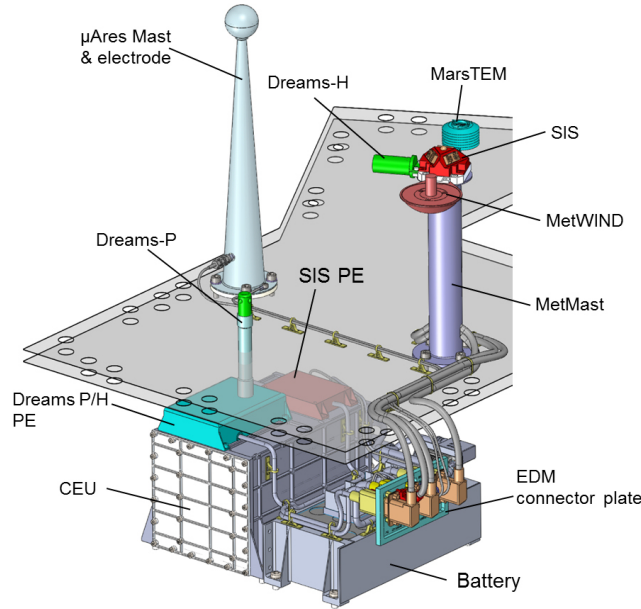


Figure 2: The DREAMS science package. (Credit: Thales Alenia, DREAMS team, ESA)

MarsTem is a temperature instrument, which has two platinum *Resistance Temperature Detector* (RTD) sensors: One fine definition sensor realized as an exposed net and a coarse definition sensor with higher response time, located inside the support structure of the instrument.

DREAMS-P and DREAMS-H are the pressure and humidity instruments designed and built by FMI. These instruments will be described in detail in the Chapter 3 of this thesis.

MetWind is a hot film anemometer which is capable of measuring horizontal wind speed and direction. An instrument with the same design was aboard the Beagle 2 lander, which was lost before or during its landing in 2003.

MicroAres is the first instrument to study electric fields on Mars. It consists of a single probe, which measures the DC and AC electric fields in the vicinity of the lander. The lander is scheduled to land during the dust storm season to increase the probability of dust devils and high winds, which are likely to cause elevated levels of electric field activity in the proximity of the lander.

The *Solar Intensity Sensor* (SIS) is an instrument designed for determining the solar irradiance intensity and direction. The measurement principle is based on Silicon photodetectors, interference and density filters, as well as mechanical field of view limiting masks. The instrument can also measure dust opacity, and was originally designed for the MetNet lander, which is further discussed in Section 7. [9]

3 System Design of DREAMS-P/H instrument

3.1 General description

The DREAMS-P and DREAMS-H instruments build on the extensive experience of FMI with similar instruments. In this Chapter, the high level design of the pressure instrument and the humidity instrument is presented. The interfaces between the pressure and humidity instruments, as well as interfaces between the instruments, the lander and the environment are also presented in the next Sections, with the focus on the most relevant interfaces for the DREAMS-P/H controller unit. The internal structure of the instrument controller housed on the DREAMS-P board is presented in Chapter 4.

The DREAMS-P element is placed on top of the DREAMS *Common Electronics Unit* (CEU). DREAMS-H is placed onto the side of an instrument mast, which is attached to the CEU box. While the pressure and humidity instruments are not mechanically connected, they are in electrical sense a single integrated system. [9]

3.2 DREAMS-P pressure instrument

The DREAMS-P element of the system contains the two pressure transducers, referred to as DREAMS-P1 and DREAMS-P2, and the *Proximity Electronics* (PE). The Proximity Electronics, also referred to as the controller, are used to control the pressure and humidity transducers, to perform scientific measurements and to communicate with the CEU of the DREAMS package. These components will be further discussed in Chapter 4.

The pressure transducers of DREAMS-P are based on the Vaisala Barocap® technology. The operation principle of the micro-machined silicon sensors is presented in Figure 3. A capacitor is formed between the electrodes placed on the glass substrate and silicon diaphragm of the vacuum chamber. As the ambient pressure varies, the level of diaphragm deflection towards the vacuum chamber varies, thus changing the capacitance of the sensor.

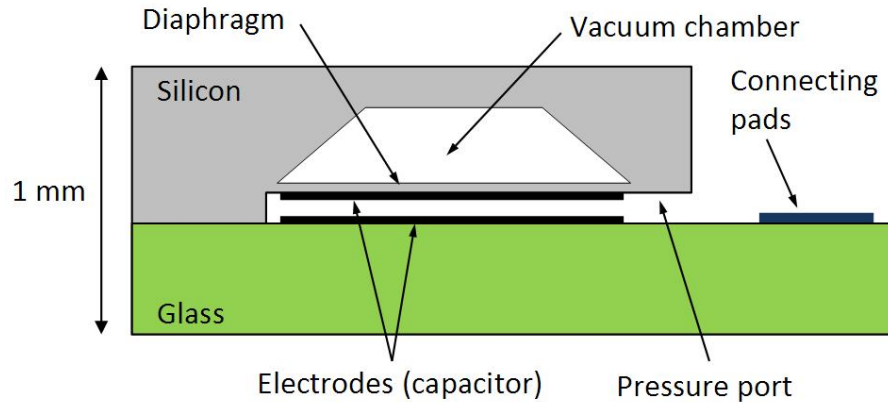


Figure 3: Barocap® sensor simplified construction. (Credit: FMI DREAMS team)

Measurements are performed by connecting the capacitive sensors as a part of an RC oscillator circuit. Capacitance of a Barocap® sensor is determined by frequency of this oscillator. The frequency can be determined by two alternative methods.

The first method to determine the frequency is to count the number of received pulses within a fixed time window. The frequency can be then directly calculated by dividing the received pulse count with measurement time. This method gets more accurate as the measured frequency rises, because the error is proportional to the length of a measurement pulse.

The method used for the DREAMS-P/H instrument is ideally limited in accuracy only by the length of one reference pulse, thus it is accurate also with lower frequencies. However, this method cannot be used for relatively high frequencies for reasons explained in Section 5.3.5. In this approach, the frequency is determined by simultaneously counting a number of sensor pulses and reference oscillator pulses until a preset amount of sensor pulses is received. Equation 1 shows how the channel frequency f can be calculated using the number of sensor pulses received n , the number of reference pulses counted c and the reference oscillator frequency f_{CLK} .

$$f = f_{CLK} \frac{n}{c} \quad (1)$$

Then, the pressure sensor capacitance can be deduced by comparing the frequency measured for the Barocap® channel, with frequencies of known stable capacitors in the same transducer. Finally, the ambient pressure can be obtained from the sensor capacitance. The transducers also include Vaisala Thermocap® channels, which are measured to negate the temperature dependencies of the pressure sensors. This thesis will not go into detail to explain the further calibration and data analysis techniques, of which some are proprietary.

For operation, the pressure transducers require a stable +5 V voltage, which is regulated by dedicated voltage references. The references are supplied with a +12 V voltage fed through dual transistor switches, operated by the instrument controller.

The DREAMS-P transducer is separated from the proximity electronics and other noise sources with a structure called the *Faraday hut*. The Faraday hut consists of blank pieces of *Printed Circuit Board* (PCB), which are soldered to form an electrically conductive box on top of and under the transducer area of the DREAMS-P PCB. The inside of the hut is metallized and connected to the DREAMS-P analog ground. One half of the hut covers one pressure transducer on top of the PCB and the another a second transducer under the PCB.

The DREAMS-P mechanical configuration is visualized in Figure 4. The instrument is installed into an aluminum cover with openings for the pressure inlet tube, as well as for the CEU and DREAMS-H cable harnesses. The DREAMS-P circuit board is fixed to this mechanical cover with screws, and the cover is secured on top of the common electronics box with screws. The size of the assembled instrument without the pressures inlet system is about 9 cm × 6 cm × 2 cm.

The pressure instrument is placed inside the EDM lander warm compartment with the CEU box and DREAMS batteries. This means that for useful pressure measurements, there needs to be a way to access the atmosphere from the warm

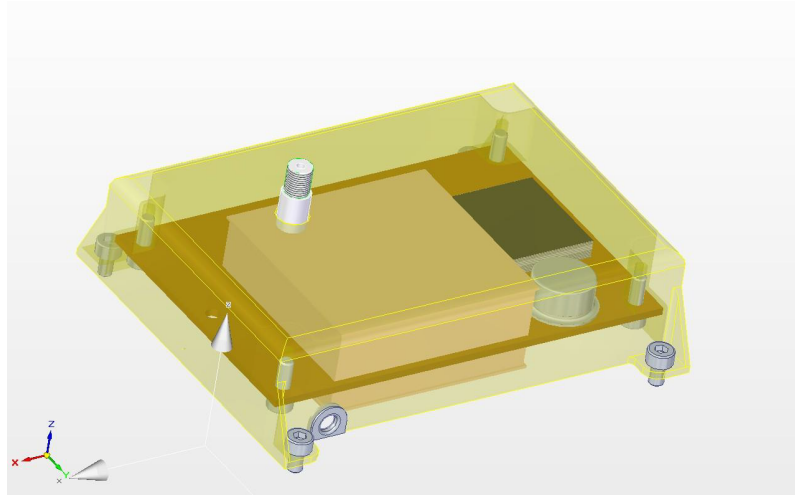


Figure 4: *Computer Aided Design* (CAD) model of the DREAMS-P instrument inside its mechanical cover. (Credit: FMI DREAMS team)

compartment. This is accomplished by a pressure inlet system, illustrated in Figure 5. A rigid interface tube structure mounted on top of the DREAMS-P mechanical cover protrudes through the lander warm compartment thermal hardware. The tube is connected to the smaller pressure sensor pipe leading into the Faraday hut of DREAMS-P. A protective hat called the gas inlet block is installed on top of the interface tube, to prevent dust easily entering the instrument.

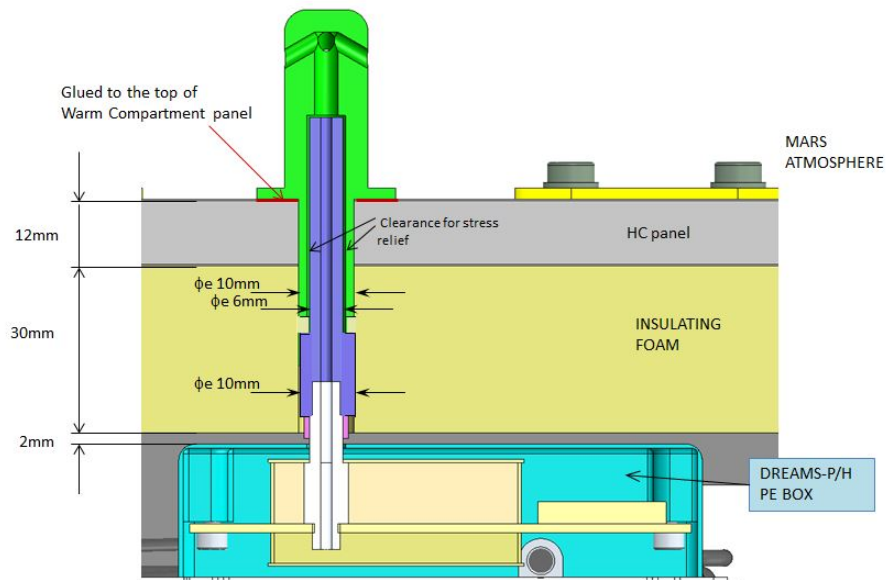


Figure 5: DREAMS-P pressure inlet system. (Credit: FMI DREAMS team)

3.3 DREAMS-H humidity instrument

The humidity transducer, DREAMS-H, is based on the Vaisala Humicap® sensor technology. These sensors change their capacitance relative to the ambient humidity. The measurement process of Humicap® sensors is similar to the measurement process of Barocap® sensors, presented in Section 3.2. However, the Humicap® sensors are more temperature dependent, so accurate reference temperature measurements are even more important with DREAMS-H.

Operating voltage for DREAMS-H is regulated with a +5 V voltage reference, similarly as for DREAMS-P. Due to the limited space on the DREAMS-H PCB, the voltage is regulated on the DREAMS-P board and routed through the cable harness to the humidity instrument.

A special feature about DREAMS-H transducer is the need for defrosting and regeneration of the Humicap® sensors. The sensors operate by absorbing humidity into a polymer between the plates of a capacitor, thus changing the dielectric constant of the material and capacitance of the sensor. However, the measurements may be corrupted by water vapor forming ice inside the sensors in freezing temperatures, or substances outgassing e.g. from the lander and contaminating the sensor.

The Humicap® sensors have built in heating resistors to deal with the aforementioned situations. In case of ice in the sensor, defrosting can be performed by heating the sensor up to +100 °C for 3–5 minutes [10]. And, regeneration of contaminants is achieved by heating the sensor up to +145...+150 °C [10] for 5–8 minutes [11].



Figure 6: DREAMS-H qualification model without the dust filter. (Credit: FMI DREAMS team)

As seen in Figure 6, the DREAMS-H is protected from external *Electromagnetic Interference* (EMI) sources with a cylindrical, open-ended *Faraday shield*. The aluminum shield also provides some mechanical protection against debris during the landing. Perforations are made on one quadrant of the cylinder to facilitate easy exchange of gases i.e. water vapor, between the instrument and the environment. The open end of the cylinder is covered with a filter cap to prevent dust entering into the transducer. Because the filter slows down the movement of humidity through it, the cap only partly covers the perforated area. The Faraday shield holes are facing downward and the Humicap® sensors are mounted below the DREAMS-H PCB to mitigate the effects of dust entering through the uncovered perforations in the shield. The assembled cylindrical instrument is about 5 cm long and has a diameter of about 2 cm.

3.4 Transducer control

All transducers, DREAMS-P1, DREAMS-P2 and DREAMS-H are controlled similarly. Each transducer has 8 channels populated with sensors and reference capacitors. These channels are connected to a Vaisala proprietary *Application Specific Integrated Circuit* (ASIC), with a few other supporting components.

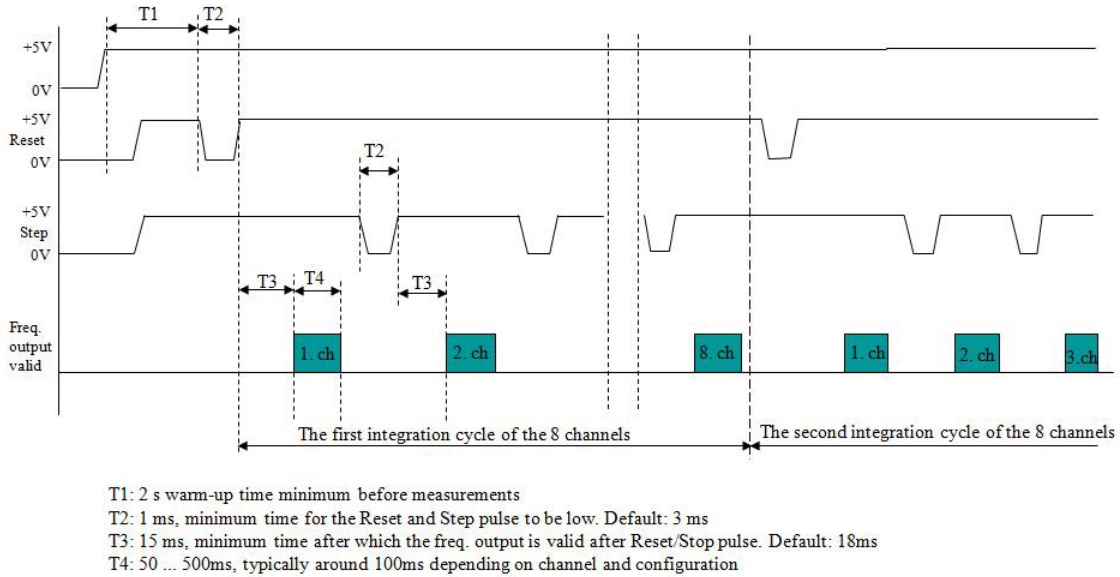


Figure 7: DREAMS-P/H control signal timing. (Credit: FMI DREAMS team)

There is one frequency channel available from the frequency output line of a transducer at a time. The channel shown on this line is selected by issuing RESET and STEP signals to the Vaisala ASIC. RESET and STEP lines are normally in high state as seen in Figure 7. A control signal is issued by pulling the respective control line to ground for at least 1 ms. A STEP signal increments the channel number, while a RESET signal always returns the transducer to channel number one. Following a control signal, the measurement channel is changed and the new channel stabilizes in about 15 ms.

The Vaisala ASIC draws power from the stable regulated +5 V supply. When the power is switched on, it takes around 2 s before the output frequency signal is considered valid.

3.5 Interfaces

3.5.1 Electrical interfaces

This Section explains the electrical interfaces to the DREAMS-P/H system and inside it. First, the interface between the DREAMS-P/H and the DREAMS CEU is described. After that, the interface between the DREAMS-P and the DREAMS-H is discussed.

The DREAMS-P/H system is connected to the DREAMS CEU by a pig-tail harness directly soldered to the proximity electronics on DREAMS-P board. The far end of the pig-tail harness has a Soriau HD-sub 15 male connector. Table 1 describes the signals carried by the CEU cable harness pins illustrated in Figure 8.

Table 1: CEU connector pin signals.

| Pin | Signal |
|-----|--|
| 1 | +12 V in |
| 2 | Digital ground (NC on DREAMS-P) |
| 3 | Not connected |
| 4 | +RXD |
| 5 | −RXD |
| 6 | Analog ground (+12 V GND) |
| 7 | Digital ground (NC on DREAMS-P) |
| 8 | Not connected |
| 9 | RX-shield (NC on DREAMS-P) |
| 10 | TX-shield (Digital ground) |
| 11 | +5 V in |
| 12 | Digital ground (+5 V GND) |
| 13 | Not connected |
| 14 | +TXD |
| 15 | −TXD |

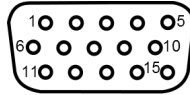


Figure 8: CEU connector pin layout.

The cable harness carries +5 V operating voltage for the proximity electronics and +12 V for the transducer regulators. The ground planes for these voltage rails are separated in the DREAMS-P/H system, and only connected in CEU. This is to reduce PE digital noise in the transducer supply ground.

Communication signals are carried inside shielded differential twisted pairs. One pair (TXD) is used for transmitting telemetry from DREAMS-P/H to CEU, and the another one (RXD) for receiving telecommands from CEU to DREAMS-P/H. The TXD twisted pair shield is connected to DREAMS-P digital ground, while the RXD twisted pair shield is grounded on the CEU side. In addition, the complete CEU wire harness is covered with a braided shield connected to the DREAMS-P mechanical cover and the HD-sub 15 connector housing.

The purpose of the two-level shielding of the data lines is to reduce *Electromagnetic Compatibility* (EMC) issues. Besides the shielding, ESA also requires that

the power and other signals are separated by grounded pins in the CEU connector to mitigate EMC problems. These pins (2, 7 and 12) are connected to the digital ground in DREAMS CEU, but left unconnected on the DREAMS-P side to prevent ground loops.

DREAMS-P and DREAMS-H both have a dedicated pig-tail cable harness directly soldered on the instrument PCB, which is used to connect them together. The DREAMS-P part of the cable is connected with a Soriau 13-socket Micro Comp female connector to the DREAMS-H 13-pin Micro Comp male connector. The connectors are joined through a interconnection plate in the DREAMS structure. Table 2 describes the signals carried by the DREAMS-H cable harness pins illustrated in Figure 9.

Table 2: DREAMS-H connector pin signals.

| Pin/Socket | Signal |
|------------|--------------------------------------|
| 1 | Power_H (+5 V analog) |
| 2 | Heater in (c. +5 V analog) |
| 3 | Heater in (c. +5 V analog) |
| 4 | Heater return (Analog ground) |
| 5 | Not connected |
| 6 | STEP_H |
| 7 | RESET_H |
| 8 | Analog ground |
| 9 | Not connected |
| 10 | Heater return (Analog ground) |
| 11 | Not connected |
| 12 | Not connected |
| 13 | Frequency out |

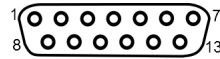


Figure 9: DREAMS-H connector pin layout.

The cable harness between DREAMS-P and DREAMS-H carries the DREAMS-H RESET and STEP control signals, as well as the frequency output signals. The regulated +5 V transducer supply voltage and the heater supply voltage are also carried in this cable. Two input and two return lines are used for the Humicap® sensor heating to reduce resistive losses in the lines. The power lines are separated in the connector with analog ground pins.

3.5.2 Telemetry and telecommand data interface

The DREAMS-P/H receives commands from and transmits data to the CEU via an asynchronous RS-422 connection operating at 100 kbps baud rate. RS-422 is a differential serial communication standard which supports one transmitter at a time and multiple receivers. The Freescale MCU is connected to an external peripheral chip, MAX488MJA RS-422 transceiver, which interfaces the *Serial Communication Interface* (SCI) module of the MCU with the differential data bus. MAX488 also effectively isolates the DREAMS-P/H system from the CEU.

A single character on the RS-422 line consists of 11 bits: a start bit, 8 data bits and a stop bit. Note that a parity bit is not used. Instead, data transmission is validated using checksums. The data bytes have the following properties [11]:

- Start bit = 0
- Data bits: *least significant bit* (lsb) first
- Stop bit = 1

3.5.3 Debugger interface

An interface for debugging during development and testing of DREAMS-P/H software was deemed necessary in the system hardware design. The Freescale MC9S12XEP100 microcontroller, used in DREAMS-P/H proximity electronics, can be programmed and monitored through its *Background Debug Module* (BDM) interface. This interface can be accessed in practice by inserting the MCU to a socket of an evaluation board or directly through a set of pins on the microcontroller. These pins are routed to a set of four access pads on the DREAMS-P PCB.

A cable with a six-pin pin header connector is soldered on to the 4 debugger interface pads of the DREAMS-P PCB. The interface cable is attached to the board during the beginning of the instrument assembly process. Signals going through the interface are listed in Table 3. This cable can then be used to program and debug the microcontroller without removing the MCU from the instrument, as described in Section 4.5.

Table 3: Debug interface signals.

| Pin | Signal |
|-----|-------------------------------------|
| 1 | BGND (Debugger two-way data) |
| 2 | Not connected |
| 3 | Not connected |
| 4 | Ground |
| 5 | Reset |
| 6 | VDD |

The debugger interface is used extensively in the software verification test procedure. After the test process is successfully carried out, the debugger interface cable can be removed. It is removed as the instrument qualification model and flight model assembly process continues towards the complete assembled instrument.

3.6 System requirements

As a space instrument, the DREAMS-P/H encounters environments, which would be unusual for a meteorology instrument on Earth. The most harsh environments are encountered during the launch, cruise to Mars and on the surface of Mars.

The launch of the ExoMars EDM will exert significant vibration and shock loads on the pressure and humidity instruments. The vibrations are caused by the launch vehicle rocket engines, whereas the shocks occur mainly due to pyrotechnical events such as stage separations. This imposes requirements on the mechanical design of the instrument support structures, as well as on the workmanship of the electronic assemblies.

During the cruise to Mars, DREAMS-P/H will be subjected to the vacuum of space, elevated radiation doses and an exotic thermal environment. These effects are partly mitigated by the protective cruise stage around the lander, but not fully.

First of all, vacuum causes some materials to release excessive outgassing compounds. The outgassed material may ruin e.g. sensor heads, and the parent material properties may change. This limits the selection of available materials for the instruments. This problem concerns especially adhesives and coatings.

When the spacecraft exits the Earth's atmosphere, elevated levels of galactic cosmic rays and solar energetic particles start causing radiation damage to the electronic components, if powered. This is especially dangerous for the microcontroller intended for controlling the DREAMS-P/H measurements. The radiation might cause bit flips in the MCU memory, which can lead to hanging of the system or in the worst case, damage to the microcontroller. These effects need to be taken into account in the design of the DREAMS-P/H controller software. Fortunately, the cruise stage provides significant protection against the radiation load, and the radiation levels observed during the cruise of Mars Science Laboratory were relatively small [12].

