Master's Thesis

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Measurement

Software-defined oscilloscopes with terminal interface

Bc. Jozef Dujava

Supervisor: doc. Ing. Jan Fischer, CSc.
May 2022

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Dujava Jozef** Personal ID number: **466177**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Measurement**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master's thesis details

Master's thesis title in English:

Software-defined oscilloscopes with terminal interface

Master's thesis title in Czech:

Softwarově definované osciloskopy s terminálovým rozhraním

Guidelines:

Design and implement software-defined instruments with a terminal user interface. Display the measurements graphically on a PC. The "Dataplotter" application may be used for this purpose. The design shall focus on real-time and equivalent-time sampling oscilloscopes which will be used for teaching purposes. Additionally work on instruments such as: logic analyzer, event logger, counter, pulse and signal generators. Implement these on STM32 microcontrollers (primarily F303, F446, G431, F103, L072, F042, L412). Design the firmware so that only small modifications will be required to implement similar instruments on other types of STM32 microcontrollers.
Design a terminal user interface for controlling multiple instruments, fully implemented in the microcontroller firmware (without the need for specialty PC software). Also design any necessary additional electronic circuits (amplifiers, signal shapers) and verify their function in the implemented instruments. Determine the parameters and practical limits of such instruments.

Bibliography / sources:

[1] Yiu, J.: The Definitive Guide to ARM® Cortex®-M3 and Cortex®-M4 Processors,
[2] STMicroelectronics: RM0316, STM32F3 Reference manual
[3] Maier, J.: Univerzální GUI pro osciloskopické PC aplikace, BP, ČVUT- FEL, 2021

Name and workplace of master's thesis supervisor:

doc. Ing. Jan Fischer, CSc., Department of Measurement, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **29.06.2021** Deadline for master's thesis submission: _____

Assignment valid until:
by the end of winter semester 2022/2023

doc. Ing. Jan Fischer, CSc. Supervisor's signature	Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt	Student's signature

Acknowledgements

I would like to thank my supervisor, doc. Ing. Jan Fischer, CSc., for his patient guidance and insight. I also thank my family for supporting me throughout my studies.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

In Prague on 17 May 2022

.....
Bc. Jozef Dujava

Abstract

This thesis is focused on the design and implementation of software-defined instruments, primarily for teaching purposes. The main instrument is a mixed-signal oscilloscope capable of both real-time and equivalent-time sampling. Additionally, pulse generators, arbitrary generators and a frequency counter were also implemented. The implemented firmware was adapted for multiple types of STM32 microcontrollers. A terminal user interface for the software-defined instruments was implemented directly within the microcontroller firmware. The universal PC application 'Data Plotter' is used to display acquired oscilloscope data and access the terminal user interface. The parameters of the developed instruments were measured and evaluated.

Keywords: oscilloscope, software-defined instrument, virtual instrument, terminal user interface, stm32 microcontroller

Supervisor: doc. Ing. Jan Fischer, CSc.

Abstrakt

Táto práca sa zaoberá návrhom a realizáciou softvérovo definovaných prístrojov na výukové účely. Hlavným prístrojom je mixed-signal osciloskop vzorkujúci ako v reálnom tak aj v ekvivalentnom čase. Taktiež boli implementované ďalšie prístroje a to pulzné generátory, funkčné generátory a frekvenčný čítač. Implementovaný firmvér bol adaptovaný pre viaceré typy mikrokontrolérov STM32. Tieto softvérovo definované prístroje využívajú terminálové používateľské rozhranie implementované priamo vo firmvéri mikrokontroléru. Na zobrazenie záznamov osciloskopu a terminálového používateľského rozhrania je využitá univerzálna PC aplikácia 'Data Plotter'. Parametre vyvinutých prístrojov boli zmerané a vyhodnotené.

Kľúčové slová: osciloskop, softvérovo definovaný prístroj, virtuálny prístroj, terminálové používateľské rozhranie, stm32 mikrokontrolér

Preklad názvu: Softvérovo definované osciloskopy s terminálovým rozhraním

Contents

Figures

Tables

Chapter 1
Introduction

Laboratory instruments such as oscilloscopes and signal generators are indispensable for teaching and development of electronics. Unfortunately, professional commercially available instruments are generally costly and often not affordable for students or hobbyists. However, low-cost software-defined instruments (SDI) can prove to be adequate alternatives for many use cases. Also known as virtual instruments (VI), these may typically consist of a microcontroller (MCU) connected to a PC with minimal additional hardware necessary as the MCUs typically feature timers, Analog-to-Digital converters (ADCs) and Digital-to-Analog converters (DAC). Figure 1.1 shows a diagram of such an instrument.

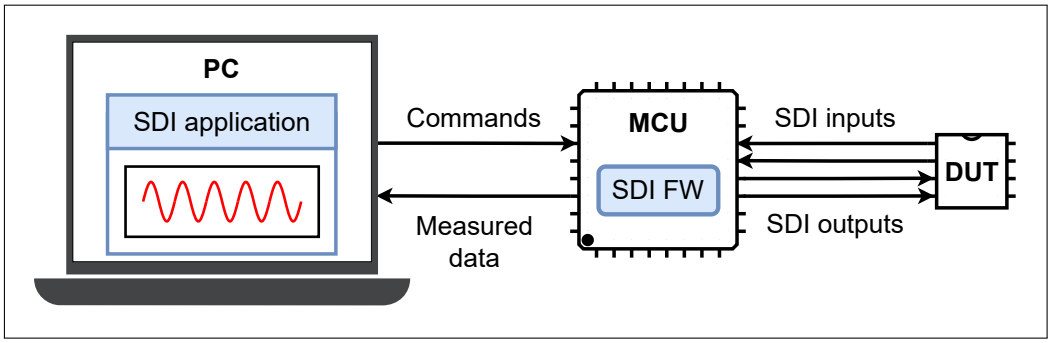

Figure 1.1: General diagram of a microcontroller-based software-defined instrument

The aim of this work was to implement these types of software-defined instruments for STM32 microcontrollers. These microcontrollers are utilized in various courses at the Faculty of Electrical Engineering, CTU, often being provided to students for use at home as well. Therefore, developing software-defined instruments for them allows most assignment troubleshooting to be done at home. This lets class time be used more effectively, focusing on teaching new concepts instead. Indeed, multiple software-defined instrument platforms for STM32 microcontrollers had been created at the Department of Measurement in the past for this purpose. However, the SDI platform developed in this work is more universal and provides additional functionality.

A unique feature of the implemented SDI platform is an oscilloscope capable of equivalent-time sampling. This allows sampling periodic input signals with a significantly higher equivalent sampling rate. An interesting application of this sampling method is step response analysis, where both the input and output signals of a system (device under test) can be captured with a significantly higher resolution than is possible with real-time sampling. Additionally, instruments such as a frequency counter and precise, synchronizable pulse and arbitrary generators were also developed.

Chapter 2
Analysis

2.1 Motivation

In both electronics development and teaching, laboratory measurement instruments such as oscilloscopes or logic analyzers are often a necessity. They allow the user to visualize signals in the tested circuit and greatly ease troubleshooting thereof. Other instruments such as signal generators can also prove to be very useful. Unfortunately, laboratory-grade instrumentation is typically rather costly and out of reach of many hobbyists and students. When students do have access to laboratories during their courses, such as at the Faculty of Electrical Engineering, CTU, there is often a certain time pressure or not enough instruments for everyone to use concurrently. Sometimes it may not even be possible to access the laboratories, such as the recent period of distant teaching during the Covid-19 pandemic.

To solve these issues and allow students to work on assignments and projects at home, an alternative approach using low-cost software-defined instruments (SDI) can be used. These can be based on STM32 microcontroller kits which are loaned out to students as part of multiple courses anyway. Then, to troubleshoot the assignments, two students can use one of their kits as a measurement instrument to test their firmware running on the other kit. Besides this application, such instruments can also be of good use to hobbyists for very little investment. The aim of this work is to implement such an SDI platform and make it available for a variety of STM32 microcontrollers.

2.2 State of the art

Multiple SDI projects have been created at the Department of Measurement, FEE CTU, in the past. These include oscilloscopes, logic analyzers, frequency counters and various types of generators. Typically, they combine the firmware of the microcontroller with a PC application created specifically for that project. This not only greatly increases the complexity of the initial implementation, but also makes adding features and support for other microcontrollers in the future more difficult. Some of the most recent and relevant SDI projects are listed below.

- **LEO - Little Embedded Oscilloscope** [1]
  - Provides multiple software-defined instruments:
    - 4-channel oscilloscope, sampling at up to 4 MSps
    - 2-channel signal generator/DC voltage source
    - 4-channel voltmeter
  - Only implemented for the STM32F303RE microcontroller
  - The associated PC application only works on Windows

- **Zero eLab Viewer (F0-Lab, G0-Lab)** [2]
  - Provides multiple software-defined instruments:
    - 3 or 4-channel oscilloscope
    - 3-channel voltmeter
    - pulse generator
  - Has versions for STM32F042F6 [7], STM32G030J6 [8], STM32F103C8 [6]
  - The "Zero eLab Viewer" PC application works on Windows and Linux
- **EMBO - Embedded Oscilloscope** [4]
  - Provides multiple software-defined instruments:
    - 4-channel oscilloscope
    - 4-channel logic analyzer
    - 4-channel voltmeter
    - 2 pulse generators
    - 1 signal generator
    - frequency counter
  - Adapted for STM32F103C8, STM32F103RE, STM32F303RE, STM32L412KB
  - The associated PC application works on Windows, MacOS and Linux
  - Created for the "Software defined oscilloscope based on STM32F103" master's thesis [10]
- **ELA - Logic Analyzer** [3]
  - A logic analyzer with:
    - 8 channels
    - up to 12 MHz sampling frequency
    - I2C, SPI, UART and Neopixel protocol decoding
    - 2 pulse generators
  - Only implemented for the STM32F303RE microcontroller
  - Uses the open source "PulseView" PC application – works on Windows, MacOS and Linux
  - Created for the "Microcontroller Based Logic Analyser" bachelor's thesis [9]
- **STM32 Virtual Counter** [5]
  - A frequency counter with:
    - frequency, period, frequency ratio measurements
    - duty cycle measurement
    - 2 synchronizable pulse generators
  - Adapted for the STM32F042F6, STM32F042K6, STM32F303RE microcontrollers
  - The associated PC application works on Windows, MacOS and Linux
  - Created for the "Microcontroller-based Virtual Instrument for Signal Analysis in the Modulation Domain" bachelor's thesis [11]

2.3 Goals of this work

The aim of this work is to implement a more universal SDI platform with additional instruments and features compared to the previously described projects. Specifically, the following instruments shall be included:

- Oscilloscope
  - Multiple analog channels
  - Normal and auto trigger
  - Support for both real-time and equivalent-time sampling
  - Precise sampling frequency adjustment
- Logic analyzer
  - As many channels as possible
  - Higher sampling rate than oscilloscope (for digital signals)
- Pulse generators
  - As many as possible
  - Precisely adjustable frequency and duty cycle
- Signal generators
  - As many as possible
  - Selection of generated function (e.g. sine wave, sawtooth)
  - Adjustable frequency, amplitude, DC offset and duty cycle
- Frequency counter
  - High frequency resolution
  - Large input frequency range

The capabilities of the developed software-defined instruments are ultimately decided by the MCU used. Given this variability, it was determined a terminal user interface (TUI) implemented in the MCU firmware would be best suited for this application. By using a universal terminal user interface, the need to create a custom PC application is eliminated, allowing a general-purpose terminal application to be used instead. The implemented terminal user interface shall:

- Facilitate full control of every virtual instrument
- Display precise values of measurements instrument parameters
- Be clear and easy to use, displaying only relevant information

Finally, all developed firmware shall be modular, automatically adapting to the capabilities of a given MCU. It shall also be portable, not requiring large modifications in order to add support for a new microcontroller.

2.4 Overview of considered STM32 microcontrollers

2.4.1 STM32F303RE microcontroller overview

The STM32F303RE is an ARM Cortex-M4 32-bit microcontroller with rich analog peripherals including 4 ADCs and two DAC channels. Table 2.1 shows a brief overview of the capabilities of this MCU. Table 2.2 describes the timers available in this microcontroller.

f_{CPU}	Up to 72 MHz
Flash	512 KiB
SRAM	64 KiB
CCMRAM	16 KiB
ADCs	Four 12-bit ADCs, up to 5 MSps
DACs	Two 12-bit DAC channels
DMA	Two controllers, 12 channels total

Table 2.1: Overview of STM32F303RE MCU capabilities

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels
Advanced	TIM1, TIM8, TIM20	16-bit	Up, Down, Up/Down	Any integer between 1 and 65536	Yes	4
General-purpose	TIM2	32-bit	Up, Down, Up/Down	Any integer between 1 and 65536	Yes	4
General-purpose	TIM3, TIM4	16-bit	Up, Down, Up/Down	Any integer between 1 and 65536	Yes	4
General-purpose	TIM15	16-bit	Up	Any integer between 1 and 65536	Yes	2
General-purpose	TIM16, TIM17	16-bit	Up	Any integer between 1 and 65536	Yes	1
Basic	TIM6, TIM7	16-bit	Up	Any integer between 1 and 65536	Yes	0

Table 2.2: Timers available in STM32F303RE (adapted from [23, p. 26])

2.4.2 STM32G431KB microcontroller overview

The STM32G431KB is an ARM Cortex-M4 32-bit microcontroller. Its analog peripherals are inferior to the STM32F303RE, however its maximum clock frequency is significantly higher. Table 2.3 shows a brief overview of the capabilities of this MCU. The timers available are detailed in Table 2.4.

f_{CPU}	Up to 170 MHz
Flash	128 KiB
SRAM	22 KiB
CCMRAM	10 KiB
ADCs	Two 12-bit ADCs, up to 4 MSps
DACs	Two 12-bit DAC channels
DMA	Two controllers, 12 channels total

Table 2.3: Overview of STM32G431KB MCU capabilities

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels
Advanced motor control	TIM1, TIM8	16-bit	Up, down, Up/down	Any integer between 1 and 65536	Yes	4
General-purpose	TIM2	32-bit	Up, down, Up/down	Any integer between 1 and 65536	Yes	4
General-purpose	TIM3, TIM4	16-bit	Up, down, Up/down	Any integer between 1 and 65536	Yes	4
General-purpose	TIM15	16-bit	Up	Any integer between 1 and 65536	Yes	2
General-purpose	TIM16, TIM17	16-bit	Up	Any integer between 1 and 65536	Yes	1
Basic	TIM6, TIM7	16-bit	Up	Any integer between 1 and 65536	Yes	0

Table 2.4: Timers available in STM32G431KB (adapted from [29, p. 32])

2.4.3 STM32F103C8 microcontroller overview

The STM32F103C8 is a widely used ARM Cortex-M3 32-bit microcontroller. Table 2.5 shows a brief overview of the capabilities of this MCU. The timers available are detailed in Table 2.6.

f_{CPU}	Up to 72 MHz
Flash	64 KiB
SRAM	20 KiB
ADCs	Two 12-bit ADCs, up to 1 MSps
DMA	One 7-channel controller

Table 2.5: Overview of STM32F103C8 MCU capabilities

Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/compare channels
TIM1	16-bit	Up, down, up/down	Any integer between 1 and 65536	Yes	4
TIM2, TIM3, TIM4	16-bit	Up, down, up/down	Any integer between 1 and 65536	Yes	4

Table 2.6: Timers available in STM32F103C8 (adapted from [34, p. 17])

2.4.4 STM32F042F6 microcontroller overview

The STM32F042F6 is an ARM Cortex-M0 32-bit microcontroller. Table 2.7 shows a brief overview of the capabilities of this MCU. The timers available are detailed in Table 2.8.

f_{CPU}	Up to 72 MHz
Flash	32 KiB
SRAM	6 KiB
ADCs	One 12-bit ADC, up to 1 MSps
DMA	One 5-channel controller

Table 2.7: Overview of STM32F042F6 MCU capabilities

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/compare channels
Advanced control	TIM1	16-bit	Up, down, up/down	integer from 1 to 65536	Yes	4
General purpose	TIM2	32-bit	Up, down, up/down	integer from 1 to 65536	Yes	4
	TIM3	16-bit	Up, down, up/down	integer from 1 to 65536	Yes	4
	TIM14	16-bit	Up	integer from 1 to 65536	No	1
	TIM16 TIM17	16-bit	Up	integer from 1 to 65536	Yes	1

Table 2.8: Timers available in STM32F042F6 (adapted from [36, p. 21])

2.4.5 STM32L072KZ microcontroller overview

The STM32L072KZ is an ultra-low-power ARM Cortex-M0+ 32-bit microcontroller. Table 2.9 shows a brief overview of the capabilities of this MCU. The timers available are detailed in Table 2.10.

f_{CPU}	Up to 32 MHz
Flash	192 KiB
SRAM	20 KiB
ADCs	One 12-bit ADC, up to 1 MSps
DMA	One 7-channel controller

Table 2.9: Overview of STM32L072KZ MCU capabilities

Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/compare channels
TIM2, TIM3	16-bit	Up, down, up/down	Any integer between 1 and 65536	Yes	4
TIM21, TIM22	16-bit	Up, down, up/down	Any integer between 1 and 65536	No	2
TIM6, TIM7	16-bit	Up	Any integer between 1 and 65536	Yes	0

Table 2.10: Timers available in STM32L072KZ (adapted from [39, p. 31])

2.4.6 STM32L412KB microcontroller overview

The STM32L412KB is an ultra-low-power ARM Cortex-M4 32-bit microcontroller. Table 2.11 shows a brief overview of the capabilities of this MCU. The timers available are detailed in Table 2.12.

f_{CPU}	Up to 80 MHz
Flash	128 KiB
SRAM	40 KiB
ADCs	Two 12-bit ADCs, up to 5 MSps
DMA	Two controllers, 14 channels total

Table 2.11: Overview of STM32L412KB MCU capabilities

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels
Advanced control	TIM1	16-bit	Up, down, Up/down	Any integer between 1 and 65536	Yes	4
General-purpose	TIM2	32-bit	Up, down, Up/down	Any integer between 1 and 65536	Yes	4
General-purpose	TIM15	16-bit	Up	Any integer between 1 and 65536	Yes	2
General-purpose	TIM16	16-bit	Up	Any integer between 1 and 65536	Yes	1
Basic	TIM6	16-bit	Up	Any integer between 1 and 65536	Yes	0

Table 2.12: Timers available in STM32L412KB (adapted from [37, p. 40])

2.4.7 STM32F446RE microcontroller overview

The STM32F446RE is a high-end ARM Cortex-M4 32-bit microcontroller with rich analog peripherals including 3 ADCs and two DACs. It also offers the largest variety of timers among all considered STM32 microcontrollers. Table 2.13 shows a brief overview of the capabilities of this MCU. Table 2.14 describes the timers available in this microcontroller.

f_{CPU}	Up to 180 MHz
Flash	512 MiB
SRAM	128 KiB
ADCs	Three 12-bit ADCs, up to 2.4 MSps
DACs	Two 12-bit DAC channels
DMA	Two controllers, 16 channels total

Table 2.13: Overview of STM32F446RE MCU capabilities

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels
Advanced-control	TIM1, TIM8	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4
General purpose	TIM2, TIM5	32-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4
	TIM3, TIM4	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4
	TIM9	16-bit	Up	Any integer between 1 and 65536	No	2
	TIM10, TIM11	16-bit	Up	Any integer between 1 and 65536	No	1
	TIM12	16-bit	Up	Any integer between 1 and 65536	No	2
	TIM13, TIM14	16-bit	Up	Any integer between 1 and 65536	No	1
Basic	TIM6, TIM7	16-bit	Up	Any integer between 1 and 65536	Yes	0

Table 2.14: Timers available in STM32F446RE (adapted from [33, p. 30])

Chapter 3

Developed software-defined instrument platform

A firmware package implementing a variety of software-defined instruments for multiple types of STM32 microcontrollers was developed in this work. It provides the following instruments: mixed-signal oscilloscope, pulse generators, arbitrary generators, frequency counter. Each type of instrument and its implementation are discussed in detail in chapters 5 through 8. The developed SDI platform was named **"Versatile STM32 Virtual Instrument (VSVI)"**. In order to use it, the MCU must first be programmed with the VSVI firmware. Then, the MCU is connected to any desktop or laptop PC via USB. The PC provides the user interface and power supply (USB) for the SDI platform. If the microcontroller does not have a USB peripheral, a USB-to-UART converter (e.g. CH340 or ST-Link) is used. A diagram of the implemented virtual instrument is shown in Figure 3.1.

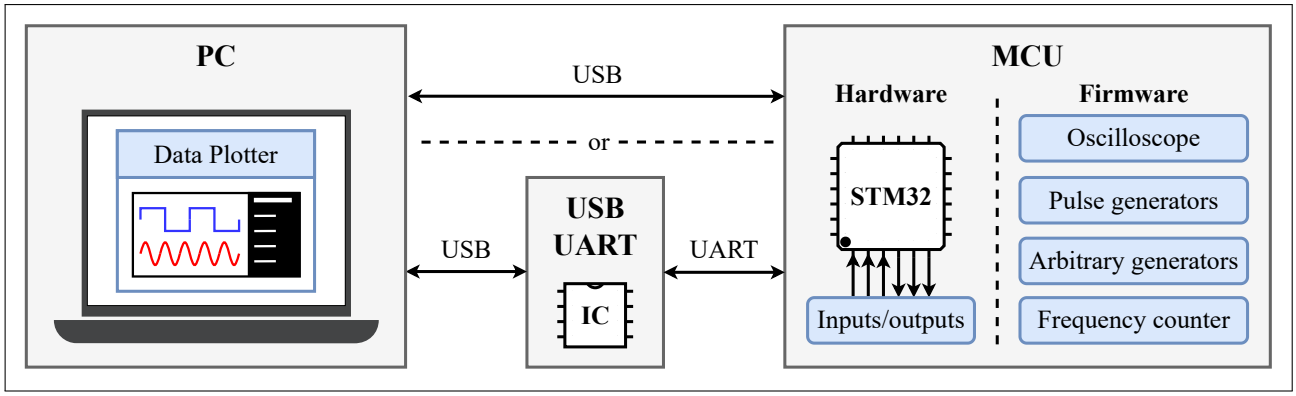

Figure 3.1: Diagram of the developed software-defined instrument platform

Instead of developing a specialized accompanying PC application, the already existing "Data Plotter" application [12] is used. It provides a comprehensive GUI for oscilloscope-type instruments. It can receive waveform data and send commands to the connected MCU as serial data via USB in Virtual COM Port mode. A terminal emulator window is also embedded in the application. This is used to display a terminal user interface (TUI) implemented within the VSVI firmware, as detailed in chapter 4. This interface gives the user full control of all the implemented virtual instruments. No modifications of the Data Plotter application were necessary, demonstrating the truly universal nature of this terminal-based approach. Figure 3.2 shows the typical breadboard-based usage of the VSVI platform, specifically with the STM32G431KB microcontroller. The corresponding block diagram of the available software-defined instruments is shown in Figure 3.3. The main program loop of the implemented VSVI firmware is illustrated in Figure 3.4.

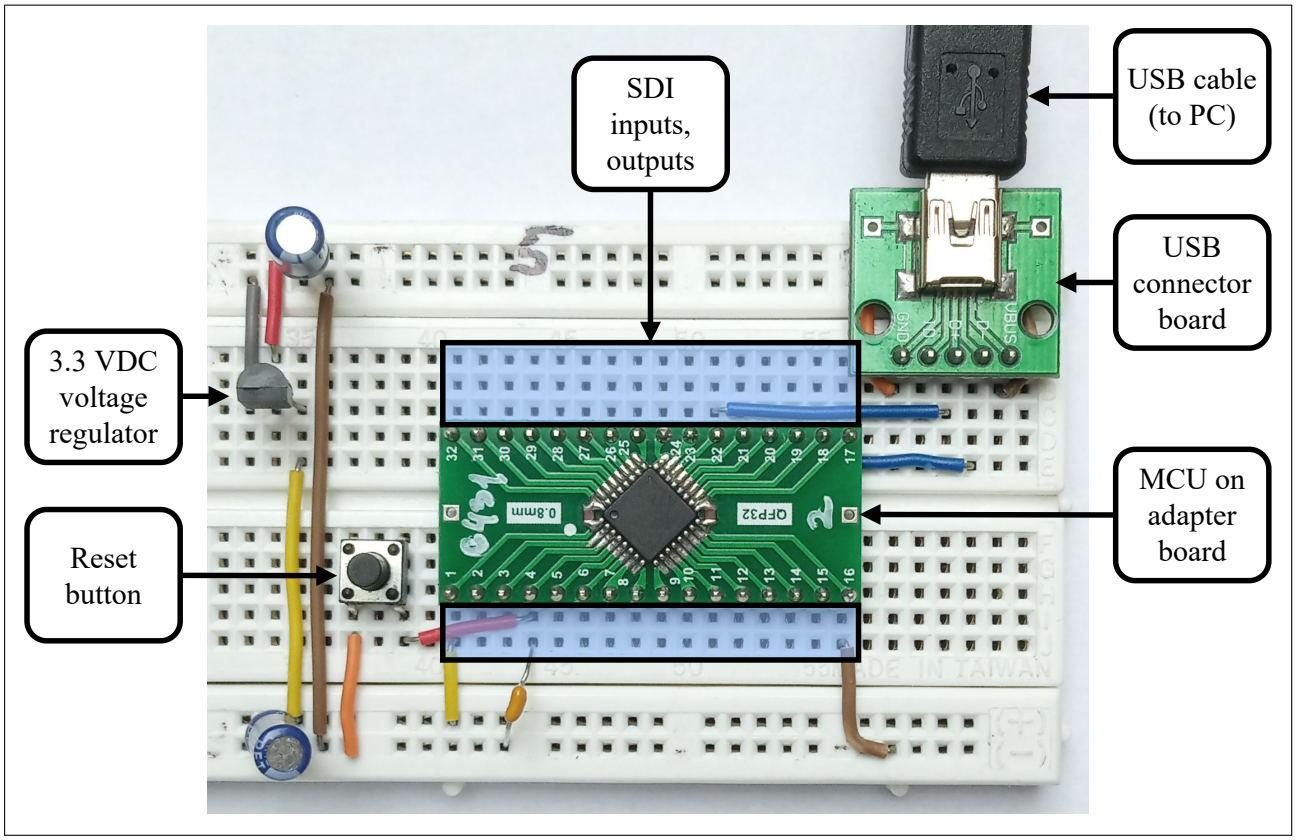

Figure 3.2: Developed software-defined instrument platform in use on a breadboard (STM32G431KB)

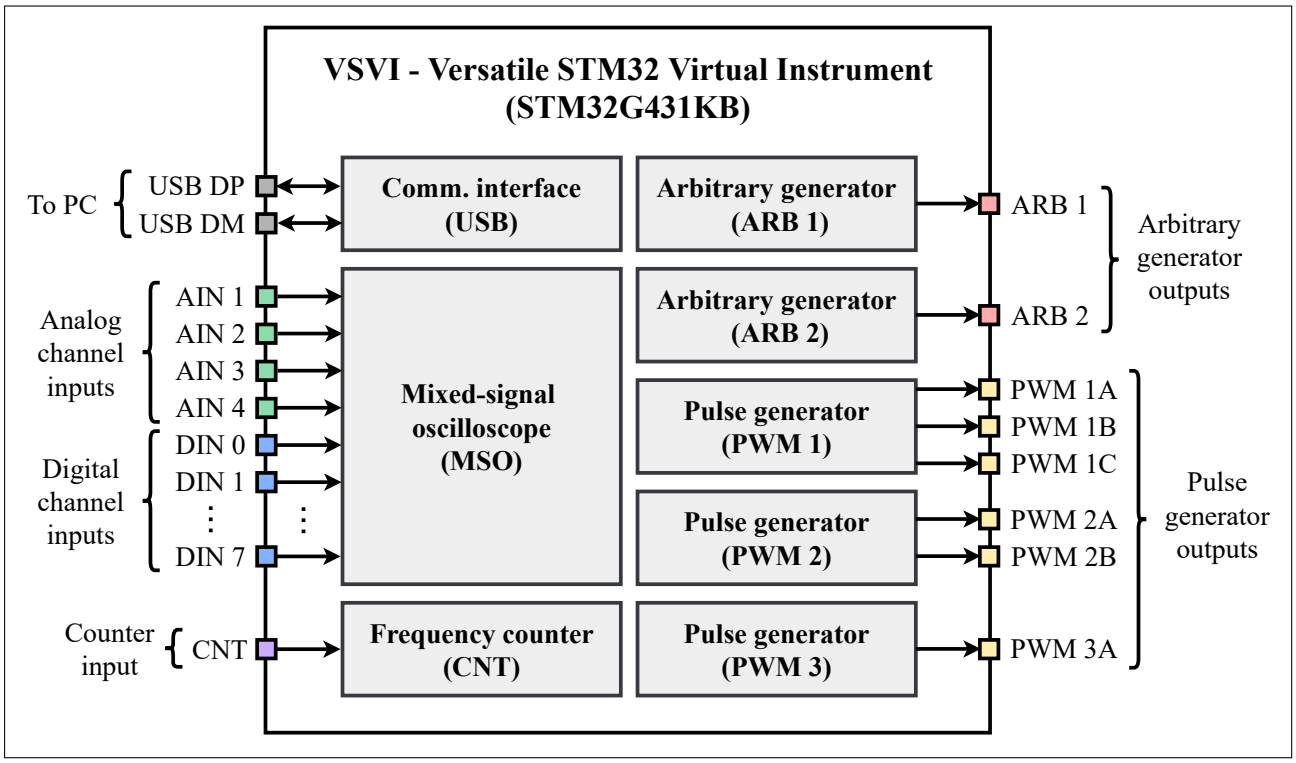

Figure 3.3: Block diagram of software-defined instruments implemented for STM32G431KB

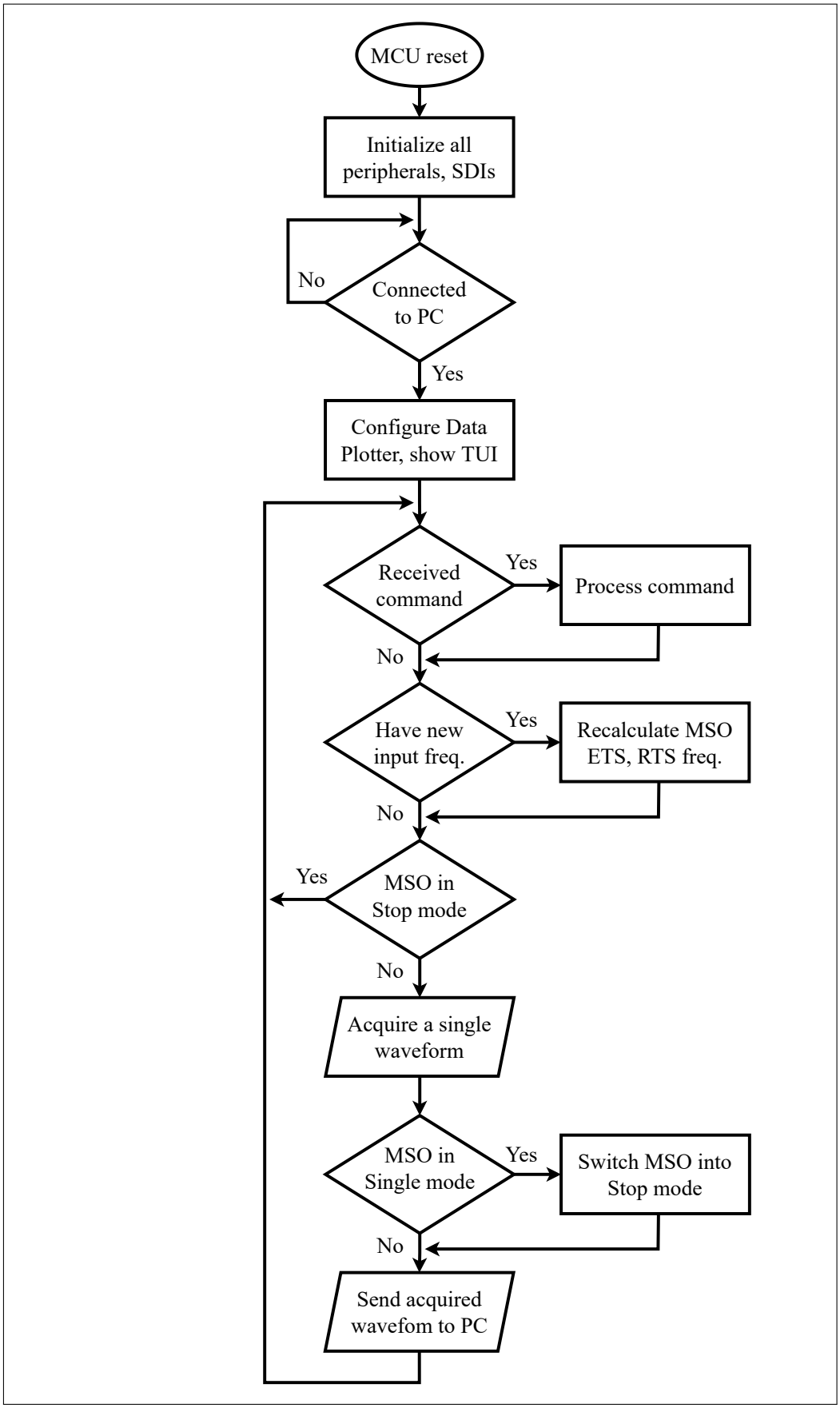

Figure 3.4: Main program loop of the developed SDI firmware

The implemented firmware was optimized primarily for the mixed-signal oscilloscope, which is the main focus of this work. Nonetheless, all instruments can be used concurrently without any issues. However, there are certain STM32 MCU hardware limitations (e.g. DMA throughput) that affect the maximum achievable SDI specifications, for example the maximum oscilloscope sampling rate. The VSVI firmware automatically adapts to these limitations by not allowing the user to set instrument parameters outside their reliable ranges – ensuring the correct functionality of the SDIs at all times. This feature stands in contrast to other previously developed SDI platforms such as LEO or EMBO where corruption of the measured data is expected in some use cases.

3.1 Development of SDI firmware for STM32 microcontrollers

The firmware developed in this work primarily had to implement a variety of different types of software-defined instruments. This includes configuration and utilization of MCU hardware resources and sending measured data to the PC. The firmware also receives and processes various instrument commands from the PC. Additionally, it is also responsible for displaying the terminal user interface, including dynamic updates based on instrument configuration. All of this is achieved from within the main program loop illustrated in Figure 3.4. As can be seen, the operation of the firmware revolves mainly around the mixed-signal oscilloscope. Received commands are only processed when the oscilloscope acquisition is stopped, as PC communication interferes with its sampling by increasing load on the CPU and DMA controllers. The implementations of each type of instrument and the terminal user interface are described in more detail in the following chapters 4 through 8.

3.1.1 HAL library and CubeMX software

A Hardware Abstraction Layer (HAL) library is provided for STM32 microcontrollers by their manufacturer, STMicroelectronics. This library serves to simplify the low-level programming of the MCU hardware. The programmer needs only to use the CubeMX software (shown in Figure 3.5) to select and configure a given microcontroller – the startup configuration code is generated automatically. This approach may be suitable for most applications, but is not ideal for this work for a variety of reasons:

- The CubeMX software only generates startup code, but the hardware configuration of the MCU must often change at runtime in this application.
- HAL library functions are generally not suitable for low-latency applications, as they include a lot of code that is not strictly necessary.
- The HAL library code introduces substantial Flash size overhead. For example, when using the HAL library for timers, the resulting firmware was over 10 kiB larger compared to a version using the MCU registers directly.
- Using the HAL library has not proved to significantly ease programming. The documentation for this library is often sparse and not straightforward. It is often not clear what exactly a certain function will do, or when it should be called. This often lead to analyzing the source code of HAL functions while debugging – rendering using the library rather pointless.

Compared to the HAL library documentation, the reference manuals for each MCU are much more detailed in the capabilities of each peripheral as well as the steps necessary to achieve a given functionality. It was therefore decided the firmware would be programmed primarily using MCU registers directly. Nonetheless, the HAL library and the CubeMX software were still used for some parts of the firmware, namely:

- Clock configuration (RCC)
- USB interface (Virtual COM Port)
- Programming Flash memory

These parts of the firmware would be tedious to implement using registers only, and even more difficult to adapt to multiple types of microcontrollers. They are also well suited for HAL since they do not need to change at runtime and are not part of time-intensive procedures.

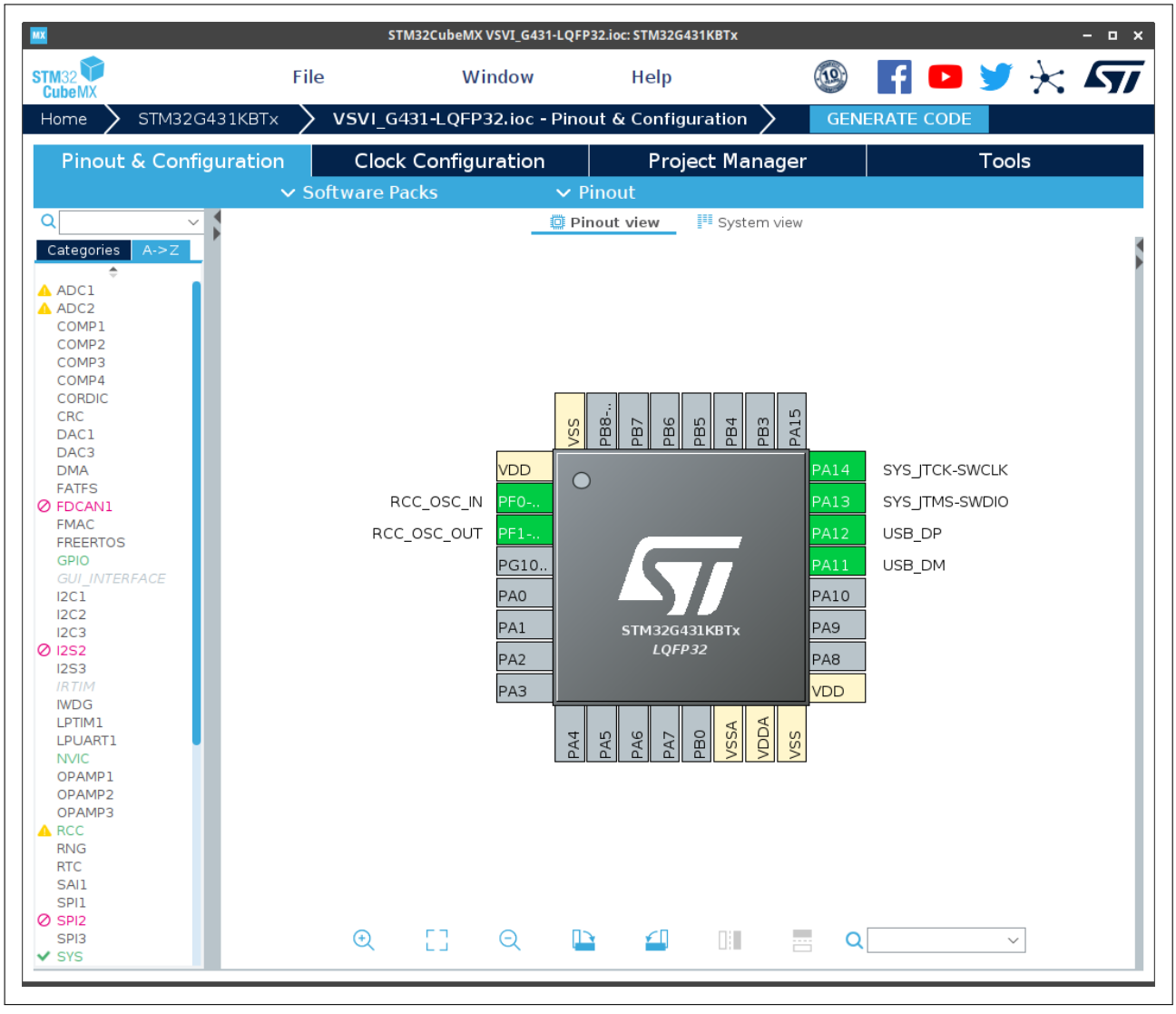

Figure 3.5: CubeMX software for generation of STM32 startup code using HAL libraries

3.1.2 Effects of STM32 MCU differences on firmware development

One of the major objectives of this work was to develop firmware that would be easily adaptable for many types of STM32 microcontrollers. This is a significantly more difficult task than may be obvious at first glance due to the many differences among these MCUs. Parameters like maximum clock frequency or Flash and SRAM size are relatively easy to adapt to, although they limit the performance of the virtual instruments. However, there are also multiple differences which are significantly more difficult to account for, such as the number and capabilities of:

- **Analog-To-Digital converters (ADCs)**: Ranging from a single ADC capable of sampling at only 1.1 MSps (STM32L072KZ) to 4 ADCs sampling at up to 5.1 MSps (STM32F303RE).
- **Timers**: Ranging from just 4 (STM32L072KZ) to 11 (STM32F303RE) with different capabilities
- **Digital-To-Analog converters (DACs)**: Ranging from 2 DAC channels with update rate 1 MSps (STM32G431KB, STM32F303RE, STM32L072KZ) to none (STM32F103C8, STM32L412KB).

More substantial differences are found in the embedded Direct Memory Access (DMA) controllers. There are either 1 or 2 in a given MCU, often depending on the package size. DMA controllers have different number of channels with differing request mapping methods:

- **DMAMUX**: Additional peripheral that allows arbitrary mapping of any DMA request to any channel of any DMA controller. Can also generate its own requests with multiple synchronization options (STM32G431KB) [31, p. 405].
- **Logical OR**: Multiple requests are connected to a single channel via a logical OR function. A small number of requests may be remapped between DMA controllers using SYSCFG registers (STM32F103, STM32F303)[26, p. 268].
- **Per-channel multiplexing**: Multiple requests are connected to each channel via a multiplexer. A configuration register selects a specific request to be connected through to the DMA controller (STM32F446, STM32L072, STM32L412)[38, p. 296].

The STM32F103C8 microcontroller even differs in basic functionality such as selecting the GPIO pin alternate function – using a set of alternate function registers (AFIO) to both configure the EXTI interrupt lines and to select the alternate function from a set of remaps on a per-peripheral basis (e.g. TIM CCx) [35, p. 175]. In the other microcontrollers, the EXTI lines are configured using SYSCFG registers while alternate functions are selected on a per-pin basis using GPIO configuration registers.

Outright firmware compatibility is not guaranteed even when two microcontrollers offer the same feature, controlled by the same registers. This is due to differences in register naming in the MCU driver, i.e. header file containing macros with register addresses. For example, the EXTI pending register is named "PR" in the STM32G431KB driver but "PR1" in the STM32F303RE driver. Both registers are named "EXTI_PR1" in the respective MCU reference manuals. The GPIO registers are similarly problematic, for example the "OSPEEDR" register is incorrectly named "OSPEEDER" in the STM32F303RE driver.

3.1.3 Overview of developed firmware code base

All parts of the code developed in this work use the `APP_` prefix to distinguish them from other code (HAL and standard C libraries). The firmware is divided into modules corresponding to specific MCU peripherals and software-defined instruments. Each module consists of a header file (`/Inc/APP/<module>.h`) and a source file (`/Src/APP/<module>.c`). Header files contain macros, type definitions and function declarations. The source files contain the actual implementation, which is kept as MCU-agnostic as possible. However, C preprocessor conditionals had to be used to select the appropriate implementation in some places, such as DMA request mapping or alternate function configuration mentioned in section 3.1.2. These apply to a whole family of MCUs, therefore their modification should not be necessary in most cases.

Each instance of a peripheral or SDI has a dedicated configuration (`APP_<module>_cfg`) and data (`APP_<module>_data`) structure. Data structures are variable (stored in SRAM) and hold all state variables of a given instance. Configuration structures are constant (stored in Flash) and contain configuration parameters specific to the MCU used. All of them are consolidated in one source file (`/Src/APP/platform/<platform>.c`) for easy portability. An accompanying header file (`/Inc/APP/platform/<platform>.h`) contains all platform-specific macros and type definitions. The MCU-specific configuration files contain parameters such as:

- The number of instruments available for each type and their specifications, e.g. maximum oscilloscope sampling rates
- Peripheral (TIM, ADC, DAC, DMA) configurations for each instrument
- Interrupt vectors and handlers
- GPIO pin assignments, alternate functions

A set of utility macros and functions was also implemented, for example simplified functions for converting floating point values to strings and vice versa – saving a large amount of Flash by not using the `_printf_float` functions from the `stdio.h` library. Similarly, a sine function approximation using Taylor series was implemented in order to avoid including the `math.h` library. Other implemented functions include GPIO configuration, string alignment and Base64 en/decoding (used to store arbitrary waveform data more efficiently in configuration profiles).

3.2 Using STM32 microcontroller timers

The STM32 microcontroller timers are used by each implemented software-defined instrument. As such, it was necessary to understand their operation and develop a set of functions for using them (to avoid using HAL libraries as previously mentioned). There are various types of timers with different capabilities embedded within STM32 MCUs, namely basic, general-purpose and advanced timers. The block diagram of a general-purpose timer is shown in Figure 3.6. The operation of a timer is controlled by its registers, primarily:

- **PSC**: Prescaler – divides the frequency of the incoming clock signal such that a single positive output signal transition is generated after every PSC positive input signal transitions.
- **CNT**: Counter – incremented with each positive transition of the signal coming from the prescaler.

- **ARR**: Auto-Reload Register – a timer update is generated when the counter reaches the value in this register. This resets the counter and optionally generates an update interrupt (UI) or DMA request.

- **CCRx**: Capture/Compare register of channel "x". In input capture mode, the value of CNT is captured in this register when the capture event occurs, typically when an edge is detected on the "CHx" input. In output compare mode, the comparison between CNT and CCRx values determines the output value of the timer channel according to the output compare mode used. A capture/compare interrupt (CCxI) or DMA request can also be generated.

Typically, all of the CNT, ARR and CCRx registers are 16 bits wide, with values between 0 and 65535. Some MCUs offer a single 32-bit timer (still with 16-bit prescaler). The frequency of the CHx output compare signals and the timer updates is given by

$$f = \frac{f_{TIM}}{(PSC + 1) \cdot (ARR + 1)}, \tag{3.1}$$

i.e. 1 must be added to the values of the PSC and ARR registers to determine the actual number of cycles within a timer period. The output compare modes used in this work are:

- **Toggle**: The output signal is toggled (high-to-low or low-to-high) every time the counter register value matches the capture/compare register value, i.e. when

$$\text{CNT} = \text{CCRx}. \tag{3.2}$$

- **PWM mode 1**: The output signal is high while the counter register value is below the capture/-compare register value, i.e. when

$$0 \leq \text{CNT} < \text{CCRx}. \tag{3.3}$$

  The output signal is low for the rest of the timer period, i.e. when

$$\text{CCRx} \leq \text{CNT} \leq \text{ARR} \tag{3.4}$$

  The duty cycle of the output signal is then

$$D = \frac{CCRx}{ARR + 1} \cdot 100\ \%. \tag{3.5}$$

  Therefore, if multiple output compare channels of the same timer use this mode, all of their rising edges are aligned (with the start of the timer period) while their pulse widths/duty cycles can vary according to the values of their CCRx registers. Both the input capture/output compare modes are configured using the CCMRx register.

Another important feature of the STM32 timers are their trigger and slave mode controllers. These allow synchronization of timers with other timers or other peripherals, e.g. using a timer as a timebase for triggering ADC conversions. The timer slave modes used in this work include:

- **Trigger**: The timer is started on a rising edge of the trigger signal.
- **Gated**: The timer runs while the trigger signal is high, stops when the trigger signal goes low.
- **External clock mode 1**: The timer is clocked by the trigger signal, incrementing the counter on every rising edge.

These slave modes are configured using the SMCR register, while the trigger output (TRGO) signal is selected within the CR2 register. The trigger signal can come from another timer via the ITRx internal trigger connection, the external trigger (ETR) pin, or the filtered input channel 1 or 2 [26, p. 650].

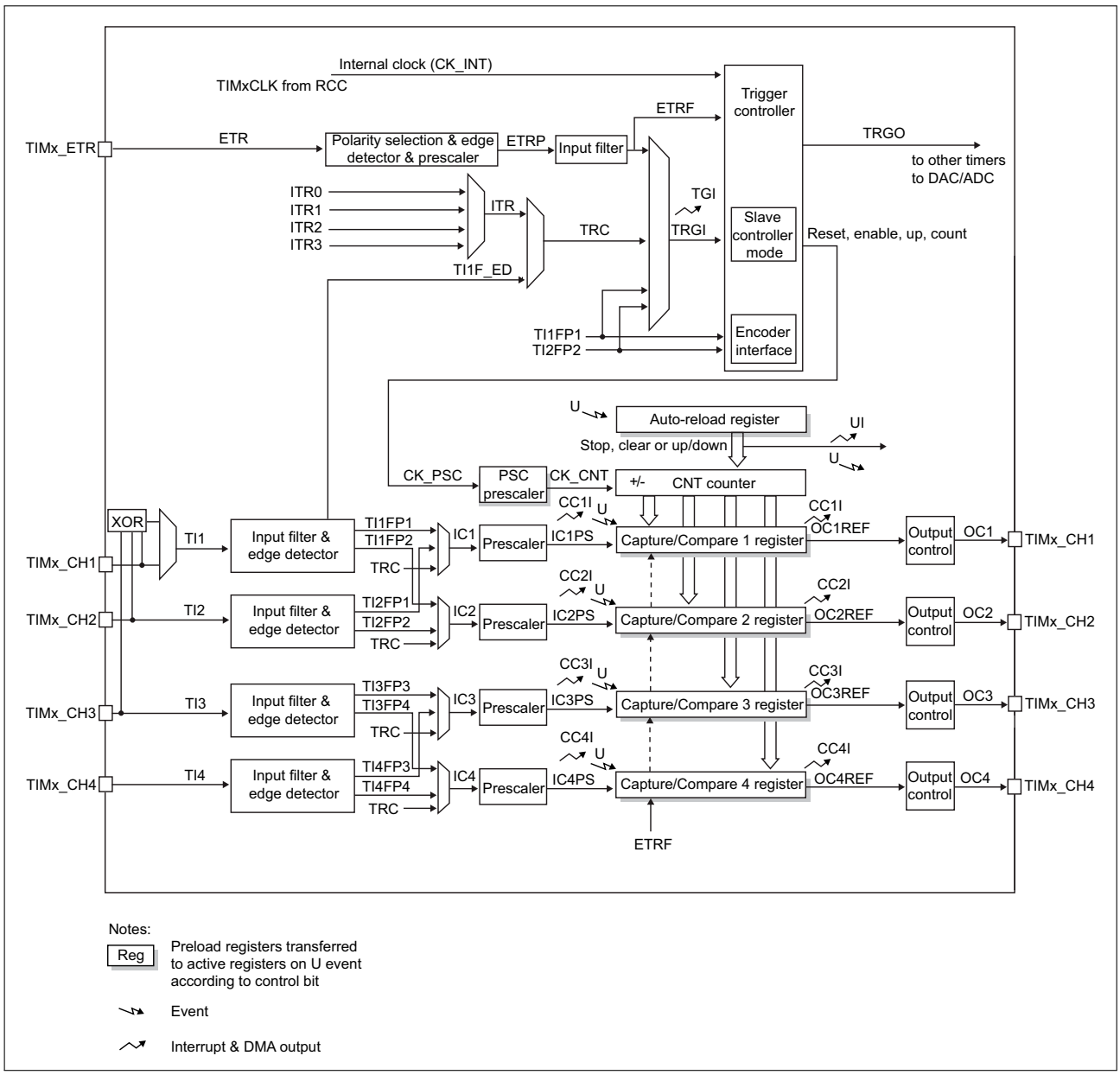

Figure 3.6: General-purpose timer block diagram. Adapted from [40, p. 446].

3.2.1 Algorithm for computing STM32 timer register values (PSC, ARR)

An algorithm for computing the values of the PSC and ARR registers needed to achieve a given timer frequency according to 3.1 was developed. This algorithm iteratively finds the two closest factorizations of the integer frequency ratio

$$N_f = \left\lfloor \frac{f_{TIM}}{f} \right\rfloor \tag{3.6}$$

where f_{TIM} is the MCU's timer clock frequency and f the requested frequency. The algorithm ensures that the requested frequency falls between the two actually achievable frequencies given by the two sets of factors computed. In the end, the option that is closest to the requested frequency is picked. The algorithm works as follows:

1. Initialize the loop variables ARR, PSC to 1, the *low* and *high* register values PSC_L, ARR_L and PSC_H, ARR_H to 0. Also initialize the minimal remainders $R_{L,min}, R_{H,min}$ to infinity.
2. Set

   $$PSC = \left\lceil \frac{N_f}{ARR_{max}} \right\rceil, \tag{3.7}$$

   this is the minimum prescaler value which ensures $ARR \leq ARR_{max}$.
3. While $PSC^2 \leq N_f$, do:
   - Compute

     $$ARR = \lfloor N_f / PSC \rfloor, \tag{3.8}$$
     $$R_L = N_f \mod PSC, \tag{3.9}$$
     $$R_H = PSC - R_L. \tag{3.10}$$

   - If $R_L < R_{L,min}$, set

     $$R_{L,min} = R_L, \tag{3.11}$$
     $$ARR_L = ARR - 1, \tag{3.12}$$
     $$PSC_L = PSC - 1. \tag{3.13}$$

   - If $R_H < R_{H,min}$, set

     $$R_{H,min} = R_H, \tag{3.14}$$
     $$ARR_H = ARR, \tag{3.15}$$
     $$PSC_H = PSC - 1. \tag{3.16}$$

   - Set $PSC = PSC + 1$.
4. Now,

   $$N_L = (PSC_L + 1)(ARR_L + 1) \tag{3.17}$$

   is the closest possible frequency ratio such that $N_L \leq N_f$ and

   $$N_H = (PSC_H + 1)(ARR_H + 1) \tag{3.18}$$

   is the closest possible frequency ratio such that $N_H \geq N_f$.

5. Compute the two frequencies

$$f_L = \frac{f_{TIM}}{N_L}, \tag{3.19}$$

$$f_H = \frac{f_{TIM}}{N_H} \tag{3.20}$$

and find the absolute errors between them and the desired frequency f, i.e.

$$e_L = |f - f_L|\,, \tag{3.21}$$

$$e_H = |f - f_H|\,. \tag{3.22}$$

6. Finally, pick the set of values that gave the frequency with the lower absolute error. This is the closest frequency that can be generated.

The loop variables ARR, PSC are the actual factors of N_f as used in eq. 3.6, while ARR_L, PSC_L and ARR_H, PSC_H are timer register values where 1 is subtracted from the real factor values. It is only necessary to increment PSC up to the square root of N_f in order to find all the factors, since those above the square root will have been computed as ARR values already. This also ensures that $PSC \leq ARR$ which gives the highest possible duty cycle resolution.

Chapter 4

User interface for software-defined instruments

The typical approach of SDI platforms implemented in the past was to pair the MCU firmware with a custom PC application to provide a graphical user interface (GUI) for displaying measured data and controlling the instruments. The PC application for the popular Little Embedded Oscilloscope (LEO) platform is shown in Figure 4.1 as an example. This approach certainly has advantages, such as familiar user experience and visual appeal. However, it is not without its disadvantages, namely the need to implement this specialized PC application in the first place. Moreover, it needs to be adapted every time changes are made to the SDI firmware, such as adding new features or support for new MCUs with different capabilities.

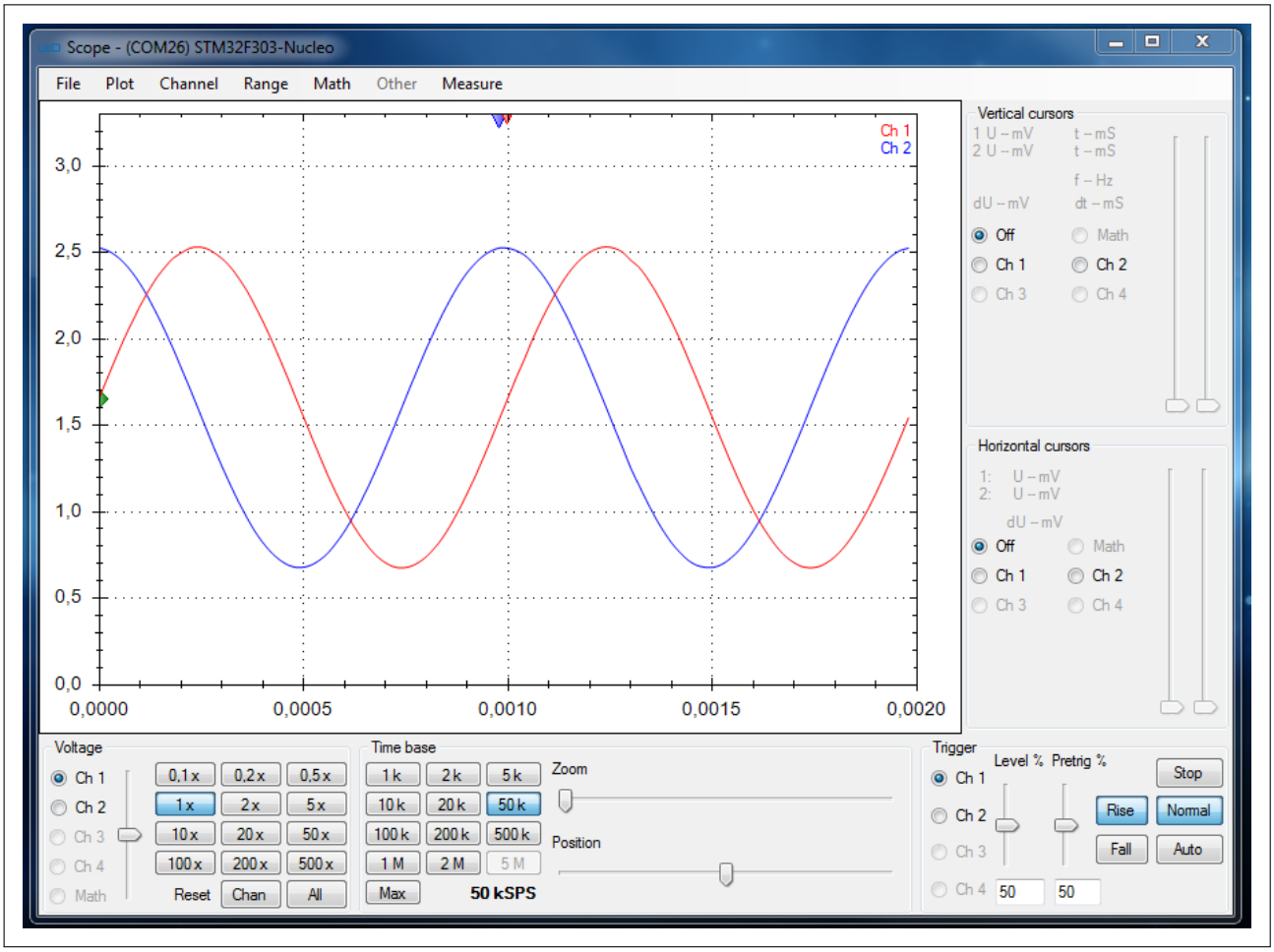

Figure 4.1: Oscilloscope GUI in PC application used by Little Embedded Oscilloscope (LEO)

A terminal (text-based) user interface (TUI) is a simpler alternative to a GUI. Traditionally, these were used by computer terminals before the advent of GUIs, however they are still commonly used in terminal emulators, predominantly on Unix-like systems. A terminal user interface consists primarily of text without advanced graphical features. However, when supported, ANSI escape sequences can be used to format the displayed text, e.g. changing its position, foreground and background colors. These sequences were first standardized in ECMA-48 [14], then incorporated into the ANSI X3.64 standard [15]. Using these, certain parts of the interface can be emphasized and distinct graphical elements such as buttons, panels and toolbars can be created. Figure 4.2 shows an example of a TUI created using ANSI escape sequences.

