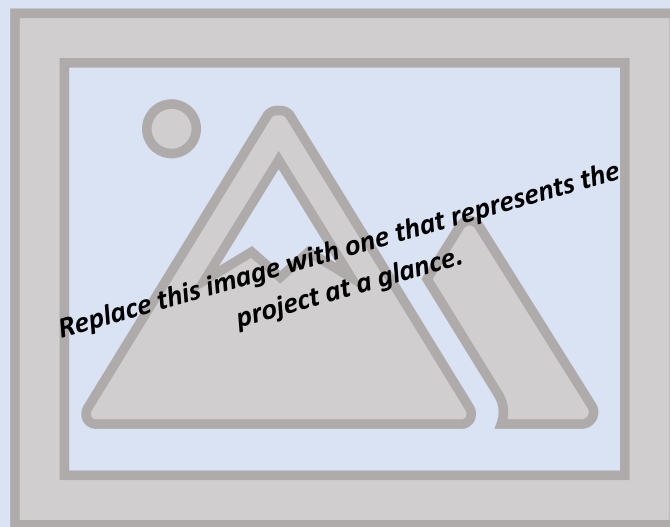


# Product report

---

*Guide for writing a product report*



---

**Embedded Systems Engineering  
Academy Engineering and Automotive  
HAN University of Applied Sciences**

Authors

**Hugo Arends**

Date

**August 2024**

Version

**1.4**

## Revisions

Version	When	Who	What
0.1	May 2022	Hugo Arends	First version of the document.
0.2	June 2022	Hugo Arends	Processed feedback from John van den Hooven and Peter Bijl.
0.3	July 2022	Hugo Arends	Processed feedback from Jeroen Veen.
0.4	Aug. 2022	Hugo Arends	Added summary and textual changes.
0.5	Aug. 2022	Hugo Arends	H6.2 Power consumption added and textual changes.
0.6	Aug. 2022	Hugo Arends	Processed feedback from Danielle van den Hoogen.
1.0	Aug. 2022	Hugo Arends	First version of the document.
1.1	Aug. 2022	Hugo Arends	Textual changes en translated into English.
1.2	Mar. 2023	Hugo Arends	Added pinout overview paragraph to Section 5. References according APA.
1.3	June 2024	Hugo Arends	Move Preliminary research to Appendix.
1.4	July 2024	Hugo Arends	Research rewritten according to Research Skills.

## Preface

*This document is a guide to writing a product report. All pages with a blue background, like this one, describe instructions for writing the various parts of such a product report. Each section is followed by an example in the form of a case study. Such a case study is clearly recognizable because the background of those pages is white. The case study is also the thread running through this manual and a subject that is fascinating to me: a clock. The name of this fictional case study project is **Kairos**.*

*I think it is important to emphasize that there is not one correct product report for a project. There are always multiple ways to describe the same system. What matters is that client(s) and contractor(s) understand each other during the different phases of a project. The product report is a tool for this.*

*The main goals for writing a product report are:*

- *Product (re)production*

*The product report must be complete enough that the product can be (re)built using the documentation. This means that the complete electronic schematics, software listings, connection diagrams of cables and connectors, etc. must be present (in appendices).*

- *Enable maintenance*

*Being able to make changes or expansions in the future without having to redo the design. Choices should therefore be recorded with justification to avoid having to make the same considerations again.*

- *Transfer knowledge to fellow engineers*

*Make sure that the development to be continued by colleagues. Transfer the design to a production department or to a project group that will make the product ready for production. When working in project groups, fellow students must be able to find out all the details of the product to also learn those things in which they themselves were less explicitly involved. They must be able to prepare for the exam with the aid of the product report.*

*This manual was created after I spent 15 years supervising project groups, interns and graduate students in the Embedded Systems Engineering program at HAN University of Applied Sciences. Therefore, the information in this document builds on the fantastic work of my (retired) colleagues. There were two reasons for me to review and supplement their work.*

*First, I felt that the previous product report manuals aspired to too abstract a level of design. Time and again, it proved difficult for students to divide a system into purely functional blocks, with no distinction between hardware and software. This revised manual takes a systems approach, designing an architecture directly from subsystems.*

*Second, there were no manuals available for the final phases of a project, namely realization and testing. I have added those in this manual, including examples.*

*Hugo Arends  
Arnhem 2022*

## Summary

*Description of the starting points, goals to be achieved, what was and was not achieved, **results achieved**; the summary should give an overall impression of the whole assignment and is maximum 1 A4.*

## Summary - Case study Kairos

The Kairos project was conducted for the company *ESE clockworks*. The goal of the project is to realize and document a working prototype that demonstrates how wireless communication and a backup battery could be added to the clocks produced by the company. The goals will be realized through a prototype based on the FRDM-KL25Z development kit.

A preliminary research has been started to determine how the FRDM-KL25Z development kit can be made suitable for a backup battery. This requires the addition of extra components and various modifications to the development kit. In addition, a preliminary research was carried out into the detection of a hand located in front of the clock. A TCRT5000 reflective optical sensor was chosen, mainly because of its favourable cost and form factor.

The main function of the system is to display time. In consultation with the client, a list of functional and technical specifications was drawn up. Also a user interface was designed, including examples of the screen displays, to give a complete picture of what the system would eventually look like.

By means of an architectural diagram, the system was divided into subsystems. This architectural diagram shows the subsystems and the interfaces. The hardware and software specifications of the interfaces are discussed in detail. The main application is designed according to the cyclic executive with interrupt scheduler mechanism.

The realization of the subsystems is discussed in detail by the electrical schematics and calculations for voltages and currents. The realized PCB is also shown and explained. For the purpose of software development, the settings of the Keil  $\mu$ Vision IDE are described. Then the software realization of the switches, the OLED display and the main application is explained using code snippets and measurements with a logic analyzer.

An acceptance test was performed to test the realized prototype against the functional specifications. For this purpose, five test scenarios were documented. Three of the test scenarios were successful, one test scenario was partially successful and one test scenario was not successful.

The main goals of the Kairos project were the use of the FRDM-KL25Z board and the addition of wireless communication for remote time synchronization. In addition, insight was gained into the power consumption of the FRDM-KL25Z board. These main goals were all achieved and a working prototype was delivered. The designed, realized and tested clock that was created during the Kairos project can be called a great success. Thanks to this project, the company *ESE clockworks* is one step closer to the realization of a new generation of digital clocks.

## Contents

Revisions.....	2
Preface.....	3
Summary .....	4
Summary - Case study Kairos .....	5
Contents .....	6
1 Introduction.....	9
1.1 Reason.....	9
1.2 Objective .....	9
1.3 Report structure.....	9
1 Introduction - Case study Kairos .....	10
2 Functional design.....	11
2.1 Functional specifications.....	11
2.2 Technical specifications .....	12
2.3 User interface.....	13
3 Functional design - Case study Kairos .....	14
3.1 Functional specifications.....	14
3.2 Technical specifications .....	16
3.3 User interface.....	16
4 Technical Design.....	19
4.1 Architecture .....	19
4.2 Interfaces .....	20
4.2.1 Power supply .....	20
4.2.2 Microcontroller – Sensor .....	20
4.2.3 Microcontroller – Actuator.....	20
4.2.4 Microcontroller – Communication – PC driver – App .....	21
4.3 Software .....	21
4 Technical design - Case study Kairos .....	22
4.1 Architecture .....	22
4.2 Interfaces .....	22
4.2.1 Power supply .....	22
4.2.2 Microcontroller – RGB LED .....	23
4.2.3 Microcontroller – Switches.....	23

4.2.4	Microcontroller – IR sensor .....	24
4.2.5	Microcontroller – RTC.....	24
4.2.6	Microcontroller – OLED display .....	25
4.2.7	Microcontroller – Bluetooth – Driver – App.....	25
4.3	Software .....	26
5	Realisation .....	28
5.1	Hardware .....	28
5.2	Software .....	28
5	Realisation - Case study Kairos.....	29
5.1	Hardware .....	29
5.1.1	Pinout overview .....	29
5.1.2	Electrical schematic .....	29
5.1.3	PCB design .....	30
5.2	Software .....	31
5.2.1	Keil µVision IDE .....	31
5.2.2	Switches.....	32
5.2.3	OLED display .....	33
5.2.4	Main.....	36
6	Testing .....	37
6	Testing - Case study Kairos.....	38
6.1	Acceptance testing.....	38
6.2	Power consumption.....	38
7	Conclusions and recommendations .....	39
7	Conclusions and recommendations - Case study Kairos .....	40
8	References.....	41
	Appendix A .....	42
	Appendix B .....	43
	Appendix n.....	44
	Appendix A – Making the FRDM-KL25Z suitable for backup battery .....	45
	Research question .....	45
	Research method .....	45
	Research data .....	45
	Conclusion .....	46
	Appendix B – Detecting a hand in front of the clock.....	47
	Research question .....	47

Research method .....	47
Research data .....	47
Conclusion .....	47
Appendix C – Pinout and peripheral overview .....	49
Appendix D – Electrical schematic.....	53
Appendix E – Test scenario's .....	54
Test scenario 1: Displaying the time .....	54
Test scenario 2: Customize date and time .....	54
Test scenario 3: Detecting a hand in front of the clock .....	56
Test scenario 4: TBD .....	57
Test scenario 5: TBD .....	57
Appendix F – Power consumption measurements .....	58
Measurement setup .....	58
No adjustments .....	60
Power supply adjustments .....	61
SWD adjustments .....	63
Notes .....	65
Appendix n.....	66



# 1 Introduction

*The introduction is the entrance to the report. It describes why and how the project will be conducted. The introduction is written using the guidelines described in the book Schrijven voor Technici (Berckel, 2017). This means that the introduction is chapter 1, is written without (inter)headings and consists of three paragraph groups, describing the reason, objective and structure of the report.*

## 1.1 Reason

*The reason is written from the perspective of the client. The first paragraph describes relevant background information and provides an informative situation outline of the current situation. The second paragraph describes the problem or desire of the client. The third paragraph describes the importance/relevance of the project.*

## 1.2 Objective

*The objective includes a main question (propositional or interrogative) and the purpose of the project. It also describes the way in which the main question will be answered, or in other words, the working method of the project. Finally, the preconditions and starting points are described.*

## 1.3 Report structure

*Description of the contents of the report by chapter, starting with chapter 2. After reading this paragraph, the main thread of the report is clear.*

# 1 Introduction - Case study Kairos

This project is commissioned by the company *ESE clockworks*. The company specializes in making digital clocks based on NXP KL2 series microcontrollers. These clocks have an OLED display that shows the date and time and push buttons to set that date and time.

Through this project, *ESE clockworks* wants to explore two new technologies for them. First, the company is considering adding wireless communication to the clocks so that the time can be set remotely rather than by pushbuttons alone. Second, the company is considering adding a backup battery to the clocks so that the time will continue to count if the clock loses power.

Based on the knowledge and insights gained from this project, *ESE clockworks* can make future-proof decisions for their latest generation of clocks. Since this is the right time to design and realize this new functionality, this project has been named **Kairos**<sup>1</sup>.

The goal of the project is to realize a working prototype. The knowledge gained, particularly in the field of wireless communication and a backup battery, will be extensively documented.

The goal will be achieved by a project-based approach. A team of five embedded systems engineers will be assembled to carry out the project, with one of them representing the role of project leader. First, a plan of approach is drawn up, which describes in detail what exactly is to be delivered and how this is to be done. This plan of approach is discussed with the client and constitutes a go/no-go decision moment. The total project has a lead time of six months and will be executed and documented according to the V-model. There will be regular consultation with the client to monitor progress and adjust the project goals if needed.

*ESE clockworks* requires that the prototype will be realized with a FRDM-KL25Z development kit. This development kit has excellent low power characteristics and the possibility to use a backup battery. In addition, the current digital clocks from *ESE clockworks* show the date and time on a 64x128 OLED screen with a diameter of 0.96 inches. The same display should be used in this project for compatibility reasons. Synchronizing the time must be done wirelessly. Which medium will be used for this (infrared, bluetooth, wifi, etc.) will be determined in a later phase of the project. This means that the clock is basically a receiver, but that a transmitter must also be realized. The focus for this project is, however, the receiver. The transmitter is a PC or laptop on which already existing applications may be used to send the time. Which applications that will be is determined in the design phase of this project. Thus, no PC application needs to be developed. Since the prototype will also be used for promotional purposes, *ESE clockworks* wanted a nice gimmick to be added: if someone holds his hand in front of the clock, the company name must be shown on the display.

To answer the main question of this report, Chapter 2 Preliminary study describes questions and answers that are important at the start of the project. Chapter 3 describes the functions of the system. In other words, **what** the system should be able to do. In Chapter 4 the system is designed by its main functions. Here it is described **how** the system is realized. The hardware interfaces are specified, as well as the chosen software architecture. Then Chapter 5 describes in detail how the hardware and software of this system were realized. The tests performed with the prototype are described in chapter 6. And finally, in Chapter 7, the conclusions and recommendations follow.

---

<sup>1</sup> [https://nl.wikipedia.org/wiki/Kairos\\_\(mythologie\)](https://nl.wikipedia.org/wiki/Kairos_(mythologie))

## 2 Functional design

*The purpose of creating a functional design is to obtain a complete specification of the system to be developed in consultation with the customer/client. You look with the eyes of the customer. A good functional design is indispensable in the development of a product. It must meet the requirements and wishes of the customer/client. If the customer is not satisfied, all the work has been for nothing. When writing the functional design, keep in mind that it describes **what** the product to be developed should do. Not **how** it should do it.*

*The executor of the project translates the specifications from the customer/client into a detailed functional specification. A distinction is made between purely functional specifications and technical specifications. The trick is to apply enough detail when drawing up the specifications so that the client and contractor understand each other, but not too much detail either, because that would be at the expense of the project's lead time. It is often impossible to capture all the details precisely. It is then advisable to agree to work in iterations, with the specifications being gradually updated.*

### 2.1 Functional specifications

*The functional specification precisely defines all the functions that the system to be realized must be able to perform. The basis for this is laid down in the introduction chapter. The detailed description is done in this section using the SMART criteria (contributors, SMART criteria, 2022). Ideally, each specification should therefore meet the following criteria:*

**Specific** - Specifications should be specific and not generic. They should not be open to misinterpretation when read by others.

**Measurable** - Specifications should be quantifiable to verify completion. Prevent a specification from being fully verifiable by avoiding the use of non-quantitative terms (best, optimal, fastest).

**Assignable** - Ensure that the specification can realistically be achieved given existing conditions and available resources.

**Realistic** - Specifications must be relevant to a project. While all specifications are important, they are prioritized to indicate early on which specifications will add the most value to the company. Priorities are assigned according to the MoSCoW method (contributors, MoSCoW method, 2022). Engineers will initially try to achieve all Must-have, Should-have and Could-have specifications, but the Could-have and Should-have specifications will be the first to be released if the project lead time is compromised. The Won't-have specifications provide the opportunity to delineate the project.

**Time-related** - Each specification must be time-bound or specify when or how quickly a requirement must be completed or executed.