In addition to the radiation environment, it is noteworthy to realize that heat transfer in vacuum of space is possible only by radiation and conduction. Thus, in the absence of convection, heat transfer is slower, and hotspots in the instrument may form. However, the thermal dissipation power produced by DREAMS-P and DREAMS-H instruments is small. Also, the temperature inside the cruise stage is actively controlled to stay near room temperature values.

On the other hand, on Mars the temperature variations are much more severe. Again, DREAMS-P is in a thermally controlled warm compartment, but DREAMS-H is installed outside on the lander meteorological mast. The ambient temperature might vary in the range of $-90\text{ }^{\circ}\text{C}$ to $+10\text{ }^{\circ}\text{C}$. There is a very thin atmosphere, but heat transfer by convection is small compared to other transfer mechanisms. As DREAMS-H needs to directly sense the ambient gases, the only

solution is to qualify the instrument for these extreme temperatures.

Another challenge of the DREAMS mission after landing on Mars is the limited amount of energy available for operating the instruments. Solar panels power the lander during the cruise phase. But after that, as primarily an entry, descent and landing technology demonstration mission, ExoMars EDM is powered solely by batteries. The nominal lifetime of the DREAMS science mission is estimated to be four Martian days or *sols* [9].

The DREAMS-P/H power consumption figures were determined in the preliminary design of the system. These limits were presented to the DREAMS team in the FMI pressure and humidity instrument proposal, and accepted. In addition to these power consumption limits, the operation of instruments needs to be carefully scheduled, which limits the scientific operation of DREAMS-P and DREAMS-H.

Lastly, the problem with almost any space instrument is the impossibility to repair them after the launch. The only solution for this is the extensive environmental, mechanical, electrical and operational testing and qualification of the system. However, the flight model of the DREAMS-P/H instrument will have the possibility to update the operational software in-flight. Nevertheless, performing this action is not desired, since updating software remotely always carries also great risks.

4 DREAMS-P/H instrument controller unit

4.1 Controller overview

The DREAMS-P/H controller, also referred to as proximity electronics, is built around the Freescale MC9S12XEP100 microcontroller unit. The controller main components and interfaces are seen in Figure 10. The controller components are discussed in more detail in Sections 4.2 and 4.3, while the external interfaces are described in Section 3.5.

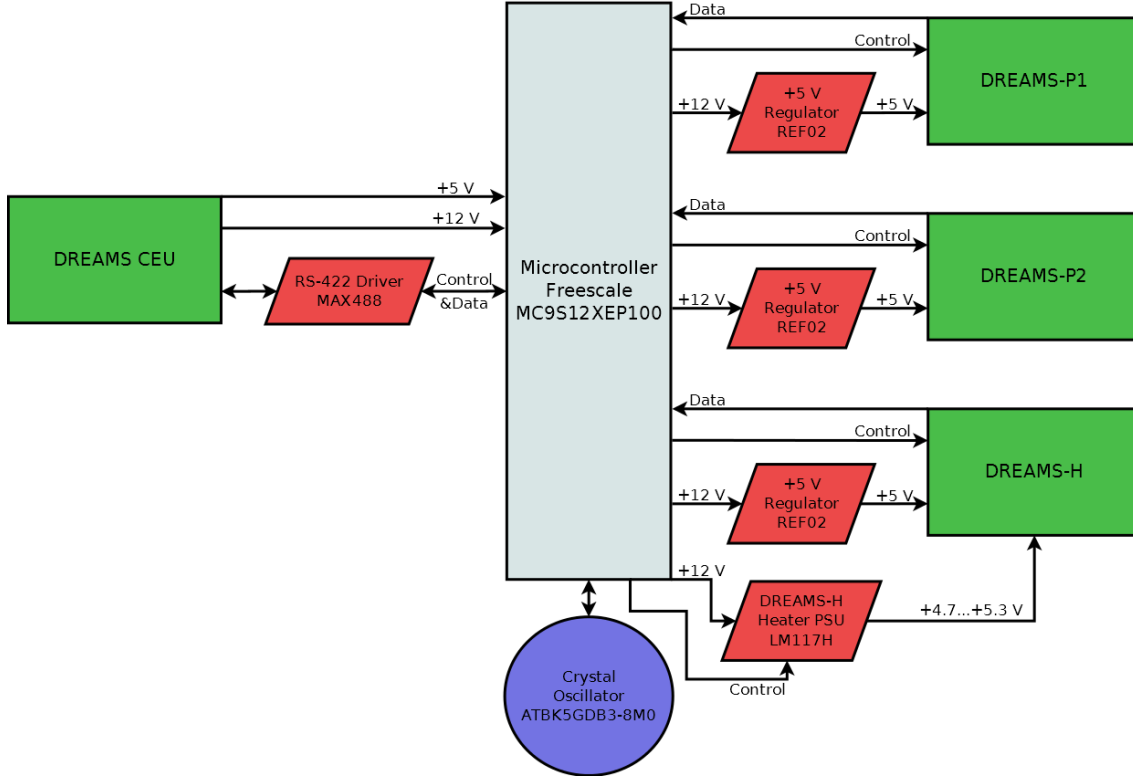


Figure 10: Simplified DREAMS-P/H instrument controller block diagram.

DREAMS-P/H controller telecommand and telemetry exchange occurs through the RS-422 transceiver, which converts differential data flow into 0 V and +5 V bits, suitable for the MCU *Serial Communication Interface* (SCI). Timing for serial communications, as well as for microcontroller operation and transducer pulse measurements is provided by the external 8 MHz crystal oscillator.

STEP and RESET control signals are used to control the transducers by the MCU. Scientific data is returned in the form of frequency signals.

Power for the controller and transducers is provided by regulated +5 V and +12 V voltage lines from the DREAMS CEU. The MCU and RS-422 transceiver are operated from the +5 V supply. The microcontroller steps the voltage further down to nominal operating voltages of +2.82 V and +1.84 V, with its internal regulator.

The flow of power to the pressure and humidity transducers is controlled by the MCU. The +5 V operating voltages for DREAMS-P and DREAMS-H are regulated locally on the DREAMS-P board from the +12 V supply. Furthermore, the microcontroller adjusts the output voltage of the DREAMS-H heater power supply for defrosting or regeneration. Heating voltage is supplied from +12 V and can be adjusted between +4.7 V...+5.3 V.

4.2 Microcontroller

4.2.1 Overview

The microcontroller chosen for the DREAMS-P/H controller unit is the commercially available automotive Freescale MC9S12XEP100. FMI has gained experience with this microcontroller from the *MetNet* project, where the MCU is used as the heart of the Russian build *MetNet On-Board Computer* (OBC). The *MetNet* project is further discussed in Chapter 7.

The 16-bit MC9S12XEP100 is a member of the S12X microcontroller family, which is descendant of the Freescale 68HC12 microcontroller family, and ultimately the Motorola 6800 microprocessor. The S12X series of MCUs has an integrated co-processor, known as the XGATE, running at twice the clock frequency of the main processor. The XGATE can be used simultaneously with the main CPU to handle computationally lighter tasks, e.g. serial communications. The MC9S12XEP100 has a maximum CPU bus frequency of 50 MHz, which translates into maximum XGATE bus frequency of 100 MHz. The MCU can be operated with a 3.3 V or a 5 V operating voltage supply.

The microcontroller has three types of memory: 64 KB of RAM and 1024 KB of P-Flash and 32 KB of D-Flash. The RAM is be used for temporary data storage during program execution. D-Flash can be used for non-volatile data preservation when the MCU is powered off. The application code is executed from the non-volatile P-Flash memory.

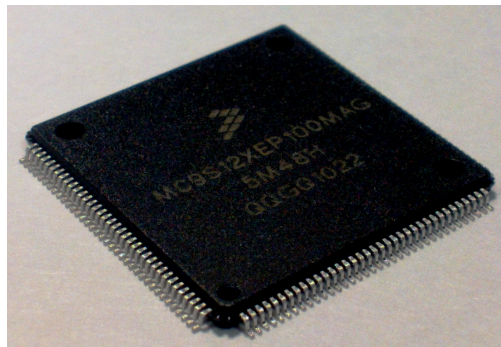


Figure 11: Freescale MC9S12XEP100 microcontroller in a 144-pin LQFP package.

A 144-pin *Low profile Quad Flat Package* (LQFP) was selected from the available 208-pin *Molded Array Process Ball Grid Array* (MAPBGA), 144-pin LQFP, 112-pin LQFP and 80-pin LQFP packages. This package, seen in Figure 11, provides the

most connections to outside world, while maintaining manual solderability. Manual soldering is desired in space instruments as the payloads are usually one-off products. The LQFP package also enables simple visual inspection as a verification method for the solder joint quality.

4.2.2 Peripheral modules

A multitude of peripheral modules is integrated on the microcontroller. The following modules are the most important for the DREAMS-P/H application:

- Port Integration Module
- Memory Mapping Control
- XGATE co-processor
- Clocks and Reset Generator
- Enhanced Capture Timer
- Periodic Interrupt Timer
- Serial Communication Interface
- Flash Modules

The Port Integration Module is used to configure the input/output pins of the MCU. The pins are grouped as 8 pin groups called ports, which are controlled via common configuration registers. There are 119 IO-pins available in the 144-pin LQFP package. These pins are utilized to operate DREAMS-P/H power switches, send control signals to the measurement transducers and to operate the DREAMS-H heater. The Port Integration Module also routes the transducer frequency measurement lines and the serial communications lines into relevant modules inside the MCU.

All MC9S12XEP100 memory areas can be accessed with a local or global addressing scheme. Additionally, XGATE has its own addressing scheme. The complete addressing process is controlled by the *Memory Mapping Control* (MMC) module. In the local addressing scheme, the memory is divided into areas called pages. To access data in the page, the page number is written into a relevant page register e.g. RPAGE for RAM. Then, the location can be accessed via a memory window (4 KB for RAM) inside the 16-bit local address space of the microcontroller. Alternatively, all memory locations can be accessed using the global addressing scheme. In this scheme, every memory address can be directly accessed using their 24 bit global addresses. The global scheme is applied whenever manual addressing is done in the DREAMS-P/H software, to clarify memory accesses.

The XGATE co-processor can be used to access almost all resources on the MCU. However, it is a *Reduced Instruction Set Computer* (RISC) processor, which

is optimized mainly for handling communication protocols to lower the interrupt-load of the main processor [13]. In DREAMS-P/H, the XGATE is used to handle all RS-422 communications related activities.

All necessary clock frequencies for the MCU operation are generated by the *Clocks and Reset Generator* (CRG). The module can be used to multiply or divide the frequency coming in from an external crystal oscillator. Also, all microcontroller resets are generated by the CRG. In DREAMS-P/H, among other tasks, CRG is used to multiply the crystal oscillator frequency and to generate watchdog resets.

The *Enhanced Capture Timer* (ECT) module has a free-running timer, which can be used for input capture or output compare. In input capture, an input pin is monitored for an active signal edge. When an edge is detected, the instantaneous timer count value is stored into a register. In output compare, the state of an output pin is switched when the free-running timer count reaches a predefined value. In DREAMS-P/H, input compare is used to detect the frequency pulses arriving from the pressure and humidity transducers.

When activated, the *Periodic Interrupt Timer* (PIT) generates interrupts after a predefined number of clock cycles. The time-out periods can be selected between 1 and 2^{24} bus clock cycles. As the DREAMS-P/H controller is fully interrupt-driven, the Periodic Interrupt Timer has an important role. The PIT is used to wake up the main processor from the WAIT mode after a specific task e.g. DREAMS-H regeneration has completed.

The Serial Communication Interface manages the physical layer of serial communications. The module converts the received bitstream into bytes, checks the incoming bytes for errors and converts the data bytes to be transmitted into bitstream. In DREAMS-P/H, the SCI module is connected to the MAX488 RS-422 transceiver.

The flash modules are used to store the application code of the microcontroller and non-volatile data. The flash areas for program code and non-volatile data are called P-Flash and D-Flash, respectively. The configuration tables of the DREAMS-P/H controller are stored on D-Flash, and fetched from there to RAM during execution. The purpose and structure of the configuration tables are discussed in more detail in Section 4.4.5.

4.2.3 Space qualification

There are generally two distinct approaches in qualifying electronic components for the harsh environment of space. The traditional way is to design, manufacture and test the components with the requirements of space environment in mind straight from the beginning. This generally results in small production batches, which are screened to select qualified components. Often, a large portion of the components in a batch may fail the qualification process, due to anomalies in the manufacturing process.

Another approach to component qualification is the use of *Commercial Off-The-Shelf* (COTS) components. COTS parts are mass produced components. The components may be intended e.g. for industrial, medical or automotive applications,

or normal consumer electronic devices. Because of the large production quantities and manufacturing process optimization, the reliability of these devices is generally high. However, COTS components are not designed for space, and thus need to be subsequently qualified for the loads specific for space, such as radiation and vacuum.

DREAMS-P/H project follows the growing trend of using COTS components. This allows for flexibility in component selection. Relevant tests have been performed on a batch of Freescale MC9S12XEP100 MCUs in order to qualify them for space.

Thermal qualification for the microcontroller was performed in the FMI space laboratory in Helsinki, Finland. The samples were cycled between $-65\text{ }^{\circ}\text{C}$ and $+70\text{ }^{\circ}\text{C}$ in an environmental chamber. Some of the samples were cycled 50 times, some 100. The microcontrollers were also subjected to extended cold periods in $-86\text{ }^{\circ}\text{C}$. All of the samples were successfully functionally tested after the thermal tests using an evaluation board [14]. This evaluation board is further discussed in Section 4.5.1.

The ionizing radiation tests will be done in the near future at the University of Helsinki Kumpula Accelerator Laboratory. A jig for powering up five microcontrollers during the radiation testing has been assembled. This is because most damaging radiation effects are observed only on powered devices. The design of the jig permits the unobstructed delivery of radiation on top of the MCUs.

Mechanical qualification of the MCU was done at HI-REL Laboratories, USA. HI-REL analyzed 60 samples of the MC9S12XEP100 MCUs with X-ray and *Scanning Acoustic Microscopy* (SAM) techniques. No unacceptable defects were observed in these inspections.[15] Three samples were further examined via a *Destructive Physical Analysis* (DPA) against a set of military standards, including MIL-STD-1580B. No unacceptable defects were observed in this analysis either.[16]

The microcontroller chip carrier plastic casing material was tested for outgassing at Instituto Nacional de Técnica Aeroespacial (INTA), Spain. The measured *Recovered Mass Loss* (RML) of the material was 0.073 % and the *Collected Volatile Condensable Materials*(CVCN) ratio was 0.000 %.[17] These values are clearly under the ESA outgassing limits of $< 1\text{ }%$ for RML and $< 0.1\text{ }%$ for CVCN. [18]

The MCU will undergo further environmental testing as part of the assembled DREAMS-P/H instrument. Also, more tests will be performed with the assembled DREAMS science package and EDM lander.

4.3 Supporting components

4.3.1 DREAMS-H heater power source

The voltage for the DREAMS-H heater resistors is provided by a National Semiconductor LM117 [19] adjustable linear voltage regulator. The output voltage of the LM117 is set with two resistors: R_1 connected between the regulator output pin and voltage adjustment pin, and R_2 connected between the voltage adjustment pin and ground. The output voltage can be calculated with Equation 2.

$$V_{OUT} = V_{REF}(1 + \frac{R_2}{R_1}) + I_{ADJ}R_2. \quad (2)$$

In nominal operation conditions the reference voltage V_{REF} is 1.25 V and the current from the adjustment pin, I_{ADJ} is 100 μA . [20]

The defrosting and regeneration of the Humicap® sensors in Martian conditions is typically done with 5 V voltage. However, because the atmospheric temperature on Mars varies widely, the capability to adjust the heating voltage is desired. The experience with FMI's REMS-H humidity instrument on board the Curiosity rover confirms this need. Thus, heating voltage options of about 4.7 V, 5 V and 5.3 V were required for DREAMS-P/H.

The selection of the heating voltage is executed by changing the R_2 by pulling various resistor combinations to ground utilizing the MCU, as illustrated in Figure 12. A 243 Ω resistor was selected as R_1 . This resulted in the selection of resistor values of 768 Ω , 4.75 k Ω and 10 k Ω , for forming the R_2 . The 768 Ω resistor is directly connected from the voltage adjustment pin to ground, while the 4.75 k Ω and 10 k Ω resistors are connected to the adjustment pin and can be drawn to ground with the microcontroller. This configuration resulted in the heating voltages presented in Table 4.

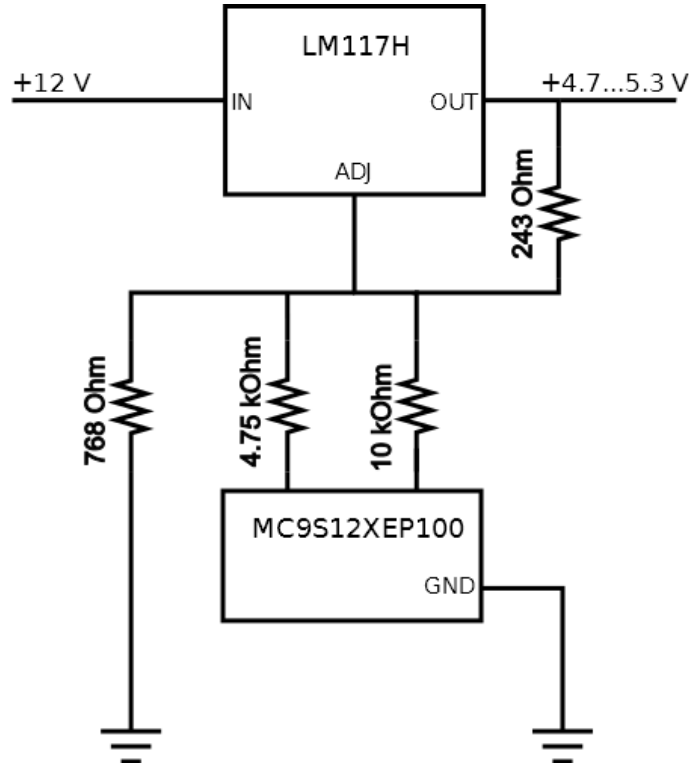


Figure 12: DREAMS-H heater power supply adjustment principle.

4.3.2 Crystal oscillator

DREAMS-P/H pulse counting measurements are performed by the microcontroller ECT module. That module uses the MCU bus frequency to timestamp the rising edge of each pulse. The first and last pulse timestamps of a measurement are used

Table 4: Calculated DREAMS-H heater voltages.

| Voltage setting | Resistors pulled to ground | Voltage |
|-----------------|--------------------------------|---------|
| 1 | 768 Ω , 4.75 k Ω | 4.72 V |
| 2 | 768 Ω , 10 k Ω | 4.99 V |
| 3 | 768 Ω | 5.28 V |

to calculate the measurement result as explained in Section 5.3.5. The generation of the required stable 8 MHz [21] bus frequency is explained below.

The MC9S12XEP100 microcontroller can use its internal Pierce oscillator or an external crystal oscillator as its clock source [13]. To guarantee high reliability and precision for frequency measurements an external Bliley Technologies crystal oscillator is used as the DREAMS-P/H clock source.

The Bliley Technologies ATBK5GB3-8M0 is an 8 MHz AT cut crystal oscillator, designed for aerospace use, harsh temperatures and large shock loads. The crystal is packaged into a TO-8 type package and mounted with five through-hole leads for mechanical reliability. [22]

DREAMS-P/H is specified to perform pulse counting measurements with a reference frequency of 8 MHz [21]. The MCU does not directly use the oscillator clock for pulse measurements. Instead, it uses the bus clock frequency, which is generated by dividing the oscillator clock frequency by two. From the FMI MetNet project a batch of space qualified Bliley 8 MHz oscillators is already available. The frequency of these oscillators can be multiplied with the *Phase Locked Loop* (PLL) module of the microcontroller. In DREAMS-P/H the PLL is used to reach an oscillator frequency of 16 MHz, and thus a bus frequency of 8 MHz.

4.4 Controller operational requirements

4.4.1 General

Requirements for DREAMS-P/H controller operation are defined in detail in the *DREAMS-P/H Software Requirements* document [21]. These requirements are complemented by the implementation concepts and restrictions in the *DREAMS-P/H Software Specification* document [23]. It is not the purpose of this thesis to restate all of the software requirements in detail. Instead, the Section 4.4 will describe the most important and driving operational requirements of the controller.

The highest level of operational control of the DREAMS-P/H is achieved through the controller commands. Telecommands via the RS-422 link are the only external interface for controlling the instrument. Hence, the system shall be controlled by a set of 15 telecommands presented in Table 8 [23].

The general structure of a telecommand is presented in Table 5. Each command is always started with the synchronization/address byte. The upper nibble of the byte 0 is always the hexadecimal *0xA*, and the lower nibble denotes if the command

is targeted to the system as a whole or for a specific transducer. The *0xA* pattern helps the MCU to recognize a beginning telecommand in case of noise or anomalies on the line. The allowed byte 0 address values are described in Table 6 [21].

Table 5: Telecommand package structure.

| Byte number | Value or designation | Description |
|---------------|----------------------|---|
| 0 | "1010" & ADDR | 4 synch frame bits and address of destination |
| 1 | Length | Number of bytes (n) in packet, following this byte |
| 2 | Command | Command byte + possible parameters (max 254 bytes in total) |
| $2 + (n - 1)$ | Checksum | XOR checksum of the command field |

Table 6: Command destination based on the telecommand synchronization/address byte 0.

| Byte 0 | Command destination |
|--------|------------------------|
| 0xA0 | DREAMS-P/H system |
| 0xA1 | DREAMS-P1 |
| 0xA2 | DREAMS-P2 |
| 0xA4 | DREAMS-H |
| 0xA5 | DREAMS-P1 and DREAMS-H |
| 0xA6 | DREAMS-P2 and DREAMS-H |

Next, the byte 1 of a telecommand determines the length of the remaining command in bytes. Byte 2 is the command byte, which defines the action requested by the command. In some commands, the command byte is followed by parameters affecting the command execution. Lastly, the command ends with a checksum byte, which is used to detect transmission errors in telecommands.

Incoming telecommands are always checked for errors. Their format needs to be correct: Byte 0 needs to match Table 6, the length field and actual length must agree and the received checksum has to match the checksum calculated from the telecommand. If the command format is incorrect, an error message shall be sent to the DREAMS CEU.

After telecommand reception, the command is also inspected to follow further command-specific and instrument state dependent requirements. If all of the requirements are met, an acknowledgement telemetry packet (ACK) shall be sent out and the command executed. The ACK packet may be followed by further telemetry

packages, depending on the command. Table 7 describes the general structure of telemetry packets.

Table 7: Telemetry package structure.

| Byte number | Value or designation | Description |
|---------------|----------------------|--|
| 0 | "1010" & ADDR | 4 synch frame bits and address of origin |
| 1 | Length | Number of bytes (n) in packet, following this byte |
| 2 | Telemetry | Telemetry type + data (max 254 bytes in total) |
| $2 + (n - 1)$ | Checksum | XOR checksum of the telemetry field |

The parameters for each type of command are listed in Table 8. More information about command handling requirements, e.g. various error types is available within Section 5.3.2, which describes the telemetry and telecommand handling implementation.

4.4.2 Instrument control

The most important instrument control requirement is that the DREAMS-P/H instrument shall be controllable under all operating conditions. This means that it is possible to stop the current operation regardless of the controller status. This is implemented with two commands, "REBOOT" and "STANDBY" which stop any operation like measurements or regeneration.

Also, the instrument status information shall be accessible and reflect the reality at all times. In effect, the "STATUS" command shall always provide a response containing the current DREAMS-P/H status, regardless what it is.

Transducers shall be powered on with "POWER_ON" commands and powered off with "POWER_OFF" commands. These commands will affect the transducer power switches as well as their STEP and RESET control signals. In addition, issuing "REBOOT" or "STANDBY" shall power off all transducers as well as return the controller to its default idle state. "REBOOT" additionally performs a software reset.