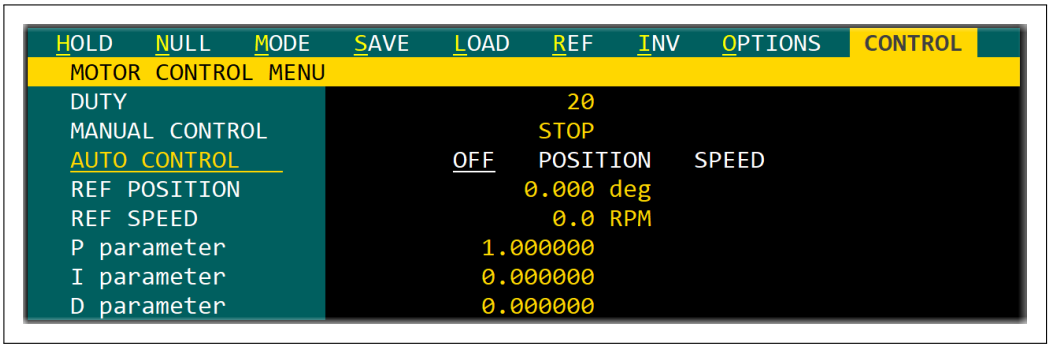

Figure 4.2: Motor controller TUI created using ANSI escape sequences. Adapted from [16, 42]

While a terminal user interface is not generally as easy to use or as visually appealing as a GUI, it is much simpler to implement and lends itself well to being incorporated into the STM32 MCU firmware itself. That way, all changes due to differing MCU capabilities and available features are constrained to the MCU firmware, with no need to modify a PC application. Moreover, a universal terminal emulator application can then be used to display the TUI. This approach is therefore very flexible when it comes to adding new features or support for new microcontrollers which is why it was used in the VSVI platform implemented in this work.

Some measurements such as oscilloscope waveforms cannot be displayed in a terminal though. The data can be transferred to the PC but a separate application is needed to view it. Ideally, such an application would also incorporate a terminal emulator in which the TUI could be displayed directly. This application would then have the advantage of being universal in terms of the user interface and easy to support by MCU firmware if a simple data format for oscilloscope waveforms is used. The Data Plotter application [12] shown in Figure 4.3 fulfills these requirements and was therefore chosen to be used with the VSVI platform. This application was developed for this exact use case as part of the bachelor's thesis "The Universal GUI for PC Based Oscillographs"[13].

4.1 Possibilities for terminal user interfaces using Data Plotter

Data Plotter provides a universal GUI for oscilloscope-like instruments with a multi-channel waveform display, including analog and digital/logical channels. It also provides a plethora of display options (waveform colors, scaling, zoom, etc.) and math including FFT analysis. Crucially, it also incorporates a terminal emulator window in the sidebar on the right side of the window. The terminal supports ANSI escape sequences, allowing the creation of various graphical elements.

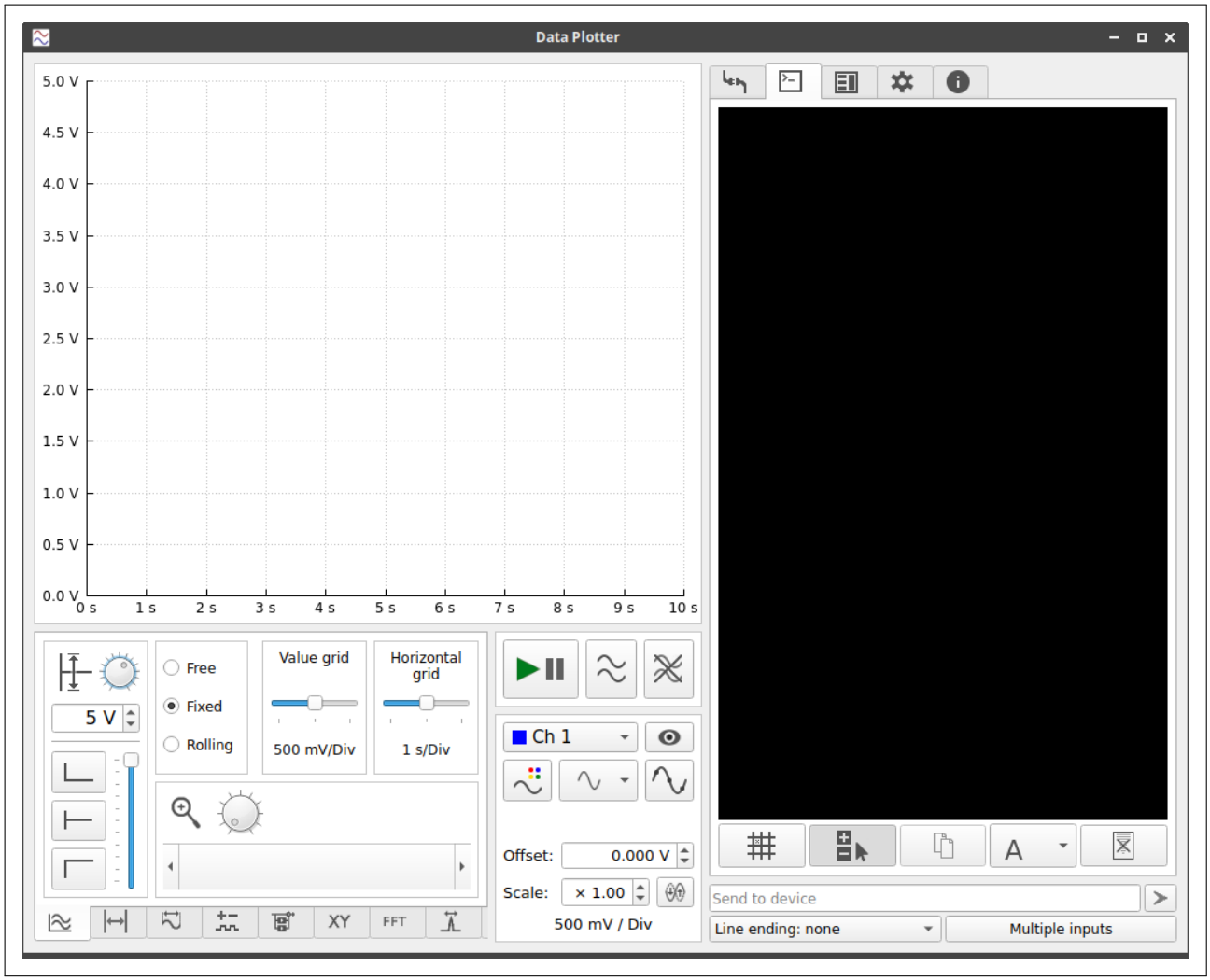

Figure 4.3: Data Plotter PC application with empty terminal window in the sidebar

A novel feature of Data Plotter's terminal is the support for clickable buttons within the displayed terminal interfaces. This is achieved by Data Plotter reading the character displayed in the terminal at the position of the mouse cursor, then sending this character to the MCU as a response to a mouse click. This allows the implemented terminal user interface to be controlled in a manner very similar to a GUI for a user-friendly experience. However, each command can then only be a single ASCII character. This of course limits the maximum number of commands and the amount of information contained within each.

Additionally, every occurrence of a command character is equivalent, regardless of whether it occurs within a button or not – potentially sending unwanted commands to be sent. To alleviate this issue, Data Plotter supports whitelisting or blacklisting of background colors – a character can be made clickable only when displayed with certain background colors. These terminal options can be set within the Data Plotter GUI shown in Figure 4.4 or by commands from the MCU. This way, every clickable button can be rendered in a color not occurring elsewhere in the TUI to ensure no other characters remain clickable. Additionally, for a cleaner look, the unique characters within each button can be hidden by setting their foreground color to be the same as the background – leaving a blank button.

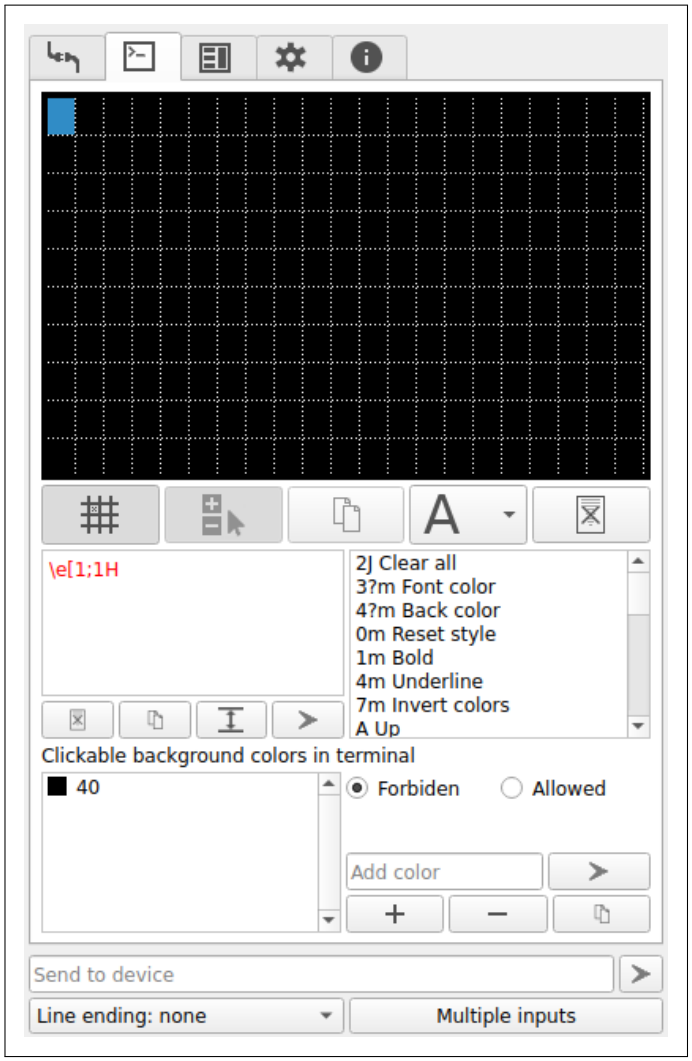

Figure 4.4: Data Plotter terminal options

Data Plotter also allows the user to send text commands to the MCU directly via the "Send to device" text field underneath the terminal window, as seen on the bottom of Figure 4.4. The line endings can be configured while the text field can be pre-filled by a command from the connected microcontroller. This complements the single-character commands sent from the clickable buttons described above by allowing the user to enter numeric values, for example. All of these Data Plotter features are used by the terminal user interface implemented in the VSVI firmware, as detailed in the following sections.

4.2 Developed terminal user interface

The terminal user interface developed in this work is shown in Fig. 4.5. It is optimized for use with Data Plotter, relying on clickable buttons for all navigation and commands. The VSVI platform offers a variety of instruments with numerous parameters and settings, all of which could not be displayed at once in the relatively small Data Plotter terminal window. Therefore, the user interface has been segmented into tabs, one for each type of virtual instrument. This way, all the parameters can be logically grouped together and displayed with sufficient clarity.

Each of the TUI tabs displays only the currently relevant instrument parameters, adjusting dynamically to the instrument configuration. The name of the currently opened tab is displayed at the very top of the interface. Switching between tabs is achieved via a drop-down menu, as shown in Figure 4.5. This menu is opened when the user clicks the blue button located to the left of the tab name. This approach is also used to select certain setting values, as detailed in section 4.2.2.

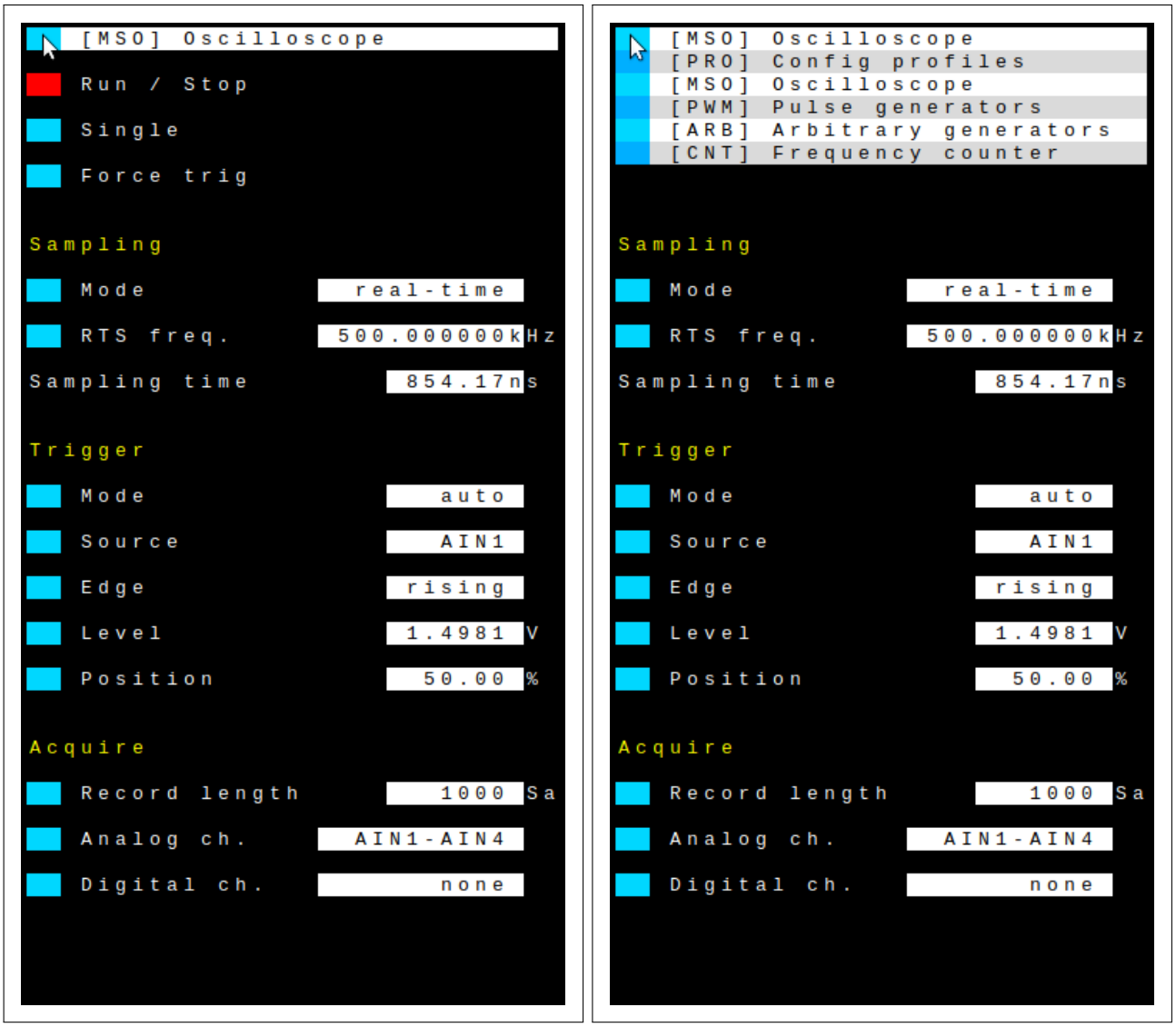

Figure 4.5: Implemented terminal user interface, showing the oscilloscope tab. Tab selection drop-down menu opened on the right.

Within each tab, a number of instrument settings/parameters are shown. Their current values are highlighted in white blocks, with units to their right, if applicable. The width of the value block is adjusted for the precision with which the value is displayed. If the value of an instrument parameter can be changed by the user, a clickable "edit" button is shown to the left of the parameter name. This button is typically blue, with the exception of oscilloscope Run/Single/Stop settings. The behavior of the TUI when this button is clicked depends on the type of the parameter being changed, as described in the following sections 4.2.1 through 4.2.4.

4.2.1 TUI toggle settings

For settings with only a few discrete options, e.g. "on"/"off", repeatedly clicking the edit button cycles through all the available options. The new value is applied immediately, there is no confirmation necessary. This input method is implemented by each TUI tab directly.

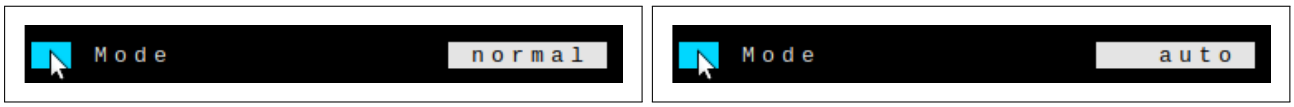

Figure 4.6: Toggle setting in terminal user interface. The oscilloscope trigger mode setting is shown.

4.2.2 TUI drop-down menu

The "toggle setting" input method described above is very simple to implement but it becomes tedious when there are more than a few options to cycle through. For this reason, a drop-down menu was implemented for settings with a larger number of discrete options. This menu is shown under the parameter value block and lists all the available options, each with a button on the left side. Clicking one of these buttons selects the corresponding option and applies it immediately. The drop-down menu is also closed at the same time, no confirmation is necessary. It is also closed if the user click the edit button of any parameter, leaving the value of the parameter currently being adjusted unchanged.

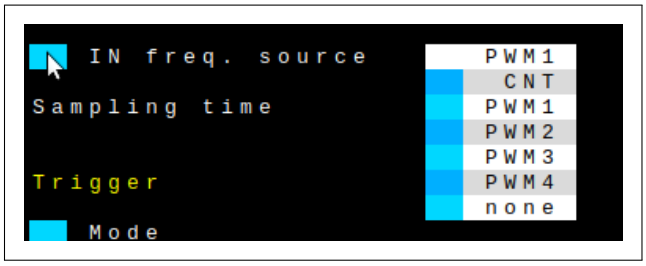

Figure 4.7: Drop-down menu in terminal user interface. The oscilloscope input frequency source setting is shown.

This input method is implemented centrally for all TUI tabs. When the user selects a setting for which the drop-down menu is implemented, the active tab generates the list of menu options. A centralized TUI function (used by all tabs) then renders the drop-down menu according to this list, automatically assigning a command character to each (written into the button used to select that option). This function also handles all the formatting such as matching the width of the drop-down menu to the width of the parameter value block or selecting between displaying one or two columns of options based on their width. When the button associated with one of the menu options is clicked, the received command character is parsed by another centralized TUI function. This function matches the command to one of the options, which it then returns as a string to the active TUI tab for processing. This method simplifies the implementation of each individual TUI tab and prevents firmware redundancy.

4.2.3 TUI numeric value editor

A digit-wise editor displayed in a row under the value block is provided for numeric values. The "<" and ">" buttons select a digit in the value. The "+" and "-" buttons then increment/decrement the selected digit. The rightmost digit position is a special case, as it contains the metric unit prefix character (as defined in Table 4.1). Incrementing/decrementing at this position cycles through the available prefixes. The new value is saved when the user clicks any parameter edit button. There may also be a number of presets available below, which can be used in the same way as the drop-down menu described above.

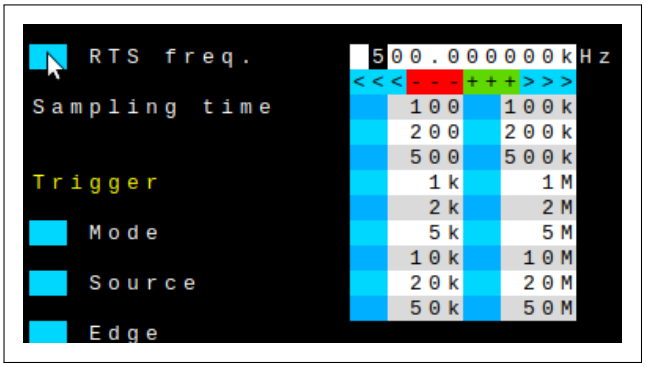

Figure 4.8: Numeric editor in terminal user interface. A number of presets are available. The oscilloscope real-time sampling frequency parameter is shown.

Similarly to the case of the drop-down menu, the numeric value editor is also implemented in centralized TUI functions. To render the numeric editor, the active tab only passes the current parameter value as a string to one of the centralized functions. This value is copied into a centralized string buffer. All subsequent user interaction such as highlighting or changing the selected digit is then also handled centrally. Finally, when the user finishes the editing process by clicking one of the parameter edit buttons, the resulting value is passed as a string back to the active TUI tab for processing. If there are presets available, they're handled the same way as options in the drop down menu, also returning the selected value as a string. The tab can then parse this string either as an integer or floating point number, depending on the type of parameter.

Prefix	nano	micro	mili	-	kilo	mega	giga
Character	"n"	"u"	"m"	" "	"k"	"M"	"G"
Multiplier	10^{-9}	10^{-6}	10^{-3}	1	10^{3}	10^{6}	10^{9}

Table 4.1: Supported metric unit prefixes

The parameter value entered by the user often cannot be achieved exactly. The firmware then sets the closest possible value, which is subsequently shown in the TUI. However, the originally selected value is saved and the firmware will attempt to achieve a closer match if the instrument configuration is changed later.

4.2.4 TUI direct value entry using Data Plotter

While the value input options described so far are certainly usable, they are not quite as convenient as being able to directly type the value on the keyboard. Therefore, another method was also implemented, using the "Send to device" function of Data Plotter. Located below the terminal window, this text field is pre-filled with the current parameter value when the value editor (drop-down or numeric) is opened. The user may then type the desired value in this field directly and apply it by clicking the send button.

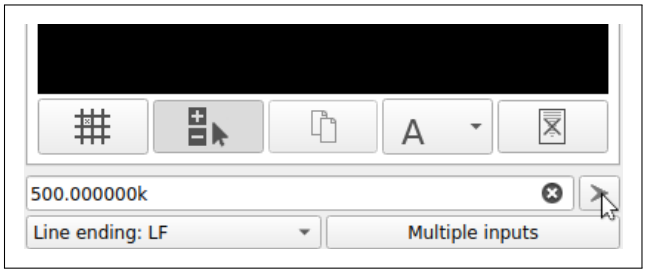

Figure 4.9: Direct TUI parameter value entry using Data Plotter. The text field is pre-filled with the current parameter value.

The text received by the MCU when using this option is parsed by the same centralized function that parses all other TUI commands, distinguished by the end of line character. When a parameter value is received in this way, it is passed directly to the active TUI tab as a string. The tab then processes it the same way as if it came from the drop-down menu or numeric value editor.

Chapter 5

Mixed-signal oscilloscope

Oscilloscopes are among the most important types of test equipment for electronics development or for teaching electronics. They can be used to analyze the signals at various points of a circuit under test, allowing the user to verify its functionality or find any potential issues. Unfortunately, affordable low-end oscilloscopes may only offer two channels or lack other important features. Moreover, the proportion of digital circuits in electronics is ever-increasing, and with that comes the need to analyze many digital signals at once, for example data buses and interfaces such as SPI or I2C. This often exceeds the 4 channels found on standard oscilloscopes.

However, in most cases, there is no need to visualize the actual waveforms of digital signals, only to differentiate between high and low logical levels. This can be achieved by logic analyzers which are typically more affordable and offer significantly more channels than oscilloscopes, e.g. 8 or 16. The two types of instruments are often combined into a mixed-signal oscilloscope (MSO). An example of a commercially available MSO with 4 analog and 8 digital channels is shown in Figure 5.1. This type of instrument has the advantage of synchronized sampling of all analog and digital channels, allowing the user to time-correlate events happening in the digital and analog parts of the tested circuit. However, these instruments are often more than double the price of a comparable standard oscilloscope.

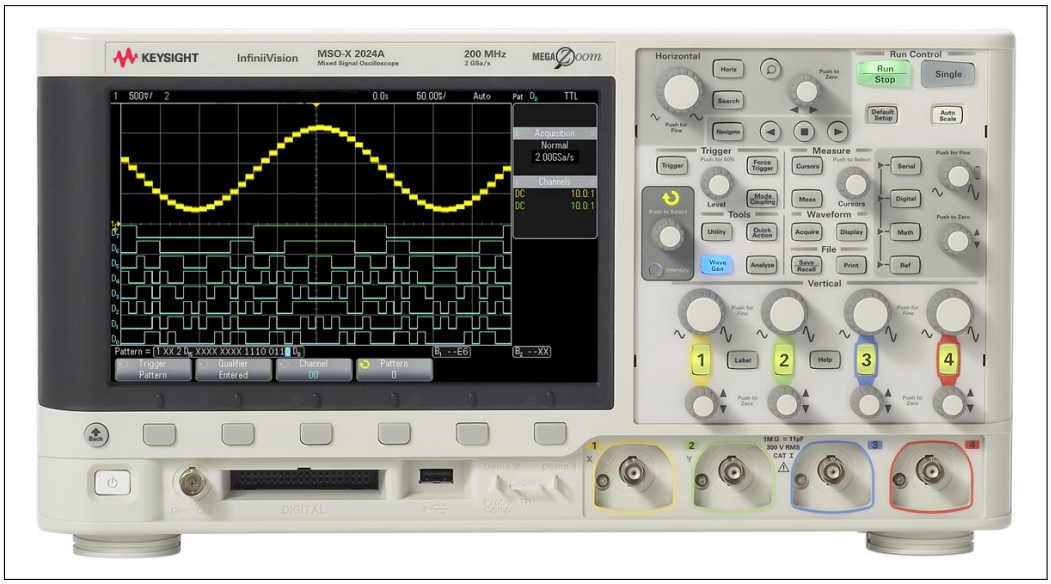

Figure 5.1: Commercially available mixed-signal oscilloscope – Keysight MSOX2014A. Adapted from [20].

The main goal of this work was to implement low-cost, portable alternatives to these instruments in the form of a software-defined oscilloscope and logic analyzer implemented for various STM32

MCUs. Upon consideration, the decision was made to combine the two instruments and create a software-defined mixed-signal oscilloscope with synchronous sampling of 4 analog channels and up to 16 digital channels. This is a novel approach compared to the SDI platforms described in section 2.2, in that these platforms always implemented oscilloscopes and logic analyzers separately, with no means to trigger each other or time-correlate acquired waveforms. Additionally, the MSO instrument implemented in this work adds support for equivalent-time sampling, dramatically increasing the effective sampling rates achievable for periodic input signals. As can be seen in Figure 5.2, the oscilloscope relies on the aforementioned Data Plotter PC application to display the acquired waveforms.

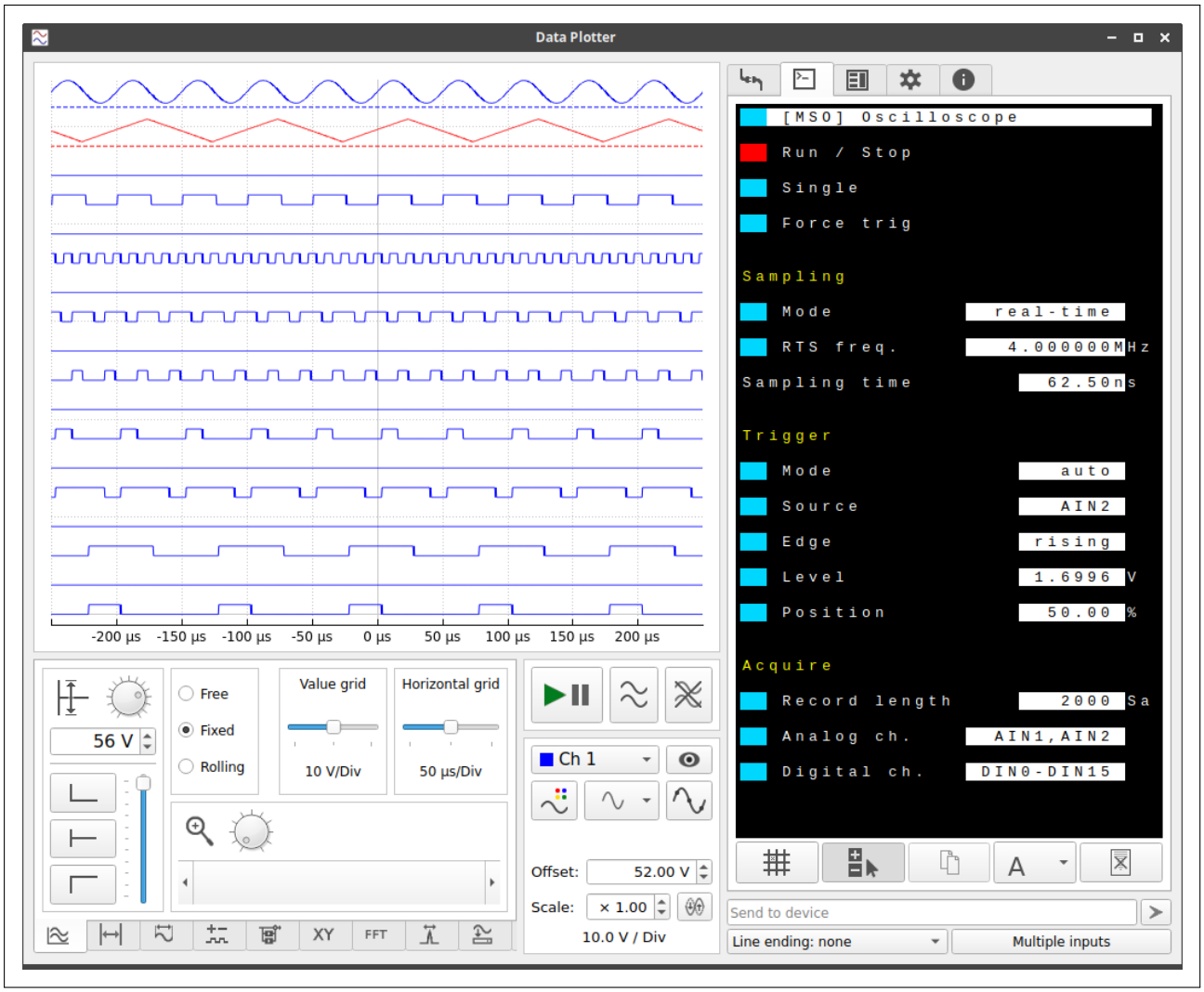

Figure 5.2: Implemented software-defined mixed-signal oscilloscope used within Data Plotter

The block diagram of the implemented MSO, including the STM32 hardware resources used is shown in Figure 5.3. The analog channels are implemented using the analog-to-digital converters embedded in the STM32 MCUs. These offer 12-bit resolution with max. sampling rates between 1 MSps and 5 MSps depending on the MCU (see section 2.4). The timing of their sampling is tightly controlled by triggering the ADC conversions from the sampling timer. After each ADC conversion, the sample data is transferred by a DMA controller from the ADC data register to a circular waveform buffer within the embedded SRAM memory.

The digital channels are essentially just digital GPIO pins in input mode. The current state (logical low or high) of each of up to 16 pins of a given GPIO port (e.g. GPIOA, GPIOB) can be read from the GPIO input data register (IDR). As such, up to 16 digital channels can be sampled by a single register read operation if all of them are assigned to pins of the same GPIO port. The sampling of digital channels is performed directly by a DMA controller, transferring the value of the GPIO IDR register to a circular waveform buffer in SRAM. The request for this transfer comes from one of the capture/compare channels of the sampling timer. This guarantees precise sampling rate control and synchronicity with analog channels sampled by ADCs. However, as ADC conversions are significantly slower than DMA transfers, the digital channel sampling rates can be greatly increased when all analog channels are disabled.

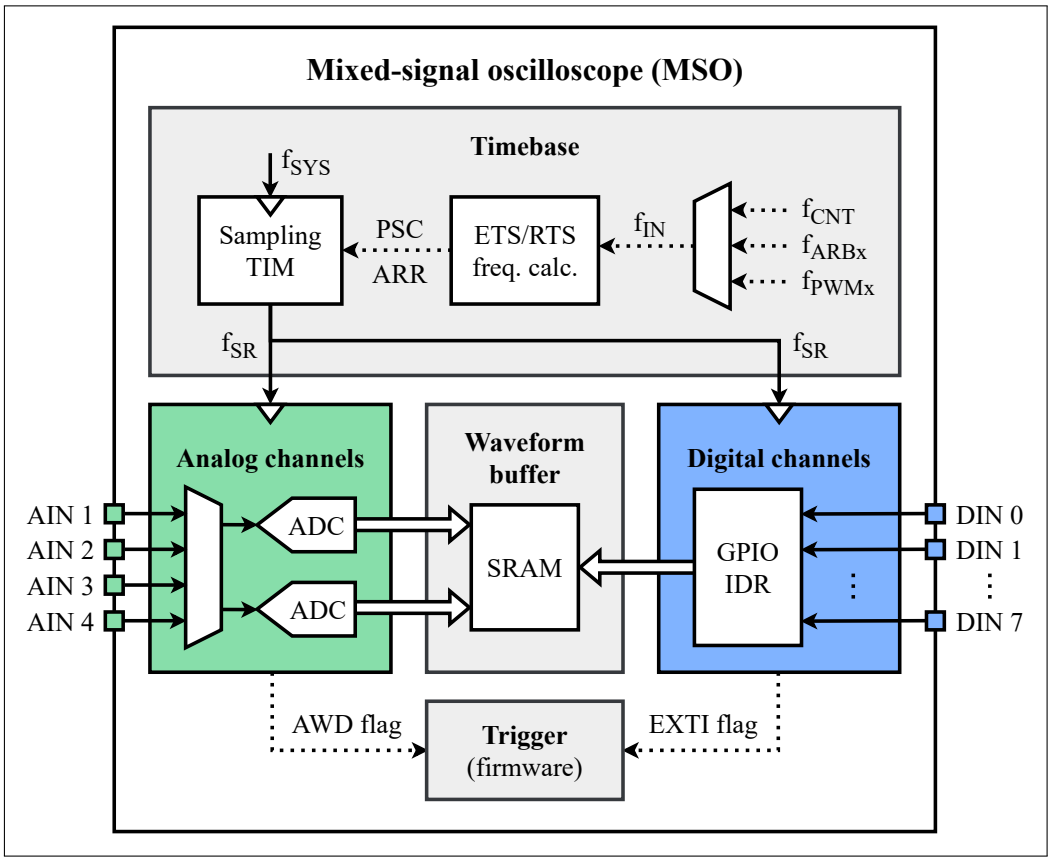

Figure 5.3: Block diagram of implemented mixed-signal oscilloscope

Standard "Stop/Run/Single" oscilloscope modes are used, accompanied by a "Force trigger" option. The oscilloscope trigger is implemented by firmware within the acquisition loop illustrated in Figure 5.5. The oscilloscope can be triggered when a rising or falling edge is detected within the input signal. Auto trigger is also supported, forcing the MSO to trigger when more than 5 record lengths have elapsed with no normal trigger event detected, i.e. when

$$N_A \geq 5 \cdot N_R, \tag{5.1}$$

where N_A is the number of acquired samples and N_R is the MSO record length.

The analog watchdog (AWD) feature of the STM32 analog-to-digital converters is used for triggering from an analog channel. The analog watchdog sets the AWD flag in the ADC status register and optionally issues an interrupt if the value of the last ADC sample is outside the watchdog "guarded" area [31, p. 653]. However, this feature cannot actually detect an edge in a signal, only indicating when the sample value is above or below a given threshold. Because of this, a two-stage approach is needed, as illustrated in Figure 5.4. For example, when triggering on a rising edge, the guarded area is first set to the area above the trigger level. Then, when the first sample goes below the trigger level, the first AWD flag is set. The guarded area is then changed to be the area below the trigger level. The trigger event occurs when the second AWD flag is set, as the input signal had by that point necessarily made a transition from below to above the trigger threshold.

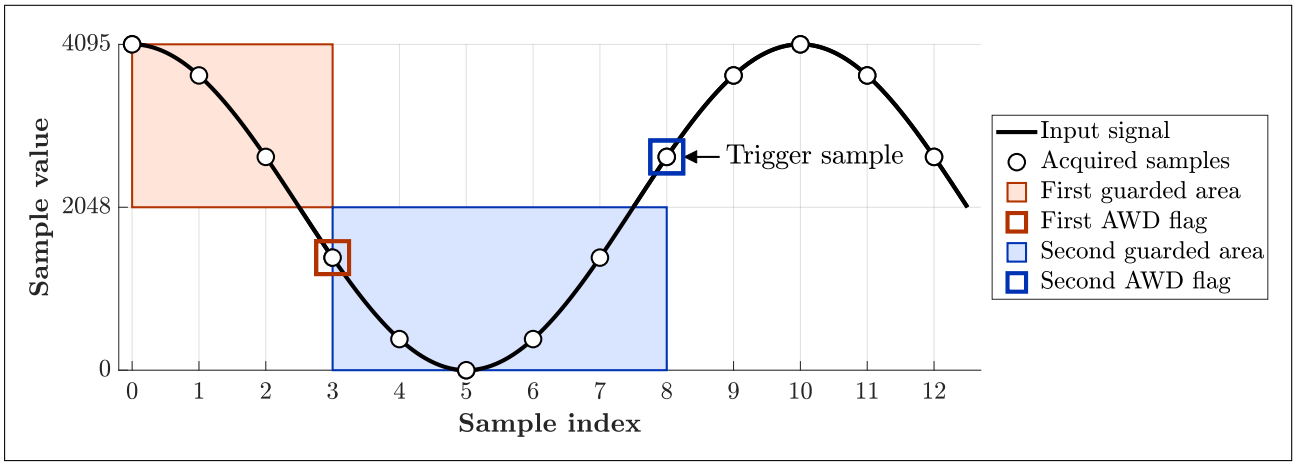

Figure 5.4: Use of the analog watchdog ADC feature for oscilloscope trigger. A rising edge is detected.

When triggering from a digital channel, the extended interrupts and events controller (EXTI) is used to directly detect rising or falling edges in the input signal. This is achieved by a pair of hardware edge detectors feeding into the EXTI controller with configurable edge polarity. Whether the edge detection is active on any given GPIO pin can also be configured, allowing the trigger to occur from any digital channel selectively. Typically, an interrupt would be generated when an event is detected. However, interrupts introduce additional delays before the interrupt handler function code is executed, on the order of tens of CPU cycles [19]. Therefore, the implemented acquisition loop (see Figure 5.5) reads the EXTI interrupt pending register (PR) instead. A logical high value in this register indicates an input signal edge has been detected since the last time the firmware cleared this register.

Since the trigger is only firmware-based, there is some inherent delay between the trigger event and the firmware loop acknowledging it. This issue is particularly exacerbated if an interrupt occurs around the same time as the trigger event. This causes the recorded trigger time (sample index) to be delayed from the actual trigger time. To compensate for this issue, the following steps are taken:

- An extra $N_E = 16$ samples are added to each acquisition beyond the user-selected record length.
- At the end of each acquisition, the firmware checks the waveform buffer at the recorded trigger sample index to determine if the selected trigger edge occurred at that point.
- If no edge is found, the firmware steps backwards through the previous samples to determine the actual position of the trigger.
- In case no edge is found within the extra samples, the contents of the waveform buffer are discarded and a new acquisition is started.

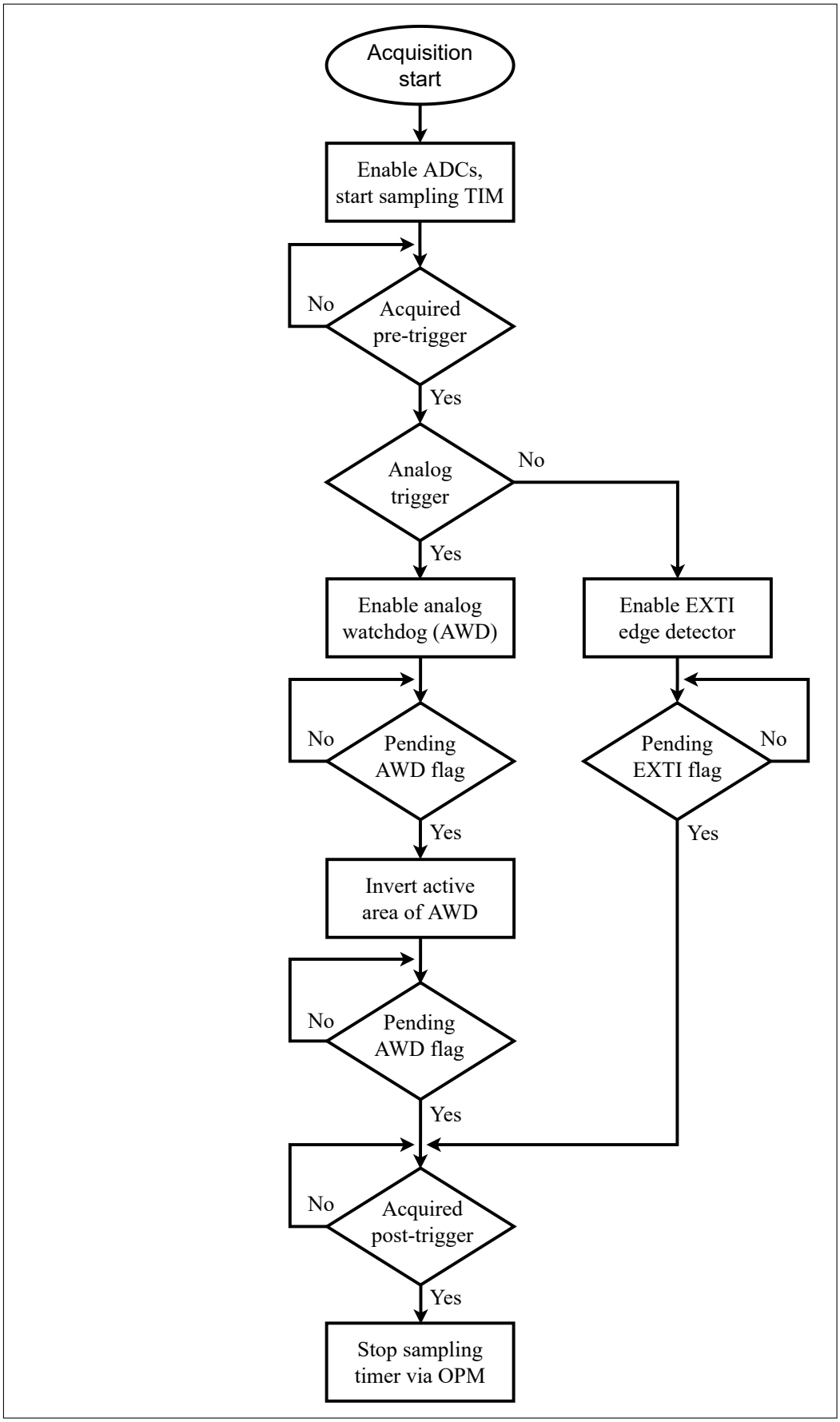

Figure 5.5: Mixed-signal oscilloscope firmware acquisition loop

5.1 Real-time sampling

Typically, oscilloscopes use real-time sampling (RTS) of their input channels. Using RTS, samples are captured at the real-time sampling frequency f_{SR}, i.e. with the real-time sampling period of T_{SR} between subsequent samples. In accordance with the well-known Nyquist theorem, this sampling mode only captures the input signal correctly if the highest frequency $f_{IN,max}$ in the signal is lower than the Nyquist frequency, i.e.

$$f_{IN,max} < f_{NYQ} \tag{5.2}$$

$$f_{IN,max} < \frac{f_{SR}}{2}. \tag{5.3}$$

When this condition is not satisfied, aliasing occurs. In other words, the oscilloscope must capture more than 2 samples per each period of the input signal. This is only an absolute minimum value, however – the record length would need to be very large to accurately represent the signal by averaging many acquired periods. A more practical lower bound is approximately 10 samples per period. The following figures 5.6 through 5.8 illustrate real-time sampling at various sampling rates (samples per period). In figure 5.6, the input signal frequency is equal to the Nyquist frequency, causing the reconstructed signal to be a constant zero – this case demonstrates why the Nyquist theorem requires the input frequency to be *strictly* lower than the Nyquist frequency.

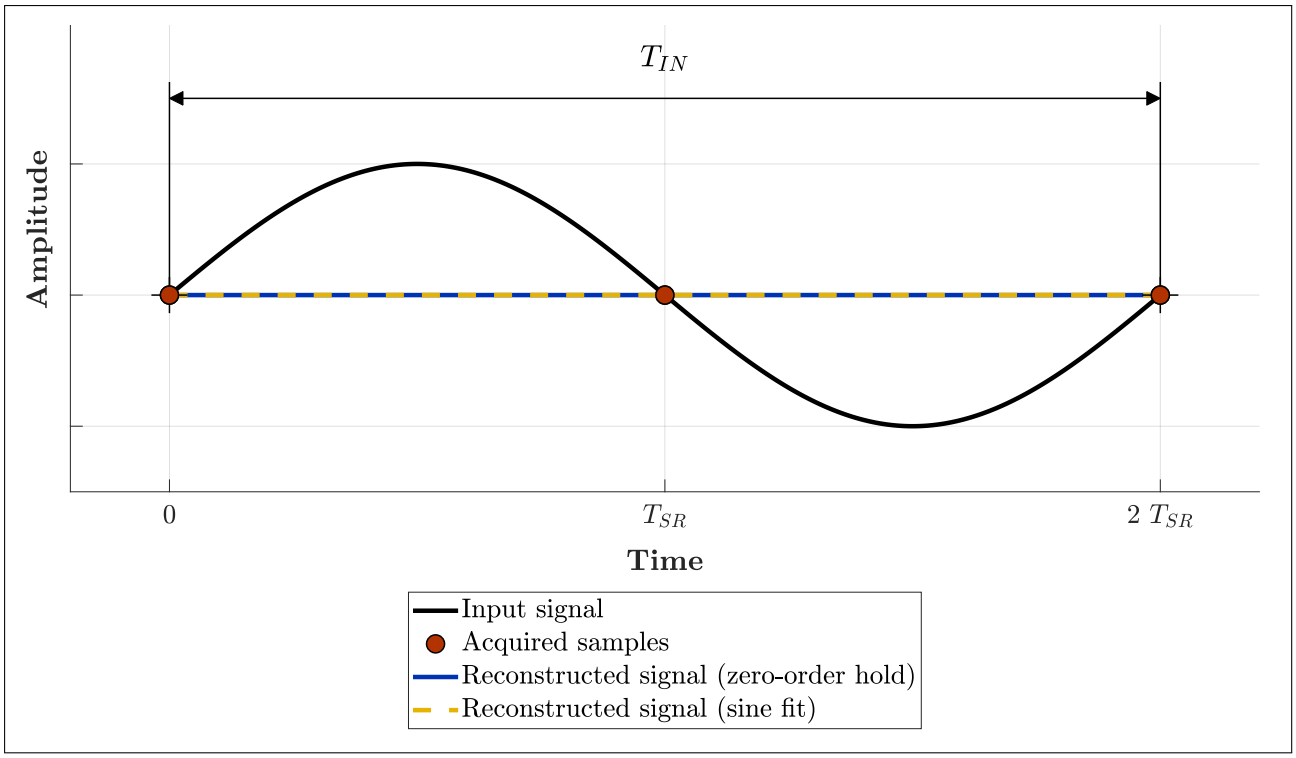

Figure 5.6: Real-time sampling at 2 samples per period (Nyquist limit). Real-time sampling period is $T_{SR} = 1/2 \cdot T_{IN}$. Acquired samples are all zero, producing an incorrect signal reconstruction.

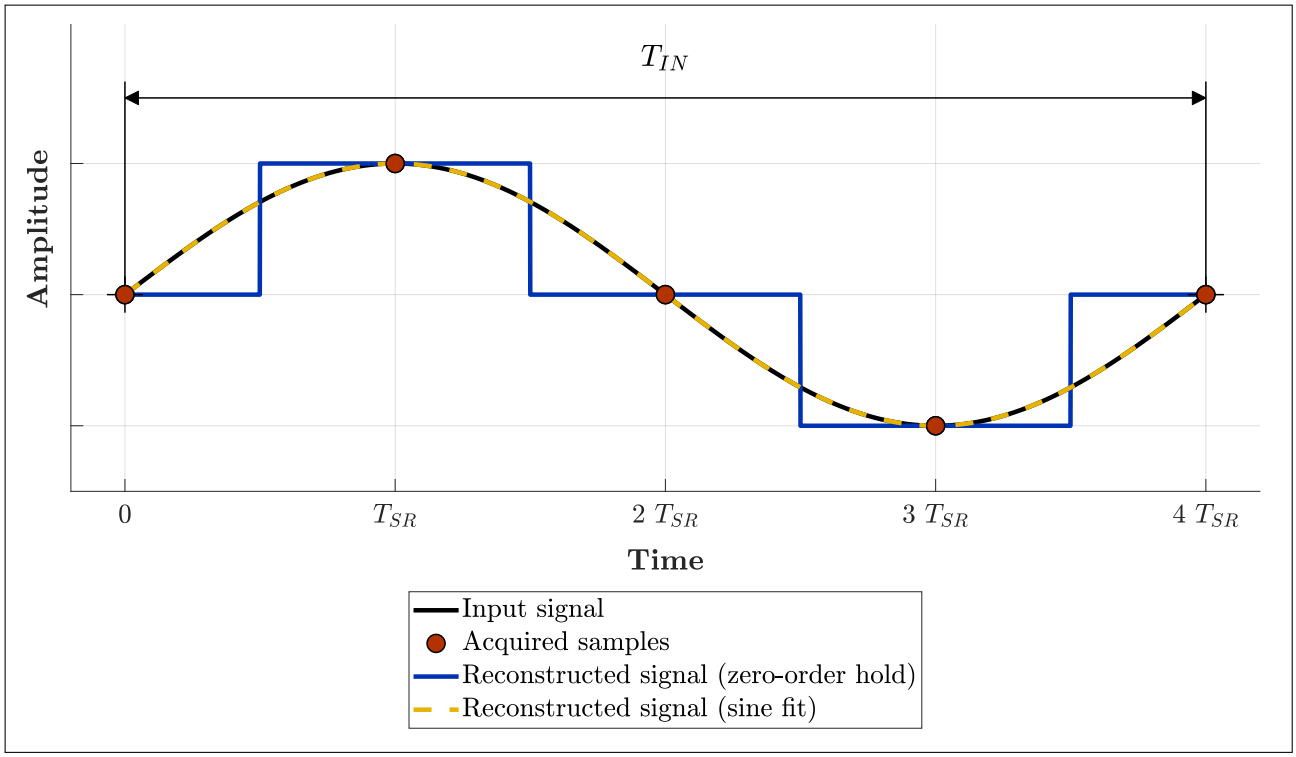

Figure 5.7: Real-time sampling at 4 samples per period. Real-time sampling period is $T_{SR} = 1/4 \cdot T_{IN}$. Reconstructed 1 signal period from acquired samples.

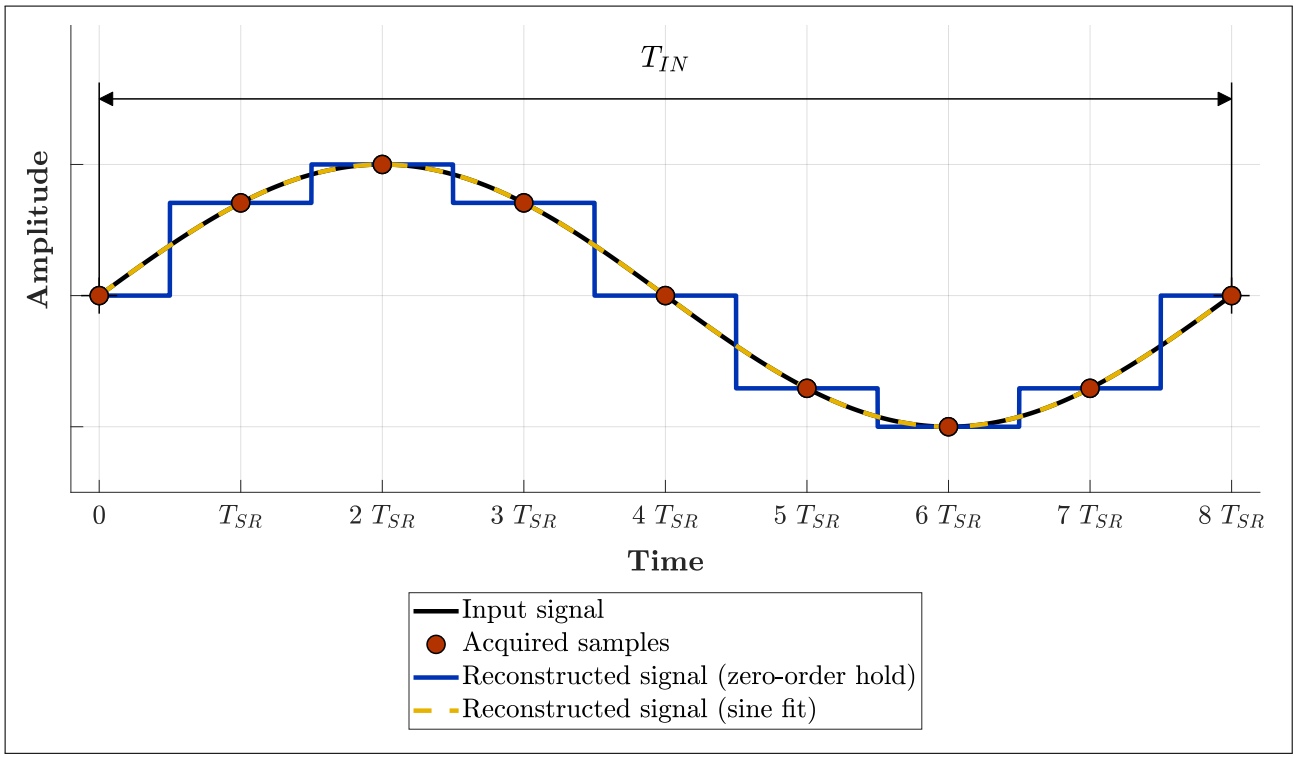

Figure 5.8: Real-time sampling at 8 samples per period. Real-time sampling period is $T_{SR} = 1/8 \cdot T_{IN}$. Reconstructed 1 signal period from acquired samples.

However, the real-time sampling mode may not be suitable in all cases, for example with a high-impedance input signal source. This source must charge the ADC sampling capacitor within the sampling time t_{LATR} which is rather short (tens of nanoseconds) at high sampling rates. The sampling capacitor charging rate is limited by the source impedance R_{AIN}. Effectively, a low-pass RC filter is formed, as can be seen in Figure 5.10. This significantly lowers the usable analog bandwidth of the oscilloscope. To resolve this issue, a buffer amplifier can be used to present a low-impedance source to the ADC as shown in Figure 5.11. The firmware always maximizes the sampling time (for a given sampling rate) and displays it in the TUI alongside other MSO parameters, as seen in Figure 5.2.

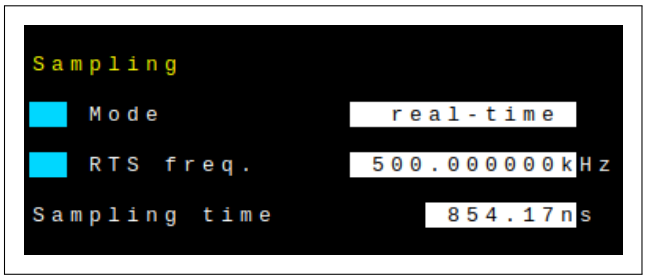

Figure 5.9: Sampling settings in TUI, MSO in real-time sampling mode

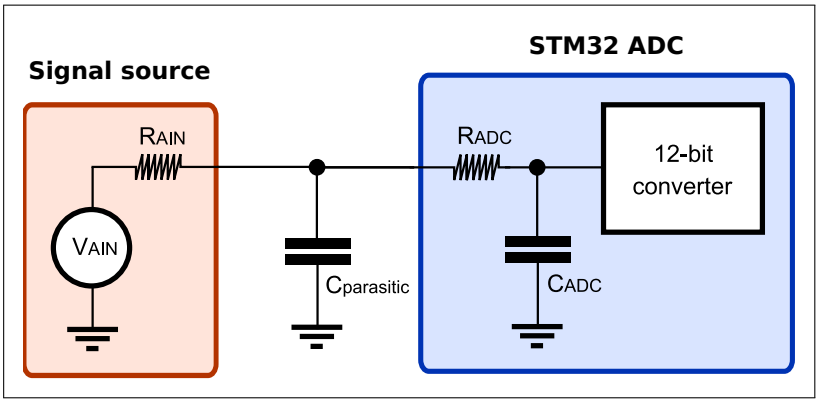

Figure 5.10: MSO input signal connected directly to the STM32 ADC. Adapted from [23, p. 144].

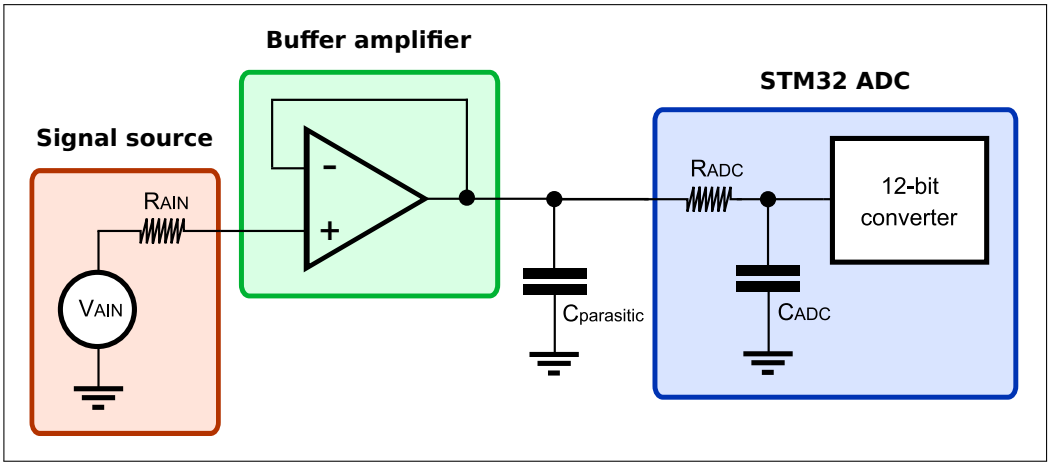

Figure 5.11: Use of amplifier to buffer high-impedance MSO input signal. Adapted from [23, p. 144].

5.2 Equivalent-time sampling

While aliasing is generally an undesirable effect, it can also be utilized to sample periodic signals in equivalent-time sampling (ETS) mode. Assuming every period of the input signal is equivalent, samples taken from multiple periods can be combined. Traditionally, this is done by randomly timed sampling, where the time elapsed since the trigger event is recorded alongside each sample. The waveform can then be reconstructed by placing each sample at the correct time [17]. This method is visualized in Figure 5.12;

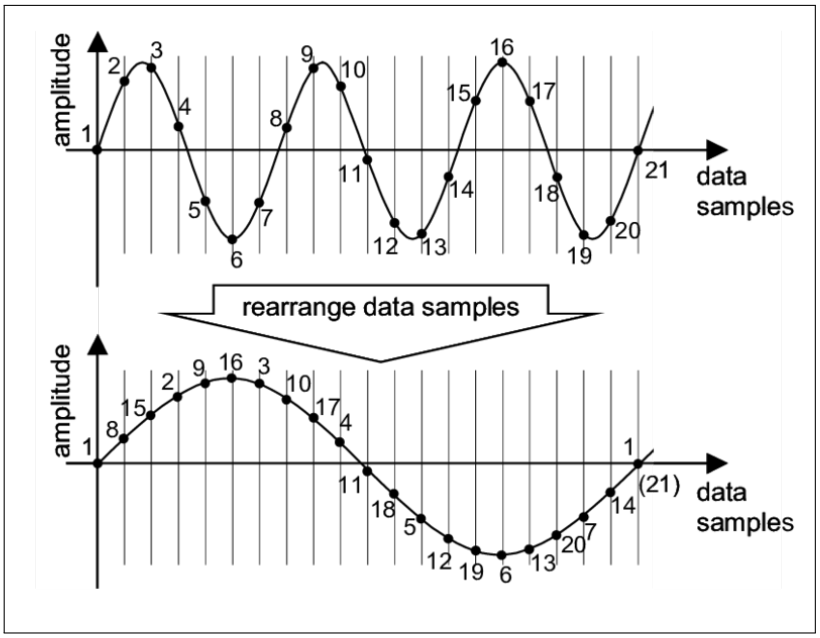

Figure 5.12: Usual method of equivalent-time sampling. Adapted from [18].