*Use a table to display the functional specifications in a clear and grouped manner. A maximum of three levels are applied to make a functional specification progressively more in-depth and thus more SMART. Also give each specification an identification number so that it can be easily referred to in the rest of the report.*

SMART functional specification		
#	MoS CoW	Description
<b>F1</b>	<b>M</b>	<b>SMART functional specification of the highest level.</b>
F1.1	M	SMART functional sub specification with a more specific elaboration.
F1.2	M	SMART functional sub specification with a more specific elaboration.
<b>F2</b>	<b>S</b>	<b>SMART functional specification of the highest level. This specification is so clearly described that no more specific elaboration is needed.</b>
<b>F3</b>	<b>M</b>	<b>SMART functional specification of the highest level.</b>
F3.1	<b>M</b>	SMART functional sub specification with a more specific elaboration.
F3.1.1	<b>M</b>	SMART functional sub-sub specification with more specific elaboration.
F3.1.2	<b>C</b>	SMART functional sub-sub specification with more specific elaboration.
F3.2	<b>C</b>	SMART functional sub specification with a more specific elaboration.
F3.2.1	<b>C</b>	SMART functional sub-sub specification with more specific elaboration.
F3.2.2	<b>C</b>	SMART functional sub-sub specification with more specific elaboration.
F3.3	<b>S</b>	SMART functional sub specification with a more specific elaboration.
F3.3.1	<b>S</b>	SMART functionele sub-sub specificatie met een specifiekere uitwerking.
F3.3.2	<b>S</b>	SMART functionele sub-sub specificatie met een specifiekere uitwerking.
<b>F4</b>	<b>M</b>	<b>SMART functional specification of the highest level. This specification is so clearly described that no more specific elaboration is needed.</b>

## 2.2 Technical specifications

*The technical specifications contain additional specifications that the customer/client requires of the product. This can be, for example, the type of microcontroller or the programming language with which the product must be realized. Use a table to display the technical specifications in a clear and grouped way. Also give each specification an identification number so that it can be easily referred to in the rest of the report.*

SMART technical specification		
#	MoS CoW	Description
<b>T1</b>	<b>M</b>	<b>SMART technical specification of the highest level.</b>
T1.1	M	SMART technical sub specification with a more specific elaboration.
T1.2	M	SMART technical sub specification with a more specific elaboration.
<b>T2</b>	<b>M</b>	<b>SMART technical specification of the highest level. This specification is so clearly described that no more specific elaboration is needed.</b>
<b>T3</b>	<b>C</b>	<b>SMART technical specification of the highest level. This specification is so clearly described that no more specific elaboration is needed.</b>

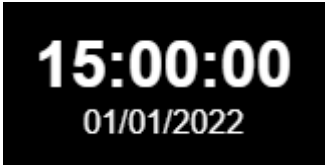
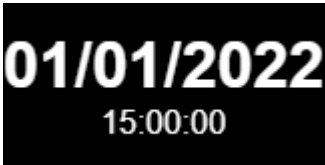

## 2.3 User interface



*Provide an sketch of the appearance of the system to be developed. What will the system look like to the user? What inputs are there and what outputs? What will change in the outputs if a particular input changes? A user interface sketch is another representation of the same system as described in the previous sections. It is a communication tool between the client and the contractor to clarify the functionality of the system to be realized.*

### 3 Functional design - Case study Kairos

#### 3.1 Functional specifications

The main function of the system is to display time. In consultation with the client, the following list of functional specifications was prepared using the SMART criteria (contributors, SMART criteria, 2022). The specifications are prioritized in the second column according to the MoSCoW method (contributors, MoSCoW method, 2022).

SMART functional specification		
#	MoS CoW	Description
<b>F1</b>	<b>M</b>	<b>The clock displays the time.</b>
F1.1	M	The time is displayed on a 64x128 OLED screen.
F1.2	S	Time is displayed in three different ways. Fonts and other display options will be agreed upon with the client once an initial prototype is available.
F1.2.1	M	The clock shows the time large and the date small. 
F1.2.2	M	The clock shows the date large and the time small. 
F1.2.3	S	The clock shows the time by hands and the date. 
F1.3	M	A switch (working name SW1) is used to select the screen mode as described in F1.2. The switch cycles, so to say, through the different display modes. So after the last display option is selected, the first display option is shown.
<b>F2</b>	<b>M</b>	<b>The time can be adjusted on the clock.</b>
F2.1	M	A switch (working name SW2) activates the setup. In the setup the date and time can be set.

SMART functional specification		
#	MoS CoW	Description
F2.2	M	<p>As soon as SW2 is pressed, the first setup option appears. A setup screen displays one of the numbers to be set. Pressing SW1 increases the value by one. If the value goes out of range, the first value of the respective range is displayed.</p> 
F2.3	M	Pressing SW2 moves to the next setup option.
F2.4	M	The setup options are selected in the following order: day → month → year → hours → minutes → seconds. Exiting the last option returns to the last display mode.
<b>F3</b>	<b>S</b>	<b>The clock shows the name of the company.</b>
F3.1	S	<p>As soon as a hand (or other object) is within 10cm of the front of the clock, the name of the company is displayed.</p> 
F3.2	S	Once there is no object within 10cm of the front of the clock, the company name will remain on the screen for about 5 seconds. After that, the screen mode will resume as it was before the company name was displayed.
<b>F4</b>	<b>M</b>	<b>The clock can synchronize time wirelessly using an app.</b>
F4.1	M	The app is a terminal application running on a laptop that sends the current date and time. As an extension to this project, a graphical application may be developed at a later stage.
F4.2	C	The distance between laptop and clock is at least 3 meters.
F4.3	W	No information is sent from the clock to the app. However, it should be considered in the design that this will be possible in the future.
<b>F5</b>	<b>M</b>	<b>The clock independently keeps track of time.</b>
F5.1	M	Time is tracked in seconds accurately.
F5.2	M	The accuracy of tracked time is plus or minus one minute per day.
<b>F6</b>	<b>M</b>	<b>The status of the time is displayed.</b>

SMART functional specification		
#	MoS CoW	Description
F6.1	M	An LED flashes every second (100ms on, 900ms off).
F6.2	S	<p>The color of this LED depends on the following:</p> <ul style="list-style-type: none"> <li>• The time is <i>not</i> set by the user and <i>not</i> synchronized by the app, then the LED color is red.</li> <li>• The time is <i>not</i> set by the user and <i>is</i> synchronized by the app, then the LED color is green.</li> <li>• If the time <i>is</i> set by the user and <i>not</i> synchronized by the app, the LED is off.</li> <li>• If the time <i>is</i> set by the user and <i>is</i> synchronized by the app, the LED color depends on which of the two actions happened last. The color is then green if the last action was sync by the app and off if the last action was set by the user.</li> </ul>

### 3.2 Technical specifications

The following table shows the technical specifications of the project.

SMART technical specification		
#	MoS CoW	Description
T1	M	The FRDM-KL25Z development kit is used.
T2	M	The hardware is realized in the form of a shield for on the FRDM-KL25Z development kit. A housing is not realized.
T3	M	Programming is done in the C language according to the C17 standard.
T4	M	A backup battery is used to keep the time up to date if there is no other power source. To save energy, all other functions may be turned off, such as the OLED screen.

### 3.3 User interface

To give an idea of what the clock will look like, a mockup view is shown in Figure 1. In consultation with the client, the location of the OLED display was determined. The switches will be placed at the bottom of the PCB. The placement of the other components is not fixed and will also depend on the final component selection.



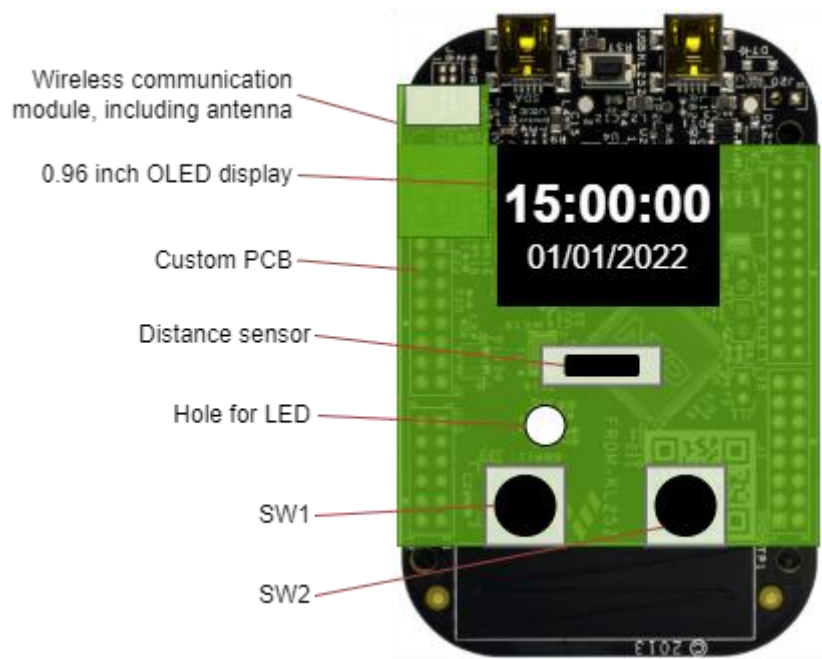


Figure 1. Overview of the user interface.

The image of the FRDM-KL25Z board is taken from <https://www.nxp.com/design/development-boards/freedom-development-boards/mcu-boards/freedom-development-platform-for-kinetis-kl14-kl15-kl24-kl25-mcus:FRDM-KL2>.

Figure 2 shows what will be shown on the OLED display when SW1 is pressed.



Figure 2. The different ways time can be displayed.

Figure 3 shows an example of setting the time. Pressing SW2 starts the setup and 'freezes' the time, as it were. In the example, SW2 is then used to select the various setup options. Arriving at the hours, SW1 is pressed three times to increase the number of hours by three. Then the setup is closed by pressing SW2 three times. The adjusted time is displayed on the screen and the seconds start counting again.

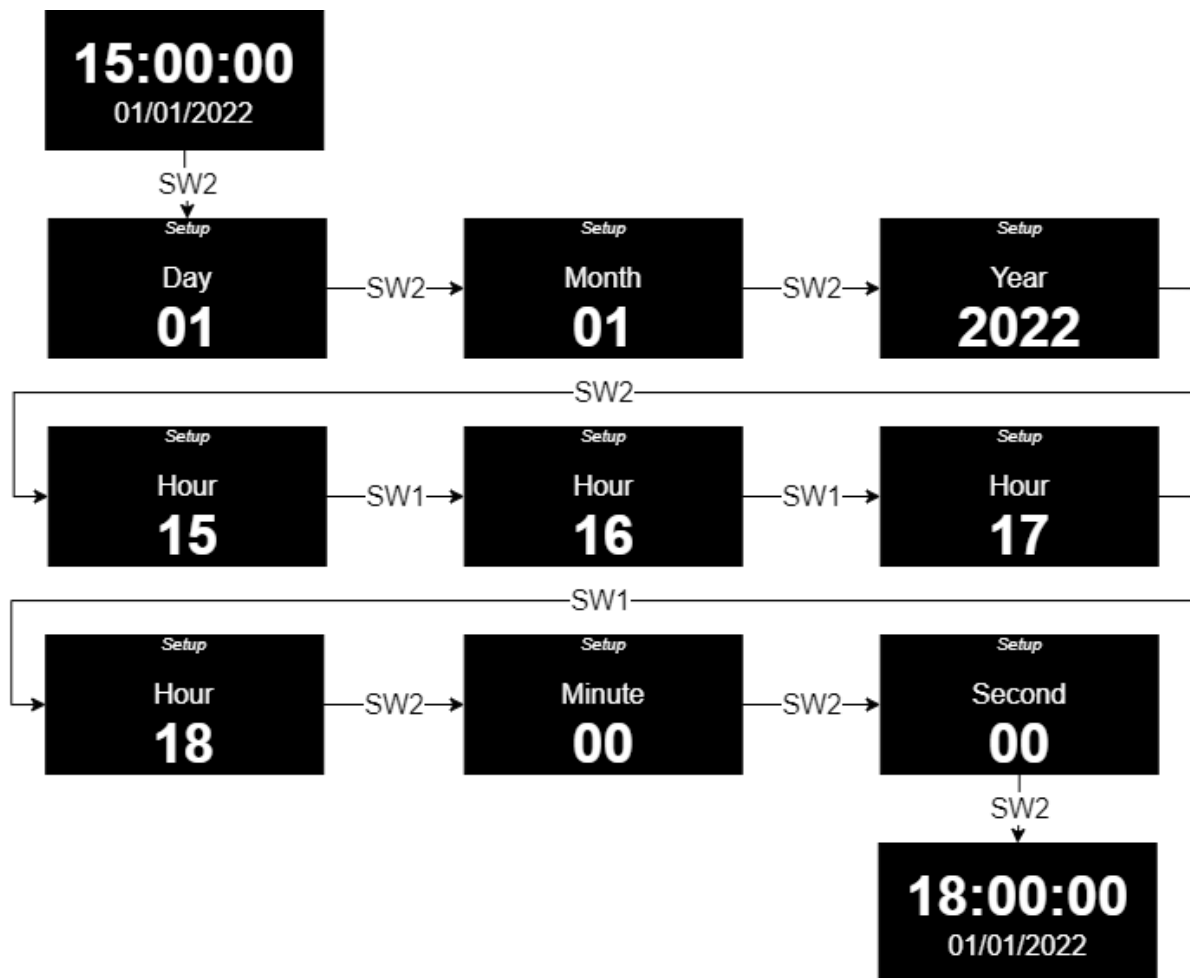


Figure 3. The setup options and an example of changing the time.

## 4 Technical Design

In this phase, thought is given to the way of realizing the various functions. This is about the **how** of the product to be developed. It is not the intention to describe new functional or technical specifications. That is the **what** and is described in the previous chapter. The purpose of the technical design is to translate the functional design into a technical implementation. You are now looking with the eyes of the designer.

### 4.1 Architecture

The system is first divided into subsystems. Each subsystem has strong internal coherence and relatively little interaction with the other subsystems. The coherence between the subsystems is represented in an architecture diagram. A reasoned choice is made for the interface between the subsystems based on the functional and/or technical specifications.

Figure 4 shows an architecture that is generally applicable to embedded systems. The heart is the microcontroller that communicates through various interfaces with the subsystems. These interfaces must be unambiguously defined at this stage of the project.

Sensors have an arrow towards the microcontroller, e.g. for analog measurement, but can also have a double arrow, e.g. for a serial bus like I2C. The latter also applies to actuators, but actuators can also be realized with a single arrow towards the actuator. Think for example of a PWM signal.

A commonly used subsystem is communication with another device, such as a laptop, smartphone, etc. That interface usually also communicates in two directions, but it does not have to.

It is common to take into account that the microcontroller needs to be able to provide software updates. Therefore, there is often a programming and/or debugging subsystem. There are several interfaces that are common for debugging, such as SWD and JTAG.

Finally, the diagram shows how the power supply is controlled. From a voltage source, a voltage converter is used to realize the supply voltage ( $V_{dd}$ ) for the embedded system. To keep the schematic clear,  $V_{dd}$  is not drawn to all subsystems. It may be necessary to have multiple voltage levels available in the embedded system, for example to control motors.

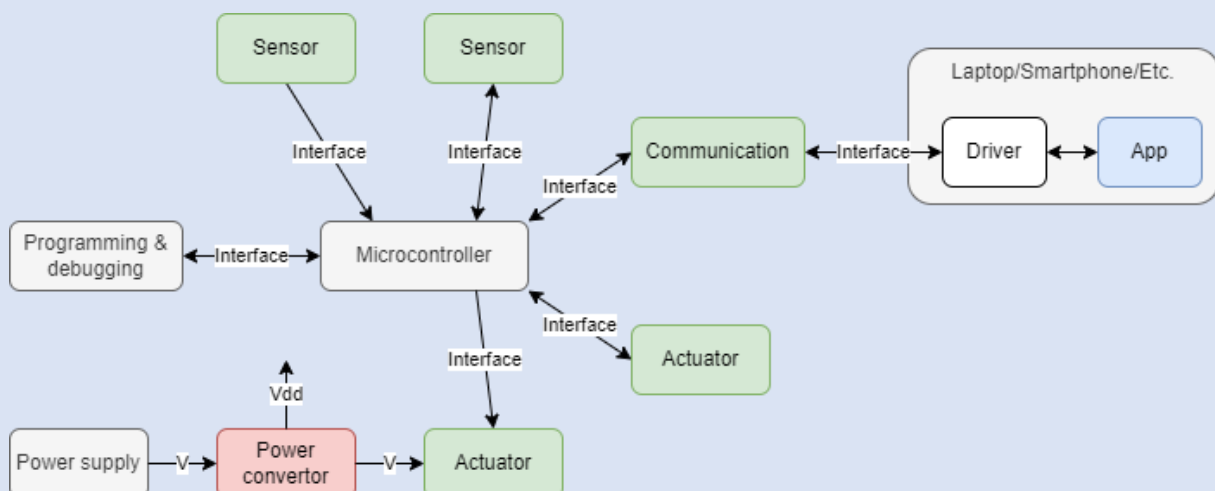


Figure 4. General architectural diagram for an embedded system.

