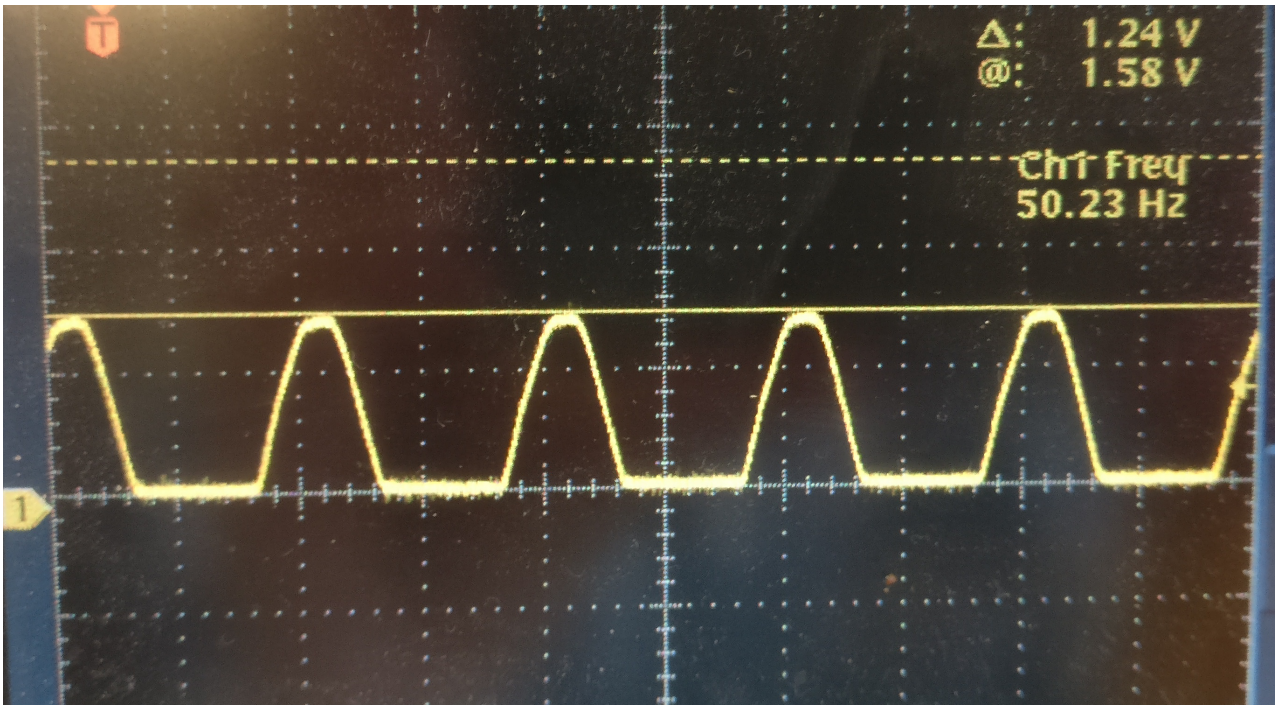
CHALMERS



Konstruktion av ingångsmodul för CAN-buss

Examensarbete inom högskoleingenjörsprogrammet Elektroingenjör

Albert Overland

Institutionen för Elektroteknik

CHALMERS TEKNISKA HÖGSKOLA

Göteborg, Sverige 2018

EXAMENSARBETE

Konstruktion av ingångsmodul för CAN-buss

Albert Overland

Institutionen för Elektroteknik

CHALMERS TEKNISKA HÖGSKOLA

Göteborg, Sverige 2018

Konstruktion av ingångsmodul för CAN-buss

Albert Overland

Institutionen för Elektroteknik

Chalmers tekniska högskola

SE-412 96 Göteborg

Telefon: +46 (0)31-772 1000

Omslag:

Likriktad växelspänning från elnätet som insignal till 12-bitar AD-omvandlare.

Göteborg, Sverige 2018

Konstruktion av ingångsmodul för CAN-buss

Albert Overland

Institutionen för Elektroteknik, Chalmers tekniska högskola

Examensarbete

Sammanfattning

Trigentic AB utvecklar sedan 2001 distribuerade elsystem bestående av I/O-moduler som kommunicerar genom CAN-buss. Det finns ett behov av en kostnadseffektiv ingångsmodul som kan utföra shuntmätning, samtidig mätning av växelspänning och växelström samt beräkna frekvens- och RMS-värden för spänning, ström och effekt. En prototyp tas fram på kopplingsbräde med en mikrokontroller från STMicroelectronics och en 24-bitar AD-omvandlare från Analog Devices. Särskilda krav ställs på upplösningen och samplingsfrekvensen hos shuntmätningen för att kunna detektera små och snabbt växlande DC-nivåer med varaktighet i storleksordningen 10^{-4} s. Prototypen ska kunna använda simultana sampel för att beräkna aktiv effektförbrukning hos komplexa laster samt ha god precision hos frekvensmätningen av signaler upp till 100 Hz. Ingångssignalerna filtreras med Butterworth-filter för att minimalt påverka amplituden i passbandet. Prototypen klarar av detektering av snabbt varierande DC-nivåer och resulterar i shuntmätningar med den maximala medelvärdesavvikelsen 3 %. Prototypen kan detektera skillnader i aktiv effekt på grund av den fasförskjutning som introduceras av komplexa laster. Prototypen kan även beräkna frekvens ur sampel med mycket hög precision på ± 0.01 Hz.

Nyckelord: frekvensmätning, strömmätning, spänningsmätning, mätinstrument, MCU, CAN-buss, Trigentic AB

Abstract

Since 2001, Trigentic AB is developing distributed power systems consisting of I/O-modules communicating over CAN-bus. There is a need for a cost-effective input module capable of measurements on a shunt, simultaneous measurements of AC-voltage and AC-current in addition to calculating frequency- and RMS-values consisting of voltage, current and power from samples. A prototype is developed on a breadboard connecting a STMicroelectronics microcontroller and a 24-bit AD-converter from Analog Devices. There are special requirements regarding the resolution and sampling frequency of the shunt measurement to enable the detection of changing DC-levels with tiny amplitudes and brief durations on the order of 10^{-4} s. The requirements also encompass the ability of utilizing simultaneous samples for calculating active power in loads with a reactive component. Finally, calculations of frequency from samples should be precise up to 100 Hz. The input signals are filtered using Butterworth filters in order to distort the amplitude of the signal as little as possible in the passband. The resulting prototype is able to detect quickly changing DC-levels and is able to provide shunt measurements with a maximum average error of 3 %. The prototype is able to detect variations in the measured active power due to the phase shift introduced by loads with reactive components in addition to calculating frequency from samples with a very high precision of ± 0.01 Hz.

Förord

Examensarbetet genomförs som avslutande del på högskoleingenjörsprogrammet med inriktning elektroteknik vid Chalmers tekniska högskola.

Särskilt tack riktas till min bror Christian Forsberg för hans rekommendation till Trigentic AB, min ena handledare Henrik Niklasson för vägledning och hjälp med hårdvarurelaterade svårigheter, min andra handledare Daniel Johansson för hjälp med mjukvarurelaterade problem, VD för Trigentic AB Magnus Jirhem för att ha gett mig chansen till examensarbete, handledare och examinator vid Chalmers tekniska högskola Thomas Rylander för hjälp med skrivandeprocessen.

Förkortningar

RMS - Root Mean Square (sv. effektivvärde)

ISR - Interrupt Service Routine

NVIC - Nested Vector Interrupt Controller

AD7124 - Namn på 24-bitar analog-till-digitalomvandlare från Analog Devices

IRQ - Interrupt Request

HAL - Hardware Abstraction Layer

DC - Direct Current (sv. likström)

AC - Alternating Current (sv. växelström)

I/O - Input/Output

MCU - Microcontroller Unit (sv. mikrokontroller)

ADC - Analog-to-Digital Converter (sv. analog-till-digital-omvandlare)

SPI - Serial Peripheral Interface

DSP - Digital Signal Processing

GPIO - General Purpose Input/Output

DMA - Direct Memory Access

PGA - Programmable Gain Array

CS - Chip Select

SCK - Serial Clock

MISO - Master In Slave Out

MOSI - Master Out Slave In

EXTI - External Interrupt

word - 32 bitar

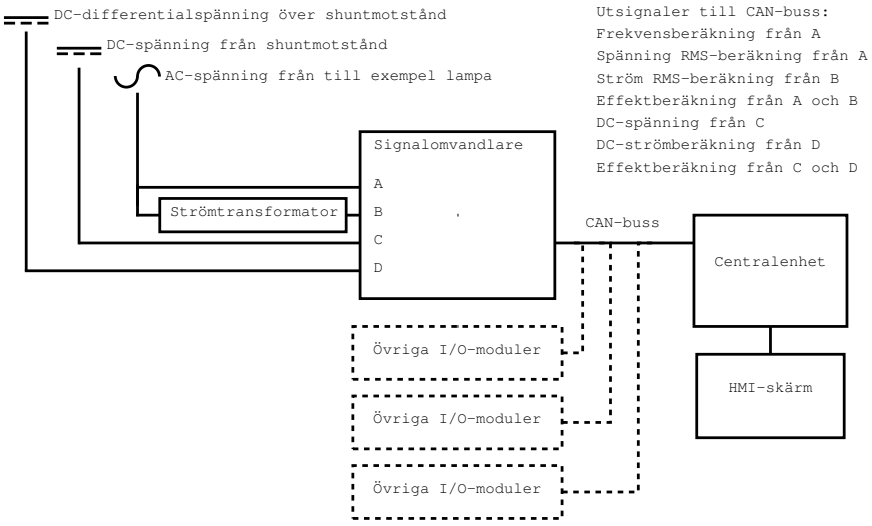
Innehåll

1 Introduktion

1.1 Bakgrund

Trigentic AB utvecklar sedan 2001 distribuerade elsystem bestående av I/O-moduler som kommunicerar genom CAN-buss (Computer Area Network). Elsystemen används i nuläget främst i båtar men kan även användas i andra fordon. I/O-modulernas ingångar kan kopplas till elektriskt styrd utrustning, till exempel en vattenpump eller lampa, och styra dess funktion från en HMI-skärm (Human-Machine Interface) kopplad till centralenheten. Alternativt kan information hämtas, till exempel strömförbrukning, från en enhet och resultatet visas på HMI-skärmen. Kommunikation med centralenheten sker över CAN-buss vilken alla I/O-moduler i systemet är kopplade till.

Det finns ett behov av en kostnadseffektiv signalomvandlare som likt de övriga I/O-modulerna kan anslutas till nätverket och utföra likströmmätning över shuntmotstånd, växelspänningsmätning och växelströmsmätning (indirekt med strömtransformator). Signalomvandlaren behöver använda de uppmätta signalerna för att utföra beräkning av frekvens, kvadratiska medelvärden för växelström och växelspänning samt effekt. Figur 1.1 visar överblick av signalomvandlaren och dess tänkta in- och utsignaler.



Figur 1.1: *Signalomvandlarens tänkta plats i systemet med insignaler och utsignaler till CAN-buss.*

1.2 Syfte

Projektet går ut på att konstruera ett ingångssteg till nätverket som kan utföra analogmätning vid diskreta tidpunkter där både DC och AC-signaler ska kunna mätas. De signaler som ska mätas behöver filtreras med analoga filter för att förberedas för sampling. Förändringar i likströmsnivån s.k. strömspikar som existerar under mycket kort tid (ca. 0.1 ms) ska kunna detekteras. Detta ställer krav på systemets samplingshastighet.

AC signaler i frekvensbandet 0-100 Hz ska kunna samplas. Flera insignaler ska kunna samplas samtidigt och resultatberäkningar är i vissa fall beroende av att två eller flera av de ingående storheterna är uppmätta vid samma tidpunkt, vilket är fallet vid effektberäkning. Samplade signaler ska kunna behandlas digitalt (t.ex. beräkning av kvadratiska medelvärden) samt göras kompatibla med CAN-buss. Komponenter och konstruktionslösning ska väljas för att uppnå så låg strömförbrukning som möjligt.