This traditionally used method requires significantly more memory and post-processing, which would be a rather slow process on the MCUs used. For these reasons, the alternative "stroboscopic" equivalent-time sampling method is used. This alternative method is demonstrated in Figures 5.13 through 5.16. It involves setting the real-time sampling period T_{SR} to be slightly longer than N_P input signal periods T_{IN}. Then, the equivalent-time sampling rate f_{SE} can be calculated according to

$$T_{SR} = N_P \cdot T_{IN} + T_{SE} \iff f_{SR} = \frac{f_{IN} \cdot f_{SE}}{f_{IN} + N_P \cdot f_{SE}} \tag{5.4}$$

$$T_{SE} = T_{SR} - N_P \cdot T_{IN} \iff f_{SE} = \frac{f_{IN} \cdot f_{SR}}{f_{IN} - N_P \cdot f_{SR}}. \tag{5.5}$$

Using this method, periodic signals can be sampled at significantly higher equivalent rates than using real-time sampling. The limitation is no longer the maximum sampling rate of the ADCs, but the frequency resolution of the sampling timer. When $N_P = 1$, the max. ETS frequency is equal to the timer clock frequency (up to 2 orders of magnitude above max. ADC sampling frequency). When more input periods are skipped between samples, $N_P > 1$, the max. ETS frequency is further multiplied as is the time needed to acquire a whole waveform. An additional benefit of this sampling method is that the sampling time can be substantially increased, allowing the use of high-impedance input signal sources. Since the sampling capacitor voltage tracks the input voltage for a longer time, the effect of transients is minimized and the full analog bandwidth of the ADCs can be utilized.

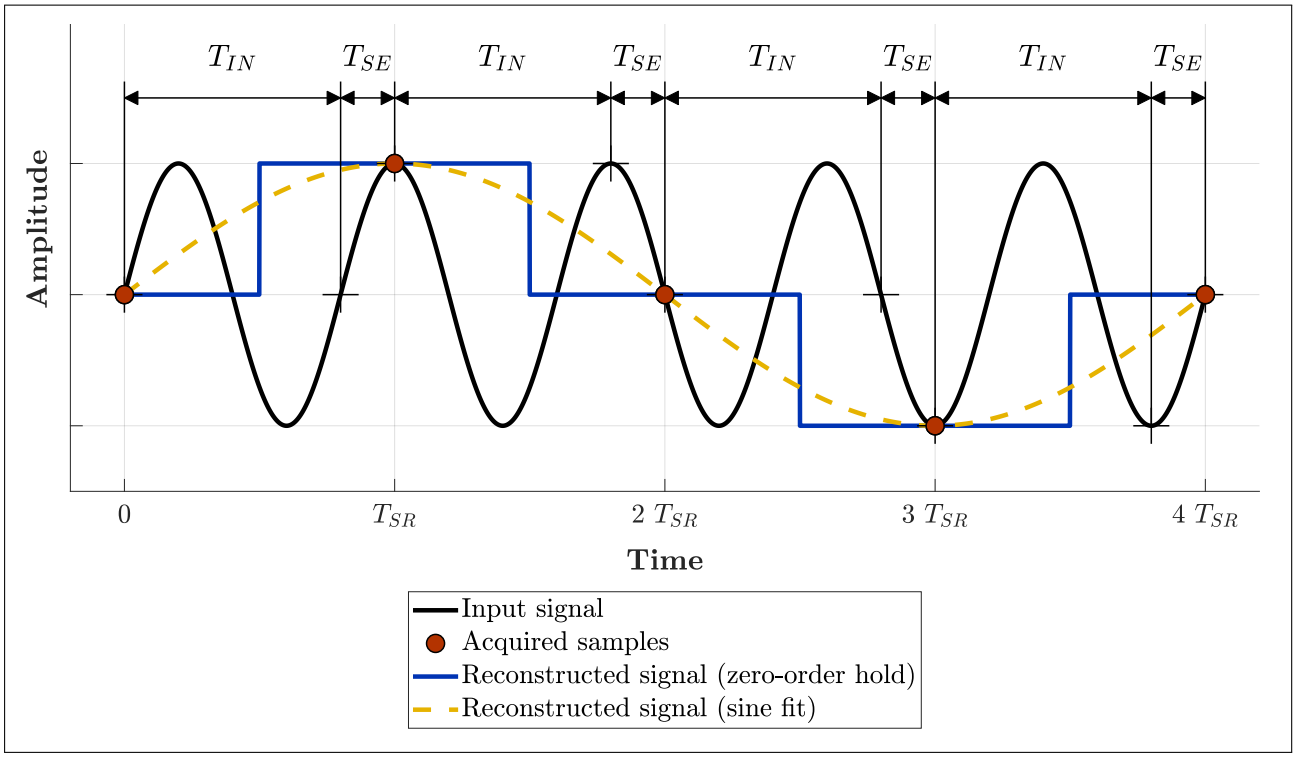

Figure 5.13: Equivalent-time sampling at 4 samples per period with 1 whole period between samples ($N_P = 1$). Equivalent-time sampling period is $T_{SE} = 1/4 \cdot T_{IN}$, real-time sampling period is $T_{SR} = N_P \cdot T_{IN} + T_{SE} = 5/4 \cdot T_{IN}$. Reconstructed 1 period from samples acquired from 5 periods.

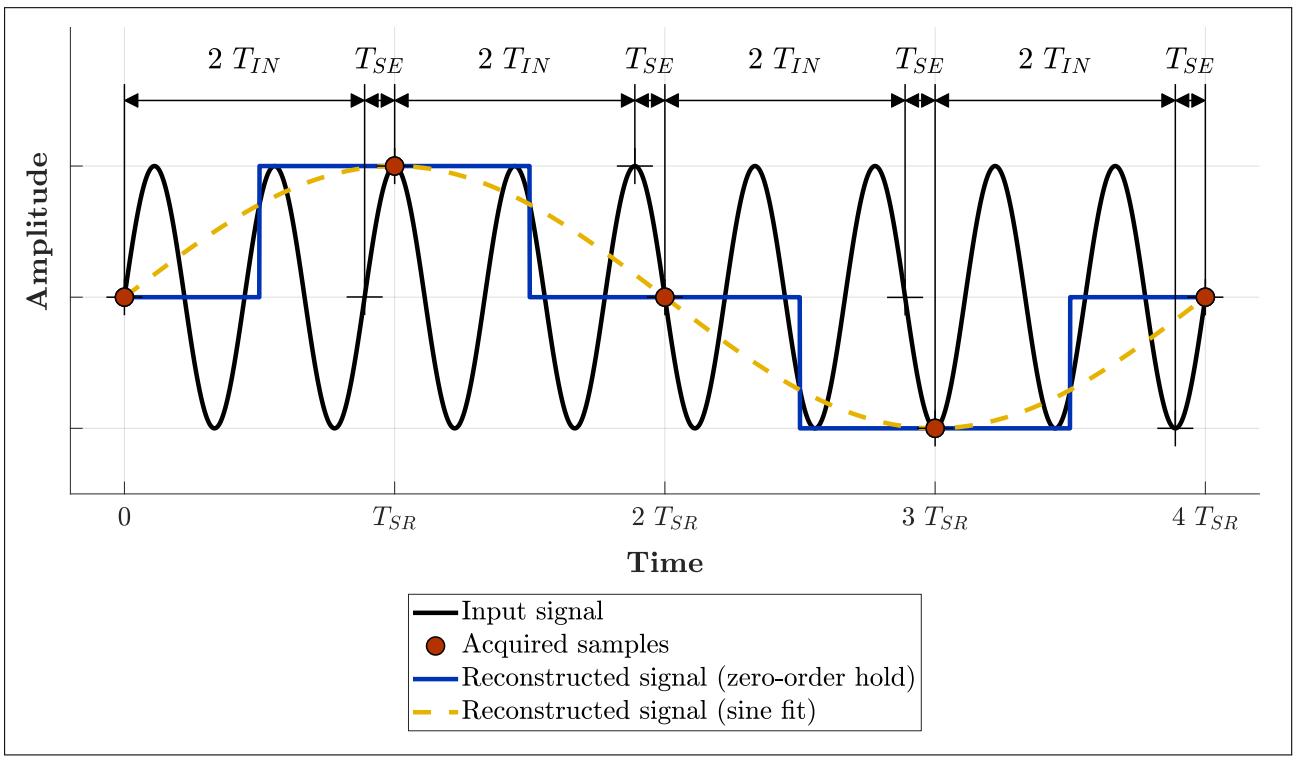

Figure 5.14: Equivalent-time sampling at 4 samples per period with 2 whole periods between samples ($N_P = 2$). Equivalent-time sampling period is $T_{SE} = 1/4 \cdot T_{IN}$, real-time sampling period is $T_{SR} = N_P \cdot T_{IN} + T_{SE} = 9/4 \cdot T_{IN}$. Reconstructed 1 period from samples acquired from 9 periods.

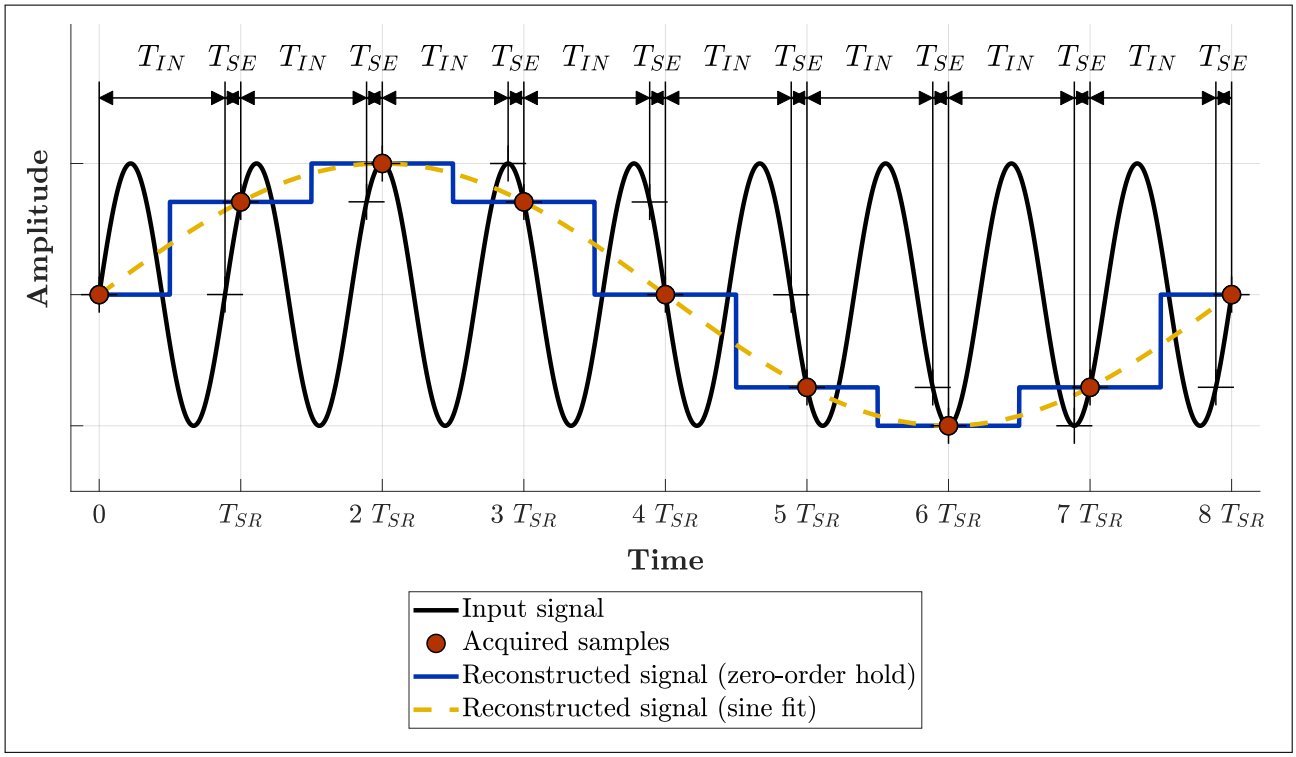

Figure 5.15: Equivalent-time sampling at 8 samples per period with 1 whole period between samples ($N_P = 1$). Equivalent-time sampling period is $T_{SE} = 1/8 \cdot T_{IN}$, real-time sampling period is $T_{SR} = N_P \cdot T_{IN} + T_{SE} = 9/8 \cdot T_{IN}$. Reconstructed 1 period from samples acquired from 9 periods.

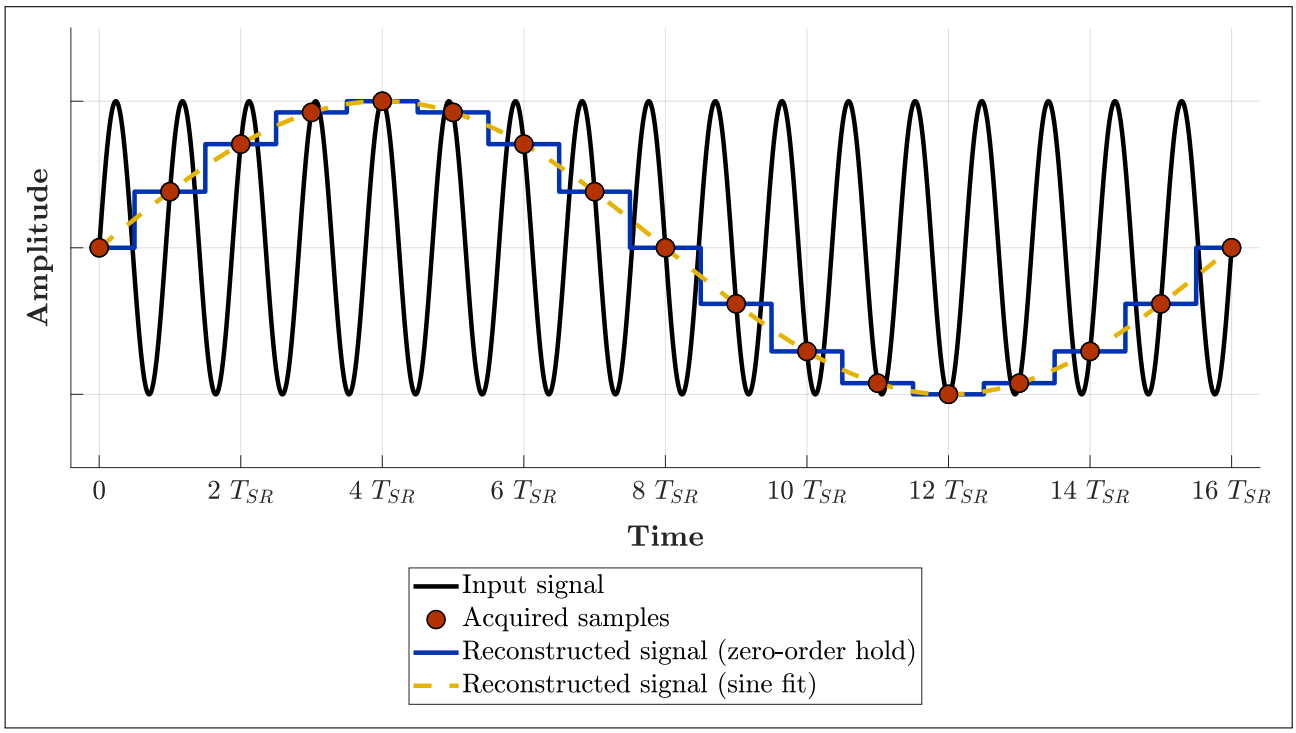

Figure 5.16: Equivalent-time sampling at 16 samples per period with 1 whole period between samples ($N_P = 1$). Equivalent-time sampling period is $T_{SE} = 1/16 \cdot T_{IN}$, real-time sampling period is $T_{SR} = N_P \cdot T_{IN} + T_{SE} = 17/16 \cdot T_{IN}$. Reconstructed 1 period from samples acquired from 17 periods.

The implemented method of equivalent-time sampling relies on the knowledge of the precise fundamental input signal frequency f_{IN} in order to set the real-time sampling frequency f_{SR} so that the desired f_{SE} is achieved. There are three supported ways to obtain the input signal frequency which can be selected within the oscilloscope TUI tab shown in Figure 5.17. The options are:

- **Manual input**: The user may simply enter the value of the input signal frequency if it is known.
- **Self-generated signal**: It is ideal if the situation allows for the MSO input signal being measured to be derived from a signal generated by the MCU itself. This eliminates any effects of frequency drift, as the MSO sampling clock and the generated signal are both derived from the same clock source. The MSO can automatically set its input signal frequency value to the output signal frequency of any pulse or arbitrary generator. This option is particularly useful for performing step response analysis of a circuit, for example comparing slew rates of the pulse and arbitrary generator output signals as shown in Figure 7.7.
- **External signal measured by frequency counter**: In case it is not possible to derive the MSO input signal from the MCU itself, there is also the option to measure the frequency of the input signal. This is done using the implemented frequency counter described in 8. This option was used for all ETS mode testing in chapter 10.

In all cases, the frequency of the input signal must remain constant for the whole duration of the acquisition, otherwise the ETS frequency will exhibit jitter – "stretching" or "squeezing" parts of the acquired waveform as seen in Figure 10.16.

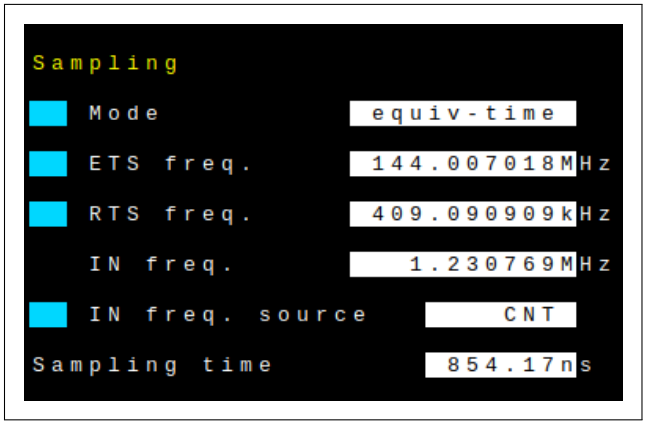

Figure 5.17: Mixed-signal oscilloscope TUI tab (equivalent-time sampling mode)

Within the oscilloscope TUI tab, the user can set either the equivalent-time or real-time sampling frequency. There are two modes of operation associated with these settings:

- The user sets the "ETS freq." parameter. The firmware then finds an RTS frequency such that the resulting ETS frequency is as close as possible to the entered value. If the input frequency changes, the firmware readjusts the RTS frequency to maintain the desired ETS frequency.
- The user sets the "RTS freq." parameter. In that case, the "ETS freq." parameter only shows the calculated ETS frequency, based on the input frequency. If the input frequency changes, the RTS frequency remains unchanged, while the ETS frequency is recalculated.

5.3 Interleaved sampling

When real-time sampling is used, the maximum oscilloscope sampling rate is severely limited by the max. sampling rate of the ADCs. Typically, the minimum ADC sampling period is 14 or 15 ADC clock cycles, while the ADC clock frequency may be reduced compared to the CPU clock. However, if there are multiple ADCs embedded within the MCU, it is possible to use multiple of them to sample the same input signal. Since the sampling of each ADC can be started independently by the capture/compare channels of the sampling timer, a time offset between the ADCs can be set. Interleaved sampling can thus be achieved by multiple ADCs sampling the input signal alternatively at regular intervals – half the sampling timer period for 2 ADCs per channel, quarter for 4 ADCs per channel. The effective MSO sampling rate is then multiplied by the number of ADCs used to sample each channel. Of course, this mode can only be used if the number of active analog channels is lower than the number of ADCs available. The following section 5.4 details the supported interleaved sampling modes, depending on the number of ADCs available in the MCU.

The maximum real-time sampling rate for the digital channels is limited by the duration of a DMA transfer, i.e. 6 AHB cycles in the best case when GPIO and SRAM are not busy. However, the AHB bus is only occupied for 3 of those cycles. As illustrated in Figure 5.18, the same DMA controller can start another channel's request arbitration during the Ack cycle and the last 2 cycles of AHB bus access [21, p. 10]. Therefore, it is possible to also use interleaved sampling for the digital channels. Two channels of the same DMA controller are used, with alternating DMA requests coming from two capture/compare channels of the sampling timer. The minimum sampling period is then 3 AHB cycles, doubling the maximum achievable sampling rate for the digital MSO channels.

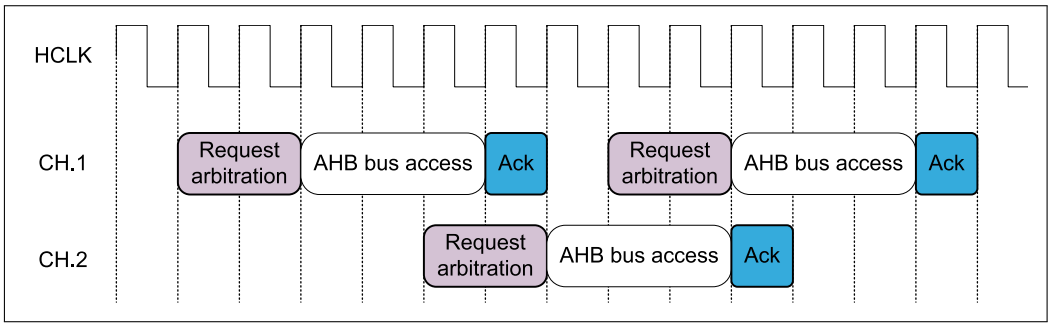

Figure 5.18: Timing diagram of two active DMA channels on AHB bus. Adapted from [21, p. 10].

5.4 Supported MSO channel configurations

All the supported mixed-signal oscilloscope channel configurations are described in this section. A channel configuration includes the number of enabled analog and digital channels as well as the number of ADCs available. Multiple configurations are available when interleaved sampling is used. If the selected sampling rate exceeds the rate achievable without interleaving, the oscilloscope automatically switches into the interleaved mode (if the number of enabled analog channels and ADCs allows it). Each configuration is primarily described by a timing diagram. A corresponding diagram of the waveform buffer is also included to show the order of the samples within. The firmware must rearrange these samples into the correct order before the waveform data is sent to Data Plotter and displayed. In order to save space, the legends for both types of diagrams are shown separately in Figures 5.20, 5.19.

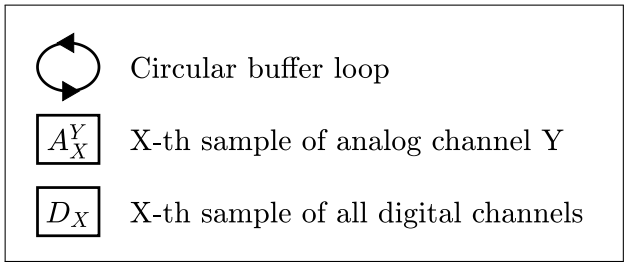

Figure 5.19: Legend for MSO waveform buffer diagrams

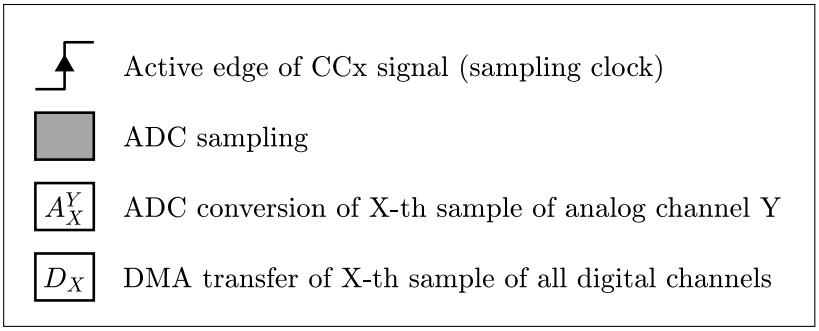

Figure 5.20: Legend for MSO timing diagrams

5.4.1 Digital channel configurations

All supported MCUs are functionally equivalent in terms of digital channels. Interleaved sampling by 2 DMA channels is available. It is activated if all analog channels are disabled or if they are interleaved by 2 ADCs per channel. Digital channels cannot be enabled if analog channels are interleaved by 4 ADCs per channel – there are no sampling timer capture/compare channels left. The number of digital channels has no impact on their sampling – each digital sample is 2 bytes long, each bit corresponding to one digital channel. Therefore, up to 16 digital channels can be sampled by a single DMA transfer. If analog channels are enabled, the sampling clock of the digital channels is delayed by the ADC sampling time t_{SMP} and trigger latency t_{LATR} to ensure synchronization of analog and digital samples.

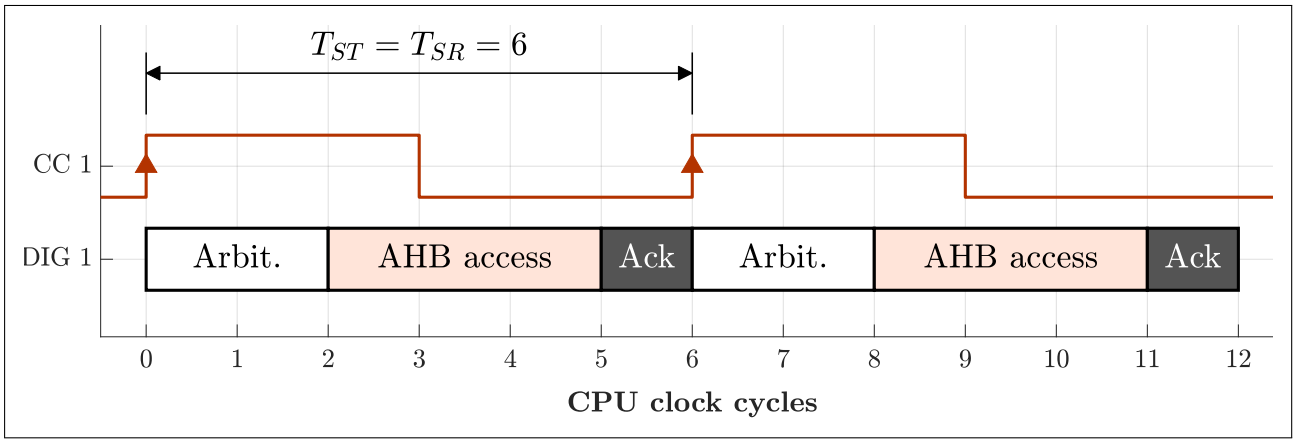

Figure 5.21: MSO timing diagram, digital channels only. Record length is k samples after k sampling timer periods (2 shown).

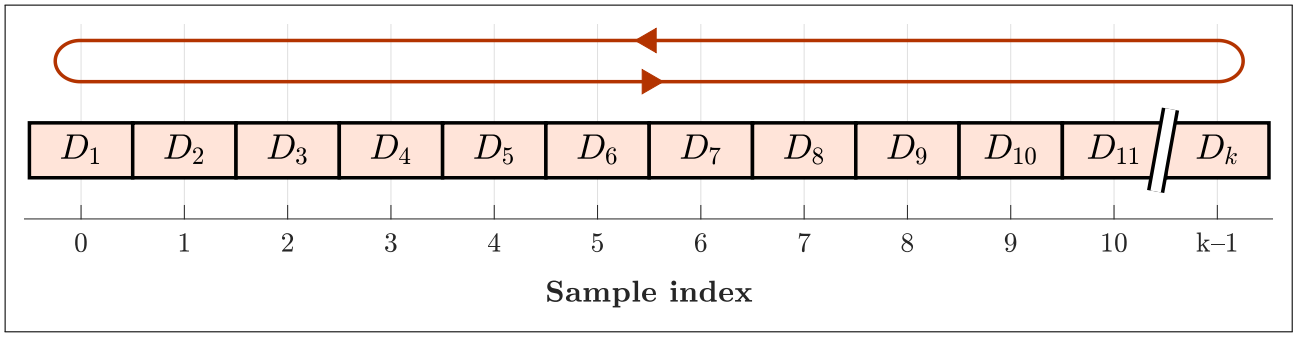

Figure 5.22: MSO waveform buffer, digital channels only. Record length is k samples after k sampling timer periods.

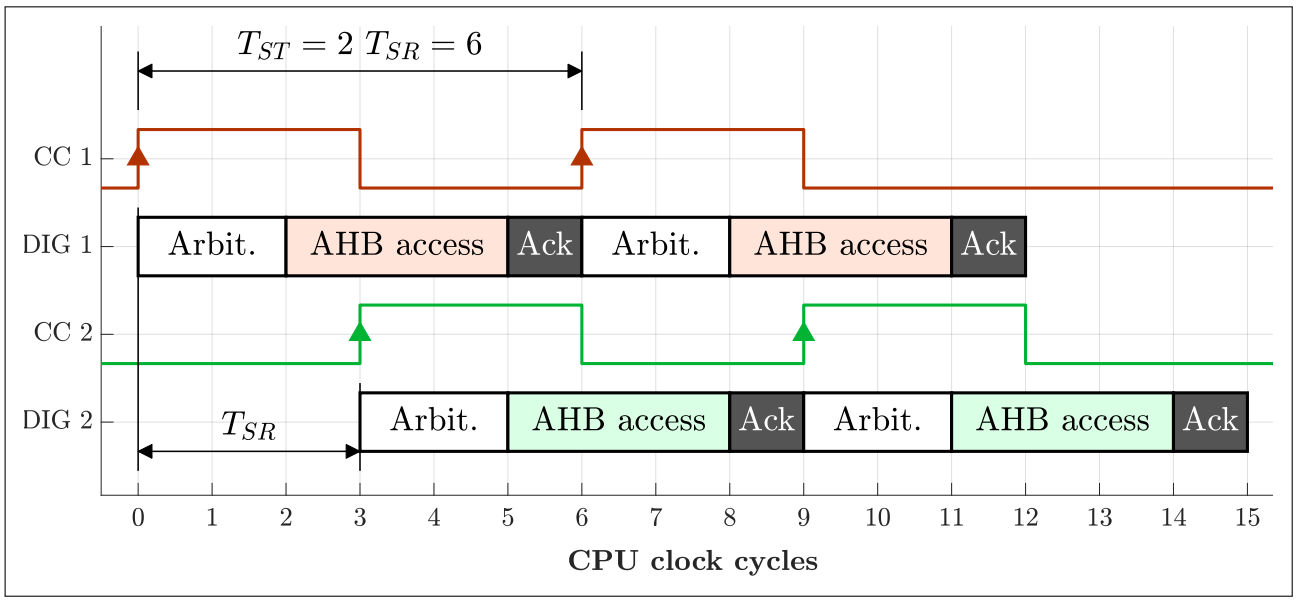

Figure 5.23: MSO timing diagram, digital channels only, interleaved by 2 DMA channels. Maximum sampling rate is doubled (vs. using 1 DMA channel). Record length is $2k$ samples after k sampling timer periods (2 shown).

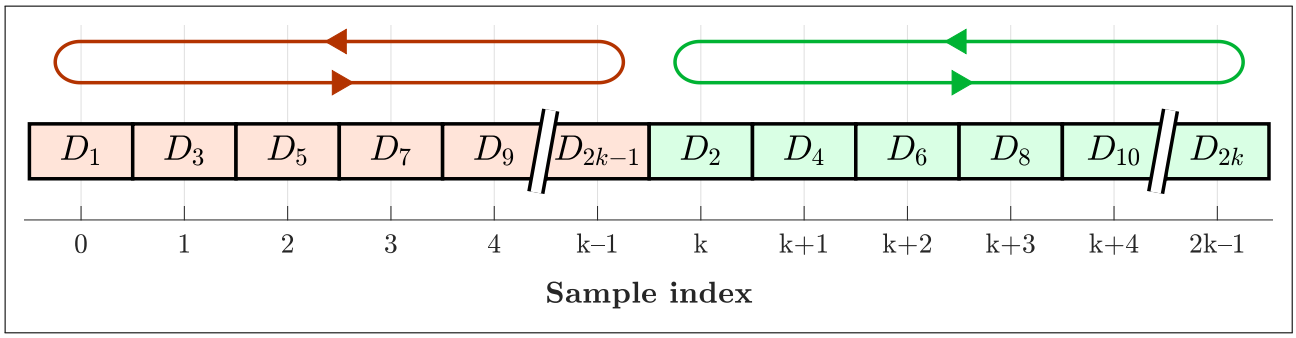

Figure 5.24: MSO waveform buffer, digital channels only, interleaved by 2 DMA channels. Record length is $2k$ samples after k sampling timer periods.

5.4.2 Analog channel configurations for MCUs with 1 ADC

Microcontrollers with only a single embedded ADC are the most limited, as detailed in the following figures. When more than one analog channel is enabled, each is sampled alternately by the ADC, introducing a time delay between the acquired samples. The maximum real-time sampling rate is halved if two channels are enabled and quartered if four channels are enabled. Interleaved sampling is not possible.

MCU with 1 ADC sampling 1 analog channel

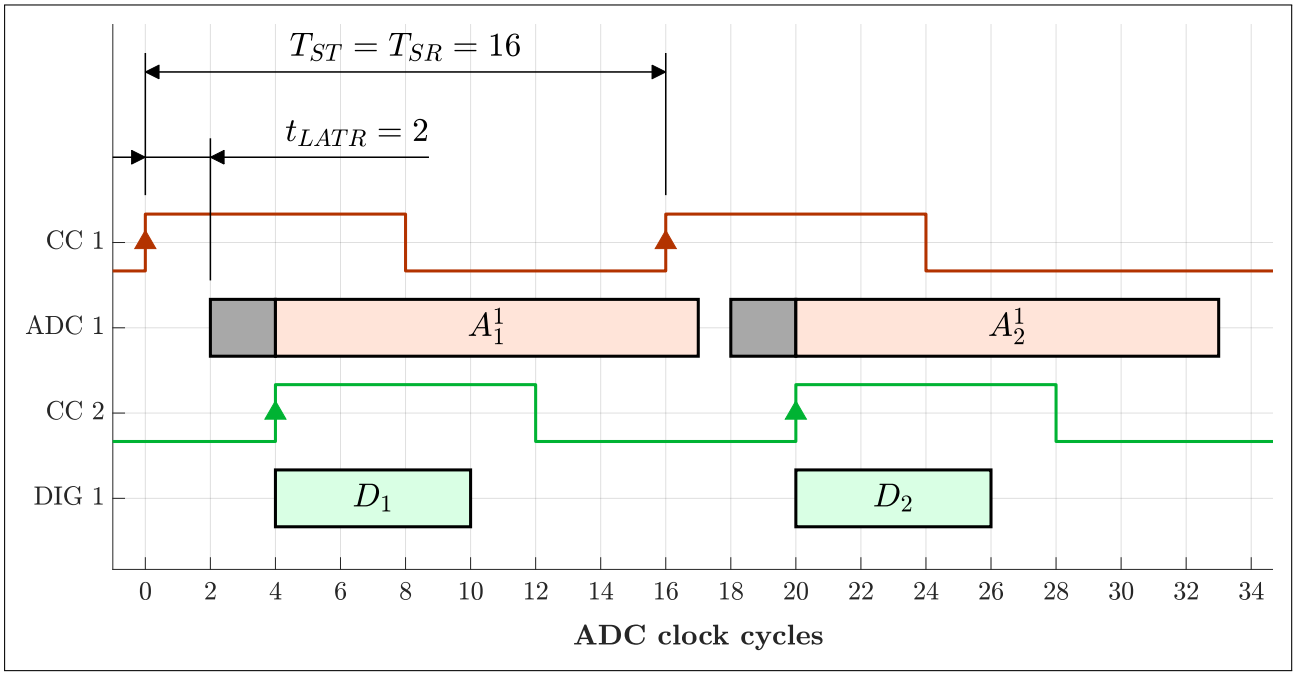

Figure 5.25: MSO timing diagram, 1 analog channel (MCU with 1 ADCs). Using 1 ADC independently. Record length is k samples after k sampling timer periods (2 shown).

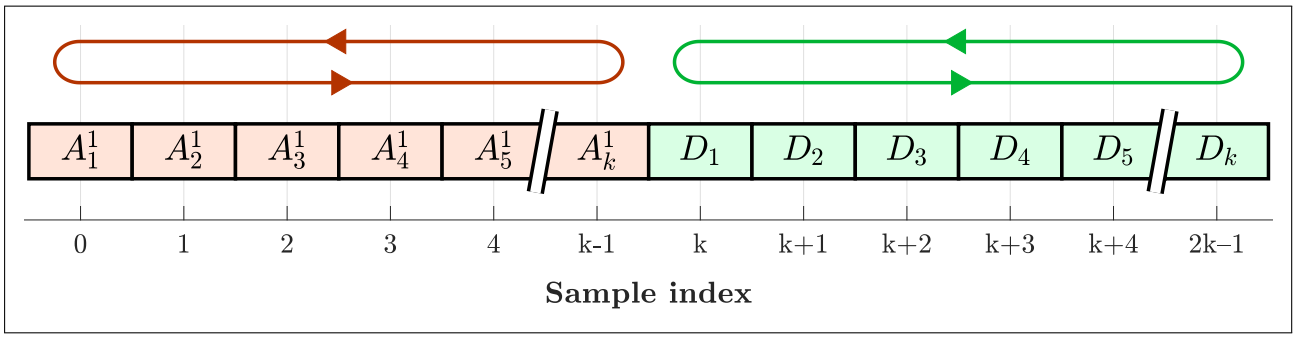

Figure 5.26: MSO waveform buffer, 1 analog channel (MCU with 1 ADCs). Using 1 ADC independently. Record length is k samples after k sampling timer periods.

MCU with 1 ADC sampling 2 analog channels

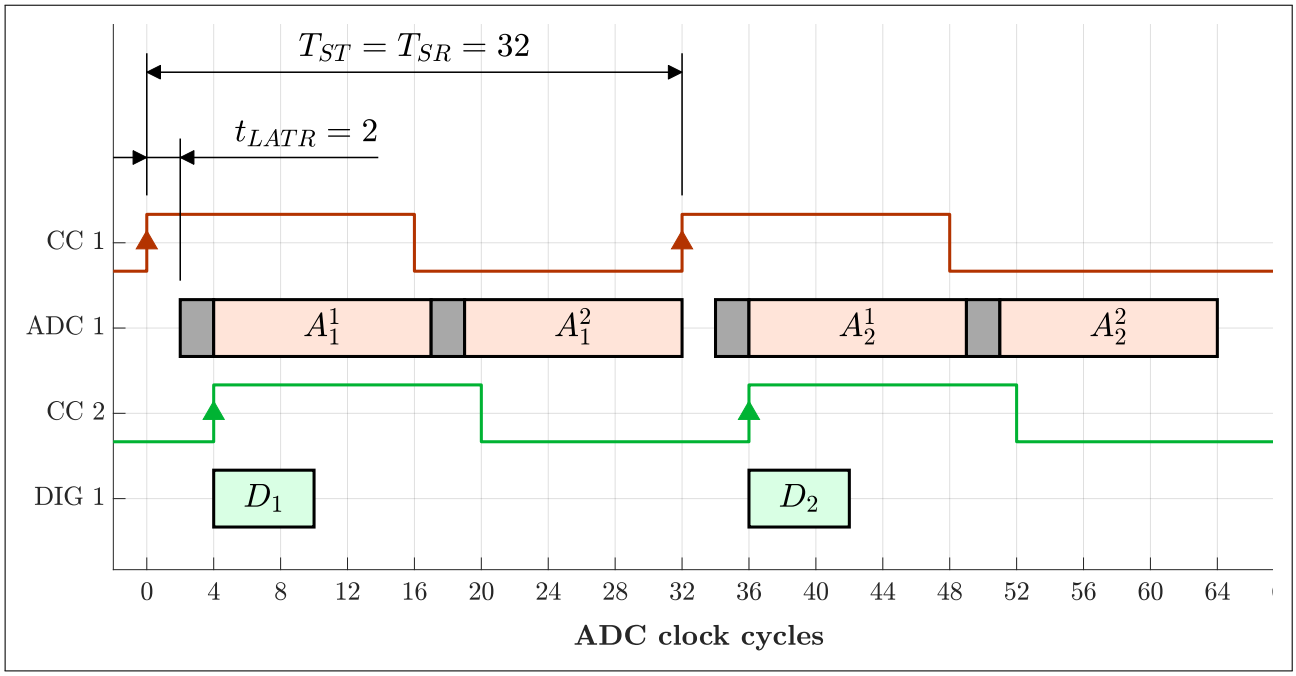

Figure 5.27: MSO timing diagram, 2 analog channels (MCU with 1 ADC). Maximum sampling rate is halved (vs. sampling 1 analog channel). Record length is k samples after k sampling timer periods (2 shown).

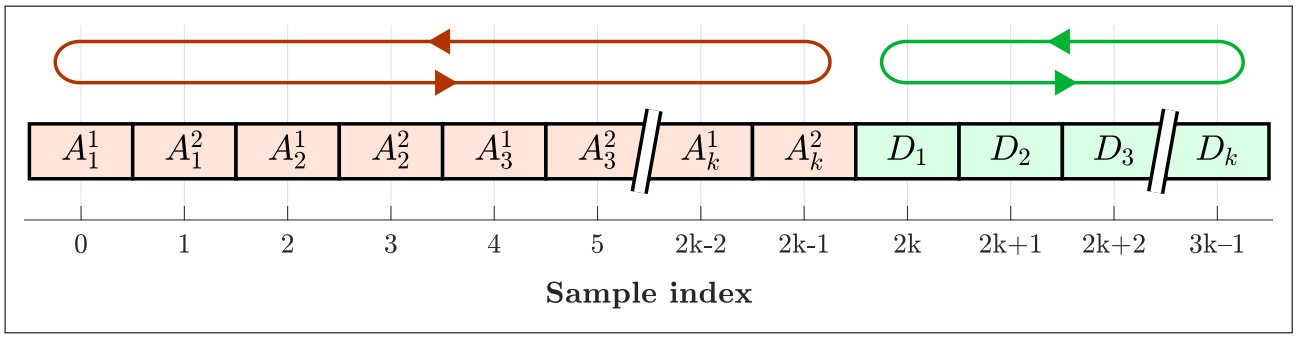

Figure 5.28: MSO waveform buffer, 2 analog channels (MCU with 1 ADC). Record length is k samples after k sampling timer periods.

MCU with 1 ADC sampling 4 analog channels

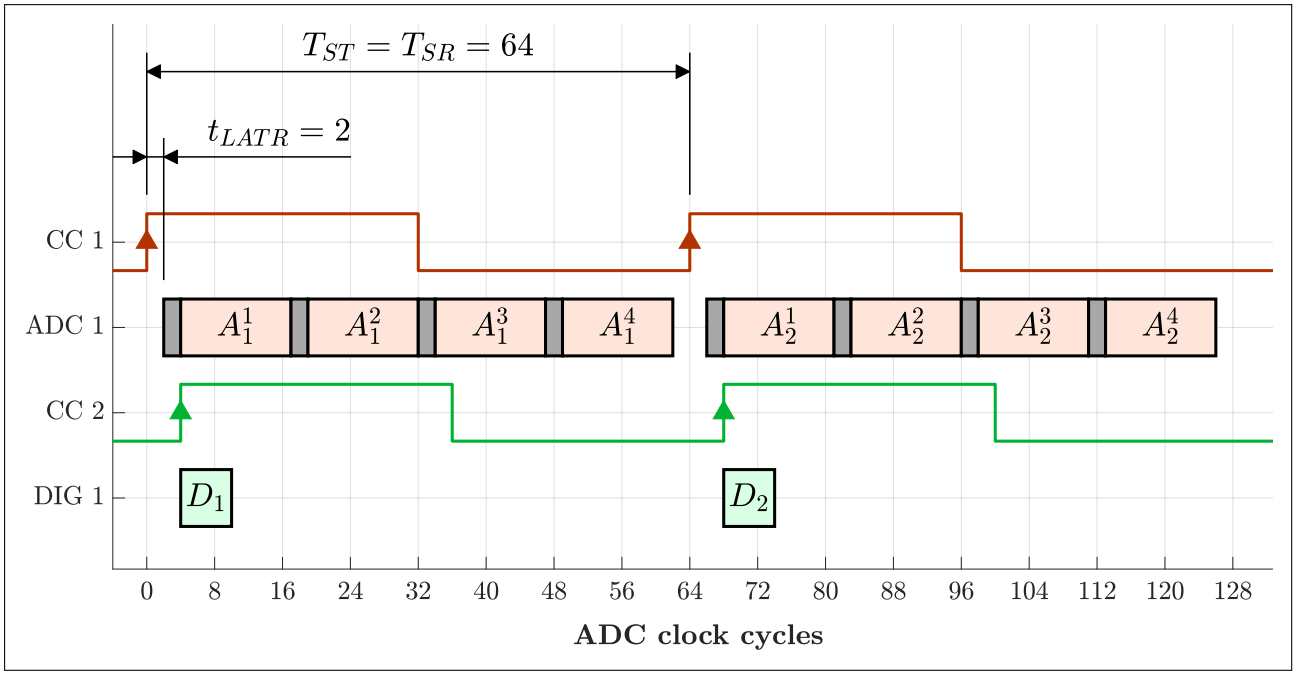

Figure 5.29: MSO timing diagram, 4 analog channels (MCU with 1 ADC). Maximum sampling rate is quartered (vs. sampling 1 analog channel). Record length is k samples after k sampling timer periods (2 shown).

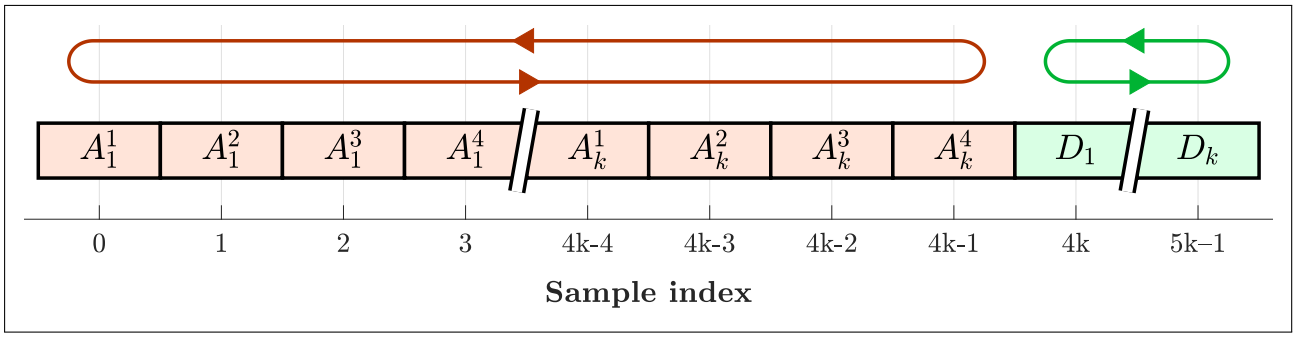

Figure 5.30: MSO waveform buffer, 4 analog channels (MCU with 1 ADC). Record length is k samples after k sampling timer periods.

5.4.3 Analog channel configurations for MCUs with 2 ADCs

Microcontrollers with two embedded ADCs give a modest variety of possible analog channel configurations, as detailed in the following figures. When all four channels are enabled, they must be sampled alternately, halving the maximum sampling rate and introducing a delay between channel samples. For two analog channels, the pair of ADCs samples them simultaneously. Interleaved sampling is available when a single analog channel is enabled, doubling the maximum sampling rate.

MCU with 2 ADCs sampling 1 analog channel

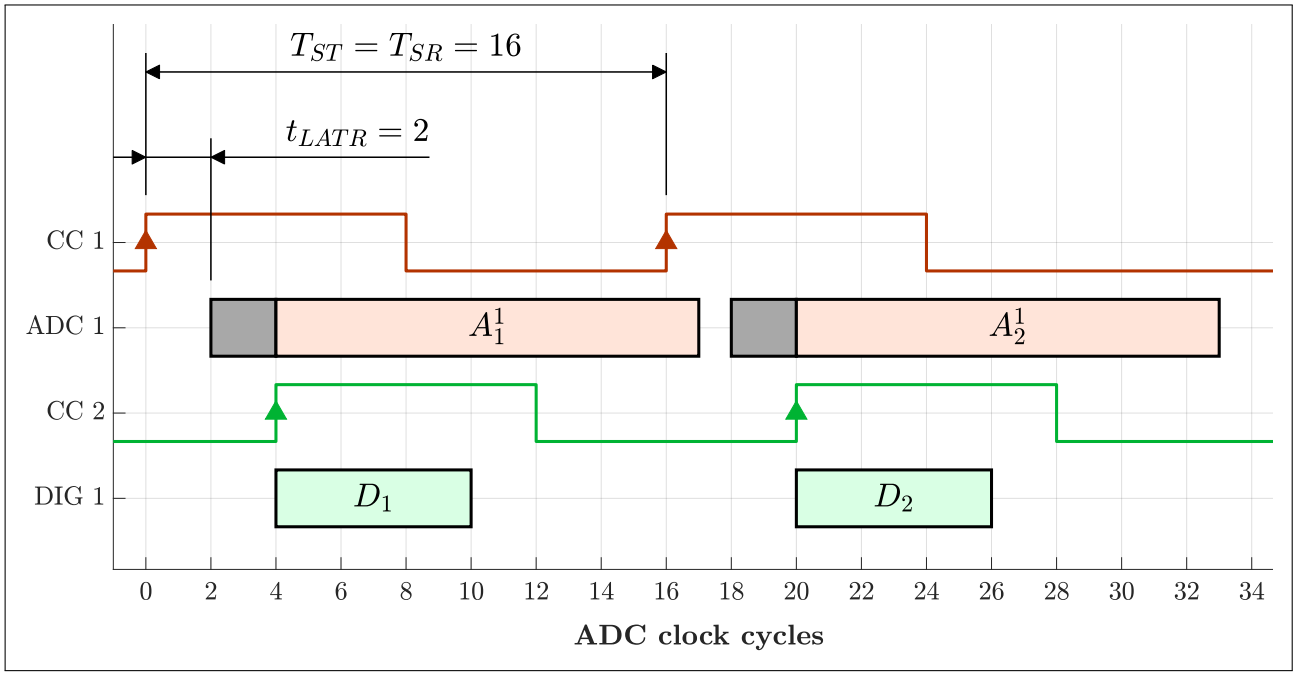

Figure 5.31: MSO timing diagram, 1 analog channel (MCU with 2 ADCs). Using 1 ADC independently. Record length is k samples after k sampling timer periods (2 shown).

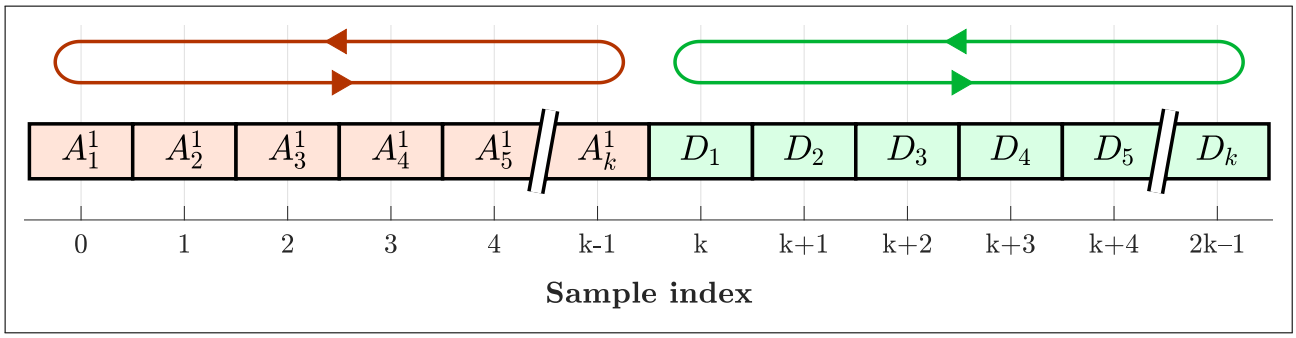

Figure 5.32: MSO waveform buffer, 1 analog channel (MCU with 2 ADCs). Using 1 ADC independently. Record length is k samples after k sampling timer periods.

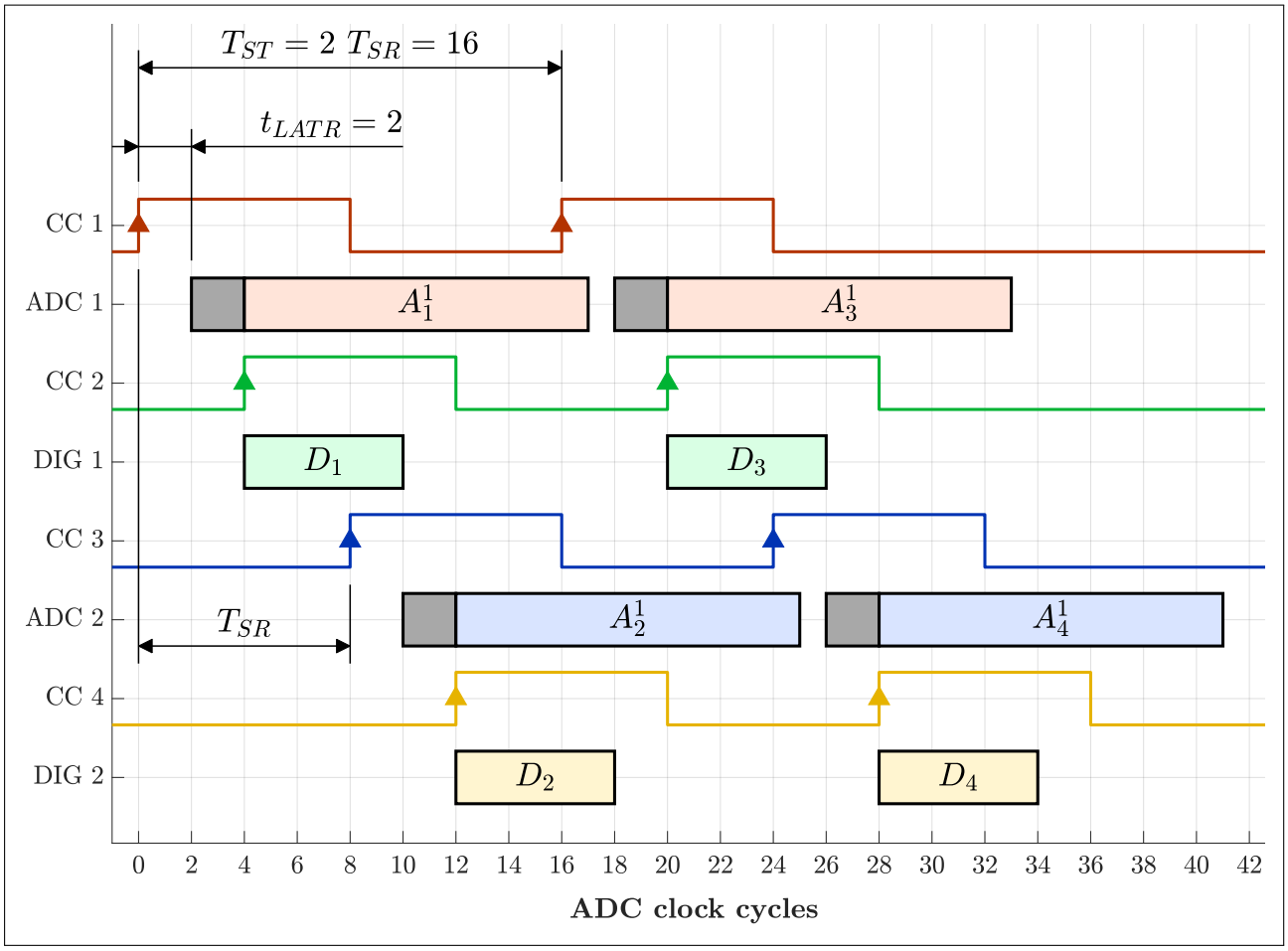

Figure 5.33: MSO timing diagram, 1 analog channel, interleaved by 2 ADCs (MCU with 2 ADCs). Using 2 ADCs independently. Maximum sampling rate is doubled (vs. sampling 2 analog channels). Record length is $2k$ samples after k sampling timer periods (2 shown).

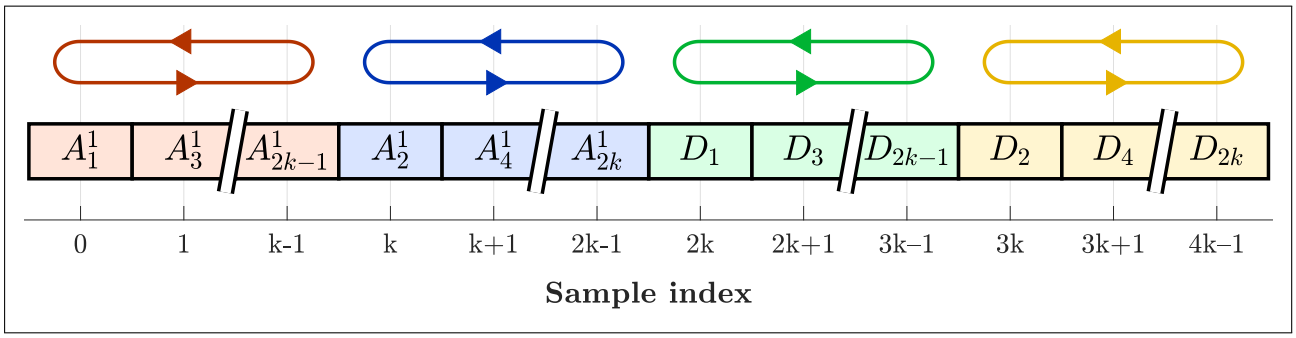

Figure 5.34: MSO waveform buffer, 1 analog channel, interleaved by 2 ADCs (MCU with 2 ADCs). Using 2 ADCs independently. Record length is $2k$ samples after k sampling timer periods.