Simultaneous power for DREAMS-P1 and DREAMS-P2 shall be prevented by software. This is to prevent electromagnetic interference propagating from one pressure transducer to another. Also, powering up any of the transducers during the DREAMS-H regeneration shall be prohibited. Failure to follow this requirement might damage the Humicap® sensors or disturb the DREAMS-P measurements. The pressure measurements might get corrupted because the heater regulator may connect the analog and digital ground planes through a resistor in order to adjust

Table 8: The DREAMS-P/H controller shall implement the following telecommands.

| Command | ID | Number of 8 bit parameters | Parameter name | Parameter data |
|----------------------|------|----------------------------|--|--|
| "REBOOT" | 0x00 | 1 | Verif | 0x55 |
| "STANDBY" | 0x01 | 1 | Verif | 0xAA |
| "UPDATE_CONF" | 0x02 | 66 | Table_id Conf. data | Index specifying the transducer New configuration data (65 bytes) |
| "READ_CONF" | 0x03 | 1 | Table_id | Index specifying the transducer |
| "MEMORY_LOAD" | 0x04 | 5...253 | Address Data | Start address of the memory area to be updated (3 bytes) Update data |
| "MEMORY_VER" | 0x05 | 3 | Address | Start address of the memory area to be updated (3 bytes) |
| "MEMORY_UPD" | 0x06 | 3 | Address | Start address of the memory area to be updated (3 bytes) |
| "MEMORY_DUMP" | 0x07 | 4 | Address Length | Start address of the memory area to be dumped (3 bytes) Number of bytes requested |
| "STATUS" | 0x08 | 0 | | |
| "POWER_ON" | 0x09 | 0 | | |
| "POWER_OFF" | 0x0A | 0 | | |
| "MEAS" | 0x0B | 1 | Transducer | 1:P1, 2:P2, 4:H, 5:P1+H, 6:P2+H |
| "REGENERATE" | 0x0C | 3 | Voltage Time | 0:Automatic voltage, 1...3:Force selected voltage Duration of heating in seconds (2 bytes) |
| "TEST_SENSOR_CONFIG" | 0x10 | 2 | Pulses Time-out Channel_mask | Number of pulses to be measured (2 bytes) Time delay between channel selection and measurement start [ms] (2 bytes) Bit mask: Channels on/off, MSb: Ch. 8, LSb = Ch. 1 |
| "TEST_SENSOR_MEAS" | 0x11 | 0 | | |

the regeneration voltage. Additionally, simultaneous regeneration and measurement would exceed the power budget available for DREAMS-P/H.

One part of the instrument control is the ability to access and manipulate the memory of the Freescale microcontroller. In effect, "MEMORY_DUMP" command shall be able to access and retrieve data from any memory area. This is important for debugging purposes after instrument integration. Furthermore, small memory updates shall be possible using the combination of "MEMORY_LOAD", "MEMORY_VER" and "MEMORY_UPD" commands. First, "MEMORY_LOAD" stores the update data and destination address into a RAM buffer. Then, "MEMORY_VER" can be used to return this data from the buffer for the user to verify correct data before executing the update. Lastly, "MEMORY_UPD" is issued to execute the update by copying the data from the buffer to the destination addresses. However, to protect software stability, memory areas containing program code, which is being executed, shall not be updated. More information about updating the program code is available in Section 5.2.5.

4.4.3 Heater control

In DREAMS-H heater control, the driving requirements concern controlling the heating voltage and safety of the instrument.

The heater voltage shall be controlled manually or automatically to reach the required defrosting and regeneration temperatures in the Martian environment. In practice, the "REGENERATE" command may specify one of three predefined voltage levels. Alternatively, the controller may perform a temperature measurement with the Thermocap® sensor present in the DREAMS-H transducer, to determine the ambient temperature and select the heating voltage determined by configuration table information.

To prevent damage to the DREAMS-H instrument, some precautions need to be taken. The automatic heating function shall prevent heating in too warm temperatures. Likewise, powering up the DREAMS-H transducer shall be prohibited during regeneration or defrosting. Lastly, the heating process must be possible to stop at all times using the "REBOOT" or "STANDBY" commands.

4.4.4 Pressure and humidity measurements

It is required that the pressure and humidity measurements are performed in a way that they produce reliable results with consistent accuracy. Besides that, the control interface of the measurements shall be flexible and reliable.

The time base for the transducer pulse measurements is 8 MHz. When performing pulse counting, the software shall be able to determine the rising edge of a measurement pulse within the uncertainty of ± 1 reference pulse. This means that while measuring the time it takes to receive a number of pulses (e.g. 500) on a stable frequency, the reference clock time difference between the individual measurements is at maximum ± 2 pulses.

For flexible control of the measurements, it shall be possible to schedule measurements with two different approaches: manual and automatic. The manual measure-

ments are configured with the "TEST_SENSOR_CONFIG" command and initiated with "TEST_SENSOR_MEAS". The purpose of the manual test measurement commands is to maintain backwards compatibility to older meteorologic Mars instruments by FMI. These commands can be used in tests with old FMI ground support equipment.

On the other hand, the automatic measurements are intended for operational measurements on Mars and more advanced ground tests. Configuration tables, stored in the non-volatile memory of the microcontroller, are used to define the timing, number of repetitions and other specifications of the measurements. The command "MEAS" starts the first measurement sequence, which will be repeated automatically as defined in the configuration tables. The automatic measurement shall be able to measure DREAMS-P1 and DREAMS-H, or DREAMS-P2 and DREAMS-H simultaneously. This feature enables data acquisition without the active involvement of DREAMS CEU in the measurement control process.

4.4.5 Configuration tables

During the development and the mission of the DREAMS-P/H instrument, various measurement scenarios are encountered. Thus, the measurement parameters need to be easily reprogrammed and managed. This is accomplished by using configuration tables, which specify the measured channels during a measurement sequence, the number of pulses counted on the respective channels and the time-out periods of these measurements.

Additionally, the configuration tables store the number of repetitions of the measurement sequences and the time between these cycles. There are also fields in the tables for determining the DREAMS-H heating temperature limits.

Each transducer (DREAMS-P1, DREAMS-P2 and DREAMS-H) has a separate configuration table. The settings inside these tables can be updated with the "UPDATE_CONF" command and retrieved with the "READ_CONF" command.

The configuration table header information is illustrated in Table 9. Table 10 describes the format of the channel-wise configuration items. And, Table 11 explains the DREAMS-H regeneration temperature limits. The actual values in the configuration tables are Thermocap® pulse measurement results for these temperature limits. The regeneration limit fields are present in all three configuration tables, but only the limits in the DREAMS-H table are used for actual regeneration settings.

Table 9: Configuration table header information.

| Number of repetitions | Delay between measurement sequences [10 ms increments] |
|--|--|
| 0: continuous | |
| > 0: number of measurement cycle repetitions | 10...655 350 ms |

Table 10: Configuration table channel information.

| Channel | ON/OFF | Time-out [ms] | Number of pulses for measurement |
|---------|-------------|----------------------|-------------------------------------|
| 0...7 | 0/1 (1 bit) | 1...65 535 (16 bits) | 1...65 535 (16 bits) |

Table 11: Configuration table automatic DREAMS-H regeneration temperature limits.

| Temperature limit | Explanation |
|-------------------|---|
| Minimum | Lowest possible temperature for performing a successful regeneration with the highest heating voltage |
| Low/Normal | Temperature limit for deciding if the highest or medium heating voltage should be used for regeneration |
| Normal/High | Temperature limit for deciding if medium or the lowest heating voltage should be used for regeneration |
| Maximum | Highest temperature for performing a safe regeneration with the lowest heating voltage |

4.5 Development tools

4.5.1 Evaluation board

Most of the development, prototyping and programming of the Freescale microcontroller was performed with a SofTec Microsystems *EVB9S12XEP100 Starter Kit* evaluation board. The board, seen in Figure 13, is equipped with extensive features, some of them especially aimed for the automotive industry. The main interfaces of the board are [24]:

- USB to BDM interface for debugging
- BDM connector for external debugger device
- Header connectors for all MCU pins
- 12 V DC power connector
- Five CAN interfaces
- Six LIN interfaces
- Two RS-232 interfaces

- Set of user inputs and outputs, e.g. buttons and *Light Emitting Diodes* (LEDs)
- Prototyping area for through-hole and surface mount components

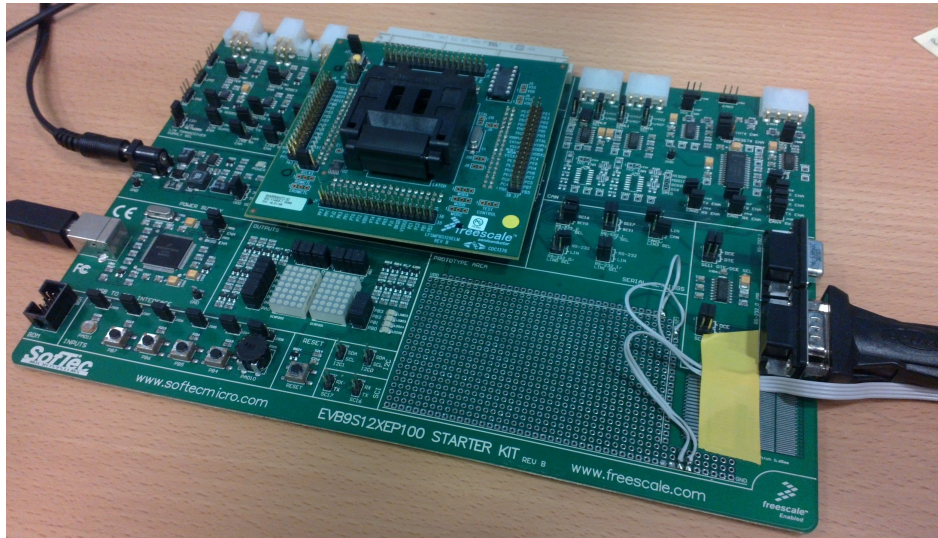


Figure 13: SofTec Microsystems EVB9S12XEP100 Starter Kit evaluation board, with a Freescale LFSMPBS12XELM daughter board attached on top.

It is possible to control the MCU via a direct USB connection from a PC to the evaluation board. However, a *P&E microcomputer systems USB BDM Multilink* interface can be also used to connect the PC with the EVB9S12XEP100. The P&E USB Multilink debugger is used also for the programming and debugging of the assembled DREAMS-P/H instrument.

As the MCU is soldered directly to the evaluation board, the EVB9S12XEP100 cannot be easily used to program more microcontroller units. Therefore, a daughter card, the *Freescale LFSMPBS12XELM*, was attached to the evaluation board. The daughter card is equipped with a socket for 144-pin LQFP MCUs. This socket can be utilized for safe non-damaging testing and programming of multiple MC9S12XEP100 chips.

The MCU pin header connectors were used during software development to interface the MCU with signal generators, multimeters and various models of earlier FMI pressure and humidity instruments. To help this, a development interface was built as a part of this thesis. The interface provides easy connectivity on to the evaluation board for pressure and humidity instruments using the default FMI pinout. Additionally, the development interface indicates the simulated status of the DREAMS-P/H power switch control pins and heater control pins with LEDs.

A serial RS-422 link for communicating with DREAMS CEU needed to be implemented during software development. A RS-232 interfaces on the evaluation board was used to simulate this link. This is possible, because the serial line protocol from the MCU to the RS-232 or RS-422 driver peripheral is identical. An RS-232/USB interface was used to connect the evaluation board RS-232 port to an USB port on the software development PC.

4.5.2 Freescale CodeWarrior IDE software

Freescale CodeWarrior *Integrated Design Environment* (IDE) is a software development suite for embedded programming of 8- and 16-bit microcontrollers manufactured by Freescale. The version used for the development of DREAMS software is the version 5.9.0, released in 2008. Freescale refers to this version as the "Classic IDE", and it supports programming in Assembly, C and C++ languages, and some of the C++ dialects. [25]

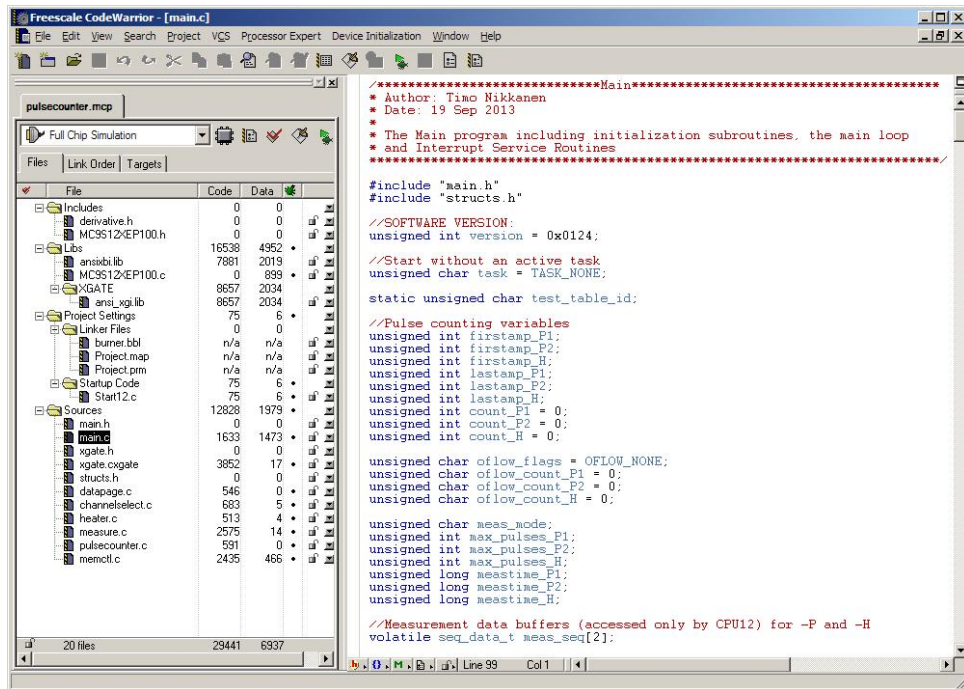


Figure 14: Freescale CodeWarrior main development window.

As the CodeWarrior IDE is opened, the main window, seen in Figure 14, is displayed. Adding and editing source files is done in this view. CodeWarrior also automatically creates default code for device initialization, device register aliases, memory mapping etc. The user can customize these files later, for example to change the MCU behavior during different types of resets.

Compiling the code is done with the compiler integrated to the CodeWarrior IDE. The compiler provides error and warning messages related to normal C syntax, as well as messages related to the device data and control registers. Prior to compilation, connection method to the target device needs to be defined. In the DREAMS-P/H project, the *P&E USB BDM Multilink* connection is selected for assembled instruments and the evaluation board. The evaluation board can also be programmed and debugged using the *SoftTec HCS12* connection. In addition, it is possible to simulate the execution of programmed software by selecting *Full Chip Simulation* as the connection method.

After successful compilation, the CodeWarrior True-Time Simulator & Real-Time Debugger application can be launched. Compilation is automatically per-

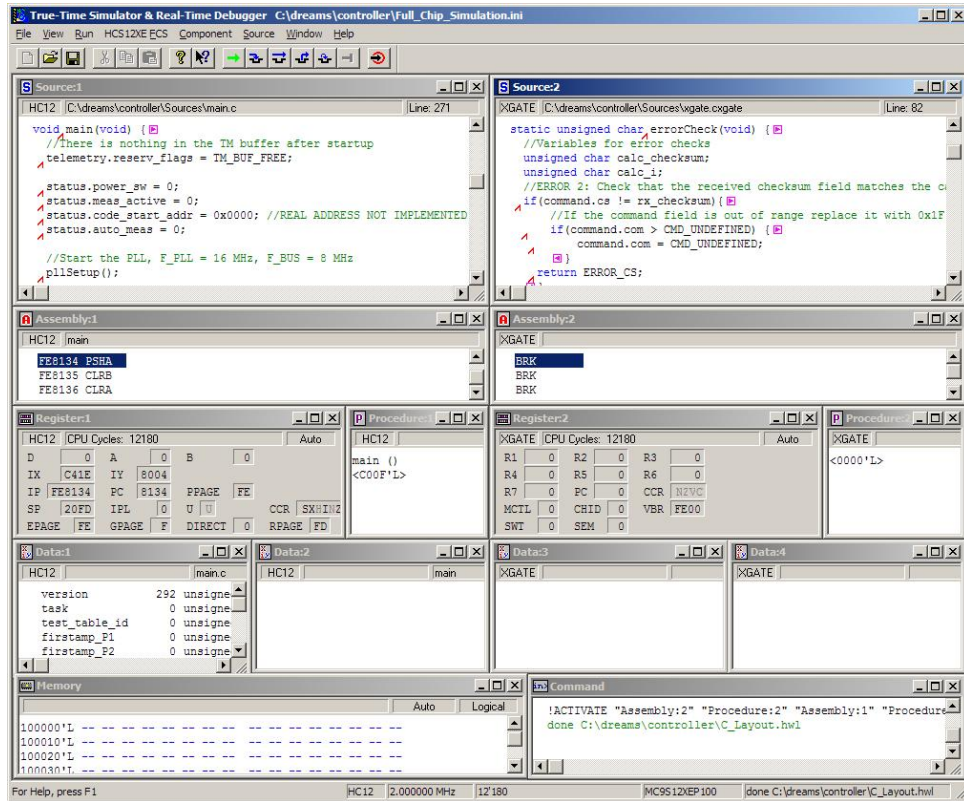


Figure 15: Freescale CodeWarrior True-Time Simulator & Real-Time Debugger.

formed, if there are changes to be compiled in the source files while launching the debugger software. At start-up, the debugger application clears the MCU program and data memory and writes the new program code into the MCU.

It is noteworthy to realize that there is no clear way of verifying automatically, if the new program code was uploaded to the microcontroller or not. This is because sometimes the debugger software just intercepts the MCU running the old code while displaying the new code. This flaw was handled later in the development by storing a version number of each update to a known memory location and inspecting this memory location always after launching the debugger application.

The debugger application, seen in Figure 15, has all the traditional debugger features: Program execution can be started and stopped, breakpoints and watch-points can be used to halt execution whenever a specific line is reached or a specific variable is accessed. It is possible to run the program line by line and manually modify variable values on the fly. In addition, it is possible to monitor any address in the microcontroller memory map, including device control registers.

As MC9S12XEP100 MCU has two CPU cores, the HCS12 main processor and the XGATE co-processor, debugging for both of these cores is needed. This is accomplished by the debugger software monitoring the program execution of both cores. Breakpoints can be placed simultaneously on code intended for both cores. However, when program execution is halted or run in steps, only one core is controlled by the debugger and the another one keeps on running independently.

4.5.3 DREAMS EGSE software

To simulate the DREAMS Common Electronics Unit, a software for the *DREAMS EGSE* (Electrical Ground Support Equipment) was developed in-house at FMI by Jouni Rynö. The *Tool Command Language* (TCL) based application, seen in Figure 16, is a simple software interface for issuing telecommands to and receiving telemetry from the DREAMS-P/H system. The complete DREAMS EGSE system consists of a Linux laptop PC, a USB/RS-422 interface and the TCL-based application software.

Telecommands are issued in the DREAMS EGSE software using syntax:

"ADDRESS" COMMAND_ID par_1...par_n,

where "ADDRESS" is replaced with "SYSTEM" for commands affecting the system as a whole. If the command affects specifically one or more of the instrument transducers, "DREAMS-P1", "DREAMS-P2", "DREAMS-H", "DREAMS-P1 DREAMS-H" or "DREAMS-P2 DREAMS-H" is used instead. The COMMAND_ID is replaced with the desired command from the Table 8. The optional parameters par_1...par_n can be input in decimal format e.g. 150, or in hexadecimal format e.g. 0xF8.

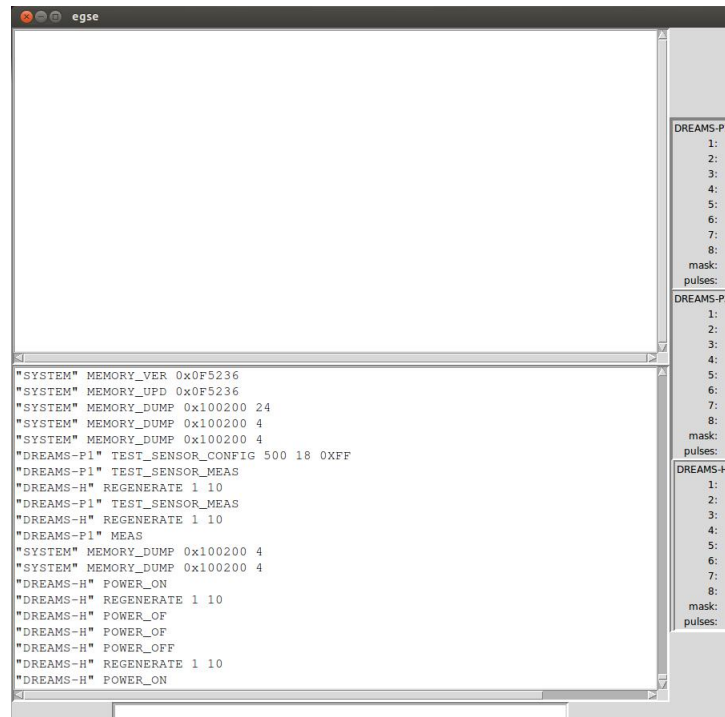


Figure 16: DREAMS EGSE software.

5 Controller operational software

5.1 Software system description

The DREAMS-P/H controller software is predominately programmed in C, with only a couple lines of Assembly instructions in the source files. The implementation strives to be primarily as simple and reliable as possible to meet the software specifications [23] and requirements [21]. A second objective is to optimize the implementation for low power consumption. On the other hand, optimizing code for efficiency or high-speed responsiveness has not been a specific goal in the development process. This is due to the relatively slow nature of the system input signals, e.g. serial communications or measurement transducer pulses.

A distinct feature of the software implementation is the parallel operation of the main HCS12 processor and the XGATE co-processor. The architecture and instruction set of the processors is quite different. HCS12 is a *Complex Instruction Set Computer* (CISC), while XGATE is a RISC computer. However, the instruction set differences are not clearly visible to the developer due to the use of C language.

Cooperation between the processors is achieved by a shared memory area and interrupts. Both cores are able to interrupt each other and peripheral interrupts can be configured to be handled by XGATE instead the main processor.

The compiled qualification model software consist of about 30 KB code and 7 KB data. The MC9S12XEP100 has 1024 KB of P-Flash intended for code and 32 KB D-Flash intended for data. These numbers result in the relatively low levels of resource utilization of 2.9 % for P-Flash and 21.9 % for D-Flash.

The controller software top-level implementation consists of a fairly simple main loop and a set of *Interrupt Service Routines* (ISRs). This is illustrated in Figure 19 in Section 5.3.1. As the software structure is highly integrated and interlinked, it is difficult to clearly divide the program code into distinct software modules based simply on the software structure. However, a division based on different tasks and operations performed by the software is presented in Section 5.3.

5.2 Operative principles

5.2.1 Non-volatile memory management

The DREAMS-P/H configuration tables are stored on the D-Flash side of the MC9S12XEP100 built-in flash modules. See Section 4.2.2 for details on the MCU flash implementation. The storing of the configuration tables to flash memory streamlines the controller operation, as the CEU does not have to send the configuration data to DREAMS-P/H after every reset.

Programming flash takes relatively long time compared to reading it. This is mainly due to the fact that a whole 256 byte D-Flash sector needs to be erased and rewritten to modify a single bit in the sector. Also, the flash manipulation commands are executed with a frequency of 1 MHz instead of the 8 MHz bus frequency of DREAMS-P/H. This means that there is a concrete risk of losing data if the MCU is reset e.g. due to a power loss during the flash operation.