Flera mätningar ska kunna göras samtidigt, till exempel shuntmätning, effektmätning och frekvensmätning.

1.2.1 Kravspecifikation

Prototypen ska kunna:

- Detektera förändringar i likströmsnivå i storleksordningen 10 mA över shuntmotståndet på 100 $\mu\Omega$ där den minsta tidsskalan är 0.1 ms.
- Beräkna medeleffekt över shuntmotståndet och producera ett linjärt samband mellan strömmen genom shuntmotståndet och beräknad effekt inom valda tidsskalor större än eller lika med 0.1 ms.
- Beräkna aktiv effekt ur simultana sampel av växelström och växelspänning samt kunna detektera minskad aktiv effektförbrukning vid inkoppling av kapacitiva eller induktiva laster.
- Beräkna frekvens ur sampel av växelspänning till en noggrannhet av ± 0.01 Hz jämfört med referensinstrument.
- Samtliga mätningar och beräkningar ska kunna göras parallellt.

1.3 Avgränsningar

Ingångssteget kommer vara en prototyp som visar principen för parallell sampling och signalbehandling vilket senare kan skalas upp och/eller ingå i en fullständig produkt. Antal kanaler är således begränsat till en högupplöst kanal för mätning av likström, en kanal för mätning av likspänning, två kanaler för samtidig sampling av växelspänning och växelström.

Prototypen byggs på kopplingsbräde med utvärderingsversioner av MCU (mikrokontroller) och extern AD-omvandlare (analog-till-digital-omvandlare). Detta försämrar upplösningen hos AD-omvandlare på grund av högre nivåer av brus. Bruset uppstår som ett resultat av sämre galvanisk kontakt och längre ledningslängder jämfört med en motsvarande konstruktion på mönsterkort.

Programmeringsspråket C används för programmering av mikrokontrollern.

1.4 Precisering av frågeställningen

- Vad är beräknad aktiv respektive passiv effektförbrukning för prototypen?
- Hur mycket skiljer sig en samplad utsignal jämfört med motsvarande insignal på ingångssidan?
- Är prototypens prestanda jämn med avseende på frekvensen hos insignalen?
- Fungerar sampling av flera signaler synkront och hur påverkas prestandan?
- Hur mycket avviker resultatberäkning av samplade signaler jämfört med ett referensinstrument?

2 Teknisk bakgrund

Detta kapitel innehåller en beskrivning av hur vissa beräkningar har utförts, översikt av hårdvarukomponenter som ingår samt funktioner ur mjukvarubibliotek som använts vid konstruktion av prototypen.

2.1 Signaltyper

I detta avsnitt finns beskrivningar av de typer av signaler som kan tolkas av prototypen och hur de teoretiskt räknas fram ur mätdata från AD-omvandlare.

2.1.1 Växelspänning

Mätning av växelspänning med en mikrokontroller som endast accepterar unipolära insignaler innebär att den negativa delen av växelspänningen behöver hanteras. Detta kan göras med en summatorkrets som höjer likspänningsnivån med ett värde motsvarande växelspänningens amplitud. Ett annat alternativ är att använda en likriktarkrets vilken konverterar växelspänningen till unipolär spänning genom att endast låta den positiva delen av signalen nå mikrokontrollern. Likriktarkretsen är lämplig för testning av prototypen då det endast behövs en diod för att säkerställa att negativ spänning inte når ingångarna hos mikrokontrollern. RMS-värde (effektivvärde) för en godtycklig kontinuerlig funktion beräknas enligt:

$$\mathrm{RMS}_{\mathrm{funktion}} = \sqrt{\frac{1}{T_2 - T_1} \cdot \int_{T_1}^{T_2} |f(t)|^2 \, dt} \tag{2.1}$$

$f(t)$: Funktion vars RMS-värde ska beräknas

$T_1 - T_2$: Intervallet där RMS-värdet ska beräknas

RMS-värdet för $f(t) = A \cdot sin(\omega t)$ där $\omega = 2\pi f = \frac{2\pi}{T}$ ges av:

$$\text{RMS}_{\text{sinus,helvåg}} = \sqrt{\frac{1}{T} \cdot \int_0^T |A \cdot \sin(\omega t)|^2 \, dt} = \sqrt{\frac{2}{T} \cdot \int_0^{\frac{T}{2}} A^2 \cdot \sin^2(\omega t) dt} =$$

$$\sqrt{\frac{2A^2}{T} \cdot \left[\frac{t}{2} - \frac{\sin(2\omega t)}{4\omega}\right]_0^{\frac{T}{2}}} = \sqrt{\frac{2A^2}{T} \cdot \left(\frac{T}{4} - \frac{\sin\left(2 \cdot \frac{2\pi}{T} \cdot \frac{T}{2}\right)}{4\omega}\right)} = \frac{A}{\sqrt{2}} \tag{2.2}$$

A: Amplitud

ω: vinkelfrekvens [rad/s]

f: frekvens [Hz]

T: periodtid [s]

För att beräkna RMS av en periodisk funktion som ges av $A\sin(\omega t)$ då $0 < t < \frac{T}{2}$ och har värdet noll för resten av perioden $\frac{T}{2} < t < T$ är det lämpligt att endast integrera halva perioden så att:

$$\text{RMS}_{\text{sinus,halvvåg}} = \sqrt{\frac{1}{T} \cdot \int_0^{\frac{T}{2}} |A \cdot \sin(\omega t)|^2 \, dt} = \sqrt{\frac{1}{T} \cdot \int_0^{\frac{T}{2}} A^2 \cdot \sin^2(\omega t) dt} =$$

$$\sqrt{\frac{A^2}{T} \cdot \left[\frac{t}{2} - \frac{\sin(2\omega t)}{4\omega}\right]_0^{\frac{T}{2}}} = \sqrt{\frac{A^2}{T} \cdot \left(\frac{T}{4} - \frac{\sin\left(2 \cdot \frac{2\pi}{T} \cdot \frac{T}{2}\right)}{4\omega}\right)} = \frac{A}{2} \tag{2.3}$$

Således behövs en kompensationsfaktor för att korrekt beräkna RMS-värde givet amplituden A för en likriktad spänning eller ström. Denna kompensationsfaktor är $\frac{1}{\sqrt{2}}$ för helvågslikriktning enligt Ekvation 2.2 och $\frac{1}{2}$ för halvvågslikriktning enligt Ekvation 2.3.

2.1.2 Ström

Två typer av strömmätning ingår i projektet.

- Mätning med shuntmotstånd för likström
- Mätning med strömtransformator för växelström

Shuntmätning innebär att ett motstånd placeras i serie med lasten och spänningsdifferensen över shuntmotståndet mäts. Med kännedom om shuntmotståndets storlek kan strömmen genom det räknas ut med Ohms lag enligt:

$$I = \frac{U}{R} \tag{2.4}$$

I: Ström genom shuntmotståndet (och lasten) [A]

U: Spänning över shuntmotståndet [V]

R: Resistans för shuntmotståndet [Ω]

Shuntmotstånd kan ha flera olika värden och dess resistans bör väljas så att den är försumbar jämfört med lastens resistans, vilket medför att shuntmotståndet inte påverkar strömmen genom lasten på ett betydande sätt. Oftast är dessa värden små och ibland mindre än 100 $\mu\Omega$ vilket också reducerar effektförbrukning och uppvärmning av shuntmotståndet. Detta ställer högre krav på mätningens precision eftersom spänningsfallet över shuntmotståndet avtar linjärt med minskande resistans (under antagandet att temperaturen hos motståndet är konstant) och mycket små shuntmotstånd kan resultera i att den sökta signalens amplitud är i samma storleksordning som bruset i kretsen.

En växelströmstranformator omvandlar ström till spänning med ett linjärt samband där proportionalitetskonstanten är känd och kan därför mätas som växelspänningsmätningen i 2.1.1. Ofta är spänningen som växelströmstransformatorn producerar liten och kan behövas förstärkas med till exempel en operationsförstärkare.

2.1.3 Effektberäkning

Effektberäkning utförs genom att spänningen och strömmens momentanvärden multipliceras enligt:

$$P = U \cdot I \tag{2.5}$$

P: Effekt [W]

U: Spänning över lasten [V]

I: Ström genom lasten [A]

Effektberäkningar från DC-kretsar över shuntmotstånd ställer inga höga krav på simultan

sampling av ström och spänning. Effektberäkningar för AC-kretsar kräver simultan sampling av ström och spänning för att direkt kunna använda momentanvärdena till att bestämma den relativa fasen (tillsammans med amplituderna) för strömmen och spänningen.

2.1.4 Frekvensberäkning

Frekvensberäkning utförs genom att räkna antal gånger samplade värden överstiger en konfigurerbar tröskelspänning under ett visst tidsintervall enligt:

$$f = \frac{N}{\Delta t} \tag{2.6}$$

f: Beräknad frekvens [Hz]
N: Antal gånger tröskelspänningen överstigits
Δt: Längd på tidsintervall [s]

För att förhindra mycket brusiga eller ojämna signaler från att överstiga tröskelspänningen flera gånger under en period kan en flagga sättas tills dess att signalen understigit ett konfigurerbart värde och på så sätt möjliggöra en ställbar hysteres.

2.2 Mikrokontroller - STM32F446RE

NUCLEO-F446RE från STMicroelectronics är ett utvärderingskort med kontakter som kan kopplas direkt till kopplingsbräde. Program skrivs till mikrokontrollerns flashminne genom USB (Universal Serial Bus) som även möjliggör debugging direkt i hårdvaran från en utvecklingsmiljö (μvision IDE från Keil i detta projekt).

Mikrokontrollern använder sig av en 32-bitar ARM (Advanced RISC Machine) processor med stöd för DSP-instruktioner (Digital Signal Processing) från biblioteket CMSIS-DSP. Processorn stödjer även interrupthantering med prioritetssättning som kan ändras under körning. Mikrokontrollern stöder flera periferienheter såsom CAN, SPI (Serial Peripheral Interface), DMA (Direct Memory Access), GPIO (General Purpose Input/Output) och 3 st 12-bitar AD-omvandlare. Processorns maximala klockfrekvens är 180 MHz i “overdrive mode” och 168 MHz utan. Periferienheterna klockas av processorn med konfigurerbar omskalning vilket möjliggör ställbar klockfrekvens hos dessa.

2.3 STM32F4 HAL Library

STMicroelectronics tillhandahåller programmeringsbiblioteket HAL (Hardware Abstraction Layer) Library ämnat att förenkla användandet av mikrokontrollerns periferienheter samt minimera eventuella kompatibilitetsproblem om användaren bestämmer sig för att byta till en annan mikrokontroller i STM32-serien. Biblioteket innehåller funktioner för initiering och styrning av mikrokontrollerns periferienheter. De flesta funktioner som anropas av användaren i HAL Library returnerar statuskoder av datatypen HAL_StatusTypeDef som kan användas för debugging.

2.3.1 Systemklocka och Nested Vectored Interrupt Controller

Här listas funktioner ur HAL CORTEX Library som används i projektet. Funktionerna används för att konfigurera systemklockan och interrupthantering.

___weak HAL_StatusTypeDef HAL_RCC_OscConfig(RCC_OscInitTypeDef RCC_OscInitStruct)

Beskrivning: Konfigurerar vilka oscillatorer som ska användas som systemklocka.

RCC_OscInitStruct: Pekare till struct som innehåller information om klockkälla och hastighet.

HAL_StatusTypeDef HAL_RCC_ClockConfig(RCC_ClkInitTypeDef RCC_ClkInitStruct, uint32_t FLatency)

Beskrivning: Konfigurerar klockkälla för periferienheter.

RCC_ClkInitStruct: Pekare till struct som innehåller information om klockkällor för periferienheter.

FLatency: Anger fördröjning i antal klockcykler när data hämtas från flashminnet, detta eftersom flashminnet är långsammare än processorn.

uint32_t HAL_SYSTICK_Config(uint32_t TicksNumb)

Beskrivning: Konfigurerar räknare för systick (systemklocka) som genererar periodiska interrupts.