MCU with 2 ADCs sampling 2 analog channels

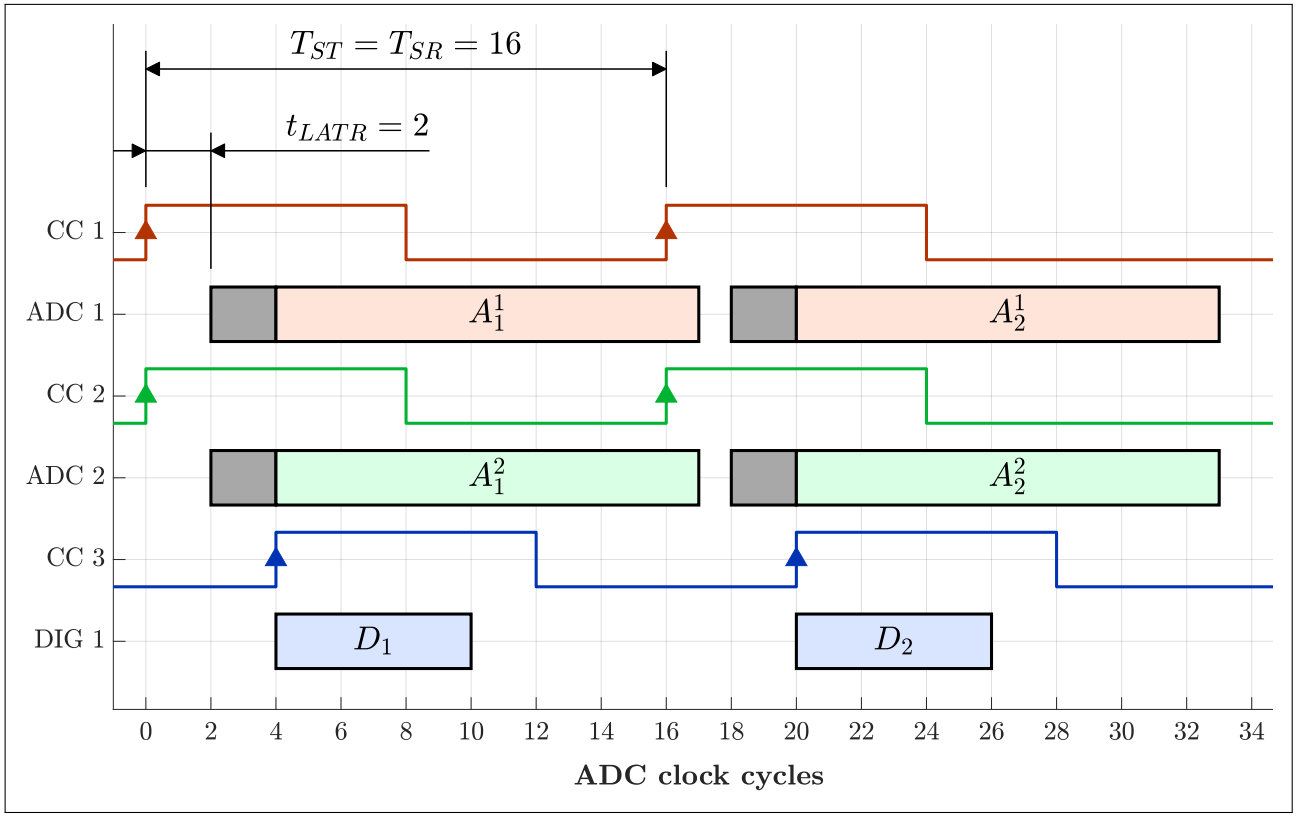

Figure 5.35: MSO timing diagram, 2 analog channels (MCU with 2 ADCs). Using 2 ADCs independently. Record length is k samples after k sampling timer periods (2 shown).

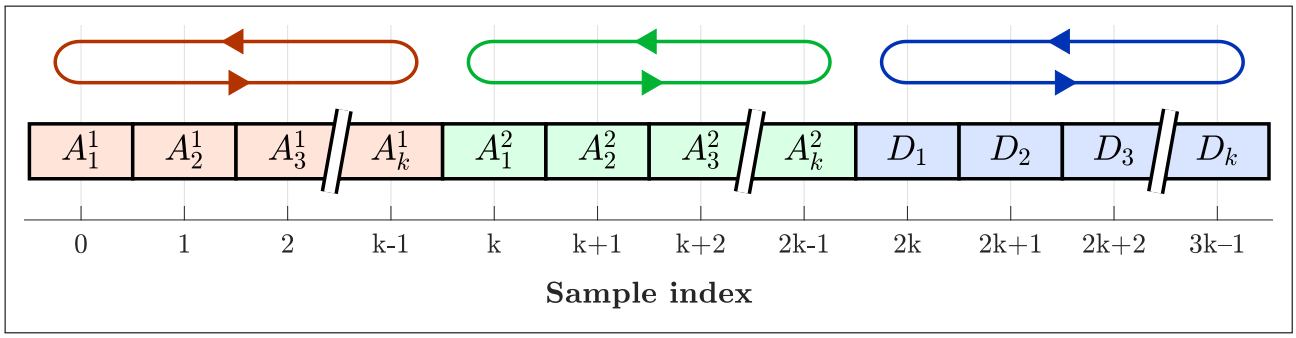

Figure 5.36: MSO waveform buffer, 2 analog channels (MCU with 2 ADCs). Using 2 ADCs independently. Record length is k samples after k sampling timer periods.

MCU with 2 ADCs sampling 4 analog channels

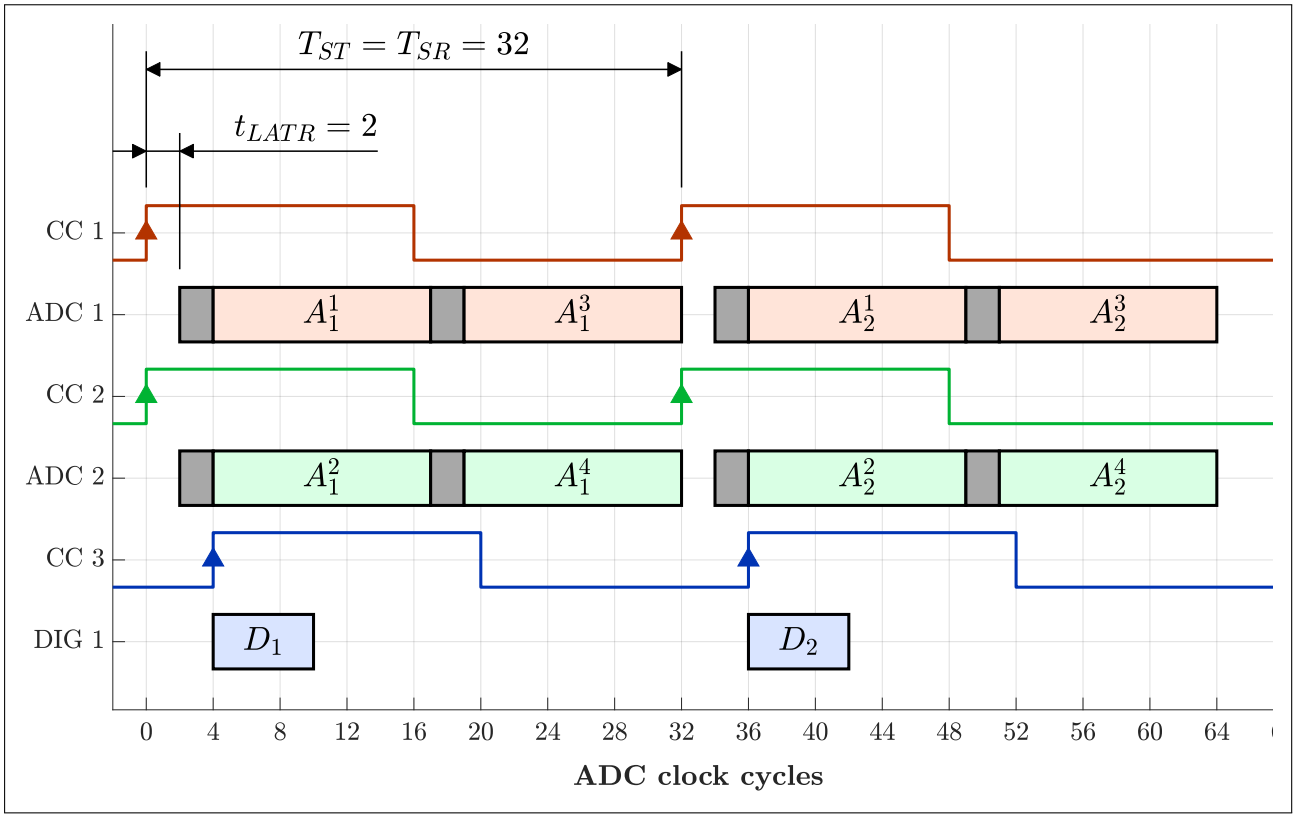

Figure 5.37: MSO timing diagram, 4 analog channels (MCU with 2 ADCs). Using 2 ADCs independently. Maximum sampling rate is halved (vs. sampling 2 analog channels). Record length is k samples after k sampling timer periods (2 shown).

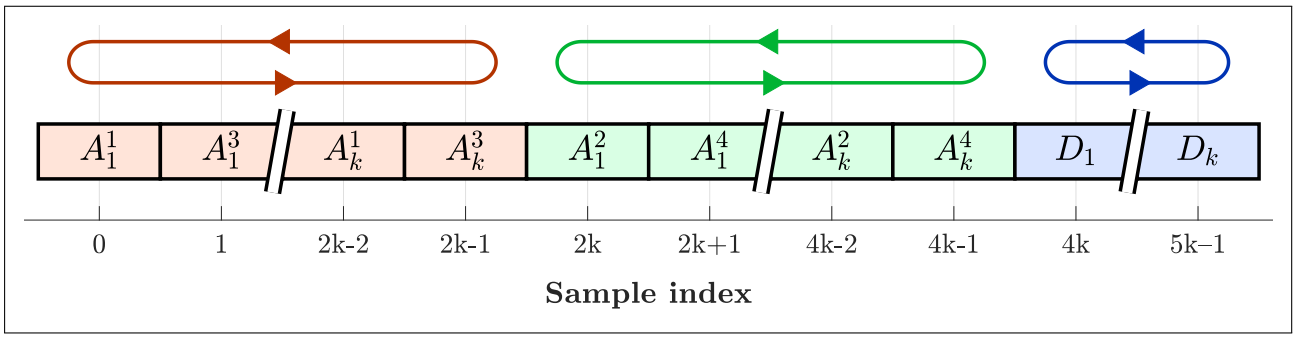

Figure 5.38: MSO waveform buffer, 4 analog channels (MCU with 2 ADCs). Using 2 ADCs independently. Record length is k samples after k sampling timer periods.

5.4.4 Analog channel configurations for MCUs with 4 ADCs

Microcontrollers with four embedded ADCs give the largest variety of possible analog channel configurations, as detailed in the following figures. When all four channels are enabled, there is one ADC for each, sampling simultaneously. For two analog channels, a pair of ADCs can be used for interleaved sampling to double the maximum sampling rate. Finally, for a single analog channel, either two or all four ADCs can be used for interleaving. The sampling rate is quadrupled in the latter case.

MCU with 4 ADCs sampling 1 analog channel

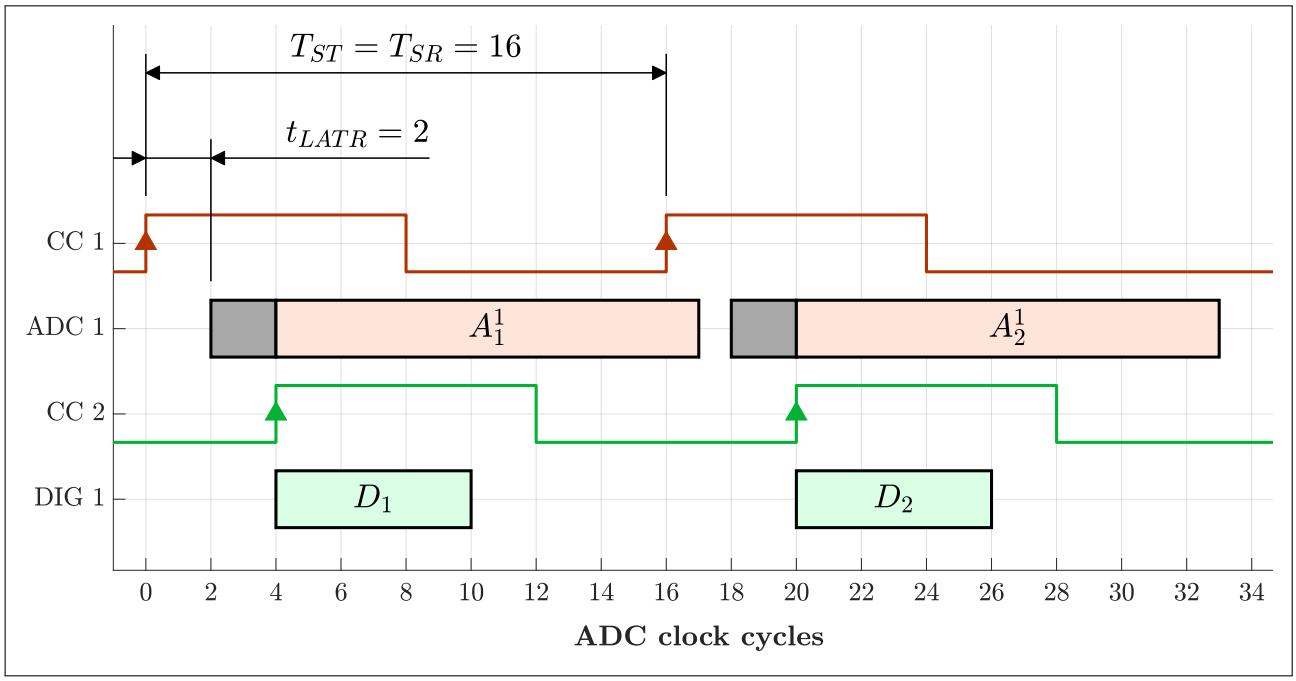

Figure 5.39: MSO timing diagram, 1 analog channel (MCU with 4 ADCs). Using 1 ADC independently. Record length is k samples after k sampling timer periods (2 shown).

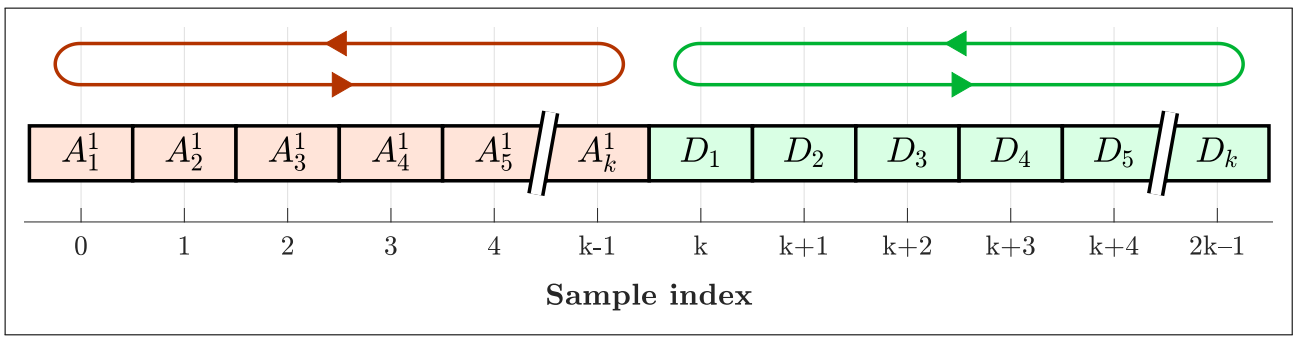

Figure 5.40: MSO waveform buffer, 1 analog channel (MCU with 4 ADCs). Using 1 ADC independently. Record length is k samples after k sampling timer periods.

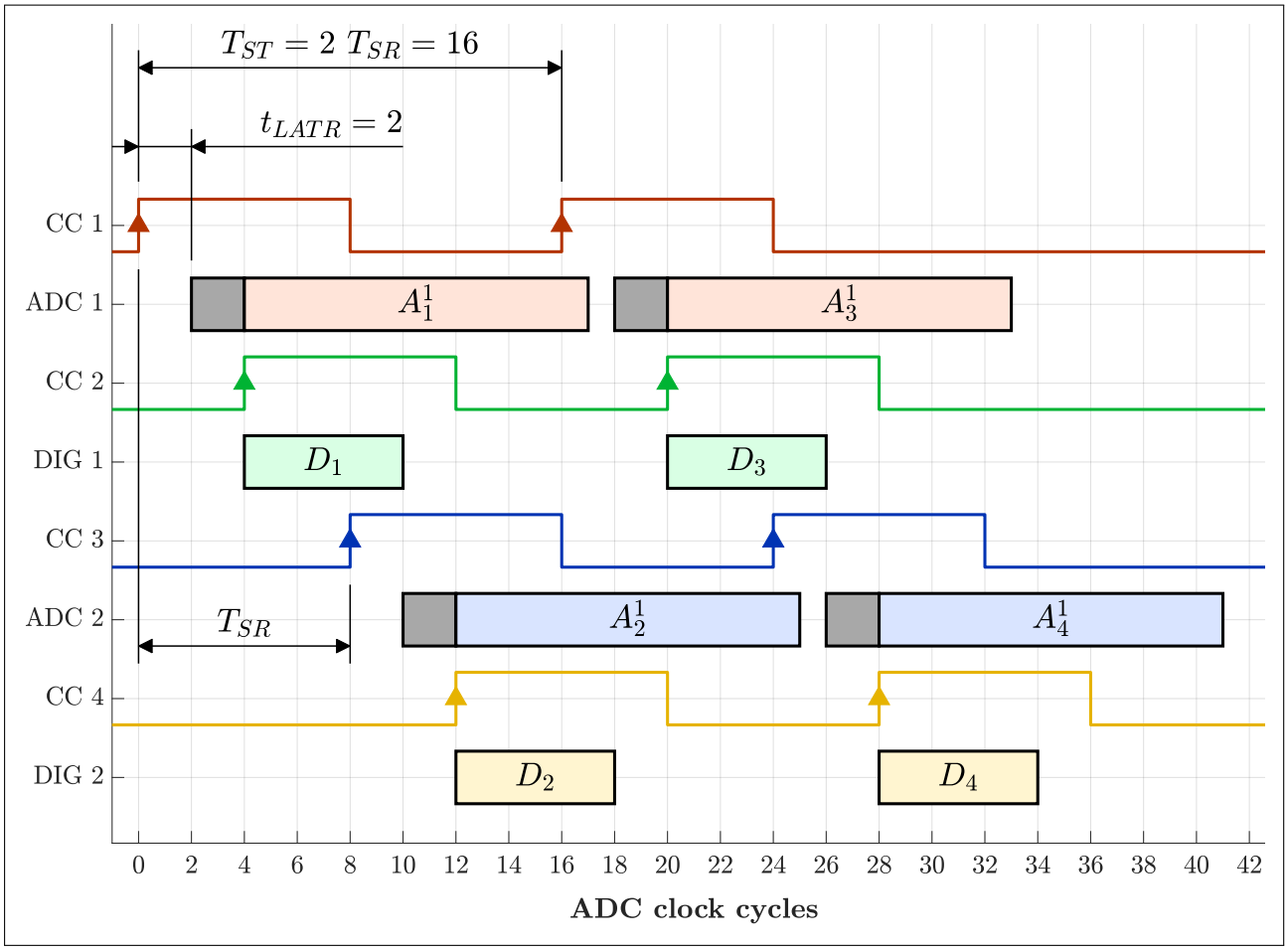

Figure 5.41: MSO timing diagram, 1 analog channel, interleaved by 2 ADCs (MCU with 4 ADCs). Using 2 ADCs independently. Maximum sampling rate is doubled (vs. sampling 4 analog channels). Record length is $2k$ samples after k sampling timer periods (2 shown).

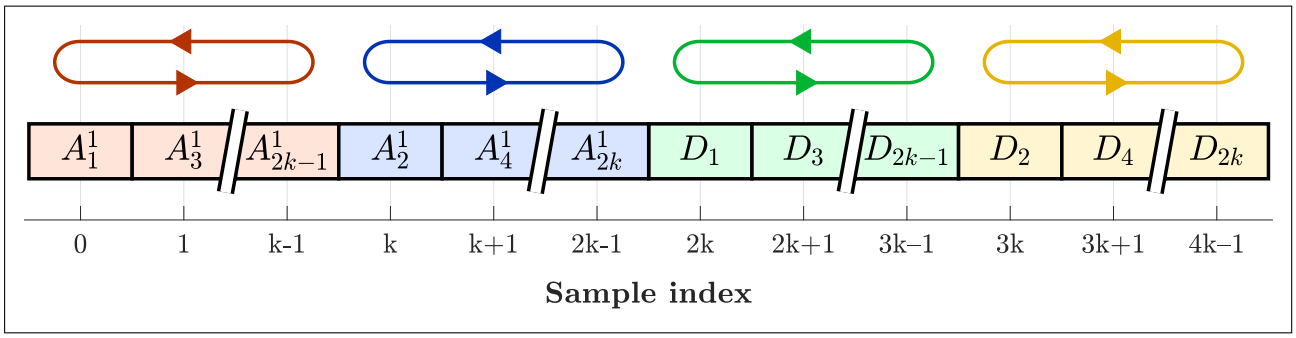

Figure 5.42: MSO waveform buffer, 1 analog channel, interleaved by 2 ADCs (MCU with 4 ADCs). Using 2 ADCs independently. Record length is $2k$ samples after k sampling timer periods.

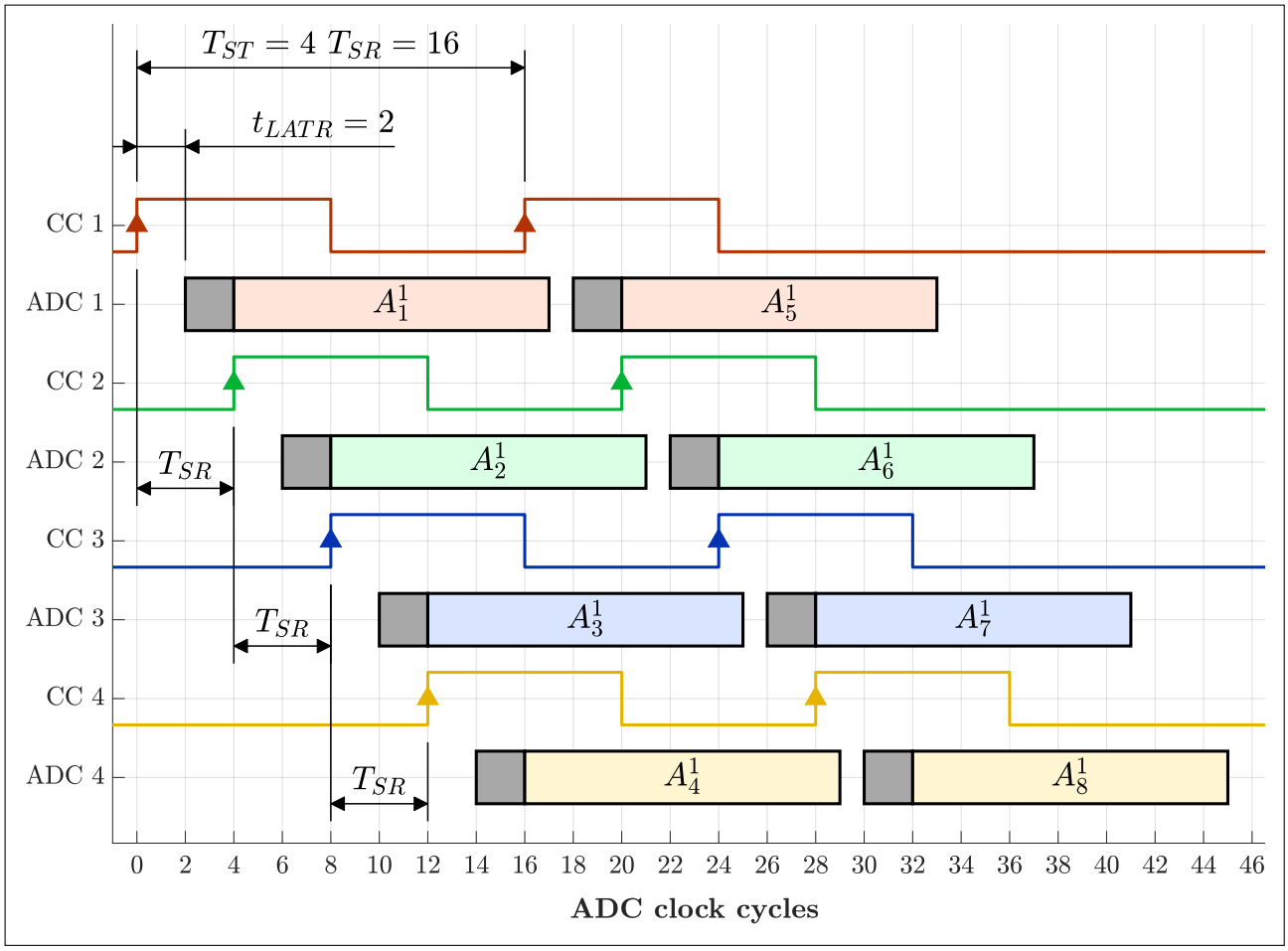

Figure 5.43: MSO timing diagram, 1 analog channel, interleaved by 4 ADCs (MCU with 4 ADCs). Using 4 ADCs independently. Maximum sampling rate is quadrupled (vs. sampling 4 analog channels). Record length is $4k$ samples after k sampling timer periods (2 shown).

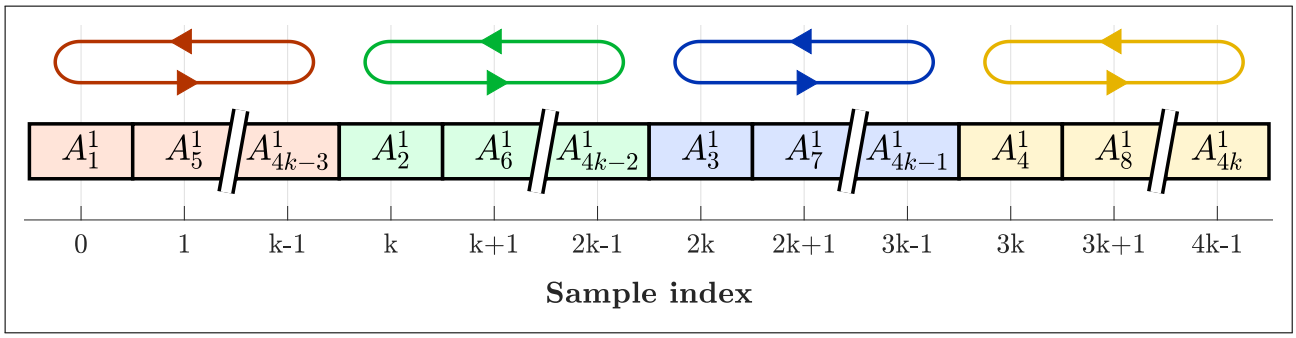

Figure 5.44: MSO waveform buffer, 1 analog channel, interleaved by 4 ADCs (MCU with 4 ADCs). Using 4 ADCs independently. Record length is $4k$ samples after k sampling timer periods.

MCU with 4 ADCs sampling 2 analog channels

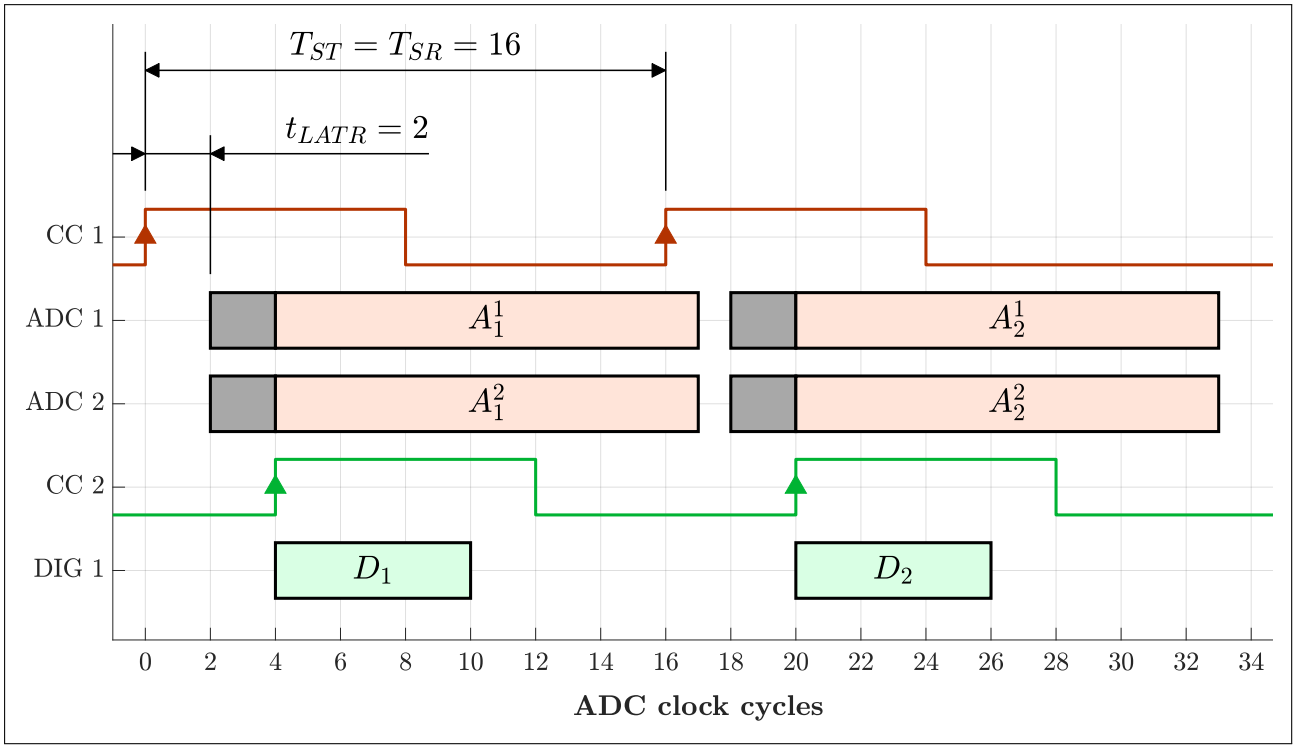

Figure 5.45: MSO timing diagram, 2 analog channels (MCU with 4 ADCs). Using 2 ADCs in dual simultaneous mode. Record length is k samples after k sampling timer periods (2 shown).

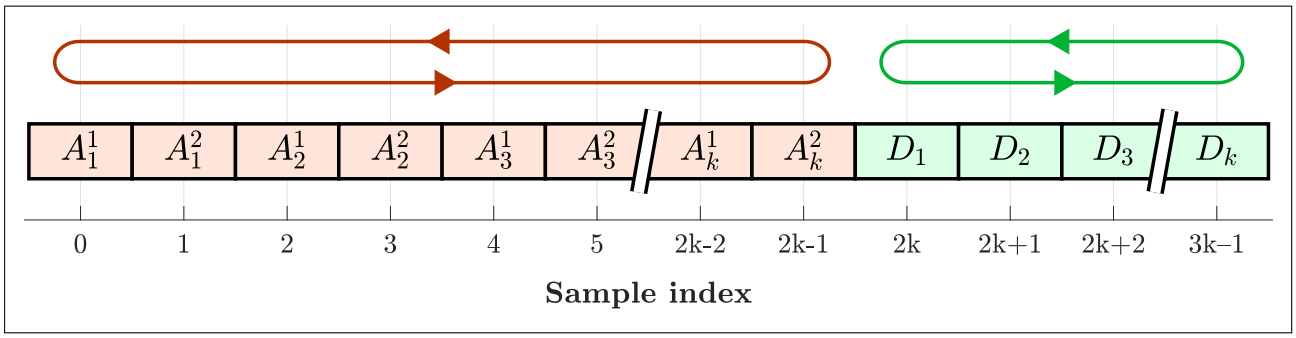

Figure 5.46: MSO waveform buffer, 2 analog channels (MCU with 4 ADCs). Using 2 ADCs in dual simultaneous mode. Record length is k samples after k sampling timer periods.

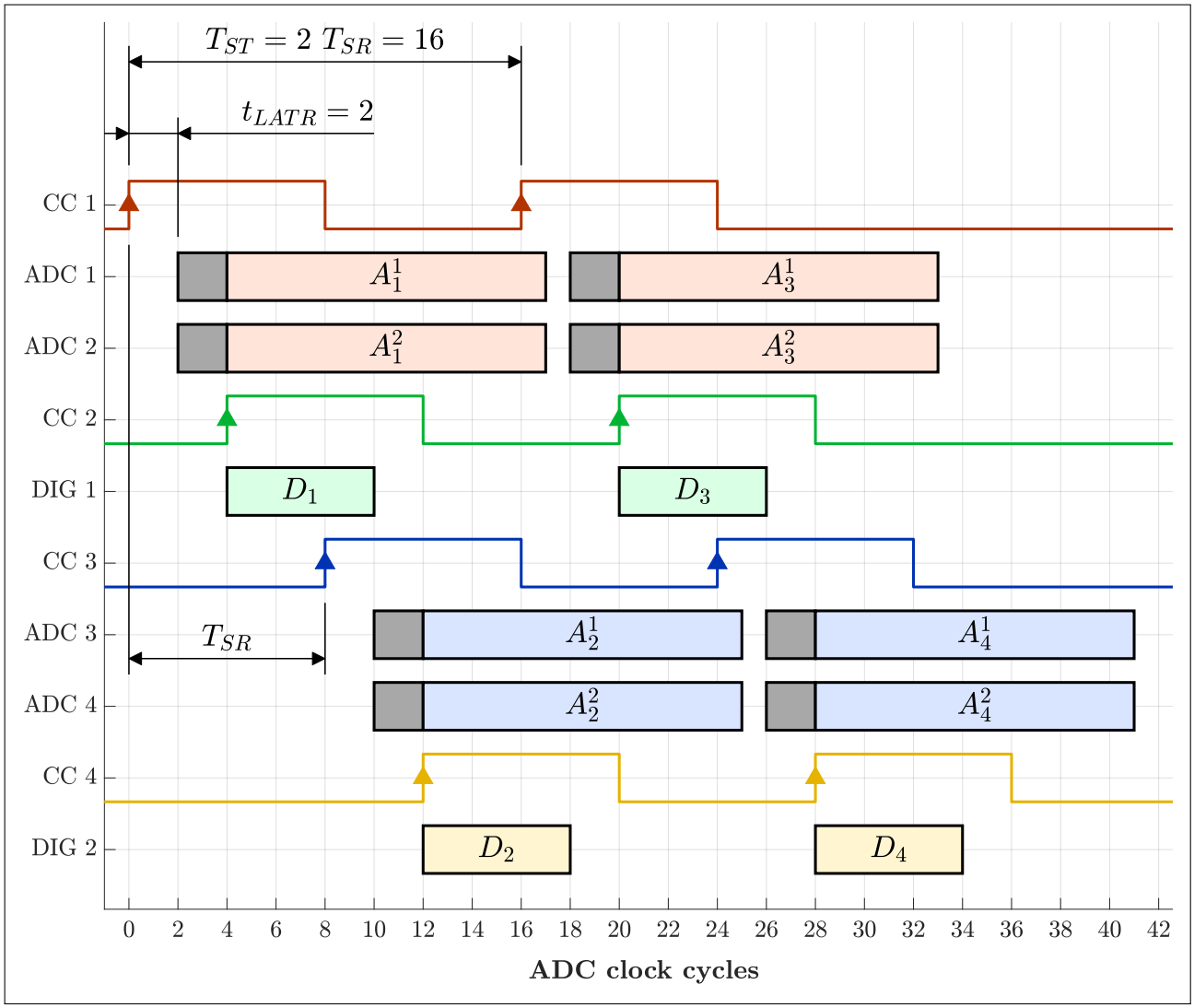

Figure 5.47: MSO timing diagram, 2 analog channels, each interleaved by 2 ADCs (MCU with 4 ADCs). Using 2 pairs of ADCs in dual simultaneous mode (4 ADCs total). Record length is $2k$ samples after k sampling timer periods (2 shown).

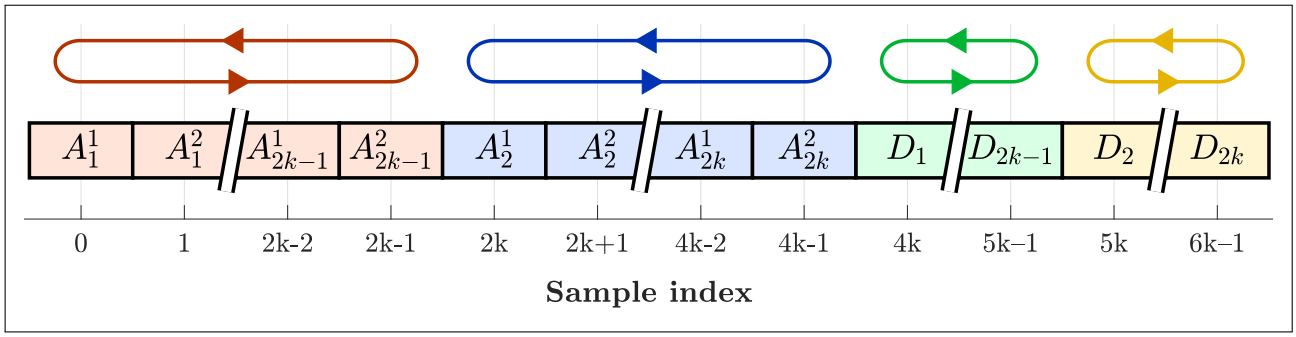

Figure 5.48: MSO waveform buffer, 2 analog channels, each interleaved by 2 ADCs (MCU with 4 ADCs). Using 2 pairs of ADCs in dual simultaneous mode (4 ADCs total). Record length is $2k$ samples after k sampling timer periods.

MCU with 4 ADCs sampling 4 analog channels

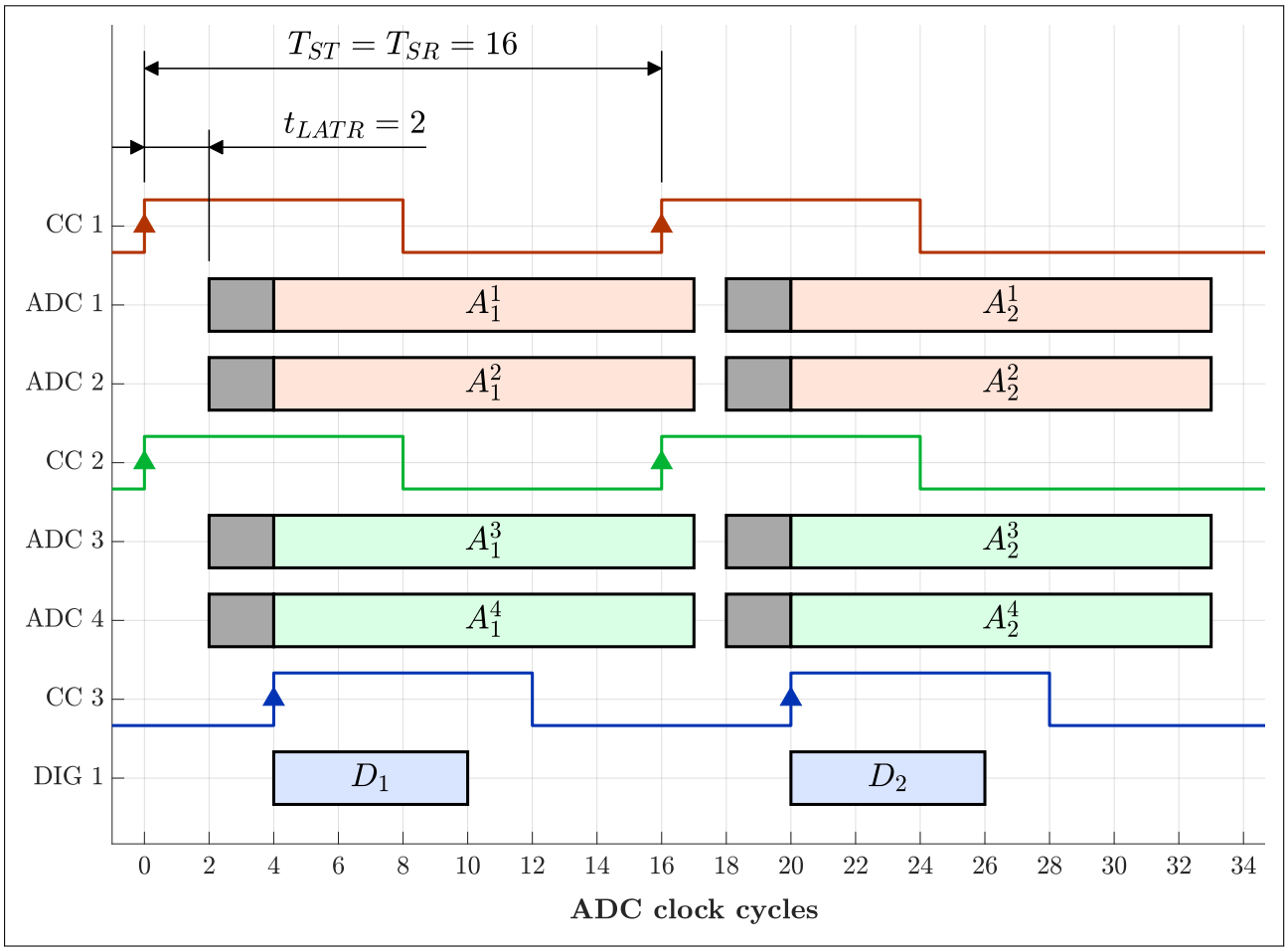

Figure 5.49: MSO timing diagram, 4 analog channels (MCU with 4 ADCs). Using 2 pairs of ADCs in dual simultaneous mode (4 ADCs total). Record length is k samples after k sampling timer periods (2 shown).

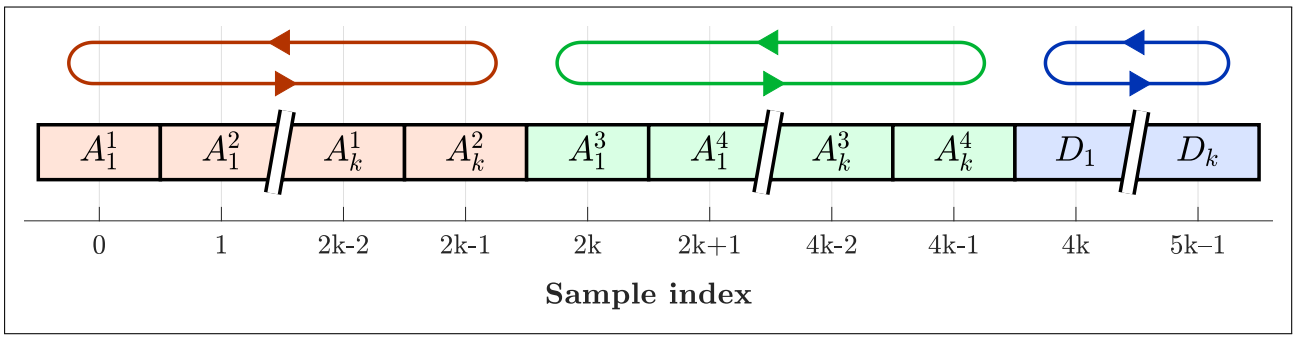

Figure 5.50: MSO waveform buffer, 4 analog channels (MCU with 4 ADCs). Using 2 pairs of ADCs in dual simultaneous mode (4 ADCs total). Record length is k samples after k sampling timer periods.

Interleaved sampling issues caused by STM32F303RE erratum

Unfortunately, during the development of the mixed-signal oscilloscope, issues caused by STM32F303RE microcontroller erratum "2.3.1: DMA Overrun in dual interleaved mode with single DMA channel" [24] were encountered. The errata sheet states that ADC overruns may occur when dual interleaved ADC mode is used with MDMA mode (one DMA channel is used to transfer samples from 2 ADCs simultaneously). The listed workaround calls for separate DMA channels to be used for each ADC.

The dual interleaved mode was used at first in attempts to implement interleaving of 1 analog channel by 4 ADCs. ADC 1 (master) was triggered first (from TIM CCx). Using dual interleaved mode, ADC 2 (slave) was set to start its conversion 4 cycles later. ADC 3 (master) was triggered 4 cycles after ADC 2 (8 cycles after ADC 1) from another CCx the same TIM. Finally, ADC 4 (slave) was set to start its conversion 4 cycles after ADC 3 (8 cycles after ADC2, 12 cycles after ADC1). The interleaved conversion time was therefore 4 cycles, while the time between conversions of the same ADC was 16 cycles. Since the minimum conversion time of a single ADC with min. sampling time is 14 CPU cycles, this approach should work. Unfortunately, it did not, regardless of whether MDMA mode or separate DMA channels were used – ADC overruns still occurred frequently.

In the end, it was found that ADC overruns occurred whenever a master ADC conversion was started before the end of the associated slave ADC conversion. For example, considering the minimum conversion time of 14 ADC cycles and a master-slave delay of 4 cycles, overruns occurred whenever the master ADC was retriggered less than 18 cycles after the first trigger (while the slave ADC conversion was still ongoing). Again, this occurred with both MDMA mode and when using separate DMA channels. The reason why the workaround stated in the errata sheet does not work remains unknown.

This issue was eventually resolved by abandoning the dual interleaved mode altogether. Instead, all four ADCs are used independently, triggered from 4 different TIM capture/compare channels. The CCRx register values are offset by 1/4 of the ARR register value each (each ADC is delayed by a quarter of the sampling timer period T_{ST}). Unfortunately, this approach involves the use of 4 distinct DMA channels, with a new DMA request generated every 4 CPU cycles (at the highest sampling rate). This does not leave sufficient AHB bandwidth for digital channels and especially arbitrary generators, since DACs are accessed through the APB bus running at half the speed of the AHB bus. Additionally, any SRAM or ADC access by the CPU also has the potential to cause an ADC overrun. To resolve this, all firmware variables were moved into the CCMRAM memory – accessing CCMRAM does not conflict with DMA accesses to SRAM, which remained reserved for the oscilloscope and arbitrary generator circular buffers. Accesses to the ADC peripherals by the CPU were also reduced to the bare minimum (checking AWD flags for trigger).

Following this issue and the extensive debugging involved, the dual interleaved mode was no longer considered reliable and was subsequently avoided even for interleaving 2 ADCs per channel. Instead, multiple TIM CCx triggers are used with ADCs in independent mode (2 ADCs, 1 channel) or dual simultaneous mode (4 ADCs, 2 channels). Extensive testing showed no ADC overrun issues with this implementation.

Chapter 6

Pulse generators

In the course of electronics work, it is often necessary to not only measure signals but also generate them. Pulse generators are perhaps the simplest type of signal generators, only capable of generating square waves as illustrated in Figure 6.1. Typically, pulse generators are used to drive power switches, modulators or to generate digital clock signals. They can also be used to generate various reference signals, e.g. for step response analysis of a circuit.

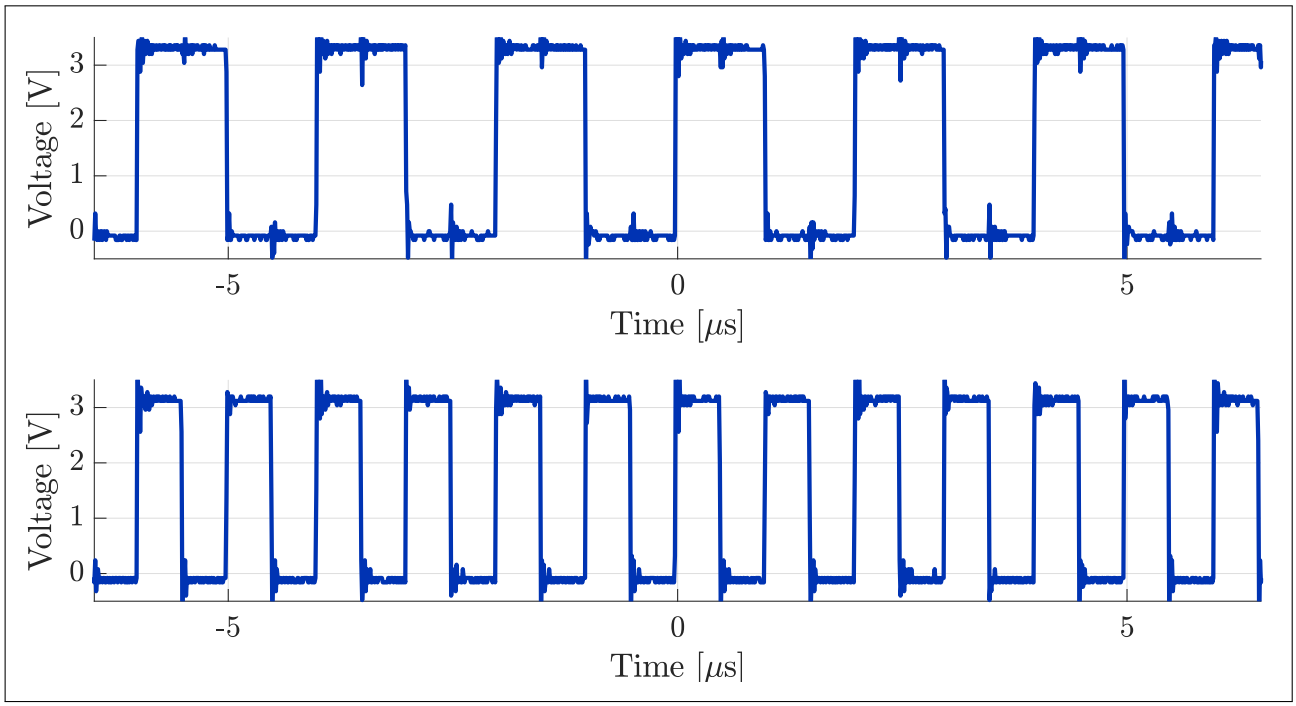

Figure 6.1: Examples of pulse generator output signals

The developed VSVI platform provides one or more pulse generators depending on MCU capabilities. The pulse generator TUI tab is shown in Figure 6.2. The pulse generators are denoted as "PWM" and numbered ("PWM 1", "PWM 2", etc.). Each generator can have one or more output channels, denoted by a letter suffix ("PWM 1A", "PWM 1B", "PWM 2A", etc.). The output signal's positive duty cycle/pulse width can be set individually for each channel, while the frequency is the same for all channels of a given generator. Each channel's output can be individually disabled, leaving the corresponding GPIO pin floating (high-Z state).

Each pulse generator is implemented using one MCU timer as illustrated in Figure 6.3. The frequency of the generated signal is equal to the timer frequency obtained from the timer clock frequency f_{TIM} according to equation 3.1. Therefore, all generated frequencies are fractions of the timer clock frequency and the frequency resolution is greatly diminished when the generated frequencies approach the timer clock frequency. Theoretically, the maximum output frequency is half the timer clock frequency. However, it may not be possible to drive the corresponding GPIO output pin at that frequency, depending on the load capacitance and other factors in accordance with the I/O AC characteristics found in the MCU datasheet [23, p. 122]. Each output channel corresponds to a capture/compare channel of the pulse generator's timer. The "PWM mode 1" output compare mode is used to generate signals with various duty cycles while keeping their rising edges aligned, as described in section 3.2.

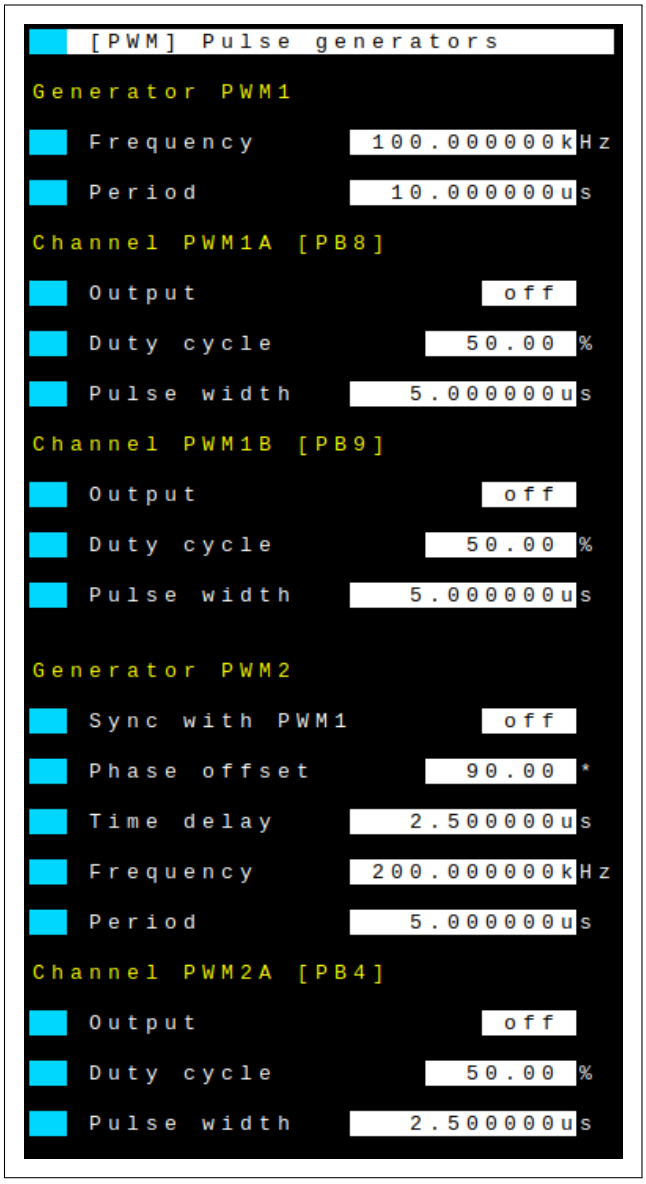

Figure 6.2: Pulse generators tab in terminal user interface

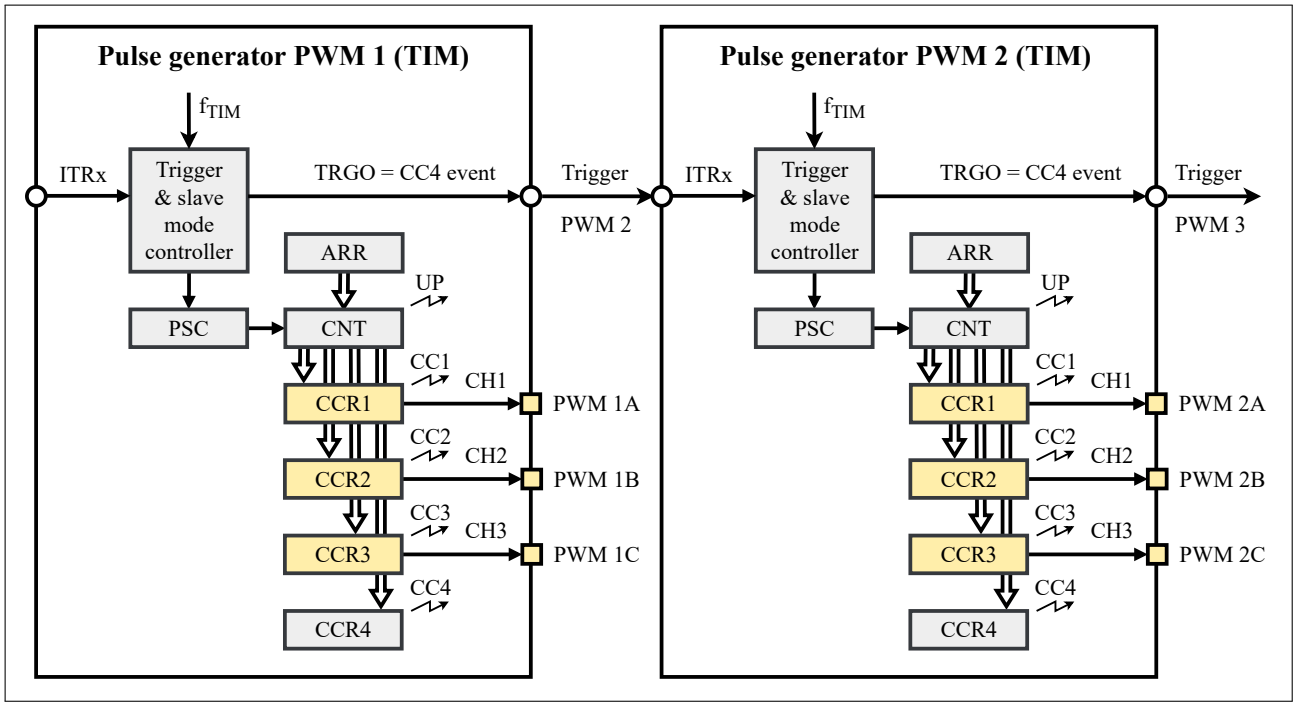

Figure 6.3: Block diagram of implemented pulse generators

6.1 Synchronization of pulse generators

If there are multiple pulse generators available, it is possible to synchronize them in a master/slave scheme. Each generator can be a slave of the previous generator, i.e. the PWM 3 generator can be a slave of PWM 2, which itself can be a slave of PWM 1. The time delay of the slave generator's output signals is configurable from zero (synchronized) up to the period of the master generator. Alternatively, a phase offset based on the period of the master generator can be set. Additionally, if frequency synchronization is enabled via the "Sync with PWM X" setting in the TUI, the frequency of the slave generator is then automatically kept equal to the master generator's frequency. This option is demonstrated in Figure 6.4

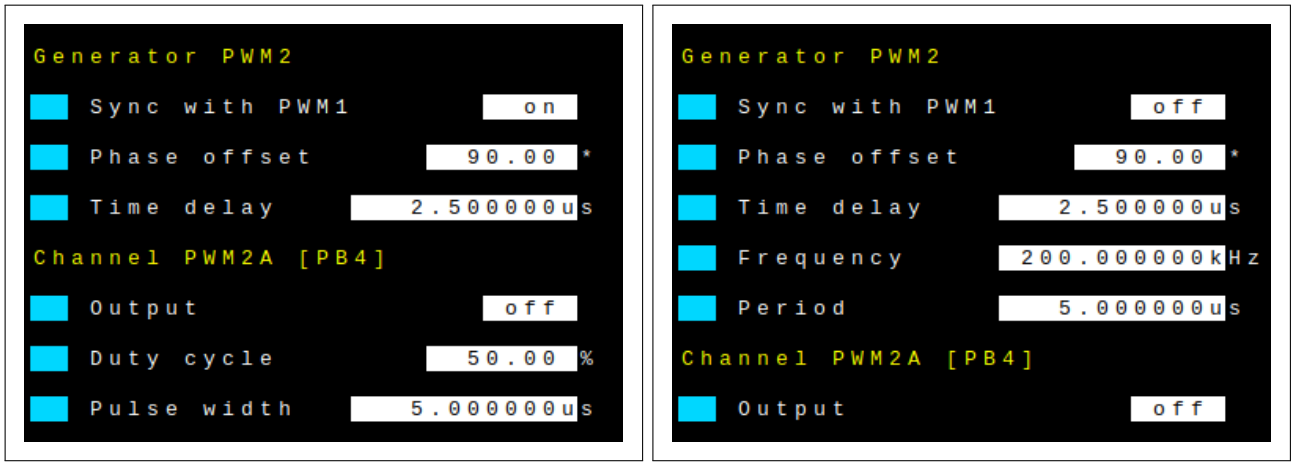

Figure 6.4: Pulse generator settings in TUI tab with and without synchronization enabled

The synchronization of pulse generators is implemented using the timer slave controllers. One of the capture/compare channels of the master generator is used as the TRGO signal which is then connected to the slave generator via the corresponding ITRx internal trigger connection as shown in Figure 6.3. The slave timer then uses the "trigger" slave mode to start on a rising edge of the trigger signal. Any number of generators can be synchronized this way. Perfect synchronization is achieved by subtracting the inherent resynchronization delay of 2 timer clock cycles from all time delay values.

6.2 Previous pulse generator project

My semestral project in 2021 involved the implementation of software-defined pulse generators with a terminal user interface. Originally, this firmware was implemented only for the STM32F103C8 microcontroller on the "Bluepill" development board. However, portability was still kept in mind and this firmware later became the basis of the firmware implemented in this work. The capabilities of the pulse generators have not changed significantly, but multiple issues were fixed and the user interface reworked for mouse-based interaction using Data Plotter.