To prevent loss of data, two copies of the configuration tables are stored into separate sectors in the D-Flash. These records of the tables are referred to as MAIN and BACKUP. One set of configuration tables occupies 210 bytes of the 256 D-Flash sector. The remaining, unused bytes in the sector are written with zeros.

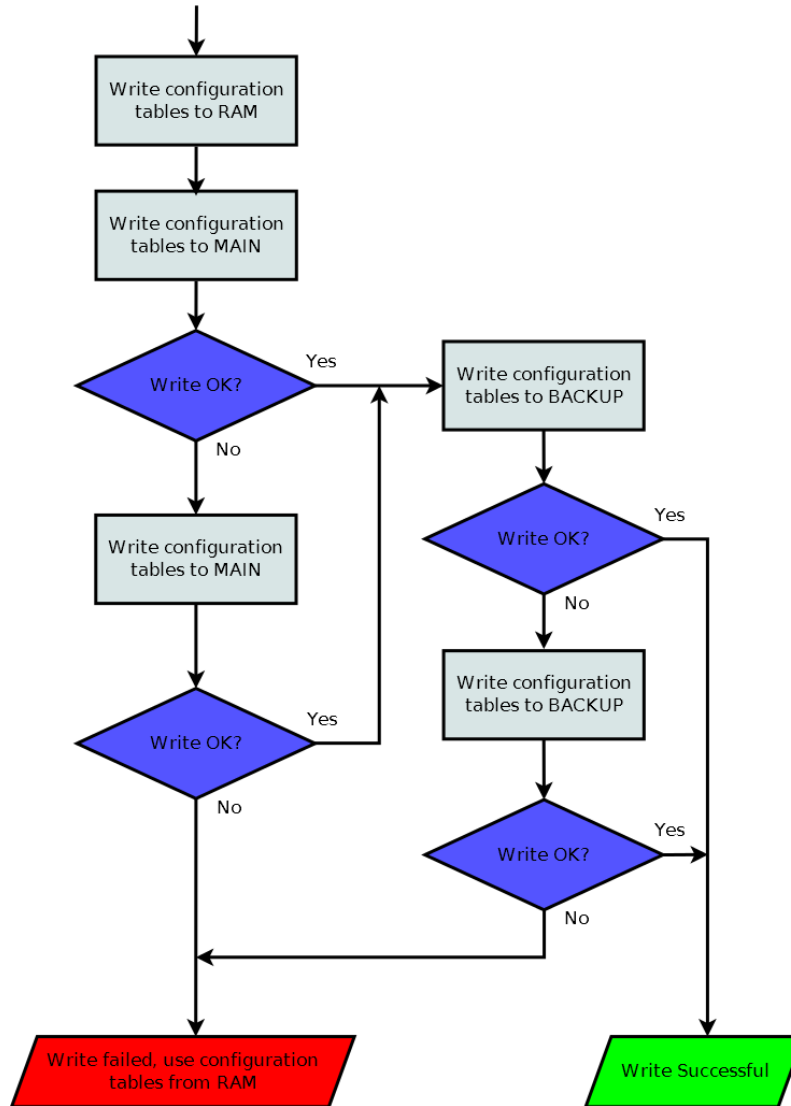


Figure 17: Configuration table update mechanism.

When a new configuration table is received, it is first stored into RAM. Then, the MAIN D-Flash sector for configuration tables is erased and the three configuration tables, including one updated and two old tables, are programmed to MAIN. After that, the newly written tables in MAIN are compared to the ones in the RAM to verify that writing the flash was successful. Next, the same process is repeated for the BACKUP D-Flash sector for configuration tables.

If the flash operation to update the MAIN entry of configuration tables fails, the erase/write process is repeated once. If the process fails again, the software simply relies on the tables stored in the RAM. The BACKUP sector is not modified in this

case to ensure that there is at least one intact, although out-dated, version of the configuration tables after the next reset.

A similar process is initiated, if the MAIN configuration tables are successfully updated, but the BACKUP update fails. First, the erase/write attempt is repeated. If the process fails again, the software relies on the MAIN entry of the tables, which is identical to the tables in RAM. The whole configuration table update process is visualized in Figure 17.

The MAIN and BACKUP D-Flash sectors are inspected after each start-up to determine their condition and to load an intact set of configuration tables into RAM. The last, non-data bytes of the MAIN and BACKUP sectors are checked to contain only zeros. If these bytes contain something else, it means that the last erase/program cycle failed and the data is corrupted. This is evident because after the D-Flash erase operation, the sector is filled with 0xFF bytes. After that check, the configuration table entries are compared to verify that they are identical. Lastly, the MAIN entry of configuration tables is loaded into RAM.

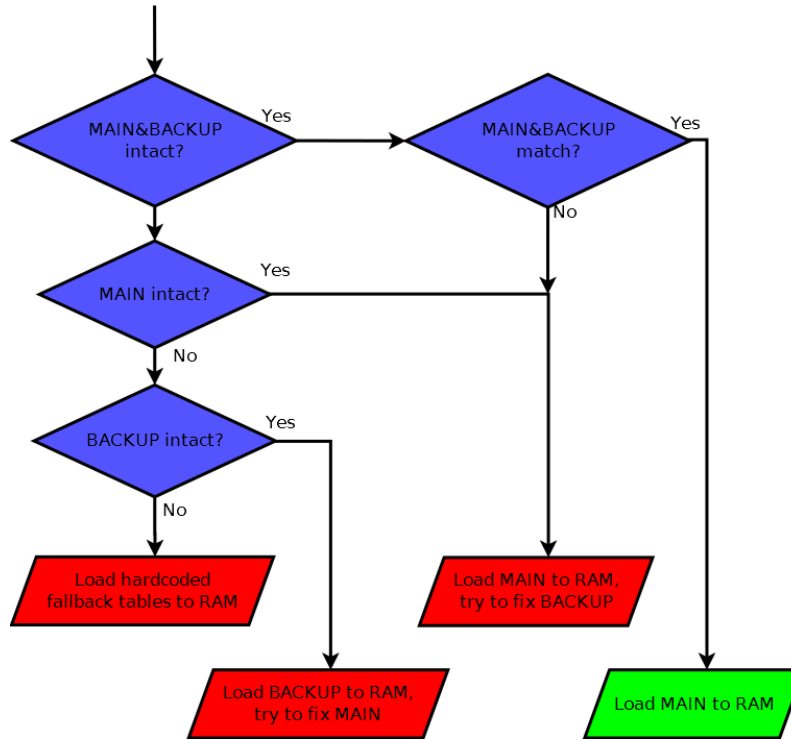


Figure 18: Configuration table initialization process following the controller start-up.

If one of the sectors does not pass the test for zeros, it is reprogrammed with the contents of the another sector. That is, if the last, non-data bytes of MAIN contain non-zero values, it is reprogrammed with the contents of BACKUP, and vice versa. If both of the sectors are corrupted, a subroutine is called to load a set of hard-coded fallback configuration tables from the P-Flash.

If the MAIN and BACKUP sectors are not identical, but they are intact, it is assumed that the MAIN sector is correct. In this case the BACKUP sector is

reprogrammed with the contents of MAIN. The configuration table initialization process is visualized in Figure 18.

5.2.2 Error correction plan

The flash modules on the Freescale MC9S12XEP100 microcontroller are implemented with *Error Correction Codes* (ECC). Using these codes, the MCU is able to automatically correct single bit faults (1 of 8 bits in a byte flipped) and detect double bit faults (2 of 8 bits in a byte flipped) during read operations. The physical flash memory is not modified by ECC. However, it is possible to attempt to fix these faults in flash by the user application.

The DREAMS-P/H instrument configuration tables are stored on microcontroller D-Flash as described in Section 5.2.1. Thus, the error correction codes detect and correct faults by default as the tables are read from the D-Flash to RAM. The redundancy of the D-Flash is increased by a software flash repair process.

When the MCU detects a single bit or a double bit fault, an interrupt is raised. The DREAMS-P/H software then retrieves the address where the error occurred and determines whether it is located at the configuration tables. If it is, first the read operation from D-Flash to RAM is completed. What happens after that is determined by the type of the bit fault.

In the case of a single bit error, the automatically corrected configuration table is loaded into RAM. The faulty sector housing the MAIN or BACKUP configuration tables is then simply erased and reprogrammed with the data in RAM.

In the case of a double bit error, it is possible to correct the D-Flash data by reprogramming the faulty sector with data from the fallback sector. If say, there is double bit fault in the BACKUP sector for configuration tables, the MAIN sector data can be used to reprogram the faulty sector. However, this operation is not performed, if the fallback sector is determined corrupted by the test for zeros described in 5.2.1.

The described software flash repair process is not used to repair faults in the P-Flash. This decision was made because of the great risk of damaging the application code in the case of an unexpected reset during the reprogramming of a P-Flash sector. These sectors are fairly large (1024 bytes each) and slow to reprogram, which increases the probability of an unexpected reset. It is also possible that such a reset is generated while the application tries to inadvertently access the sector being programmed.

After each ECC flash fault, an event packet is send to the CEU. The event packet describes the nature of the fault (single or double) and the result of the possible repair attempt. Also, the address where the error was detected is included in the packet. The following event packet types are used:

- Event packet type 3: Single bit error detected, but not fixed in flash
- Event packet type 4: Single bit error detected and fixed in flash
- Event packet type 5: Double bit error detected, but not fixed in flash

- Event packet type 6: Double bit error detected and fixed in flash with the help of a fallback sector

5.2.3 Watchdog

The asynchronous nature of embedded systems, such as the DREAMS-P/H controller, makes them susceptible to unexpected behavior in unexpected operational circumstances. For example, a specific combination of interrupts within a specific timing sequence could lead the application program to access an illegal address or stay in an endless loop. Accesses to illegal addresses cause a MCU reset by default, but if the software hangs due to e.g. an endless loop, it will normally remain in this state.

A watchdog, or a *Computer Operating Properly* (COP) watchdog as it is referred to by Freescale, is a simple solution to the aforementioned situation. The watchdog is a simple timer, which will generate a microcontroller reset, if the timer is not reinitialized within a predefined time. The timer is reset by the software by writing a hexadecimal byte `0x55` into the ARMCOP register, followed by a write of `0xAA` to the same register.

In DREAMS-P/H, the watchdog time-out is set to 2^{24} bus clock cycles, which equals to about 2.1 s, when using the 8 MHz bus frequency. The watchdog is started during the software initialization phase after the microcontroller start-up. When the application is running, the COP timer is reset whenever a telecommand is received or the end of main loop is reached. Also, whenever the MCU enters WAIT mode, the watchdog is disabled and reset. This is to prevent unnecessary MCU resets caused by the COP, during periods of microcontroller inactivity in the low power WAIT mode. When the MCU returns from the WAIT mode, the watchdog timer is reinitialized and started.

5.2.4 Software consistency check and redundancy

The DREAMS-P/H software consistency will be checked during every MCU start-up. If the check fails, the microcontroller will switch to a redundant software version in its memory. This is to increase the reliability of the software in harsh environments.

The software consistency check is performed in the very beginning of the microcontroller operation soon after accessing the reset vector, before the actual operational software is loaded from an image in the P-Flash. The check calculates the checksum of the contents of the software image and compares it to the checksum stored in the P-Flash together with the image. If the checksums match, the default software image is selected and its execution is started. If not, the same consistency check is performed for the next software image.

DREAMS-P/H will have multiple identical software images. These images are stored on separate MCU internal P-Flash memory areas. The currently used image can be deduced by the software start address information in the start-up event packet or in the status message.

The specific implementation of the software consistency check and redundancy is not yet certain. This is because the functionality is not implemented in the DREAMS-P/H qualification model, but it will be implemented in the flight model.

5.2.5 In-flight updates

In-flight updates can be used during the ExoMars EDM cruise phase to fix possible problems in the software found after the EDM integration to the launch vehicle. It can be also used for updates on ground after removing the debugger cable from the DREAMS-P/H proximity electronics board.

Like software consistency check and redundancy, the in-flight update functionality is not yet implemented on the DREAMS-P/H qualification model. However, the implementation is planned to exploit the redundant software images in P-Flash memory. The update will be always performed on a software image that is not in use at the moment. It is important that the program performing the software update is running in a write-protected memory location to prevent over-writing itself.

After the software update is complete, the MCU will be rebooted and the updated software image loaded. If the software from the updated image crashes, the old stable image will be loaded after the crash.

5.3 Software modules

5.3.1 Main program

The main program module can be divided into two parts: initialization and scheduling. The main program implementation principle is illustrated in Figure 19.

The software initialization part of the main program performs a series of operations to setup the controller for normal operation. First, all variables affecting the main program flow are initialized to their default values. Second, the PLL hardware module is started to produce 16 MHz core clock and 8 MHz bus clock frequencies from the 8 MHz crystal oscillator. Then, priorities are defined for interrupts and unused interrupt sources are disabled to prevent spurious interrupts. The XGATE co-processor is initialized for operation, thus enabling serial communications. Next, all pins are set to their default states and various peripheral modules configured for normal operation. An event packet is sent, signaling successful start-up of the instrument. Lastly, configuration tables are loaded from the microcontroller D-Flash to the RAM.

The main program scheduling part consists of interrupt service routines and main loop as seen in Figure 19. After initialization, the microcontroller has no task, so it goes through the switch statement and automatically enters the low-power WAIT mode. The main CPU stays in the WAIT mode, until it is interrupted by e.g. XGATE due to arriving telecommands as described in Section 5.3.2.

For simple and fast operations, like requests to read a configuration table, the telecommand handling ISR directly calls another subroutine to execute the operation. For more complex and slower operations, the software sets a task variable and

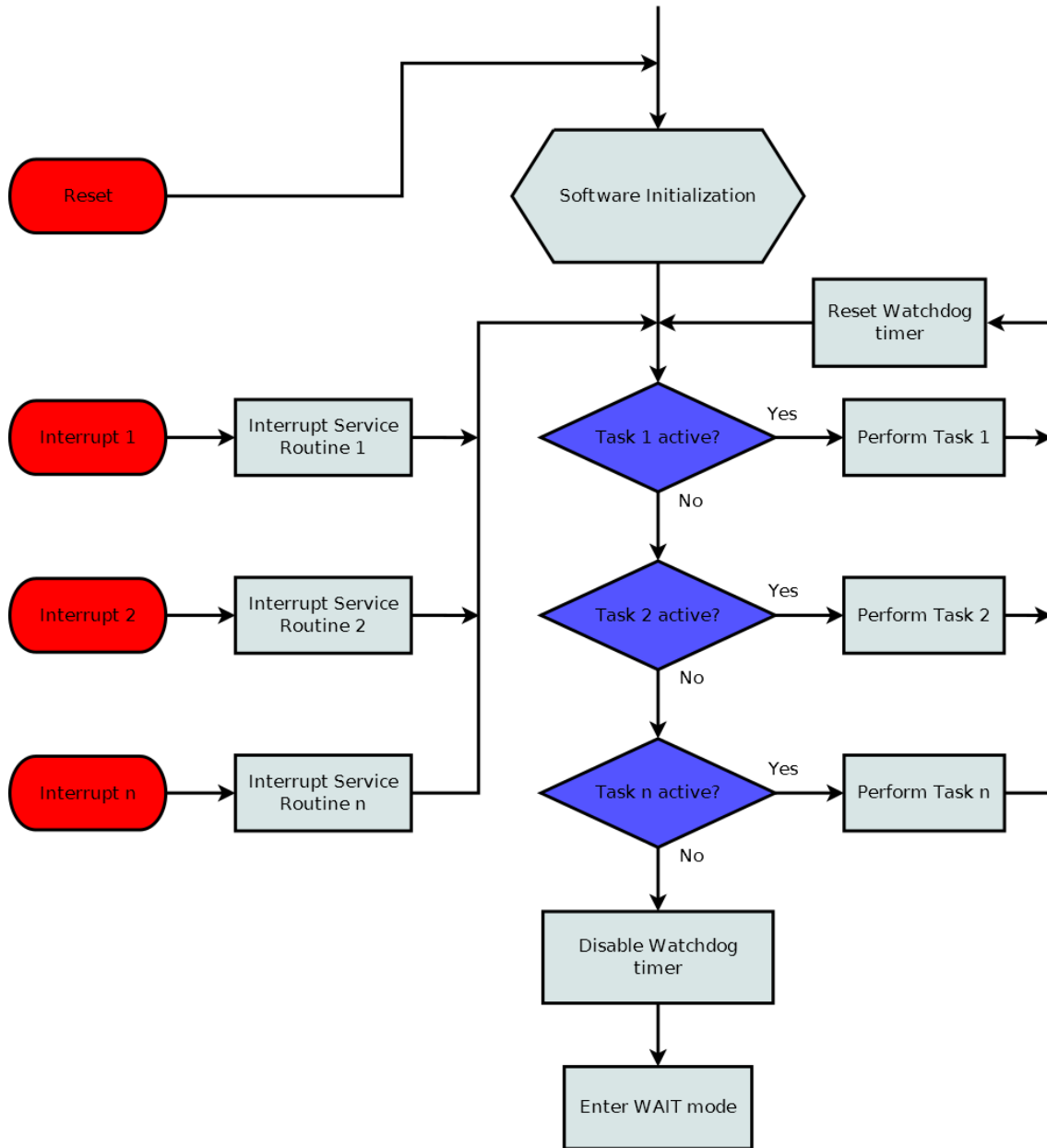


Figure 19: The principle of the highest level implementation of DREAMS-P/H software. Please note, that the interrupts can occur any time, and the software may also return to execution after an interrupt in other locations of the program. Only one task may be active at a time.

enters the main loop after the interrupt. The task can be for example to count measurement pulses from DREAMS-P1 or to schedule automatic measurements. After setting the task variable, the program alternately calls the task-appropriate subroutine and resets the watchdog. This is repeated until the task variable is changed or cleared. The aforementioned scheme enables the handling of other incoming main processor interrupts during main loop execution.

The main loop itself is a simple switch statement which inspects the task variable and then performs the operations relevant for the task. If there is no active task, the HCS12 enters WAIT mode.

5.3.2 Telecommand and telemetry

All lower level telecommand and telemetry operations are handled by the XGATE co-processor. This is done to release the main HCS12 processor for simultaneous operations, such as scientific measurements.

As the first telecommand byte arrives, the XGATE is interrupted out of idle mode. XGATE then checks the upper nibble of the first byte to match the synchronization bit pattern "1010" as defined for telecommands in Table 5. If the pattern is not found, a type 3 error message is sent. The same error message is not retransmitted as long as no correct sync patterns are detected. This prevents a continuous flow of error messages in the event of noise in the RS-422 transmission line.

As telecommand bytes arrive, the XGATE stores them into a RAM buffer shared with the main processor. During the reception, the SCI hardware module monitors the serial line for incoming bytes. If more bytes were expected, but the start bit of the next command byte does not arrive in 10 communication frequency cycles, that is 0.1 ms, the reception is interrupted and a type 1 error message is generated.

When the whole telecommand is received as specified by the length parameter, the XGATE enters an subroutine called *errorCheck()*. This subroutine checks all error conditions not already inspected. The received checksum is compared to the checksum calculated during command reception. The validity of the address field is inspected in general and in combination with the type of command received. Next, the command byte is checked to ensure that the command is defined. Also, the length parameter and extra parameters are checked to match the values defined for the command at hand in the Software specification [23]. DREAMS-P/H operational status is also inspected. Only commands "STATUS", "STANDBY" and "REBOOT" are allowed when the system is performing measurements or DREAMS-H regeneration. Lastly it is checked, if needed, that preparatory commands were received earlier, enabling the operation requested by the command being handled. If any of the aforementioned checks fails, an error message is produced according to the definitions in Table 12 and the command ignored.

An ACK message, or an error message with error code 0 is transmitted after the *errorCheck()* subroutine has verified that the received telecommand is valid. What happens next, depends on the type of command being executed. If the command is "STATUS", the XGATE fetches the controller status information and sends it out after the ACK message. In other cases, the XGATE interrupts the main processor, which then takes over the command handling.

When the main processor has assembled data, which it would like to send as telemetry, the CPU sets an software telemetry reservation flag which represents the type of telemetry to be sent, and interrupts the XGATE. If there are no ongoing transmissions, the co-processor starts immediately transmitting the contents of the telemetry buffer specified with the telemetry reservation flag. In the contrary, if

Table 12: Telemetry error types.

| Error code | Description |
|------------|--|
| 0 | No error. Command accepted |
| 1 | Reception time-out |
| 2 | Incorrect checksum |
| 3 | Illegal byte 0 (sync pattern or address) |
| 4 | Undefined command or length parameter not as specified |
| 5 | Context error |
| 6 | Incomplete command sequence |

there is an unfinished telemetry packet being transmitted, this packet will be first finished. Next, the XGATE starts transmitting the next pending telemetry packet with the highest priority as per Table 13.

Table 13: Telemetry buffer reservation flag priorities.

| Priority level | Buffered data |
|----------------|-----------------------------|
| 1 | Acknowledgment message |
| 2 | Status message |
| 3 | Illegal byte 0 error |
| 4 | Reception time-out error |
| 5 | Event package |
| 6 | Regeneration report |
| 7 | Configuration table dump |
| 8 | Memory dump |
| 9 | Memory verification message |
| 10 | DREAMS-P data package |
| 11 | DREAMS-H data package |

Command acknowledgement and various error messages are given the highest telemetry transmission priorities. This is to ensure that DREAMS CEU receives the information of the effects of a telecommand on DREAMS-P/H as soon as possible. This also helps debugging.

There are 8 types of different telemetry packets. Every command, except "REBOOT", produces first an acknowledgment package. This package may be followed by another telemetry packet produced by the telecommand. In addition, the system produces autonomously event packages describing operations not initiated by telecommands. These event packages include boot messages and error correction messages, as described in Section 5.2.2. The various packet types are described in Table 14.

Table 14: Telemetry package types.

| Type | Originator address | Description |
|-----------|------------------------|---------------------------------|
| 0x00–0x1F | same as in telecommand | Acknowledgment or error message |
| 0x20 | 0xA0 | Instrument status |
| 0x21 | 0xA0 | Event package |
| 0x22 | 0xA0 | Memory verification message |
| 0x23 | 0xA0 | Memory verification dump |
| 0x40 | 0xA1, 0xA2, 0xA4 | Detector data package |
| 0x41 | 0xA1, 0xA2, 0xA4 | Configuration table dump |
| 0x42 | 0xA4 | Regeneration data |

5.3.3 Instrument control

The instrument control software module is responsible for the following tasks:

- Power switch operations for DREAMS-P1, DREAMS-P2 and DREAMS-H
- RESET and STEP control signal operations for DREAMS-P1, DREAMS-P2 and DREAMS-H
- Measurement scheduling

The power switch operations affect transducer supply voltage and control signal levels. When a "POWER_ON" command is issued, the software drives high a pin controlling the +12 V supply voltage transistor switches. Immediately after this, the transducer STEP and RESET control signals are driven high to their nominal +5 V voltage. "POWER_OFF" and "STANDBY" commands will cause the opposite – drive down the control signals and switch off the supply.

RESET and STEP control signals are issued before and during test and automatic measurement sequences. Each sequence is started by issuing a RESET signal. This is done to ensure that the first channel of the transducer is selected and thus prevent corrupted measurement data. Channels from two to eight can be then accessed by issuing STEP pulses. RESET and STEP signals are implemented as 3 ms low-state pulses on the respective +5 V control lines as seen in Figure 20.

The scheduling of measurements depends on the type of measurements performed: test measurements or automatic measurements. There is also the special case of automatic regeneration measurements discussed in Section 5.3.4. However, all types of measurement sequences are initiated with a RESET signal and the following channels are selected with STEP signals. If channels are to be skipped, a delay of 3 ms is left between each consecutive STEP signal.

Test measurement scheduling is defined with the parameters of the "TEST_SENSOR_CONFIG" command. These parameters define the channels to be measured, pulses to be counted on these channels and a single time delay period between selecting a channel with a RESET or a STEP signal and start of the pulse counting.