TicksNumb: Antal klockcykler mellan interrupts.

void HAL_SYSTICK_CLKSourceConfig(uint32_t CLKSource)

Beskrivning: Konfigurerar vilken klockkälla som ska användas som tidsbas för systick.

CLKSource: Anger klockkälla som ska användas.

void HAL_NVIC_SetPriority(IRQn_Type IRQn, uint32_t PreemptPriority, uint32_t SubPriority)

Beskrivning: Anger prioritet för interrupt. Lågt värde innebär hög priorite. Till exempel betjänas interruptprioritet 0 innan 5.

IRQn: Vilken interrupt det gäller.

PreemtPriority: Anger prioritet hos interrupt.

SubPriority: Anger prioritet mellan interrupts som har samma värde hos PreemtPriority.

HAL_NVIC_SetPriorityGrouping(uint32_t PriorityGroup)

Beskrivning: Anger om subprioriteringar behövs och isåfall hur många nivåer, fler subprioriteringar resulterar i färre preemptprioriteringar.

PriorityGroup: Inställning som styr fördelning mellan antal nivåer hos preemptprioriteringar och subprioriteringar.

void HAL_NVIC_EnableIRQ(IRQn_Type IRQn)

Beskrivning: Tillåter processorn att hantera interrupts.

IRQn: Anger vilken interrupt som ska tillåtas.

void HAL_NVIC_DisableIRQ(IRQn_Type IRQn)

Beskrivning: Maskar interrupt från processorn.

IRQn: Vilken interrupt som ska maskas.

2.3.2 Analog-to-Digital Converter

Här listas funktioner ur HAL ADC Library som används i projektet. Funktionerna används för att konfigurera de inbyggda 12-bitar AD-omvandlarna i mikrokontrollern.

HAL_StatusTypeDef HAL_ADC_Init(ADC_HandleTypeDef* hadc)

Beskrivning: Konfigurerar ADC-parametrar som gäller för samtliga ADC-kanaler.

hadc: Pekare till ADC-instans (struct) som innehåller periferienhetens konfigurationsvariabler och tillstånd.

HAL_StatusTypeDef HAL_ADC_ConfigChannel(ADC_HandleTypeDef* hadc, ADC_ChannelConfTypeDef* sConfig)

Beskrivning: Konfigurerar ADC-parametrar som gäller för en specifik kanal.

hadc: Pekare till ADC-instans (struct) som innehåller periferienhetens konfigurationsvariabler och tillstånd.

sConfig: Konfigurationsinformation om en specifik kanal, till exempel samplingstid.

HAL_StatusTypeDef HAL_ADC_Start(ADC_HandleTypeDef* hadc)

Beskrivning: Aktiverar en ADC periferienhet.

hadc: Pekare till ADC-instans (struct) som innehåller periferienhetens konfigurationsvariabler och tillstånd.

HAL_StatusTypeDef HAL_ADCEx_MultiModeStart_DMA(ADC_HandleTypeDef* hadc, uint32_t* pData, uint32_t Length)

Beskrivning: Startar simultan sampling av samtliga aktiverade AD-omvandlare i DMA-läge.

hadc: Pekare till ADC-instans (struct) som innehåller periferienhetens konfigurationsvariabler och tillstånd.

pData: Pekare till buffer där data från AD-omvandlarna sparas.

Length: Storleken på buffer som pData pekar på.

void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef *hadc)

Beskrivning: Funktionen anropas i en ISR (Interrupt Service Routine) när halva buffern hos ADC-instansen är fylld.

hadc: Pekare till ADC-instans (struct) som innehåller periferienhetens konfigurationsvariabler och tillstånd inklusive storlek och status hos buffer.

void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc)

Beskrivning: Funktionen anropas i en ISR när hela buffern hos ADC-instansen är fylld.

hadc: Pekare till ADC-instans (struct) som innehåller periferienhetens konfigurationsvariabler och tillstånd inklusive storlek och status hos buffer.

2.3.3 Serial Peripheral Interface

Här listas funktioner ur HAL SPI Library som används i projektet. Funktionerna används för att konfigurera SPI-parametrar samt hantera sändning och mottagning av data över SPI.

HAL_StatusTypeDef HAL_SPI_Transmit(SPI_HandleTypeDef *hspi, uint8_t pData, uint16_t Size, uint32_t Timeout)

Beskrivning: Skickar data uppdelad i bytes över SPI.

hspi: Pekare till SPI-instans (struct) som innehåller periferienhetens konfigurationsvariabler och tillstånd.

pData: Pekare till buffer där data som ska skickas finns.

Size: Antal bytes som ska skickas.

Timeout: Maximal tid som funktionen försöker skicka data, baseras på systick.

HAL_StatusTypeDef HAL_SPI_Receive(SPI_HandleTypeDef *hspi, uint8_t *pData, uint16_t Size, uint32_t Timeout)

Beskrivning: Mottager data uppdelad i bytes över SPI. Denna funktion används som debug-verktyg.

hspi: Pekare till SPI-instans (struct) som innehåller periferienhetens konfigurationsvariabler och tillstånd.

pData: Pekare till buffer där data som ska mottagas lagras.

Size: Antal bytes som ska mottagas.

Timeout: Maximal tid som funktionen försöker motta data, baseras på systick.

HAL_StatusTypeDef HAL_SPI_Transmit_IT(SPI_HandleTypeDef *hspi, uint8_t *pData, uint16_t Size)

Beskrivning: Skickar data uppdelad i bytes över SPI och genererar interrupt när all data skickats.

hspi: Pekare till SPI-instans (struct) som innehåller periferienhetens konfigurationsvariabler och tillstånd.

pData: Pekare till buffer där data som ska skickas finns.

Size: Antal bytes som ska skickas.

HAL_StatusTypeDef HAL_SPI_Receive_IT(SPI_HandleTypeDef *hspi, uint8_t *pData, uint16_t Size)

Beskrivning: Mottager data uppdelad i bytes över SPI och genererar interrupt när all data mottagits.

hspi: Pekare till SPI-instans (struct) som innehåller periferienhetens konfigurationsvariabler och tillstånd.

pData: Pekare till buffer där data som ska mottagas lagras.

Size: Antal bytes som ska mottagas.

void HAL_SPI_TxCpltCallback(SPI_HandleTypeDef *hspi)

Beskrivning: Funktionen anropas i en ISR när all data som ska skickas har skickats.

hspi: Pekare till SPI-instans (struct) som innehåller periferienhetens konfigurationsvariabler och tillstånd.

void HAL_SPI_RxCpltCallback(SPI_HandleTypeDef *hspi)

Beskrivning: Funktionen anropas i en ISR när all data som ska mottages har mottagits.

hspi: Pekare till SPI-instans (struct) som innehåller periferienhetens konfigurationsvariabler och tillstånd.

2.3.4 Direct Memory Access

Här listas funktioner ur HAL DMA Library som används i projektet. Funktionerna används för att konfigurera den automatiska minneshanteringen med mikrokontrollerns DMA-kontroller.

HAL_StatusTypeDef HAL_DMA_Init(DMA_HandleTypeDef *hdma)
Beskrivning: Konfigurerar DMA-parametrar.
hdma: Pekare till DMA-instans (struct) som innehåller information om hur DMA ska konfigureras.

2.3.5 General Purpose Input/Output

Här listas funktioner ur HAL GPIO Library som används i projektet. Funktionerna används för att konfigurera funktionen hos de fysiska in- och utgångarna på mikrokontrollern.

void HAL_GPIO_Init(GPIO_TypeDef *GPIOx, GPIO_InitTypeDef GPIO_Init)
Beskrivning: Aktiverar en bestämd GPIO med en specifik konfiguration.
GPIOx: Vilken pin som ska konfigureras.
GPIO_Init: Pekare till struct som innehåller information om hur vald pin ska konfigureras.

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
Beskrivning: Funktionen anropas i en ISR när ett bestämt villkor uppfyllts hos en specifik GPIO pin.
GPIO_Pin: Anger vilken pin som ska kunna generera interrupt.

2.3.6 Timer

Här listas funktioner ur HAL TIM Library som används i projektet. Funktionerna används för att konfigurera periferienheter som kan mäta tid och utföra PWM-funktioner (Pulse Width Modulation).

HAL_StatusTypeDef HAL_TIM_Base_Init(TIM_HandleTypeDef *htim)

Beskrivning: Konfigurerar tidsinställningar för timern.

htim: Pekare till struct som innehåller information om hur timern ska konfigureras. Till exempel kan man konfigurera prescaler, periodtid och om räknaren ska räkna upp eller ner.

HAL_StatusTypeDef HAL_TIM_ConfigClockSource(TIM_HandleTypeDef htim, TIM_ClockConfigTypeDef *sClockSourceConfig)

Beskrivning: Konfigurerar vilken klockkälla som ska användas till räknaren.

htim: Pekare till struct som innehåller information om hur timern ska konfigureras, till exempel prescaler och periodtid.

sClockSourceConfig: Pekare till struct som innehåller information om vilken klockkälla som ska användas till räknaren.

HAL_StatusTypeDef HAL_TIM_PWM_Init(TIM_HandleTypeDef *htim)

Beskrivning: Konfigurerar tidsinställningar för timern som ska användas till PWM.

htim: Pekare till struct som innehåller information om hur timern ska konfigureras, till exempel prescaler och periodtid.

HAL_StatusTypeDef HAL_TIM_PWM_ConfigChannel(TIM_HandleTypeDef *htim, TIM_OC_InitTypeDef* sConfig,
uint32_t Channel)

Beskrivning: Konfigurerar PWM-variabler.

htim: Pekare till konfigurationen för den timer som ska fungera som PWM.

sConfig: Konfigurationsinformation, till exempel pulskvot.

Channel: Anger vilken PWM-kanal som avses, om samma timer används till flera PWM.

HAL_StatusTypeDef HAL_TIM_PWM_Start(TIM_HandleTypeDef *htim, uint32_t Channel)

Beskrivning: Startar generering av PWM-signal.

htim: Pekare till konfigurationen för den timer som ska fungera som PWM.

Channel: Anger vilken PWM-kanal som avses.

2.4 24-bitar AD-omvandlare - AD7124

AD7124 kommunicerar över SPI där data skickas och mottages digitalt i paket om 8 bitar (1 byte). Efter uppstart eller reset av AD7124 väntar enheten på en skrivning till dess kommunikationsregister. Kommunikationsregistret innehåller 2 st flaggbitar på de mest signifikanta bitarna vilka följs av 6 bitar som specificerar vilket register som ska skrivas till eller läsas från, se Tabell 2.1. När bit 7 är satt till 1 ignoreras alla efterföljande bitar, således behöver kommunikation med enheten där ett register ska skrivas till eller läsas från alltid börja med att den första biten är satt till 0. Bit 6 informerar AD7124 om det är en läs- eller skrivoperation som ska utföras.

Tabell 2.1: Kommunikationsregister AD7124.

Bit 7	**Bit 6**	**Bit 5-0**
$\overline{\text{Write Enable}}$	Read/$\overline{\text{Write}}$	Register ID

Information om samtliga registers ID (Addr.) finns i Appendix A.1.