This previous firmware implemented a standalone terminal user interface (shown in Figure 6.5) which could be used in any terminal emulator application. The terminal user interface was operated via the keyboard. The user first navigated to the parameter they wished to adjust on a 2D coordinate system using the W, A, S, D keys. Then, when the spacebar was pressed, a numeric value editor similar to the one described in section 4.2.3 was shown. This editor could likewise be operated using the W, A, S, D keys, confirming the selected value with another press of the spacebar.

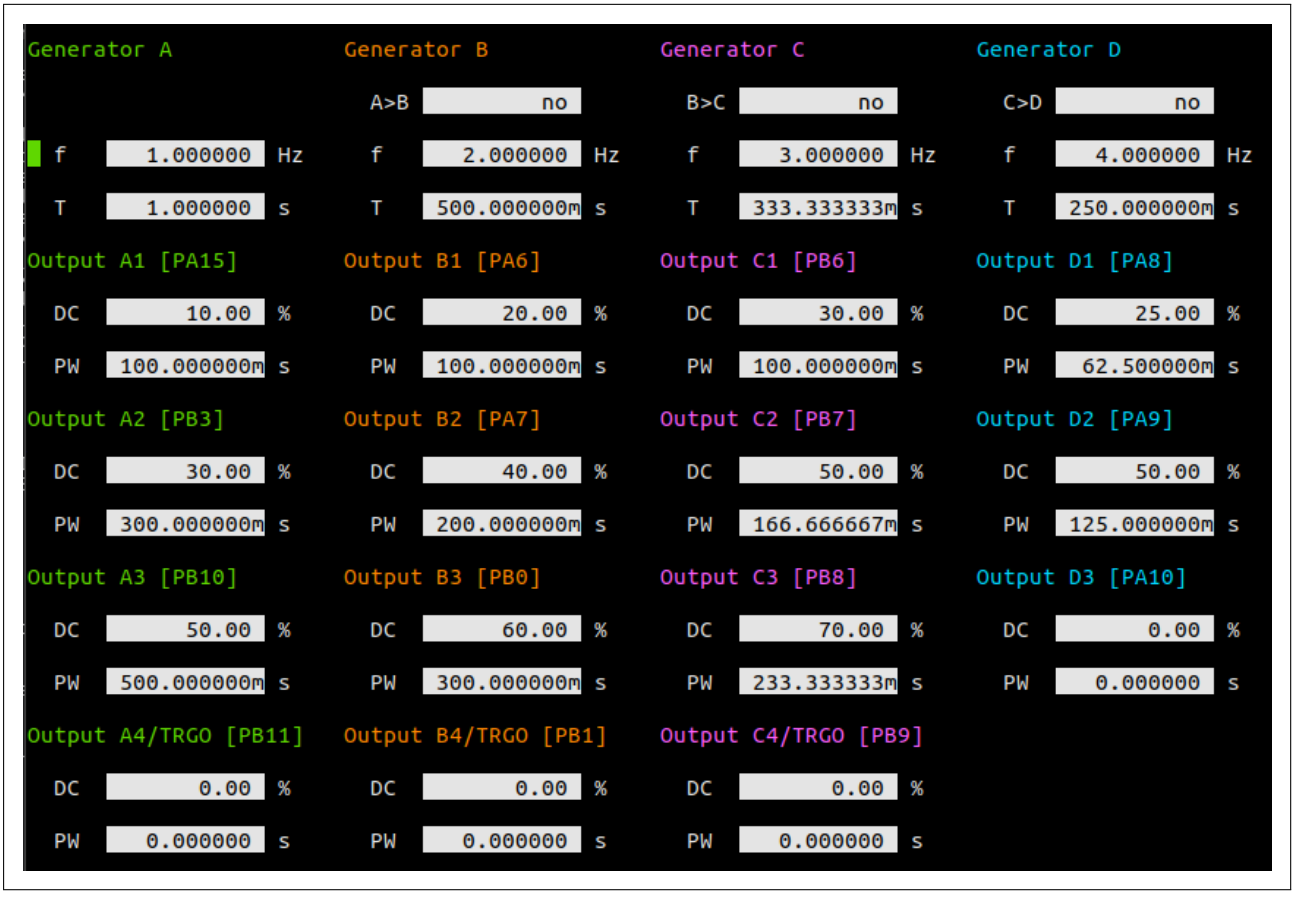

Figure 6.5: Pulse generator application and TUI developed in previous project

Chapter 7

Arbitrary generators

For some applications, it may be useful to generate signals other than just square waves (which can be generated by the pulse generators described in chapter 6). For example, it may be necessary to generate sine or triangle waves. Function generators are capable of generating these standard types of signals. Arbitrary generators can additionally generate atypical, custom waveforms defined by a set of samples, typically by using a digital-to-analog converter (DAC). An example of the kind of signals that can be generated by arbitrary generators is shown in Figure 7.1.

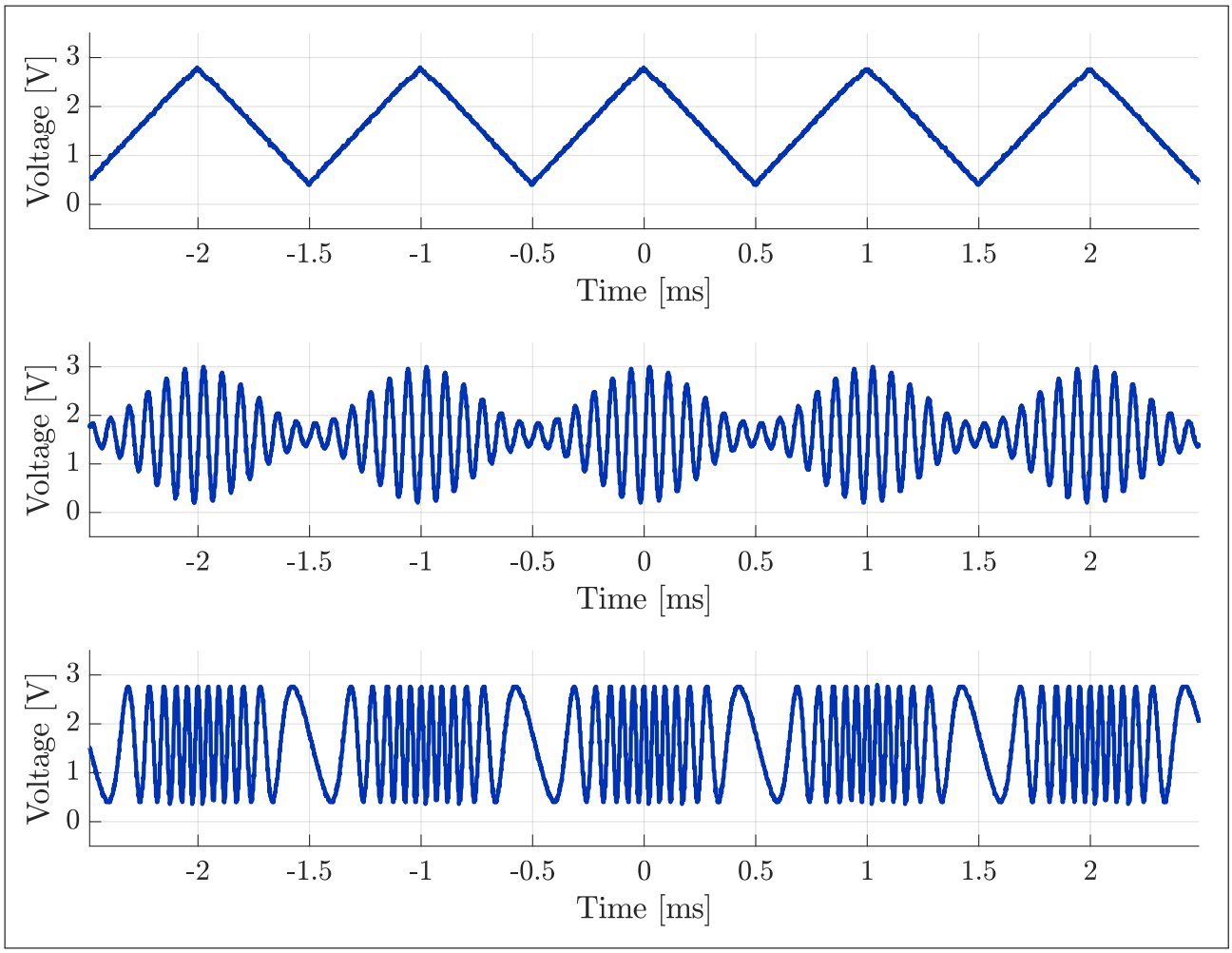

Figure 7.1: Examples of arbitrary generator output signals

The developed VSVI platform provides a pair of software-defined arbitrary generators for all MCUs with an embedded DAC. The arbitrary generators are denoted as "ARB 1" and "ARB 2". The corresponding terminal user interface tab can be seen in Figure 7.2. The frequency of all generated signals can be set precisely using the STM32 timers. The generated waveforms can be either:

- selected from a predefined set of functions:
  - sine wave ("sinusoid")
  - triangle wave with adjustable duty cycle ("sawtooth")
  - square wave with adjustable duty cycle ("square")
  - DC voltage ("DC only")
- fully custom, according to a wave file downloaded from the PC ("from PC"), see section 7.1

When using one of the predefined waveform functions, the output signal amplitude and DC offset voltages can be adjusted. Additionally, the phases of both generators' output signals can be set if synchronization is enabled (see section 7.2).

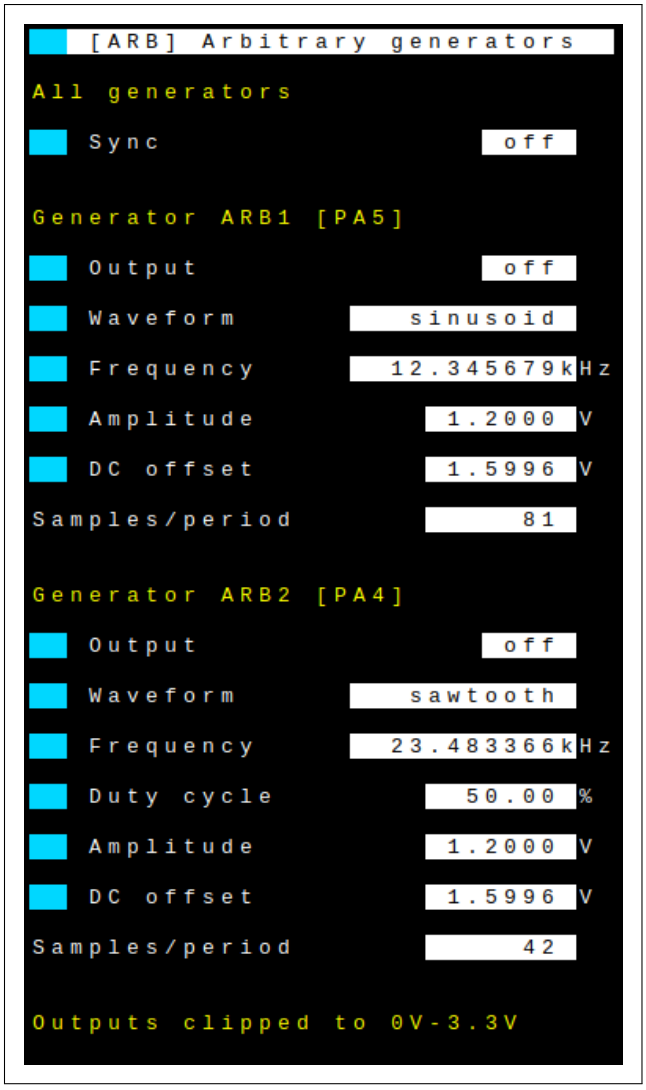

Figure 7.2: Arbitrary generators tab in terminal user interface

Figure 7.3 shows a block diagram of the implemented software-defined arbitrary generators. The generators use the two channels of the embedded DAC to convert digital samples into output signal voltages. The digital-to-analog conversions are clocked from hardware timers to ensure precise control of the sampling frequency – up to 1 Msps is achievable by the embedded DACs according to the MCU datasheets [23], [29], [39]. The sample data is transferred from circular buffers in SRAM memory to DAC registers using DMA transfers. It must be noted that this puts additional load on the DMA controllers and AHB bus, limiting the maximum sampling rates of the mixed-signal oscilloscope when arbitrary generators are enabled.

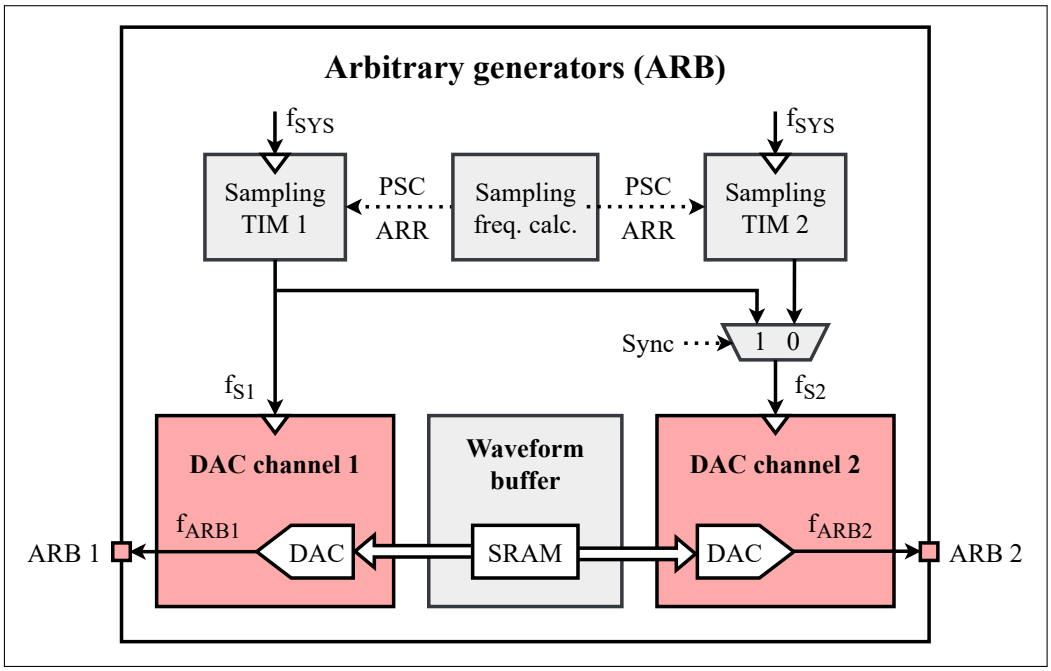

Figure 7.3: Block diagram of implemented arbitrary generators

Before the generators can be enabled, their circular SRAM memory buffers have to be populated with the samples representing one period of the signal to be generated. The minimum number of samples per period is set to

$$N_{SX,min} = 4, \tag{7.1}$$

the maximum to

$$N_{SX,max} = 1000. \tag{7.2}$$

This maximum amount of samples was deemed to be a suitable compromise between the precision of the waveform generation and the required size of the SRAM sample buffers. The number of samples per period N_{SX} used for a given output frequency f_{ARBX} is then

$$N_{SX} = \min\left(N_{SX,max}, \left\lfloor \frac{f_{SX,max}}{f_{ARBX}} \right\rfloor\right) = \min\left(1000, \left\lfloor \frac{1\ \text{MHz}}{f_{ARBX}} \right\rfloor\right) \tag{7.3}$$

where

$$f_{SX,max} = 1 \text{ MHz} \tag{7.4}$$

is the maximum sampling frequency of the DACs. The output frequency is limited to

$$f_{ARBX,max} = \left\lfloor \frac{f_{SX,max}}{N_{SX,min}} \right\rfloor = \left\lfloor \frac{1 \text{ MHz}}{4} \right\rfloor = 250 \text{ kHz}. \tag{7.5}$$

Once the SRAM buffers are filled with generated samples, the sampling timers are enabled. These are basic timers with no capture/compare channels or slave mode circuitry. After each DAC conversion, a DMA transfer loads the next sample from the circular buffer into the DAC data holding register to repeat the cycle. The DMA controller is set to circular buffer mode, looping back to the start of the buffer after each signal period has been generated.

7.1 Using custom waveforms with arbitrary generators

The arbitrary generators provide a number of built-in waveform functions (sine wave, sawtooth, etc.), but custom waveforms can also be used. In that case, only the frequency of the generated signal can be set. All other waveform parameters are unavailable as no samples are generated internally. Instead, the waveform must be loaded from a wave file downloaded via Data Plotter from the connected PC. Figure 7.4 shows the appearance of the TUI throughout this process. The supported wave file format is:

- A comma-separated list of numeric values (.csv).
- Each value represents a single sample of the output signal, in Volts.
- The length of a single value (excl. comma) cannot exceed 62 characters.
- Metric prefix characters (e.g. '123m' for 123 mV) are supported.
- Characters other than commas, digits, metric prefix characters and the decimal point are ignored.
- Values will be clipped to remain within the generator's output voltage range.
- If there are more values than can be used by the generator, they will be ignored.

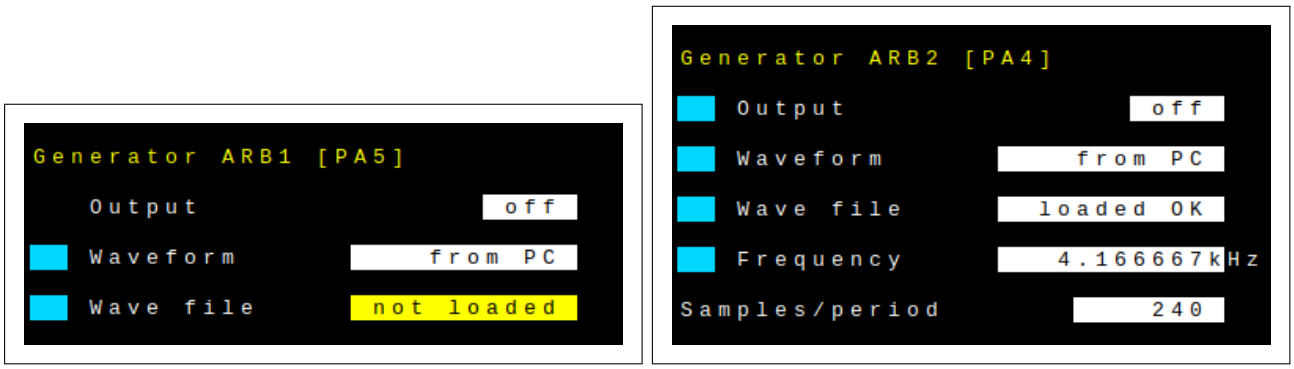

Figure 7.4: Arbitrary generator settings in TUI tab when using custom waveforms. States before and after a wave file is loaded from the PC shown on the left and right respectively.

7.2 Synchronization of arbitrary generators

The developed firmware also supports arbitrary generator synchronization. When enabled, the output signal frequency settings of both generators are replaced by a common frequency a common frequency setting at the top of the TUI tab, as shown in Figure 7.5. The sampling frequency and number of samples per period is also forced to be identical. This is due to the need to clock both DAC channels from same timer, as the two basic timers cannot be synchronized. The initial phase of both generators can be adjusted once synchronization is enabled. If custom waveforms are generated by both generators, the longer waveform will be truncated to the length of the shorter waveform.

Since the basic timers used by the arbitrary generators cannot be synchronized, it is not possible to synchronize arbitrary generators with pulse generators either. Whenever the user changes one of the pulse generator parameters, all pulse generators are stopped and restarted. Similarly, both arbitrary generators are stopped and restarted whenever one of their settings is changed. Every time this happens, a different phase offset is introduced between the pulse and arbitrary generators.

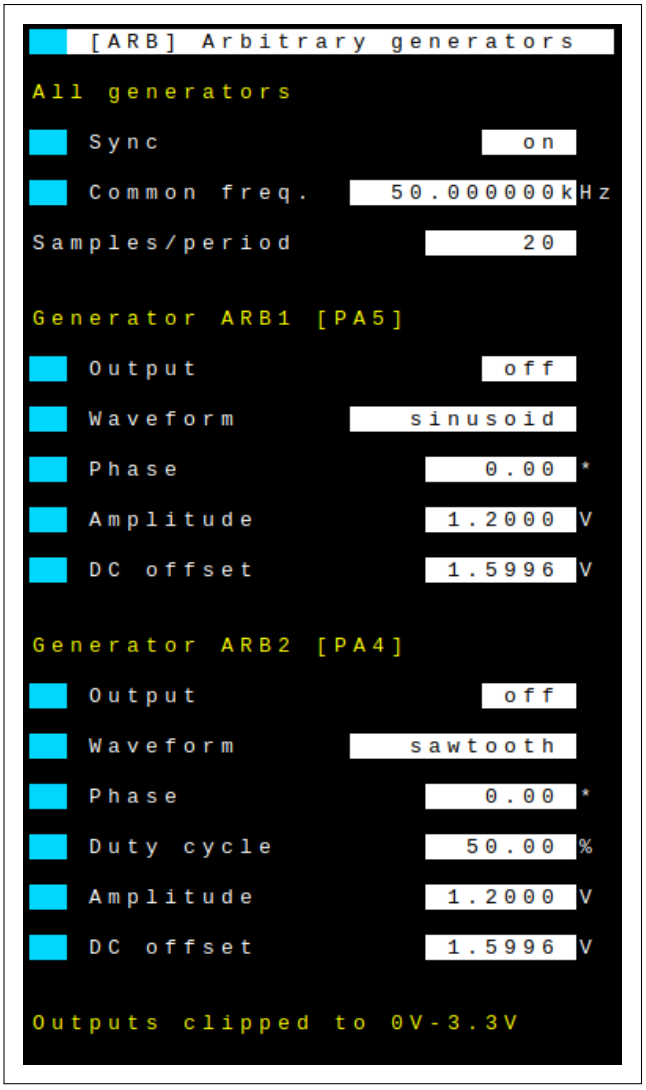

Figure 7.5: Arbitrary generator tab in terminal user interface when synchronization is enabled

7.3 Arbitrary generator output impedance and slew rate

It must be noted that the DACs embedded in STM32 microcontrollers exhibit high output impedances in the order of 15 kΩ. This makes the generated signals quite susceptible to electromagnetically induced noise and crosstalk, particularly from the sharp transitions of the pulse generator signals. It also limits the kinds of loads that can be connected to the output of the generators – any output current significantly lowers the output voltage. For example, the `LD2` green LED on Nucleo-F303RE boards is connected via the SB21 solder bridge to the output of the `ARB2` arbitrary generator (pin `PA5`)[27, p. 23]. When this connection is not removed, a significant difference between the two arbitrary generators' output signals can be seen when both are set to generate the same waveform, as demonstrated by Figure 7.6. The slew rate of the DAC outputs is also quite limited, as demonstrated in Figure 7.7. This makes them unsuitable for generating signals with sharp transitions such as square waves.

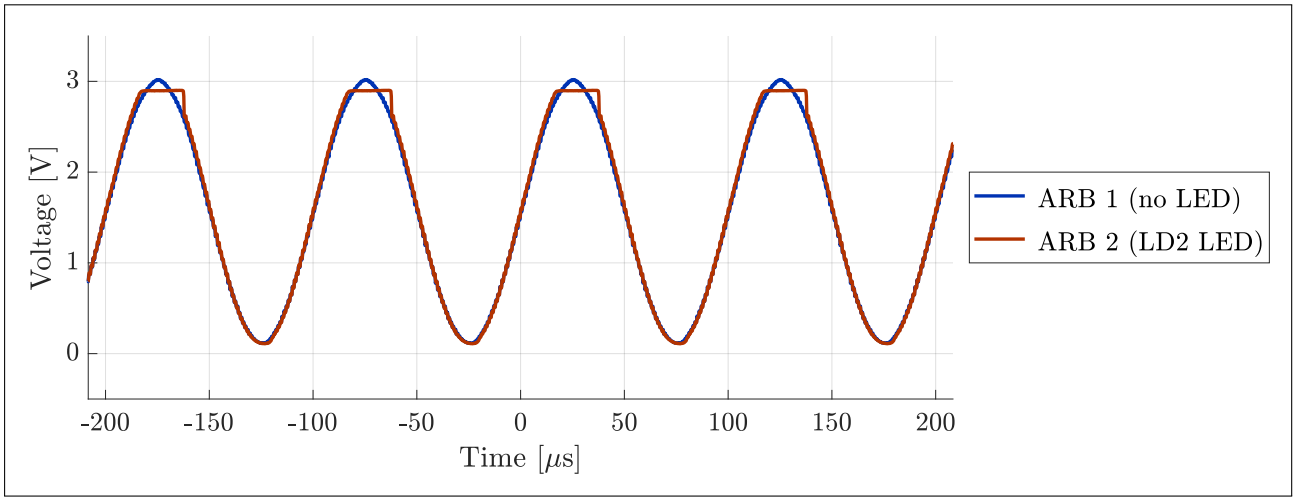

Figure 7.6: Output loading effect of the LD2 green LED on arbitrary generator (Nucleo-F303RE). The LED is connected to ground via a 510 Ω resistor [28]. Both generators set to generate the same signal, a 10kHz, 3V peak sine wave. Sampled by VSVI MSO in real-time sampling mode at 4.8 Msps.

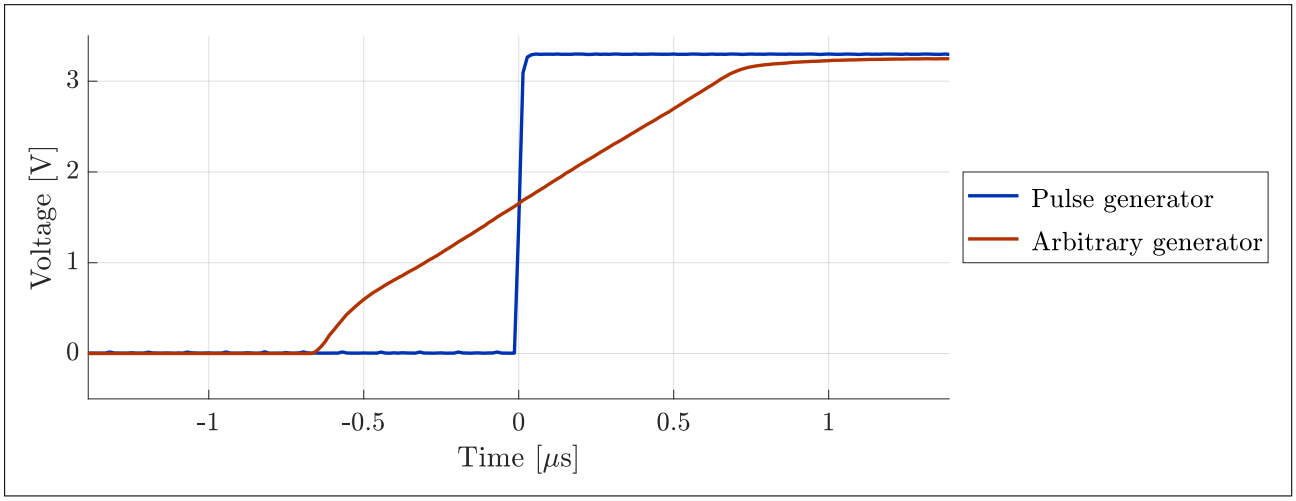

Figure 7.7: Comparison of pulse and arbitrary generator output signal slew rates (STM32F303RE). Both generators set to generate a 10kHz, 3.3V peak, 50% duty cycle square wave. Sampled by VSVI MSO in equivalent-time sampling mode at 72 Msps.

Chapter 8

Frequency counter

The final software-defined instrument implemented in this work is a frequency counter. This instrument can be used to measure the frequency or period of an external signal, for example clocks in digital circuitry. However, the primary motivation for its implementation was to allow the mixed-signal oscilloscope to use equivalent-time sampling with externally-generated signals. The frequency of the MSO's input signal is measured by the frequency counter and used by the MSO to calculate an appropriate real-time sampling frequency. Figure 8.1 shows the terminal user interface tab of the frequency counter. This is the only TUI tab with no user-configurable settings.

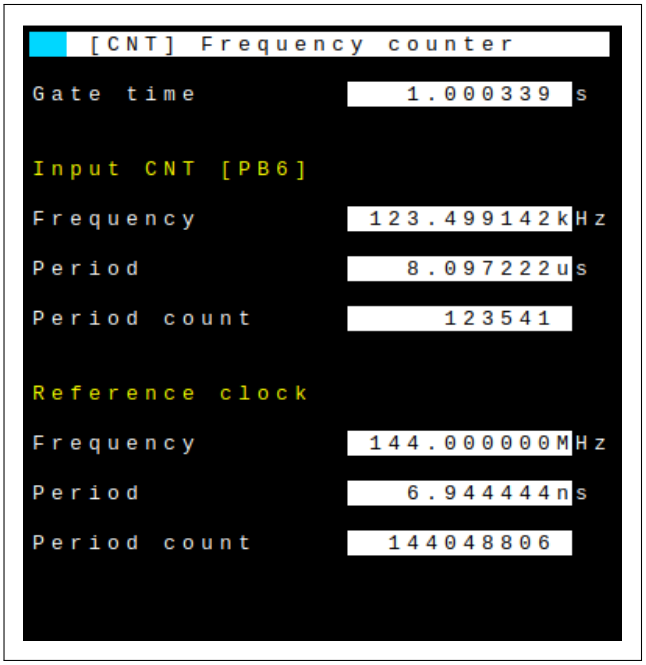

Figure 8.1: Frequency counter tab in terminal user interface

The input frequency is determined by the counter using a modified frequency ratio measurement. During the gate time t_{GATE}, the counter counts the number of input signal periods N_{INP} and the number of reference signal periods N_{REF}. The reference period frequency f_{REF} is known and typically significantly higher than the input frequency f_{INP} which can subsequently be determined by

$$f_{INP} = f_{REF} \cdot \frac{N_{INP}}{N_{REF}}. \tag{8.1}$$

However, since this naive implementation does not ensure that a whole number of input periods are counted, it will exhibit large quantization errors when the input signal period approaches the gate time. In order to surpass this limitation, the measurement method was slightly modified. When the nominal gate time has passed, the counter does not stop counting immediately but waits until the next input signal period is counted. This ensures that a whole number of input periods is counted at all times. The gate time is then

$$t_{GATE} = N_{INP} \cdot T_{INP} \approx N_{REF} \cdot T_{REF}, \tag{8.2}$$

where T_{INP}, T_{REF} are the input and reference periods respectively. The worst-case frequency resolution of a frequency counter is the reciprocal value of its gate time. It was determined that a resolution of at least 1 Hz was necessary, therefore the nominal gate time is set to

$$t_{GATE,nom} = 1 \text{ s}. \tag{8.3}$$

In order to further increase the counter resolution, the reference frequency is set as high as possible, i.e.

$$f_{REF} = f_{TIM}. \tag{8.4}$$

The frequency counter is implemented by an input timer working in conjunction with a reference timer, as illustrated in Figure 8.2. The counter input pin `CNT` is connected to an input capture channel of the input timer. The input timer's counter is clocked by rising edges of the input signal, using the "external trigger mode 1" of its slave mode controller. The reference timer is clocked from the internal timer clock. Figure 8.3 illustrates the operation the frequency counter firmware.

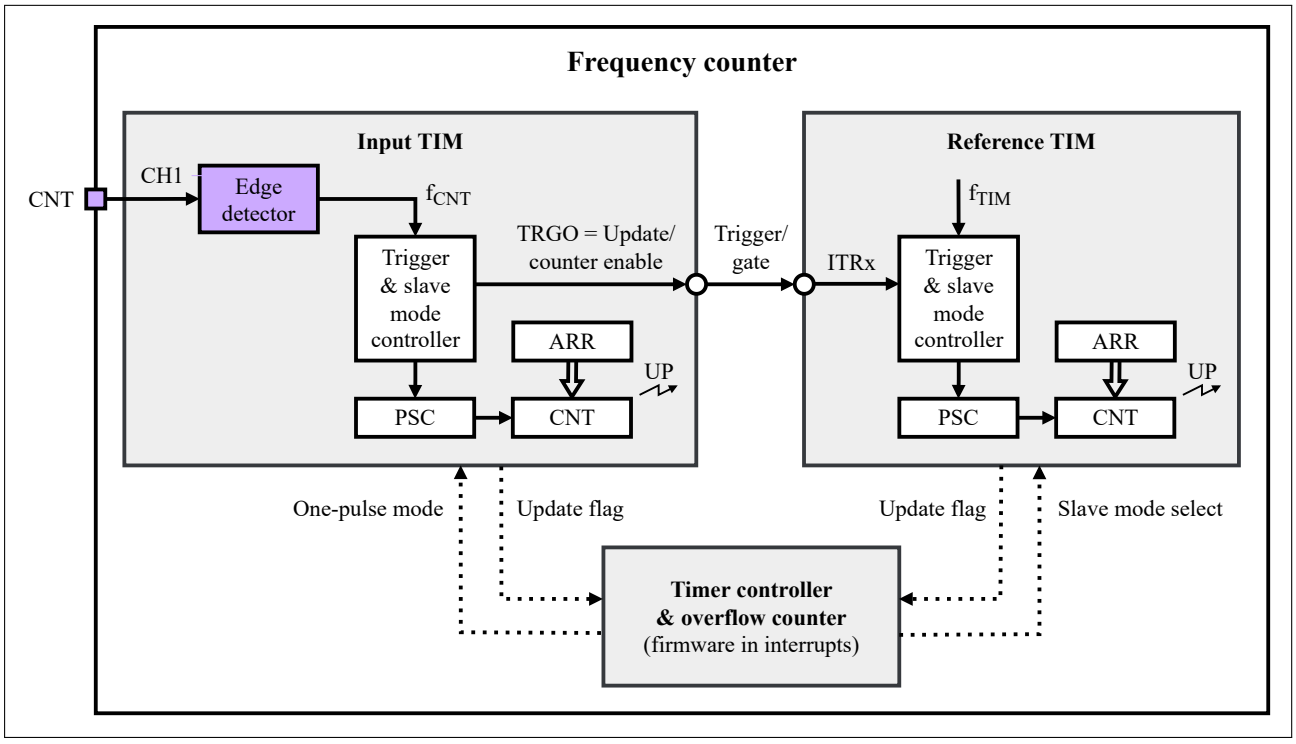

Figure 8.2: Block diagram of implemented frequency counter

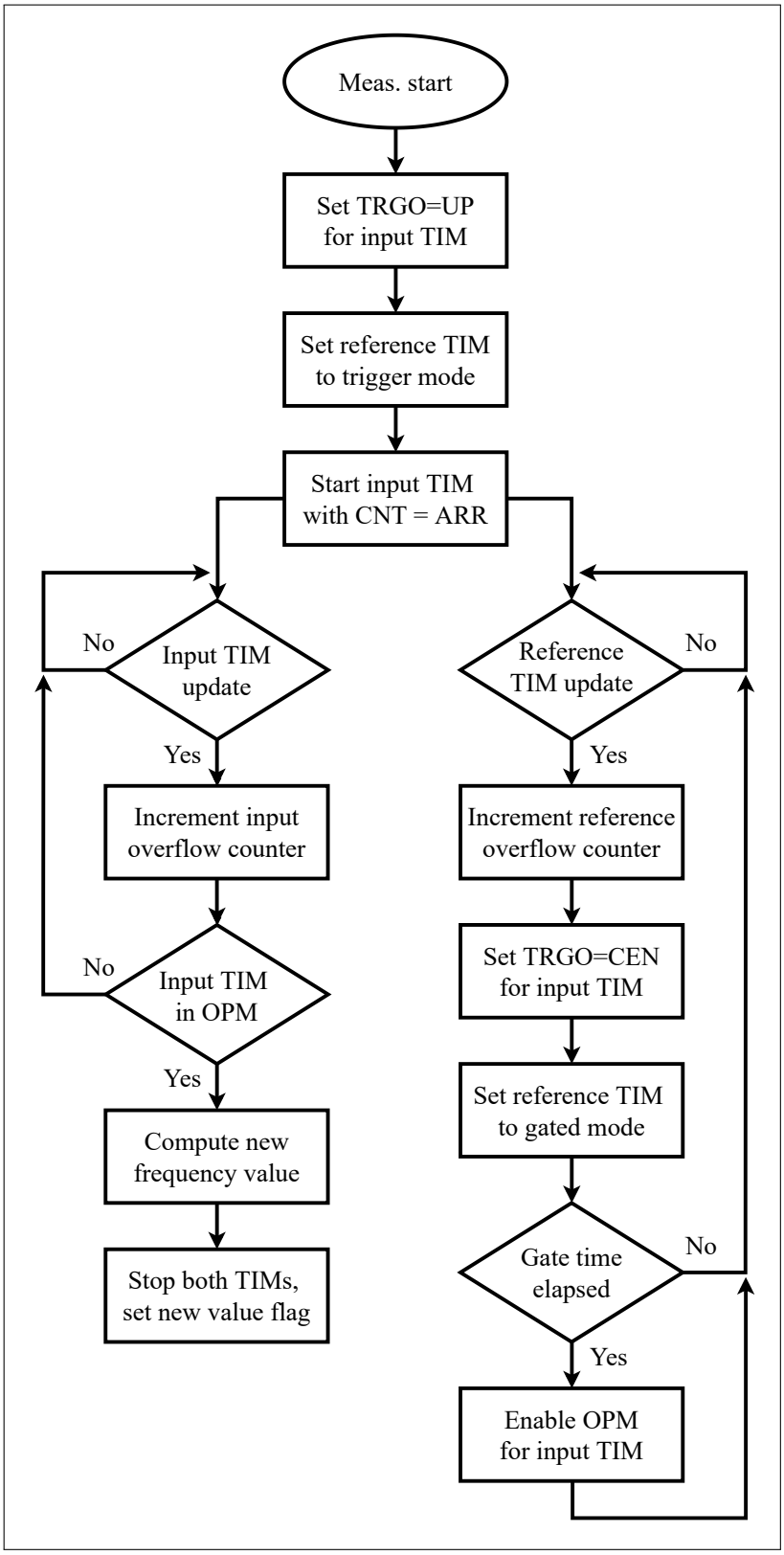

Figure 8.3: Frequency counter firmware algorithm diagram

If the input and reference periods were only counted using the 16-bit hardware timer counter registers, the maximum count would be 65535 periods. This would limit both the input and reference frequency to 65.5 kHz if the 1-second nominal gate time were used to yield a frequency resolution of 1 Hz or better. A workaround was implemented for this issue – both timers generate an interrupt when they overflow (update interrupt). The interrupt handlers then increment firmware-based overflow counters, thus removing any limits on the maximum number of periods counted.

Initially, the update event flag is set as the TRGO output of the input timer and the reference timer is in the trigger slave mode, connected to the input timer via the corresponding ITRx internal timer trigger connection. In order to start the reference timer immediately when the input timer counts the first input period, the input timer's counter is preset to the ARR value – generating an update event when it is incremented for the first time

When the first reference timer update interrupt occurs, the Counter Enable (CEN) bit as set as the TRGO signal of the input timer and the reference timer is switched into gated slave mode. That way, the reference timer will be stopped exactly when the input timer is disabled. The reference timer update interrupt handler further checks whether the nominal gate time has elapsed (based on the value in the firmware overflow counter). If it has, the input timer is put into One-Pulse Mode (OPM) and its ARR value is changed to be the slightly above the current value of the CNT register. This causes the input timer to be disabled after a few input periods, stopping the reference timer at the same time. It also generates an update interrupt, whereupon the interrupt handler calculates the newly measured input frequency according to eq. 8.2. This concludes the measurement cycle.

Chapter 9

Instrument configuration profiles

The developed VSVI platform provides a large number of user-configurable parameters for the implemented software-defined instruments. Normally, all of these parameters would be reset to default values after an MCU reset. The user would be required to reconfigure them all every time they wish to use the instruments, which is rather inconvenient. It would be useful to implement an option to store these settings such that they persist between MCU resets. To facilitate this, so-called "configuration profiles" were implemented.

Each profile contains all the user-configurable parameters of every implemented virtual instrument, The current values can be stored in a profile, while previously stored values can be recalled from one. Profiles can be stored either in the MCU Flash memory or in the connected PC using Data Plotter. Figure 9.1 depicts the TUI tab used to manage configuration profiles. A profile is being loaded from the PC, showing a file selection dialog.

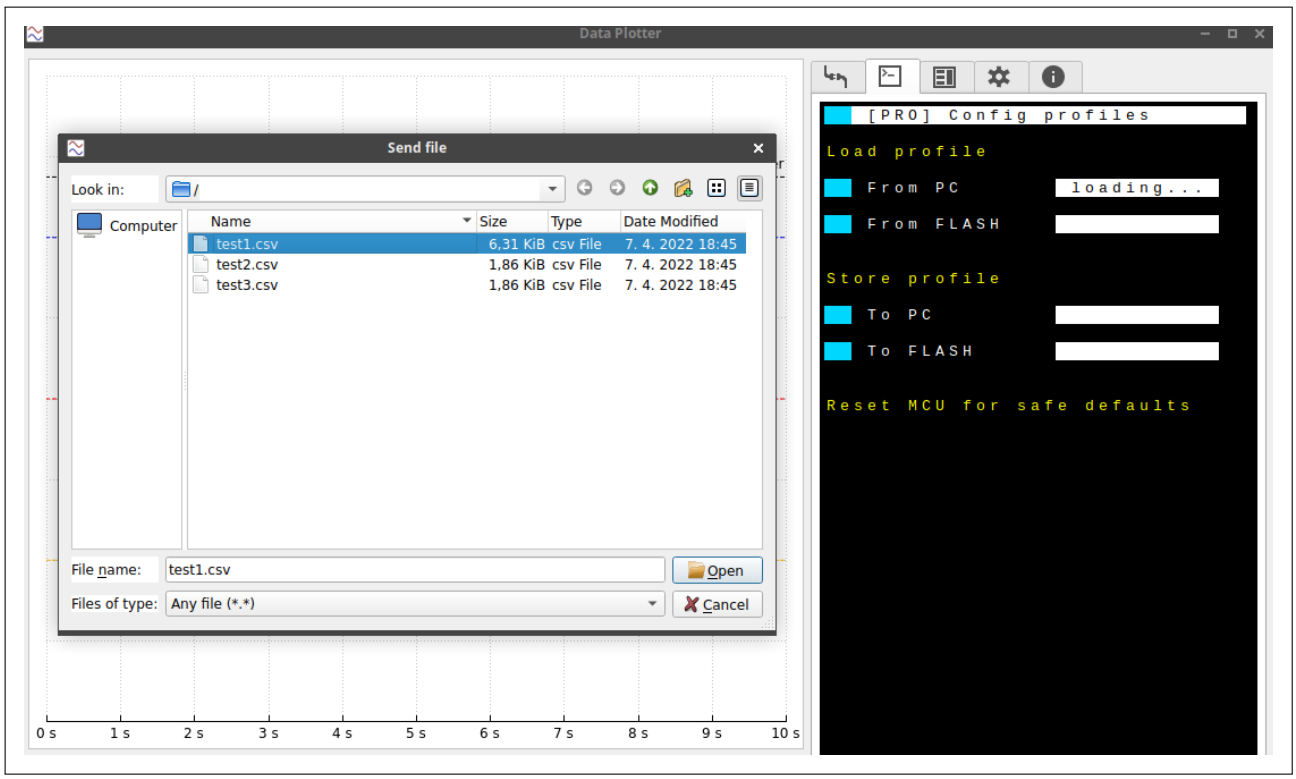

Figure 9.1: Configuration profiles TUI tab with open file selection dialog

The file transfer protocol implemented by Data Plotter is used to store configuration profiles to the connected PC. This protocol does not support binary data, therefore the profiles must be stored as text files. The comma-separated value (.csv) format was chosen for its simplicity. All values are ASCII text strings. The delimiter character is a comma (","). The format of a configuration profile file is as follows:

- Start of profile character ("#")
- Firmware version descriptor, e.g. "G431-LQFP32_R20220416"
- For each type of instrument:
  - Start of instrument type character (">")
  - Instrument code, i.e. "MSO", "PWM" or "ARB"
  - For each parameter:
    - Instrument instance identifier, e.g. "2" for "PWM 2"
    - Parameter code, see Tables 9.1 through 9.3
    - Channel instance identifier, e.g. "A" for "PWM 2A"
    - Parameter value, converted to a text string

The inclusion of the firmware version descriptor ensures compatibility – the firmware will reject configuration profiles created in another firmware version.

Code	Parameter
A	Number of enabled analog channels
D	Number of enabled digital channels
N	Record length (number of samples)
P	Trigger position (pre-trigger)
T	Sampling mode (ETS or RTS)
S	Input signal frequency source
I	Input signal frequency
R	Real-time sampling frequency
E	Equivalent-time sampling frequency
C	Trigger source channel
G	Active trigger edge
V	Trigger level (voltage)
M	Trigger mode (auto or normal)

Table 9.1: Mixed-signal oscilloscope configuration profile parameters

Code	Parameter
S	Generator synchronization
F	Generator frequency
R	Generator period
E	Channel output enable
D	Channel duty cycle
P	Channel pulse width

Table 9.2: Pulse generator configuration profile parameters

Code	Parameter
E	Generator enable
W	Waveform type
A	Amplitude (voltage)
C	DC offset (voltage)
F	Frequency
D	Duty cycle
P	Initial phase
N	Number of samples (per period)

Table 9.3: Arbitrary generator configuration profile parameters

If an arbitrary generator is using a custom waveform (loaded from the connected PC), its samples are also stored in the configuration profile. Each 12-bit sample value is converted into two ASCII characters using Base64 encoding. Therefore, including the comma separator, 3 bytes are used to store every sample in the profile. In the worst-case scenario where both arbitrary generators use custom waveforms with 1000 samples, the size of the profile will thus be above 6 kB. As such, only one profile can be stored into the MCU Flash memory. The stringified arbitrary generator sample values are stored in the profile after the "number of samples" parameter of each arbitrary generator.

Chapter 10

Versions of developed SDI platform (supported MCUs)

10.1 STM32F303RE version of developed SDI platform

The primary adaptation of the developed VSVI platform is for the Nucleo-F303RE development board shown in Figure 10.1. This board is based on the STM32F303RE microcontroller in the LQFP64 package and is one of the most widely used STM32 platforms at FEE CTU. It includes an ST-Link V2-1 debugger/programmer which can be used to program the MCU using the onboard Mini-USB connector. The ST-Link additionally serves as a USB-UART converter, being connected to the USART2 interface of the STM32F303RE MCU. The ST-Link clock derived from an onboard 8 MHz crystal oscillator is also used as the clock source for the STM32F303RE MCU in HSE bypass mode. Figure 10.2 shows the pinout of this version of the VSVI platform.

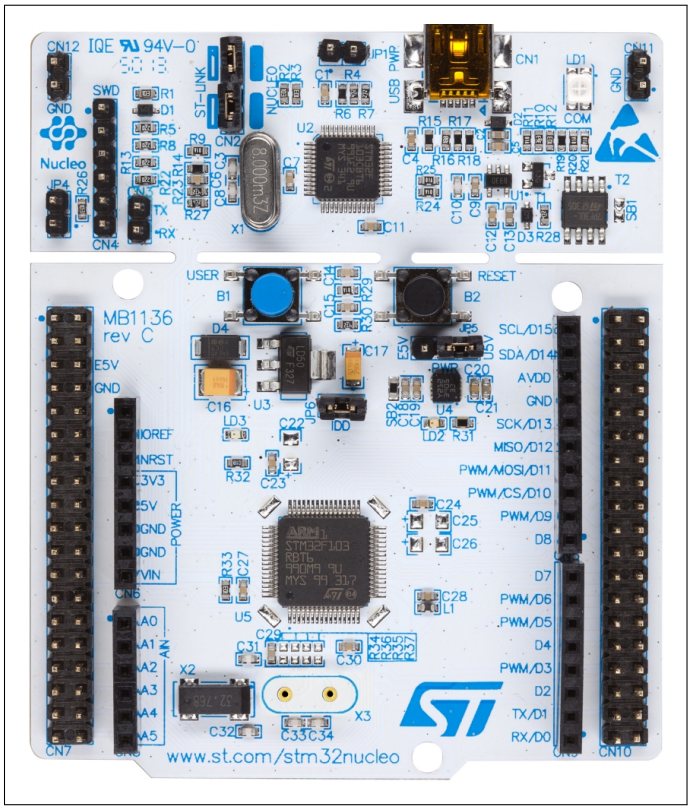

Figure 10.1: Nucleo-F303RE development board with STM32F303RE MCU. Adapted from [25].

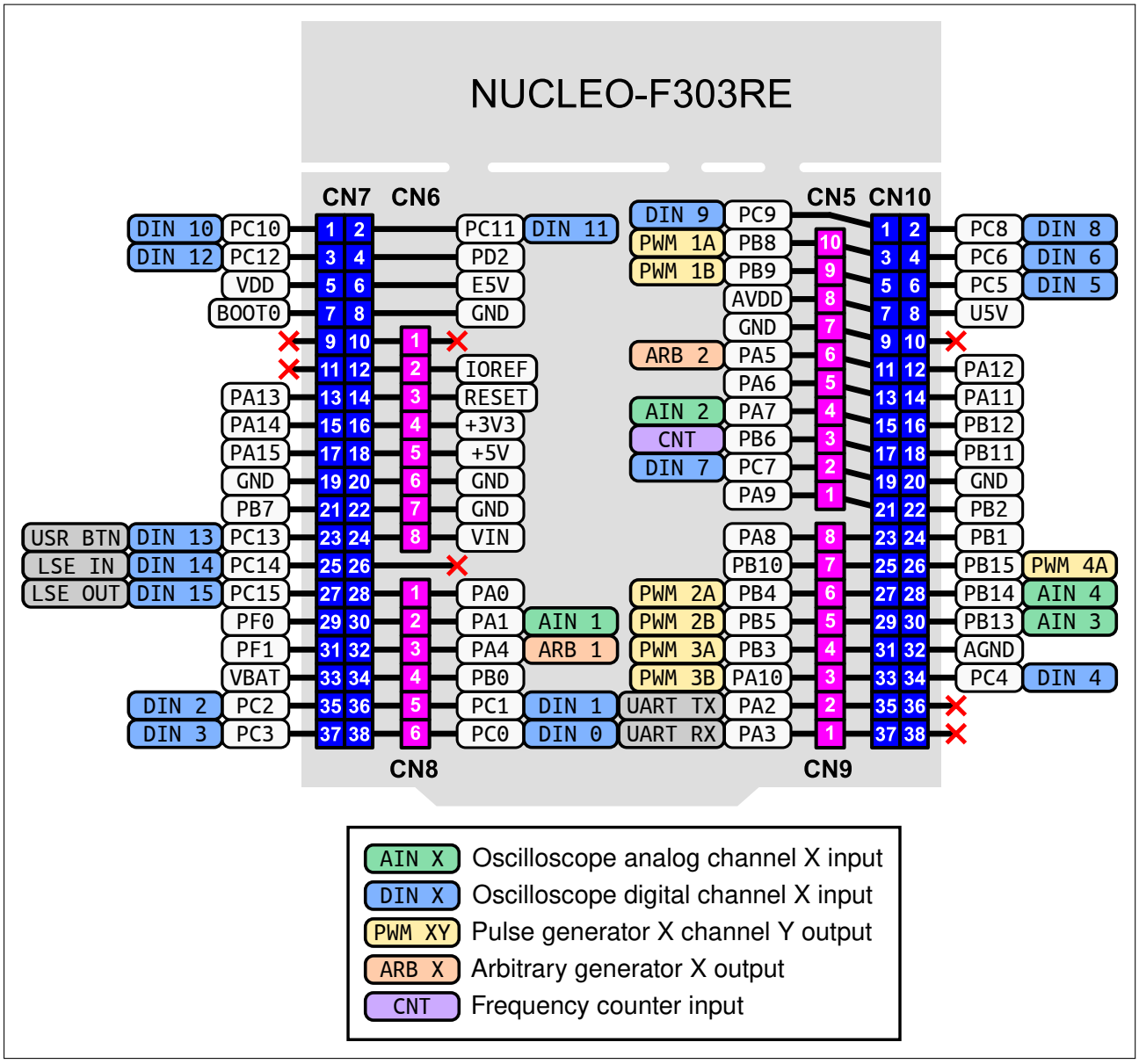

Figure 10.2: Pinout of SDIs developed for STM32F303RE (Nucleo-F303RE development board)

10.1.1 Mixed-signal oscilloscope (STM32F303RE)

- **4 analog channels** (inputs `AIN 1, AIN 2, AIN 3, AIN 4`) with **12-bit resolution**
- **16 digital channels** (inputs `DIN 0 - DIN 15`)
- Maximum real-time sampling rate:
  - **24 MSps** for all digital channels with analog channels disabled
  - **18 MSps** for one analog channel with 4 ADCs interleaving, digital channels disabled
  - **5 MSps** for all digital and analog channels with no interleaving

- Table 10.1 shows all the possible channel configurations and their respective max. sampling rates when all arbitrary generators are disabled. When any of the arbitrary generators is enabled, Table 10.2 applies instead.

- Maximum equivalent-time sampling rate:
  - Depends on the input signal frequency
  - Typically up to 72 MSps, max. 720 MSps with 1 sample per 10 input signal periods

Digital \ Analog	Disabled	1 channel	2 channels	4 channels
Disabled	-	18.0 MSps (i4) 10.3 MSps (i2) 5.1 MSps (-)	10.3 MSps (i2) 5.1 MSps (-)	5.1 MSps (-)
16 channels	24 MSps (-)	10.3 MSps (i2) 5.1 MSps (-)	10.3 MSps (i2) 5.1 MSps (-)	4.8 MSps (-)

Table 10.1: Maximum MSO sampling rates for STM32F303RE with arbitrary generators disabled. Used ADC interleaving mode indicated in parentheses ("-" = none, "i2" = 2 ADCs/channel, "i4" = 4 ADCs/channel).

Digital \ Analog	Disabled	1 channel	2 channels	4 channels
Disabled	-	7.2 MSps (i2) 4.8 MSps (-)	7.2 MSps (i2) 4.8 MSps (-)	4.0 MSps (-)
16 channels	8 MSps (-)	5.5 MSps (i2) 4 MSps (-)	5.5 MSps (i2) 4 MSps (-)	3.4 MSps (-)

Table 10.2: Maximum MSO sampling rates for STM32F303RE with arbitrary generators enabled. Used ADC interleaving mode indicated in parentheses ("-" = none, "i2" = 2 ADCs/channel).

- Maximum record length depends on the number of enabled channels according to Table 10.3

Digital \ Analog	Disabled	1 channel	2 channels	4 channels
Disabled	-	30.8 kSa	15.4 kSa	7.7 kSa
16 channels	30.8 kSa	15.4 kSa	10.2 KSa	6.1 kSa

Table 10.3: Maximum MSO record lengths for STM32F303RE

- All RTS and ETS configurations, including interleaved sampling modes, were tested as shown in Figures 10.3 through 10.6.

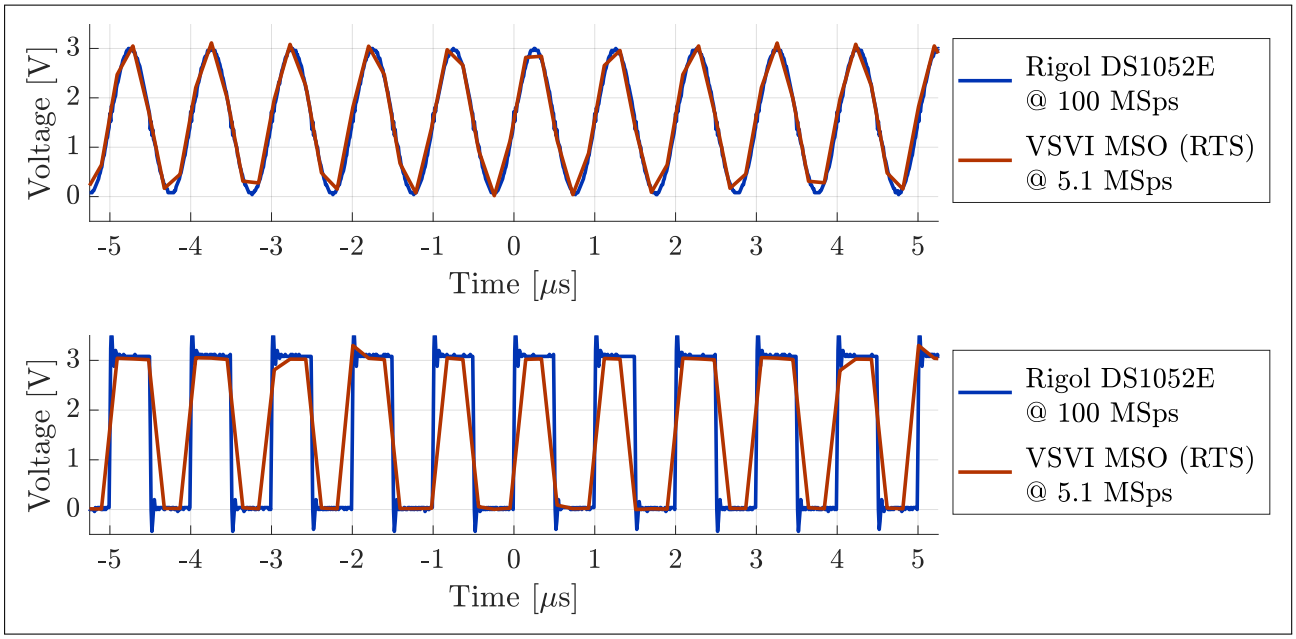

Figure 10.3: Waveforms captured by STM32F303RE oscilloscope in RTS mode at 5.1 MSps. The test signals were a 3V peak sine wave (top) and 50% duty cycle square wave (bottom) with a frequency of 1 MHz, generated by the Tektronix AFG3102 function generator. They were captured by both the VSVI MSO and the Rigol DS1052E oscilloscope.

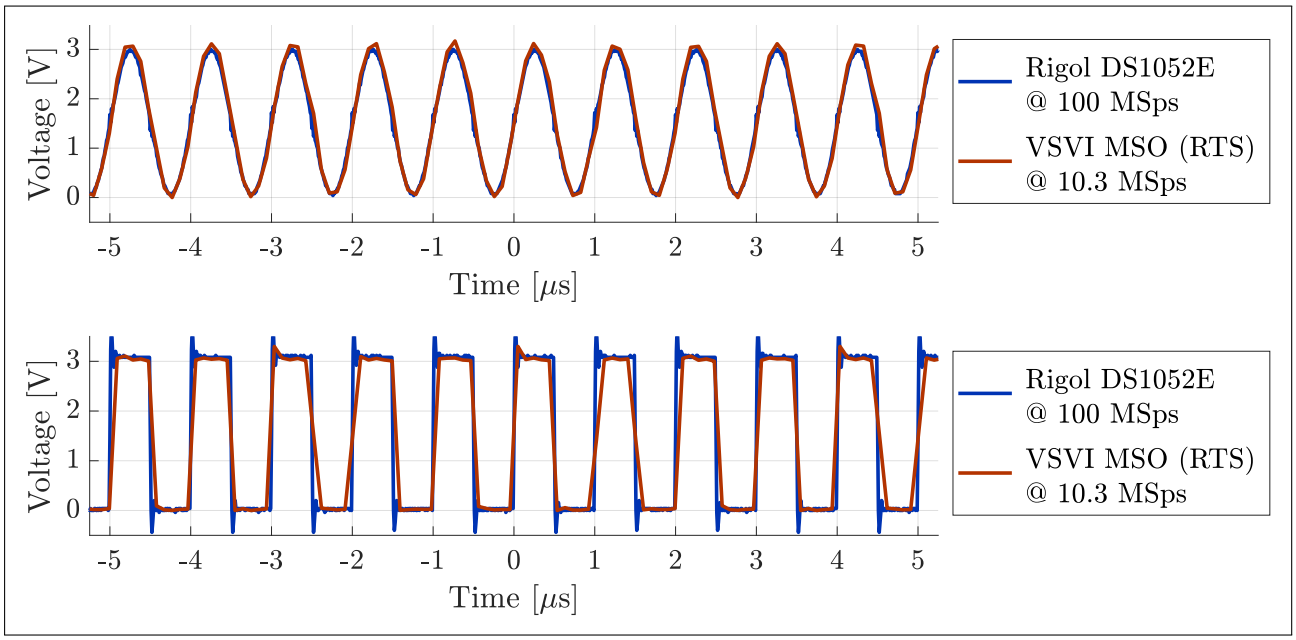

Figure 10.4: Waveforms captured by STM32F303RE oscilloscope in RTS mode at 10.3 MSps (2 ADCs interleaved). The test signals were a 3V peak sine wave (top) and 50% duty cycle square wave (bottom) with a frequency of 1 MHz, generated by the Tektronix AFG3102 function generator. They were captured by both the VSVI MSO and the Rigol DS1052E oscilloscope.