## 4.2 Interfaces

For each interface, the electrical and/or data communication properties are described. Sometimes these choices are dictated by the (technical) requirements, but often you have to make choices here as a designer. Also for each interface between the microcontroller and other modules a design is made for the software driver by means of a UML sequence diagram.

### 4.2.1 Power supply

The power supply specification specifies what voltages are needed in the system, what voltage sources there are, how they are converted, and what maximum current can be expected. A specification is formulated in a clearly recognizable manner:

**Specification** Here comes the text of the specification.

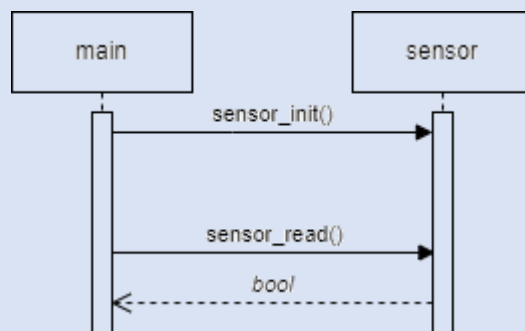
### 4.2.2 Microcontroller – Sensor

For sensors, a description of what the sensor measures will be given. Where possible, the choice is linked to a specification. It is important to be as complete as possible, thinking of quantities, units, range, precision, sample frequency, etc. A specification will be clearly recognizable:

**Specification** Here comes the text of the specification.

It is also described that a software driver is realized. A driver for a sensor has at least one function to initialize the driver and one or more functions to read values from the sensor. Optionally, a function for writing to the sensor can be described, for example, to write configuration parameters. As a prefix for the names of the functions, choose the names that were also used in the architectural diagram, or an abbreviation of them.

**Specification**



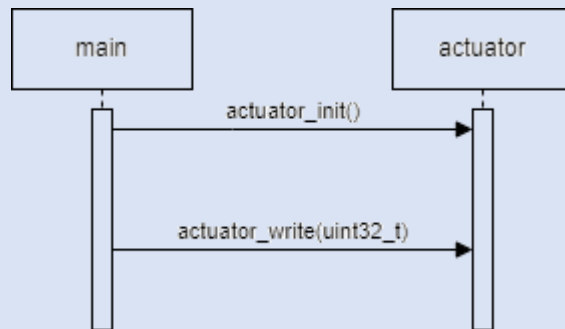
### 4.2.3 Microcontroller – Actuator

For actuators, the output signal is described in as much detail as possible. Here, where possible, the choice is linked to a specification. Again, think of quantities, units, range, precision, frequency, etc. A specification is clearly recognizable:

**Specification** Here comes the text of the specification.

It is also described that a software driver will be realized. A driver for an actuator has at least a function to initialize the driver and one or more functions to write values to the actuator. Optionally, a function to read the actuator can be specified, for example to read the state of an actuator. As a prefix for the names of the functions, choose the names that were also used in the architectural diagram, or an abbreviation of them.

### Specification



#### 4.2.4 Microcontroller – Communication – PC driver – App

The specification of communication with other devices has two parts: the interface(s) and the data format.

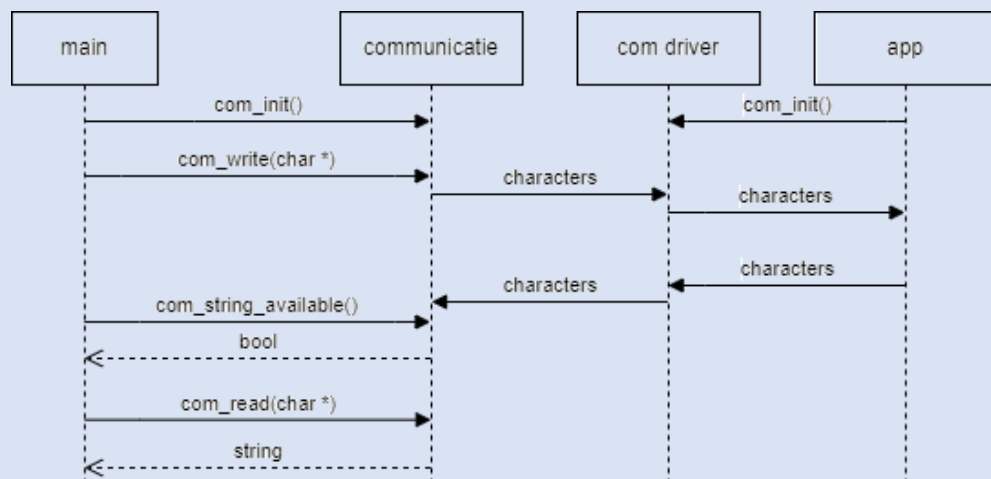
With respect to the interface, the following must be specified:

- electrical – voltage, current, etc.
- protocol – RS232, I2C, parallel, etc.
- protocol settings – such as bitrate, etc.

In addition, it must be unambiguously defined how data will be exchanged between the microcontroller main and app, or in other words the data format. Will an existing data format (such as JSON, XML, CSV, etc.) be used, or will a self-devised data format be implemented? In the case of the latter, then that data format must be unambiguously specified in this section.

It is also described that a software driver will be realized. A driver for communication is realized for a microcontroller, but not for the PC. The latter is usually available. A communication driver for a microcontroller has an initialization function, a write function and a read function. The function parameters depend on the selected data format.

### Specification



#### 4.3 Software

For the main program, one or more software designs are shown and described. There are several methods to describe such a design, such as a flowchart, state diagram, sequence diagram, class diagram, etc. From the descriptions it should be clear what architecture has been chosen, e.g. event driven, cyclic executive with interrupts, an RTOS, a state machine, or something similar.

## 4 Technical design - Case study Kairos

### 4.1 Architecture

The architecture diagram for the clock is shown in Figure 5. The time is kept locally by a real-time clock (RTC) module. The OLED display shows that time. The time is synchronized wirelessly with the app via bluetooth. The RGB LED shows the synchronization status with colours, but the blue LED is not used. Two switches allow the user to adjust the display mode. To detect if the user holds a hand in front of the clock, an infrared sensor is used.

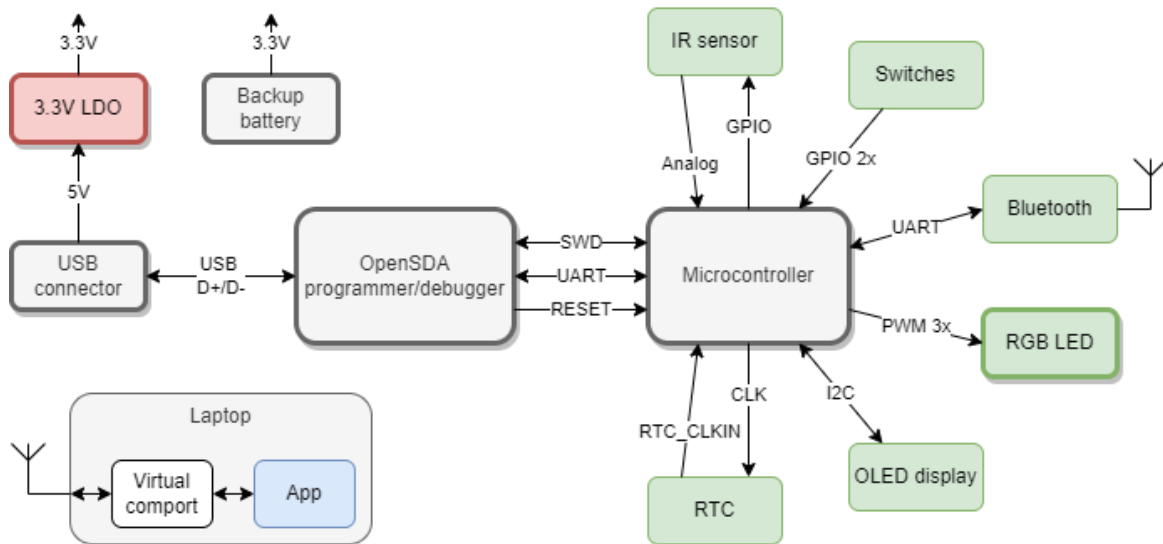


Figure 5. Architectural diagram of the clock.

According to Technical Specification T1, the FRDM-KL25Z development kit should be used. The functionality of the blocks in Figure 5 with a thick edge are already present on this development kit, including an MKL25Z128VLK4 microcontroller. The specifications of the board are described in the user manual (Freescale Semiconductor, Inc., 2013). All information about the microcontroller is described in the datasheet (Freescale Semiconductor, Inc., 2014) and the reference manual (Freescale Semiconductor, Inc., 2012).

Through the OpenSDA programmer/debugger, the microcontroller can be programmed with software updates. The CMSIS-DAP debug application will be programmed on this (NXP, 2022). This application implements, besides a programmer/debugger, also a USB CDC interface that allows serial communication between a host and the microcontroller. This capability will be used for debug purposes during the realization of this project.

To keep power management as simple as possible, all subsystems must operate at 3.3V. That 3.3V is provided by a low dropout (LDO) voltage regulator and/or a backup battery. To keep the architecture diagram clear, the 3.3V arrows to all subsystems have been omitted.

A detailed description of all interfaces follows in the next section.

### 4.2 Interfaces

#### 4.2.1 Power supply

The system has two power sources. The first is 5V through the USB connector. A low dropout (LDO) voltage regulator lowers this to 3.3V.

**Specification** The LDO must reduce a voltage of 5V to 3.3V. The maximum current is 500mA.

The second power source is the backup battery. This battery is directly connected to the same 3.3V supply rail, i.e. without a voltage regulator, as described in the preliminary research in section 0.

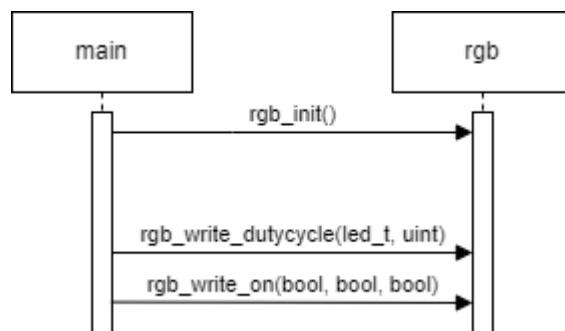
#### 4.2.2 Microcontroller – RGB LED

The RGB LED on the FRDM-KL25Z board is very bright when turned on with a digital output pin. Three PWM signals are therefore generated so that the LEDs can be dimmed.

**Specification** *Each PWM signal has a frequency of at least 100Hz and an adjustable duty cycle of 100 steps.*

A driver is realized to configure the hardware. It is obvious to use a timer for this. Per LED it must be possible to configure the desired duty cycle. In addition, a convenience function is created that can be used to control with a single function whether each LED should be on or off. For this a fixed duty cycle, which is to be determined, will be set.

**Specification**



#### 4.2.3 Microcontroller – Switches

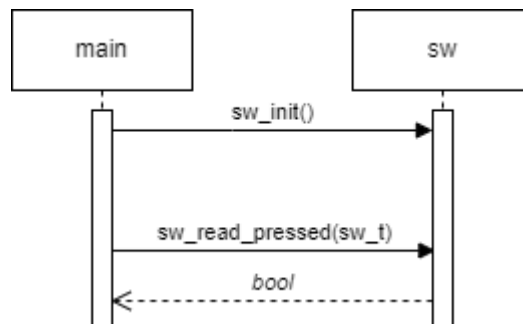
The switches are connected to two digital input pins. Both switches are low active.

**Specification** *Both switches are low active:*

- *Not pressed, the digital input of the microcontroller is logic 1 (3.3V).*
- *Pressed, the digital input of the microcontroller is logic 0 (GND).*

A driver is realized to configure the hardware. In addition, a function is realized where the parameter indicates from which switch the state should be read. The returned state is true if the switch is pressed (logic 0) and false if the switch is not pressed (logic 1).

**Specification**



The driver reads the instantaneous value of the switch based on polling. That means that if the switch is pressed between two read operations, then this is not remembered by the driver. So the main must poll the switches fast enough.

**Specification** *The state of both switches is read 10 times per second.*

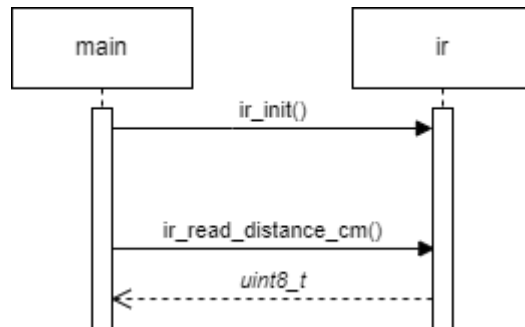
#### 4.2.4 Microcontroller – IR sensor

According to Functional Specification F3, the clock must be able to detect a hand held at different distances from the clock. From the preliminary investigation described in section 0, it shows that the TCRT5000 reflective optical sensor from Vishay (Vishay Semiconductors, 2017) is the most suitable for this purpose.

**Specification** *It measures 10 times per second whether a hand is detected and if so, its distance in cm.*

A driver is realized that configures the hardware and returns the distance in cm.

**Specification**



#### 4.2.5 Microcontroller – RTC

The selected microcontroller has a built-in RTC module. This module must be provided externally with a 32.768 kHz clock signal. It is possible to generate that clock signal with the microcontroller. However, an external connection must then be made between two pins of the microcontroller, namely PTC1 and PTC3. Details of this are described in an online thread of the NXP Community (Adrián Sánchez Cano, 2013).

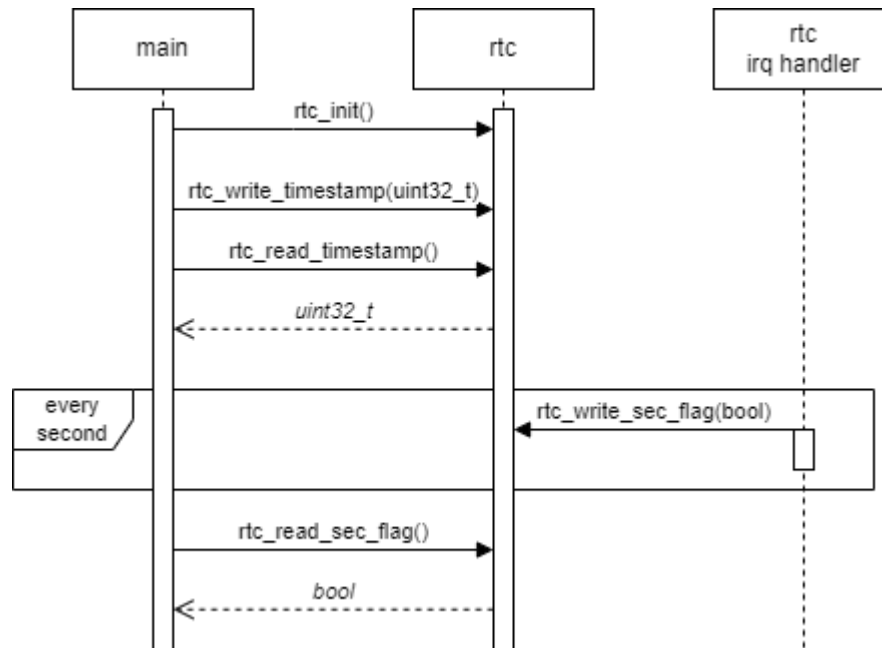
**Specification** *In order to use the internal RTC, PTC1 and PTC3 must be externally connected.*

A driver is realized to configure the hardware. The driver generates an interrupt every second. This interrupt sets a global flag that is used in `main` for various purposes, including updating the display.