2.4.1 Drivrutin för SPI-kommunikation med AD7124

Följande funktioner implementerades för att sköta kommunikationen mellan mikrokontroller och AD7124 för konfiguration av enheten.

void reset_ad7124_spi(SPI_HandleTypeDef *hspi)
Beskrivning: Utför mjukvarureset av AD7124 genom att skriva 8 bytes innehållande 1:or. Efter skrivning exekveras en fördröjningsrutin baserad på systick för att enheten ska hinna starta om.
hspi: Pekare till SPI-instans (struct) som innehåller periferienhetens konfigurationsvariabler och tillstånd. Används av HAL SPI Library-funktionerna.

void wait_spi_rdy(SPI_HandleTypeDef *hspi)
Beskrivning: Väntar på att AD7124 är redo att skicka eller mottaga data över SPI. Detta görs

genom att kontinuerligt läsa *Error*-registret hos AD7124 tills dess att alla bitar i registret är 0 (3 byte register) eller att en ställbar timeout triggas. Läsning av *Error*-registret utförs genom att skriva 0x46 till kommunikationsregistret och sedan invänta 3 bytes respons.
hspi: Pekare till SPI-instans (struct) som innehåller periferienhetens konfigurationsvariabler och tillstånd. Används av HAL SPI Library-funktionerna.

void wait_conv_rdy(SPI_HandleTypeDef *hspi)
Beskrivning: Väntar tills ett AD-omvandlat värde finns tillgängligt i *Data*-registret hos AD7124 för läsning. Detta görs genom att kontinuerligt läsa av *Status*-registret genom att skriva 0x40 till kommunikationsregistret tills dess att $\overline{\text{RDY}}$-flaggan i registret är satt till 1.
hspi: Pekare till SPI-instans (struct) som innehåller periferienhetens konfigurationsvariabler och tillstånd. Används av HAL SPI Library-funktionerna.

void transmit_spi(SPI_HandleTypeDef *hspi, uint8_t *reg, uint16_t datasize)
Beskrivning: Skriver data till AD7124.
hspi: Pekare till SPI-instans (struct) som innehåller periferienhetens konfigurationsvariabler och tillstånd. Används av HAL SPI Library-funktionerna.
reg: Pekare till variabel innehållande information att skrivning till register ska ske, vilket register som ska skrivas till samt vad som ska skrivas till det.
datasize: Storlek på registret som ska skrivas till.

void read_register_spi(SPI_HandleTypeDef *hspi, uint8_t *reg, uint8_t *rxBuffer, uint16_t datasize)
Beskrivning: Skickar förfrågan innehållande information om vilket register som ska läsas och mottager sedan registrets innehåll.
hspi: Pekare till SPI-instans (struct) som innehåller periferienhetens konfigurationsvariabler och tillstånd. Används av HAL SPI Library-funktionerna.
reg: Pekare till variabel innehållande information att läsning av register ska ske samt vilket register som ska läsas.
rxBuffer: Pekare till buffer där det lästa registervärdet ska lagras.
datasize: Storlek i bytes hos registret som ska läsas.

uint8_t calibrate_pga_gain(SPI_HandleTypeDef *spiHandler)

Beskrivning: Ställer automatiskt in PGA-parameter (Programmable Gain Array) för att se till att AD-omvandlad signal ligger inom omvandlingsområdet. Detta görs genom att ställa in maximal förstärkning och kontinuerligt läsa AD-omvandlade värden från *data*-registret. Om kvantiserat värde är det maximala AD-omvandlaren kan tolka vid ett givet PGA-värde minskar förstärkningen med hälften och läser återigen in värden från *Data*-registret. Denna procedur upprepas tills dess att inkommande signal inte ligger utanför omvandlingsområdet eller når minsta möjliga förstärkning. Returnerar vilket PGA-värde som valts av funktionen. hspi: Pekare till SPI-instans (struct) som innehåller periferienhetens konfigurationsvariabler och tillstånd. Används av HAL SPI Library-funktionerna.

3 Metod

Konstruktion av prototypen består av tre huvudsakliga moment:

1. Val av mikrokontroller.
2. Dimensionering av analoga filter.
3. Digital kommunikation och signalbehandling.

Projektet började med att undersöka vilken hårdvara som kan tänkas behövas. Först valdes en mikrokontroller som har flera AD-omvandlare för att möjliggöra simultan sampling av insignaler, mikrokontrollern behöver även stöd för digital signalbehandling. Val av mikrokontroller begränsades till komponenter från STMicroelectronics. Upplösningen hos AD-omvandlarna i de mikrokontrollers som även hade stöd för digital signalbehandling var begränsad till 12-bitar. Eftersom shuntmätning resulterar i signaler med mycket liten amplitud (mindre än 10 mA) drogs slutsatsen att en AD-omvandlare med högre upplösning behövdes för detta ändamål. En 24-bitars AD-omvandlare som kommunicerar över SPI valdes ut för att komplettera AD-omvandlarna i mikrokontrollern för mätningar som kräver hög precision.

Efter mikrokontroller och AD-omvandlare valts ut genomfördes dimensionering av analoga filter. Dimensioneringsmålet är att filtren ska ha en så nära konstant överföringsfunktion (till amplitud) som möjligt. Eftersom samplade signaler ska kunna användas till effektberäkningar dimensionerades filtren för att påverka amplituden hos inkommande signaler i hela passbandet så lite som möjligt (ungefär 1 %).

Den digitala kommunikationen och signalbehandlingen var det största momentet i projektet och inleddes med att upprätta fungerande kommunikation mellan mikrokontrollern och den externa AD-omvandlaren över SPI med hjälp av STMicroelectronics HAL drivrutin. När SPI-länken fungerade konfigurerades den externa AD-omvandlaren genom skrivning till dess interna register. SPI-länken konfigurerades för detektera att AD-omvandlat värde finns tillgängligt och genererar en IRQ när värdet överförts från extern AD-omvandlare till mikrokontrollern för att meddela att data finns tillgängligt för beräkning eller dylik signalbehandling.

När datainhämtningen fungerade för extern AD-omvandlare över SPI konfigurerades de interna AD-omvandlarna i mikrokontrollern. En AD-omvandlare användes för att läsa in likspänningsnivån tillhörande shuntmätningen för att kunna beräkna effektvärden över shunten.

AD-omvandlarna konfigurerades för simultan sampling av insignaler för att kunna användas för effektberäkning av AC-signaler. De interna AD-omvandlarna ställdes in till att använda sig av mikrokontrollerns DMA-kontroller (Direct Memory Access) för att reducera antal klockcykler som tillägnas minneshantering.

Med en prototyp som kan sampla och utföra enkla beräkningar på inkommande signaler testades noggrannheten hos mätningarna jämfört med ett referensinstrument. Flera olika mätningar utfördes samtidigt för att verifiera att de externa AD-omvandlarna kan användas samtidigt som de interna AD-omvandlarna och att mikrokontrollern hinner med relaterade beräkningar.

4 Genomförande

Detta kapitel beskriver konstruktionen av prototypen med avseende på dimensionering av filter, programmering av mikrokontroller och extern AD-omvandlare. Tillvägagångssätt för verifiering av funktion finns även i detta kapitel.

4.1 Val av mikrokontroller

Vid val av mikrokontroller fanns följande begränsingar:

- Mikrokontroller från STMicroelectronics.
- Minst två AD-omvandlare.
- Stöd för SPI för kommunikation med extern AD-omvandlare.
- Stöd för digital signalbehandling (flyttalsprocessor).
- Stöd för CAN-buss.
- Finns tillgänglig som utvärderingskort för prototypbygge.
- Mikrokontroller med lägst strömförbrukning som uppfyller ovanstående krav.

Mikrokontrollern valdes från STMicroelectronics enligt Trigentic AB:s önskemål. Detta begränsade valet till STM32F4- och STM32F7-serien då STM32F3-serien och tidigare använder sig av ARM Cortex-M3 processor som inte innehåller någon flyttalsprocessor. Flyttalsprocessorn ökar prestandan för digital signalbehandling avsevärt. Mikrokontrollern som valdes var STM32F446RE ur STM32F4-serien då de extra funktionerna som finns på STM32F7-serien inte behövs för projektet samt att den passiva strömförbrukningen är högre för den senare modellen. En slutgiltig produkt skulle kunna använda en annan processor med några ändringar om så skulle vara önskat då källkoden är byggd på STM32 HAL-drivrutinen vilken båda serierna använder sig av. Utvärderingskortet för STM32F446RE heter NUCLEO-F446RE, vilken har lättillgängliga kontakter för prototypbygge på kopplingsbräde och kan programmeras över USB.

4.1.1 Val av 24-bitar AD-omvandlare

Inga mikrokontrollers från STMicroelectronics har en inbyggd 24-bitar AD-omvandlare vilket innebar att en extern AD-omvandlare krävdes för att uppfylla kravet på upplösning. AD-omvandlaren som valdes var AD7124 från Analog Devices med samplingshastighet upp till 19200 Sps (sampel per sekund). Detta bedömdes vara tillräckligt då de absolut minsta tidsskalorna hos inkommande signal är 0.1 ms. AD7124 innehåller "Programmable Gain Array" vilket möjliggör ställbar förstärkning av inkommande signal. Detta ger en mer flexibel lösning då AD-omvandlaren kan tolka ett större amplitudintervall och samtidigt behålla hög upplösning hos signaler med liten amplitud.

AD7124 finns som utvärderingskort, EVAL-AD7124-4, vilket använts för prototypbygget.

4.2 Konstruktion av DC-filter

DC-filter används som ingångssteg till AD7124 för att grovfiltrera inkommande signal genom att minska brus och fungera som en buffer. Detta undviker situationer där AD7124 utsätts för stora spänningssvängningar på differentialingångarna som indirekt mäter strömmen genom shuntmotståndet. Av detta skäl valdes ett aktivt filter för ingången till AD7124 och filtret är av typen Butterworth, vilket innebär maximalt platt karaktäristik i passbandet. Eftersom samplade signaler efter Butterworth-filtret ska kunna användas till beräkning av effekt krävs det att signalens amplitud som når AD7124 helst påverkas så lite som möjligt i passbandet. Det bestämdes att 1 % dämpning vid den högsta frekvensen 10 kHz var acceptabelt då de resistorer och kondensatorer som ingår i filtret har 1 till 2 % precision. För att konstruera ett n-te gradens Butterworth-filter med amplituddämpning $1-G$ vid vinkelfrekvensen ω beräknas brytfrekvensen ω_c som åstadkommer detta med den generella formeln i Ekvation 4.1.

$$G(\omega) = \frac{1}{\sqrt{1 + \left(\frac{\omega}{\omega_c}\right)^{2n}}} \tag{4.1}$$

ω: Ingående frekvens [rad/s]

ω_c: Önskad brytfrekvens [rad/s]

$G(\omega)$: Förstärkning vid vinkelfrekvens $\omega = 2\pi f$

n: Filtergrad (antal poler)

Med $G(\omega) = 0.99$, $n = 2$ och $\omega = 2\pi 10000$ fås önskad brytfrekvens för 1 % dämpning vid 10 kHz i Ekvation 4.2.

$$\omega_c = \frac{2\pi 10000}{\sqrt[4]{\frac{1}{0.99^2} - 1}} \approx 1.6645 \cdot 10^5 \text{ [rad/s]} \tag{4.2}$$

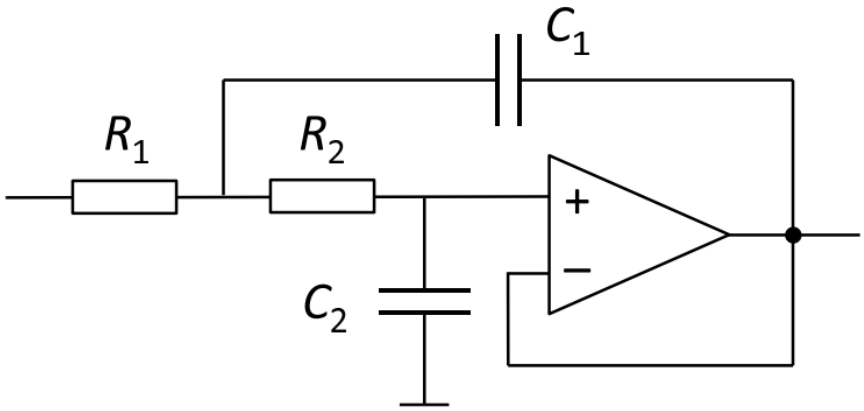
Butterworth-polynom för lågpassfilter av grad 2 vid önskad övre brytfrekvens ω_c ges av Ekvation 4.3.

$$P(a) = 1 + 1.414a + a^2, \; a = \frac{s}{\omega_c} \tag{4.3}$$

Med önskad brytfrekvens från Ekvation 4.2 fås överföringsfunktionen för filtret i Ekvation 4.4.

$$H(s) = \frac{1}{P\left(\frac{s}{\omega_c}\right)} = \frac{1}{1 + 8.4950 \cdot 10^{-4}s + 3.6094 \cdot 10^{-9}s^2} \tag{4.4}$$

Filterarkitekturen som valdes var Sallen-Key på grund av dess flexibilitet där storleken på resistorerna kan väljas, beroende på implementering, för att skydda kretsen från stora spänningsvariationer. Sallen-Key-arkitekturen bidrar även med en buffrad signal då operationsförstärkaren fungerar som en spänningsföljare.