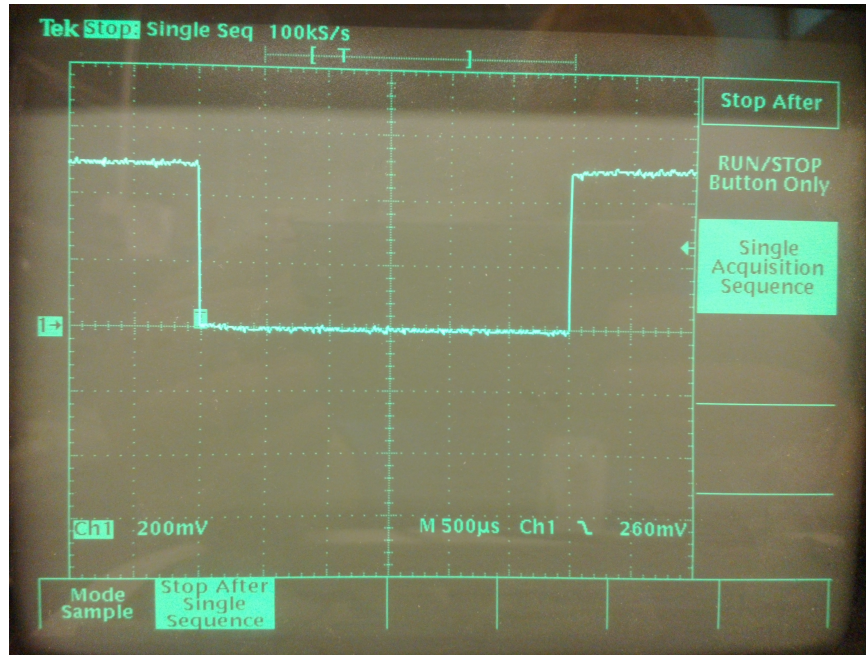


Figure 20: 3 ms high-low-high RESET pulse measured with an oscilloscope.

The measured data is stored to the respective data buffer and sent out after the last measured channel.

Automatic measurement scheduling is started by initializing the inter-sequence timer. The configuration tables define the inter-sequence time-out, as well as, number of sequence repetitions, measured channels, time delays before measurement of each channel and pulses to be counted. The amount of pulses measured and delays before each channel measurement are defined individually for each channel.

The first automatic sequence begins after the starting of the inter-sequence timer. Following the completion of the sequence, data is sent out in the same format as with the test measurement sequence. The MCU goes then to WAIT mode. The inter-sequence timer interrupts the MCU every 10 ms to increment the timer and to check if the next measurement sequence should be started. The next sequence is started, as the timer reaches the value defined in the configuration table.

If the inter-sequence time-out is reached during a measurement sequence, this will stop the measurement after pulse counting is finished on the active channel. The measured data is then sent out with unmeasured channel data replaced with zeros. This scenario causes inter-sequence time-out to stretch since measurement is only stopped after the completion of measurement on the active channel.

5.3.4 Heater control

The heater control module executes the regeneration of the Humicap® humidity sensors. The regeneration parameters, voltage and duration, are defined with the "REGENERATE" command. The command can also request that the voltage is selected automatically depending on the DREAMS-H temperature. The regeneration

hardware is described in Section 4.3.1

In the beginning of the regeneration operation, instrument status flags are changed to indicate the ongoing regeneration. Then, in the case of manual regeneration, commanded heater voltage is switched on. The heater voltage is selected by adjusting the resistance on the heater power supply reference pin. Immediately after turning on the heaters, the regeneration timer is also initialized and started. As the timer reaches the set time limit, regeneration is stopped and a regeneration data telemetry packet is sent out.

If commanded, the automatic regeneration measurement is executed to determine the approximate temperature of the DREAMS-H Humicap® sensors before selecting a voltage for regeneration. Command "REGENERATE" with "0" as the voltage parameter starts the measurement sequence. First, the DREAMS-H transducer is powered up. The MCU then waits for 18 ms, sends a RESET signal and waits another 2000 ms for the DREAMS-H frequency signal to stabilize. Following that, the DREAMS-H channel 6 occupied with the Thermocap® sensor is selected by issuing STEP signals. 500 pulses are counted and the measurement time is compared to the limits defined in DREAMS-H configuration table. If the time is outside the allowed range defined by the values representing maximum and minimum heating temperatures, the regeneration is aborted. This is indicated by a regeneration telemetry packet with "0" as the voltage option and "0" as duration. If the measurement time is inside the allowed range, the regeneration voltage is selected using the low/normal and normal/high temperature limits in the configuration table.

5.3.5 Pulse counting

Pulse counting is the method for extracting the scientific information from the pressure and humidity transducers. Thus, the development and testing of the pulse counting software module has been a central goal for this thesis. The simplified source code for pulse measurements is attached in Annex A.

The pulse counting is implemented using the ECT module of the MC9S12XEP100 microcontroller. During pulse measurements, ECT is configured to interrupt the MCU every time a rising edge is detected on the measured transducer frequency output. The timestamp of this edge is determined by a free-running timer operating at the 8 MHz bus frequency. ECT stores this value into one of the input capture registers. When the timer overflows by passing its maximum value of 65535, it produces an interrupt and resumes counting up from zero. With an 8 MHz bus frequency, this happens every 8.2 ms. Practically, this means that the timer will overflow multiple times during most pulse counting operations with typical measurement frequencies from about 1 kHz to 16 kHz.

As the first rising edge enters the MCU, the responsible interrupt service routine copies the timestamp from the input capture register into a variable called *firstamp*. The software timer overflow flag is also cleared to ensure no overflows received before the first rising edge are included. The following rising edge timestamps will be stored into a variable called *lastamp*. The timestamp in *lastamp* is overwritten every time an active edge arrives. Simultaneously, the variable *count* is incremented to keep

track of the amount of edges already counted. To calculate the duration of n pulses, the process has to capture $n + 1$ rising edges.

During the pulse measurement, the main program repeatedly calls the *pulsecounter()* subroutine, which simply checks if the number of pulses to be counted is reached. If that is not the case, *pulsecounter()* checks also if a timer overflow has occurred, and increments variable *overflow_count* if needed.

When the desired number of pulses is reached, the *pulsecounter()* disables further interrupts produced by incoming pulses and timer overflows. The subroutine then checks if there was an unregistered timer overflow just before the last rising edge. If this seems to be the case, *overflow_count* is incremented only if the last edge timestamp is small enough. This is to ensure that no overflows are counted which have happened just after the last rising edge. The measurement result *meastime* is then calculated using Equation 3.

$$meastime = 65536 * overflow_count + lastamp - firstamp \quad (3)$$

The implemented pulse counting process is limited by the time it takes to handle the ECT rising edge and timer overflow interrupts. When an interrupt is triggered, the HCS12 core must first store its current state, before it moves on to the interrupt service routine. The time it takes to do this is known as the *context switching time*. It takes a fixed amount of bus clock cycles to perform the context switching and interrupt handling of a rising edge detection interrupt and a subsequent timer overflow interrupt. If the measurement frequency is increased enough, the period of the measurement signal in terms of bus clock cycles decreases. At some point the period becomes shorter than the number of cycles it takes to handle subsequent interrupts, and the measurement accuracy is compromised. In DREAMS-P/H, this problem is not expected since the difference of measurement frequency (< 20 kHz) and bus clock frequency (8 MHz) is fairly large. However, the system is carefully tested bearing this feature in mind.

Together with the pulse measurement operation described above, there is a simultaneous parallel process, which effectively acts as an 1 kHz software high-pass filter. The purpose of this process is to skip the measurement of a channel, if it is broken and does not produce a nominal frequency. In practice, a PIT timer with a time-out period of 1 ms is started when a measurement sequence is begun. The timer is reset every time a rising edge is detected. However, if no edge is detected in time, the pulse measurement is stopped and *meastime* is filled with zeros.

5.3.6 Memory management and manipulation

Tasks for the memory management and manipulation software module include:

- In-flight software updates
- Software consistency check and redundancy
- Robust configuration table storage

– Manipulation of memory locations in RAM and flash memory

In-flight updates, configuration tables, software checks and redundancy are discussed in Section 5.2. Hence only the manipulation of various memory locations is presented here.

Manipulation operations of the MCU memory include read accesses to all memory addresses and write accesses to RAM and D-Flash addresses. Read operations are initiated with the "MEMORY_DUMP" command, while the write process is intended to be executed as a three-step process using commands "MEMORY_LOAD", "MEMORY_VER" and "MEMORY_UPD".

The parameters of the "MEMORY_DUMP" command define the start address and number of bytes (maximum 250) of the memory dump operation. The controller simply fetches the data from the defined global addresses, formats it into a telemetry packet and sends it to the CEU.

Memory write operations are started by issuing the "MEMORY_LOAD" command with the global destination start address and the data. The highest bit in the destination parameter of the command defines if the destination is located in D-Flash (1) or RAM (0). The status of this bit will affect the way how the data is programmed into the memory. The actual destination address is defined by the 23 lower bits. After receiving the command, the MCU stores the address, data and their XOR checksum into a RAM buffer.

The "MEMORY_VER" command is then intended to be issued to verify the success of the "MEMORY_LOAD" telecommand. "MEMORY_VER" checks that the destination address received with the command and the one in the memory load buffer match, and then returns the address and data from the buffer. However, this step in the memory update process can be omitted. It is only intended for verifying the data in the buffer before the actual update.

Lastly, the "MEMORY_UPD" command is issued to execute the actual update. The command first checks that the address received with "MEMORY_UPD" is identical to the one in the memory load buffer. If match is not found, an error message of type 5 is sent out. Next, a new checksum is calculated using the destination address and data in the memory load buffer. This checksum is compared to the checksum stored in the memory load buffer. If the checksums match, the update is executed. If the checksums do not match, the update is aborted and an error message is sent out. In the case of a memory update in the RAM area, the successful command simply replaces the old data using a loop, two bytes at a time.

The "MEMORY_UPD" operation is a bit more complex, if the memory update is performed in the D-Flash area. This is because, to modify a byte in D-Flash, the whole 256 byte sector containing the byte has to be erased first. Hence, the D-Flash memory update is started by copying the sector to be modified into RAM. The data is then modified in RAM accordingly to the "MEMORY_UPD" command. Lastly, the D-Flash sector is erased and written with the new data from RAM.

The memory update process is implemented with only few error checks and assumes that the user is aware of the limitations of the operation. This means that the updates must start from an even address and only even number of bytes

are allowed in a update. Furthermore, D-Flash updates may not cross flash sector borders. These limitations are imposed to make update process implementation as robust as possible.

6 Test procedures and test results

6.1 Test philosophy

The DREAMS-P/H controller software and system functionality is tested on three different levels of fidelity: Verification test, Full Functional Test and Abbreviated Functional Test. These test levels are summarized in Table 15.

Table 15: Test parts included in different test procedure levels.

| Test part | Verification Test | Full Functional Test | Abbreviated Functional Test |
|--|-------------------|----------------------|-----------------------------|
| Controller start-up | Yes | Yes | Yes |
| Power commands | Yes | Yes | Yes |
| DREAMS-H regeneration | Yes | Yes | Yes |
| Test measurement | Yes | Yes | No |
| Configuration table handling | Yes | Yes | No |
| Automatic measurement | Yes | Yes | Yes |
| Memory control | Yes | Yes | No |
| Telecommand and telemetry contents | Yes | Yes | Yes |
| Detailed verification of telecommand and telemetry | Yes | No | No |
| Power consumption | Yes | No | No |

The purpose of the verification test is to ensure that the implemented software works as specified and designed, including all nominal and anomalous situations. Key points of the controller program execution flow are monitored during the test with the P&E Multilink USB debugger interface. In addition, the RS-422 interface and instrument control signals are monitored simultaneously. The verification test procedure for DREAMS-P/H qualification model is presented on a general level in Section 6.3, with the test results in Section 6.4. The whole stepwise QM verification procedure is available in Annex B.

The *Full Functional Test* (FFT) is done before the instrument delivery and at least once after its integration to the whole DREAMS package. Software execution flow is not monitored anymore in this test, nor would it be possible since the debugging interface is removed as the instrument is assembled. The purpose of the test is to verify that the instrument works correctly in all nominal use cases.

The *Abbreviated Functional Test* (AFT) is performed to verify the instrument operation after all integration steps and modifications. AFT is also performed during or after all environmental tests. The AFT is a shorter version of the FFT and tests only the most critical functions. Hence, test measurement, configuration table or

memory control commands are not tested in the abbreviated functional test, unlike in the full functional test.

In addition to the three tests presented above, the DREAMS-P/H controller can be bypassed with a test access cable for instrument calibration tests. Thus, pressure and humidity sensors can be directly interfaced with the calibration equipment while the controller is powered off.

6.2 Verification test setup

Following pieces of ground support equipment are used for the software tests:

- Two adjustable DC voltage supplies OR one dual output voltage supply for +5 V and +12 V
- Two multimeters
- RS-422/USB interface
- RS-422/power cable
- PC with DREAMS EGSE software
- P&E USB Multilink debugging interface
- PC with CodeWarrior IDE software
- Stable oscillator

Figure 21 describes the normal configuration of the ground support equipment during the software verification test. First of all, DREAMS-P and DREAMS-H are connected to each other. Power is provided by the DC voltage supplies adjusted to +5 V and +12 V through the RS-422/power cable harness. The PC running DREAMS EGSE software is used to simulate the DREAMS CEU. This PC is connected to the other end of the RS-422/power cable via an RS-422/USB interface box. The software execution is simultaneously monitored with a separate PC running the Freescale CodeWarrior IDE and debugging software. This PC is interfaced to the MCU on the DREAMS-P board with the P&E USB Multilink debugger interface.

The operation of supporting test software is discussed in Section 4.5. For Freescale CodeWarrior IDE software, see Section 4.5.2. Likewise, for the FMI-developed DREAMS EGSE software, see Section 4.5.3.

In addition to the normal test setup presented in Figure 21, additional equipment is used for some parts of the test. Multimeters are used to measure control signal levels and measure instrument current consumption. A stable oscillator is used to verify the measurement concept and accuracy. Both of these devices can be seen in Figure 22. Additionally, an oscilloscope, seen in Figure 23, is used to measure control signal timings.

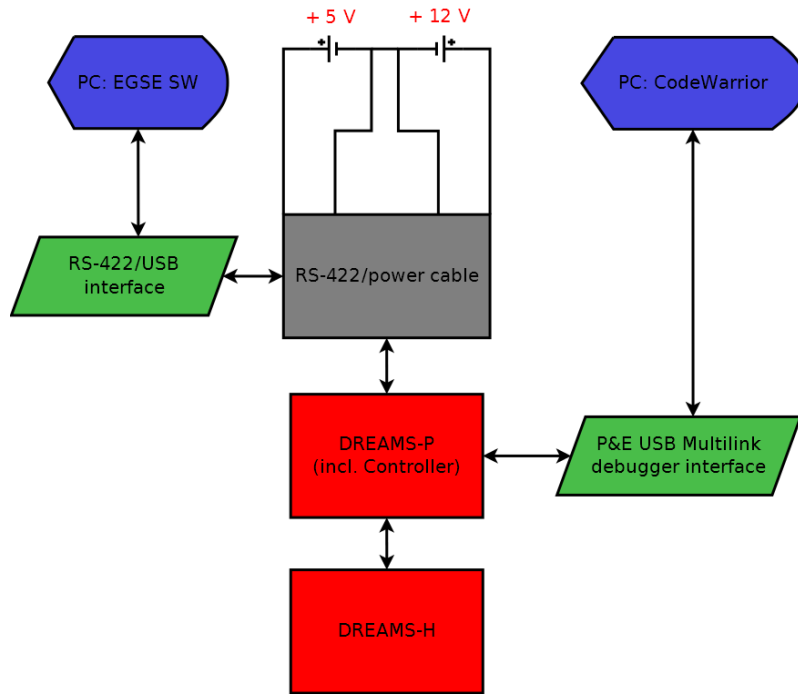


Figure 21: Basic structure of the test setup during the software verification test.

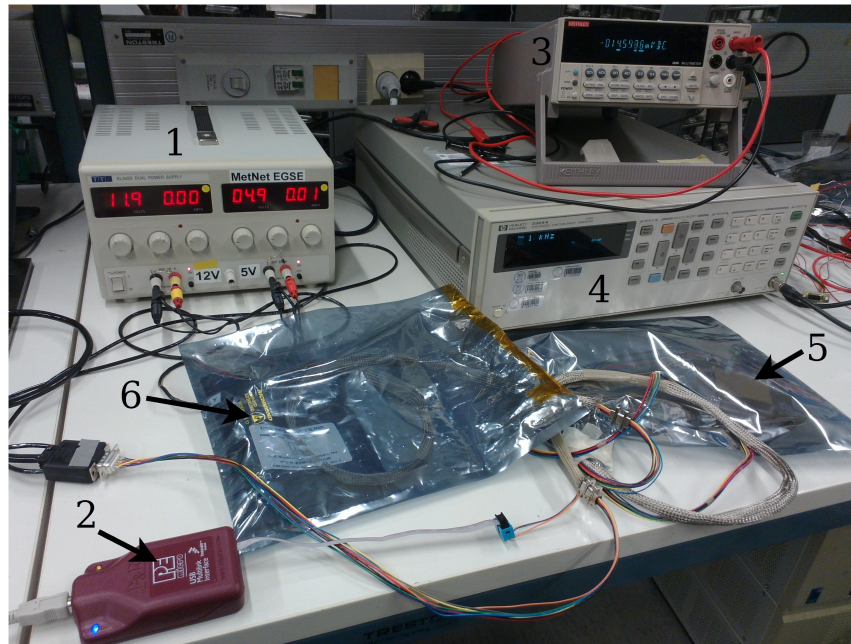


Figure 22: The physical test setup: Dual power supply(1) on the upper left, the purple P&E USB Multilink debugger interface(2) on the lower left, multimeter(3) on the upper right, with the stable oscillator(4) below, DREAMS-P(5) and DREAMS-H(6) in the front, packed into antistatic bags to maintain cleanliness requirements.

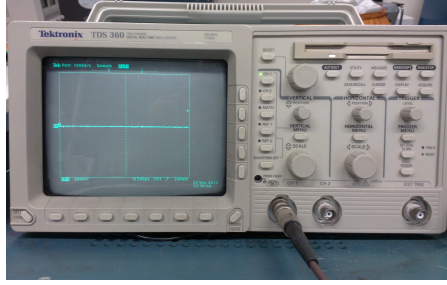


Figure 23: The oscilloscope used for control signal timing tests.

6.3 Verification test procedure

6.3.1 Controller start-up

The normal start-up sequence is verified in the controller start-up test. The main purpose of the test is to verify that as soon as the MCU is powered up, it will start executing initialization tasks, send a start-up event package, enter the main loop and go to a low-power WAIT mode to stand by for incoming commands.

After a normal start-up is verified by the reception of a event packet, the watchdog function is tested. The watchdog is triggered in the test by halting the software execution by the debugger interface. This prevents restarting the COP time-out period, thus generating a watchdog reset, which is verified by the reception of an event package.

6.3.2 Power commands

The correct operation of the power commands is especially important, since incorrect operation of the power switches might damage the DREAMS-P/H instrument or systems interfaced with it. The power commands test verifies the following properties:

- Transducer power, STEP and RESET lines are at 0 V, when the transducer is powered off
- Transducer power, STEP and RESET lines are at +5 V, when the transducer is powered on
- System status telemetry byte 3 reflects correctly the state of the power switches
- Only single transducers, or the combination of DREAMS-P1 and DREAMS-H or the combination of DREAMS-P2 and DREAMS-H are allowed on simultaneously

The voltage level requirements are tested by disconnecting DREAMS-H from the DREAMS-P board, and measuring the respective DREAMS-H power and control lines from the connector pins. DREAMS-P1 and DREAMS-P2 lines cannot be directly measured, but their operation is verified indirectly by comparing the

"POWER_ON" and "POWER_OFF" command handling program code of each of the transducers as well as current consumption measurements.

System status is requested after each power command. All possible "POWER_ON", "POWER_OFF" and "STANDBY" command combinations for each transducer and transducer combination are tested to see that the byte 3 in the status telemetry packet always represents the current state of the power switches. The same byte is used to verify that illegal power switch combinations do not occur.

6.3.3 DREAMS-H regeneration

Like the power commands, also the "REGENERATE" command could cause hardware damage if operating incorrectly. Furthermore, even correct regeneration command operation could damage DREAMS-H Humicap® sensors. Thus, the "REGENERATE" command shall never be given in room temperature, while the DREAMS-H is connected.

The DREAMS-H regeneration test verifies the following properties:

- "REGENERATE" command with voltage setting "1" produces a voltage of 4.7 V across the DREAMS-H heater resistors
- "REGENERATE" command with voltage setting "2" produces a voltage of 5.0 V across the DREAMS-H heater resistors
- "REGENERATE" command with voltage setting "3" produces a voltage of 5.3 V across the DREAMS-H heater resistors
- "REGENERATE" command with voltage setting "0" performs a temperature measurement and automatically selects the regeneration voltage
- The DREAMS-H transducer and DREAMS-H heater cannot be powered on simultaneously
- The actual regeneration time corresponds to the duration parameter in the "REGENERATE" command

The regeneration voltage is measured from the connector pins in the disconnected DREAMS-H pigtail connector on the DREAMS-P board, similarly as with the power command test above. DREAMS-H power line is monitored simultaneously to ensure that the transducer and the heater are never powered on at the same time. The duration of regeneration operations is determined with a simple stopwatch.

6.3.4 Single test measurement commands

This part of the test procedure verifies the correct operation of the simplified measurements using test measurement commands "TEST_SENSOR_CONFIG" and "TEST_SENSOR_MEAS". The single test measurement commands test verifies the following properties:

- Each transducer produces scientifically valid data

- The test measurement procedure does not introduce random errors to the data
- The pulse counting reference clock frequency is stable

Multiple test measurements are performed with each transducer to prove that the pulse counting software works in principle and does not produce random errors. After this, the gathered measurement data can be compared to the measurements with the proven FMI calibration laboratory equipment. DREAMS-H can be measured directly with the old calibration system, whereas test access pads on the proximity electronics board can be used to access DREAMS-P transducers directly. More detailed analysis on the pulse counting system is executed after the automatic measurement tests presented in 6.3.6

A stable oscillator is used to determine the reference clock frequency and its stability. First, the DREAMS-H is disconnected and the signal output of the stable oscillator is connected between the DREAMS-H frequency and ground lines. Then the stable oscillator frequency is varied, while sending "TEST_SENSOR_MEAS" commands addressing the DREAMS-H transducer. Lastly, the pulse measurement results are used to determine the microcontroller bus clock frequency f_{BUS} using Equation 4:

$$f_{BUS} = \frac{f_{OSC} \cdot c}{n}, \quad (4)$$

where f_{OSC} is the frequency of the stable oscillator, c is the measurement duration in bus clock cycles and n is the number of transducer pulses counted. The theoretical uncertainty of these measurements is ± 2 bus clock cycles. This is because the first and last rising edges of a measurement are each defined with a theoretical uncertainty of ± 1 bus clock cycles. Using this information, the uncertainty of the bus clock frequency is calculated. In order to provide first order proof of MCU bus clock stability, every measurement with different stable oscillator frequency setting must give, within the determined uncertainty, the same calculated bus clock frequency.

6.3.5 Configuration table handling

Configuration table operations are necessary, when automatic measurement commands are used. The configuration table handling test verifies the following properties:

- "READ_CONF" command returns correctly the addressed configuration table data
- "UPDATE_CONF" command updates the addressed configuration table data in RAM
- "UPDATE_CONF" command updates the addressed configuration table data in MAIN and BACKUP D-Flash sectors
- Configuration table information is preserved while the microcontroller is powered off and retrieved from D-Flash memory after a reset

In this test, the data in the RAM and D-Flash configuration table structures is monitored with the debugger. "READ_CONF" command is tested by simply issuing the command and verifying that the received configuration table is identical to the addressed table in RAM, as well as, in the MAIN and BACKUP sectors of D-Flash. These memory sectors are further discussed in Section 5.2.1.