The driver also gives the possibility to read and write the unix timestamp. The unix timestamp is stored as a 32-bit unsigned integer.



### Specification



#### 4.2.6 Microcontroller – OLED display

According to functional specification F1.1, the date and time must be displayed on a 128x64 OLED display. The selected OLED display is controlled with an I2C interface and contains an SSD1306 single-chip OLED driver from Solomon Systech. The documentation of this SSD1306 (Solomon Systech Limited, 2008) describes how to drive the OLED display.

During the microcontroller classes at the HAN such an OLED display has already been worked with and a driver is available. No design is shown in this section for that reason. Experiments have been done with this driver, see Appendix X, and this has shown that the driver implements all functions (and more) to realize the functional requirements.

An important feature of this driver is that it works on a frame buffer basis. This means that the data to be displayed is first stored locally in the microcontroller in an array, before it is sent to the OLED display via I2C. This sending is done polling based.

#### 4.2.7 Microcontroller – Bluetooth – Driver – App

According to functional specification F4, wireless communication with an app running on a laptop is required. A serial connection was chosen, with communication realized via bluetooth. This means that the microcontroller communicates via a bluetooth module and the laptop must have bluetooth and the appropriate drivers to open a virtual com port. The microcontroller communicates with the bluetooth module via a UART.

**Specification** *Serial communication is configured at 9600 bps, 8 data bits, 1 stop bit and no parity.*

Communication between microcontroller and app is done based on ASCII strings. A string consists of several characters and is terminated with an LF character ('\n'), a CR character ('\r') or both. The purpose of this is synchronization. This means that incoming characters must be buffered and only processed when CR/LF characters are detected.

**Specification** *The maximum number of characters to be buffered is 80.*

All characters in a string are readable ASCII characters. This makes it possible to send and receive a message through a terminal application.

**Specification** *All communication between microcontroller and app is done in variable length frames. A frame is terminated with a CR, LF or with both characters and then the previous data is processed.*

According to functional specification F4.1, the time is sent by the app to the microcontroller. A unix timestamp<sup>2</sup> is used for this purpose.

**Specification** *The time is sent from the app to the microcontroller in the form of the unix timestamp.*

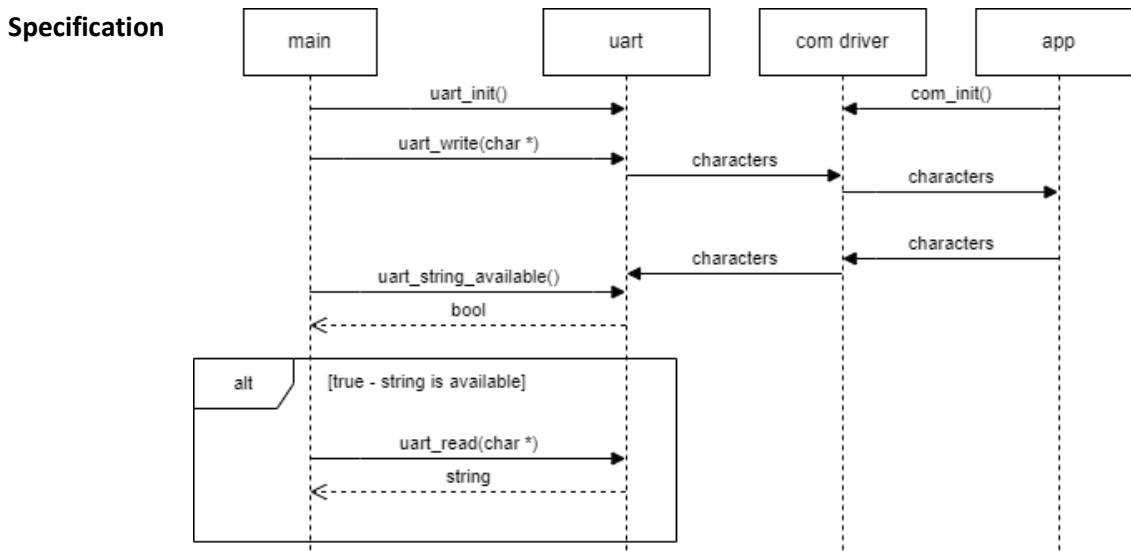
Examples of valid unix timestamp strings are:

- "0\n"
- "10000\r\n"
- "1650033178\n\r"
- "1640991600\n"

For the microcontroller, a driver is realized that configures the hardware. To meet functional specification F4.3, the driver provides a function to transmit a string of characters. The driver does not itself add a CR/LF character to a string; that will have to be done in main.

The driver implements a function to check if the CR/LF character has been received. If so, the driver provides a function to read all characters received so far. All received CR/LF characters are discarded and the string terminator character '\0' is added at the end.

The app initializes the com driver. The characters of the unix timestamp string are sent by the app. It doesn't matter if that happens character-by-character, or if all characters are sent as soon as Enter is pressed, for example. All incoming characters are literally displayed in the app.



## 4.3 Software

The clock main is implemented via a *cyclic executive with interrupt scheduler*. This architecture provides a modular approach and the microcontroller can be put in a low-power mode when no code

<sup>2</sup> <https://www.unixtimestamp.com/>

needs to be executed. A pure event-triggered architecture is not preferable because too many peripherals can become active at the same time, which would cause low-priority events to wait too long for processing.

In the chosen architecture, the main loop will be executed 10 times per second. At each loop, the value of a sensor is checked and whether new data has been received. If yes, then in main the appropriate action will be performed. If all actions have been performed and less than 100ms has elapsed, the microcontroller will enter a low power mode. In Figure 6 this procedure is shown in a flowchart.

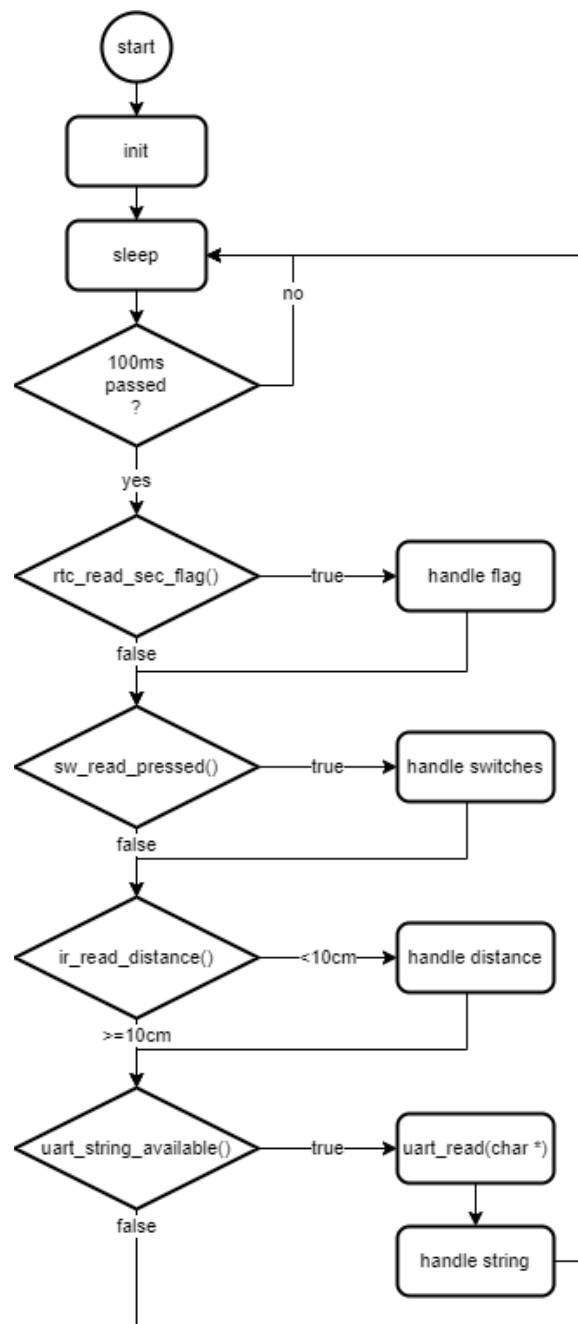


Figure 6. Flowchart of the main-loop.

## 5 Realisation

*Details of the realized hardware and software with accompanying explanations and calculations (such as power consumption, values of components, etc.). Complete and detailed diagrams of the hardware and listings of the software are included in the appendices.*

### 5.1 Hardware

*The realized hardware is explained by means of wiring diagrams. It is also clarifying to include a picture of, for example, a realized PCB. Preferably use images of a part of the wiring diagram. Not everything needs to be explained. Choose two or three of the most relevant subsystems. The complete wiring diagram must be included in the appendices.*

### 5.2 Software

*The realized software is explained by means of code snippets. Make sure the code is easy to read by using syntax highlighting. Use code snippets that are no longer than 20 lines and that each line of code fits on one line in the report. Not all of the realized code needs to be explained. Choose two or three of the most relevant subsystems. The full code is included as an appendix. Also pay attention to the software development environment. In doing so, ask yourself what is important information for a fellow engineer who will be using the same development environment for the first time.*

## 5 Realisation - Case study Kairos

### 5.1 Hardware

The hardware of the clock is described using the pinout overview, electrical schematics and PCB design.

#### 5.1.1 Pinout overview

Because the FRDM-KL25Z development kit is used, the number of available IO pins is limited to the IO pins present on the headers. The overview described in (Freescale Semiconductor) was used as a starting point to create the table in [Appendix C](#). The table shows in an overview which pins are used for which function and which peripherals are in use.

#### 5.1.2 Electrical schematic

The complete electrical schematic of the clock is given in [Appendix D](#). This section explains the most important sub-diagrams.

##### 5.1.2.1 Switches

The signals SW1 and SW2 are connected to the pins PTD3 and PTD5 of the microcontroller. No pullup resistors were applied, because they are internally available in the microcontroller. Also, no anti-bouncing hardware was realized, because the software controls the logical value of the switches based on polling.

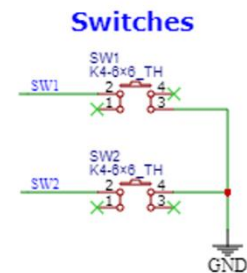


Figure 7. Hardware schematic switches.

##### 5.1.2.2 RTC

For the use of the RTC module in the microcontroller an external connection must be made between PTC1 and PTC3. This connection is realized by means of a jumper, so that the pins could also be used for other purposes. Strictly speaking, laying a trace between these two pins is sufficient.

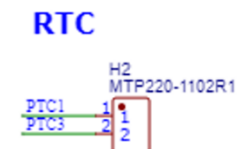


Figure 8. Hardware schematic RTC.

##### 5.1.2.3 OLED display

The 64x128 OLED module contains a 4-pin male header. On this module there are already pullup resistors for the I2C signal, so those are not added to this design as well.

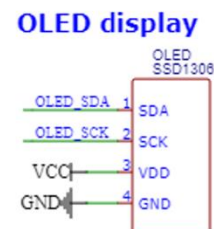


Figure 9. Hardware schematic OLED display.

#### 5.1.2.4 Bluetooth module

There are several suitable bluetooth modules, including the HC05 and the HC06. Initially, an HC06 is chosen, but if it proves to be poorly available, then the HC05 pin is compatible, except for the power supply voltage. Therefore, via a solder bridge, the possibility is realized to determine during assembly what power supply voltage is used by the bluetooth module.

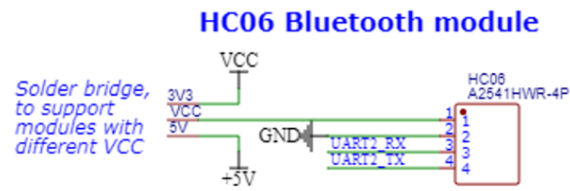


Figure 10. Hardware schematic bluetooth module.

#### 5.1.2.5 IR sensor

The LED of the IR sensor is turned on by the microcontroller with a low active signal called IR\_LED. The voltage drop across the LED is 1.25V, as described in the TCRT5000 datasheet (Vishay Semiconductors, 2017). Thus, at a supply voltage of 3.3V, there is  $3.3V - 1.25V = 2.05V$  across resistor R2. By choosing 100 Ohms for R2, the current through the LED becomes equal to  $\frac{2.05V}{100\Omega} = 20.5mA$ . This is well within the range of 60mA indicated in the datasheet as the maximum forward current.

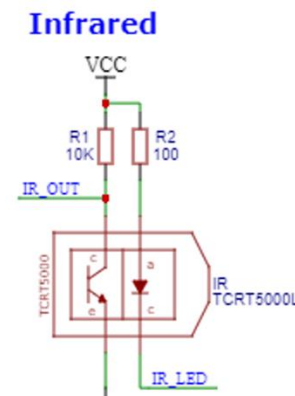


Figure 11. Hardware schematic IR sensor.

According to the datasheet, the collector current is allowed to be a maximum of 100mA. With 10k Ohms and 3.3V, this is also well within the limit.

If no infrared light is reflected, the transistor will not conduct and the voltage on IR\_OUT will be nearly equal to VCC. The more the transistor conducts as a result of reflection by the infrared LED, the lower the voltage on IR\_OUT will be. This output voltage is connected to PTB0 (not shown in Figure 11, it is in [Appendix D](#)), so that it can be read with single ended channel 8 of ADC0.

#### 5.1.3 PCB design

The first stage of the PCB design is the placement of the components. This is shown in Figure 12. All major components are placed as specified by the client (see also Figure 1), except the header for the Bluetooth module. On second thought, it was more convenient to place the header on the right side of the PCB, because then the USB connector of the FRDM-KL25Z development kit remains easily accessible.

Next, the PCB traces were laid out. Because of its relative simplicity, a two-layer PCB was chosen. To simplify the placement of the PCB traces, the PCB traces on the top layer are primarily in a horizontal direction and the PCB traces on the bottom layer are in a vertical direction. The result is shown in Figure 13.

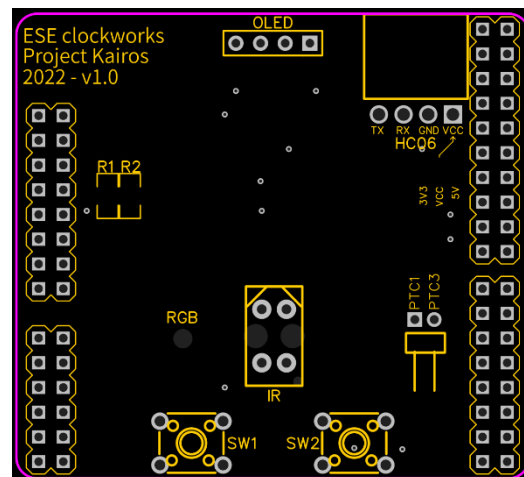


Figure 12. Placing of the components on the PCB.