Figur 4.1: *Lågpassfilter av Sallen-Key-arkitektur.*

Överföringsfunktionen för Sallen-Key kan härledas till Ekvation 4.5.

$$H(s) = \frac{1}{1 + (R_1 + R_2)C_2 s + R_1 R_2 C_1 C_2 s^2} \tag{4.5}$$

Jämförelse av Ekvationerna 4.4 och 4.5 resulterar i ett underbestämt ekvationssystem där lösningen som valdes ses i Ekvation 4.6.

$$R_1 = R_2 = 4.2475 \cdot 10^3 \ [\Omega], \ C_1 = 2.0006 \cdot 10^{-9} \ [\text{F}], \ C_2 = 1.0000 \cdot 10^{-9} \ [\text{F}] \tag{4.6}$$

4.3 Konstruktion av AC-filter

Frekvensspannet för AC-signaler är mellan 0 och 100 Hz där 12-bitar AD-omvandlarna som dessa signaler når är mer robusta än den 24-bitar AD-omvandlaren som används för shuntmätning. Av detta skäl användes endast första gradens passiva filter på AC-ingångarna. Likt signaler från DC-filtret ska AC-signaler användas till beräkningar där signalens amplitud inte får påverkas särskilt mycket av filtret i passbandet. För att dimensionera ett första gradens Butterworth-filter används Ekvation 4.1. Med $G(\omega) = 0.99$, $n = 1$ och $\omega = 2\pi 100$ fås önskad brytfrekvens för 1 % dämpning vid 100 Hz enligt Ekvation 4.7.

$$\omega_c = \frac{2\pi 100}{\sqrt[2]{\frac{1}{0.99^2} - 1}} \approx 4.4095 \cdot 10^3 \ [\text{rad/s}] \tag{4.7}$$

Resistansen i första gradens filter med kondensator vald till 100 nF kan beräknas enligt Ekvation 4.8.

$$R = \frac{1}{\omega_c C} = 2.267.8 \cdot 10^3 \ [\Omega] \tag{4.8}$$

4.4 Konfigurering av SPI

Kommunikationen mellan AD7124 och NUCLEO-F446RE sker över SPI där mikrokontrollern behöver anpassas till hur AD7124 kommunicerar. Mikrokontrollern kommer vara *master* i SPI-kommunikationen och är därför ansvarig för sändning av klockpulser för att kunna överföra och ta emot data. Klockhastigheten hos SPI-periferienheten är ställbar i programvaran och sattes till den högsta hastigheten utan att distorsion uppstår, genom empiriskt tillvägagångsätt bestämdes denna hastighet till 2.625 Mbit/s.

SPI-periferienheten konfigurerades till att använda SPI mode 3 vilket krävs av AD7124. Detta innebär att klocksignalen ligger hög (logiskt värde 1) när ingen data skickas eller tas emot, vid dataöverföring klockas data ut på fallande flank och samplas på stigande flank. Fullständig källkod för initiering av SPI-enheten visas i Appendix B.1.

Prototypen innehåller endast en kommunikationslänk över SPI, det vill säga endast en *master* och en *slave*. Detta gör styrning av $\overline{\text{CS}}$ (Chip Select) överflödig då endast ett val av *slave* finns, det vill säga *slave* = AD7124, vilket resulterade i att denna kontakt jordades.

SPI-gränssnittet på AD7124 var kopplat till en 120-pin kontakt ämnad för utvärdering av produkten i kombination med ett moderkort och tillhörande programvara som tillhandahålls av Analog Devices. Kontaktpunkter för SCK (Serial Clock), MOSI (Master Out Slave In), MISO (Master In Slave Out) och $\overline{\text{CS}}$ fanns tillgängliga för att använda utvärderingskortet med en mikrokontroller, labbsladdar löddes fast till dessa kontaktpunkter och anslöts via kopplingsbräde till SPI-kontakterna på mikrokontrollern.

4.5 Konfigurering av AD7124

För att på ett smidigt sätt kunna kommunicera med AD7124 behövs en drivrutin som automatiserar grundläggande funktioner som till exempel reset, läsning och skrivning till register. Analog Devices tillhandahåller en drivrutin för AD7124 vilken kräver att en kopia av alla registertillstånd hos AD7124 lagras i mikrokontrollerns minne. Denna drivrutin testades men visade sig svår att få kompatibel med funktionerna i HAL SPI Library. Dessutom fanns det minst ett fel där ett registervärde i drivrutinen innehöll ett värde vid reset som skiljde sig från databladet. Istället skapades en egen drivrutin med liknande funktioner men som inte behöver lagra registertillstånden. Drivrutinen använder sig av funktioner ur HAL SPI Library, implementerade funktioner och deras beskrivning visas i avsnitt 2.4.1.

När AD7124 startar genomför den en automatisk återställning av alla dess register, men för säkerhets skull kallas reset-funktionen *reset_ad7124_spi* innan någon annan kommunikation med enheten påbörjas. Efter reset skrivs kommandon sekventiellt till AD7124 med funktionen *transmit_spi* för att bestämma kanal (kanal 0), ingångsportar (Ain2+, Ain3-), unipolär/bipolär kodning av AD-omvandlade värden (unipolära), förstärkning (1) och antal sampel per sekund (19200 Sps). Efter konfigurering av AD-kanal används funktionen *calibrate_pga_gain* för att ställa in adekvat förstärkning beroende på insignalens amplitud, det valda PGA-värdet sparas i en variabel *pga_setting* för att kunna beräkna korrekt värde över shunt då detta ändras med olika värden på förstärkningen. Med en referensspänning $V_{\text{ref}} = 2.5$ V och godtyckligt värde på shuntresistorn beräknas skalfaktorn för AD7124 enligt Ekvation 4.9.

$$\alpha = \frac{1}{R_{\text{shunt}}} \cdot \frac{V_{\text{ref}}}{G_{\text{PGA}} \cdot 0xFFFFFF} \tag{4.9}$$

α: Skalfaktor för AD7124-sampel [A]

R_{shunt}: Resistans hos shuntmotstånd [Ω]

V_{ref}: Referensspänning för AD7124 [V]

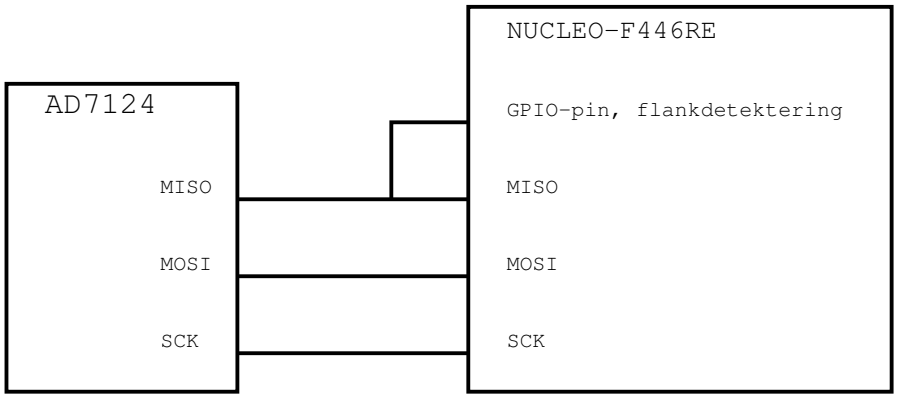
G_{PGA}: Programmerbar förstärkning hos AD7124, 2^0 till och med 2^7

$0xFFFFFF$: Maximalt värde ett 24-bitars tal kan anta i hexadecimal form

Efter inställning av registervärden och bestämning av skalfaktor är konfigureringen av AD7124 färdig och insignaler kan AD-omvandlas kontinuerligt och läggs i *Data*-registret redo för läsning.

4.5.1 Läsning och hantering av data från AD7124

När AD7124 är konfigurerad för kontinuerlig AD-omvandling av värden på dess ingångar omvandlas värden kontinuerligt, dessutom indikeras att ett konverterat värde finns tillgängligt i *data*-registret genom att skicka en kort låg puls (logiskt värde 0) på MISO. Detta eftersom MISO ligger hög då ingen data sänds över SPI. Genom att koppla MISO till en GPIO på mikrokontrollern och generera en IRQ då negativ flank upptäcks kan mikrokontrollern detektera när det är lämpligt att skicka en förfrågan om läsning av *Data*-registret hos AD7124. En schematisk bild över den fysiska kopplingen visas i Figur 4.2.

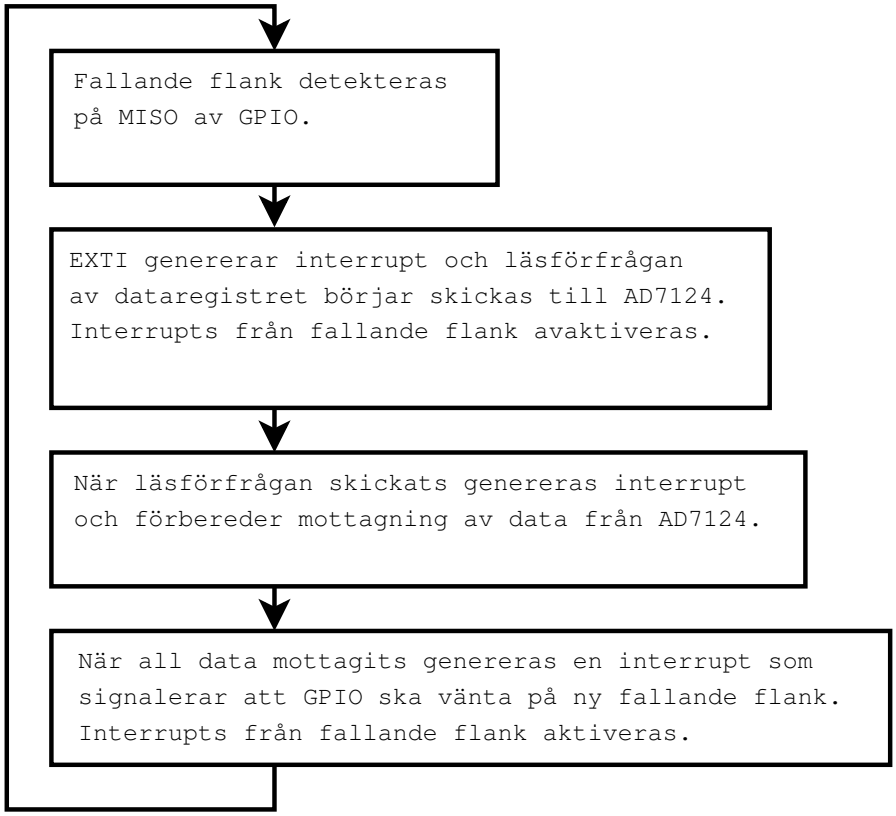


Figur 4.2: *Koppling av en GPIO-kontakt till MISO för detektering av fallande flank.*

Läsning av konverterade värden utförs genom att använda funktionen *receive_spi*. Denna funktionen låser processorn tills dess att en förfrågan av registerläsning skickats och svar mottagits. För att frigöra processorn under SPI-kommunikationen testades en DMA-implementering av SPI-kommunikationen. Tyvärr visade sig detta vara nästintill omöjligt att få kompatibel med hur HAL SPI-funktionerna var skrivna. Istället användes en interruptbaserad lösning för att frigöra klockcykler hos processorn jämfört med en lösning där processorn är blockerad och väntar på att SPI-kommunikationen ska avslutas.