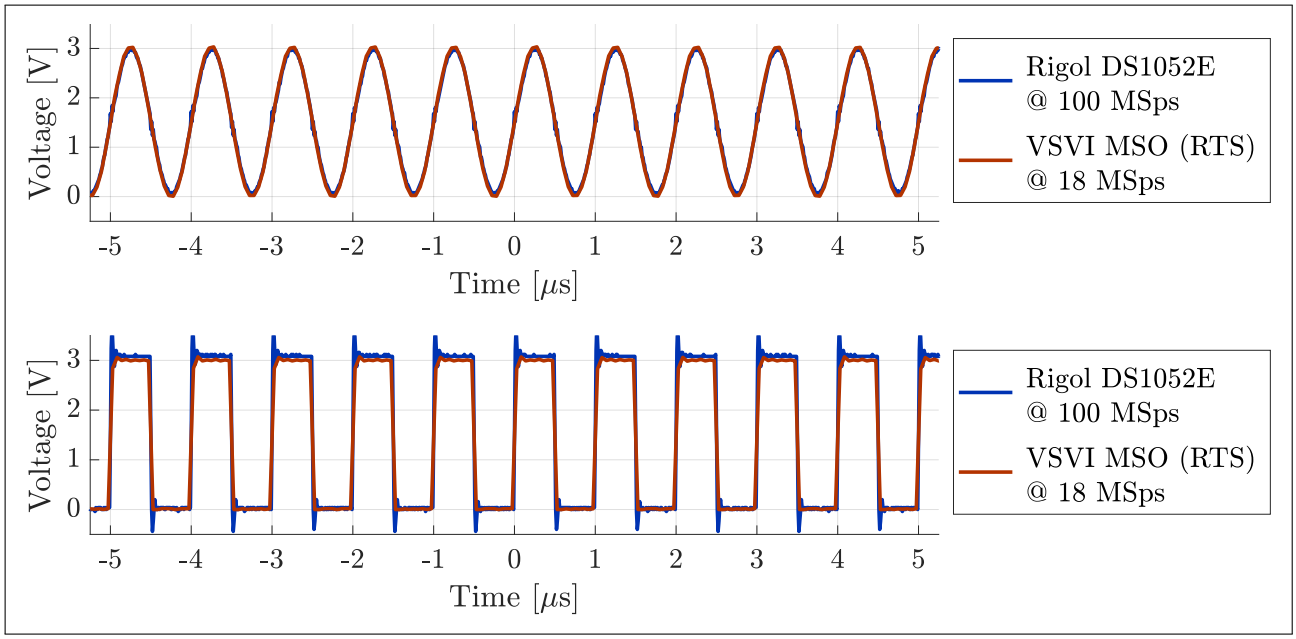

Figure 10.5: Waveforms captured by STM32F303RE oscilloscope in RTS mode at 18 MSps (4 ADCs interleaved). The test signals were a 3V peak sine wave (top) and 50% duty cycle square wave (bottom) with a frequency of 1 MHz, generated by the Tektronix AFG3102 function generator. They were captured by both the VSVI MSO and the Rigol DS1052E oscilloscope.

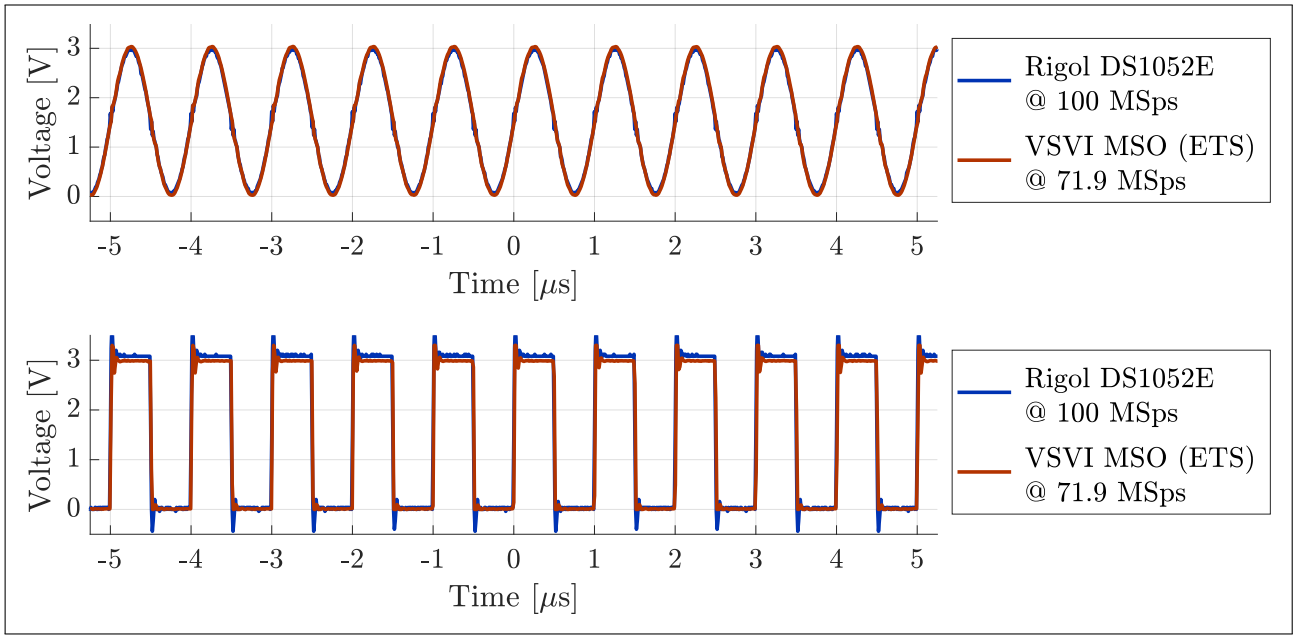

Figure 10.6: Waveforms captured by STM32F303RE oscilloscope in ETS mode at 71.9 MSps (equivalent). The test signals were a 3V peak sine wave (top) and 50% duty cycle square wave (bottom) with a frequency of 1 MHz, generated by the Tektronix AFG3102 function generator. They were captured by both the VSVI MSO and the Rigol DS1052E oscilloscope. The VSVI frequency counter measurement was used as the input signal frequency for ETS mode.

- **When interleaved sampling of the analog channels is used, the user must connect external jumper links between analog input pins:**
  - When 1 analog channel is enabled and interleaving 2 ADCs per channel, connect:
    - `AIN 1` (PA1) and `AIN 3` (PB13)
  - When 2 analog channels are enabled and interleaving 2 ADCs per channel, connect:
    - `AIN 1` (PA1) and `AIN 3` (PB13)
    - `AIN 2` (PA7) and `AIN 4` (PB14)
  - When 1 analog channel is enabled and interleaving 4 ADCs per channel, connect:
    - `AIN 1` (PA1) and `AIN 3` (PB13)
    - `AIN 2` (PA7) and `AIN 4` (PB14)
    - `AIN 3` (PA13) and `AIN 4` (PB14) or `AIN 1` (PA1) and `AIN 2` (PA7)
- **Some digital channels are limited on stock Nucleo-F303 boards:**
  - `DIN 13` (PC13) is connected to the USER push button, including a 4.7 kΩ pull-up resistor to the 3.3V rail and a bypass capacitor. External signals should not be connected to this pin unless the solder bridge SB17 is removed, disconnecting the button circuitry.
  - `DIN 14` and `DIN 15` are not available. The corresponding MCU pins PC14 and PC15 are connected to the 32 kHz LSE crystal oscillator and NOT connected to the pin headers. To use these digital channels, R34 and R36 must be removed (disconnecting the crystal) and SB48 and SB49 must be connected (connecting the MCU pins to the pin headers).

10.1.2 Pulse generators (STM32F303RE)

- **4 generators** available:
  - `PWM 1` with two channels (`PWM 1A, PWM 1B`)
  - `PWM 2` with two channels (`PWM 2A, PWM 2B`)
  - `PWM 3` with two channels (`PWM 3A, PWM 3B`)
  - `PWM 4` with one channel (`PWM 4A`)
- All can run independently or be synchronized with adjustable phase/time delay
- Maximum output signal frequency 72 MHz (50% duty cycle only)
- The output signals were validated and the frequency characteristic measured, as shown in Figures 10.7 and 10.8 respectively.

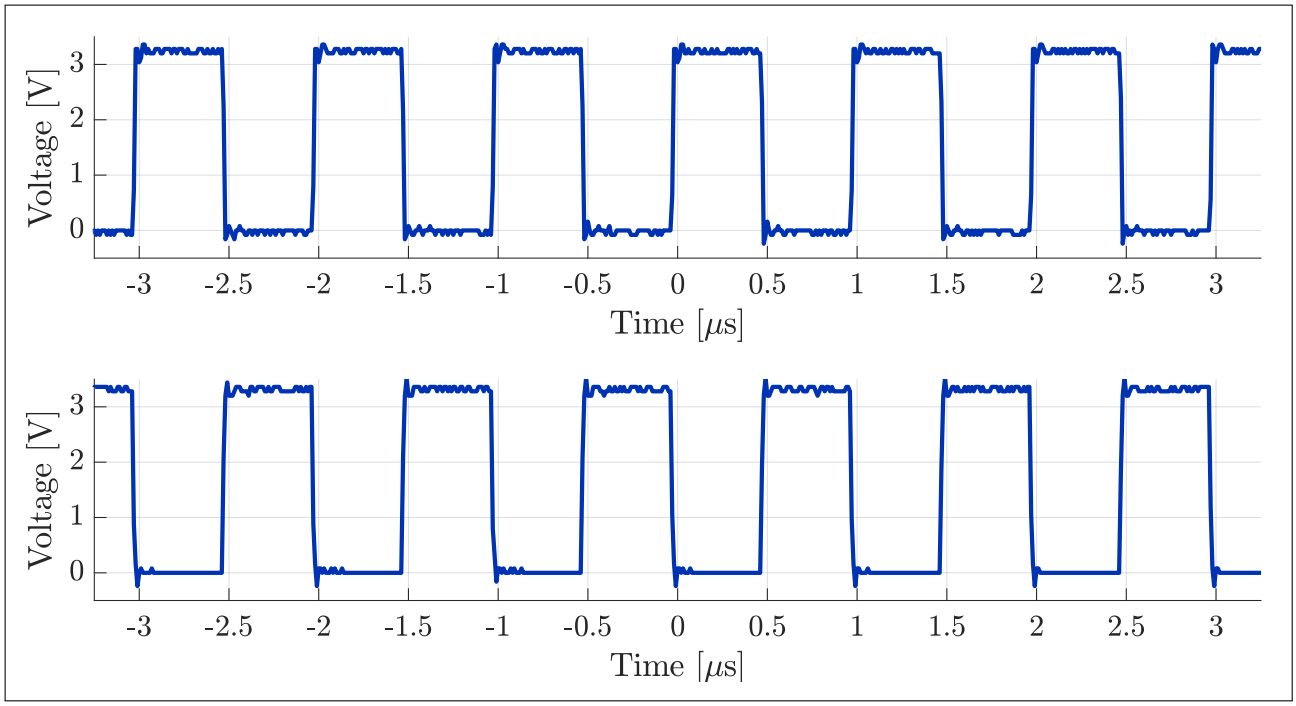

Figure 10.7: Output signals of STM32F303RE pulse generators. The test signals were 50% duty cycle, 3.3V peak square waves with frequencies of 1 MHz and 180° phase offset generated by the VSVI pulse generators. The waveforms were recorded by the Rigol DS1052E oscilloscope sampling at 100 MSps.

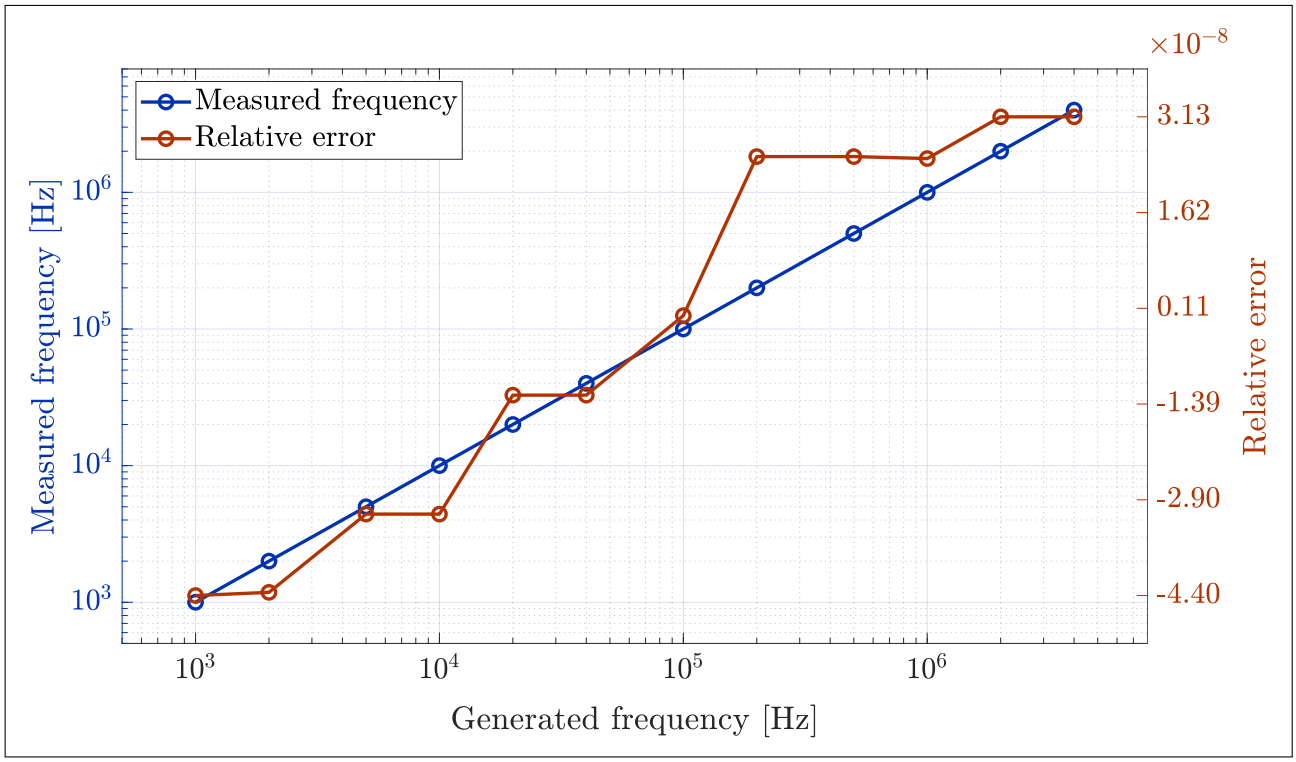

Figure 10.8: Measured frequency characteristic of STM32F303RE pulse generators. The test signals were a 50% duty cycle, 3.3V peak square waves generated by the VSVI pulse generators. Their frequencies were measured from waveforms recorded by the Rigol DS1052E oscilloscope.

10.1.3 Arbitrary generators (STM32F303RE)

- **2 generators** available:
  - `ARB 1`
  - `ARB 2`
- Both can run independently or be synchronized with adjustable phases
- Maximum sampling frequency 1 MHz
- Maximum output signal frequency 250 kHz (4 samples per period)
- Up to 1000 samples per period for output signal frequencies ≤ 1 kHz
- The output signals were validated as shown in Figures 10.9 and 10.10.

10.1.4 Frequency counter (STM32F303RE)

- Input signal frequency range: 10 Hz – 72 MHz
- Gate time ≈ 1 s gives update rate ≈ 1 Hz
- Frequency resolution ≤ 1 Hz using modified frequency ratio measurement (see chapter 8)
- Measured frequency characteristic shown in Figure 10.11

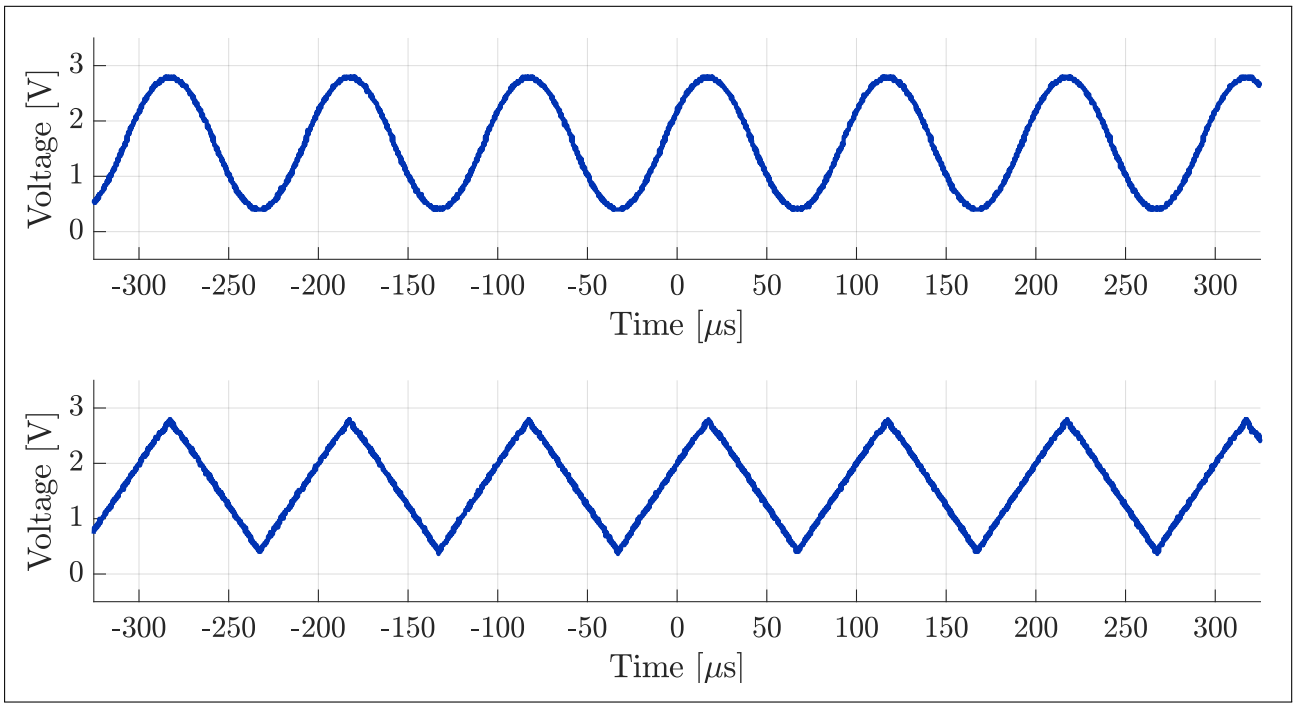

Figure 10.9: Output signals of STM32F303RE arbitrary generators. The test signals were sine (top) and triangle (bottom) waves with frequencies of 10 kHz and 0° phase offset generated by the VSVI arbitrary generators. Their amplitude was set to 1.2 V with a DC offset of 1.6 V. The waveforms were recorded by the Rigol DS1052E oscilloscope sampling at 10 MSps.

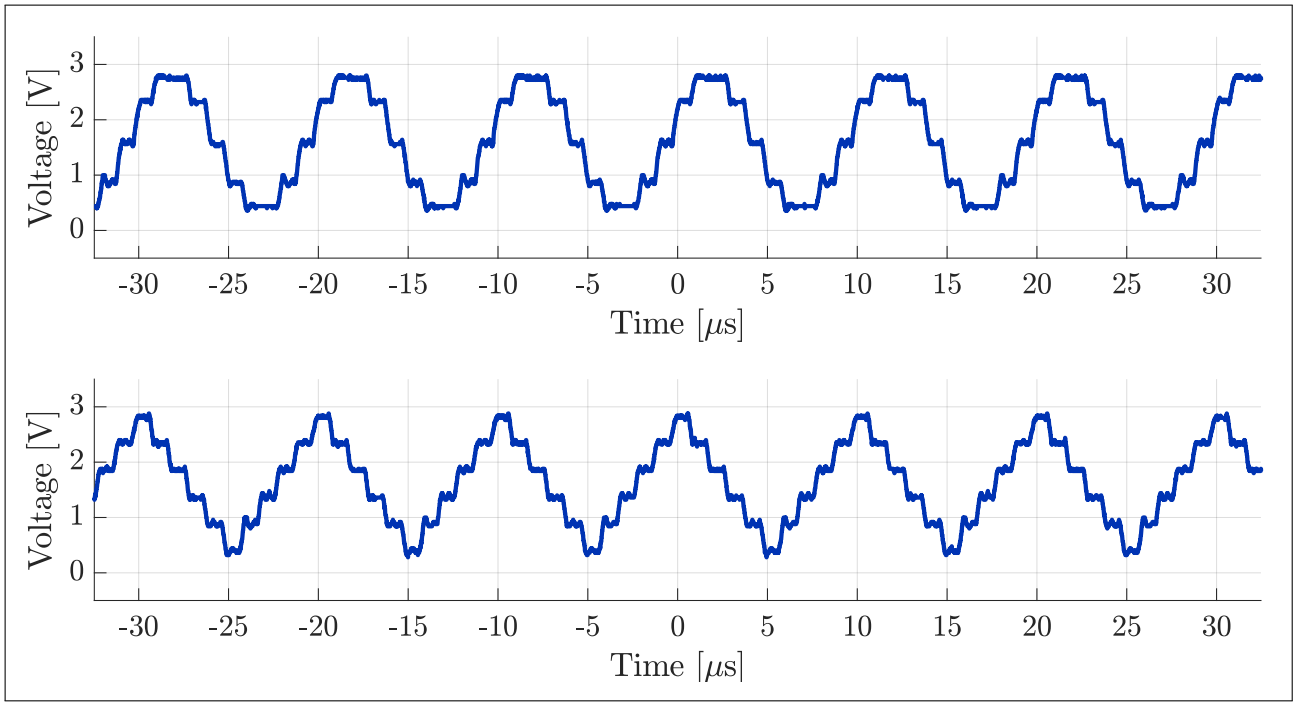

Figure 10.10: Output signals of STM32F303RE arbitrary generators. The test signals were sine (top) and triangle (bottom) waves with frequencies of 100 kHz generated by the VSVI arbitrary generators. Their amplitude was set to 1.2 V with a DC offset of 1.6 V. The synchronization was disabled, giving a random phase offset. The waveforms were recorded by the Rigol DS1052E oscilloscope sampling at 100 MSps.

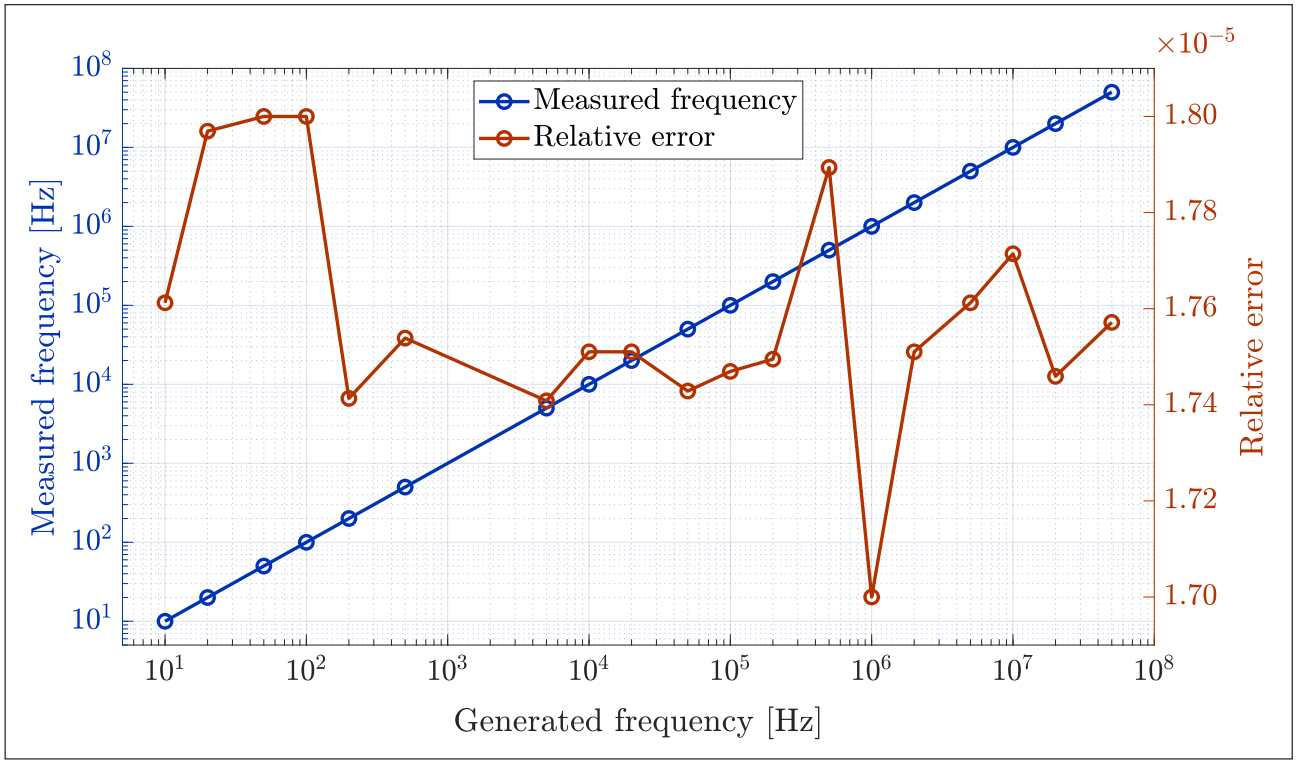

Figure 10.11: Measured characteristic of STM32F303RE frequency counter. The test signal was a 50% duty cycle, 3.3V peak square wave generated by the Tektronix AFG3102 function generator. Its frequency was measured by the VSVI frequency counter.

10.2 STM32G431KB version of developed SDI platform

The VSVI platform was also adapted for the STM32G431KB microcontroller in the LQFP32 package. The embedded USB peripheral is used for communication with the PC. An external 3.3 V voltage regulator is needed to supply the `VDD` supply voltage. An 8 MHz crystal oscillator should also be connected to the `HSE IN`, `HSE OUT` pins with the appropriate load capacitors (see [29, p. 100], [22]). Using this crystal oscillator significantly improves clock stability and the frequency accuracy of the generators and frequency counter. It is also absolutely necessary for MSO equivalent-time sampling of externally-generated signals. In case no crystal oscillator is detected at MCU startup, the internal RC oscillator (HSI) is used instead. Figure 10.12 shows the pinout of the VSVI platform for this MCU, as soldered on an LQFP32-to-DIP adapter for use in breadboards.

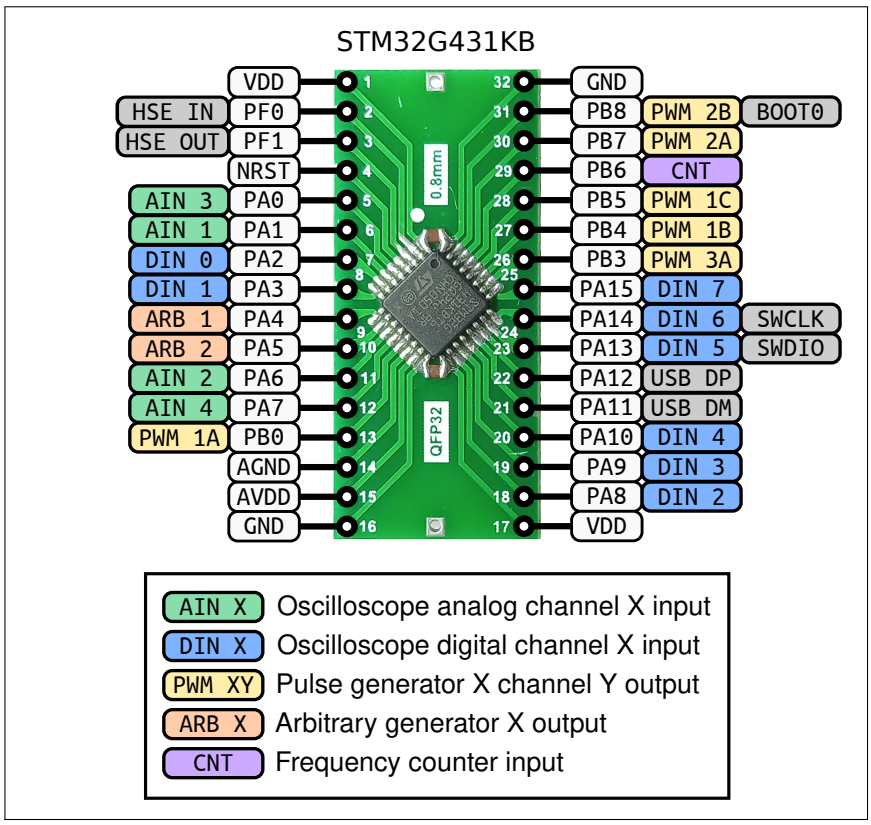

Figure 10.12: Pinout of SDIs developed for STM32G431KB (LQFP32-to-DIP adapter)

10.2.1 Mixed-signal oscilloscope (STM32G431KB)

- **4 analog channels** (inputs `AIN 1, AIN 2, AIN 3, AIN 4`) with **12-bit resolution**
- **8 digital channels** (inputs `DIN 0 - DIN 7`)
- Maximum real-time sampling rate:
  - **20.8 MSps** for all digital channels with analog channels disabled
  - **6.5 MSps** for one analog channel with 2 ADCs interleaving, digital channels disabled
  - **1.7 MSps** for all digital and analog channels with no interleaving

- Table 10.4 shows all the possible channel configurations and their respective max. sampling rates when all arbitrary generators are disabled. When any of the arbitrary generators is enabled, Table 10.5 applies instead.

- Maximum equivalent-time sampling rate:
  - Depends on the input signal frequency
  - Typically up to 104 MSps, max. 1.04 GSa/s with 1 sample per 10 input signal periods

Digital \ Analog	Disabled	1 channel	2 channels	4 channels
Disabled	-	6.5 MSps (i2) 3.4 MSps (-)	3.4 MSps (-)	1.7 MSps (-)
8 channels	20.8 MSps (-)	6.5 MSps (i2) 3.4 MSps (-)	3.4 MSps (-)	1.7 MSps (-)

Table 10.4: Maximum MSO sampling rates for STM32G431KB with arbitrary generators disabled. Used ADC interleaving mode indicated in parentheses ("-" = none, "i2" = 2 ADCs/channel).

Digital \ Analog	Disabled	1 channel	2 channels	4 channels
Disabled	-	6.5 MSps (i2) 3.4 MSps (-)	3.4 MSps (-)	1.7 MSps (-)
8 channels	13.0 MSps (-)	6.5 MSps (i2) 3.4 MSps (-)	3.4 MSps (-)	1.7 MSps (-)

Table 10.5: Maximum MSO sampling rates for STM32G431KB with arbitrary generators enabled. Used ADC interleaving mode indicated in parentheses ("-" = none, "i2" = 2 ADCs/channel).

- Maximum record length depends on the number of enabled channels according to Table 10.6

Digital \ Analog	Disabled	1 channel	2 channels	4 channels
Disabled	-	9.2 kSa	4.6 kSa	2.3 kSa
8 channels	9.2 kSa	4.6 kSa	3.1 KSa	1.8 kSa

Table 10.6: Maximum MSO record lengths for STM32G431KB

- All RTS and ETS configurations, including interleaved sampling modes, were tested as shown in Figures 10.13 through 10.17.

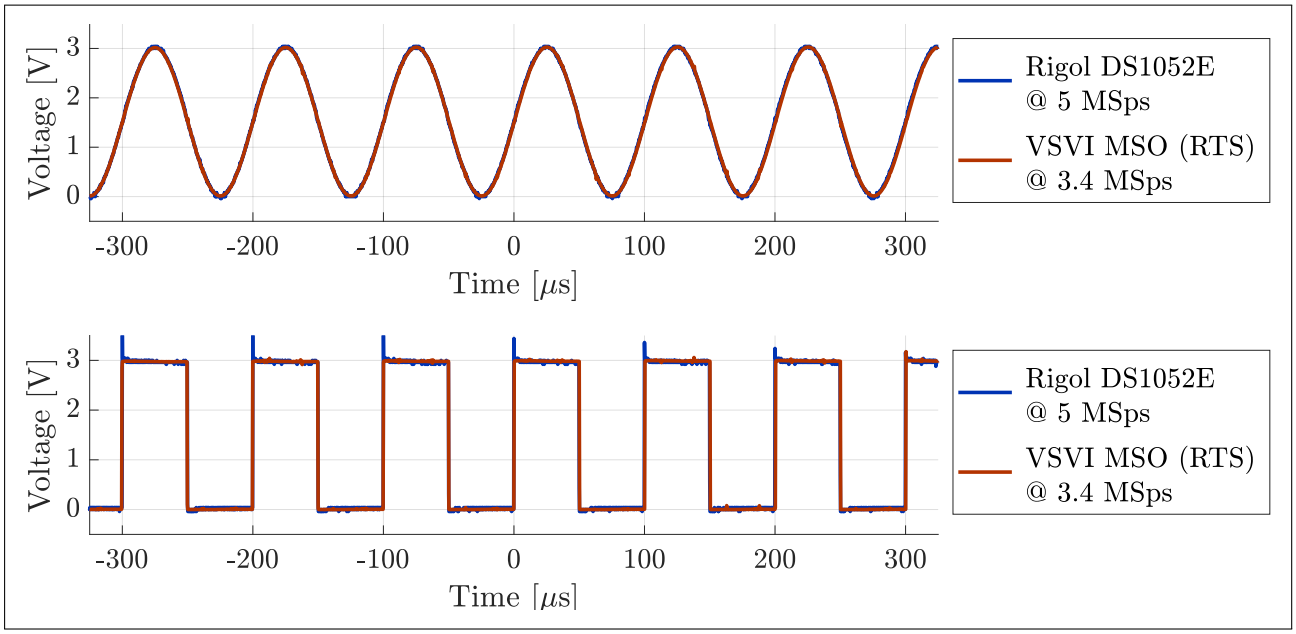

Figure 10.13: Waveform captured by STM32G431KB oscilloscope in RTS mode at 3.4 MSps. The test signals were a 3V peak sine wave (top) and 50% duty cycle square wave (bottom) with a frequency of 10 kHz, generated by the Tektronix AFG3102 function generator. Captured by both the VSVI MSO and Rigol DS1052E oscilloscope.

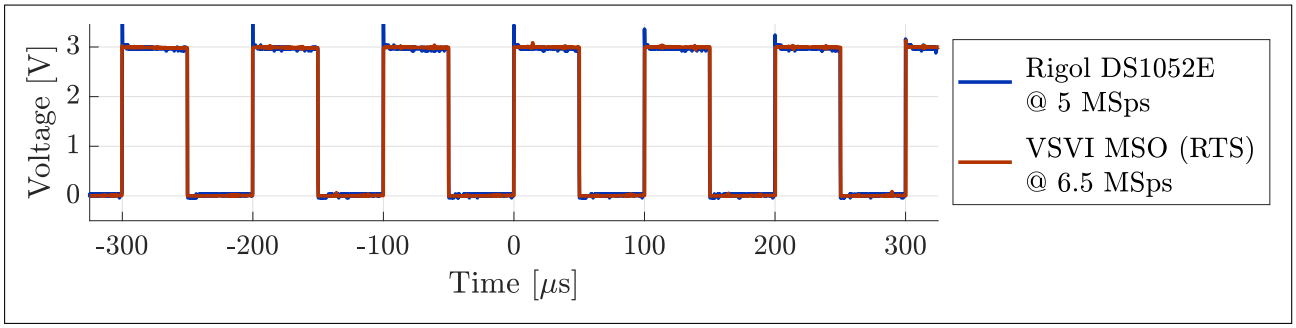

Figure 10.14: Waveform captured by STM32G431KB oscilloscope in RTS mode at 6.5 MSps (2 ADCs interleaved). The test signal is a 3V peak, 50% duty cycle square wave with a frequency of 10 kHz, generated by the Tektronix AFG3102 function generator. Captured by both the VSVI MSO and Rigol DS1052E.

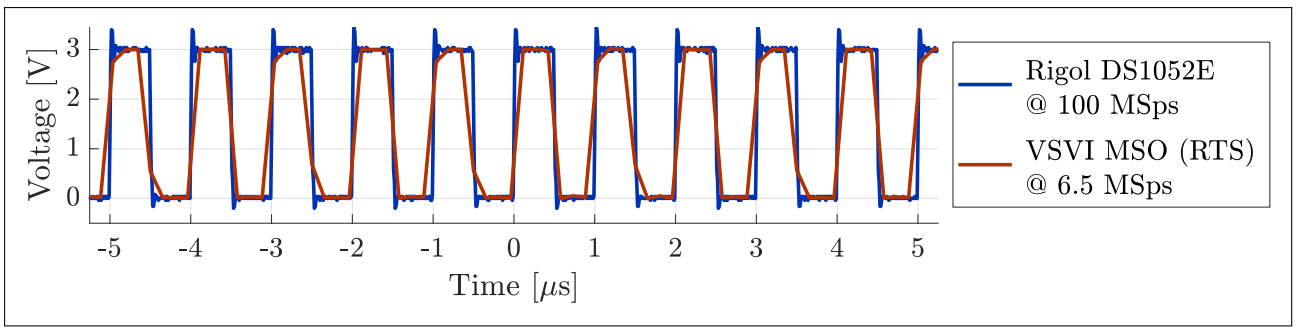

Figure 10.15: Waveform captured by STM32G431KB oscilloscope in RTS mode at 6.5 MSps (2 ADCs interleaved). The test signal is a 3V peak, 50% duty cycle square wave with a frequency of 1 MHz, generated by the Tektronix AFG3102 function generator. Captured by both the VSVI MSO and Rigol DS1052E.

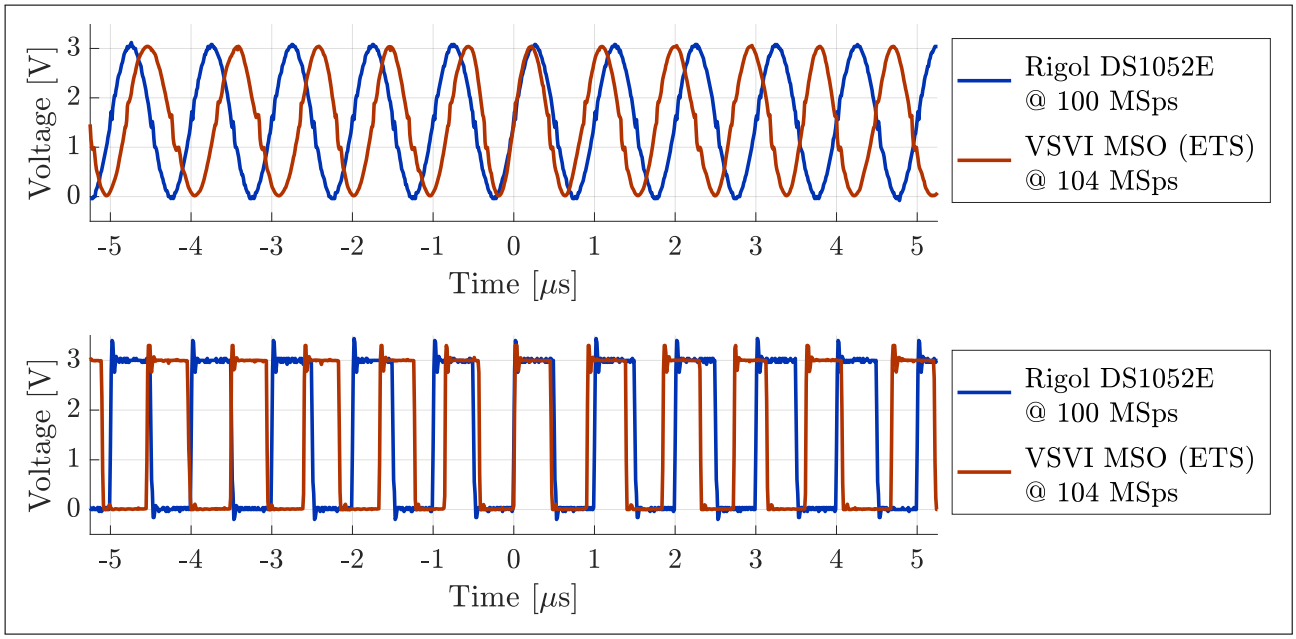

Figure 10.16: Waveforms captured by STM32G431KB oscilloscope in ETS mode at 104 MSps (equivalent, HSI RC clock). The test signals were a 3V peak sine wave (top) and 50% duty cycle square wave (bottom) with a frequency of 1 MHz, generated by the Tektronix AFG3102 function generator. They were captured by both the VSVI MSO and the Rigol DS1052E oscilloscope. The VSVI frequency counter measurement was used as the input signal frequency for ETS mode.

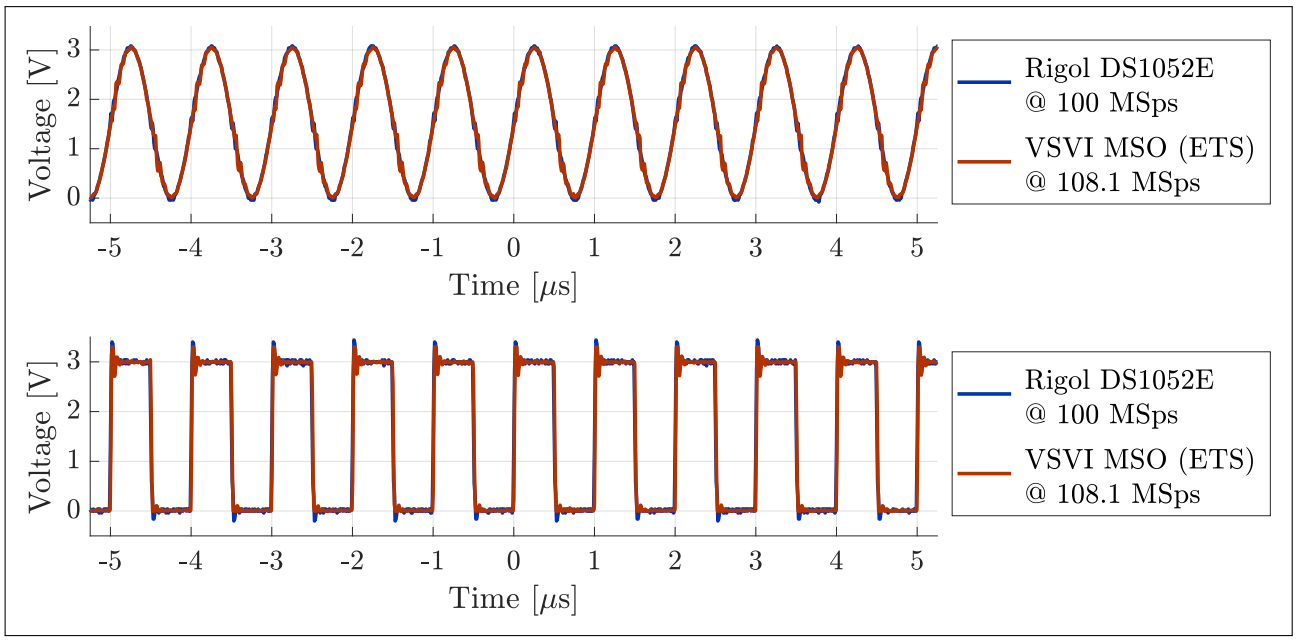

Figure 10.17: Waveforms captured by STM32G431KB oscilloscope in ETS mode at 108.1 MSps (equivalent, HSE crystal clock). The test signals were a 3V peak sine wave (top) and 50% duty cycle square wave (bottom) with a frequency of 1 MHz, generated by the Tektronix AFG3102 function generator. They were captured by both the VSVI MSO and the Rigol DS1052E oscilloscope. The VSVI frequency counter measurement was used as the input signal frequency for ETS mode.

- It can be seen from Figure 10.16 that the internal HSI RC oscillator is not a stable enough clock source for ETS mode. The same issue will occur if the MCU clock source was stable (i.e. using external HSE crystal oscillator) but the input signal frequency is not.
- **When interleaved sampling of analog channel 1 is used, the user must connect external jumper links between the following analog input pins:**
  - `AIN 1` (PA1) and `AIN 2` (PA6)

10.2.2 Pulse generators (STM32G431KB)

- **3 generators** available:
  - `PWM 1` with three channels (`PWM 1A, PWM 1B, PWM 1C`)
  - `PWM 2` with two channels (`PWM 2A, PWM 2B`)
  - `PWM 3` with one channel (`PWM 3A`)
- All can run independently or be synchronized with adjustable phase/time delay
- Maximum output signal frequency 52 MHz (50% duty cycle only)
- The output signals were validated and the frequency characteristic measured, as shown in Figures 10.18 and 10.19 respectively.

10.2.3 Arbitrary generators (STM32G431KB)

- **2 generators** available:
  - `ARB 1`
  - `ARB 2`
- Both can run independently or be synchronized with adjustable phases
- Maximum sampling frequency 1 MHz
- Maximum output signal frequency 250 kHz (4 samples per period)
- Up to 1000 samples per period for output signal frequencies ≤ 1 kHz
- The output signals were validated as shown in Figures 10.20 and 10.21.

10.2.4 Frequency counter (STM32G431KB)

- Input signal frequency range: 10 Hz – 52 MHz
- Gate time ≈ 1 s gives update rate ≈ 1 Hz
- Frequency resolution ≤ 1 Hz using modified frequency ratio measurement (see chapter 8)
- Accuracy is diminished when not using the HSE crystal oscillator as clock source, as demonstrated by frequency characteristics shown in Figures 10.22 and 10.23.

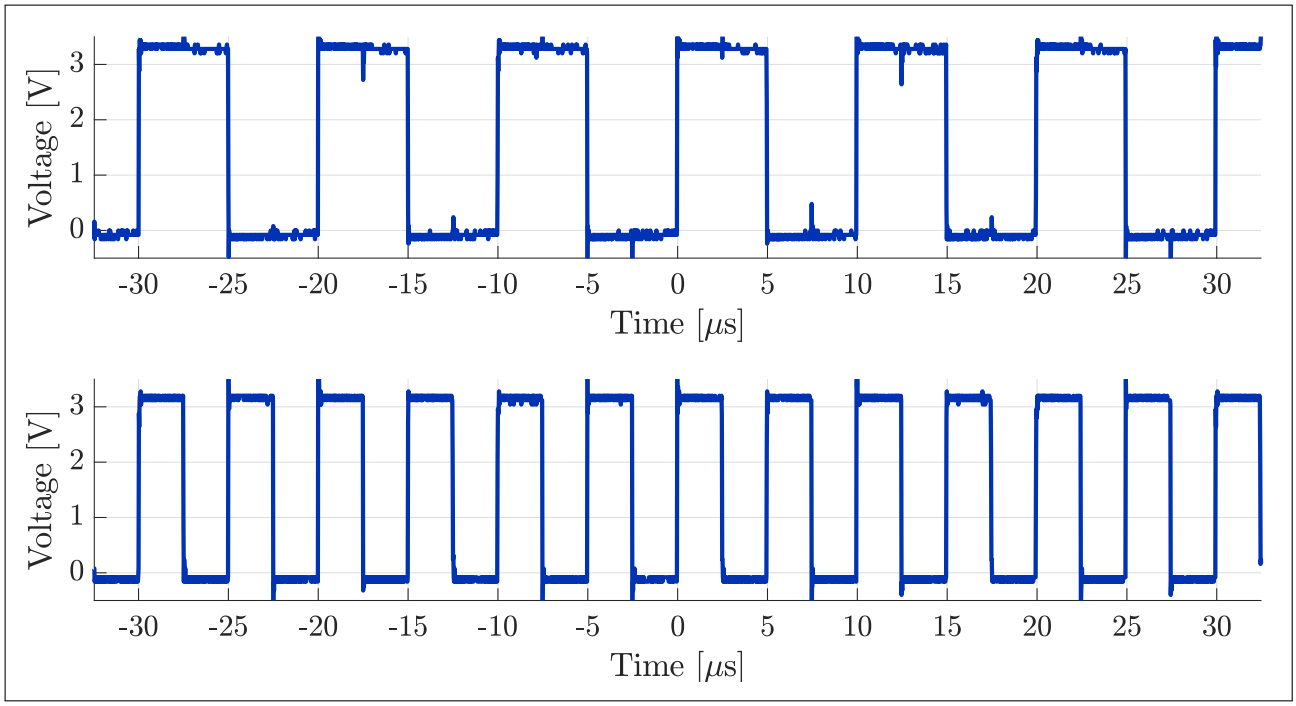

Figure 10.18: Output signals of STM32G431KB pulse generators. The test signals were 50% duty cycle, 3.3V peak square waves with frequencies of 100 kHz (top) and 200 kHz (bottom) generated by the VSVI pulse generators. The waveforms were recorded by the Rigol DS1052E oscilloscope sampling at 50 MSps.

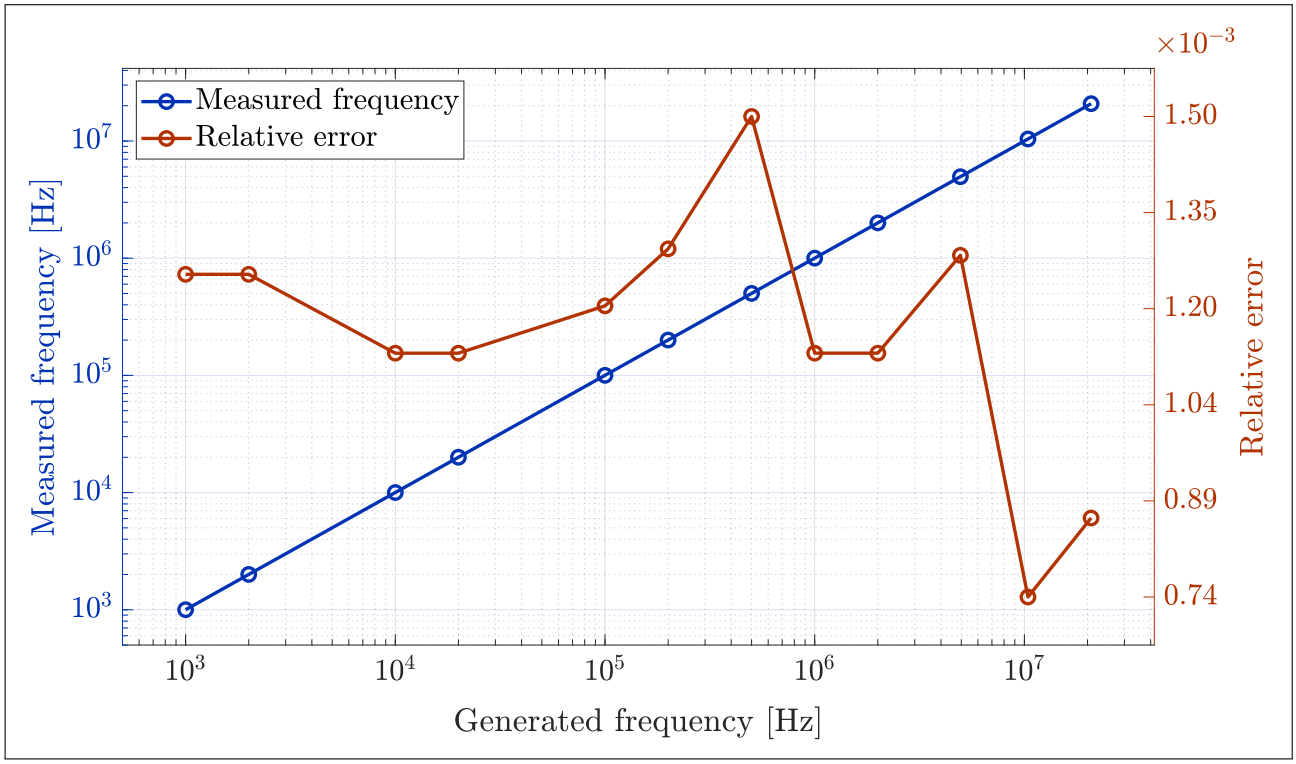

Figure 10.19: Measured frequency characteristic of STM32G431KB pulse generators (HSI RC clock). The test signals were a 50% duty cycle, 3.3V peak square waves generated by the VSVI pulse generators. Their frequencies were measured from waveforms recorded by the Rigol DS1052E oscilloscope.

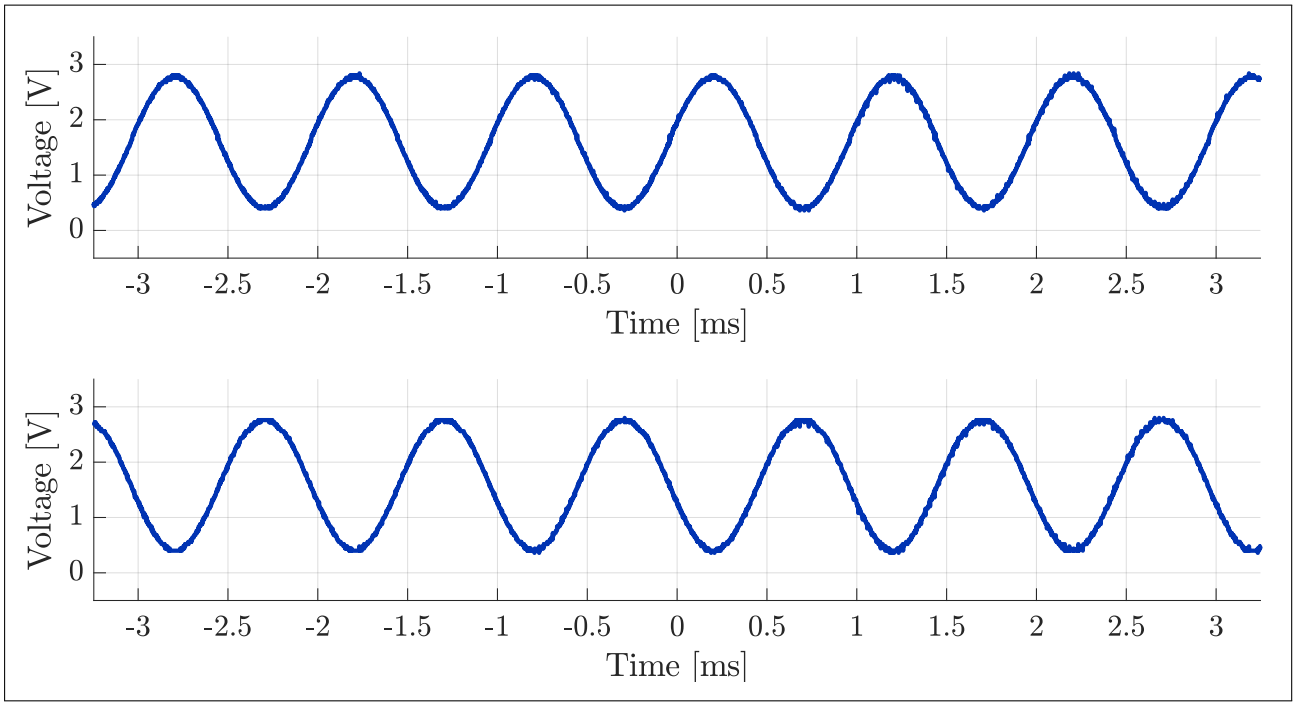

Figure 10.20: Output signals of STM32G431KB arbitrary generators. The test signals were sine waves with frequencies of 1 kHz and 180° phase offset generated by the VSVI arbitrary generators. Their amplitude was set to 1.2 V with a DC offset of 1.6 V. The waveforms were recorded by the Rigol DS1052E oscilloscope sampling at 1 MSps.

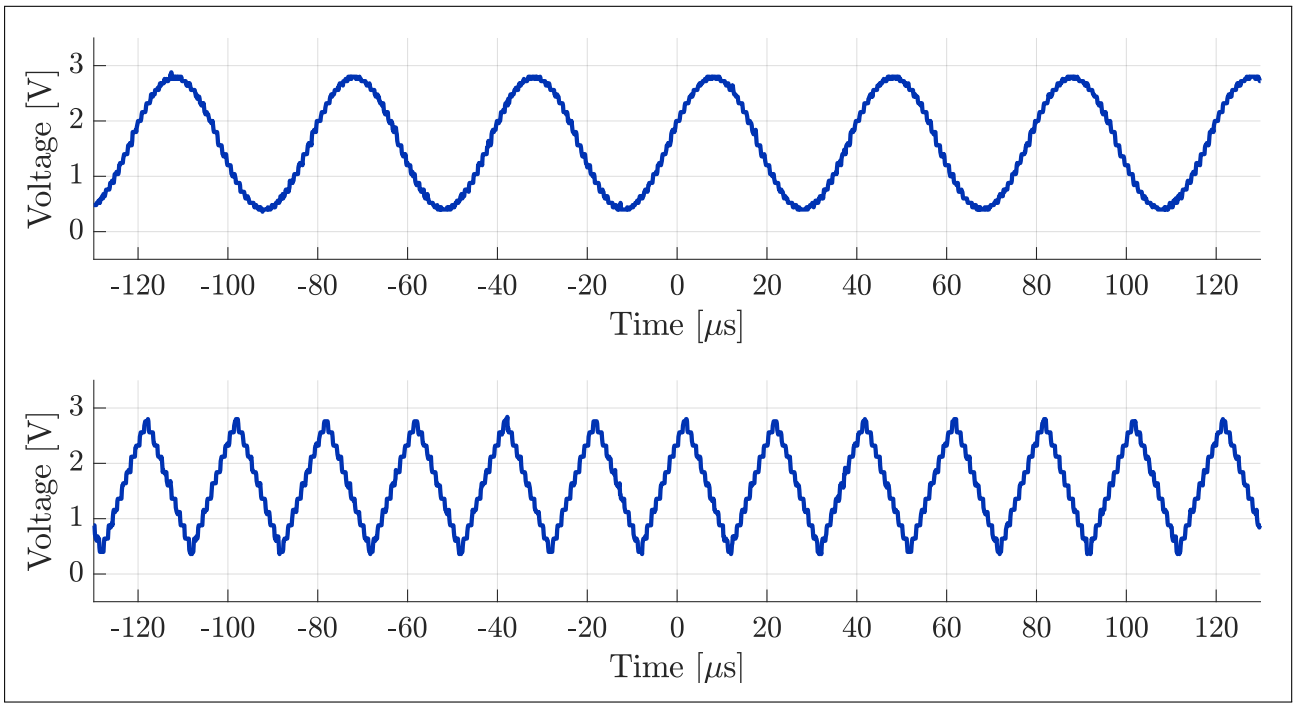

Figure 10.21: Output signals of STM32G431KB arbitrary generators. The test signals were a 25 kHz sine wave (top) and 50 kHz triangle wave (bottom) generated by the VSVI arbitrary generators. Their amplitude was set to 1.2 V with a DC offset of 1.6 V. The waveforms were recorded by the Rigol DS1052E oscilloscope sampling at 10 MSps.