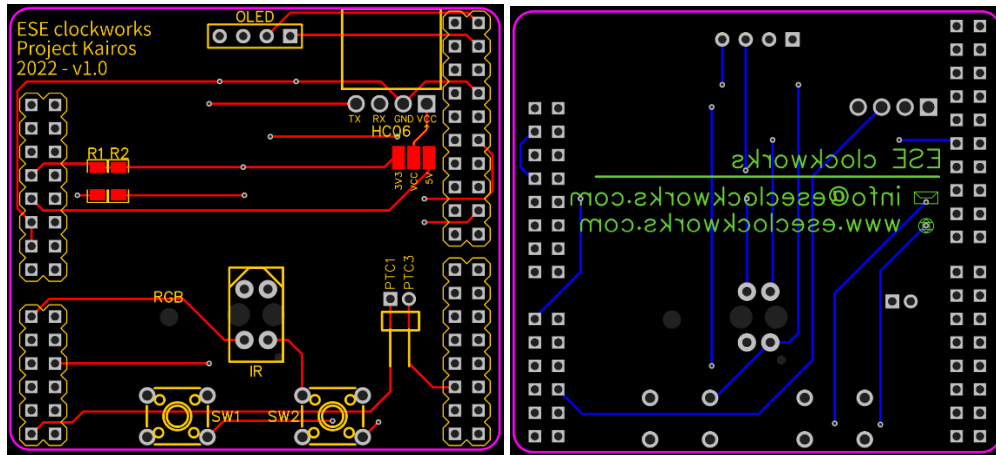


Figure 13. Top- (left) en bottom layer (right) of the PCB.

The 3D view in Figure 14 gives a good idea of the expected assembled result.

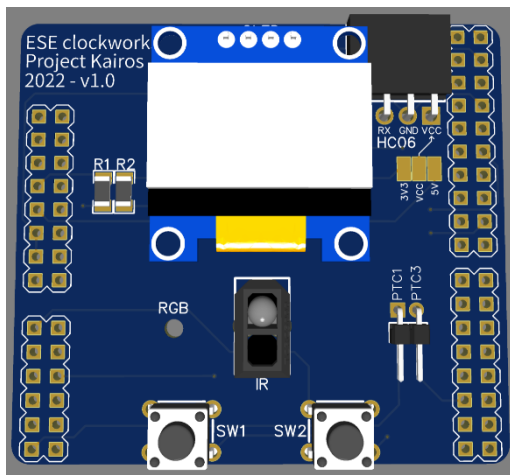


Figure 14. 3D view of the PCB.

## 5.2 Software

The complete listing of the realized software can be found in Appendix X. In this section, the main software components are explained using code snippets. First, the settings of the software development environment are discussed.

### 5.2.1 Keil $\mu$ Vision IDE

The community edition of the Keil  $\mu$ Vision IDE (ARM, 2022) is used to develop the software. A separate folder is created for each sensor and actuator. The project structure is shown in Figure 15. To configure this project the instructions from application note 232 (ARM Developer, 2022) were followed. More detailed settings specific to this project can be found in Appendix X.

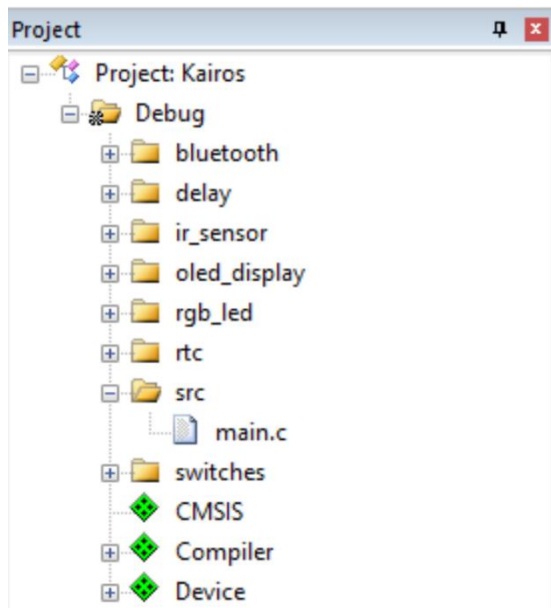


Figure 15. Project structure in Keil  $\mu$ Vision IDE.

### 5.2.2 Switches

#### switches\switches.h

```
/// Defines the type for the keys
typedef enum
{
    SW1 = 0,
    SW2,
} sw_t;

// Function prototypes
void sw_init(void);
bool sw_read_pressed(const sw_t sw);
```

The two switches are defined using an enum. The parameter `sw` in the function `sw_read_pressed()` is declared `const` since there is no reason to modify that variable in the function.

#### switches\switches.c

```
static PORT_Type * port_mapping[N_SWITCHES] = {PORTD, PORTD};
static GPIO_Type * gpio_mapping[N_SWITCHES] = {PTD, PTD};
static uint8_t pin_mapping[N_SWITCHES] = {3, 5};
```

Through three lookup tables each switch indicates which PORT peripheral, GPIO peripheral and pin number are used.

#### switches\switches.c

```
/*!
 * \brief Initialises the switches on the shield
 *
 * This functions initializes the switches on the shield.
 */
void sw_init(void)
{
    // Enable clocks to PORT
    SIM->SCGC5 |= SIM_SCGC5_PORTD_MASK;
```



```

// Configure all pins as follows:
// - MUX[2:0] = 001 : Alternative 1 (GPIO)
// - DSE = 0 : Low drive strength
// - PFE = 0 : Passive input filter is disabled
// - SRE = 0 : Fast slew rate is configured
// - PE = 1 : Internal pullup or pulldown resistor is enabled
// - PS = 1 : Internal pullup resistor is enabled
for(int i=0; i<N_SWITCHES; i++)
{
    port_mapping[i]->PCR[pin_mapping[i]] = 0b00100000011;
}

// Set port pins to inputs
for(int i=0; i<N_SWITCHES; i++)
{
    gpio_mapping[i]->PDDR &= ~(1<<pin_mapping[i]);
}
}

```

In the `sw_init()` function, each pin is configured through two for-loops. With the index `i` the applicable peripherals, registers and bit positions are selected from the lookup tables. The comment describes which bits are enabled or disabled.

#### switches\switches.c

```

/*!
 * \brief Check if a switch is pressed
 *
 * This functions checks if a switch is pressed. The function simply checks the
 * value of the switch at the moment the function is called. It doesn't
 * remember if the switch has been pressed.
 *
 * \param[in] sw Switch that will be checked, must be of type ::sw_t
 *
 * \return True if the switch is pressed, false otherwise
 */
bool sw_read_pressed(const sw_t sw)
{
    // If the key is pressed, the bit at that position will read logic 0
    return ((gpio_mapping[sw]->PDIR & (1<<pin_mapping[sw])) == 0);
}

```

Based on the parameter provided, it is checked whether that particular pin is pressed. A pressed switch reads logical 0, so after bitwise and-ing with the mask `(1<<pin_mapping[sw])`, a logical comparison to zero is made. This function again uses the lookup tables to select the corresponding peripheral, register and bit position. It does not check whether the parameter `sw` has a value within the valid range (0 or 1). In a sense, this is enforced by the compiler because it is of type `sw_t`. But theoretically, a value outside the range of `sw_t` could be passed in. Undefined behavior will then occur.

#### 5.2.3 OLED display

The software for the OLED display is realized in layers and has multiple files. This is shown in Figure 16. The blue parts are realized in such a way that they are independent of the interface. In this project the I2C interface is used, hence the I2C peripheral driver. The realization of all components is described in the following paragraphs.

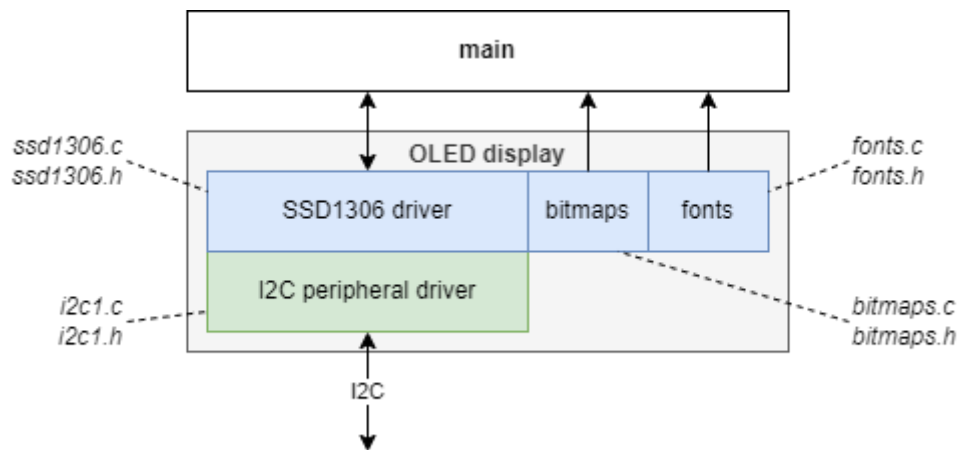


Figure 16. Layered design and files of the software of the OLED display driver.

#### 5.2.3.1 I2C peripheral driver

The I2C peripheral driver implements hardware communication via I2C. All communication is realized based on polling. This means that it waits until a byte is sent through the I2C interface. The driver implements three functions, as shown in the header file. The difference between the two write functions is the value of the control byte that is sent after the slave address is sent. For a command it is 0x00 and for data it is 0x40 (Solomon Systech Limited, 2008).

##### oled\_display\i2c1.h

```

/*!
 * \brief Definition for the I2C timeout
 *
 * This timeout value is used in loops to wait for a bit to set/reset.
 * If the bit doesn't get set, the function returns.
 */
#define I2C_TIMEOUT (10000)

// Function prototypes
void i2c1_init(void);

bool i2c1_write_cmd(const uint8_t address,
                    const uint8_t cmd[],
                    const uint32_t n);
bool i2c1_write_data(const uint8_t address,
                     const uint8_t data[],
                     const uint32_t n);

```

Polling is accomplished by waiting in while-loop and doing nothing as long as the I2C\_S\_IICIF bit is logically 0. If an error should occur, such as an incorrectly connected display, it would mean that the application would remain in that while-loop indefinitely. The I2C\_S\_IICIF bit then always remains logically 0. To prevent an endless loop, a timeout is defined. How long this timeout lasts depends on the core clock. This can be adjusted by means of a definition.

The timing diagram in Figure 17 shows the start of successful communication with the OLED display. The slave address 0x78 is acknowledged and then 0x00 is sent to indicate that the following bytes should be interpreted as command bytes. These command bytes are not shown in Figure 17.

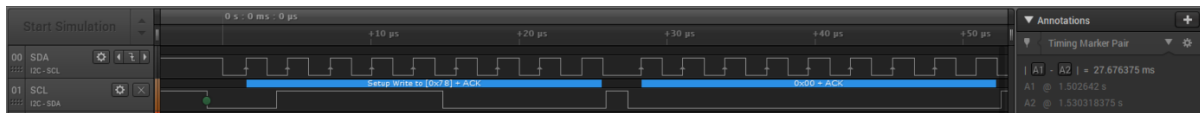


Figure 17. Timing diagram of the I2C communication met het OLED display.

### 5.2.3.2 SSD1306 driver

The SSD1306 driver is the interface for the main application. There are several functions available, which send those appropriate commands and data to the OLED display via the I2C peripheral driver. Some examples are the function `ssd1306_setcontrast()` to set the contrast and `ssd1306_setpixel()` to turn on or off a pixel at a (x,y) location. The commands and data to be sent were determined from the OLED display datasheet (Solomon Systech Limited, 2008).

The SSD1306 driver is implemented using a frame buffer in which 8 pixels are stored in a `uint8_t`. This is declared in the file `ssd1306.c`. The define `SSD1306_SIZE` gives the number of pixels divided by 8 and for a 128x64 is equal to 1024, or 1 kilobyte.

```
oled_display\ssd1306.c
uint8_t ssd1306_framebuffer[SSD1306_SIZE];
```

All functions that display data use this frame buffer. This means that, for example, turning a pixel on or off happens in the framebuffer and not on the OLED display. To display the framebuffer on the OLED display, the function `ssd1306_update()` must be called. The timing diagram in Figure 18 shows that it takes about 27.7ms to send the complete frame buffer, i.e. all 128x64 pixels.

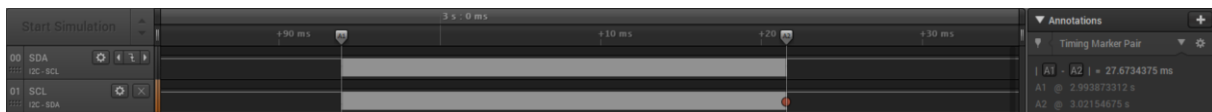


Figure 18. Timing diagram showing the transmission of a framebuffer.

A framebuffer implementation was chosen because via I2C the pixels cannot be read. However, it is necessary to know the values of the pixels if display updates need to take place. An additional advantage is that it is relatively fast to update that framebuffer because no communication needs to take place. Although it takes a relatively long time to send a complete frame buffer, this time is always the same, which is about 30ms.

### 5.2.3.3 Fonts

The SSD1306 driver supports several fonts. These fonts are defined in font tables in the file `fonts.c`. Adding a new font is done using the following steps:

1. Copy a font version  $\geq 3.0.0$  from the following website:  
<http://oleddisplay.squix.ch/>
2. Add the generated code to the file `fonts.c`.
3. Remove `PROGMEM` from the declaration.
4. Add an external declaration to the file `fonts.h`.
5. Use the font by passing a pointer to the `ssd1306_setfont()` function.

### 5.2.3.4 Bitmaps

The SSD1306 driver supports the display of bitmaps. Adding a new bitmap involves the following steps:

1. Create a 128x64 bitmap, for example with the following tool:  
[http://en.radzio.dxp.pl/bitmap\\_converter/](http://en.radzio.dxp.pl/bitmap_converter/)
2. Add the generated code to the file `bitmaps.c`.

3. Add an external declaration to the file `bitmaps.h`.
4. Draw the bitmap by passing a pointer to the function `ssd1306_drawbitmap()`.

#### 5.2.4 Main

The main application first initializes all software by calling the `init()` functions and showing a message on the OLED display. Then the endless loop is started and with the `__WFI()` instruction the "Normal Wait - via WFI" chip power mode is activated, see reference manual (Freescale Semiconductor, Inc., 2012) table 7.1. In this power mode, all peripherals continue to work and the core operates in deep sleep mode. The NVIC remains sensitive to interrupts, so the core comes out of deep sleep mode when any interrupt occurs, such as an RTC interrupt.

When the core comes out of deep sleep mode, the flags are polled to check which action to perform. The code snippet below shows this for two examples.

##### main.c

```

/*!
 * \brief Main application
 *
 * This main application of the Kairos project. It implements a cyclic
 * executive with interrupts scheduling mechanism.
 */
int main(void)
{
    // Initialize all modules and show message on OLED display

    while(1)
    {
        __WFI();
        // Core wakes up from interrupt, continue code execution

        // -----
        // RTC
        // -----
        if(rtc_read_sec_flag())
        {
            // The RTC has generated an interrupt indicating that a
            // second has passed

            // Read the current time
            uint32_t timestamp = rtc_read_timestamp()

            // Convert the timestamp and show datetime based on the mode
            // ...

        }

        // -----
        // SW1
        // -----
        if(sw_read_pressed(SW1))
        {
            // Switch 1 is pressed, handle the event
            // ...

        }

        // Etc.
    }
}

```

## 6 Testing

*Unambiguous representation of how the system, hardware and/or software was tested. What hardware and or software modules were tested, how were the functional specifications tested during the acceptance test? What test setup was used and what were the final results. Did the tests meet the functional and technical specifications? The results are accompanied by a clear description of any remaining problems and how they might be explained. Were any 'work arounds' performed during testing? The tests must be described in such a way that each test can be reproduced by others.*

## 6 Testing - Case study Kairos

This section describes the acceptance tests and tests for power consumption performed for the Kairos project.