Detektering av att ett konverterat värde finns tillgängligt hos AD7124 utfördes genom att använda mikrokontrollerns EXTI-funktion kopplad till en GPIO. Denna GPIO var i sin tur kopplad till MISO som genererar fallande flank vid färdigkonverterat värde. GPIO-kontakten som används för att detektera fallande flank hos insignal genererar en IRQ. IRQ-funktionens ISR exekverar funktionen *HAL_GPIO_EXTI_Callback*. MISO överför även värdet lagrat i *Data*-registret då en läsförfrågan av *Data*-registret skickas. Med andra ord har MISO två funktioner, en som $\overline{\text{ready}}$-signal och en som Data Out-signal.

Eftersom MISO används till att detektera att färdigkonverterat värde finns tillgängligt i *Data*-registret för läsning och överföring, är det nödvändigt att interruptdetekteringen på GPIO/MISO pausas tills dess att hela värdet i *Data*-registret är överfört från AD7124 till mikrokontrollern och först då tillåta nya interrupt på GPIO. Annars kommer själva dataöverföringen trigga flera IRQ:s (så länge det AD-omvandlade värde inte alltid råkar vara 0) på GPIO och köa dessa till NVIC. ISR-funktionen *HAL_GPIO_EXTI_Callback*, som hör ihop med IRQ triggad på GPIO, maskar således generering av IRQ:s hos EXTI-funktionen, hindrar NVIC från att acceptera IRQ:s från EXTI samt startar en interruptbaserad läsförfrågan av *Data*-registret med funktionen *HAL_SPI_Transmit_IT*. När läsförfrågan har skickats signaleras detta med en IRQ vars ISR kallar funktionen *HAL_SPI_TxCpltCallback*. Denna funktionen kallar i sin tur *HAL_SPI_Receive_IT* för att kunna läsa in responsen från AD7124. När det AD-omvandlade värdet i *Data*-registret har lästs över SPI genereras ytterliggare en interrupt som kallar funktionen *HAL_SPI_RxCpltCallback*. I denna funktionen hanteras den lästa datan och placeras i minnet för kommande beräkningar. funktionen avslutas genom att återigen tillåta NVIC att acceptera IRQ:s från EXTI och tar bort maskningen av interruptgenerering på EXTI för att på nytt kunna detektera fallande flank på $\overline{\text{ready}}$/MISO-linan. Ett flödesschema över kommunikationen mellan mikrokontroller och AD7124 visas i Figur 4.3.



Figur 4.3: *Flödesschema över interruptbaserad kommunikation över SPI mellan AD7124 och mikrokontroller.*

Beräkningar med momentanvärden som ingångsvariabler kan placeras i *HAL_SPI_RxCpltCallback*-funktionen om strikta timingkrav finns, annars bör detta undvikas för att processorn inte ska vara låst i interruptkontext.

AD7124 används till shuntmätningen och momentanströmmen kan enkelt räknas ut från samplat värde från AD7124 enligt Ekvation 4.10 när *skalfaktorAD7124* är bestämd enligt Ekvation 4.9.

$$i = \alpha \cdot S_{\text{AD7124}} \tag{4.10}$$

i: Momentanvärde för ström [A]

α: Skalfaktor för AD7124-sampel [A]

S_{AD7124}: Ett sampel från AD7124

4.6 Konfigurering av 12-bitar AD-omvandlare med DMA

AD-omvandlarna i mikrokontrollern används för att sampla växelspänning och växelström för att kunna beräkna effekt. Samtidig sampling av minst två av de tre tillgängliga AD-omvandlarna i mikrokontrollern behövs för att kunna beräkna effekt för att ta hänsyn till fasberoendet mellan spänning och ström. Mikrokontrollern tillhandahåller ett så kallat *ADC multimode* där AD-omvandlarna kan ställas in till att sampla samtidigt. I *ADC multimode* fungerar två av de tre AD-omvandlarna som slavenheter till den tredje. Detta innebär att om två av AD-omvandlarna samplar simultant används inte den tredje. En lösning för att utnyttja alla tre AD-omvandlarna är att även låta den tredje sampla samtidigt som de andra två.

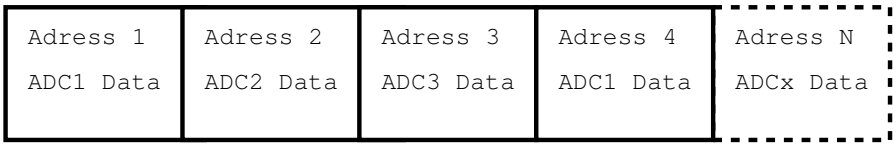
En DMA-implementering av AD-omvandlarna lyckades, vilket frigör klockcykler från processorn då den inte behöver ägna lika mycket tid åt minneshantering. DMA-kontrollern konfigurerades till att lagra insamlad data som datatypen *word* (32-bitar) då funktionerna i biblioteket för digital signalbehandling (CMSIS-DSP) accepterar 32-bitar data som argument. DMA-kontrollern instruerades till att använda en cirkulär databuffer som fyller upp ett, av användaren, specificerat område i minnet med data. När detta område är fullt börjar den skriva över datan som först skrivits till minnet enligt FIFO (First In First Out). Detta är lämpligt då värden kontinuerligt AD-omvandlas och lagras i minnet, men ställer även krav på att relaterade beräkningar hunnit utföras innan data skrivits över. Vald konfiguration ställs in med funktionen *HAL_DMA_Init*, fullständig inställning av DMA-kontrollern kopplad till AD-omvandlarna visas i Appendix B.2 tillsammans med GPIO-konfigurationen för ADC1 (ADC-master).

För att starta simultan sampling av alla tre AD-omvandlare behöver slavenheterna initieras först med funktionen *HAL_ADC_Start* och efter detta initieras masterenheten i DMA-läge med funktionen *HAL_ADCEx_MultiModeStart_DMA*. Programkod för AD-omvandlarnas inställningar, bland annat för att möjliggöra simultan sampling av tre AD-omvandlare, visas i Appendix B.3.

4.6.1 Hantering av data från AD-omvandlare med DMA

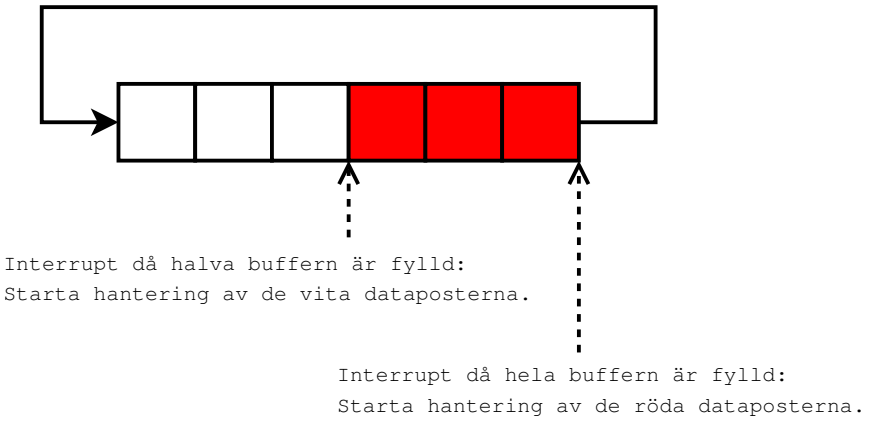
När flera AD-omvandlare samplar samtidigt i DMA-läge lagras samplad data i samma minnesområde med successiva minnesaddresser. Med andra ord separeras minnesposterna med en

offset där ADC1 har offset 0 *word*, ADC2 har offset 1 *word* (32 bitar) och ADC3 har offset 2 *word* (64 bitar) se Figur 4.4.



Figur 4.4: *Minnesstruktur där ADC-data lagras av DMA-kontrollern. Vilken ADC* ($x = 1,\ 2$ *eller* 3) *som ett värde tillhör på en specifik adress* N *kan kan beräknas med:* $x = (N \bmod 3) + 1$.

DMA-kontrollern sköter all lagring av data från de tre AD-omvandlarna med en cirkulär databuffer. En IRQ genereras då det specificerade minnesområdet är fullt och DMA-kontrollern börjar genast skriva över de äldsta lagrade dataposterna samtidigt som mikrokontrollern hanterar IRQ:n i associerad ISR. Detta kan leda till att de första (äldsta) värdena i den cirkulära databufferten skrivits över innan mikrokontrollern haft en chans att utföra någon beräkning på datan eller ens ha kännedom om att datan existerat. För att lösa detta användes en generering av IRQ från DMA-kontrollern när buffern när halvfylld. I interruptens ISR kallas funktionen *void HAL_ADC_ConvHalfCpltCallback*. I funktionen påbörjas beräkningar på första halvan av databuffern medan DMA-kontrollern fyller på andra halvan av dataposterna. När andra halvan är fylld genereras en IRQ vars ISR kallar *HAL_ADC_ConvCpltCallback*. Denna funktion hanterar data i andra halvan av buffern medan DMA-kontrollern börjar skriva över den första halvan av buffern, se Figur 4.5.



Figur 4.5: *Hantering av data med uppdelad DMA-buffer.*

4.7 Interruptprioritering med NVIC

Med flera interruptkällor där vissa innehåller fler tidskritiska operationer än andra finns behovet av att ha en inbördes prioritering av förekommande interrupts. NVIC (Nested Vectored Interrupt Controller) stöder möjligheten att avbryta en IRQ och hantera en annan IRQ för att sedan återuppta den första. Prioritetssättning av IRQ visade sig vara viktigt för SPI-kommunikationen med AD7124 då den innehåller en tidskritisk operation när läs- och skrivkommandon skickas till AD7124. Prioritering av interrupts kan sättas med HAL NVIC-biblioteket där lågt värde indikerar hög prioritet. Först behöver prioriteringsstrukturen bestämmas då det finns två olika prioriteringstyper, *preemptive priority* och *subpriority*. Här avser *preemptive priority* prioritering av interrupts som kan avbryta andra interrupts, *subpriority* avser prioritering av interrupts inom interrupts som har samma *preemptive priority*. För Cortex-M4 i STM32F446RE behöver summan av antal prioritetsnivåer för *preemptive priority* och *subpriority* vara 4, det vill relationen mellan dem är ömsesidigt uteslutande. Med funktionen *HAL_NVIC_SetPriorityGrouping(uint32_t PriorityGroup)* sattes *preemptive priority* till 4 vilket resulterar att *subpriority* är 0.

Tester av prioritering visade att problem uppstod om interrupts associerade med AD-omvandlaren och DMA-kontrollern har högre prioritet än SPI-kommunikationen. Inga problem upptäcktes med periodiska interrupts från timers, inklusive systick som måste ha högst prioritet. Prioritering av interrupts för prototypen kan ses i Tabell 4.1.

Tabell 4.1: Prioritering av interrupts

Prioritering	**Interruptkälla**
0	systick
1	Timers
2	SPI-kommunikation
3	AD-omvandlare och DMA-kontroller

4.8 Konfiguration av systemklocka

Inställning av systemklocka innefattar val av klockkälla, systemklockhastighet, periferiklockhastighet och *systick*-intervall. Räknaren *systick* används för att generera periodiska interrupts och detta kan användas till fördröjningsrutiner. För NUCLEO-F446RE kan klockkällan väljas att vara en extern kristalloscillator eller en inbyggd RC-oscillator där kristalloscillatorn är den mer precisa. Kristalloscillatorn som finns på NUCLEO-F446RE genererar frekvensen 8 MHz vilket är lågt jämfört med mikrokontrollerns klockningsfrekvens som är maximalt 180 MHz under korta perioder (overdrive), och normalt jobbar vid 168 MHz. För att åstadkomma en klockfrekvens på 168 MHz givet en 8 MHz kristalloscillator användes mikrokontrollerns inbyggda fastlåsta slinga (Phase-locked loop, PLL) och tillhörande *prescalers*. När systemklockan är inställd används denna i kombination med *prescalers* för att ställa in klockningshastigheten hos mikrokontrollerns periferienheter. Valda inställningar visas i Tabell 4.2. Observera att de räknare som är kopplade till periferienheterna jobbar vid dubbla frekvensen dvs. 42 MHz.