"UPDATE_CONF" command is tested by issuing the command with a new configuration table and verifying that the tables in the aforementioned locations are updated. The controller is rebooted after the update to verify that the new table data is retained during a reset. These tests are repeated for each of the three the configuration tables (DREAMS-P1, DREAMS-P2 and DREAMS-H).

6.3.6 Automatic measurement commands

The DREAMS-P/H controller will be engaged in automatic measurements for the majority of the time it is powered on. Also, these measurements produce the scientific data, which is the prime purpose of the system. Thus, stability of the software during automatic measurements is of high importance. The automatic measurement commands test verifies the following properties:

- Automatic measurement sequence interval is as defined in the respective configuration table
- The instrument status message correctly describes the state of the automatic measurement in status bytes 4 and 8.
- Each transducer produces scientifically valid data
- The automatic measurement procedure does not introduce random errors to the data

A stopwatch is used to determine the interval between the automatic measurement sequences. Number of sequence repetitions e.g. 10 is defined within a configuration table and the measurement is started by issuing the "MEAS" command. The stopwatch is then started when the first data packet arrives and stopped when the last data packet arrives. The interval between individual sequences can be calculated by dividing the measured time with the number of sequence repetitions minus one.

The instrument status is polled during the automatic measurements to verify that the status reflects the actual state of the instrument. This protects the measurement process by preventing the execution of all other commands than "STANDBY", "REBOOT" and "STATUS".

Longer automatic measurement series in different pressures, temperatures and humidities are performed to verify the quality of the obtained measurement data. The automatic process is further tested by replacing the DREAMS-H instrument with the stable oscillator, similarly as in Section 6.3.4. Long measurement series of different stable frequencies should reveal any rare random errors in the pulse counting software.

6.3.7 Memory control commands

The memory control commands can be used for debugging purposes and simple small updates to the MCU memory. This part of the test verifies the following properties:

- "MEMORY_DUMP" command can be used to read microcontroller memory locations from RAM, D-Flash and P-Flash
- "MEMORY_LOAD" command prepares for an "MEMORY_UPD" command by storing the uploaded destination address and data into a buffer
- "MEMORY_VER" command returns the destination address and data uploaded with an earlier "MEMORY_LOAD" command
- "MEMORY_UPD" command executes the update using the destination address and data uploaded with an earlier "MEMORY_LOAD" command

The debugger interface and software is used to verify the correct execution of all memory commands. The "MEMORY_DUMP" command is tested by monitoring various memory locations in the MCU address map, while requesting memory dumps from these locations. RAM, D-Flash and P-Flash areas are tested to ensure that changing the memory type does not change the behavior of the "MEMORY_DUMP" command.

"MEMORY_LOAD", "MEMORY_VER" and "MEMORY_UPD" are tightly interrelated commands, and hence they are tested in a sequence. First, the "MEMORY_LOAD" command is issued. The destination address is inspected after the command to ensure that the update was not executed yet. Secondly, "MEMORY_VER" is sent and the corresponding telemetry packet is inspected to verify that the data sent earlier to the memory load buffer is returned correctly. Lastly, "MEMORY_UPD" is issued and the updated data is observed in the destination address. This sequence is preceded and followed by "MEMORY_DUMP" commands to see the changing memory data also in the telemetry. Memory locations in RAM and D-Flash are used to verify the described command sequence. P-Flash cannot be modified since the process could make the software unstable.

6.3.8 Telecommand and telemetry handling

Telemetry provided by the DREAMS-P/H system is used by the DREAMS CEU to determine the instrument status, deliver measurement data and thus help control DREAMS-P/H. Vice versa, telecommands sent by the CEU affect the operation of DREAMS-P/H. This data interface is critical, since it is shared between two organizations in the ExoMars EDM project: Finnish Meteorological Institute on the DREAMS-P/H side and Thales Alenia Space on the DREAMS CEU side. Hence, the telemetry and telecommand software implementation must accurately follow the software specification [23] and requirements [21] documents. Specifically, the following properties are verified in this part of the test:

- DREAMS-P/H detects all defined error condition types in incoming telecommands and responds with an appropriate error message
- Faulty telecommands do not trigger unexpected operations or interfere with the normal execution of the DREAMS-P/H software

The telecommand error conditions are tested by sending various erroneous commands representing all different types of errors discussed in Section 5.3.2. The corresponding error messages are analyzed to determine conformity to software specification. Instrument telemetry flow is monitored, and status is requested and inspected after every erroneous command. This is to ensure that unexpected operations were not triggered.

6.3.9 Power consumption

As the DREAMS instrument suite is fully battery powered, its operational lifetime depends on the power consumption of DREAMS CEU and all of the payloads. Thus, the power consumed by DREAMS-P/H instrument affects directly the operations planning of the whole instrument package. The power consumption test determines the +5 V and +12 V power consumption during following operational modes:

- Controller idle
- DREAMS-P1 measurement and data transmission
- DREAMS-P2 measurement and data transmission
- DREAMS-H measurement and data transmission
- Combined DREAMS-P1 and DREAMS-H measurement and data transmission
- Combined DREAMS-P2 and DREAMS-H measurement and data transmission

The power measurements are carried out by connecting a multimeter acting as an ammeter in series with the laboratory power supply and DREAMS-P/H instrument. The measurements are first performed for the +5 V supply, followed by the +12 V supply.

6.4 Verification test results

6.4.1 Controller start-up

After powering the instrument on, the DREAMS-P/H instrument generated an event packet signaling a successful start-up sequence. Following this, the software on the MCU was updated using the debugger interface. This was done to ensure that the most recent version is uploaded to the microcontroller. Then, the software version

was verified to be up-to-date by issuing a "MEMORY_DUMP" command fetching the correct version number from global address *0x100200*.

Next, the watchdog time-out was deliberately triggered by halting the program execution with the debugger software. This resulted in the reset of the MCU and a new event packet approximately 2 s after halting the program, as expected.

6.4.2 Power commands

For testing the power commands, the DREAMS-H instrument was detached from the DREAMS-P board and the pins of the DREAMS-H connector were accessed with a multimeter. Measurements with the multimeter verified that the voltages on power, STEP and RESET lines reacted to "POWER_ON" and "POWER_OFF" commands as specified.

After the direct voltage measurements, DREAMS-H was reconnected to the controller on the DREAMS-P board. Then, all possible combinations of subsequent "POWER_ON", "POWER_OFF" and "STANDBY" commands were tested. The "STATUS" command was issued to poll the instrument status after each command combination. The controller followed the power switch combination requirements, and correct status was observed in every status message.

6.4.3 DREAMS-H regeneration

During the regeneration test, the DREAMS-H was disconnected and voltages on the regeneration and power pins were simultaneously monitored with two multimeters. Manual regeneration was performed with the "REGENERATE" command, using the three different available voltage levels. Table 16 presents the measured regeneration voltages and the voltages expected based on the calculations in Section 4.3.1. Each of the voltage settings produced a slightly lower voltage than expected.

Table 16: Measured DREAMS-H regeneration voltages.

| Voltage setting | Measured voltage | Expected voltage |
|-----------------|------------------|------------------|
| 1 | 4.65 V | 4.72 V |
| 2 | 4.92 V | 4.99 V |
| 3 | 5.20 V | 5.28 V |

Automatic regeneration operations were tested by connecting a stable oscillator on the DREAMS-H frequency and ground pins. Regeneration was performed with different stable oscillator frequencies, and voltages on the power and heater lines of DREAMS-H were monitored. First, a +5 V voltage was observed on the power line as expected, lasting approximately two seconds following the automatic regeneration command. Then, one of the voltage levels presented in Table 16 was observed on the DREAMS-H heater line. Frequency on the stable oscillator was varied and all

different regeneration voltage settings were measured, including 0 V for frequencies outside the specified minimum and maximum values.

Next, it was confirmed that simultaneous regeneration and powering or measurement of DREAMS-H is not possible. The system refused to perform regeneration while the instrument was powered, and refused to start a measurement while the regeneration was going on. These attempts produced the expected error messages.

Lastly, the regeneration timer was verified to be working correctly. A 60 s regeneration was issued with the "REGENERATE" command and the heater voltage was measured to be on for the correct duration using a stopwatch.

6.4.4 Single test measurement commands

The test measurement command test was started by measuring every channel of each transducer with the same settings at least 15 times. The incoming measurement data was observed with the DREAMS EGSE software and no random errors were spotted during the process. The frequencies produced by the channels matched expectations based on different types of channels on previous FMI pressure and humidity instruments.

After this, the DREAMS-H transducer was replaced with the stable oscillator and measurements were performed on a set of various frequencies from 800 Hz to 21 kHz. The gathered measurements were used to determine the controller bus frequency as 8.0027 MHz, using the process described in Section 6.3.4.

Also, an oscilloscope was used to observe STEP and RESET signals while issuing "TEST_SENSOR_MEAS" commands. Both of the signals were measured to be 0 V pulses lasting for 3 ms, just as specified. In the case of multiple STEP signals following each other, the spacing between subsequent pulses was verified to be 3 ms.

6.4.5 Configuration table handling

Configuration table handling was tested with two sets of configuration tables for the transducers, versions A and B. First, a version A configuration table was uploaded for each transducer with "UPDATE_CONF" commands. The successful update of a table was verified with a "READ_CONF" command. Then, a controller reboot was issued and the "READ_CONF" operation repeated to verify successful storage of the configuration tables in D-Flash.

During the operations presented above, the structures in RAM hosting the configuration table data were monitored to match the data uploaded with "UPDATE_CONF" commands. Simultaneously, D-Flash memory sectors MAIN and BACKUP were monitored and verified to contain the same data. Lastly, the whole process including the reboots was repeated with version B configuration tables.

6.4.6 Automatic measurement commands

The automatic measurements were first tested for correct sequence timing in nominal situations. Default configuration tables were uploaded to the controller and the duration from the first to last measurement sequence was verified to be as specified

with a stopwatch. Simultaneously, system status was requested during and after the measurement series. In all cases, the status information correctly described the system state.

Then, the configuration tables were exchanged to ones, which specify continuous automatic measurements. Continuous measurements were started with "MEAS" commands issued for DREAMS-P1 and DREAMS-H transducers individually, as well for the combination of DREAMS-P2 and DREAMS-H. The generated data was monitored for random errors for at least 3 minutes during each of the automatic measurement series. No random errors were observed. Measurements with DREAMS-H were performed with the actual instrument, as well as with the stable oscillator connected on the DREAMS-H frequency line.

Lastly, the automatic measurements were tested in anomalous conditions. A new set of configuration tables was uploaded to the controller. These tables were designed deliberately with a shorter sequence interval than it takes time to complete a measurement sequence. This correctly caused the automatic measurement process to stop the ongoing measurement, send out the data from successfully measured channels and start the next measurement sequence. However, the beginning of the next measurement sequence was observed to be delayed slightly. This is because the software implementation checks for the need to start the next sequence only between the measurement of different channels. When this sequence time-out occurs during a channel measurement, the software first waits the measurement to finish, and thus delays the start of the next measurement.

6.4.7 Memory control commands

Memory control tests were started by inspecting an area of RAM with the "MEMORY_DUMP" command. The data returned by the command was verified by inspecting the addressed memory area with the debugging software. Then, a sequence of "MEMORY_LOAD", "MEMORY_VER" and "MEMORY_UPD" commands was used to update the contents of the same RAM area. Again, a successful update was verified with the "MEMORY_DUMP" command and the debugging software. The process was repeated for another memory location in RAM.

Next, the same memory dump and update steps were successfully performed for two different memory locations in D-Flash. Lastly, two memory locations in P-Flash were inspected with the "MEMORY_DUMP" command and verified to return correct data with the debugger.

6.4.8 Telecommand and telemetry handling

The DREAMS-P/H system was subjected to 18 different erroneous telecommand scenarios to test its telecommand and telemetry handling abilities. These scenarios correctly produced all of the six different possible error messages presented in Section 5.3.2. The instrument status was inspected between the erroneous commands, and no unexpected changes to the status were observed.

Some of the erroneous telecommand scenarios were combinations of multiple commands or errors. These parts of the test verified the ability of the controller to

handle and queue correctly multiple telemetry request in a short amount of time.

6.4.9 Power consumption

The DREAMS-P/H power consumption was measured to verify and correct the power consumption approximations made during the instrument design phase. The information obtained from this test is also essential for DREAMS mission planning.

In the course of the current measurements, it became clear that the power consumption on the +5 V line is radically larger during a brief moment in the beginning of each measurement sequence than during the actual measurements. This was deduced to be probably caused by the RESET signal going low in the beginning of each measurement sequence. Thus, it was deemed necessary to measure the power consumption also while the instrument is powered on, but not issuing RESET signals or sending out data, as this is the state which is maintained most of the time during a measurement sequence.

The current consumption was measured on the +5 V line and +12 V line in every operational situation. The results of the power consumption measurements are presented in Table 17.

Table 17: Instrument power consumption. Only the maximum power is listed in cases of fluctuating power consumption.

| Instrument mode | +5 V Current [mA] | +5 V Power [mW] | +12 V Current [mA] | +12 V Power [mW] | Total power [mW] |
|--|-------------------------|-----------------------|--------------------------|------------------------|------------------------|
| Idle | 6.09 | 30.45 | 0.00 | 0.00 | 30.45 |
| DREAMS-P1 power on | 6.53 | 32.65 | 5.85 | 70.20 | 102.85 |
| DREAMS-P2 power on | 6.52 | 32.60 | 5.91 | 70.92 | 103.52 |
| DREAMS-H power on | 6.62 | 33.10 | 7.14 | 85.68 | 118.78 |
| DREAMS-P1 and DREAMS-H power on | 7.05 | 35.25 | 13.03 | 156.36 | 191.61 |
| DREAMS-P2 and DREAMS-H power on | 7.04 | 35.20 | 13.08 | 156.96 | 192.16 |
| DREAMS-P1 power on and data transmission | <11 | < 55 | <5.9 | < 70.8 | < 125.8 |
| DREAMS-P2 power on and data transmission | <11 | < 55 | <6.0 | < 72 | < 127 |
| DREAMS-H power on and data transmission | <11 | < 55 | <7.5 | < 90 | < 145 |
| DREAMS-P1 and DREAMS-H power on and data transmission | <11 | < 55 | <13.4 | < 160.8 | < 215.8 |
| DREAMS-P2 and DREAMS-H power on and data transmission | <12 | < 60 | <13.5 | < 162 | < 222 |

From the results it is observed, that the DREAMS-P/H power consumption is radically lower than in previous FMI pressure and humidity instrument designs based on FPGAs. The microcontroller based DREAMS-P/H system consumed 30.45 mW of power when idle. Whereas, the most recent FPGA-based pressure and humidity instrument combination consumed 211 mW in its idle mode [26].

6.4.10 Analysis

The DREAMS-P/H qualification model verification test can be considered successful. Nevertheless, some minor anomalies were discovered, which need to be addressed.

Firstly, as described in Section 6.4.3, the DREAMS-H regeneration produced anomalously low voltages on all voltage settings. However, these differences are relatively small, ranging from 70 mV to 80 mV. The cause of this anomaly is most probably in the heater power supply components. The properties of LM117 regulators vary slightly from component to component, as well as reference resistor values vary within their tolerances. Thus, every built DREAMS-H heater power supply produces slightly different voltages. This needs simply to be taken into account while calibrating the regeneration temperature limits of the mission.

Another anomaly, related to the automatic measurement sequences, was discovered in Section 6.4.6. This anomaly only occurs in a special case of measurements performed with configuration tables unsuitable for the prevailing environmental conditions. The anomaly causes a small time duration increment of variable length into the inter-sequence measurement interval. This means that the time-step in a measurement sequence series becomes longer and varies slightly. Nevertheless, the data returned from the measurements is still correct. Hence, it was decided to stay with the current implementation, taking into account the possibility to fix this timing issue with a command updating the responsible configuration table.

Besides these two anomalies, the DREAMS-P/H controller performed all operations as specified in the software requirements. Thus, the software is verified for use on the DREAMS-P/H instrument qualification model, after an analysis is performed to verify that the measured data is correct also in large time series. This analysis is not part of this thesis, and is instead performed by the Finnish Meteorological Institute scientists, experienced in the study of the data from previous Mars pressure and humidity instruments.

7 Conclusions

This thesis has presented the design and testing of an embedded system used to gather scientific pressure and humidity measurements on the surface of Mars. First, the various levels of hardware design for the ExoMars EDM lander, the DREAMS science package, the DREAMS-P and DREAMS-H instruments, and the DREAMS-P/H instrument controller were presented in increasing detail. Then, the controller operational software was developed around the instrument hardware environment and requirements imposed on the system. Lastly, the verification testing of the instrument qualification model was planned and successfully executed.

The most imminent future developments concerning this thesis will be related to the assembly of the DREAMS-P and DREAMS-H flight models. Some features, like the possibility for an in-flight software update still need to be implemented for the FMs. Also, the verification test procedure needs to be updated to reflect also these new features. And finally, the verification test is to be performed on the flight models.

Looking further in the future, beyond the ExoMars EDM mission, there is potential to expand and modify the system presented in this thesis into a variety of other applications. Three different areas of future development for the DREAMS-P/H controller system can be envisioned:

- Using the design in other space missions
- Modernizing FMI laboratory test equipment
- Incorporating more sensors into the design

Utilization of the system presented in this thesis in other space missions is fairly likely. Almost always when there is a call for instrument proposals for a Mars lander, Finnish Meteorological Institute is involved. The miniature, low-power design of the pressure and humidity instruments makes it easy to incorporate these instruments into the design of a lander. The new instrument design based on a microcontroller instead of a FPGA further lowers the instrument power consumption and offers easier modification opportunities.

Currently, FMI is observing the evolving situation around the joint ESA and Russian ExoMars 2018 lander/rover project. The mission configuration is still unclear due to the restructuring of the ExoMars program. However, FMI is willing to cooperate in the project, if there is room for pressure and humidity instruments on the ExoMars 2018 landing platform or on the rover.

Also, NASA has announced its plans to perform another mission using a copy of the Mars Science Laboratory platform currently operating on Mars. The mission is scheduled to launch towards Mars in 2020 and is now looking for payloads [27]. FMI is participating in three international instrument proposals with a pressure and humidity instrument combination, also for this project.

Lastly, FMI has almost finished the development of the pressure and humidity instruments, MetBaro and MetHumi, for the MetNet precursor mission. MetNet is FMI's plan to deploy a network of small hard landers around Mars to perform

meteorological measurements. The purpose of the MetNet precursor mission is to deploy one such lander to provide proof-of-concept for the lander platform. The qualification level models of the precursor lander subsystems and payloads are currently nearing completion and the mission is awaiting a launch opportunity. The mission will be launched in 2018, or later.

The MetNet precursor mission pressure and humidity instruments are currently designed to use the old FPGA-based controller design. However, subsequent MetNet landers could utilize the microcontroller based instrument controller design presented in this thesis.

Use of the instrument controller design based on the MC9S12XEP100 MCU could be also broadened to the atmospheres of other planetary bodies. Only minor electrical and software modifications would be required, should a possibility arise to utilize the system in e.g. a Titan lander. The controller unit could be also used in short term missions outside any atmospheres. However, the system reliability would likely diminish for longer missions in the more severe radiation environment of vacuum, as the MCU cannot handle excessive doses of radiation based on preliminary approximations, but would have to be replaced by a radiation-tolerant component.

Additionally, the system developed in this thesis can be used to modernize some of the testing and calibration equipment in the FMI space laboratory. For example, flight component screening of Barocap® pressure and Humicap® humidity sensors is currently done with an old FPGA-based test setup. The documentation of this setup is lacking and the system is based on components now becoming obsolete and unavailable. Hence, these test systems need to be updated to maintain continued ability to screen components, and perform testing and calibration for future instruments. The DREAMS-P/H system could be easily modified for this use, providing flexibility through the ability to update the software of the assembled test setup.

Finally, FMI is currently studying the possibilities to include more sensors into its future planetary meteorology instruments. Besides the present pressure and humidity transducers, future missions could also measure e.g. temperature, wind or acceleration. The controller design presented in this thesis has a lot of scalability for more features in terms of unused pins, memory and peripheral modules. These resources could be utilized to design highly integrated transducer combinations with only a single electrical interface to the lander bus. This would mean lower power consumption and mass per instrument transducer, while simplifying the electrical, data and documentation interfaces shared between different organizations.

References

- [1] European Space Agency. *ExoMars website*. Accessed 12 November 2013. Available online. <http://exploration.esa.int/mars/>.
- [2] K.F. Wakker. *Astrodynamics-I*. Delft University of Technology, Delft, the Netherlands, 2010.
- [3] D.R. Williams. *Viking Mission to Mars*. Website. NASA Goddard Space Flight Center. Accessed 12 November 2013. Available online. <http://nssdc.gsfc.nasa.gov/planetary/viking.html>.
- [4] H. Kahanpää. *Mars Phoenix -luotaimen paineinstrumentin testaus ja datankäsittelyalgoritmien kehitys*. Master's thesis, Aalto-yliopisto, Espoo, Finland, 2011.
- [5] T. Owen et al. The Composition of the Atmosphere at the Surface of Mars. *Journal of Geophysical Research*, 82(28), September 1977. DOI: 10.1029/JS082i028p04635.
- [6] J.T. Schofield et al. The Mars Pathfinder Atmospheric Structure Investigation/Meteorology (ASI/MET) Experiment. *Science*, 278(5344), December 1997. DOI: 10.1126/science.278.5344.1752.
- [7] *The Finnish Meteorological Institute Research website*. Accessed 12 November 2013. Available online. <http://en.ilmatieteenlaitos.fi/research>.
- [8] K. Lodders and B. Fegley. *The Planetary Scientist's Companion*. Oxford University Press, 1998.
- [9] European Space Agency. *EXM-PL-ICD-ESA-00030. DREAMS Surface Payload Experiment Interface Control Document (SPL E-ICD)*, 1.0 edition, August 2013.
- [10] M. Komu. Finnish Meteorological Institute. Personal communication, October 2013.
- [11] Finnish Meteorological Institute. *FMLS-DREAMS-DS-002-PFM-08. Interface Description: DREAMS-P and DREAMS-H*, 1.8 edition, October 2013.
- [12] C. Zeitlin et al. Measurements of Energetic Particle Radiation in Transit to Mars on the Mars Science Laboratory. *Science*, 340(6136), May 2013. DOI: 10.1126/science.1235989.
- [13] Freescale Semiconductor. *MC9S12XEP100 Reference Manual*, 1.25 edition, February 2013.
- [14] Finnish Meteorological Institute. *FMI-S-GEN-UC-TR-001-AD-00. Freescale u-Controller Thermal Test*, May 2010.

- [15] HI-REL Laboratories. Report Number: US- 100416x. Washington, USA. Technical report, December 2010.
- [16] HI-REL Laboratories. Destructive Physical Analysis 100416. Washington, USA. Technical report, December 2010.
- [17] Instituto Nacional de Técnica Aeroespacial. *ESE-RPT-4316-055-INTA-13. Outgassing Test: DMAV-1314*, Spain. October 2013.
- [18] *ECSS-Q-ST-70-02C. European Cooperation for Space Standardization*. Noordwijk, the Netherlands, second edition, 2008.
- [19] Finnish Meteorological Institute. *FMLS-DREAMS-P-LI-001-PFM-08. DREAMS-P Declared Component List(DCL)*, 1.8 edition, September 2013.
- [20] National Semiconductor. *LM117/LM317A/LM317 3-Terminal Adjustable Regulator – Data sheet*, 2007.
- [21] Finnish Meteorological Institute. *FMLS-DREAMS-SP-002-PFM-03. DREAMS-P/H Software Requirements*, 3.0 edition, September 2013.
- [22] Bliley Technologies. *AT Cut TO Style Crystals – Data sheet*, 1.0 edition.
- [23] Finnish Meteorological Institute. *FMLS-DREAMS-SP-001-PFM-05. DREAMS-P/H Software Specification*, 1.5 edition, September 2013.
- [24] SofTec Microsystems. *EVB9S12XEP100 User’s Manual*, 2.0 edition, 2006.
- [25] Freescale Semiconductor. *CodeWarrior Development Studio for Freescale HCS12(X) Microcontrollers – Data sheet*, 2007.
- [26] W. Schmidt. Finnish Meteorological Institute. Personal communication, November 2013.
- [27] D. Brown and S. DeWitt. *NASA Announces Robust Multi-Year Mars Program; New Rover To Close Out Decade Of New Missions*. Press release. Accessed 12 November 2013. Available online. <http://mars.jpl.nasa.gov/news/whatsnew/index.cfm?FuseAction=ShowNews&NewsID=1401>.