### 6.1 Acceptance testing

A total of five test scenarios were performed. Each test scenario aims to verify the operation of one or more functional specifications. There are three possible outcomes, where a functional specification realized **fully** (✓), **partly** (🟡) or **not** (✗) as specified. The table below shows the result at a glance. [Appendix E](#) describes for each test scenario which preparations have to be made, which steps have to be performed and what is the expected output of the system at each step. If the tested output is equal to the expected output, the test is marked as correct. At the end of each test, there may be a comment.

	Test scenario 1	Test scenario 2	Test scenario 3	Test scenario 4	Test scenario 5
F1	✓				
F2		✓			
F3			🟡		
F4				✗	
F5					✓
F6					✓

### 6.2 Power consumption

During the development of the product, power consumption was not specifically considered in terms of any hardware or software modifications. The initial focus was on making the requested functional specifications work.

However, measurements were made to determine which hardware modifications to the FRDM-KL25Z would result in which power savings. These measurements are described in detail in [Appendix E](#). Table 1 summarizes these results.

Table 1. Summary power consumption measurements.

Nr.	Measured current [μA]	Saved current [μA]	Changes to the FRDM-KL25Z board
1	8290	-	None.
2	5080	3210	Cut trace J20.
3	375	4705	Remove R74.
4	375	0	Cut trace J11.
5	38	337	Cut trace J14.

By making the above modifications, the FRDM-KL25Z board consumes 38μA. It must be said that no other hardware is active, such as an LED or an OLED display. Since a CR2032 battery has a capacity of 225mAh, the system can theoretically operate  $225\text{mAh} / 38\mu\text{A} = 5921$  hours. This is equivalent to 246 days.

## 7 Conclusions and recommendations

*Reflection on the goals of the project. What are the results? What has been achieved and what has not been achieved? What can be added to, expanded on, improved?*

## 7 Conclusions and recommendations - Case study Kairos

The main goals of the Kairos project were the use of the FRDM-KL25Z development kit and the addition of wireless communication for remote time synchronization. In addition, insight into energy consumption was gained. These main goals were all achieved and a working prototype was delivered.

The KL25 microcontroller has many more peripherals than was necessary for this project. For the realization of a product, a microcontroller must have at least 32kB FLASH memory, 8kB RAM memory, a timer, a UART, an ADC, an RTC, two GPIO pins and low-power capabilities. In the KL family of NXP several solutions can be found for this, but manufacturers such as STMicroelectronics and Atmel offer similar solutions.

Wireless communication is realized by means of an HC06 bluetooth module. These modules are, despite the prevailing shortage of chips, readily available and not expensive. For the microcontroller there is a serial (UART) connection and for most operating systems a driver is available that gives access to the module through a virtual COM port. The data transfer in this project between app and microcontroller was kept as simple as possible because the focus was on the embedded system. If the project is developed further, where an app will be developed, then it is recommended to move to a more practical data format, JSON or CSV. On the other hand, ASCII strings does have its advantages, for example because the text sent remains readable.

To gain insight into the power consumption, several hardware modifications were made to the FRDM-KL25Z board. However, within the set duration of the project there was not enough time to integrate these modifications into the project. Therefore, in consultation with the client, it was decided to document the modifications (see [Appendix F](#)). In future development of digital clocks it is recommended to take these findings into account. Especially disabling the SDA interface and the way the core is put in deep sleep mode are important.

The designed, realized and tested clock that was created during the Kairos project can be called a great success. Thanks to this project, the company *ESE clockworks* is one step closer to the realization of a new generation of digital clocks.



## 8 References

- Adrián Sánchez Cano. (2013, 3 5). *Using RTC module on FRDM-KL25Z*. Retrieved from <https://community.nxp.com/docs/DOC-94734>
- ARM. (2022, 04 26). *μVision® IDE*. Retrieved from <https://www2.keil.com/mdk5/uvision/>
- ARM Developer. (2022, 04 26). *KAN232 - MDK V5 Lab for Freescale Freedom KL25Z Board*. Retrieved from <https://developer.arm.com/documentation/kan232/latest/>
- Berckel, M. v.-v. (2017). *Schrijven voor technici*. Noordhoff Uitgevers B.V.
- contributors, W. (2022, 07 06). *MoSCoW method*. (Wikipedia, The Free Encyclopedia) Retrieved 07 06, 2022, from [https://en.wikipedia.org/w/index.php?title=MoSCoW\\_method&oldid=1091822315](https://en.wikipedia.org/w/index.php?title=MoSCoW_method&oldid=1091822315)
- contributors, W. (2022, 05 25). *SMART criteria*. (Wikipedia, The Free Encyclopedia) Retrieved 07 07, 2022, from [https://en.wikipedia.org/w/index.php?title=SMART\\_criteria&oldid=1089766780](https://en.wikipedia.org/w/index.php?title=SMART_criteria&oldid=1089766780)
- ELECFREAKS. (2022, 04 19). Retrieved from Ultrasonic Ranging Module HC - SR04: <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>
- Freescale Semiconductor, I. (n.d.). *FRDM-KL25Z Pinouts (Rev 1.0)*. Retrieved 3 31, 2023, from <https://www.nxp.com/document/guide/get-started-with-the-frdm-kl25z:NGS-FRDM-KL25Z>
- Freescale Semiconductor, Inc. (2012, 9). *KL25 Sub-Family Reference Manual, Rev. 3*.
- Freescale Semiconductor, Inc. (2013, 10 24). *FRDM-KL25Z User's Manual, Rev. 2.0*. Retrieved from <https://www.nxp.com/document/guide/get-started-with-the-frdm-kl25z:NGS-FRDM-KL25Z>
- Freescale Semiconductor, Inc. (2014, 08). *Kinetis KL25 Sub-Family, 48 MHz Cortex-M0+ Based Microcontroller with USB, Rev 5*.
- Hmneverl. (2015, 11 18). *De beslismatrix, het maken van keuzes*. (Info.NU.nl) Retrieved 07 06, 2022, from <https://mens-en-samenleving.info.nu.nl/diversen/164525-de-beslismatrix-het-maken-van-keuzes.html>
- NXP. (2022, 04 19). *Kinetis® KL2x-72/96 MHz, USB Ultra-Low-Power Microcontrollers (MCUs) based on Arm® Cortex®-M0+ Core*. Retrieved from [https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/general-purpose-mcus/kl-series-cortex-m0-plus/kinetis-kl2x-72-96-mhz-usb-ultra-low-power-microcontrollers-mcus-based-on-arm-cortex-m0-plus-core:KL2x?tab=Buy\\_Parametric\\_Tab#/](https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/general-purpose-mcus/kl-series-cortex-m0-plus/kinetis-kl2x-72-96-mhz-usb-ultra-low-power-microcontrollers-mcus-based-on-arm-cortex-m0-plus-core:KL2x?tab=Buy_Parametric_Tab#/)
- NXP. (2022). *OpenSDA Serial and Debug Adapter*. Retrieved from <https://www.nxp.com/design/software/development-software/sensor-toolbox-sensor-development-ecosystem/opensda-serial-and-debug-adapter:OPENSDA?&tid=vanOpenSDA>
- Solomon Systech Limited. (2008, 4). *SSD1306: Advanced Information*. Retrieved from <https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf>
- Vishay Semiconductors. (2017, 8 9). *TCRT5000(L), Reflective Optical Sensor with Transistor Output, Rev. 1.7*.

## Appendix A

## Appendix B

## Appendix n

## Appendix A – Making the FRDM-KL25Z suitable for backup battery

### Research question

This document describes the research into how the FRDM-KL25Z can be made suitable for a backup battery. The addition of a backup battery is necessary to be able to meet requirement T4. The research questions are as follows.

	Research question	Prio	Req
Q1	What modifications are needed to make the FRDM-KL25Z suitable for a backup battery?	M	T4
Q2	What additional components are needed to make the FRDM-KL25Z suitable for a backup battery?	M	T4

### Research method

Desk research was conducted to answer the research questions. On the FRDM-KL25Z development kit, preparations have been made to use a backup battery. Although the preparations help to apply a backup battery relatively easily, several hardware modifications must be made. The user manual (Freescale Semiconductor, Inc., 2013) describes these modifications and the implications are discussed step-by-step.

### Research data

#### Additional components

In the user manual (Freescale Semiconductor, Inc., 2013), *Table 3. FRDM-KL25Z Power Supplies*, the options for powering the development kit are indicated. An important note relates to the placement of the battery holder:

*“If a coin cell battery is to be used add a small amount of solder to the coin cell ground pad before adding the battery holder. Also, it is recommended to populate D7 as a protection diode when using a coin cell battery.”*

A suitable battery holder can be ordered from Mouser with order number [534-3003](#). This battery holder is suitable for 20mm batteries, such as the CR2032.

Due to major shortages in the component market, the prescribed Shottky diode D7 is not available. A suitable alternative can be ordered from Farnell at the time of writing with order number [2918857](#).

#### Board modifications

*Figure 3. Power Supply Schematic* in (Freescale Semiconductor, Inc., 2013) shows the schematic of the power supply circuit. This schematic is reproduced in Figure 19 and is annotated. The

descriptions of the annotations continue after Figure 19.

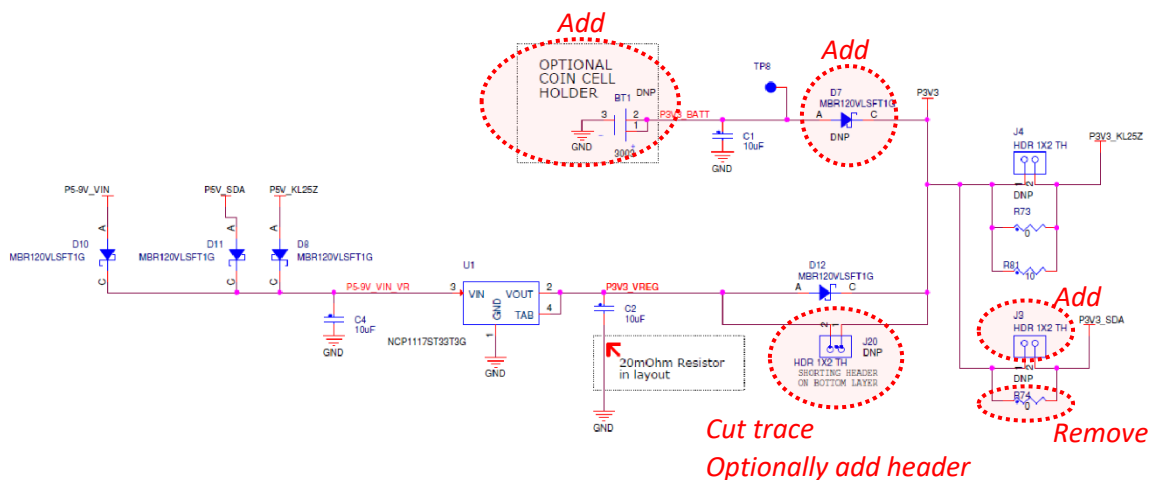


Figure 19. Annotated power supply.

As already indicated in point 1, BT1 and D7 should be added.

Shottky diode D12 normally ensures that no current can flow from the battery to U1. However, D12 is shorted by J20 in the form of a PCB trace (on the bottom layer of the PCB). The reason for this is that there is then no voltage drop across D12, because normally there is no backup battery anyway. If the backup battery is used, then J20 must be cut. Optionally, a header can be placed on J20 to later undo the cut trace with a jumper.

To save as much energy as possible when the system is powered by a battery, resistor R74 can be removed. This prevents the SDA interface from being powered by 3.3V. To still be able to use the SDA interface during development (for programming and debugging), a header must be placed on J3.

By making these adjustments, the backup battery is in parallel with the other voltage sources (P5-9V\_VIN, P5V\_SDA and P5V\_KL25Z). To prevent the backup battery from being drained too quickly during development work, it is best to remove the backup battery during that work. To prevent other devices from receiving power that could drain the backup battery, the ADC should be used to check whether another power source is present. If not, measures must be taken in software to switch off the other devices as much as possible. How long a battery will last is impossible to answer at this moment. This depends, among other things, on the battery used and the amount of power that external components require when they are switched off.

## Conclusion

Answers to the research questions.

Q1. The modifications that are needed to make the FRDM-KL25Z suitable for a backup battery are the scratching of various print traces and the placement of headers. By removing an additional resistor, the SDA will be disconnected and power consumption will decrease even further.

Q2. The additional components needed to make the FRDM-KL25Z suitable for a backup battery are a battery holder and a Shottkey diode.

## Appendix B – Detecting a hand in front of the clock

### Research question

This document describes the reserach into how a hand can be detected before the clock. Detecting a hand, or any other object, is necessary to meet requirement F3. The research questions are formulated as follows.

	Research question	Prio	Req
Q1	Which sensors are suitable for detecting a hand or other object 10cm from the clock with a precision of 1cm?	M	F3
Q2	Which form factor suits the design of the clock?	M	F3
Q3	What is the energy consumption of the sensor?	M	F3
Q4	What is the price of the sensor?	M	F3

### Research method

Empirical research has been conducted to answer the research questions. Experiments were carried out with two sensors for this purpose. The first sensor is the TCRT5000 (Vishay Semiconductors, 2017). This is a distance sensor that uses infrared light. The second sensor is the HC-SR04 (ELECTREKS, 2022). This is a distance sensor that uses ultrasound.

### Research data

TBD

*For each sensor, describe the test set-up, the hardware and software used. Record the results of measurements that demonstrate the specifications of the sensor. Summarize those measurements in a table and/or graph. Include other information, such as links to datasheets describing the form factor, etc. Another engineer must be able to repeat the research with the information described. So be complete in the description.*

### Conclusion

Answers to the research questions.

Q1. The TCRT5000 and HC-SR04 sensors are both suitable for detecting a hand or other object 10cm from the clock. The TCRT500 has a precision of 5mm. The HC-SR04 has a precision of 1cm.

Q2. The HC-SR04 is large compared to a 0.96-inch OLED display. A user's attention could then be focused too much on the sensor and the entire system would feel 'bulky'. The TCRT5000 fits perfectly in terms of size.

Q3. The HC-SR04 consumes an average of 50mW. The TCRT5000 consumes an average of 20mW.

Q4. The TCRT5000 is about five times cheaper than the HC-SR04. Two resistors still have to be purchased, but they are negligible in terms of price.

Table 2 shows the decision matrix based on the answers. Each aspect is scored on a scale of 1 to 5 and a weighting factor indicates the importance of an aspect.

Table 2: Decision matrix for choosing a sensor to detect a hand.

	Costs	Form factor	Interface	Power consumption	TOTAL
Weight	3	4	3	1	
HC-SR04	$1 \times 3 = 3$	$1 \times 4 = 4$	$3 \times 3 = 9$	$3 \times 1 = 3$	19
TCRT5000	$5 \times 3 = 15$	$5 \times 4 = 20$	$3 \times 3 = 9$	$4 \times 1 = 4$	48

Thus, the product will be developed with the TCRT5000 reflective optical sensor for detecting a hand in front of the clock.