Tabell 4.2: Klockkonfiguration.

Systemklocka	168 MHz
Räknare för periferienheter	42 MHz
SPI, ADC	21 MHz
GPIO, DMA	21 MHz
systick	1000 ms

4.9 Pin-konfiguration med GPIO

SPI, ADC och PWM använder sig alla av HAL GPIO-biblioteket för koppla en viss funktion till en viss kontakt på mikrokontrollern. För att associera en kontakt med en viss funktion och funktionalitet används *HAL_GPIO_Init*. Mappning av GPIO visas i Tabell 4.3 där ITFF är generering av interrupt på fallande flank som används för SPI-kommunikationen.

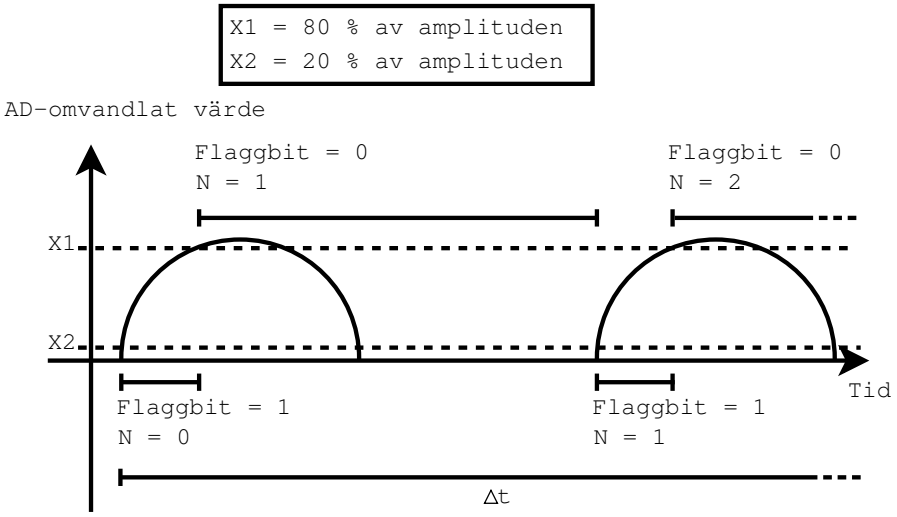
Tabell 4.3: Pin-konfiguration.

ADC1	PA0
ADC2	PA1
ADC3	PC0
PWM	PC3
SCK	PA5
MISO	PA6
MOSI	PA7
ITFF	PB0

4.10 Frekvensmätning med 12-bitar AD-omvandlare

Frekvensmätning utförs genom att kontrollera om varje AD-omvandlat värde ligger över ett ställbart värde, till exempel 80 % av amplituden, och öka en räknarvariabel N om detta är fallet. För att säkerställa att räknaren räknar en positiv flank endast en gång per period används en flaggbit. Flaggbiten nollställs efter att räknaren har räknat upp en gång. För att denna flaggbit ska sättas till 1 igen och på så sätt låta funktionen fortsätta registrera positiv flank krävs det att ett annat ställbart värde har detekterats, till exempel 20 % av amplituden, först då sätts flagbiten till 1 igen och ny period kan detekteras. Genom att mäta tiden Δt det tar för räknarvariabeln N att nå ett visst värde kan frekvensen räknas ut enligt Ekvation 2.6.

Tiden Δt mäts genom att använda en periferienhet och tillhörande HAL TIM-bibliotek som timer. Timern börjar räkna då räknarvariabeln $N = 0$ och läses av först då N nått ett bestämt värde N_{DONE}, efter avläsning av Δt räknas frekvensen ut och efter det nollställs Δt, N och tillhörande räknare för att förbereda ny frekvensberäkning. I Figur 4.6 visas flaggbitens och räknarens värde N under olika delar av den halvvågslikriktade och AD-omvandlade sinussignalens period.



Figur 4.6: *Frekvensberäkning med ställbar hysteres.*

Fullständig kod för frekvensmätning- och frekvensberäkningsfunktionen visas i Appendix B.5.

4.11 PWM-generering för test av frekvens- och shuntmätning

För att testa prototypens funktionalitet användes en av mikrokontrollerns GPIO-kontakter, se Tabell 4.3, till att generera fyrkantsvågor med ställbar periodtid och pulskvot. Frekvensmätningen kräver generering av signaler med den högsta frekvenskomponenten 100 Hz medan shuntmätningen kräver signaler upp till 10 kHz. PWM-funktionen använder sig av en timer inställd i PWM-läge. Detta innebär att timern räknar upp till ett ställbart värde som definierar fyrkantsvågens periodtid *period* och samtidigt skickas hög signal (3.3 V) kontinuerligt ut på den specificerade GPIO-kontakten. Efter en viss tid, beroende på periferienhetens klockhastighet och inställd *prescaler*, kommer räknaren nå ett konfigurerbart värde *pulse* som bestämmer pulsbredden hos den genererade fyrkantsvågen. GPIO-kontakten kommer nu istället skicka ut låg signal (0 V) på GPIO-kontakten. Med dessa tre parametrar *prescaler*, *period* och *pulse* kan en fyrkantsvåg genereras med valfri pulskvot och periodtid (förutsatt att periodtiden inte är i samma storleksordning som timerns klockfrekvens, dvs. 10 kHz $\ll$ 42 MHz). Ett exempel på generering av fyrkantsvåg med f_{sq} och pulskvot 50 % då räknaren räknar uppåt med frekvensen $f_{\text{räknare}}$ visas i Ekvation 4.11. Observera att heltalet 1 adderas automatiskt till *prescaler* i ekvationen [1, s. 573]. Heltalet 1 adderas även till *period* eftersom periodregistret börjar räkna på 0 [1, s. 523]. I detta exempel ger $prescaler = 9$, $period = 559$ och $pulse = 280$ att $f_{sq} = 7500$ Hz.

$$f_{\text{sq}} = \frac{f_{\text{räknare}}}{(prescaler + 1) \cdot (period + 1)} = 7500 \text{ [Hz]} \tag{4.11}$$

f_{sq}: Frekvens hos fyrkantsvågen [Hz]
$f_{\text{räknare}}$: Räknarens frekvens [Hz] där $f_{\text{räknare}} = 42$ MHz

Med vald konfigurering kan PWM-signalen startas med funktionen:
HAL_StatusTypeDef HAL_TIM_PWM_Start, initieringskod för PWM-modulen visas i Appendix B.4.

4.12 Växelström- och växelspänningsmätning med 12-bitar AD-omvandlare

Innan växelström och växelspänning kan mätas behöver de negativa delarna av sinusvågorna tas bort då mikrokontrollern endast hanterar positiv spänning som insignal till AD-omvandlarna. En "over-the-top"-operationsförstärkare från Linear Technology, LT1491A, med inbyggda dioder parallellkopplade mellan signalingångarna och jord användes för att likrikta inkommande ström och spänning [2, s. 9].

Växelspänningen som mättes hade effektivvärdet 230 V som fasspänning och transformerades ner med en spänningstransformator och en spänningsdelare tills den inkommande signalen låg mellan 0 och 3.3 V, vilket svarar mot värden som AD-omvandlaren kan tolka. Spänningstransformatorns sekundärsida gav effektivvärdet 8.485 V vid effektivvärdet 230 V på primärsidan och spänningsdelaren minskade ytterliggare spänningen till 1/12 av inspänningen. Ett skalningsvärde användes för att omvandla den AD-omvandlade spänningen till den faktiska spänningen. Skalningsvärdet tar även hänsyn till att endast den positiva delen av inkommande spänning registreras, det vill säga en faktor $\sqrt{2}$ enligt Ekvation 2.3, så att ett korrekt RMS-värde kan beräknas från sampel. Skalningsvärdet för varje AC-spänningssampel visas i Ekvation 4.12 där $\frac{3.3\ [\mathrm{V}]}{0xFFF}$ är AD-omvandlarens upplösning. Observera att skalningsvärdet appliceras på spänningens momentanvärde och inte effektivvärde.

$$\beta = \sqrt{2} \cdot 230 \cdot \frac{3.3\ [\mathrm{V}]}{0xFFF} = 0.26212\ [\mathrm{V}] \tag{4.12}$$

β: Skalfaktor växelspänning [V]
230: Spänningstransformatorn och spänningsdelaren minskar inspänningen med denna faktor, alltså krävs detta kompensationsvärde
$0xFFF$: Maximalt värde ett 12-bitar tal kan anta i hexadecimal form

Växelström omvandlades till växelspänning med en 16 A strömtransformator som har en sekundärsida på 333 mV vid maximal belastning. En operationsförstärkare, LTC1049, användes för att likrikta och förstärka signalen ≈ 6.96 gånger för att täcka upp större del av spänningsintervallet AD-omvandlaren kan tolka (maximalt 3.3 V). Denna konfiguration innebär

att 16 A hos strömtransformatorn motsvarar ungefär 2.3 V. Skalningsvärdet för växelström kan ses i Ekvation 4.13 där faktorn $\sqrt{2}$ är samma som för växelspänning från Ekvation 2.3.

$$\gamma = \frac{16\ [\text{A}]}{0.333\ [\text{V}]} \cdot \frac{3.3\ [\text{V}]}{0xFFF} \cdot \frac{\sqrt{2}}{6.96} = 0.0078676\ [\text{A}] \tag{4.13}$$

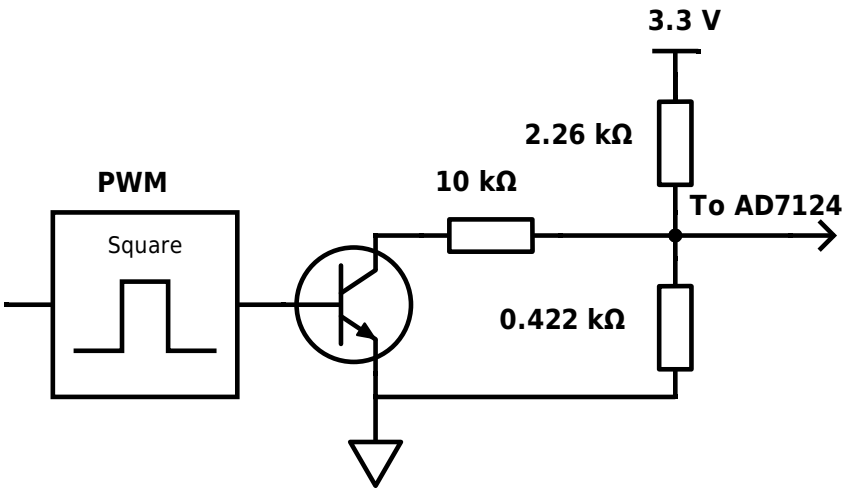
γ: Skalfaktor växelström [A]

$0xFFF$: Maximalt värde ett 12-bitar tal kan anta i hexadecimal form

Eftersom sampel från AD-omvandlarna är omskalade enligt Ekvation 4.12 och 4.13 kan de direkt användas av funktionen *void arm_rms_f32(float32_t *pSrc, uint32_t blockSize, float32_t *pResult)* ur programvarubiblioteket CMSIS-DSP [3].

4.13 Shuntmätning med 24-bitar AD-omvandlare

Differentialmätning över shuntmotstånd av 24-bitar AD-omvandlare testades inte med en faktisk shunt där hundratals ampere flyter igenom. Utan simulerades istället, av praktiska skäl, genom att använda en PWM-signal, en transistor som strömbrytare och ett resistornätverk. Detta för att kunna variera PWM-signalens pulsbredd och frekvens och på så sätt kunna se om den beräknade effekten över shuntmotståndet ökade linjärt med ökande pulsbredd och vice versa vid olika frekvenser. Testkretsen kan ses i Figur 4.7 och resulterande ekvationer när transistorn leder kan ses i Ekvation 4.14 och då den inte leder i Ekvation 4.15.