A Pulse counting software implementation

This Annex describes the simplified source code used for pulse counting measurements of the DREAMS-H transducer. Parts of the code are omitted for clarity. Source code for DREAMS-P1 and DREAMS-P2 is almost identical to DREAMS-H implementation.

The ISR triggered by capturing a rising edge from the DREAMS-H:

```

1 interrupt VectorNumber_Vectch0 void countHpulses(void) {
2     if(count_H == 0) {
3         //Capture the first rising edge timestamp from PT70
4         firstamp_H = ECT_TC0;
5         //Clear the overflow counter and flag, if the overflow happened
           before the first pulse
6         oflow_count_H = 0;
7         oflow_flags &= ~(OFLOW_H);
8     }
9     else {
10        //Capture the current, non-first rising edge timestamp from PT0
11        lastamp_H = ECT_TC0;
12    }
13    //Increase the pulse count
14    count_H++;
15
16    //Start over the < 1 kHz time-out
17    PITCFLMT |= PITCFLMT_PFLMT0_MASK;
18    PITFLT |= PITFLT_PFLT2_MASK;
19
20    //Clear the Input Capture interrupt flag
21    ECT_TFLG1 = ECT_TFLG1_C0F_MASK;
22 }
```

The ISR called after pulse counting timer overflow:

```

1 interrupt VectorNumber_Vectovf void timeroverflow(void) {
2     ECT_TFLG2 = ECT_TFLG2_TOF_MASK;
3     oflow_flags = (OFLOW_P1|OFLOW_P2|OFLOW_H);
4 }
```

The subroutine monitoring if the required amount of rising edges have been counted or if a timer overflow has happened. This subroutine is constantly looped during a DREAMS-H channel measurement:

```


1 void pulsecounterH(unsigned int max_pulses, unsigned char meas_mode) {
2     //Check if enough H pulses have been counted
3     //Count n + 1 edges to solve the duration for n pulses
4     if(count_H > max_pulses) {
5         //Disable interrupts by incoming pulses and timer overflow
6         ECT_TIE &= ~(ECT_TIE_C0L_MASK);
7         ECT_TSCR2 &= ~(ECT_TSCR2_TOL_MASK);
8
9         //Disable timer
10        PITINTE &= ~PITINTE_PINTE2_MASK;
11        PITCE &= ~PITCE_PCE2_MASK;
12        if(PITCE == 0x00) {
```

```

13         PITCFLMT &= ~PITCFLMT.PITE_MASK;
14     }
15
16     //Check if the timer overflowed just before the last pulse
17     if(laststamp_H < 0xFFFF){
18         if(overflow_flags & OFLOW_H) {
19             overflow_flags &= ~(OFLOW_H);
20             overflow_count_H++;
21         }
22     }
23
24     //Store the duration of the measurement in reference clock
25     cycles
26     meastime_H = (0x00010000 * overflow_count_H) + laststamp_H -
27     firststamp_H;
28
29     //Return to the measurement sequence scheduler
30     //CODE REMOVED FOR CLARITY
31 }
32 //Check if the timer has overflowed
33 else{
34     if(overflow_flags & OFLOW_H) {
35         overflow_flags &= ~(OFLOW_H);
36         overflow_count_H++;
37     }
38     return;
39 }

```

B Software verification test stepwise procedure

|  | | Doc.No: FMI_S-DREAMS-PR-002_QM-04 Issue: 1.4 Date: 08.10.2013 Page: 1 / 19 | | | | |
|---|---|---|--|---------------|---------|-----------|
| Software Test Procedure | | | | | | |
| Step-by-step procedure | | | | | | |
| Table 6.1. Step-by-step test procedure. | | | | | | |
| Step | Activity description | Requirement number | Required result | Actual result | Remarks | Pass/Fail |
| 1. Preparative tasks | | | | | | |
| 1A. | Read through the FMI_S-DREAMS-PR-002-QM-04 document (this one) | | Done | | | |
| 1B. | Select RS-422 operation with the selection switches on the RS-422 interface | | Done | | | |
| 1C. | Connect the USB cable between the EGSE PC and RS-422 interface | | Done | | | |
| 1D. | Provide power for the RS-422 interface | | RS-422 PSU connected to mains | | | |
| 1E. | Power on the EGSE PC and launch the DREAMS-P/H EGSE software | | Done | | | |
| 1F. | Check the calibration date of the multimeters | | Valid calibration | | | |
| 1G. | Fill the specifications of the multimeters to the instrument information table 6.2 | | Done | | | |
| 1H. | Check the calibration date of the oscilloscope and the compensation of the probe | | Valid calibration and probe correctly compensated | | | |
| 1I. | Fill the specifications of the oscilloscope to the instrument information table 6.2 | | Done | | | |
| 1J. | Adjust the DC lab supply voltages to +5 V and +12 V using the multimeter | | + 5 V: > +4.95 V < +5.05 V and +12 V: > +11.95 V < +12.05 V | | | |
| 1K. | Fill the DC lab supply specifications to the instrument information table 6.2 | | Done | | | |
| 1L. | Check the calibration date of the stable oscillator | | Valid calibration | | | |
| 1M. | Fill the specifications of the stable oscillator to the instrument information table 6.2 | | Done | | | |
| 1N. | Start a PC and launch CodeWarrior IDE, open pulsecounter.mcp | | Done | | | |
| Step | Activity description | Requirement number | Required result | Actual result | Remarks | Pass/Fail |
| 2. Connecting the cables | | | | | | |



Doc.No: FML_S-DREAMS-PR-002_QM-04
 Issue: 1.4
 Date: 08.10.2013
 Page: 2 / 19

Software Test Procedure

| 2A. | Connect the DREAMS-H cable to the DREAMS-H connector on the PE board (labeled "DREAMS-H" in EM) | | Done | | | |
|-------------------|---|---|--|---------------|---------|-----------|
| 2B. | Connect the RS-422/power cable between the RS-422 interface port 1 and DREAMS-P/H CEU connector (labeled "serial" in EM) | | Done | | | |
| 2C. | Connect the GND terminals of the power supplies to the same potential | | Done | | | |
| 2D. | Connect the power leads on the RS-422/power cable labeled +5V and +12V to the respective voltage outputs on the DC power supplies | | Done | | | |
| 2E. | Connect the debugger to the debugging connector on the PE board (labeled "debug" in EM) Ensure correct insertion! | | Done | | | |
| 2F. | Connect the debugger USB to the PC running CodeWarrior | | Done | | | |
| Step | Activity description | Requirement number | Required result | Actual result | Remarks | Pass/Fail |
| 3. Startup | | | | | | |
| 3A. | Switch on +5 V supply | 4.3.21, 4.5.2 | TM: a007210000000000 21 | | | |
| 3B. | Switch on +12 V supply | | Done | | | |
| 3C(1). | Load application to the DREAMS-P/H controller and start the executing it with the CodeWarrior Real-time debugger | 4.3.21 | TM: a007210000000000 21 | | | |
| 3C(2). | Send "SYSTEM" MEMORY_DUMP 0x100200 2 | | Telemetry gives the correct version number | | | |
| 3D. | Send "SYSTEM" STATUS | 4.2.19, 4.2.20, 4.2.21, 4.2.22, 4.2.23, 4.2.24, 4.3.13, 4.3.14, 4.5.3 | TM: a003080008, TM: a008200000000000 0020 | | | |
| 3E. | Send "SYSTEM" REBOOT 0x55 | 4.3.19, 4.3.20, 4.3.21 | TM: a007210000000000 21 | | | |



Doc.No: FML_S-DREAMS-PR-002_QM-04
 Issue: 1.4
 Date: 08.10.2013
 Page: 3 / 19

Software Test Procedure

| 3F. | Force watchdog reset: "Reset target" in debugger, place breakpoint in the start of the switch statement in the main loop in file main.c and "Start" | 4.3.21 | TM: a007210000000000 21 followed by: TM: a007210000000000 21 after ~2 s | | | |
|--|---|---|---|---------------|---------|-----------|
| Step | Activity description | Requirement number | Required result | Actual result | Remarks | Pass/Fail |
| 4. Power commands NOTE: Switch off lab DC supply when modifying test connections! | | | | | | |
| 4A. | Remove the DREAMS-H for this test and attach multimeter test clips to the 12V_GND pin (Glenair pin 1 in EM) and the POWER_H pin (Glenair pin 2 in EM) | | Done | | | |
| 4B. | After powering up the controller, measure voltage on the POWER_H pin | 4.1.10 | 0 V | | | |
| 4C. | Measure voltage on the STEP_H pin | 4.1.9 | 0 V | | | |
| 4D. | Measure voltage on the RESET_H pin | 4.1.8 | 0 V | | | |
| 4E(1) . | Send "DREAMS-H" POWER_ON and measure the voltage on the POWER_H pin | 4.3.30 | TM: a403090009, voltage: +5V | | | |
| 4E(2) . | Measure voltage on the STEP_H pin | | + 5 V | | | |
| 4E(3) . | Measure voltage on the RESET_H pin | | + 5 V | | | |
| 4F. | Send "SYSTEM" STATUS | 4.3.13, 4.3.14 | TM: a008200400000000 00CS | | | |
| 4G(1)). | Send "DREAMS-H" POWER_OFF and measure the voltage on the POWER_H pin | 4.3.31 | TM: a4030a000a, voltage: 0V | | | |
| 4G(2)). | Measure voltage on the STEP_H pin | 4.1.9 | 0 V | | | |
| 4G(3)). | Measure voltage on the RESET_H pin | 4.1.8 | 0 V | | | |
| 4H. | Send "SYSTEM" STATUS | 4.3.13, 4.3.14 | TM: a008200000000000 00CS | | | |
| 4I. | Test various combinations of power commands, request instrument status after commands and fill power switch statuses into table 6.3 . | 4.1.6, 4.1.7, 4.3.13, 4.3.14, 4.3.16, 4.3.22, 4.3.30, 4.3.31 | 1. Power switch statuses as defined in AD1 2. It is not possible to switch ON -P1 and -P2 at the same time | | | |
| Step | Activity description | Requirement number | Required result | Actual result | Remarks | Pass/Fail |



Doc.No: FML_S-DREAMS-PR-002_QM-04
 Issue: 1.4
 Date: 08.10.2013
 Page: 4 / 19

Software Test Procedure

5. DREAMS-H heater command: REGENERATE

| | | | | | | |
|-----|---|------------------------------|----------------------------------|--|--|--|
| 5A. | Remove the DREAMS-H for this test and attach multimeter test clips to the 12V_GND pin (Glenair pin 1 in QM) and the POWER_H pin (Glenair pin 2 in QM). Disconnect the debugger. | | Done | | | |
| 5B. | Attach another multimeter test clips to the H_HTR_in_1 pin (Glenair pin 6 in EM) and the H_HTR_RTN pin (Glenair pin 7 in EM) | | Done | | | |
| 5C. | Send "SYSTEM" REBOOT 0x55 | 4.3.19, 4.3.20, 4.3.21 | TM: a007210000000000 21 | | | |
| 5D. | Send "DREAMS-H" REGENERATE 0x01 15 | | TM: a4030c000c | | | |
| 5E. | Send "SYSTEM" STATUS while the regeneration is going on | 4.3.13, 4.3.14 | TM: a008200800000000 08CS | | | |
| 5F. | Read both multimeters while the regeneration is going on | | POWER_H = 0V, H_HTR_in_1=4.7V | | | |
| 5G. | Inspect the REGENERATE TM packet | 4.4.19 | TM: a40842010000000f 00CS | | | |
| 5H. | Send "DREAMS-H" REGENERATE 0x02 15 | | TM: a4030c000c | | | |
| 5I. | Send "SYSTEM" STATUS while the regeneration is going on | 4.3.13, 4.3.14 | TM: a008200800000000 08CS | | | |
| 5J. | Read both multimeters while the regeneration is going on | | POWER_H = 0V, H_HTR_in_1=5V | | | |
| 5K. | Inspect the REGENERATE TM packet | 4.4.19 | TM: a40842020000000f 00CS | | | |
| 5L. | Send "DREAMS-H" REGENERATE 0x03 15 | | TM: a4030c000c | | | |
| 5M. | Send "SYSTEM" STATUS while the regeneration is going on | 4.3.13, 4.3.14 | TM: a008200800000000 08CS | | | |
| 5N. | Read both multimeters while the regeneration is going on | | POWER_H = 0V, H_HTR_in_1=5.3V | | | |
| 5O. | Inspect the REGENERATE TM packet | 4.4.19 | TM: a40842030000000f 00CS | | | |
| 5P. | Connect the stable oscillator to the DREAMS-H frequency input | | Done | | | |
| 5Q. | Set the the stable oscillator waveform to square wave, amplitude to 5 V p-p, adjust the frequency to 1.5 kHz | | Done | | | |
| 5R. | Send "DREAMS-H" REGENERATE 0x00 5 | | TM: a4030c000c | | | |



Doc.No: FML_S-DREAMS-PR-002_QM-04
 Issue: 1.4
 Date: 08.10.2013
 Page: 5 / 19

Software Test Procedure

| | | | | | | |
|-------|---|--------|--|--|--|--|
| 5S(1) | Read both multimeters while the regeneration is going on | | First: POWER_H=5V, H_HTR_in_1=0V Then: POWER_H = 0V, H_HTR_in_1=0...5.3V | | | |
| 5S(2) | Repeat 5R-5S(1) while increasing frequency in eg. 1 kHz steps until you have deduced the rough frequency borders for max heating temperature, heating setting 1, 2, 3 and min heating temperature (0 V, 4.7 V, 5.0 V, 5.3 V and 0 V (again)) | | Found limits for all different temperature cases, preserve the rough frequency limits | | | |
| 5T. | Send "DREAMS-H" POWER_ON | | TM: a403090009, POWER_H =5V | | | |
| 5U(1) | Send "DREAMS-H" REGENERATE 0x02 15 | | TM: a4030c06CS, POWER_H = 5V, H_HTR_in_1=0V | | | |
| 5U(2) | Send "DREAMS-H" POWER_OFF | | TM: a4030a000a, POWER_H =0V | | | |
| 5U(3) | Send "DREAMS-H" REGENERATE 0x02 15 | | TM: a4030c00CS, POWER_H = 0V, H_HTR_in_1=5V | | | |
| 5V. | Send "DREAMS-H" MEAS while the regeneration is going on | 4.3.9 | TM: a4030b05CS | | | |
| 5W. | Send "DREAMS-H" REGENERATE 0x02 15 | | POWER_H = 0V, H_HTR_in_1=5V | | | |
| 5X. | Send "SYSTEM" STANDBY 0xAA while the regeneration is going on | 4.4.13 | POWER_H = 0V, H_HTR_in_1=0V | | | |
| 5Y. | Send "DREAMS-H" REGENERATE 0x02 60 and measure the time when H_HTR_in_1=5V with a stopwatch | 4.4.13 | Time = 60 s | | | |
| 5Z. | Connect DREAMS-H | | Done | | | |
| 5AA. | Send "DREAMS-H" POWER_ON | | TM: a403090009 | | | |
| 5AB. | Send "DREAMS-H" TEST_SENSOR_CONFIG 500 18 0x20 | | TM: a403100010 | | | |
| 5AC. | Send "DREAMS-H" TEST_SENSOR_MEAS | | Preserve measurement result | | | |
| 5AD. | Send "DREAMS-H" POWER_OFF | | TM: a4030a000a | | | |
| 5AE. | Send "DREAMS-H" REGENERATE 0x00 0 | | TM: a4030c00CS, measurement result in the regeneration report is +-500 of the result obtained in step 5AC | | | |



Doc.No: FML_S-DREAMS-PR-002_QM-04
 Issue: 1.4
 Date: 08.10.2013
 Page: 6 / 19

Software Test Procedure

| 5AF. | Put the device in weather cabinet, send automatic regeneration commands in different temperatures and fill the results to table 6.4. | 4.4.18 | Heating voltage levels agree with configuration table heating limits | | | |
|-------------------------------------|---|--------------------|---|---------------|---------|-----------|
| Step | Activity description | Requirement number | Required result | Actual result | Remarks | Pass/Fail |
| 6. Test measurement commands | | | | | | |
| 6A. | Send "DREAMS-P1" POWER_ON | | TM: a103090009 | | | |
| 6B. | Send "DREAMS-P1" TEST_SENSOR_CONFIG 400 18 0xFF | 4.3.33 | TM: a103100010 | | | |
| 6C. | Wait ~1 min | | Done | | | |
| 6D. | Send "DREAMS-P1" TEST_SENSOR_MEAS and store the pulse count of each channel into table 6.5. | 4.3.34 | Done | | | |
| 6E. | Repeat the measurement 15 more times and monitor the measurement times | | Measurement time difference between two consecutive sequences <500 clock cycles | | | |
| 6F. | Send "DREAMS-P2" POWER_ON | | TM: a203090009 | | | |
| 6G. | Send "DREAMS-P2" TEST_SENSOR_CONFIG 400 18 0xFF | 4.3.33 | TM: a203100010 | | | |
| 6H. | Wait ~1 min | | Done | | | |
| 6I. | Send "DREAMS-P2" TEST_SENSOR_MEAS and store the pulse count of each channel into table 6.5. | 4.3.34 | Done | | | |
| 6J. | Repeat the measurement 15 more times and monitor the measurement times | | Measurement time difference between two consecutive sequences <500 clock cycles | | | |
| 6K. | Send "DREAMS-H" POWER_ON | | TM: a403090009 | | | |
| 6L. | Send "DREAMS-H" TEST_SENSOR_CONFIG 400 18 0xFF | 4.3.33 | TM: a403100010 | | | |
| 6M. | Wait ~1 min | | Done | | | |
| 6N. | Send "DREAMS-H" TEST_SENSOR_MEAS and store the pulse count of each channel into table 6.5. | 4.3.34 | Done | | | |
| 6O. | Repeat the measurement 15 more times and monitor the measurement times | | Measurement time difference between two consecutive sequences <500 clock cycles | | | |



Doc.No: FML_S-DREAMS-PR-002_QM-04
 Issue: 1.4
 Date: 08.10.2013
 Page: 7 / 19

Software Test Procedure

| | | | | | | |
|-----|---|------------------------|---|--|--|--|
| 6P. | Disconnect the DREAMS-H and connect the stable oscillator signal line to PE DREAMS-H connector pin <code>FREQ_H_in</code> (Glenair pin 5 in EM) and the stable oscillator ground line to <code>12V_GND</code> (Glenair pin 1 in EM) | | Done | | | |
| 6Q. | Send "DREAMS-H" <code>POWER_ON</code> | | TM: a403090009 | | | |
| 6R. | Set the the stable oscillator waveform to square wave, amplitude to 5 V p-p, adjust the frequency, perform test measurements (measure each frequency multiple times) and fill in table 6.6. | 4.4.10 | Calculated bus frequencies are equal within the error margins regardless of the stable oscillator frequency | | | |
| 6S. | Attach oscilloscope probe to the DREAMS-H connector <code>Reset_H</code> pin (Glenair pin 3 in EM) and the probe ground clip to <code>12V_GND</code> pin (Glenair pin 1 in EM) in single measurement mode, trigger set for a transition from 5 V to 0 V | | Done | | | |
| 6T. | Send "DREAMS-H" <code>TEST_SENSOR_MEAS</code> | 4.1.14, 4.1.17 | RESET pulse duration = 3 ms | | | |
| 6U. | Send "DREAMS-H" <code>TEST_SENSOR_CONFIG</code> 400 18 0x80 | | TM: a403100010 | | | |
| 6V. | Attach oscilloscope probe to the DREAMS-H connector <code>Step_H</code> pin (Glenair pin 4 in EM) and the probe ground clip to <code>12V_GND</code> pin (Glenair pin 1 in EM) in single measurement mode, trigger set for a transition from 5 V to 0 V | 4.4.7 | Done | | | |
| 6W. | Send "DREAMS-H" <code>TEST_SENSOR_MEAS</code> | 4.1.15, 4.1.18, 4.1.19 | STEP pulse duration = 3 ms, duration between consecutive STEP pulses = 3 ms | | | |
| 6X. | Send "DREAMS-P2" <code>POWER_ON</code> | | TM: a203090009 | | | |
| 6Y. | Send "DREAMS-P2" <code>TEST_SENSOR_CONFIG</code> 400 18 0x55 | | TM: a203100010 | | | |
| 6Z. | Send "DREAMS-P2" <code>TEST_SENSOR_MEAS</code> | 4.1.16 | Only channels 1, 3, 5 and 7 measured, measured data is similar to the data generated at step 6D | | | |



Doc.No: FML_S-DREAMS-PR-002_QM-04
 Issue: 1.4
 Date: 08.10.2013
 Page: 8 / 19

Software Test Procedure

| 6AA. | Send "DREAMS-P2" TEST_SENSOR_CONFIG 400 18 0xAA | | TM: a203100010 | | | |
|--|--|------------------------------|---|------------------|---------|-----------|
| 6AB. | Send "DREAMS-P2" TEST_SENSOR_MEAS | 4.1.16 | Only channels 2, 4, 6 and 8 measured, measured data is similar to the data generated at step 6D | | | |
| Step | Activity description | Require ment number | Required result | Actual result | Remarks | Pass/Fail |
| 7. Configuration table commands | | | | | | |
| 7A. | Create (or if already created, check for correctness) configuration tables CONF_P1A.channelconf, CONF_P2A.channelconf and CONF_HA.channelconf presented as "Ver. A" in table 6.7. | | Done | | | |
| 7B. | During the following steps, monitor with the debugger interface the data in the config_tables structure array and the global memory locations 0x100000- 0x1000FF (MAIN flash sector for config tables) and 0x100100-0x1001FF (BACKUP flash sector for config tables) | 4.5.1 | Data in the config_tables structure as well as in the MAIN and BACKUP flash sectors matches the CONF_P1A.channe lconf data | | | |
| 7C. | Send "SYSTEM" UPDATE_CONF 0x01 CONF_P1A | | TM: a003020002 | | | |
| 7D. | Send "SYSTEM" READ_CONF 0x01 | 4.3.23, 4.3.24 | The TM packet correctly describes the contents of the CONF_P1A.channe lconf | | | |
| 7E. | Send "SYSTEM" REBOOT 0x55 | 4.3.19, 4.3.20, 4.3.21 | TM: a007210000000000 21 | | | |
| 7F. | Send "SYSTEM" READ_CONF 0x01 | 4.3.23, 4.3.24 | The TM packet correctly describes the contents of the CONF_P1A.channe lconf | | | |