## Appendix C – Pinout and peripheral overview

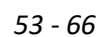
FRDM-KL25Z Pin name	FRDM-KL25Z On-board usage	Available on IO Header	I2C1	UART0	UART2	TPM0	TPM2	GPIO	ANALOG	CLKOUT
PTE0	-	J2	OLED SDA							
PTE1	-	J2	OLED SCL							
PTE2	-	J9								
PTE3	-	J9								
PTE4	-	J9								
PTE5	-	J9								
VDD	Power	-								
VSS	Power	-								
USB0_DP	USB	-								
USB0_DM	USB	-								
VOOUT33	2.2uF cap	-								
VREGIN	USB VBUS (5V)	-								
PTE20	-	J10								
PTE21	-	J10								
PTE22	-	J10			UART2_TX					
PTE23	-	J10			UART2_RX					
VDDA	Power	-								
VREFH	Power	J2								
VREFL	Power	-								
VSSA	Power	-								
PTE29	-	J10								
PTE30	-	J10								
PTE31	-	J2								
PTE24	Accelerometer I2C	-								
PTE25	Accelerometer I2C	-								
PTA0	Debug (SWD_CLK)	-								

FRDM-KL25Z Pin name	FRDM-KL25Z On-board usage	Available on IO Header	I2C1	UART0	UART2	TPM0	TPM2	GPIO	ANALOG	CLKOUT
PTA1	-	J1		UART0_RX						
PTA2	-	J1		UART0_TX						
PTA3	Debug (SWD_DIO)	-								
PTA4	-	J1								
PTA5	-	J1								
PTA12	-	J1								
PTA13	-	J2								
PTA14	Accelerometer INT1	-								
PTA15	Accelerometer INT2	-								
PTA16	-	J2						IR sensor - LED		
PTA17	-	J2								
VDD	Power	-								
VSS	Power	-								
PTA18	8MHz XTAL	-								
PTA19	8MHz XTAL	-								
PTA20	Reset	J9								
PTB0	-	J10							IR sensor - out	
PTB1	-	J10								
PTB2	-	J10								
PTB3	-	J10							Potentiometer	
PTB8	-	J9								
PTB9	-	J9								
PTB10	-	J9								
PTB11	-	J9								
PTB16	Touch Slider	-								
PTB17	Touch Slider	-								

FRDM-KL25Z Pin name	FRDM-KL25Z On-board usage	Available on IO Header	I2C1	UART0	UART2	TPM0	TPM2	GPIO	ANALOG	CLKOUT
PTB18	Red LED	-					Red LED channel 0			
PTB19	Green LED	-					Green LED channel 1			
PTC0	-	J1								
PTC1	-	J10						RTC		
PTC2	-	J10								
PTC3	-	J1								RTC
VSS	Power	-								
VDD	Power	-								
PTC4	-	J1								
PTC5	-	J1								
PTC6	-	J1								
PTC7	-	J1								
PTC8	-	J1								
PTC9	-	J1								
PTC10	-	J1								
PTC11	-	J1								
PTC12	-	J2								
PTC13	-	J2								
PTC16	-	J2								
PTC17	-	J2								
PTD0	-	J2								
PTD1	Blue LED	J2				Blue LED channel 1				
PTD2	-	J1								
PTD3	-	J2							Switch 1	
PTD4	-	J1							LED	
PTD5	-	J2							Switch 2	
PTD6	-	J2								
PTD7	-	J2								

FRDM- KL25Z Pin name	FRDM- KL25Z On-board usage	Available on IO Header	I2C1	UART0	UART2	TPM0	TPM2	GPIO	ANALOG	CLKOUT
GND	Power	J2								
P3V3	Power	J9								
P3V3	Power	J9								
P5V_USB	Power	J9								
GND	Power	J9								
GND	Power	J9								
P5-9V_VIN	Power	J9								

## Product report



## Appendix E – Test scenario's

This appendix describes in detail the test scenarios for the purpose of conducting the acceptance tests.

### Test scenario 1: Displaying the time

#### Supplies:

- Clock
  - Hardware version 1.0
  - Software version 1.1
- Micro USB cable

#### Tested functional requirements:

- F1

#### Test performed:

- 07/07/2022 by H. Arends.

#### Steps:

01	<b>Connect the clock to a power source with the micro USB cable.</b>	
	The following time is displayed: 00:00:00 01/01/1970 The time is displayed large and the date small.	
	<i>Observed as described.</i>	✓
02	<b>Press SW1 once and release SW1.</b>	
	The time is displayed small and the date large. The time counts down.	
	<i>Observed as described.</i>	✓
03	<b>Press SW1 once and release SW1.</b>	
	The time is displayed analog on the left half of the OLED display. The date is displayed on the right half, with the month written in full, i.e. 'January'. The time is ticking.	
	<i>Observed as described.</i>	✓
04	<b>Press SW1 once and release SW1.</b>	
	The time is displayed large and the date small. The time ticks every second.	
	<i>Observed as described.</i>	✓

Conclusion: ✓

Test passed.

### Test scenario 2: Customize date and time

#### Supplies:

- Clock
  - Hardware version 1.0
  - Software version 1.1

- Micro USB cable

**Tested functional requirements:**

- F2

**Test performed:**

- 07/07/2022 by H. Arends.

**Steps:**

01	<b>Connect the clock to a power source with the micro USB cable.</b>	
	The following time is displayed: 00:00:00 01/01/1970 The time is displayed large and the date small.	
	<i>Observed as described.</i>	✓
02	<b>Press SW2 once and release SW2.</b>	
	The text setup appears on the screen. Adjustable: day Value: 01	
	<i>Observed as described.</i>	✓
03	<b>Press SW1 ten times.</b>	
	Adjustable: day Value: 11	
	<i>Observed as described.</i>	✓
04	<b>Press SW2 once and release SW2.</b>	
	Adjustable: month Value: 01	
	<i>Observed as described.</i>	✓
05	<b>Press SW1 twenty times.</b>	
	Adjustable: month Value: 09	
	<i>Observed as described.</i>	✓
06	<b>Press SW2 once and release SW2.</b>	
	Adjustable: year Value: 1970	
	<i>Observed as described.</i>	✓
07	<b>Press SW2 once and release SW2.</b>	
	Adjustable: hours Value: 00	
	<i>Observed as described.</i>	✓
08	<b>Press SW1 three times.</b>	
	Adjustable: hours Value: 03	
	<i>Observed as described.</i>	✓
09	<b>Press SW2 once and release SW2.</b>	

	Adjustable: minutes Value: 00	
	<i>Observed as described.</i>	✓
10	<b>Press SW2 once and release SW2.</b>	
	Adjustable: minutes Value: 00 Waarde: depending on how fast SW2 was pressed at the beginning of the test	
	<i>Observed number of seconds is 04.</i>	✓
11	<b>Press SW2 once and release SW2.</b>	
	The following time is displayed: 03:00:?? 11/09/1970 The time is displayed large and the date small. The time ticks every second	
	<i>Observed time is 03:00:04 11/09/1970</i>	✓

Conclusion: ✓

Passed.

### Test scenario 3: Detecting a hand in front of the clock

#### Supplies:

- Clock
  - Hardware version 1.0
  - Software version 1.1
- Micro USB cable
- Ruler

#### Tested functional requirements:

- F3

#### Test performed:

- 07/07/2022 by H. Arends.

#### Steps:

01	<b>Connect the clock to a power source using the micro USB cable. Make sure no object is within 10cm of the clock.</b>	
	The following time is displayed: 00:00:00 01/01/1970 The time is displayed large and the date small.	
	<i>Observed as described.</i>	✓
02	<b>Hold your hand 15cm in front of the clock.</b>	
	No change is noticeable. The time is still displayed large and the date small.	
	<i>Observed as described.</i>	✓
03	<b>Hold your hand 8cm in front of the clock.</b>	
	The clock shows the name of the company.	
	<i>The time is still displayed large and the date small. Only from 7.5cm the hand is detected correctly.</i>	🕒



04	<b>Hold your hand 5cm in front of the clock.</b>	
	The clock shows the name of the company.	
	<i>Observed as described.</i>	✓
05	<b>Hold your hand 15cm in front of the clock.</b>	
	The clock displays the name of the company for another 5 seconds. After the 5 seconds have elapsed, the time is displayed large and the date small. The time is also displayed correctly, i.e. the elapsed time since step 3.	
	<i>Observed as described.</i>	✓

#### Conclusion: 🟡

Although most of the steps are correct, at step three a hand is detected from 7.5cm. Thus, the test is considered largely successful.

Test scenario 4: TBD

This test scenario has not been described (yet).

Test scenario 5: TBD

This test scenario has not been described (yet).

## Appendix F – Power consumption measurements

Tabel 3 vat de bevindingen samen. De gemeten stroom is de stroom wanneer de blauwe LED uit is. De gemeten stroom is cumulatief, wat betekent dat alle eerdere aanpassingen intact zijn gelaten. Door alle aanpassingen te maken, werkt het FRDM-KL25Z-bord op 38  $\mu\text{A}$ .

Tabel 3. Summary power consumption measurements.

Nr.	Measured current [ $\mu\text{A}$ ]	Saved current [ $\mu\text{A}$ ]	Changes to the FRDM-KL25Z board
1	8290	-	None.
2	5080	3210	Cut trace J20.
3	375	4705	Remove R74.
4	375	0	Cut trace J11.
5	38	337	Cut trace J14.

### Measurement setup

Figure 1 shows the measurement setup. The [PPK2](#) is the only power source (except for the RTC backup battery). The PPK2 is used in *Source meter* mode with the supply voltage set to 3000 mV.

VCC of header H2 is connected to P3V3 of the FRDM-KL25Z development board.

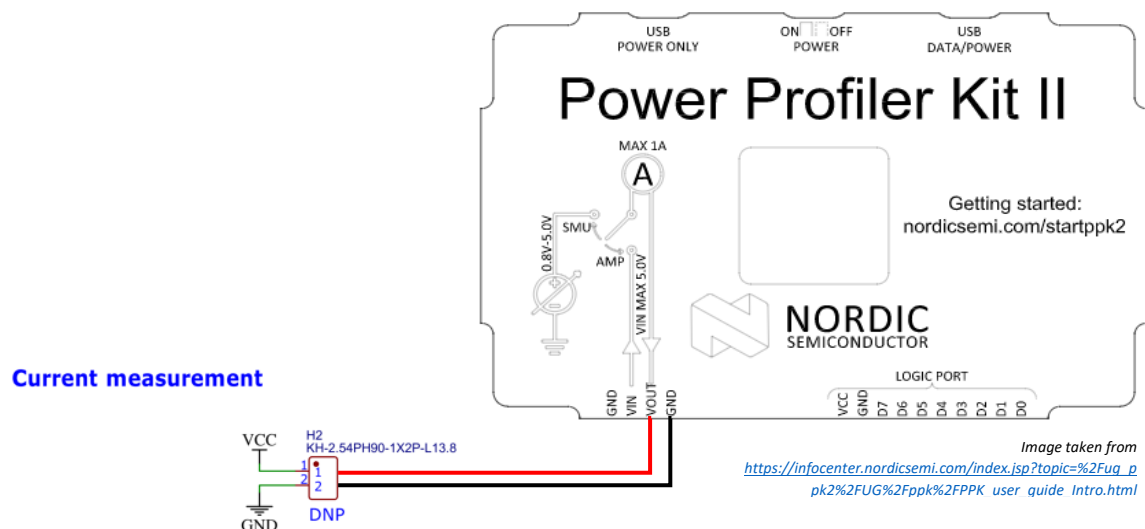


Figure 1. Measurement setup, using the PPK2 in Source meter mode.

The following code is used during measurements. This code is created in Keil  $\mu\text{Vision}$  V5.37.0.0 with optimization level set to -O3 and using the device startup files provided by Keil  $\mu\text{Vision}$ . The project incorporates a global define `CLOCK_SETUP=3`, which sets the system clocks as follows:

- Chip externally clocked, ready for Very Low Power Run mode
- Multipurpose Clock Generator (MCG) in BLPE mode
- Reference clock source for MCG module: System oscillator reference clock
- Core clock = 4MHz
- Bus clock = 1MHz

Furthermore, the lowpower timer (LPTMR) is used for generating an interrupt after a timeout. This timeout is defined in milliseconds with the function `lptmr_set_timeout_ms()`. The LPTMR functions are not given in the code example.

```
int main(void)
{
    SIM->SCGC5 |= SIM_SCGC5_PORTD(1);
    PORTD->PCR[1] = PORT_PCR_MUX(1);
    GPIOD->PDDR |= (1<<1);

    // Setup power mode as follows:
    // - RUNM = 10: Very-Low-Power Run mode (VLPR)
    // - STOPM = 010: Very-Low-Power Stop (VLPS)
    SMC->PMCTRL = SMC_PMCTRL_RUNM(0b10) | SMC_PMCTRL_STOPM(0b010);

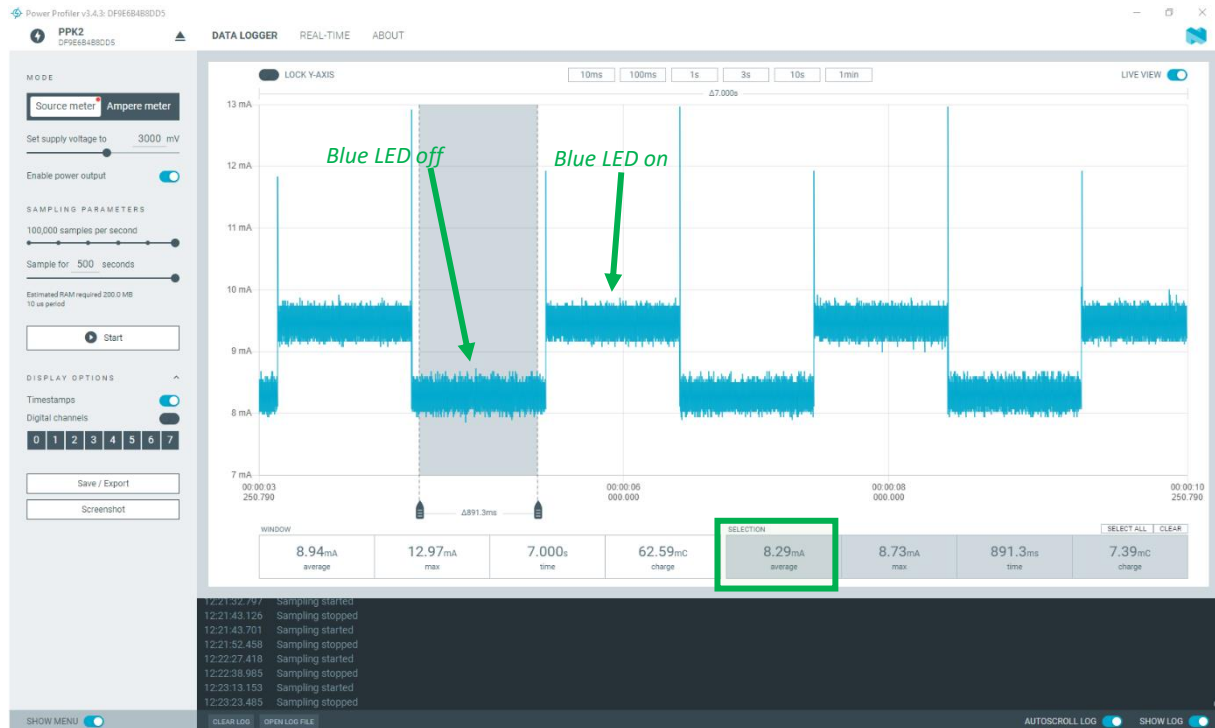
    // Configure the processor core to use deep sleep as its low power
    // mode
    SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;

    lptmr_init();
    lptmr_set_timeout_ms(1000);

    while(1)
    {
        __asm("wfi");

        if(lptmr_timeout_flag == true)
        {
            lptmr_timeout_flag = false;
            lptmr_set_timeout_ms(1000);
            GPIOD->PTOR = (1<<1);
        }
    }
}
```

## No adjustments



The selected grey area shows the average values when the blue LED is off.