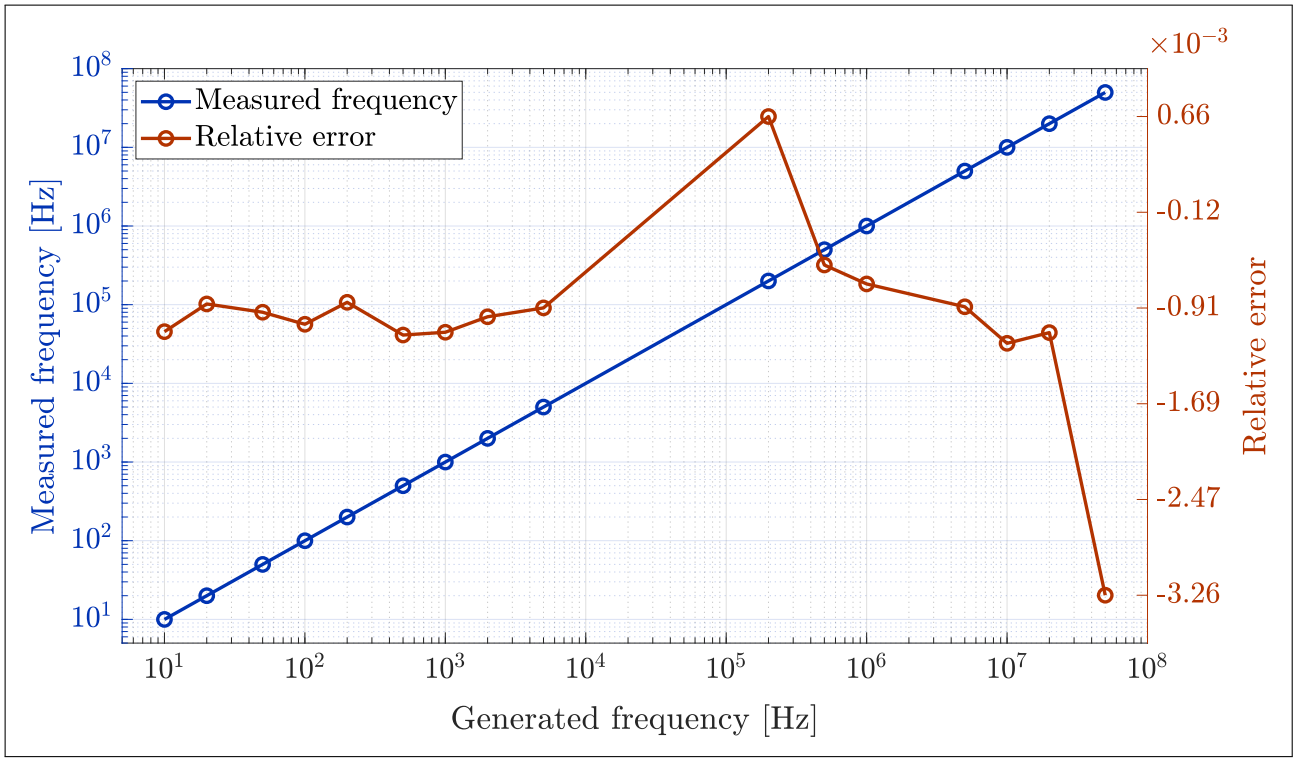

Figure 10.22: Measured characteristic of STM32G431KB frequency counter (HSI RC clock). The test signal was a 50% duty cycle, 3.3V peak square wave generated by the Tektronix AFG3102 function generator. Its frequency was measured by the VSVI frequency counter.

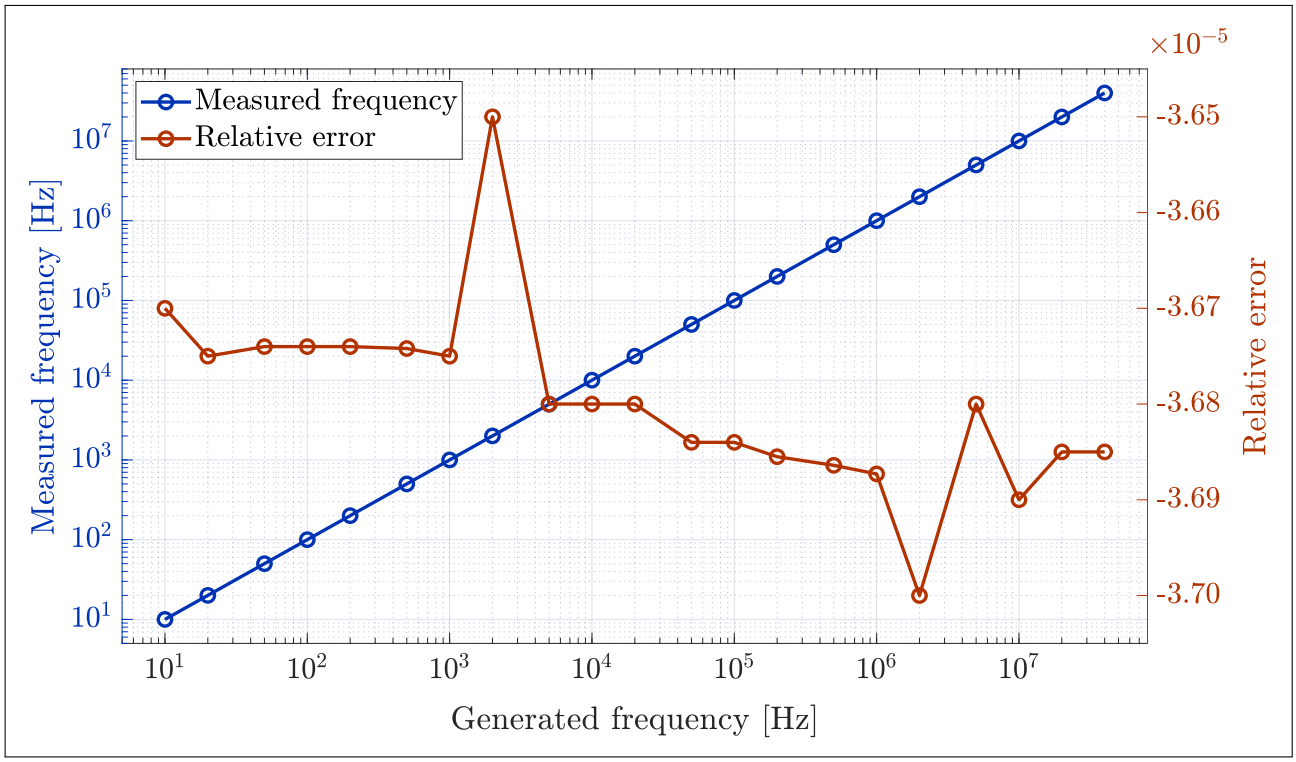

Figure 10.23: Measured characteristic of STM32G431KB frequency counter (HSE crystal clock). The test signal was a 50% duty cycle, 3.3V peak square wave generated by the Tektronix AFG3102 function generator. Its frequency was measured by the VSVI frequency counter.

Chapter 11

Evaluation of results

This chapter serves only to summarize this work and evaluate the achieved results, see chapter 10 for detailed specifications of the developed software-defined instruments. The main capabilities of the developed VSVI software-defined instrument platform are:

- Mixed-signal oscilloscope
  - Real-time and equivalent-time sampling mode, both with precise sampling frequency settings
  - Four analog channels with real-time sampling rates up to 5 MSps (STM32F303RE)
  - Interleaved ADC sampling with real-time sampling rates up to 18 MSps (STM32F303RE) [1]
  - Up to 16 digital channels with real-time sampling rates up to 24 MSps (STM32F303RE)
  - High equivalent-time sampling rates up to 104 MSps (STM32G431KB)
  - Synchronous sampling of analog and digital channels, option to trigger from either [1]
  - Equivalent-time sampling of externally-generated signals using the frequency counter [1]
- Pulse generators
  - One or more generators available for all MCUs
  - Precisely adjustable frequency and duty cycle
  - Wide frequency range, from 0.1 Hz up to 72 MHz (STM32F303RE)
  - Option to synchronize all generators with configurable phase/time delays [1]
- Arbitrary generators
  - Two generators available for all MCUs with DACs
  - Sampling rates up to 1 MSps
  - Precisely adjustable output frequency up to 250 kHz (4 samples per period)
  - Up to 1000 samples per period for output frequencies below 1 kHz
  - Generation of DC voltages or function waveforms (sine, sawtooth, square) with precisely adjustable amplitude, DC offset, duty cycle
  - Generation of custom waveforms according to wave files downloaded from the connected PC [1]
  - Option to synchronize the two generators with configurable phase angles [1]

[1] Capabilities beyond the scope of the thesis assignment

- Frequency counter
  - Resolution of 1 Hz or better, using a modified frequency ratio measurement
  - Wide input frequency range, from 10 Hz up to 72 MHz (STM32F303RE)
  - Improved accuracy at low frequencies by always counting whole input signal periods [1]
  - Can be used to support MSO equivalent-time sampling of an externally-generated signal [1]
- Configuration profiles [1]
  - Support for storing/loading all SDI parameters into/from a configuration profile [1]
  - Profiles can stored into the MCU's Flash memory or the connected PC (using Data Plotter) [1]

The VSVI platform provides a terminal user interface (TUI) implemented using ANSI escape sequences within the STM32 firmware. The TUI allows the monitoring and control of all parameters of the implemented software-defined instruments. The parameters were logically organized into multiple tabs. The universal PC application Data Plotter is used to display the acquired oscilloscope waveforms and to enable user-friendly mouse interaction with the TUI. This modular terminal user interface approach was detailed in this thesis and can serve as inspiration for other similar projects.

11.1 Comparison with existing SDI platforms

This section compares the VSVI software-defined instrument platform implemented in this work to some of the existing SDI platforms mentioned in section 2.2. In all cases, the integration of equivalent-time oscilloscope sampling is new and as such not mentioned in every comparison. Likewise, the use of a terminal user interface with Data Plotter is a novel approach – all discussed SDI platforms require the use of their custom PC application. The terminal user interface has proven to be almost as easy to use as a traditional PC application thanks to mouse interactivity enabled by Data Plotter. It also allows easy addition of new features and MCU support, without the need to obtain and install a new version of the PC application.

One of the most popular existing SDI platforms is the Little Embedded Oscilloscope (LEO) implemented for the STM32F303RE microcontroller. Compared to it, VSVI adds features such as digital oscilloscope channels (integrated logic analyzer) and advanced interleaved sampling. The latest version of LEO does support interleaved sampling, but only for 1 analog channel at up to 8 Msps (using 2 ADCs). Older versions of LEO do not support interleaved sampling at all. VSVI supports interleaved sampling of 2 analog channels at up to 10 MSps or of 1 analog channel at up to 18 MSps (for STM32F303RE). Moreover, VSVI adds additional instruments, namely pulse generators and a frequency counter. The absence of pulse generators is an issue when a signal with sharp edges (high slew rate) needs to be generated – this is something that the DAC-based function generators in LEO cannot achieve. The concurrent functionality of all LEO instruments is also not guaranteed, for example it is a known issue that corruption of oscilloscope waveform data may occur at high sampling rates when function generators are enabled (due to increased load on the DMA controllers). In contrast, the VSVI firmware accounts for these limitations automatically by limiting the max. sampling rate accordingly to ensure all measured data is correct. Additionally, the VSVI firmware has also been adapted for the STM32G431KB microcontroller, whereas LEO only supports the STM32F303RE.

[1] Capabilities beyond the scope of the thesis assignment

Zero eLab Viewer is another popular SDI platform. It is more universal than LEO, being implemented for multiple types of STM32 MCUs. Compared to it, VSVI offers additional instruments, namely arbitrary generators and frequency counter. Additionally, the oscilloscope also adds digital channels and support for interleaved sampling. However, this makes VSVI a larger firmware package with higher hardware requirements. Therefore, it couldn't be adapted for the low-end MCUs that work with Zero eLab Viewer, for example STM32F042F6 and STM32G030J6.

In comparison with EMBO - Embedded Oscilloscope, VSVI adds interleaved oscilloscope sampling and an integrated logic analyzer with more channels (16 vs. 4 on STM32F303RE) and higher max. sampling rates (24 MSps vs. 14.4 MSps on STM32F303RE). Is also provides more pulse generators (four generators with 7 total output channels vs. two channels) and arbitrary generators (two vs. one). The arbitrary generators also support the generation of custom waveforms, whereas EMBO's signal generator can only generate a number of predefined functions (sine, sawtooth, etc.). Lastly, the frequency counter in EMBO requires switching between slow and fast input signal modes, whereas VSVI automatically supports a wide range of input frequencies while maintaining its accuracy.

Chapter 12
Conclusion

The goal of this work was to develop software-defined instruments as a practical low-cost alternative to professional instruments, mainly for teaching purposes at the Faculty of Electrical Engineering, CTU. In contrast to similar projects from the past, this work sought to provide a more universal solution by supporting multiple STM32 MCUs and not relying on a specialized PC application. New features such as equivalent-time sampling of the oscilloscope were also to be implemented. Additional instruments (e.g. logic analyzer, signal generators) were also to be included whenever possible given the capabilities of the microcontroller used.

A versatile software-defined instrument platform was developed in this work. The primary instrument is a mixed-signal oscilloscope capable of both real-time and equivalent-time sampling. Interleaved sampling is also supported, multiplying the maximum sampling rate when some analog channels are disabled. Even higher sampling rates can be achieved for the digital channels. A frequency counter was implemented to support equivalent-time sampling and for general-purpose use. Additionally, multiple synchronizable pulse generators with precisely adjustable frequencies, phase delays and duty cycles were also developed. For MCUs with embedded DACs, a pair of synchronizable arbitrary generators were also implemented. Besides a number of functions, fully custom waveforms can also be generated from wave files downloaded from the connected PC. For added convenience, the values of all instrument parameters can be saved to/recalled from Flash memory or the connected PC.

To control the developed software-defined instruments, a terminal user interface was implemented within the MCU firmware. It is used in conjunction with the universal, multi-platform PC application Data Plotter (not developed in this work). This approach simplifies updates to the MCU firmware, such as adding new features or support for new MCUs and eliminates the need to install a specialized PC application, as Data Plotter is used by multiple SDI platforms currently in development at the Department of Measurement, FEE CTU. Moreover, using clickable buttons in the developed terminal user interface provides a user experience similar to that of typical PC applications.

Additionally, a comprehensive user manual and instruction video for the developed software-defined instrument platform were also created. All implemented instruments were tested and their capabilities specified. The developed firmware has been adapted for two STM32 microcontrollers, STM32G431KB and STM32F303RE. Support for more types of STM32 microcontrollers will continue to be added in the future. I believe the assignment of this thesis has been fulfilled.

Bibliography

[1] *LEO - Little Embedded Oscilloscope* [online]. Available from: `https://embedded.fel.cvut.cz/platformy/leo` [accessed 20 Jan, 2022]

[2] *Zero eLab Viewer* [online]. Available from: `https://embedded.fel.cvut.cz/SDI/STM32F042/Zeroelabwiever`[accessed 29 Apr, 2022]

[3] *ELA - Logický analyzátor s Nucleo F303RE* [online]. Available from: `https://embedded.fel.cvut.cz/platformy/ELA` [accessed 29 Apr, 2022]

[4] *EMBO - Osciloskop* [online]. Available from: `https://embedded.fel.cvut.cz/platformy/embo` [accessed 29 Apr, 2022]

[5] *Virtuální Čítač na platformě STM32* [online]. Available from: `https://mcejp.github.io/virtual-counter-cz.html` [accessed 29 Apr, 2022]

[6] *SDI STM32F103 – SDI založené na STM32F103C8, který je použitý také v Blue Pill* [online]. Available from: `https://embedded.fel.cvut.cz/SDI/STM32F103/` [accessed 29 Apr, 2022]

[7] *F0 - Lab* [online]. Available from: `https://embedded.fel.cvut.cz/platformy/F0_lab` [accessed 29 Apr, 2022]

[8] *G0-Lab s STM32G030 – Přístroje založené na STM32G030* [online]. Available from: `https://embedded.fel.cvut.cz/SDI/STM32G030` [accessed 29 Apr, 2022]

[9] VANĚČEK, Vít. *Microcontroller Based Logic Analyser*. Prague, 2020. Bachelor's thesis. Czech Technical University, Faculty of Electrical Engineering.

[10] PAŘEZ, Jakub. *Software defined oscilloscope based on STM32F103*. Prague, 2021. Master's thesis. Czech Technical University, Faculty of Electrical Engineering.

[11] CEJP, Martin. *Microcontroller-based Virtual Instrument for Signal Analysis in the Modulation Domain*. Prague, 2017. Bachelor's thesis. Czech Technical University, Faculty of Electrical Engineering.

[12] *Data Plotter* [online]. Available from: `https://embedded.fel.cvut.cz/platformy/dataplotter` [accessed 22 Apr, 2022]

[13] MAIER, Jiří. *The Universal GUI for PC Based Oscillographs*. Prague, 2021. Bachelor's thesis. Czech Technical University, Faculty of Electrical Engineering.

[14] ECMA. *Standard ECMA-48:Control Functions for Coded Character Sets, Fifth Edition* [online]. 1991. Available from: `https://www.ecma-international.org/wp-content/uploads/ECMA-48_5th_edition_june_1991.pdf` [accessed 1 May 2022]

[15] ANSI. *ANSI X3.64 Standard* [online]. Available from: https://vt100.net/annarbor/aaa-ug/section13.html [accessed 1 May 2022]

[16] PETR, David. *USB Control Unit for Optoelectronic Incremental Encoder*. Prague, 2020. Bachelor's thesis. Czech Technical University, Faculty of Electrical Engineering.

[17] Tektronix. *Real-Time Versus Equivalent-Time Sampling* [online]. Available from: https://www.tek.com/en/document/application-note/real-time-versus-equivalent-time-sampling [accessed 30 Dec, 2021]

[18] *Test of A/D Converters - Scientific Figure on ResearchGate* [online]. Available from: https://www.researchgate.net/figure/6-Equivalent-time-sampling_fig2_234091862 [accessed 30 Dec, 2021]

[19] YIU, Joseph. *A Beginner's Guide on Interrupt Latency - and Interrupt Latency of the Arm Cortex-M processors* [online]. 2016. Available from: https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/beginner-guide-on-interrupt-latency-and-interrupt-latency-of-the-arm-cortex-m-processors [accessed 28 Dec, 2021]

[20] Keysight Technologies. *MSOX2014A Mixed Signal Oscilloscope: 100 MHz, 4 Analog Plus 8 Digital Channels* [online]. Available from: https://www.keysight.com/zz/en/product/MSOX2014A/mixed-signal-oscilloscope-100-mhz-4-analog-8-digital-channels.html [accessed 1 May, 2022]

[21] STMicroelectronics. *AN2548 Application note – Using the STM32F0/F1/F3/Gx/Lx Series DMA controller* [online]. Revision 7, 2020. Available from: https://www.st.com/resource/en/application_note/cd00160362-using-the-stm32f0f1f3gxlx-series-dma-controller-stmicroelectronics.pdf [accessed 2 May, 2022]

[22] STMicroelectronics. *AN2867 Application note – Oscillator design guide for STM8AF/AL/S, STM32 MCUs and MPUs* [online]. Revision 15, 2021. Available from: https://www.st.com/resource/en/application_note/cd00221665-oscillator-design-guide-for-stm8afals-stm32-mcus-and-mpus-stmicroelectronics.pdf [accessed 5 May, 2022]

[23] STMicroelectronics. *Datasheet - STM32F303xD STM32F303xE* [online]. Revision 5, 2016. Available from: https://www.st.com/resource/en/datasheet/stm32f303re.pdf [accessed 3 Jan, 2022]

[24] STMicroelectronics. *STM32F303xB/C Errata sheet* [online]. Revision 11, 2022. Available from: https://www.st.com/resource/en/errata_sheet/es0204-stm32f303xbc-device-errata-stmicroelectronics.pdf [accessed 1 April, 2022]

[25] STMicroelectronics. *STM32 Nucleo-64 development board with STM32F303RE MCU* [online]. Available from: https://www.st.com/en/evaluation-tools/nucleo-f303re.html [accessed 4 Jan, 2022]

[26] STMicroelectronics. *RM0316 Reference manual* [online]. Revision 8, 2017. Available from: https://www.st.com/resource/en/reference_manual/dm00043574-stm32f303xb-c-d-e-stm32f303x6-8-stm32f328x8-stm32f358xc-stm32f398xe-advanced-arm-based-mcus-stmicroelectronics.pdf [accessed 3 Jan, 2022]

[27] STMicroelectronics. *UM1724 User manual - STM32 Nucleo-64 boards* [online]. Revision 14, 2020. Available from: https://www.st.com/resource/en/user_manual/um1724-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf [accessed 1 Jan, 2022]

[28] STMicroelectronics. *STM32 Nucleo (64 pins) schematics* [online]. Version 1.0, 2018. Available from: https://www.st.com/resource/en/schematic_pack/nucleo_64pins_sch.zip [accessed 1 Jan, 2022]

[29] STMicroelectronics. *Datasheet - STM32G431x6 STM32G431x8 STM32G431xB* [online]. Revision 6, 2021. Available from: https://www.st.com/resource/en/datasheet/stm32g431kb.pdf [accessed 2 Jan, 2022]

[30] STMicroelectronics. *STM32 Nucleo-32 development board with STM32G431KB MCU* [online]. Available from: https://www.st.com/en/evaluation-tools/nucleo-g431kb.html [accessed 2 Jan, 2022]

[31] STMicroelectronics. *RM0440 Reference manual* [online]. Revision 6, 2021. Available from: https://www.st.com/resource/en/reference_manual/rm0440-stm32g4-series-advanced-armbased-32bit-mcus-stmicroelectronics.pdf [accessed 2 Jan, 2022]

[32] STMicroelectronics. *STM32G4 Nucleo-32 board (MB1430) - User manual* [online]. Revision 2, 2019. Available from: https://www.st.com/resource/en/user_manual/um2397-stm32g4-nucleo32-board-mb1430-stmicroelectronics.pdf [accessed 2 Jan, 2022]

[33] STMicroelectronics. *Datasheet - STM32F446xC/E* [online]. Revision 10, 2021. Available from: https://www.st.com/resource/en/datasheet/stm32f446re.pdf [accessed 5 Feb, 2022]

[34] STMicroelectronics. *Datasheet - STM32F103x8 STM32F103xB* [online]. Revision 18, 2022. Available from: https://www.st.com/resource/en/datasheet/stm32f103c8.pdf [accessed 6 Apr, 2022]

[35] STMicroelectronics. *RM0008 Reference manual* [online]. Revision 21, 2021. Available from: https://www.st.com/resource/en/reference_manual/cd00171190-stm32f101xx-stm32f102xx-stm32f103xx-stm32f105xx-and-stm32f107xx-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf [accessed 7 Apr, 2022]

[36] STMicroelectronics. *Datasheet - STM32F042x4 STM32F042x6* [online]. Revision 5, 2017. Available from: https://www.st.com/resource/en/datasheet/stm32f042f6.pdf [accessed 10 Apr, 2022]

[37] STMicroelectronics. *Datasheet - STM32G431x6 STM32G431x8 STM32G431xB* [online]. Revision 6, 2021. Available from: https://www.st.com/resource/en/datasheet/stm32g431r6.pdf [accessed 2 Jan, 2022]

[38] STMicroelectronics. *RM0394 Reference manual* [online]. Revision 4, 2018. Available from: https://www.st.com/resource/en/reference_manual/rm0394-stm32l41xxx42xxx43xxx44xxx45xxx46xxx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf [accessed 2 Jan, 2022]

[39] STMicroelectronics. *Datasheet - STM32L072x8 STM32L072xB STM32L072xZ* [online]. Revision 5, 2019. Available from: https://www.st.com/resource/en/datasheet/stm32l072v8.pdf [accessed 9 Mar, 2022]

[40] STMicroelectronics. *RM0376 Reference manual* [online]. Revision 7, 2022. Available from: https://www.st.com/resource/en/reference_manual/rm0376-ultralowpower-stm32l0x2-advanced-armbased-32bit-mcus-stmicroelectronics.pdf [accessed 10 Mar, 2022]

[41] STMicroelectronics. *STM32L07xxx STM32L08xxx Errata Sheet* [online]. Revision 7, 2022. Available from: https://www.st.com/resource/en/errata_sheet/es0292-stm32l07xxxl08xxx-device-errata-stmicroelectronics.pdf [accessed 2 Apr, 2022]

Appendix A

List of symbols

Acronym	Meaning
MCU	Microcontroller
GUI	Graphical User Interface
TUI	Terminal User Interface
CPU	Central Processing Unit (MCU Core)
USB	Universal Serial Bus
UART	Universal Asynchronous Receiver-Transmitter
DMA	Direct Memory Access controller
SRAM	Volatile Static Random Access Memory
CCMRAM	Core-Coupled Random Access Memory (Routine Booster)
ADC	Analog-to-Digital Converter
DAC	Digital-to-Analog Converter
AWD	Analog Watchdog
GPIO	General-Purpose Input/Output
EXTI	Extended Interrupts and Events Controller
TIM	Timer
PSC	Timer Prescaler register
ARR	Timer Auto-Reload Register
CCx	Timer Capture/Compare channel X
TRGO	Timer trigger output
ITRx	Timer internal trigger input X
OPM	Timer One-Pulse Mode

Table A.1: Established acronyms

Acronym	Meaning
SDI	Software-Defined Instrument
VI	Virtual Instrument
VSVI	Versatile STM32 Virtual Instrument
RTS	Real-Time Sampling
ETS	Equivalent-Time Sampling
PWM	Pulse generator
ARB	Arbitrary generator
CNT	Frequency counter

Table A.2: Software-defined instrument acronyms used in this work

Symbol(s)	Meaning
f_{ST}	Sampling timer frequency
T_{ST}	Sampling timer period
N_T	Number of sampling timer periods elapsed in an acquisition
f_{SR}	Real-time sampling frequency
T_{SR}	Real-time sampling period
f_{SE}	Equivalent-time sampling frequency
T_{SE}	Equivalent-time sampling period
N_P	Number of input signal periods per sample in ETS mode
f_{IN}	Fundamental frequency of input signal in ETS mode
T_{IN}	Fundamental period of input signal in ETS mode
f_{NYQ}	Nyquist frequency, half of real-time sampling frequency
$f_{IN,max}$	Highest frequency in input signal
N_A	Current number of samples acquired since start of acquisition
N_R	Record length – total number of samples acquired per channel
N_E	Number of extra samples for trigger delay correction
A_X^Y	Value of the "X"-th sample of analog channel "Y"
D_X	Value of the "X"-th sample of all digital channels

Table A.3: Mixed-signal oscilloscope symbols

Symbol(s)	Meaning
f_{PWMX}	Frequency of the "PWM X" generator
T_{PWMX}	Period of the "PWM X" generator
D_{PWMXY}	Duty cycle of the "PWM XY" output (channel "Y" of generator "X")
P_{PWMXY}	Pulse width of the "PWM XY" output (channel "Y" of generator "X")

Table A.4: Pulse generator symbols

Symbol(s)	Meaning
f_{ARBX}	Frequency of the "ARB X" generator output signal
T_{ARBX}	Period of the "ARB X" generator output signal
D_{ARBX}	Duty cycle of the "ARB X" generator output signal
P_{ARBX}	Pulse width of the "ARB X" generator output signal
f_{SX}	Sampling frequency of the "ARB X" generator
T_{SX}	Sampling period of the "ARB X" generator
N_{SXX}	Number of samples generated per period of the "ARB X" generator

Table A.5: Arbitrary generator symbols

Symbol(s)	Meaning
t_{GATE}	Gate time
f_{INP}	Input signal frequency
T_{INP}	Input signal period
N_{INP}	Number of input signal periods counted within gate time
f_{REF}	Reference signal frequency
T_{REF}	Reference signal period
N_{REF}	Number of reference signal periods counted within gate time

Table A.6: Frequency counter symbols

Symbol(s)	Meaning
f_{CPU}	Core (CPU) clock frequency
T_{CPU}	Core (CPU) clock period
f_{TIM}	Timer clock frequency
T_{TIM}	Timer clock period
f_{ADC}	ADC clock frequency
T_{ADC}	ADC clock period
R_{AIN}	ADC input signal impedance
t_{LATR}	Regular ADC conversion trigger latency
t_{SMP}	ADC sampling time
t_{SAR}	ADC successive approximation time
t_{CONV}	Total ADC conversion time ($t_{SMP} + t_{SAR}$)

Table A.7: Microcontroller hardware symbols

Appendix B

Contents of the enclosed CD

- This thesis in PDF format (`/F3-DP-2022-Dujava-Jozef.pdf`)
- User manual in PDF format (`/VSVI_userManual_EN.pdf`)
- Instruction video in MP4 format (`/VSVI_instructions_EN.mp4`)
- Source code of the implemented firmware (`/<MCU version>/*`)
- Compiled firmware binaries (`/<MCU version>/release/*.bin`)

Appendix C

User manual for developed SDI platform

Czech Technical University in Prague, Faculty of Electrical Engineering
Department of Measurement, Laboratory of Videometry

VSVI - Versatile STM32 Virtual Instrument

User manual

Bc. Jozef Dujava

Contents

1 Introduction & getting started

The Versatile STM32 Virtual Instrument (VSVI) is a software-defined instrument platform implemented for multiple STM32 microcontrollers. The primary instrument is a mixed-signal oscilloscope with real-time and equivalent-time sampling capability. Additional instruments include pulse generators, arbitrary generators and a frequency counter. Very few additional components are necessary besides a supported STM32 microcontroller (MCU). The virtual instruments are operated from a PC using a terminal user interface (TUI) implemented within the MCU firmware. If supported, the MCU is connected to the PC via USB directly, otherwise an external USB-UART converter is used (e.g. CH340, ST-Link).

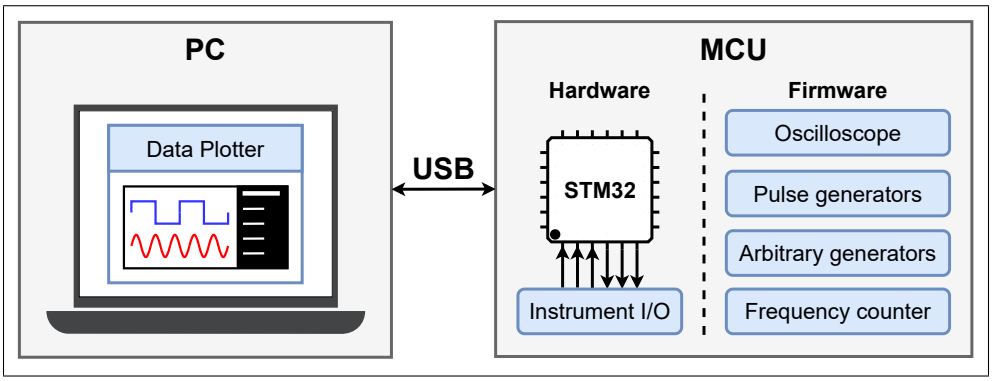

Figure 1: Diagram of the VSVI platform (USB version)

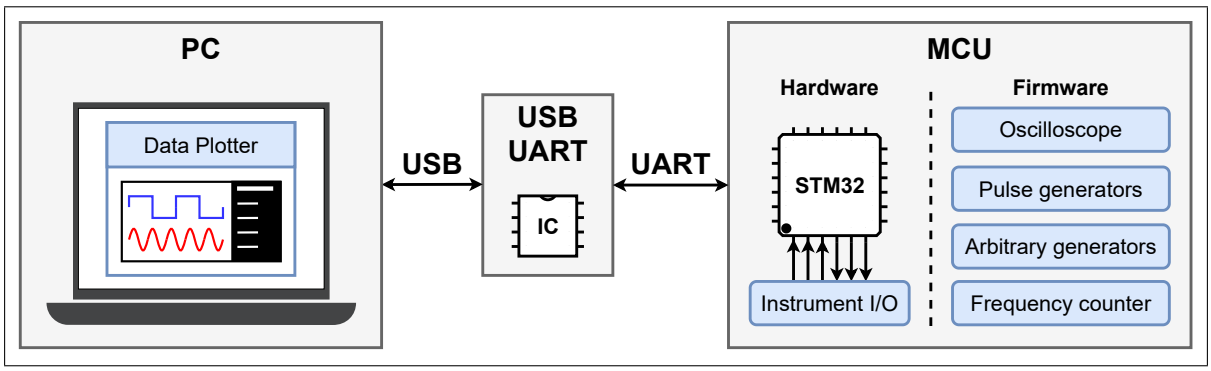

Figure 2: Diagram of the VSVI platform (UART version)

The universal Data Plotter PC application is used to display acquired oscilloscope waveforms, access the TUI and transfer files between the PC and microcontroller. This application was developed by Bc. Jiří Maier, and is not a part of my thesis. It is listed on the Embedded server[1] of the Department of Measurement, FEE, CTU in Prague. Its source code and binary releases are available from GitHub[2].

This manual describes the functionality and usage of the VSVI system. Note that certain features may only be available on higher-end MCUs, such as the STM32F303RE which is used as an example throughout this manual. Screenshots show the 17.3.2022 release of Data Plotter[3].

[1] `https://embedded.fel.cvut.cz/platformy/dataplotter`
[2] `https://github.com/jirimaier/DataPlotter`
[3] `https://github.com/jirimaier/DataPlotter/releases/tag/v2.0`

1.1 Programming the MCU

To use the VSVI platform, the microcontroller must be programmed with the appropriate firmware. This can be done via an ST-Link or serial (UART) programmer or directly via USB if a USB bootloader had been programmed previously onto the MCU. If UART is used to communicate with the PC, the baud rate is fixed within the firmware. Therefore, multiple firmware binaries had been compiled to support various baud rates. Binaries are named "**VSVI_<platform>_<comm>_<release>.bin**", where:

- **<platform>**: Name of the hardware platform. Can be either:
  - "<MCU code>−<package>", e.g. "G431-LQFP32" for standalone MCUs
  - "<MCU code>−<board name><pin count>", e.g. "G431-Nucleo32", "F303-Nucleo64", "F103-BluePill48" for development boards
- **<comm>**: Description of the PC communication interface, can be:
  - "UART−<baud>": using UART with baud rate <baud>
  - "USB−VCP": using USB in Virtual COM Port (VCP) mode.
- **<release>**: Release date in "R<YYYYMMDD>" format

1.2 Connecting the MCU to the PC

After the MCU has been programmed, connect it to the PC via the appropriate interface, according to the firmware version as described in the previous section. Also connect the necessary input and output signals according to the appropriate MCU pinout found in section 8. Open Data Plotter and select the MCU in the connection tab in the sidebar, as shown in Figure 3. The MCU's device descriptor should be listed in the drop-down menu, in this case it is "ttyACM0 - STM32 STLink".

If using UART, configure the correct baud rate in the connection window (921600 baud in this case). If using USB, the baud rate setting is irrelevant. Finally, click the connection button (indicated by cursor) to connect to the MCU. If the connection is successful, the MCU sends a welcome message which is displayed in the Data Plotter log. To start using the virtual instruments now, access the terminal user interface by switching to the terminal tab in Data Plotter.

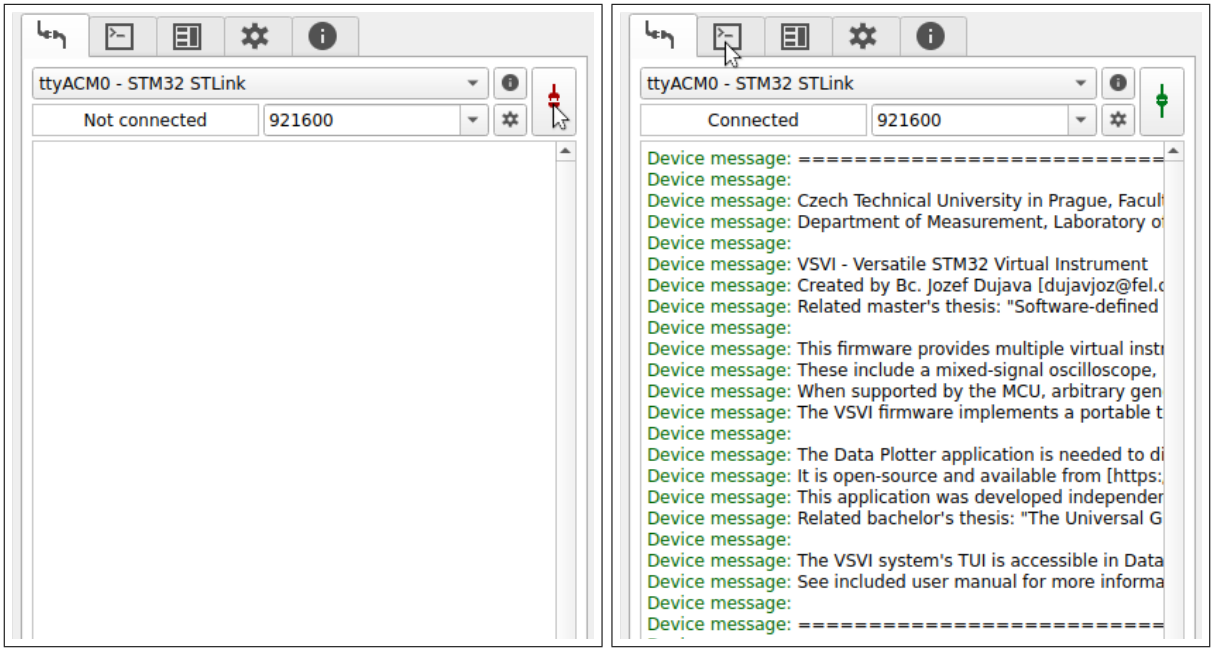

Figure 3: Data Plotter sidebar on startup (left) and after the MCU is connected (right)

2 Terminal user interface (TUI)

The terminal user interface (TUI) is displayed in Figure 4. It is compartmentalized into tabs, one for each type of virtual instrument. The name of the currently open tab is shown in the white title bar at the very top. The blue button to the left of the title opens a drop-down tab selection menu containing the name of every tab. Clicking the corresponding button on the left then opens this tab.

Within each instrument tab, there are a number of settings/parameters displayed. Each group of settings is described in detail in the "¡instrument code¿ TUI tab:" sections of this manual. Values are highlighted in white blocks, with units immediately on their right. If the value of a parameter can be changed by the user, a typically blue "edit" button is also shown to the left of the parameter's name. Its behavior depends on the parameter type, detailed in the following subsection.

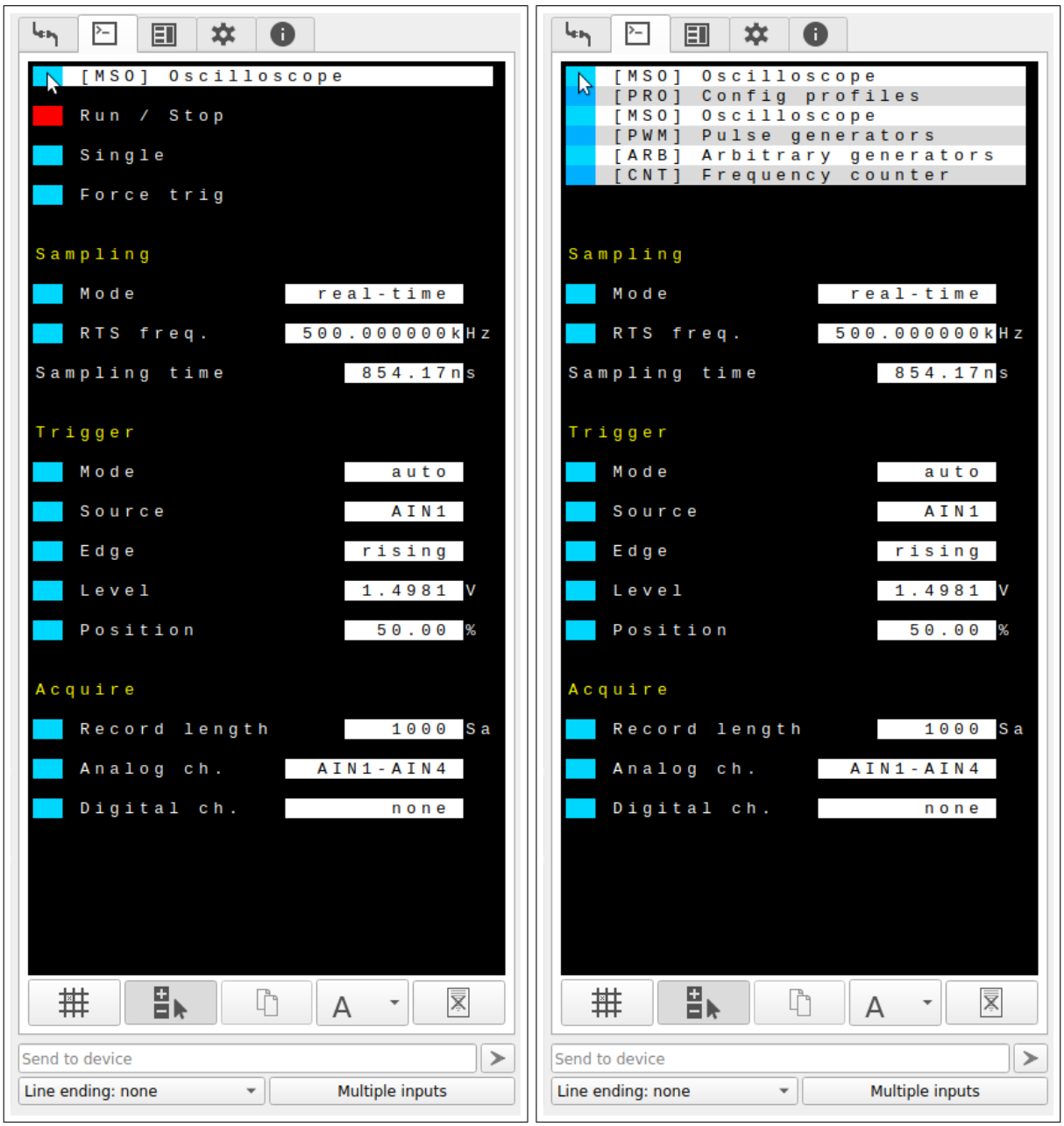

Figure 4: Oscilloscope tab of the TUI (left), with tab menu opened (right)

Metric unit prefixes are supported for numeric values, consisting of a single character at the very end of the value, e.g. "12.345k". The available prefixes are listed in the following table. Characters other than digits, metric prefix characters and the decimal point are ignored.

Prefix	nano	micro	mili	-	kilo	mega	giga
Character	"n"	"u"	"m"	" "	"k"	"M"	"G"
Multiplier	10^{-9}	10^{-6}	10^{-3}	1	10^{3}	10^{6}	10^{9}

Table 1: Supported metric unit prefixes

In many cases, the exact parameter value entered by the user cannot be achieved. The firmware then sets the closest possible value, which is shown in the TUI. However, the originally entered value is saved and the firmware will attempt to achieve a closer match if the instrument configuration is changed later.

2.1 Value editing

Parameter values can be edited in multiple ways, depending on their type. The supported types are:

- **Toggle settings**: For settings with only a few discrete options, e.g. "on"/"off", clicking the edit button repeatedly cycles through the available options. The new value is applied immediately, there is no confirmation.

Figure 5: Example of a toggle setting ("Mode" setting of the oscilloscope trigger)

- **Drop-down menu**: For settings with more discrete options, a drop-down menu is opened underneath the parameter's value block when the edit button is clicked. This menu lists all the available options, each with a button on the left side. Clicking one of these buttons selects the corresponding option, applies it immediately and closes the menu. The menu is also closed when the user clicks any parameter edit button, keeping the old value.

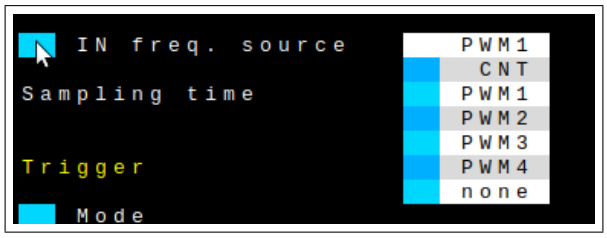

Figure 6: Example of the drop-down menu ("IN freq. source" setting of the oscilloscope)

- **Numeric editor**: A digit-wise editor displayed in a row under the value block is provided for numeric values. The "<" and ">" buttons select a digit in the value. The "+" and "-" buttons then increment/decrement the selected digit or cycle through available metric prefixes if the rightmost digit position is selected. The new value is saved when the user clicks any parameter edit button. There may also be a number of presets available below, which work identically to the drop-down menu described above.

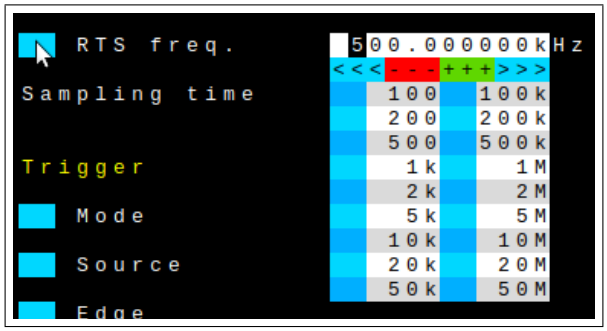

Figure 7: Example of numeric editor with presets ("RTS freq." setting of the oscilloscope)

- **Direct value entry (text field)**: Lastly, it is also possible to enter values directly using the "Send to device" function of Data Plotter located below the terminal window. This text field is pre-filled with the current parameter value when the value editor (drop-down or numeric) is opened.

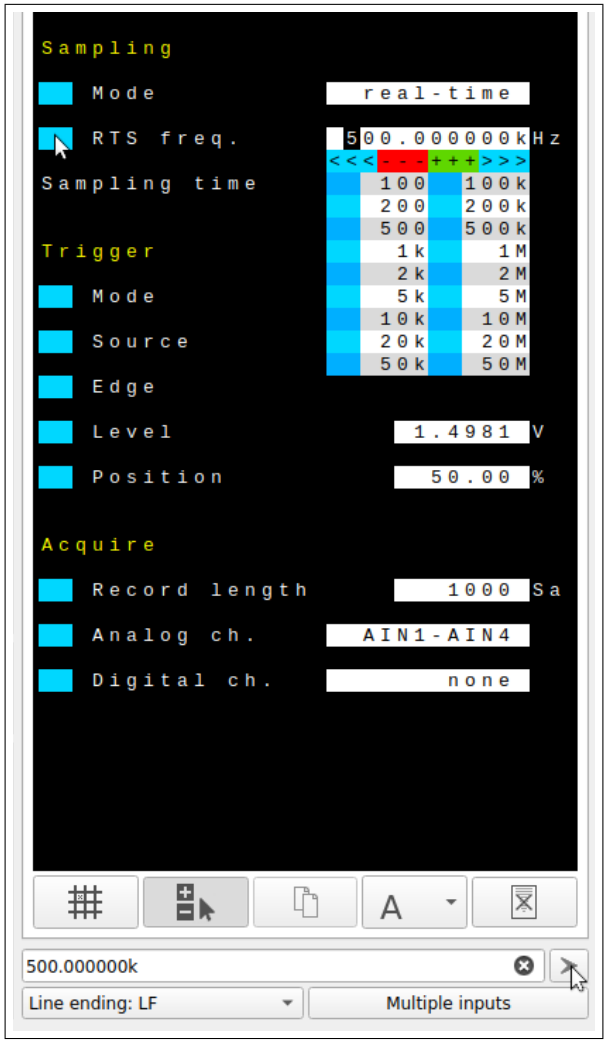

Figure 8: Pre-filled text field of the "Send to device" function for direct value entry

3 Mixed-signal oscilloscope (MSO)

A Mixed-Signal Oscilloscope (MSO) with 4 analog channels and up to 16 digital channels is implemented. Its exact capabilities depend on the MCU used. The Analog-to-Digital Converters (ADCs) embedded in the MCU are used for the analog channels. General-Purpose Input/Output (GPIO) pins in input mode are used for the digital channels. Both the analog and digital channel sampling is triggered by an embedded timer at a precise sampling frequency. Samples are transferred by DMA controllers into buffers in internal SRAM memory. After an acquisition is completed, the buffer contents are sent to the PC in order to display the acquired data in Data Plotter. Both real-time and equivalent-time sampling are implemented.

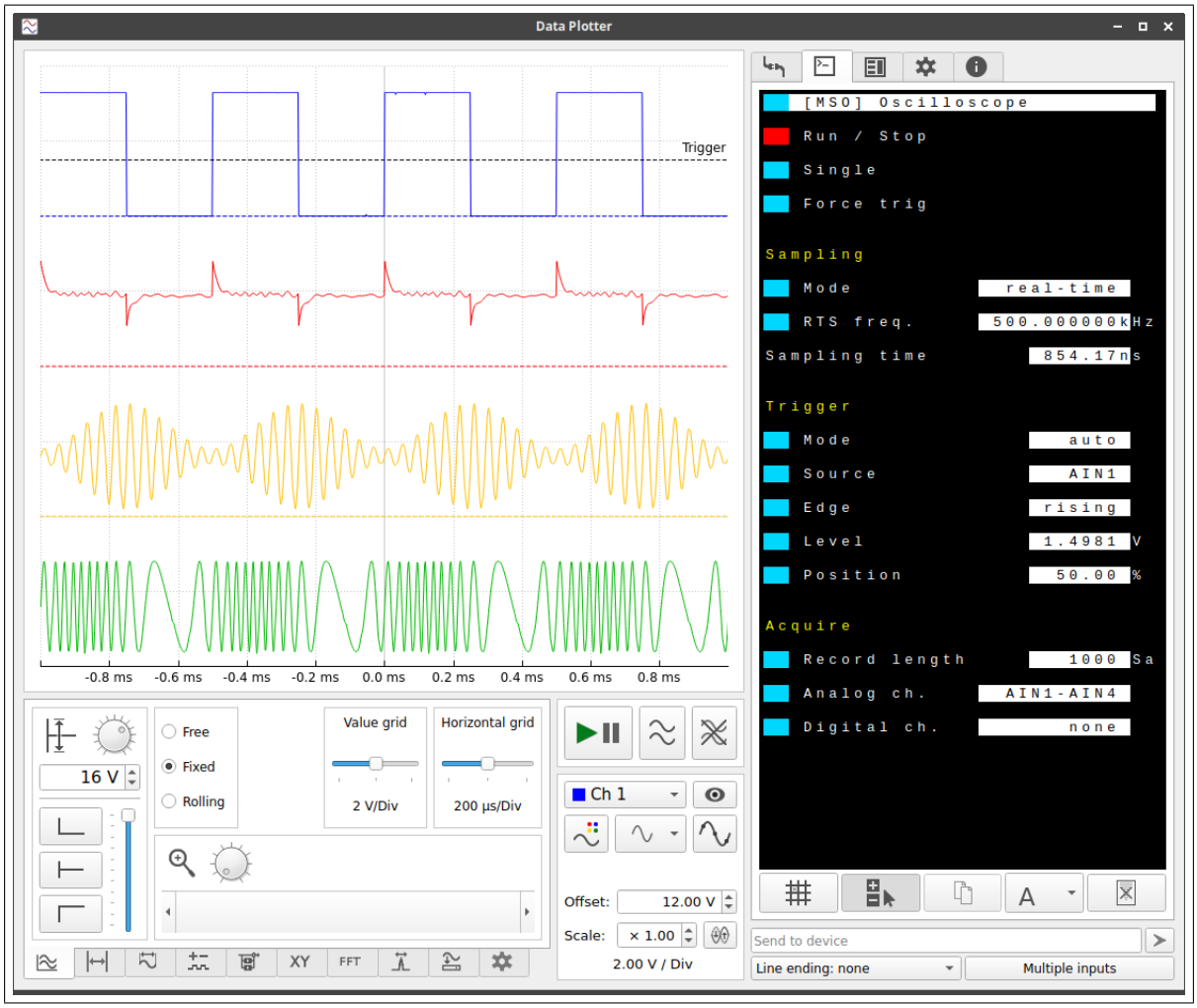

Figure 9: Oscilloscope being used within Data Plotter

Note that the acquisition is paused whenever the user is interacting with the TUI. This includes the time it takes to process a button click, as well as the entire time the value editor is open. This applies even when interacting with a different instrument's tab. This restriction is necessary to ensure communication with the PC does not disturb the operation of the oscilloscope at high sampling rates. Once the setting is stored and editor closed, the acquisition is automatically restarted.

In RTS mode, the ADCs sample the inputs at a sampling frequency f_{SR} set by the "RTS freq." parameter. The maximum sampling rate in this mode is not only limited by the maximum ADC sampling rate, but also other factors such as DMA throughput. For example, the maximum sampling rate is lower when an arbitrary generator is enabled. If the user enables more analog channels (N_C) than there are ADCs available (N_A):

- Each ADC samples multiple input channels alternately.
- The maximum sampling frequency is divided by a factor of

$$N_{alt} = \frac{N_C}{N_A}. \tag{1}$$

- Channel skew (delay) of no more than

$$T_{skew} = \frac{T_{SR}}{N_{alt}} = \frac{1}{N_{alt} \cdot f_{SR}} \tag{2}$$

  is introduced between channels sampled by the same ADC.

In summary:

- For MCUs with 4 ADCs ($N_A = 4$):
  - There is no sampling rate reduction nor channel skew.
- For MCUs with 2 ADCs ($N_A = 2$):
  - For 1 or 2 enabled analog channels ($N_C \leq 2,\ N_{alt} \leq 1$):
    - There is no sampling rate reduction nor channel skew.
  - For 4 enabled analog channels ($N_C = 4,\ N_{alt} = 2$):
    - Max. sampling rate is reduced to 1/2 of the max. sampling rate for $N_C \leq 2$.
    - Channels 1 and 2 are sampled simultaneously.
    - Channels 3 and 4 are sampled simultaneously.
    - Channels 3 and 4 are are delayed from channels 1 and 2 by at most 1/2 of the sampling period.
- For MCUs with 1 ADC ($N_A = 1$):
  - For 1 enabled analog channel ($N_C = 1,\ N_{alt} = 1$):
    - There is no sampling rate reduction nor channel skew.
  - For 2 enabled analog channels ($N_C = 2,\ N_{alt} = 2$):
    - Max. sampling rate is reduced to 1/2 of the max. sampling rate for $N_C = 1$.
    - Channel 2 is delayed from channel 1 by at most 1/2 of the sampling period.
  - For 4 enabled analog channels ($N_C = 4,\ N_{alt} = 4$):
    - Max. sampling rate is reduced to 1/4 of the max. sampling rate for $N_C = 1$.
    - Each channel is delayed from the previous one by at most 1/4 of the sampling period.

3.1 Interleaving ADCs

If there are more available ADCs than enabled analog channels ($N_A > N_C, N_{alt} < 1$), ADC interleaving can be used to achieve a higher sampling rate. This is activated automatically when the user selects a real-time sampling frequency that is beyond the capabilities of a single ADC. The "Interleaving" parameter appears, displaying the number of ADCs interleaved per analog channel, e.g. "2 ADCs/ch". If the sampling frequency is subsequently decreased, interleaving is disabled again.

In some MCUs, the firmware cannot connect the input pins of analog channels to all the interleaving ADCs. Therefore, external jumper links between the analog input pins are required. In that case, a "Links needed" parameter appears, listing the required connections. These are shown with a yellow background to alert the user to the necessity of connecting the jumper wires. If the user fails to do so, samples of different input signals will be combined into one channel, producing garbled waveform data.

3.1.1 Trigger limitations

When normal trigger mode from an analog channel is used, only one ADC can use the analog watchdog triggering feature due to limited AHB bandwidth and core speed of the MCU. Therefore, while the effective sampling rate f_S is higher due to using

$$N_{int} = \frac{1}{N_{alt}} \tag{3}$$

interleaving ADCs per analog channel, the rate at which the trigger can detect signal transitions is still equal to the sampling rate f_{S1} of a single ADC, i.e.

$$f_{S1} = \frac{f_S}{N_{int}}. \tag{4}$$

Therefore, trigger aliasing (using 1 ADC only) will occur before waveform aliasing (using multiple ADCs). Aliasing occurs when the sampling frequency is at or below the Nyquist frequency. When f_{S1} is exactly the Nyquist frequency, all the samples from the trigger ADC will be constant and no trigger will occur at all. When f_{S1} is below the Nyquist frequency, trigger events will happen at a reduced rate, missing some valid signal transitions.

3.2 Digital channel skew

The digital channels may be slightly delayed with respect to analog channels. The digital data is captured simply by a DMA transfer from the GPIO input data register, there is no sample and hold mechanism. Ongoing transfers of ADC data for analog channels or DAC data for arbitrary generators occupy the AHB bus and cause delays of variable durations. Unfortunately, there is no way to alleviate this without additional input latch circuitry.

3.3 Equivalent-time sampling (ETS) mode

Equivalent-time sampling (ETS) mode is supported for periodic input signals. Their frequency f_{IN} must be known for this implementation. The input frequency can be measured by the frequency counter or manually entered. The input signal can also be synchronous with an output signal of one of the available generators, the frequency of which is known. Then, an equivalent-time sampling rate f_{SE} higher than the maximum f_{SR} can be achieved by acquiring one sample per N input signal periods according to

$$T_{SR} = N \cdot T_{IN} + T_{SE} \iff f_{SR} = \frac{f_{IN} \cdot f_{SE}}{f_{IN} + N \cdot f_{SE}} \tag{5}$$

$$T_{SE} = T_{SR} - N \cdot T_{IN} \iff f_{SE} = \frac{f_{IN} \cdot f_{SR}}{f_{IN} - N \cdot f_{SR}} \tag{6}$$

where T_{IN}, T_{SR}, T_{SE} are the input, RTS and ETS periods respectively.

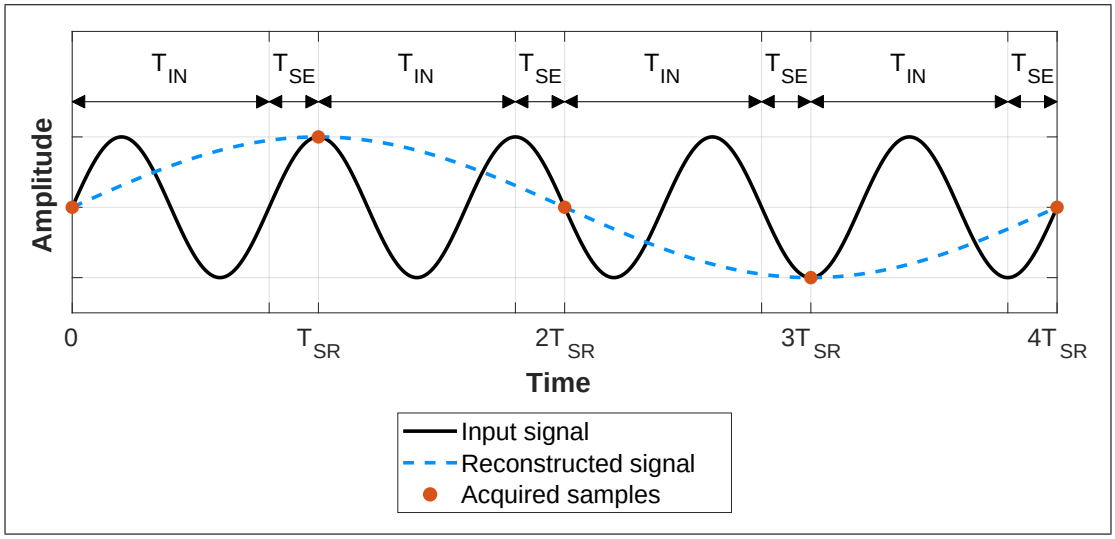

Figure 10: Equivalent-time sampling (ETS) as implemented in this firmware. Sampling at an equivalent of 4 samples per period with $N = 1$.