Doc.No: FML_S-DREAMS-PR-002_QM-04
 Issue: 1.4
 Date: 08.10.2013
 Page: 9 / 19

Software Test Procedure

| | | | | | | |
|-----|---|------------------------|---|--|--|--|
| 7G. | During the following steps, monitor with the debugger interface the data in the config_tables structure array and the global memory locations 0x100000-0x1000FF (MAIN flash sector for config tables) and 0x100100-0x1001FF (BACKUP flash sector for config tables) | 4.5.1 | Data in the config_tables structure as well as in the MAIN and BACKUP flash sectors matches the CONF_P2A.channelconf data | | | |
| 7H. | Send "SYSTEM" UPDATE_CONF 0x02 CONF_P2A. | | TM: a003020002 | | | |
| 7I. | Send "SYSTEM" READ_CONF 0x02 | 4.3.23, 4.3.24 | The TM packet correctly describes the contents of the CONF_P2A.channelconf | | | |
| 7J. | Send "SYSTEM" REBOOT 0x55 | 4.3.19, 4.3.20, 4.3.21 | TM: a00721000000000021 | | | |
| 7K. | Send "SYSTEM" READ_CONF 0x02 | 4.3.23, 4.3.24 | The TM packet correctly describes the contents of the CONF_P2A.channelconf | | | |
| 7L. | During the following steps, monitor with the debugger interface the data in the config_tables structure array and the global memory locations 0x100000-0x1000FF (MAIN flash sector for config tables) and 0x100100-0x1001FF (BACKUP flash sector for config tables) | 4.5.1 | Data in the config_tables structure as well as in the MAIN and BACKUP flash sectors matches the CONF_HA.channelconf data | | | |
| 7M. | Send "SYSTEM" UPDATE_CONF 0x04 CONF_HA | | TM: a003020002 | | | |
| 7N. | Send "SYSTEM" READ_CONF 0x04 | | The TM packet correctly describes the contents of the CONF_HA.channelconf | | | |
| 7O. | Send "SYSTEM" REBOOT 0x55 | 4.3.19, 4.3.20, 4.3.21 | TM: a00721000000000021 | | | |
| 7P. | Send "SYSTEM" READ_CONF 0x04 | | The TM packet correctly describes the contents of the CONF_HA.channelconf | | | |



Doc.No: FML_S-DREAMS-PR-002_QM-04
 Issue: 1.4
 Date: 08.10.2013
 Page: 10 / 19

Software Test Procedure

| | | | | | | |
|------|--|------------------------------|--|--|--|--|
| 7Q. | Create (or if already created, check for correctness) configuration tables CONF_P1B.channelconf, CONF_P2B.channelconf and CONF_HB.channelconf presented as "Ver. B" in table 6.7. | | Done | | | |
| 7R. | Send "SYSTEM" UPDATE_CONF 0x01 CONF_P1B | | TM: a003020002 | | | |
| 7S. | Send "SYSTEM" READ_CONF 0x01 | | The TM packet correctly describes the contents of the CONF_P1B.channelconf | | | |
| 7T. | Send "SYSTEM" REBOOT 0x55 | 4.3.19, 4.3.20, 4.3.21 | TM: a007210000000000 21 | | | |
| 7U. | Send "SYSTEM" READ_CONF 0x01 | | The TM packet correctly describes the contents of the CONF_P1B.channelconf | | | |
| 7V. | Send "SYSTEM" UPDATE_CONF 0x02 CONF_P2B | | TM: a003020002 | | | |
| 7W. | Send "SYSTEM" READ_CONF 0x02 | | The TM packet correctly describes the contents of the CONF_P2B.channelconf | | | |
| 7X. | Send "SYSTEM" REBOOT 0x55 | 4.3.19, 4.3.20, 4.3.21 | TM: a007210000000000 21 | | | |
| 7Y. | Send "SYSTEM" READ_CONF 0x02 | | The TM packet correctly describes the contents of the CONF_P2B.channelconf | | | |
| 7Z. | Send "SYSTEM" UPDATE_CONF 0x04 CONF_HB | | TM: a003020002 | | | |
| 7AA. | Send "SYSTEM" READ_CONF 0x04 | | The TM packet correctly describes the contents of the CONF_HB.channelconf | | | |
| 7AB. | Send "SYSTEM" REBOOT 0x55 | 4.3.19, 4.3.20, 4.3.21 | TM: a007210000000000 21 | | | |
| 7AC. | Send "SYSTEM" READ_CONF 0x04 | | The TM packet correctly describes the contents of the CONF_HB.channelconf | | | |



Doc.No: FML_S-DREAMS-PR-002_QM-04
 Issue: 1.4
 Date: 08.10.2013
 Page: 11 / 19

Software Test Procedure

| 7AD. | Send the "Ver. A" tables to the controller. | | Done | | | |
|--|---|--------------------|--|---------------|---------|-----------|
| Step | Activity description | Requirement number | Required result | Actual result | Remarks | Pass/Fail |
| 8. Automatic measurement commands | | | | | | |
| 8A. | Restart DREAMS-P/H EGSE software to produce a separate (or even multiple) logfile(s) for this part of the test | | Done | | | |
| 8B. | Verify that the "Ver. A" of table 6.7. configuration tables are in the controller. Disconnect the debugger after verification. | | Done | | | |
| 8C. | Send "DREAMS-P1" POWER_ON | | TM: a103090009 | | | |
| 8D. | Send "DREAMS-P1" MEAS and measure the time from the arrival of the first measurement TM data packet to the arrival of the last data packet with a stopwatch | | Measured time = (number of sequence repetitions - 1) * sequence interval ± 0.2 s | | | |
| 8E. | Send "SYSTEM" STATUS during the measurement | | TM: a00820011X00000001CS, X=0 or 1 | | | |
| 8F. | Send "SYSTEM" STATUS after the measurement | 4.4.11 | TM: a008200000000000000CS | | | |
| 8G. | Send "DREAMS-P2" POWER_ON | | TM: a203090009 | | | |
| 8H. | Send "DREAMS-P2" MEAS and measure the time from the arrival of the first measurement TM data packet to the arrival of the last data packet with a stopwatch | | Measured time = (number of sequence repetitions - 1) * sequence interval ± 0.2 s | | | |
| 8I. | Send "SYSTEM" STATUS during the measurement | | TM: a00820020X00000002CS, X=0 or 2 | | | |
| 8J. | Send "SYSTEM" STATUS after the measurement | | TM: a008200200000000000CS | | | |
| 8L. | Send "DREAMS-P2" POWER_OFF | | TM: a2030a000a | | | |
| 8M. | Send "DREAMS-H" POWER_ON | | TM: a403090009 | | | |
| 8N. | Send "DREAMS-H" MEAS and measure the time from the arrival of the first measurement TM data packet to the arrival of the last data packet with a stopwatch | | Measured time = (number of sequence repetitions - 1) * sequence interval ± 0.2 s | | | |



Doc.No: FML_S-DREAMS-PR-002_QM-04
 Issue: 1.4
 Date: 08.10.2013
 Page: 12 / 19

Software Test Procedure

| | | | | | | |
|--------|--|----------------|---|--|--|--|
| 8O. | Send "SYSTEM" STATUS during the measurement | | TM: a00820044X00000004CS, X=0 or 4 | | | |
| 8P. | Send "SYSTEM" STATUS after the measurement | 4.4.11 | TM: a00820000000000000CS | | | |
| 8Q(1). | Load "VER. C" of table 6.7. configuration tables into the controller | | Done | | | |
| 8Q(2). | Send "DREAMS-P1" POWER_ON | | TM: a103090009 | | | |
| 8R. | Send "DREAMS-P1" MEAS and collect measurements for at least 3 minutes | 4.4.6 | Monitor the incoming data for anomalies (Further analysis of the automatic measurements is done with another ground support SW) | | | |
| 8S. | Send "SYSTEM" STATUS during the measurement | | TM: a00820010X00000001CS, X=0 or 1 | | | |
| 8T. | Interrupt measurements by sending "SYSTEM" STANDBY 0xAA, followed by "SYSTEM" STATUS | 4.3.15, 4.3.16 | TM: a00820000000000000CS | | | |
| 8U. | Send "DREAMS-P2 DREAMS-H" POWER_ON | | TM: a603090009 | | | |
| 8V. | Send "DREAMS-P2 DREAMS-H" MEAS and collect measurements for at least 3 minutes | 4.4.6 | Monitor the incoming data for anomalies (Further analysis of the automatic measurements is done with another ground support SW) | | | |
| 8W. | Send "SYSTEM" STATUS during the measurement | | TM: a00820060X00000006CS, X=0, 2, 4 or 6 | | | |
| 8X. | Interrupt measurements by sending "SYSTEM" STANDBY 0xAA | | Done | | | |
| 8Y. | Disconnect DREAMS-H from the PE board and connect the stable oscillator to the controller DREAMS-H frequency input as earlier in the part 6 of the test procedure, Set the oscillator to 7 200 Hz | | Done | | | |
| 8Z. | Send "DREAMS-H" POWER_ON | | TM: a403090009 | | | |



Doc.No: FML_S-DREAMS-PR-002_QM-04
 Issue: 1.4
 Date: 08.10.2013
 Page: 13 / 19

Software Test Procedure

| | | | | | | |
|------|--|-------------------|--|--|--|--|
| 8AA. | Send "DREAMS-H" MEAS and collect measurements for at least 3 minutes | | Monitor the incoming data for anomalies (Further analysis of the automatic measurements is done with another ground support SW) | | | |
| 8AB. | Send "SYSTEM" STATUS during the measurement | | TM: a00820040X0000004CS, X=0 or 4 | | | |
| 8AC. | Interrupt measurements by sending "SYSTEM" STANDBY 0xAA | | Done | | | |
| 8AD. | Send "DREAMS-H" POWER_ON | | TM: a403090009 | | | |
| 8AE. | Send "DREAMS-H" MEAS | | Monitor the incoming data for anomalies | | | |
| 8AF. | Switch the stable oscillator signal on and off and monitor the measurement data | | Measurement keeps on going, measurement data is filled with zeros when the signal is off | | | |
| 8AG. | Interrupt measurements by sending "SYSTEM" REBOOT 0x55 | 4.3.19, 4.3.20 | Done | | | |
| 8AH. | Load "VER. D" of table 6.7. configuration tables into the controller | | Done | | | |
| 8AI. | Send "DREAMS-P1" POWER_ON | | TM: a103090009 | | | |
| 8AJ. | Send "DREAMS-P1" MEAS and measure the time from the arrival of the first measurement TM data packet to the arrival of the second to last data packet with a stopwatch | 4.4.5 | Part of the measurement channels return results, other channels return zeros and Measured time = (number of sequence repetitions - 2) * sequence interval +/- 0.2 s | | | |
| 8AK. | Send "DREAMS-P2" POWER_ON | | TM: a203090009 | | | |
| 8AL. | Send "DREAMS-P2" MEAS and measure the time from the arrival of the first measurement TM data packet to the arrival of the second to last data packet with a stopwatch | 4.4.5 | Part of the measurement channels return results, other channels return zeros and Measured time = (number of sequence repetitions - 2) * sequence interval +/- 0.2 s | | | |



Doc.No: FML_S-DREAMS-PR-002_QM-04
 Issue: 1.4
 Date: 08.10.2013
 Page: 14 / 19

Software Test Procedure

| | | | | | | |
|---|--|------------------------------------|--|--------------------------|----------------|------------------|
| 8AM. | Send "DREAMS-H" POWER_ON | | TM: a403090009 | | | |
| 8AN. | Send "DREAMS-H" MEAS and measure the time from the arrival of the first measurement TM data packet to the arrival of the second to last data packet with a stopwatch | 4.4.5 | Part of the measurement channels return results, other channels return zeros and Measured time = (number of sequence repetitions - 2) * sequence interval +/-0.2 s | | | |
| 8AO. | Send "DREAMS-P2 DREAMS-H" POWER_ON | | TM: a603090009 | | | |
| 8AP. | Send "DREAMS-P2 DREAMS-H" MEAS | | Part of the -P2 measurement channels return, other -P2 channels and all -H channels return zeros | | | |
| 8AQ. | Perform at least 30 minute automatic measurements of each individual transducers and transducer combinations using the new calibration/testing software ("MSENS"). Analyze the produced data. | | No random errors in the datasets | | | |
| Step | Activity description | Require ment number | Required result | Actual result | Remarks | Pass/Fail |
| 9. Memory control commands (THE MCU HAS TO BE REPROGRAMMED AFTER THIS) | | | | | | |
| 9A. | Monitor the memory around global address 0x0F5236 (RAM) with the debugger software | | Memory data and TM packets match | | | |
| 9B. | Send "SYSTEM" MEMORY_DUMP 0x0F5230 24 | 4.3.29, 4.5.4 | TM packet received | | | |
| 9C. | Send "SYSTEM" MEMORY_LOAD 0x0F5236 0xCA 0xFE 0xAB 0xBA | 4.3.25 | TM: a003040004 | | | |
| 9D. | Send "SYSTEM" MEMORY_VER 0x0F5236 | 4.3.27 | TM: a0092236520fcafea bbaCS | | | |
| 9E. | Send "SYSTEM" MEMORY_UPD 0x0F5236 | 4.3.28, 4.6.4 | TM: a003060006, Data changes in the memory after refreshing the memory view in the debugger | | | |



Doc.No: FML_S-DREAMS-PR-002_QM-04
 Issue: 1.4
 Date: 08.10.2013
 Page: 15 / 19

Software Test Procedure

| | | | | | | |
|------|---|---------------------------|--|------------------|---------|-----------|
| 9F. | Send "SYSTEM" MEMORY_DUMP 0x0F5230 24 | 4.3.29, 4.5.4 | TM packet contains the newly programmed data, unrelated bytes in the packet are not changed | | | |
| 9G. | Select a random RAM address (0x0F0000 - 0x0FFFFFFF) and repeat the steps 9A-9G | | Passed all steps | | | |
| 9H. | Monitor the memory around global address 0x102120 (D- flash) with the debugger software | | Memory data and TM packets match | | | |
| 9I. | Send "SYSTEM" MEMORY_DUMP 0x102120 24 | 4.3.29, 4.5.4 | TM packet received | | | |
| 9J. | Send "SYSTEM" MEMORY_LOAD 0x902120 0xFE 0xED 0x12 0x34 0x56 0x78 0xDE 0xFA | 4.3.25 | TM: a003040004 | | | |
| 9K. | Send "SYSTEM" MEMORY_VER 0x902120 | 4.3.27 | TM: a00d22202190f eed12345678defaC S | | | |
| 9L. | Send "SYSTEM" MEMORY_UPD 0x902120 | 4.3.28, 4.6.4 | TM: a003060006, Data changes in the memory after refreshing the memory view in the debugger | | | |
| 9M. | Send "SYSTEM" MEMORY_DUMP 0x102120 24 | 4.3.29, 4.5.4 | TM packet contains the newly programmed data, unrelated bytes in the packet are not changed | | | |
| 9N. | Select a random D-flash address (0x100000 - 0x107FFF) and repeat the steps 9A-9G. NB: The most significant bit has to be converted to '1' in the memory command addresses | | Passed all steps | | | |
| 9O. | Monitor the memory around global address 0x7FC120 (P-flash) with the debugger software | | Memory data and TM packets match | | | |
| 9P. | Send "SYSTEM" MEMORY_DUMP 0x7FC120 24 | 4.3.29, 4.5.4 | TM packet matches the data in P-flash | | | |
| 9Q. | Reprogram the MCU with the CodeWarrior IDE | | Done | | | |
| 9R. | Load default configuration tables into the controller | | Done | | | |
| Step | Activity description | Require ment number | Required result | Actual result | Remarks | Pass/Fail |



Doc.No: FML_S-DREAMS-PR-002_QM-04
 Issue: 1.4
 Date: 08.10.2013
 Page: 16 / 19

Software Test Procedure

| 10. Telecommand and telemetry handling | | | | | | |
|--|---|---|--|--|--|--|
| 10A. | Send "HEX" 0xa1 0x03 0x03 0x03 (Shorter than length field) | 4.2.14, 4.2.18, 4.2.27, 4.3.3 | TM:a1031f01CS | | | |
| 10B. | Send "SYSTEM" STATUS | | TM: a008200000000000 00CS | | | |
| 10C. | Send "HEX" 0xa1 0x02 0x09 0x15 (Wrong CS) | 4.2.16, 4.3.4 | TM:a1030902CS | | | |
| 10D. | Send "SYSTEM" STATUS | | TM: a008200000000000 00CS | | | |
| 10E. | Send "HEX" 0xb0 0x02 0x09 0x09 (Wrong sync pattern) | 4.2.11, 4.2.12, 4.2.25, 4.2.26, 4.2.28, 4.2.29, 4.2.30, 4.3.1 | TM: a0031f03CS | | | |
| 10F. | Send "SYSTEM" STATUS | | TM: a008200000000000 00CS | | | |
| 10G. | Send "HEX" 0xa7 0x02 0x09 0x09 (Wrong address) | 4.1.5, 4.3.2 | TM: a0031f03CS | | | |
| 10H. | Send "SYSTEM" STATUS | | TM: a008200000000000 00CS | | | |
| 10I. | Send 0xa0 0x02 0x0f 0x0f (Undefined command) | 4.2.15, 4.3.5 | TM:a0030f04CS | | | |
| 10J. | Send "HEX" 0xa1 0x03 0x09 0x00 0x09 (Longer than specified for the command) | 4.2.14, 4.2.18, 4.3.6 | TM:a1030904CS | | | |
| 10K. | Send "SYSTEM" STATUS | | TM: a008200000000000 00CS | | | |
| 10L. | Send "DREAMS-P2" POWER_ON | | TM: a203090009 | | | |
| 10M. | Send "DREAMS-P2" MEAS | | TM: a2030b000b | | | |
| 10N. | Send "DREAMS-H" POWER_ON while the measurement is going on (Normal commands are not allowed during measurement) | 4.3.7 | TM: a4030905CS, measurement keeps going on | | | |
| 10O. | Send "SYSTEM" STANDBY 0xAA | | TM: a003010001, measurement stops | | | |
| 10P. | Send "SYSTEM" STANDBY 0x00 (wrong parameter) | | TM: a0030104CS | | | |
| 10Q. | Send "DREAMS-H" POWER_ON | | TM: a403090009 | | | |



Doc.No: FML_S-DREAMS-PR-002_QM-04
 Issue: 1.4
 Date: 08.10.2013
 Page: 17 / 19

Software Test Procedure

| | | | | | | |
|-----------|--|--------|--|--|--|--|
| 10R. | Send "DREAMS-H" REGENERATE 0x01 10 (DREAMS-H powered when requesting heating) | | TM: a4030c06CS | | | |
| 10S. | Send "DREAMS-H" POWER_OFF | | TM: a4030a000a | | | |
| 10T. | Send "DREAMS-H" MEAS (sensor not powered) | 4.3.8 | TM: a4030b06CS | | | |
| 10U. | Send "DREAMS-H" REGENERATE 0x01 10 | | TM: a4030c000c | | | |
| 10V. | Send "DREAMS-H" POWER_ON while the regeneration is going on | | TM: a4030905CS | | | |
| 10W. | Send "DREAMS-P2" POWER_ON | | TM: a203090009 | | | |
| 10X. | Send "DREAMS-P2" TEST_SENSOR_MEAS (test measurement configuration not defined) | 4.3.11 | TM: a2031106CS | | | |
| 10Y. | Send "HEX" 0x11 0xA2 0x02 0x09 0x09 0xA0 0x02 0x08 0x08 (Multiple errors/commands sent in a row) | 4.2.31 | TM: a0031f03CS, TM: a203090009, TM: a003080008, TM: a008200200000000 00CS (Some of the ACK messages might be omitted) | | | |
| 10Z. | Send "SYSTEM" READ_CONF 0x05 (Parameter out of range) | 4.3.10 | TM: a0030304CS | | | |
| 10A A. | Send "SYSTEM" UPDATE_CONF 0x10 CONF_HB (Parameter out of range) | 4.3.10 | TM: a0030204CS | | | |
| 10A B. | Send "DREAMS-P2" POWER_ON and "DREAMS-P2" MEAS | | Done | | | |
| 10A C. | During the measurement, send "HEX" 0xa0 0x02 0x08 0x08 0xa0 0x03 0x01 0xaa 0xab ("SYSTEM" STATUS followed immediately by "SYSTEM" STANDBY 0xAA) | | TM: a003080008, TM: a00820020X00000 002CS, X=0 or 2, Measurement stops (Standby ACK message might be omitted) | | | |
| 10A D. | Send "SYSTEM" STATUS | | TM: a003080008, TM: a008200000000000 0020 | | | |
| 10A E. | Send "SYSTEM" MEMORY_DUMP 0x0F7200 24 | | TM packet received | | | |
| 10AF . | Send "SYSTEM" MEMORY_LOAD 0x0F7200 0xAB 0xCD | | TM: a003040004 | | | |



Doc.No: FML_S-DREAMS-PR-002_QM-04
 Issue: 1.4
 Date: 08.10.2013
 Page: 18 / 19

Software Test Procedure

| 10A G. | Send "SYSTEM" MEMORY_VER 0x0F7300 (Address does not match the address of the earlier MEMORY_LOAD command) | | TM: a0030505CS | | | |
|------------------------------|---|---------------------------|--|------------------|---------|-----------|
| 10A H. | Send "SYSTEM" MEMORY_UPD 0x0F7202 (Address does not match the address of the earlier MEMORY_LOAD command) | | TM: a0030605CS | | | |
| 10AI . | Send "SYSTEM" MEMORY_DUMP 0x0F7200 24 (Check that the memory was NOT updated) | | Verify that the telemetry gives same data as in step 10AD | | | |
| Step | Activity description | Require ment number | Required result | Actual result | Remarks | Pass/Fail |
| 11. Power consumption | | | | | | |
| 11A. | Connect the multimeter set for current measurement in series between the +5V supply positive terminal and the RS-422/power cable +5V positive wire. Disconnect the debugger for these measurements! | | Done | | | |
| 11B. | Power up the controller and measure the idle current | | Fill the result in table 6.8. | | | |
| 11C. | Verify that the "Ver. C" of table 6.7. configuration tables are in the controller | | Done | | | |
| 11D. | Send "DREAMS-P1" POWER_ON | | TM: a103090009 | | | |
| 11E. | Send "DREAMS-P1" MEAS | | TM: a1030b000b | | | |
| 11F. | Measure the +5V current | | Fill the result in table 6.8. | | | |
| 11G. | Send "SYSTEM" STANDBY 0xAA | | TM: a003010001 | | | |
| 11H. | Send "DREAMS-P2" POWER_ON | | TM: a203090009 | | | |
| 11I. | Send "DREAMS-P2" MEAS | | TM: a2030b000b | | | |
| 11J. | Measure the +5V current | | Fill the result in table 6.8. | | | |
| 11K. | Send "SYSTEM" STANDBY 0xAA | | TM: a003010001 | | | |
| 11L. | Send "DREAMS-H" POWER_ON | | TM: a403090009 | | | |
| 11M. | Send "DREAMS-H" MEAS | | TM: a4030b000b | | | |
| 11N. | Measure the +5V current | | Fill the result in table 6.8. | | | |
| 11O. | Send "SYSTEM" STANDBY 0xAA | | TM: a003010001 | | | |
| 11P. | Send "DREAMS-P1 DREAMS-H" POWER_ON | | TM: a503090009 | | | |



Doc.No: FML_S-DREAMS-PR-002_QM-04
 Issue: 1.4
 Date: 08.10.2013
 Page: 19 / 19

Software Test Procedure

| | | | | | | |
|------|--|--|--|--|--|--|
| 11Q. | Send "DREAMS-P1 DREAMS-H" MEAS | | TM: a5030b000b | | | |
| 11R. | Measure the +5V current | | Fill the result in table 6.8. | | | |
| 11S. | Send "SYSTEM" STANDBY 0xAA | | TM: a003010001 | | | |
| 11T. | Send "DREAMS-P2 DREAMS-H" POWER_ON | | TM: a603090009 | | | |
| 11U. | Send "DREAMS-P2 DREAMS-H" MEAS | | TM: a6030b000b | | | |
| 11V. | Measure the +5V current | | Fill the result in table 6.8. | | | |
| 11W. | Send "SYSTEM" STANDBY 0xAA | | TM: a003010001 | | | |
| 11X. | Connect the multimeter set for current measurement in series between the +12V supply positive terminal and the RS-422/power cable positive wire | | Done | | | |
| 11Y. | Repeat the steps 11A-11W for +12V | | Fill the results in table 6.8. | | | |