## Power supply adjustments

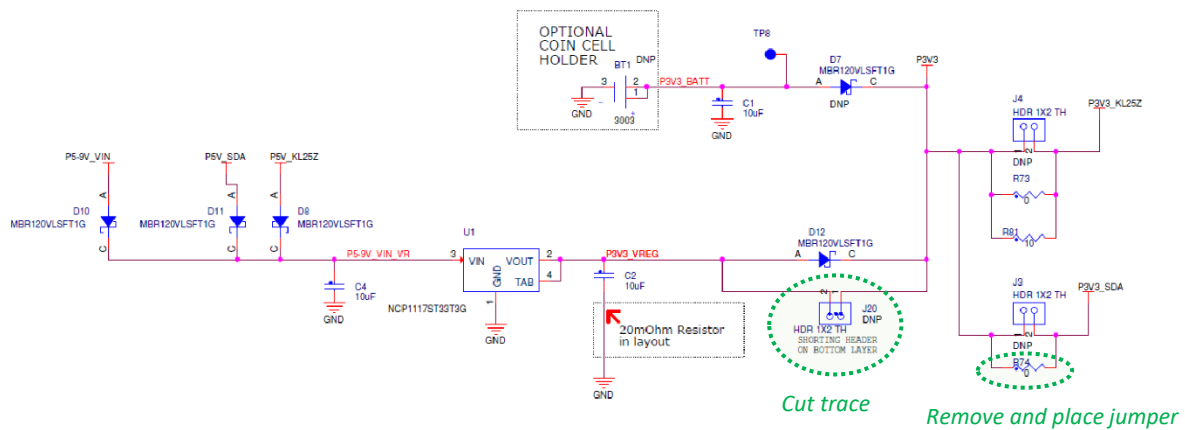
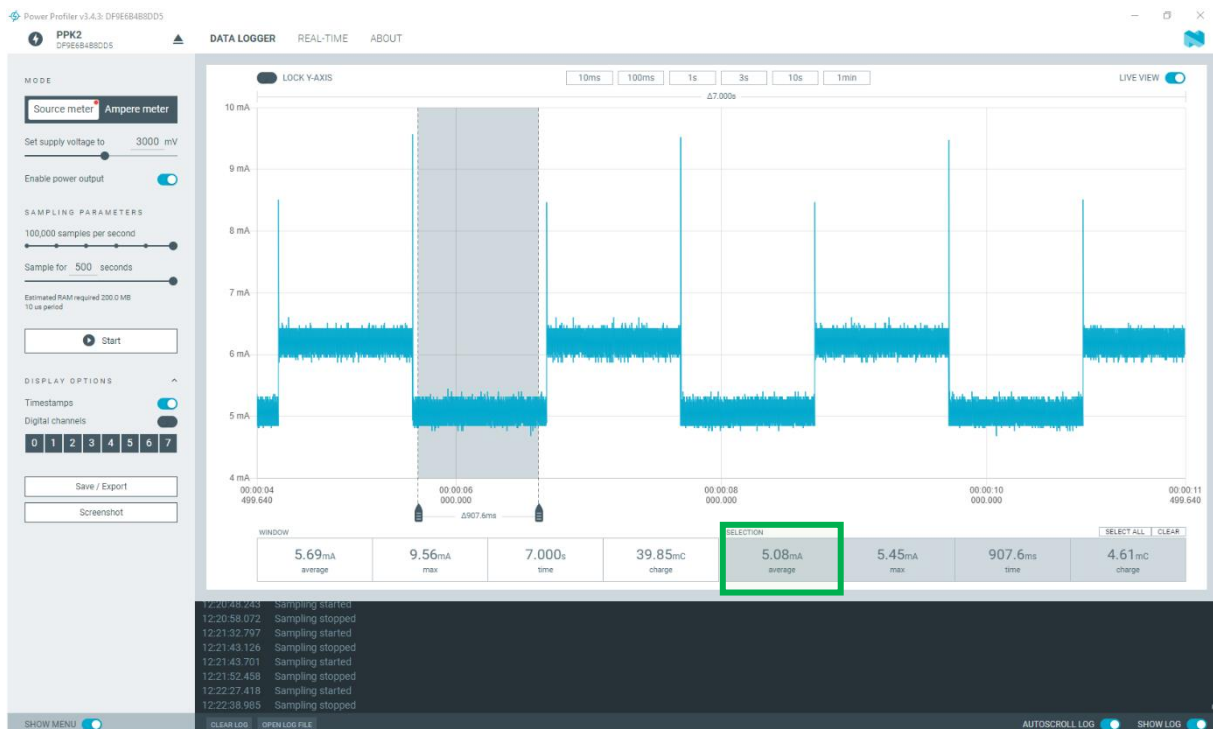


Figure 2. Power supply adjustments.

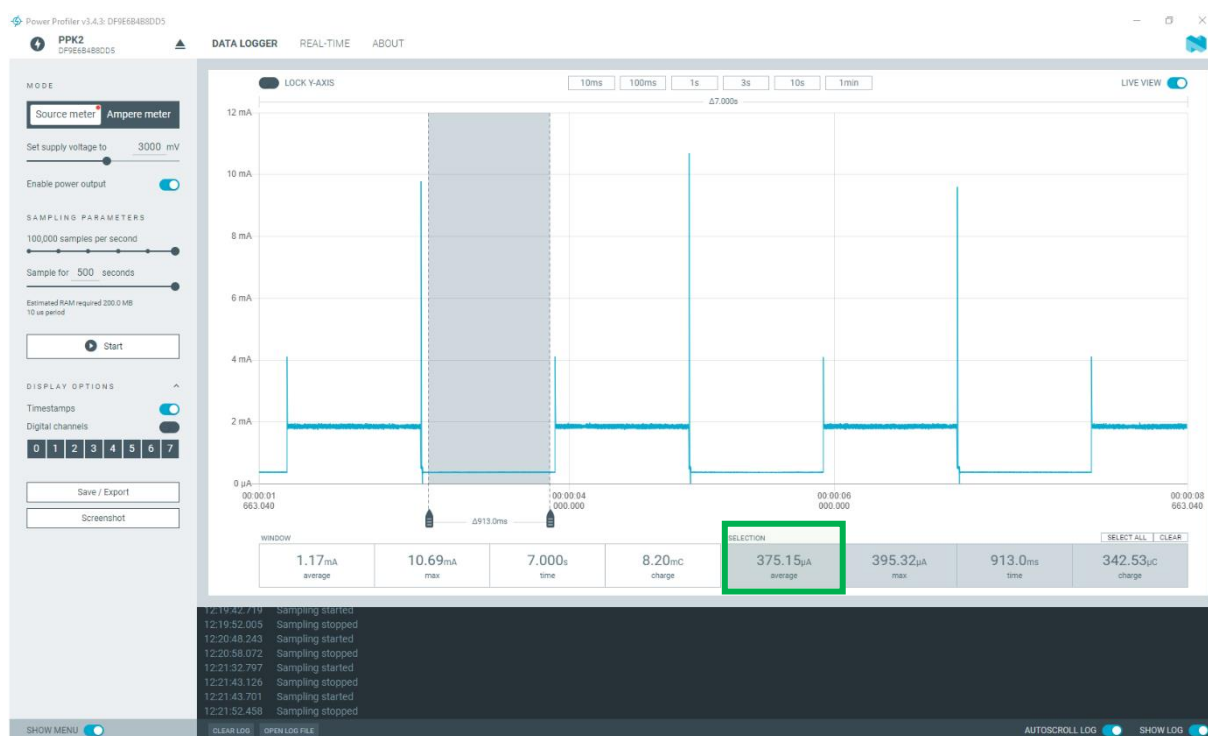
Figure 2 shows the power supply adjustments. Cutting the J20 trace prevents the current flowing from P3V3 back to U1. Optionally a 2x1 header can be soldered, to bypass the forward voltage of diode D12 when using any of the other power supply options.

Removing R74 isolates the SDA from the P3V3 supply. A 2x1 header was soldered on J3 to enable the SDA interface when programming the microcontroller.

### Cut trace J20:



Removing R74 (additionally):



## SWD adjustments

### SWD CONNECTOR

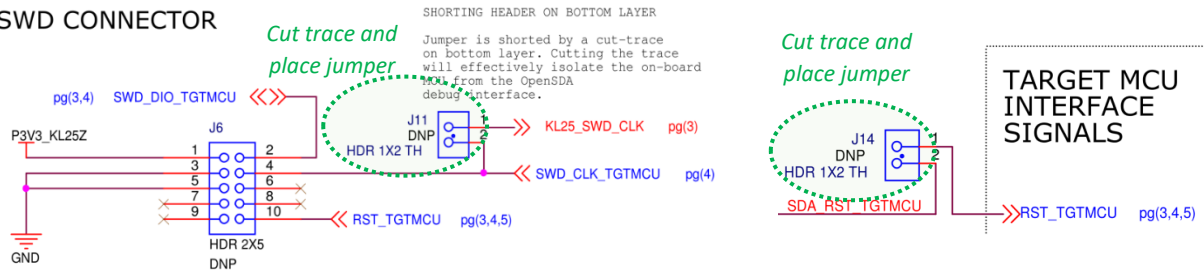
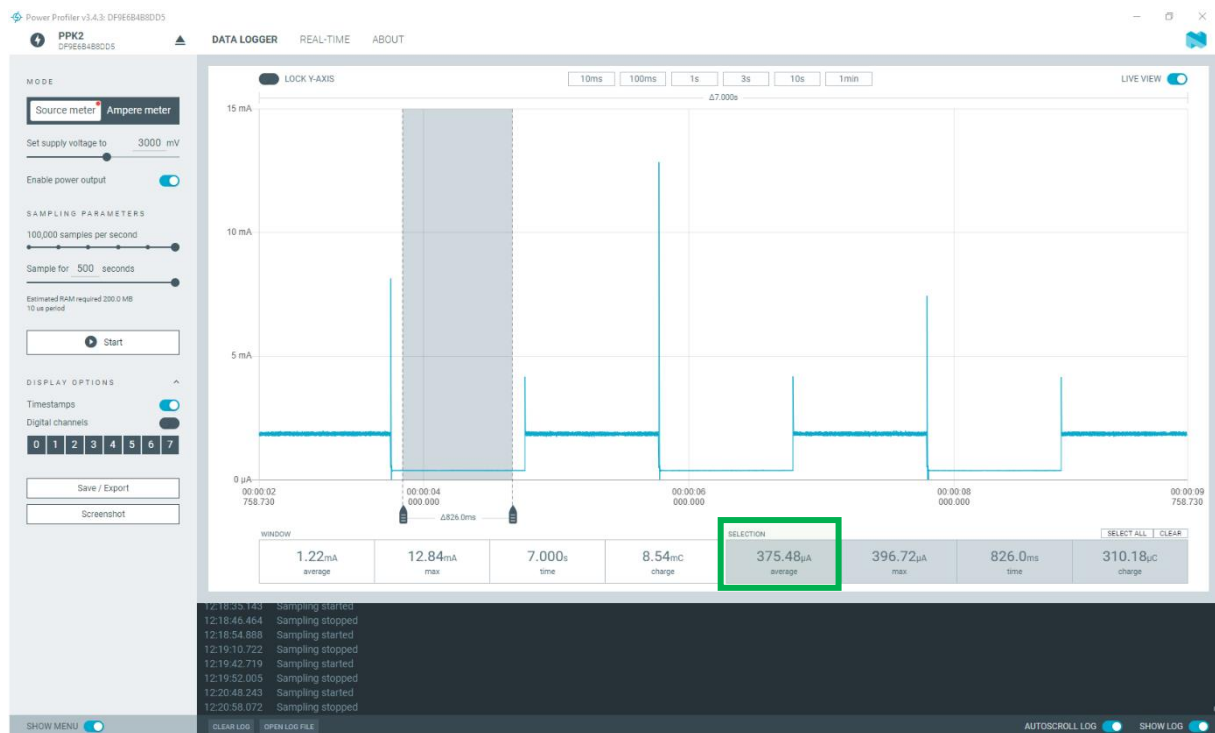


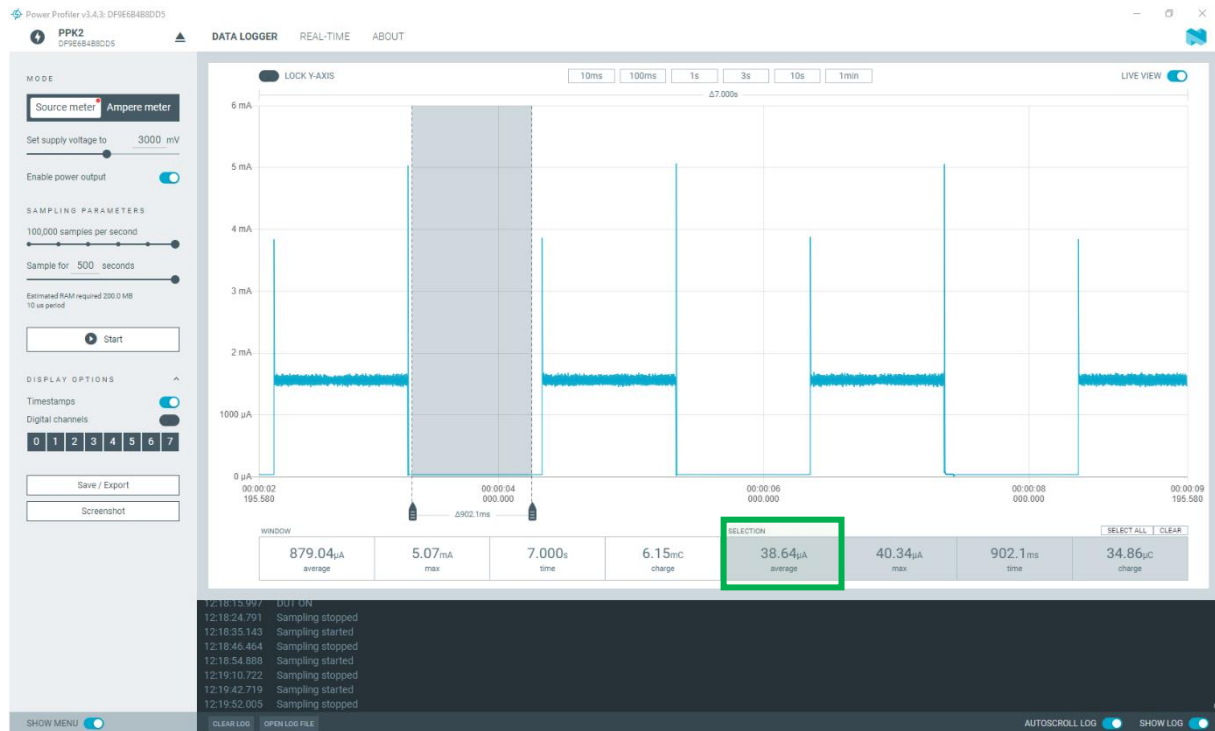
Figure 3. SWD adjustments.

Figure 3 shows the SWD interface adjustments. Header J11 isolates the SWD CLK signal. This, however, saves only approximately 1 $\mu$ A. Header J14 isolates the SWD RST signal from the SDA interface. Both headers must be shorted when programming the microcontroller.

Cut trace J11 (additionally):



Cut trace J14 (additionally):





## Notes

- The FRDM-KL25Z kit additionally features an MMA8451Q inertial sensor. After a power on reset (POR), the device is in Standby mode, using typically 1.8µA.
- To power the FRDM-KL25Z board from the optional coin cell holder (BT1), diode D7 must also be populated. Two suitable parts are:
  - Coin Cell Battery Holders TH COIN CELL BATTERY HOLDER 20mm  
Mouser order number [534-3003](#)
  - Schottky Rectifier, 20 V, 1 A, Single, SOD-123FL, 2 Pins, 350 mV  
Farnell order number [2918857](#)

A CR2032 battery has a capacity of 225mAh. In rest (not showing something on the OLED display, not fixing or not buzzing), the FRDM-KL25Z uses approximately 70µA. This means that, theoretically, the system should last for  $225\text{mAh} / 70\mu\text{A} = 3214$  hours. This is equal to 134 days.

## Appendix n