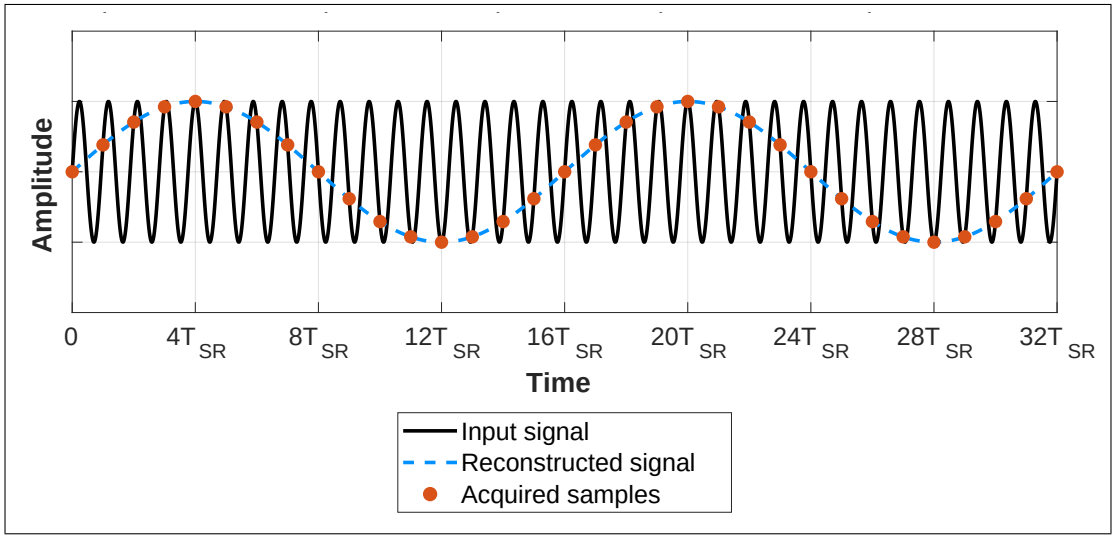

Figure 11: Equivalent-time sampling (ETS) at an equivalent of 16 samples per period with $N = 1$.

3.3.1 Trigger limitations (digital channels)

Input signal edge detection of the EXTI controller is used for digital channel trigger. In ETS mode, this will cause detection events to happen between samples, stopping oscilloscope acquistion prematurely. In normal trigger mode, if the selected edge cannot be found in the acquired data, the oscilloscope discards it and starts a new acquistion without updating the waveform displayed. This would cause the oscilloscope to appear frozen if a digital channel is used as trigger source in ETS mode with normal trigger. Therefore, there are two options for triggering in ETS mode:

- Enable at least one analog channel and use it as the trigger source. Both normal and auto trigger are available and work normally.
- If a digital channel is used, only auto trigger is available. However, input signal transitions are not detected, trigger only occurs after a number of full buffers have been acquired. For this reason, the trigger edge selection is hidden for digital channel trigger in ETS mode.

3.4 MSO TUI tab: Main controls

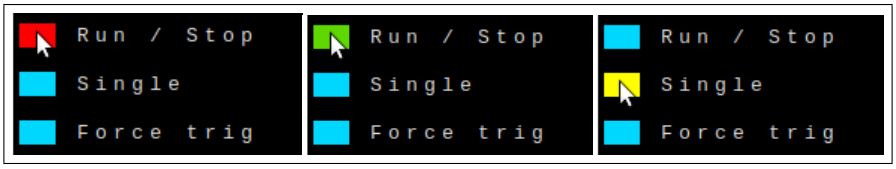

Figure 12: Main oscilloscope controls in Stop, Run and Single mode (from left to right)

3.4.1 Run/Stop

Toggling this setting starts and stops the oscilloscope acquisitions. The button is red in Stop mode, green in Run mode and blue in Single mode.

3.4.2 Single

Clicking this button enables single mode for the next acquisition, reverting to Stop mode afterwards. The button is yellow when single mode is active and waiting for trigger, otherwise blue.

3.4.3 Force trig

Forces a single trigger event to occur when clicked.

3.5 MSO TUI tab: "Sampling" menu

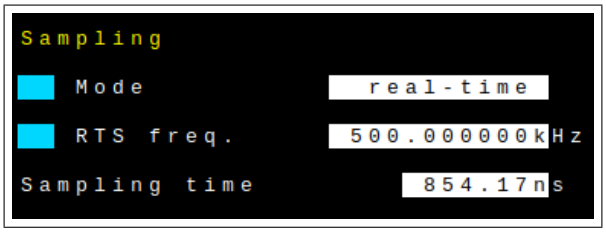

Figure 13: Sampling menu in RTS mode without ADC interleaving

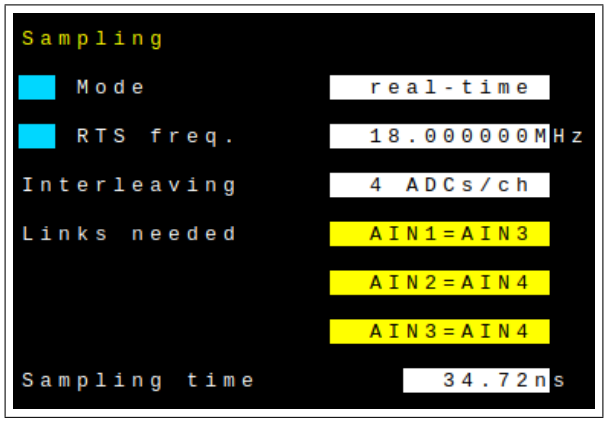

Figure 14: Sampling menu in RTS mode with active ADC interleaving

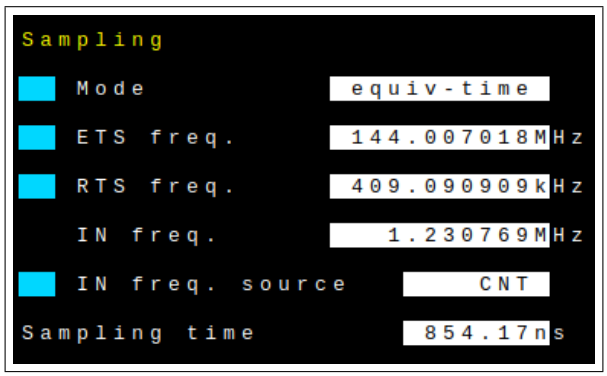

Figure 15: Sampling menu in ETS mode

3.5.1 Mode

- "real−time": Oscilloscope uses real-time sampling (RTS mode).
- "equiv−time": Oscilloscope uses equivalent-time sampling (ETS mode).

3.5.2 RTS freq.

Real-time sampling frequency, in Hertz. This is the effective sampling rate including interleaving of ADCs (each individual ADC samples at the corresponding fraction of this frequency when interleaving is active).

3.5.3 Interleaving

Currently active interleaving mode. Only shown when interleaving is active. Selected automatically based on the sampling rate and number of enabled analog channels, not user-configurable. The options are:

- "2 ADCs/ch": There are 2 ADCs sampling every enabled analog channel.
- "4 ADCs/ch": There are 4 ADCs sampling every enabled analog channel.

3.5.4 Links needed

A list of jumper links necessary for the currently active interleaving mode. A link connecting analog input pins "AIN X" and "AIN Y" is denoted as "AINX=AINY". Only shown when interleaving is active and the analog inputs cannot be connected by firmware.

3.5.5 ETS freq.

Equivalent-time sampling frequency, in Hertz. Only applicable and visible in ETS mode. This is the equivalent sampling rate for an input signal with frequency equal to the value of the "IN freq." parameter. There are 2 ways this parameter can work:

- The user can set the "ETS freq." parameter directly. The firmware then finds an RTS frequency such that the resulting ETS frequency is as close as possible to the entered value. If the input frequency changes, the firmware readjusts the RTS frequency to maintain the desired ETS frequency.
- Alternatively, the user can set the "RTS freq." parameter. In that case, the "ETS freq." parameter only shows the calculated ETS frequency, based on the input frequency. If the input frequency changes, the RTS frequency remains unchanged, while the ETS frequency is recalculated.

3.5.6 IN freq.

Input signal frequency, in Hertz. Only applicable and visible in ETS mode. This parameter can be set manually when the input frequency source ("IN freq. source" parameter) is set to "none". Otherwise, it updates automatically from the selected source and can't be changed by the user.

3.5.7 IN freq. source

Source of input signal frequency with the following options:

- "CNT": Input signal frequency measured by the frequency counter. Automatically updated at the counter's update rate (approx. 1 second).
- "PWMX": Input signal frequency is set equal to the pulse generator "X" frequency. Automatically updated whenever the user changes the generator's frequency.
- "ARBX": Input signal frequency is set equal to the arbitrary generator "X" frequency. Automatically updated whenever the user changes the generator's frequency.
- "none": No automatically updated source. Input signal frequency can be set manually in the "IN freq." parameter.

3.5.8 Sampling time

The ADC sampling time, in seconds. This is the duration of the sampling phase of the ADC, during which the sampling capacitor charges from the input pin. It determines the maximum input signal impedance R_{AIN} for a given ADC resolution. The corresponding values can be found in the MCU datasheet. **The sampling time is automatically set as high as possible for a given sampling rate**, it is not user-configurable.

3.6 MSO TUI tab: "Trigger" menu

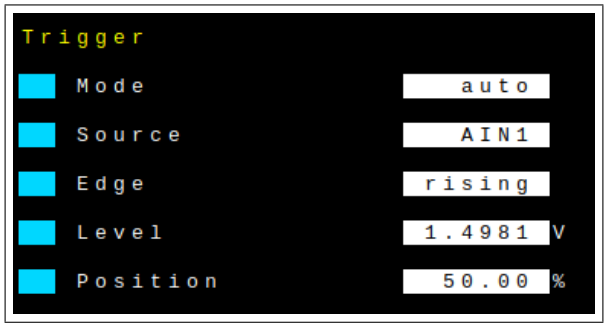

Figure 16: Trigger menu, set to trigger from an analog channel

3.6.1 Mode

- "normal": Trigger when a transition of the input signal occurs.
- "auto": Trigger on a transition, or when 5 full buffers have been acquired.

3.6.2 Source

- "AINX": Analog channel X (input pin AIN X) triggers the oscilloscope.
- "DINX": Digital channel X (input pin DIN X) triggers the oscilloscope.

Only enabled channels are available in the drop-down menu.

3.6.3 Edge

- " rising": Trigger on a rising edge only.
- " falling ": Trigger on a falling edge only.
- "either": Trigger on either a rising or falling edge.

3.6.4 Level

Vertical trigger level, in Volts. Hidden when triggering from a digital channel.

3.6.5 Position

Horizontal position of the trigger (pre-trigger) as a percentage of the record length.

3.7 MSO TUI tab: "Acquire" menu

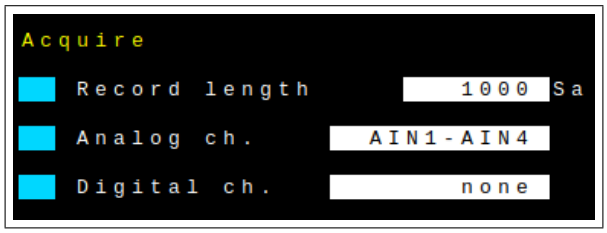

Figure 17: Acquire menu

3.7.1 Record length

Record length, as the number of samples acquired.

3.7.2 Analog ch.

Selection of enabled analog channels, the options are:

- "none": All analog channels are disabled. The digital channels must be enabled in this case.
- "AIN1": Only analog channel 1 is enabled.
- "AIN1,AIN2": Analog channels 1 and 2 are enabled.
- "AIN1−AIN4": All analog channels (1 to 4) are enabled.

3.7.3 Digital ch.

Selection of enabled digital channels, the options are:

- "none": All digital channels are disabled. At least one analog channel must be enabled in this case.
- "DIN0−DINX": All digital channels (0 to X) are enabled.

4 Pulse generators (PWM)

One or more pulse generators (depending on MCU capabilities) are also provided by this firmware. These can be used for general-purpose tasks or in conjunction with the oscilloscope in ETS mode, generating reference signals with precisely known frequencies. All generated frequencies are fractions of the MCU timer clock frequency f_{TIM} (typically equal to the system clock frequency f_{SYS}). Therefore, the available frequency resolution is diminished when approaching the timer clock frequency. Theoretically, the maximum output frequency is half the timer clock frequency. However, it may not be possible to drive the corresponding GPIO output pin at that frequency, depending on the load capacitance and other factors – consult the MCU datasheet for more details.

The pulse generators are denoted as "PWM" and numbered ("PWM 1", "PWM 2", etc.). Each generator can have one or more output channels, denoted by a letter suffix ("PWM 1A", "PWM 1B", "PWM 2A", etc.). The output signal's positive duty cycle/pulse width can be set individually for each channel, while the frequency is the same for all channels of a given generator. Each channel's output can be individually disabled, leaving the corresponding GPIO pin floating (high-Z state).

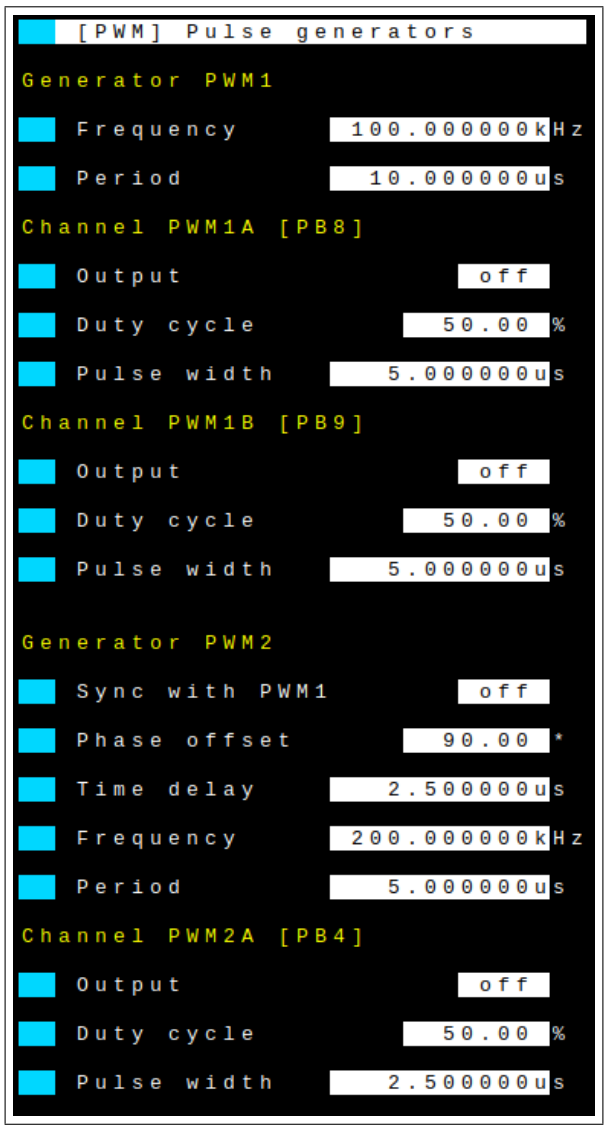

Figure 18: Pulse generator TUI tab

4.1 PWM synchronization

If there are multiple pulse generators available, it is possible to synchronize them in a master/slave scheme. The time delay of the slave generator's output signals w.r.t. the master generator's is configurable from zero (synchronized) up to the period of the master generator. A phase offset based on the period of the master generator can also be set. Additionally, frequency synchronization can be enabled via the "Sync with PWM X" setting. The frequency of the slave generator is then automatically kept equal to the master generator's frequency.

Each generator can be a slave of the previous generator, i.e. the PWM 3 generator can be a slave of PWM 2, which itself can be a slave of PWM 1. **Any number of generators can be synchronized this way, there are no additional unwanted delays introduced.**

4.2 PWM TUI tab: "Generator PWMX" menus

These menus contain all parameters specific to a given PWM X pulse generator.

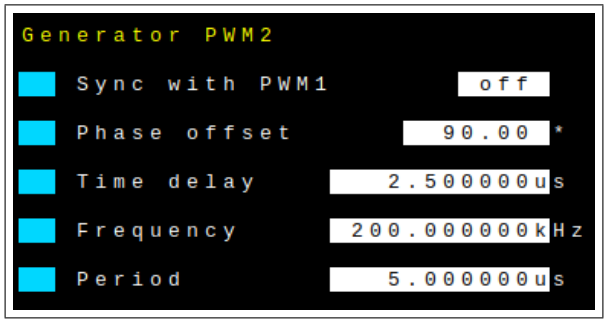

Figure 19: Generator parameters when frequency synchronization is disabled

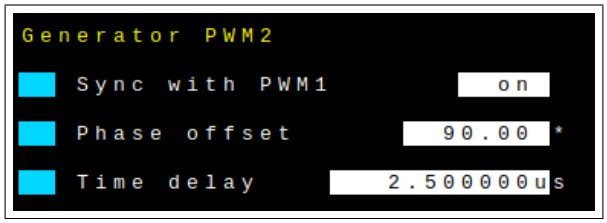

Figure 20: Generator parameters when frequency synchronization is enabled

4.2.1 Sync with PWMX

- "off": Frequency synchronization is disabled. This generator's frequency can be set independently.
- "on": Frequency synchronization is enabled. This generator's frequency is kept equal to the frequency of its master (PWM X).

4.2.2 Frequency

Generator frequency, in Hertz. Only visible when frequency synchronization is disabled.

4.2.3 Period

Generator period, in seconds. Only visible when frequency synchronization is disabled.

4.2.4 Phase offset

Phase offset of the slave generator w.r.t. the master generator, in degrees. Note the phase angle is relative to the period of the *master* generator – i.e. a 360° offset represents a delay of 1 master period.

4.2.5 Time delay

Time delay of the slave generator w.r.t. the master generator, in seconds. Maximum delay is 1 master generator period.

4.3 PWM TUI tab: "Channel PWMXY" menus

These menus contain all parameters specific to a given PWM XY output channel.

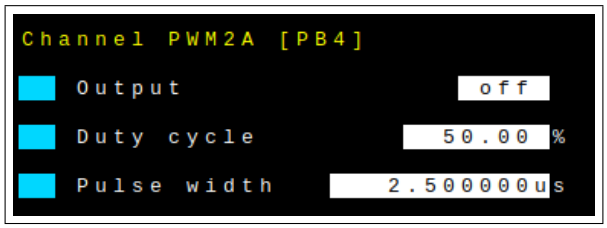

Figure 21: Channel parameters

4.3.1 Output

- "off": Output is disabled, leaving this channel's GPIO output pin floating (high-Z state).
- "on": Output is enabled, driving this channel's GPIO output pin.

4.3.2 Duty cycle

Positive duty cycle of the channel's output signal, as a percentage of the generator period.

4.3.3 Pulse width

Positive pulse width of the channel's output signal, in seconds.

5 Arbitrary generators (ARB)

One or two arbitrary generators are available in firmware versions for MCUs with embedded Digital-to-Analog Converters (DACs). These allow the generation of arbitrary signals as configured by the user. They are denoted as "ARB" and numbered (e.g. "ARB 1", "ARB 2"). The DAC sampling frequency f_S is generated by a timer, same as the pulse generators. The output signal frequency f_O is then

$$f_O = \frac{f_S}{N}, \tag{7}$$

where N is the number of samples per period of generated signal. The minimum number of samples per period is set to **4**, giving a maximum generated signal frequency of **250 kHz** when considering the typical max. DAC sampling rate of **1 MSa/s**. The max. number of samples per period (at low output signal frequencies) is typically **1000**, but it may depend on the firmware version. Between these extremes, the number of samples is automatically set to the highest value possible (not user-configurable).

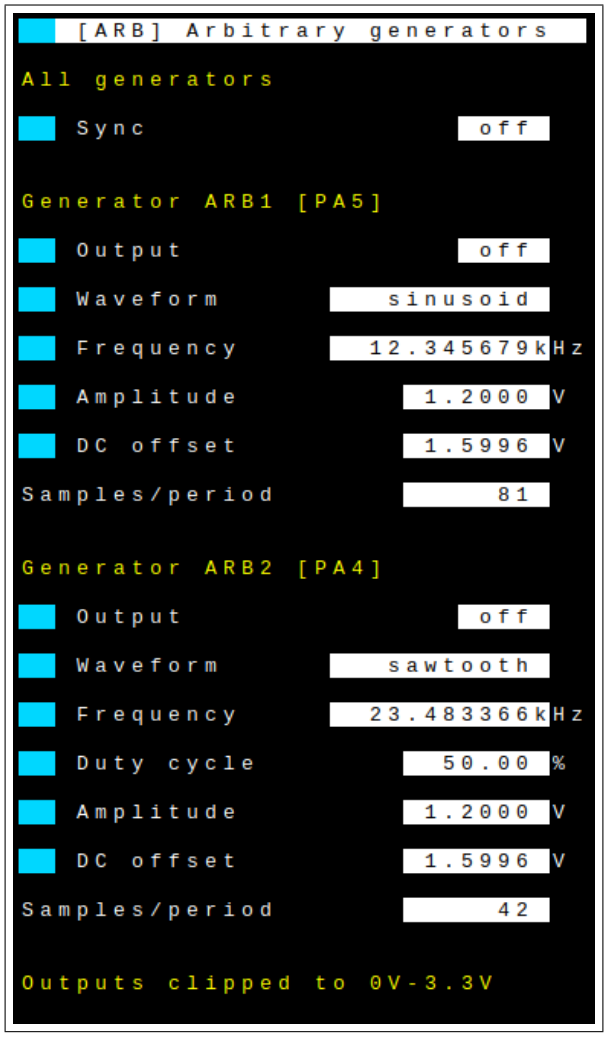

Figure 22: Arbitrary generator TUI tab

5.1 Custom waveforms

The arbitrary generators provide a number of built-in waveform functions (sine wave, sawtooth, etc.), but custom waveforms can also be used. These have to be loaded from the PC using a wave file in accordance with the following format:

- A comma-separated list of numeric values (.csv).
- Each value represents a single sample of the output signal, in Volts.
- The length of a single value (excl. comma) cannot exceed 62 characters.
- Metric prefix characters (e.g. '123m' for 123 mV) are supported.
- Characters other than commas, digits, metric prefix characters and the decimal point are ignored.
- Values will be clipped to remain within the generator's output voltage range.
- If there are more values than can be used by the generator, they will be ignored.

After a waveform has been loaded into the MCU, it can be retained between MCU resets by saving a configuration profile to the internal FLASH memory – see section 7.

5.2 ARB Synchronization

If there are two available arbitrary generators, they can be synchronized using the "Sync" setting. When enabled, both generators share a common frequency with an identical number of samples per period. The initial phase of both generators can then be set. Its resolution is determined by the number of samples per period, giving a worse resolution at higher frequencies (same as duty cycle).

If one of the generators is using a custom waveform, the other generator will be limited to the same number of samples per period. If both are using custom waveforms, the lower number of samples among them will be used for both – truncating the longer waveform.

5.3 ARB TUI tab: "All generators" menu

This menu contains settings affecting all arbitrary generators.

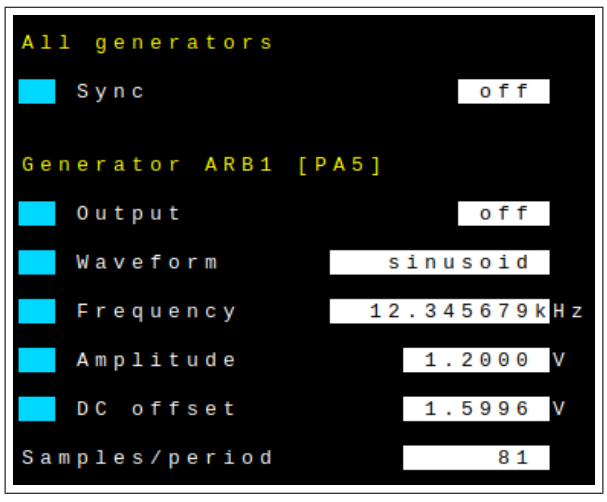

Figure 23: Generator parameters when synchronization is disabled

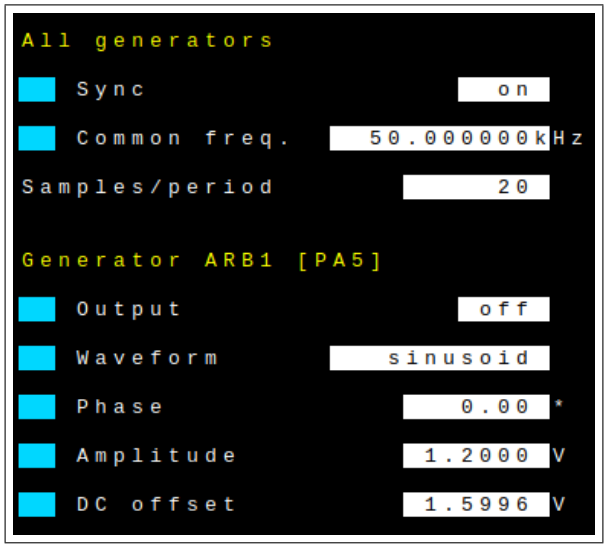

Figure 24: Generator parameters when synchronization is enabled

5.3.1 Sync

- "on": Synchronization is enabled, ARB 1 and ARB 2 frequencies are identical.
- "off": Synchronization is disabled, ARB 1 and ARB 2 frequencies can be set independently.

5.3.2 Common freq.

Common generated signal frequency, in Hertz. Only visible when synchronization is enabled.

5.3.3 Sample count

Common number of samples per period of generated signal. Calculated automatically, not user-configurable.

5.4 ARB TUI tab: "Generator ARBX" menus

These menus contain all parameters specific to a given ARB X arbitrary generator.

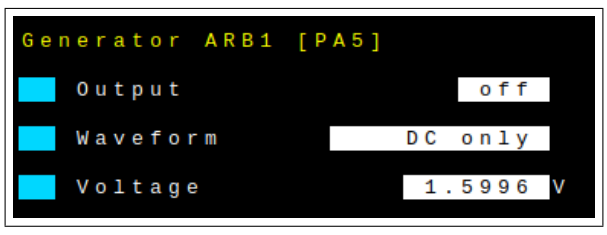

Figure 25: Generator parameters when the DC only waveform is selected

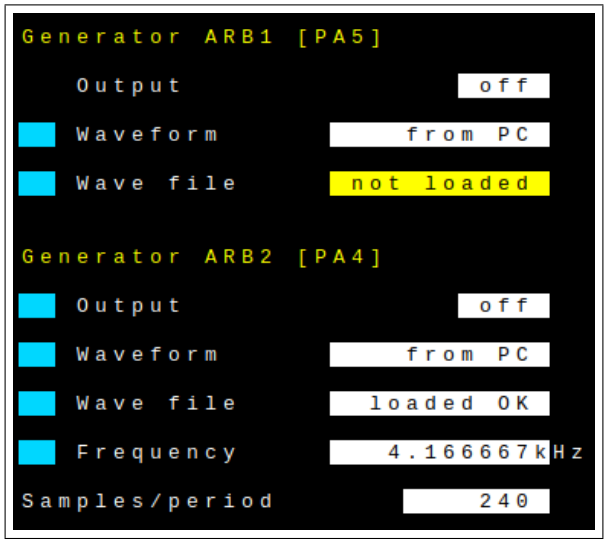

Figure 26: Generator parameters when using a custom waveform, before and after a wave file is loaded from the PC (ARB1, ARB2 respectively).

5.4.1 Output

- "off": Generator output is disabled. The GPIO output pin is left floating (high-Z state).
- "on": Generator output is enabled. The DAC drives the GPIO output pin.

5.4.2 Waveform

- "from PC": A custom waveform can be loaded from the PC.
- "DC only": Only a DC voltage is generated.
- "sinusoid": A sinusoid waveform is generated.
- "sawtooth": A sawtooth waveform with adjustable duty cycle is generated. Set duty cycle to 50 % to generate a symmetric triangle waveform.
- "square": A square waveform with adjustable duty cycle is generated.

5.4.3 Wave file

Only available when custom waveform is selected. Clicking the corresponding button opens a file selection dialog allowing the user to select a wave file to load from the PC. The value block shows the loading status, which can be one of the following:

- "not loaded": No file has been loaded since custom waveform was selected. The generator output cannot be enabled until a file is successfully loaded or the user selects a different waveform type. Highlighted in yellow to alert the user.
- "loading ... ": Wave file loading is in progress.
- "load error": An error occurred while loading the wave file. See Data Plotter's message log for the associated error message.

5.4.4 Frequency

Generated signal frequency, in Hertz. Only available when synchronization is disabled.

5.4.5 Phase

Initial signal phase, in degrees. Only available when synchronization is enabled. Hidden for custom waveform.

5.4.6 Duty cycle

Positive duty cycle of the generated signal, as a percentage of its period. Hidden for sinusoid, DC only and custom waveforms.

5.4.7 Amplitude

AC amplitude of the generated signal, in Volts. **Not** peak-to-peak value. Hidden for DC only and custom waveforms.

5.4.8 DC offset

DC offset of the generated signal, in Volts. Renamed to "Voltage" for DC only waveform. Hidden for custom waveform.

5.4.9 Sample count

Number of samples per period of generated signal. Calculated automatically, not user-configurable. Hidden when synchronization is enabled.

6 Frequency counter (CNT)

A frequency counter is also implemented. It can be used standalone or in conjunction with the oscilloscope in ETS mode, measuring its input signal frequency. The counter is based on two timers:

- The reference timer, clocked by the MCU timer clock (typically same frequency as system clock) to measure the time elapsed.
- The input timer, clocked by the input signal being measured by the counter.

The input signal frequency f_{IN} can be determined after the gate time has elapsed from the overall number of counted periods of both the reference (N_{REF}) and input (N_{IN}) timers by

$$f_{IN} = \frac{N_{IN}}{N_{REF}} \cdot f_{REF}, \tag{8}$$

where f_{REF} is the clock frequency of the reference timer. The nominal gate time set to 1 s, giving a worst-case frequency resolution of 1 Hz. This is also approximately the update rate of the counter. The counter always counts a whole number of input signal periods in order to maintain high accuracy for low-frequency input signals. This introduces a slight variance in the gate time, dependent on the input signal frequency. The frequency counter is always-on and cannot be disabled by the user. It only displays a number of measurements and system parameters, it has no configurable settings.

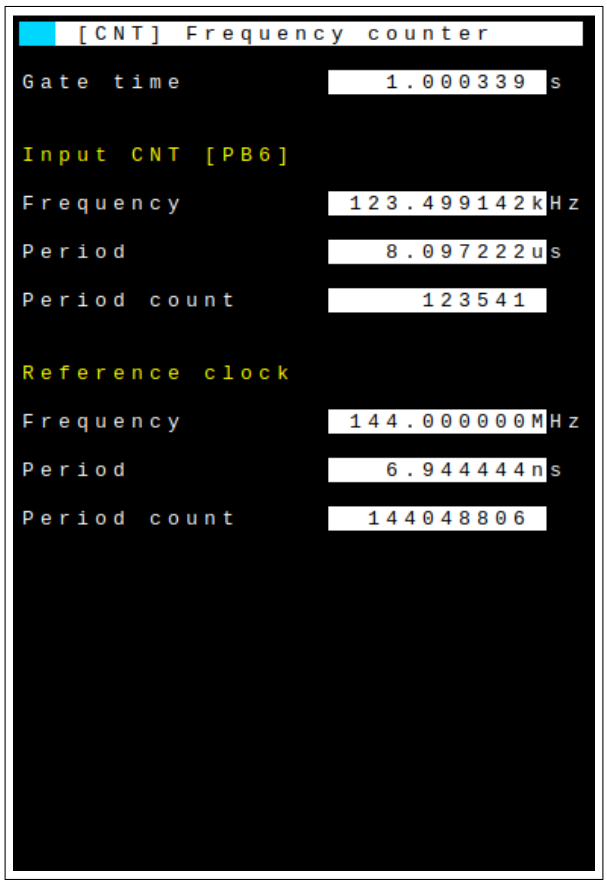

Figure 27: Frequency counter TUI tab

Note that due to the use of digital timer circuitry, the counter may not give correct readings for input signals which do not have clean, fast transitions between the logic low and high level voltages (e.g. slow or noisy analog signals). The counter may also count oscillations that are too fast to be captured by the oscilloscope.

6.1 CNT TUI tab: Gate time

The gate time, in seconds.

6.2 CNT TUI tab: "Input signal CNT" menu

6.2.1 Frequency

The measured frequency of the input signal, in Hertz.

6.2.2 Period

The measured period of the input signal, in seconds.

6.2.3 Period count

The number of input signal periods counted during the gate time.

6.3 CNT TUI tab: "Reference clock" menu

6.3.1 Frequency

The known frequency of the reference timer clock, in Hertz.

6.3.2 Period

The known period of the reference timer clock, in seconds.

6.3.3 Period count

The number of reference clock periods counted during the gate time.

7 Configuration profiles (PRO)

As the previous sections show, there is a large number of instrument parameters the user can configure. It would therefore be useful to make them persistent and avoid having to re-configure all of them manually every time the MCU is reset. To facilitate this, "configuration profiles" containing all the parameters of every instrument are implemented. The current parameter values can be stored in a profile, while previously stored values can be recalled from one.

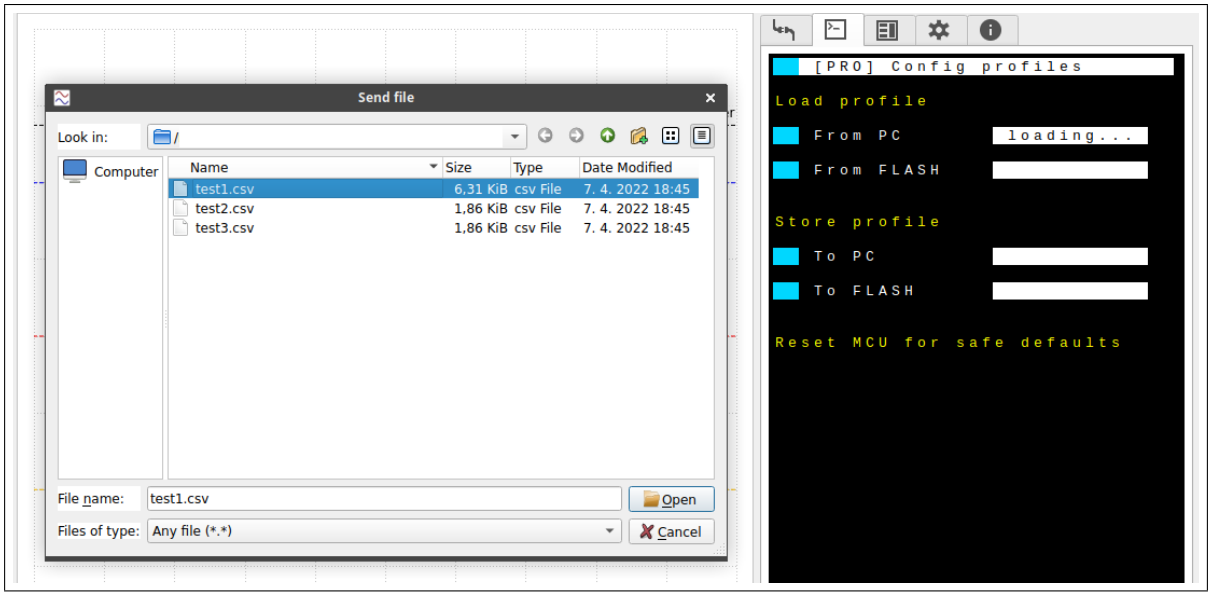

Figure 28: Configuration profiles TUI tab with open file selection dialog

A single profile can be stored in the MCU's internal FLASH memory. It can then be loaded even after an MCU reset or reprogramming. Using Data Plotter, it is also possible to download/upload profiles to/from the connected PC. The profiles are stored as non-human-readable .csv files. Their size is approx. 500 kB + 3 B per sample of arbitrary generator waveform (if using custom waveform).

Each profile contains a firmware version descriptor to ensure compatibility. Therefore, the firmware will reject a profile saved from another MCU platform or firmware release.

7.1 PRO TUI tab: "Load profile" menu

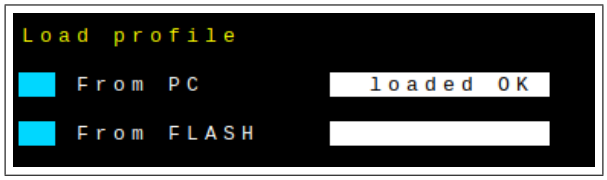

Figure 29: Load profile menu after a profile has been successfully loaded from PC

7.1.1 From PC

Loads instrument parameter values from profile stored in the PC, opening a file selection window.

7.1.2 From FLASH

Loads instrument parameter values from profile stored in the internal FLASH memory.

Value fields show the load operation status:

- (blank): Initial state (no profile loaded since last MCU reset).
- "loading ... ": Profile loading is in progress.
- "loaded OK": Profile has been loaded successfully.
- "error": An error occurred while loading profile. See Data Plotter's message log for the associated error message.

Figure 30: Load profile menu after an error occurred (left), corresponding error message (right)

7.2 PRO TUI tab: "Store profile" menu

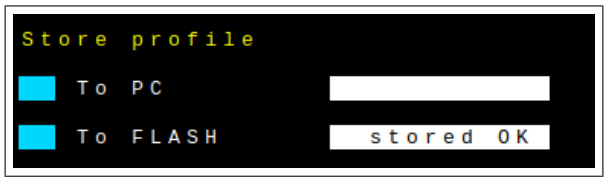

Figure 31: Store profile menu after a profile has been successfully stored to FLASH

7.2.1 To PC

Stores current instrument parameter values into a profile stored in the PC, opening a file selection window.

7.2.2 to FLASH

Stores current instrument parameter values into a profile stored in the internal FLASH memory.

Value fields show the store operation status:

- (blank): Initial state (no profile stored since last MCU reset).
- "storing ... ": Profile storing is in progress.
- "stored OK": Profile has been stored successfully.
- "error": An error occurred while storing profile. See Data Plotter's message log for the associated error message.

8 Versions of VSVI platform (supported MCUs)

8.1 "F303-Nucleo64" for STM32F303RE on Nucleo-F303RE

The primary adaptation of the developed VSVI platform is for the Nucleo-F303RE development board shown in Figure 32. This board is based on the STM32F303RE microcontroller in the LQFP64 package and is one of the most widely used STM32 platforms at FEE CTU. It includes an ST-Link V2-1 debugger/programmer which can be used to program the MCU using the onboard Mini-USB connector. The ST-Link additionally serves as a USB-UART converter, being connected to the USART2 interface of the STM32F303RE MCU. The ST-Link clock derived from an onboard 8 MHz crystal oscillator is also used as the clock source for the STM32F303RE MCU in HSE bypass mode. Figure 33 shows the pinout of this version of the VSVI platform.

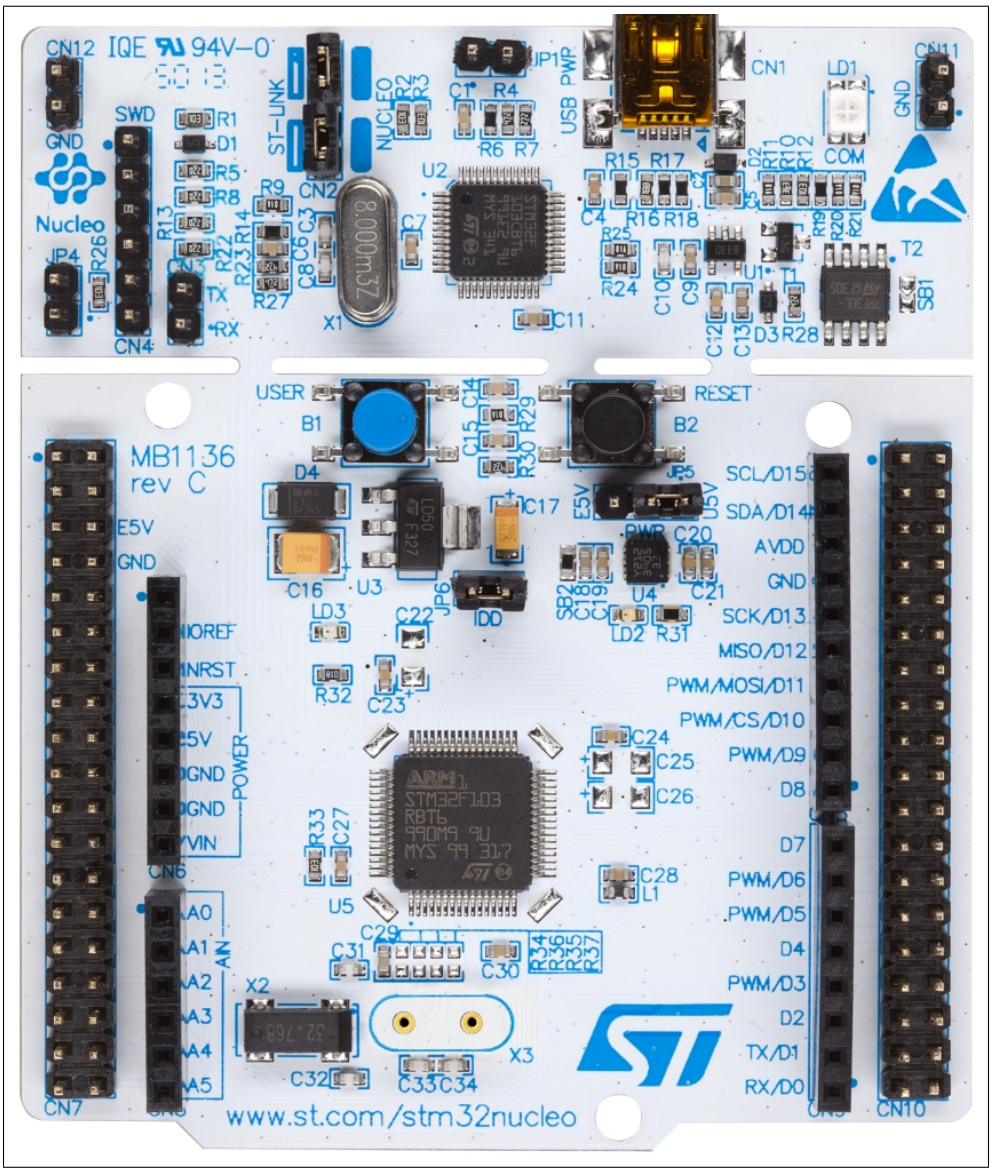

Figure 32: Nucleo-F303RE development board with STM32F303RE MCU

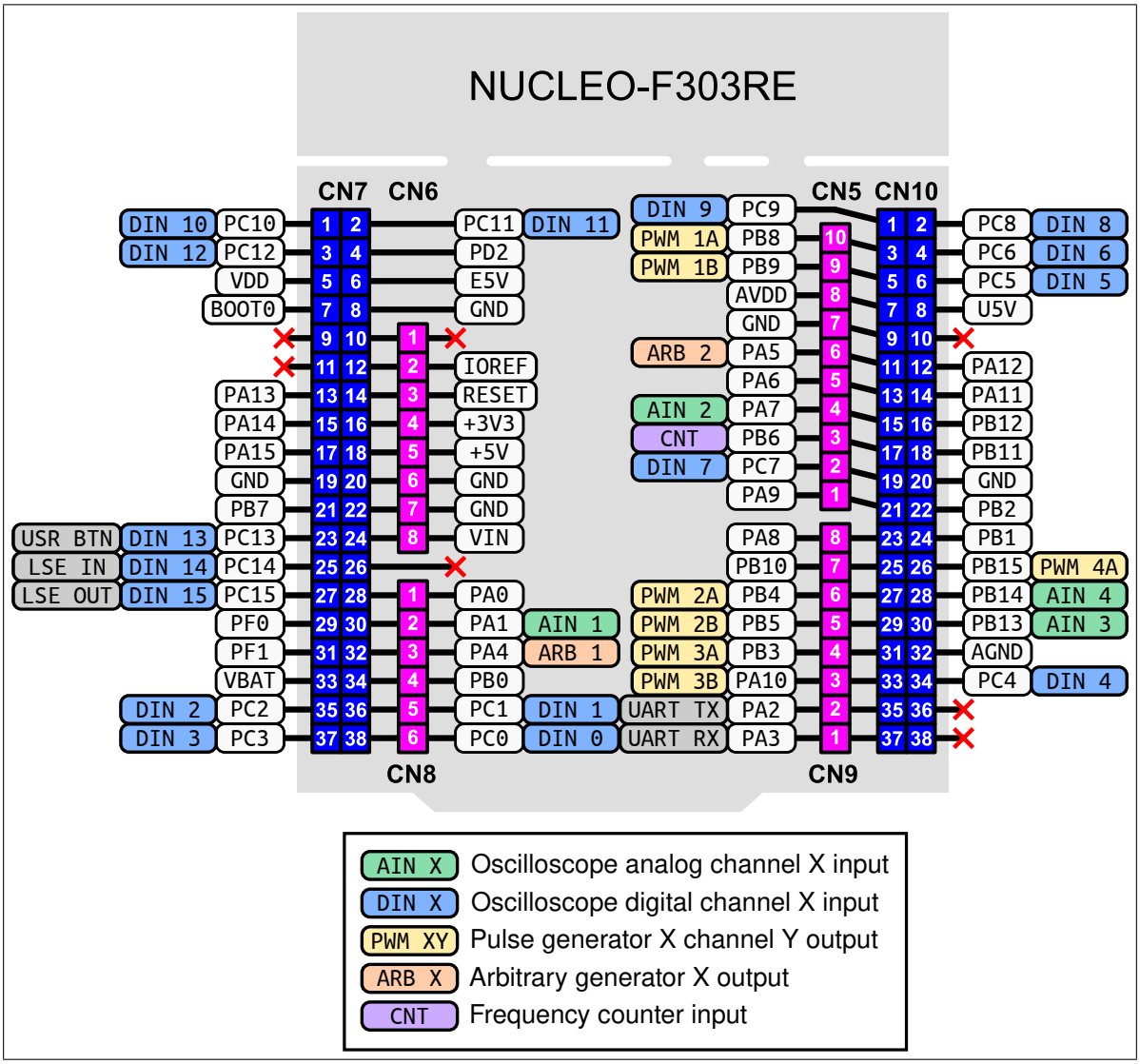

Figure 33: Pinout of SDIs developed for STM32F303RE (Nucleo-F303RE development board)

8.1.1 Mixed-signal oscilloscope (STM32F303RE)

- **4 analog channels** (inputs AIN 1, AIN 2, AIN 3, AIN 4) with **12-bit resolution**
- **16 digital channels** (inputs DIN 0 – DIN 15)
- Maximum real-time sampling rate:
  - **24 MSps** for all digital channels with analog channels disabled
  - **18 MSps** for one analog channel with 4 ADCs interleaving, digital channels disabled
  - **5 MSps** for all digital and analog channels with no interleaving

- Table 2 shows all the possible channel configurations and their respective max. sampling rates when all arbitrary generators are disabled. When any of the arbitrary generators is enabled, Table 3 applies instead.

- Maximum equivalent-time sampling rate:
  - Depends on the input signal frequency
  - Typically up to 72 MSps, max. 720 MSps with 1 sample per 10 input signal periods

Digital \ Analog	**Disabled**	**1 channel**	**2 channels**	**4 channels**
Disabled	-	18.0 MSps (i4) 10.3 MSps (i2) 5.1 MSps (-)	10.3 MSps (i2) 5.1 MSps (-)	5.1 MSps (-)
16 channels	24 MSps (-)	10.3 MSps (i2) 5.1 MSps (-)	10.3 MSps (i2) 5.1 MSps (-)	4.8 MSps (-)

Table 2: Maximum MSO sampling rates for STM32F303RE with arbitrary generators disabled. Used ADC interleaving mode indicated in parentheses ("-" = none, "i2" = 2 ADCs/channel, "i4" = 4 ADCs/channel).

Digital \ Analog	**Disabled**	**1 channel**	**2 channels**	**4 channels**
Disabled	-	7.2 MSps (i2) 4.8 MSps (-)	7.2 MSps (i2) 4.8 MSps (-)	4.0 MSps (-)
16 channels	8 MSps (-)	5.5 MSps (i2) 4 MSps (-)	5.5 MSps (i2) 4 MSps (-)	3.4 MSps (-)

Table 3: Maximum MSO sampling rates for STM32F303RE with arbitrary generators enabled. Used ADC interleaving mode indicated in parentheses ("-" = none, "i2" = 2 ADCs/channel).

- Maximum record length depends on the number of enabled channels according to Table 4

Digital \ Analog	**Disabled**	**1 channel**	**2 channels**	**4 channels**
Disabled	-	30.8 kSa	15.4 kSa	7.7 kSa
16 channels	30.8 kSa	15.4 kSa	10.2 KSa	6.1 kSa

Table 4: Maximum MSO record lengths for STM32F303RE

- **When interleaved sampling of the analog channels is used, the user must connect external jumper links between analog input pins:**
  - When 1 analog channel is enabled and interleaving 2 ADCs per channel, connect:
    - AIN 1 (PA1) and AIN 3 (PB13)

  - When 2 analog channels are enabled and interleaving 2 ADCs per channel, connect:
    * AIN 1 (PA1) and AIN 3 (PB13)
    * AIN 2 (PA7) and AIN 4 (PB14)
  - When 1 analog channel is enabled and interleaving 4 ADCs per channel, connect:
    * AIN 1 (PA1) and AIN 3 (PB13)
    * AIN 2 (PA7) and AIN 4 (PB14)
    * AIN 3 (PA13) and AIN 4 (PB14) or AIN 1 (PA1) and AIN 2 (PA7)

- **Some digital channels are limited on stock Nucleo-F303 boards:**
  - DIN 13 (PC13) is connected to the USER push button, including a 4.7 kΩ pull-up resistor to the 3.3V rail and a bypass capacitor. External signals should not be connected to this pin unless the solder bridge SB17 is removed, disconnecting the button circuitry.
  - DIN 14 and DIN 15 are not available. The corresponding MCU pins PC14 and PC15 are connected to the 32 kHz LSE crystal oscillator and NOT connected to the pin headers. To use these digital channels, R34 and R36 must be removed (disconnecting the crystal) and SB48 and SB49 must be connected (connecting the MCU pins to the pin headers).

8.1.2 Pulse generators (STM32F303RE)

- **4 generators** available:
  - PWM 1 with two channels (PWM 1A, PWM 1B)
  - PWM 2 with two channels (PWM 2A, PWM 2B)
  - PWM 3 with two channels (PWM 3A, PWM 3B)
  - PWM 4 with one channel (PWM 4A)
- All can run independently or be synchronized with adjustable phase/time delay
- Maximum output signal frequency 72 MHz (50% duty cycle only)

8.1.3 Arbitrary generators (STM32F303RE)

- **2 generators** available:
  - ARB 1
  - ARB 2
- Both can run independently or be synchronized with adjustable phases
- Maximum sampling frequency 1 MHz
- Maximum output signal frequency 250 kHz (4 samples per period)
- Up to 1000 samples per period for output signal frequencies $\leq$ 1 kHz

8.1.4 Frequency counter (STM32F303RE)

- Input signal frequency range: 10 Hz − 72 MHz
- Gate time $\approx$ 1 s gives update rate $\approx$ 1 Hz
- Frequency resolution $\leq$ 1 Hz using modified frequency ratio measurement (see section 6)

8.2 "G431-LQFP32" for STM32G431KB on LQFP32 adapter

The VSVI platform was also adapted for the STM32G431KB microcontroller in the LQFP32 package. The embedded USB peripheral is used for communication with the PC. An external 3.3 V voltage regulator is needed to supply the VDD supply voltage. An 8 MHz crystal oscillator should also be connected to the HSE IN, HSE OUT pins with the appropriate load capacitors. Using this crystal oscillator significantly improves clock stability and the frequency accuracy of the generators and frequency counter. It is also absolutely necessary for MSO equivalent-time sampling of externally-generated signals. In case no crystal oscillator is detected at MCU startup, the internal RC oscillator (HSI) is used instead. Figure 34 shows the pinout of the VSVI platform for this MCU, as soldered on an LQFP32-to-DIP adapter for use in breadboards.

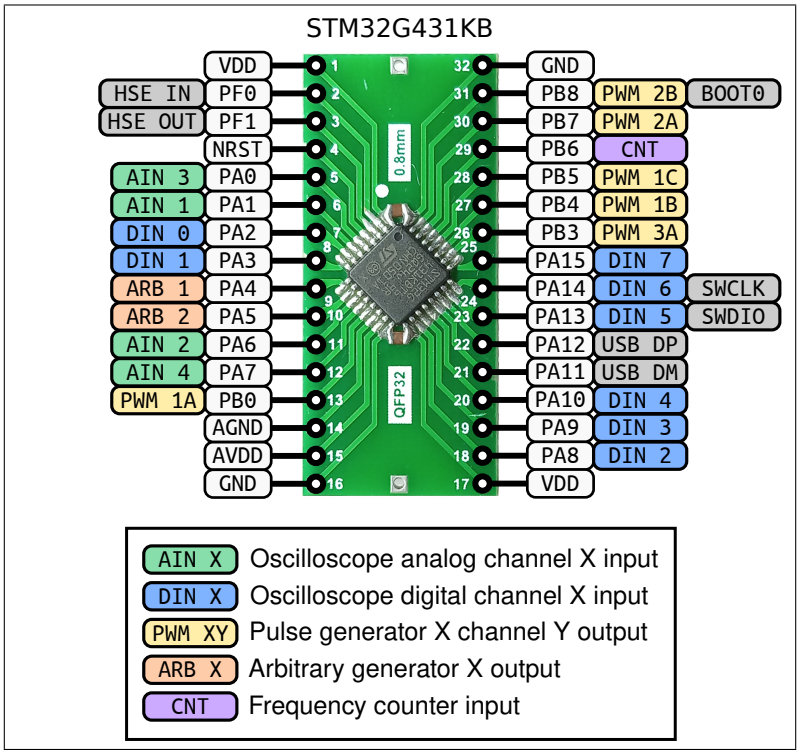

Figure 34: Pinout of SDIs developed for STM32G431KB (LQFP32-to-DIP adapter)

8.2.1 Mixed-signal oscilloscope (STM32G431KB)

- **4 analog channels** (inputs AIN 1, AIN 2, AIN 3, AIN 4) with **12-bit resolution**
- **8 digital channels** (inputs DIN 0 – DIN 7)
- Maximum real-time sampling rate:
  - **20.8 MSps** for all digital channels with analog channels disabled
  - **6.5 MSps** for one analog channel with 2 ADCs interleaving, digital channels disabled
  - **1.7 MSps** for all digital and analog channels with no interleaving
  - Table 5 shows all the possible channel configurations and their respective max. sampling rates when all arbitrary generators are disabled. When any of the arbitrary generators is enabled, Table 6 applies instead.
- Maximum equivalent-time sampling rate:
  - Depends on the input signal frequency
  - Typically up to 104 MSps, max. 1.04 GSa/s with 1 sample per 10 input signal periods

Digital \ Analog	Disabled	1 channel	2 channels	4 channels
Disabled	-	6.5 MSps (i2) 3.4 MSps (-)	3.4 MSps (-)	1.7 MSps (-)
8 channels	20.8 MSps (-)	6.5 MSps (i2) 3.4 MSps (-)	3.4 MSps (-)	1.7 MSps (-)

Table 5: Maximum MSO sampling rates for STM32G431KB with arbitrary generators disabled. Used ADC interleaving mode indicated in parentheses ("-" = none, "i2" = 2 ADCs/channel).

Digital \ Analog	Disabled	1 channel	2 channels	4 channels
Disabled	-	6.5 MSps (i2) 3.4 MSps (-)	3.4 MSps (-)	1.7 MSps (-)
8 channels	13.0 MSps (-)	6.5 MSps (i2) 3.4 MSps (-)	3.4 MSps (-)	1.7 MSps (-)

Table 6: Maximum MSO sampling rates for STM32G431KB with arbitrary generators enabled. Used ADC interleaving mode indicated in parentheses ("-" = none, "i2" = 2 ADCs/channel).

- Maximum record length depends on the number of enabled channels according to Table 7

Digital \ Analog	Disabled	1 channel	2 channels	4 channels
Disabled	-	9.2 kSa	4.6 kSa	2.3 kSa
8 channels	9.2 kSa	4.6 kSa	3.1 KSa	1.8 kSa

Table 7: Maximum MSO record lengths for STM32G431KB

- **When interleaved sampling of analog channel 1 is used, the user must connect external jumper links between the following analog input pins:**
  - AIN 1 (PA1) and AIN 2 (PA6)

8.2.2 Pulse generators (STM32G431KB)

- **3 generators** available:
  - PWM 1 with three channels (PWM 1A, PWM 1B, PWM 1C)
  - PWM 2 with two channels (PWM 2A, PWM 2B)
  - PWM 3 with one channel (PWM 3A)
- All can run independently or be synchronized with adjustable phase/time delay
- Maximum output signal frequency 52 MHz (50% duty cycle only)

8.2.3 Arbitrary generators (STM32G431KB)

- **2 generators** available:
  - ARB 1
  - ARB 2
- Both can run independently or be synchronized with adjustable phases
- Maximum sampling frequency 1 MHz
- Maximum output signal frequency 250 kHz (4 samples per period)
- Up to 1000 samples per period for output signal frequencies ≤ 1 kHz

8.2.4 Frequency counter (STM32G431KB)

- Input signal frequency range: 10 Hz $-$ 52 MHz
- Gate time ≈ 1 s gives update rate ≈ 1 Hz
- Frequency resolution ≤ 1 Hz using modified frequency ratio measurement (see section 6)

9 Known issues

9.1 Firmware freezes when configuration profile loading from PC is cancelled

Description

When the user cancels the file selection dialog, no data is received by the MCU. The firmware then keeps waiting to receive data indefinitely, appearing frozen. Since it is impossible to distinguish between TUI commands and file contents sent from the PC, the firmware cannot recover from this condition and an MCU reset is needed.

This does not apply when storing a profile to PC. In that case, the data is transferred to the PC first, before the file selection dialog is shown to the user. If the user cancels it, the data is discarded by Data Plotter.

Encountered in Data Plotter **release 17.3.2022 and earlier**, will be fixed in the next release.

Workaround

Do not cancel the file selection dialog when loading a profile from PC. Otherwise, reset MCU to recover.

9.2 TUI appears truncated after Data Plotter window is enlarged

Description

When the Data Plotter window is resized, the terminal window expands in height but its active area where the TUI is displayed does not. The TUI then appears truncated.

Encountered in Data Plotter **release 17.3.2022 and earlier**, will be fixed in the next release.

Workaround

It is still possible to use the TUI and access the hidden elements by scrolling. However, to fully use the new height of the terminal window, a new MCU connection must be made. Therefore, it's necessary to disconnect from the MCU, reset the MCU and reconnect.