Figur 4.7: *Krets för test av strömmätning över shuntmotstånd med 24-bitar AD-omvandlare.*

$$\text{Transistor på, insignal AD7124: } 3.3\ [\text{V}] \cdot \frac{422 \parallel 10000\ [\Omega]}{422 \parallel 10000\ [\Omega] + 2260\ [\Omega]} \approx 0.501409\ [\text{V}] \quad (4.14)$$

$$\text{Transistor av, insignal AD7124: } 3.3\ [\text{V}] \cdot \frac{422\ [\Omega]}{422\ [\Omega] + 2260\ [\Omega]} \approx 0.519239\ [\text{V}] \quad (4.15)$$

Enligt Ekvation 4.14 och 4.15 skickas en differentialsignal på $0.519239\ [\text{V}] - 0.501409\ [\text{V}] = 0.01783\ [\text{V}]$ till AD7124, differentialsignalen är tänkt att simulera ett konstant spänningsfall över shuntmotståndet på 0.501409 V och ett dynamiskt spänningsfall på $0.501409 + 0.01783$ V.

Signalen som skickas vidare till AD7124 behöver omvandlas med ett skalningsvärde för att kunna tolkas som ström, shuntmotståndet sattes till 100 $\mu\Omega$ och $pga\_setting = 128$ då denna PGA-inställning kan tolka unipolära värden mellan 0 och 19.53 mV [4, s. 88] vilket gör att skalningsvärdet kan beräknas med av Ekvation 4.9, resultatet ses i Ekvation 4.16.

$$\xi = \frac{1}{0.0001\ [\Omega]} \cdot \frac{2.5\ [\text{V}]}{128 \cdot 0xFFFFFF} = 0.000011642\ [\text{A}] \quad (4.16)$$

ξ: Skalfaktor AD7124 [A]

$0xFFFFFF$: Maximalt värde ett 24-bitar tal kan anta i hexadecimal form

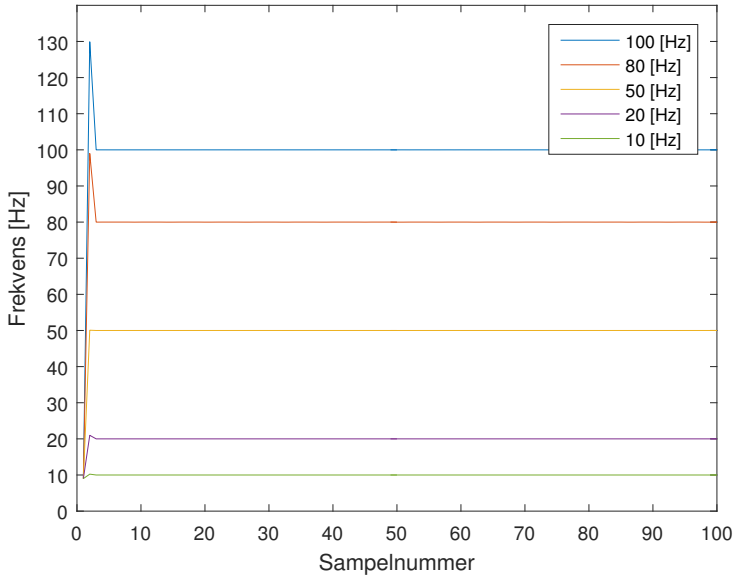
För att beräka medeleffekt multipliceras varje momentanvärde för den beräknade strömmen med likspänningskomponenten 3.3 V (uppmätt med 12-bitar AD-omvandlare) och resultatet sparas som ett 32-bitars flyttal i minnet. När ett bestämt antal värden sparas beräknas medelvärdet med funktionen *void arm_mean_f32(float32_t *pSrc, uint32_t blockSize, float32_t *pResult)* ur programvarubiblioteket CMSIS-DSP [3] och returnerar den beräknade medeleffekten.

5 Utvärdering av prototyp och insamling av data

Insamling av data utfördes i KEIL µVision® IDE över USB och datan sparades i textfiler. Textfilerna lästes in i MATLAB för att kunna visualisera insamlad data.

5.1 Test av frekvensmätning

Frekvensmätning testades i frekvensbandet 0-100 Hz och jämfördes med ett referensinstrument, FLUKE 87V Industrial Multimeter, som kan mäta frekvens med upplösningen 0.01 Hz [5, s. 3]. För varje frekvensvärde som testades samlades 100 datapunkter in under ungefär 100 s. Det första frekvensvärdet som beräknas efter uppstart eller reset av prototypen är i regel felaktigt och bör ignoreras då den består av ett gissat värde. Frekvensmätningen har även ett insvängningsförlopp vid byte av frekvens på ingången innan ett stabilt värde kan läsas av, se Figur 5.1. Insvängningsförloppet brukar vara 2 till 3 sampel och gäller alla frekvenser utom vid 50 Hz. Detta beror på att cykeltiden Δt i Ekvation 4.10 behöver ett initial värde vid uppstart av systemet, detta värde motsvarar en uppmätt frekvens på 50 Hz. Översvängningen är större vid högre frekvenser, detta beror på att det programmerbara värdet för antal räknade perioder N_{DONE}, se Ekvation 4.10, innan frekvensberäkningen utförs är satt till ett lågt initialt värde. Detta resulterar i att en insignal på 100 Hz och $N_{DONE} = 10$ endast kommer räkna 10 hela perioder innan frekvensen beräknas. Programmet kommer detektera att N_{DONE} är satt alldeles för lågt och justerar värdet för kommande mätningar vilket leder till högre precision.



Figur 5.1: *Mätningar av signaler med olika frekvenser med 12-bitar AD-omvandlare.*

Fler mätningar med frekvenser mellan 0 och 100 Hz finns att tillgå i Appendix C.

5.2 Test av shuntmätning

Shuntmätningen testades genom att låta PWM-modulen generera olika frekvenser och pulsbredder, detta för att simulera en variabel ström genom en 100 $\mu\Omega$ shuntresistor. Resultaten från Ekvation 4.14 och 4.15 kan användas för att beräkna hur stor ström som flyter genom shuntmotståndet och utifrån det beräkna förbrukad aktiv effekt. Spänningsfallet över shuntmotståndet som resultat av PWM-modulen är 0.01783 V vid 100 % pulsbredd. Strömmen genom motståndet ges då av Ekvation 5.1.

$$I_{shunt} = \frac{U_{shunt}}{R_{shunt}} = \frac{0.01783\ [\text{V}]}{0.0001\ [\Omega]} = 178.3\ [\text{A}] \tag{5.1}$$

Strömmen genom motståndet då PWM-modulen skickar signal med 0 % pulsbredd bör i teorin vara 0 A. I kretsen uppmättes spänningsfallet vid 100 % pulsbredd till 0.0190 V och vid 0 % pulsbredd till 0.0017 V. Detta innebär att en ström i praktiken alltid flyter genom shuntmotståndet och ger upphov till ett tillägg på $\frac{0.0017\ [\text{V}]}{0.0001\ [\Omega]} = 17\ [\text{A}]$ i Ekvation 5.1. Resultatet blir vid 100 % pulsbredd förväntas det uppmätta värdet vara $I_{shunt} = 178.3\ [\text{A}] + 17\ [\text{A}] = 195.3\ [\text{A}]$. Denna avvikelse från det uppmätta värdet (190 A) kan bero på att, i det ideala fallet, linjära

sambandet mellan spänningen över shuntmotståndet och genomflytande ström i någon mån påverkas av filtret. Detta eftersom de "skarpa" hörnen på fyrkantsvågen från PWM-modulen kommer filtreras bort vid alla pulsbredder < 100 %, vid 100 % pulsbredd matas endast likspänning till AD-omvandlarens ingången och inga frekvenskomponenter dämpas av filtret. En generell formel för det förväntade värdet, där kompensation finns för den konstanta strömmen på 17 A som alltid flyter genom motståndet (under antagandet att linjärt samband råder), och samtidigt tar hänsyn till inställd pulsbredd visas i Ekvation 5.2.

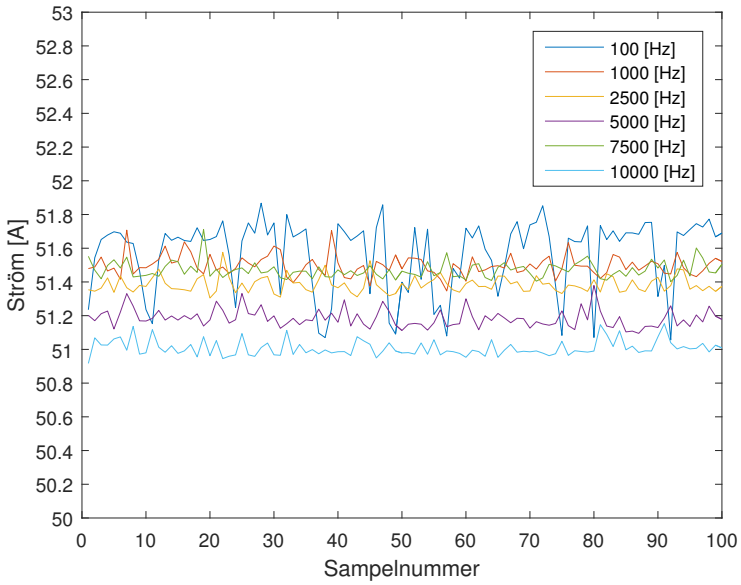
$$I_{shunt} = 178.3\ [\text{A}] \cdot \%D + 17\ [\text{A}] \tag{5.2}$$

$\%D$: Pulsbredd som procentuell andel av periodtiden

Vid 20 % pulsbredd förväntas strömmen genom shuntmotståndet vara enligt Ekvation 5.3.

$$I_{shunt,20\%} = 178.3\ [\text{A}] \cdot \frac{1}{5} + 17\ [\text{A}] = 52.66\ [\text{A}] \tag{5.3}$$

Testdata från shuntmätningen vid några olika frekvenser och 20 % pulsbredd visas i Figur 5.2. Här förväntas mätningarna ligga nära varandra eftersom de har samma pulsbredd.

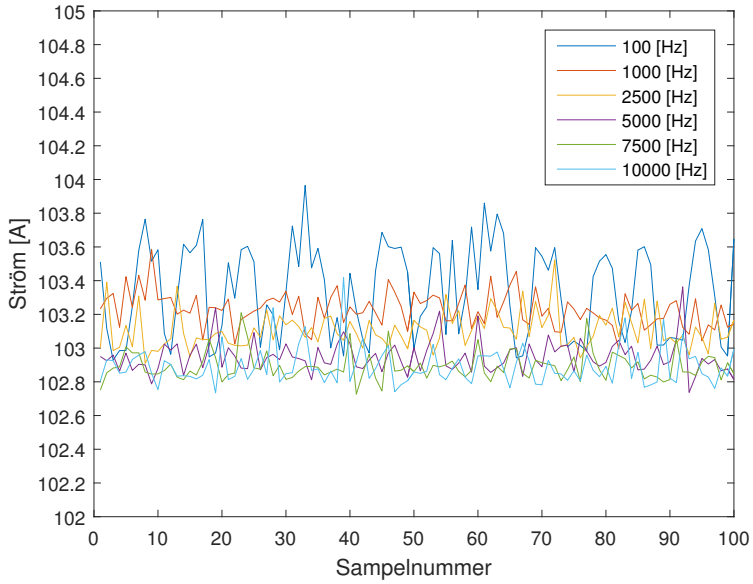


Figur 5.2: *Strömmätning av signal med 20 % pulsbredd och ett antal olika frekvenser över ett shuntmotstånd. Mätning sker differentiellt med 24-bitar AD-omvandlare.*

Vid 50 % pulsbredd förväntas strömmen genom shuntmotståndet vara enligt Ekvation 5.4.

$$I_{shunt,50\%} = 178.3\ [\text{A}] \cdot \frac{1}{2} + 17\ [\text{A}] = 106.15\ [\text{A}] \tag{5.4}$$

Testdata från shuntmätningen vid några olika frekvenser och 50 % pulsbredd visas i Figur 5.3.

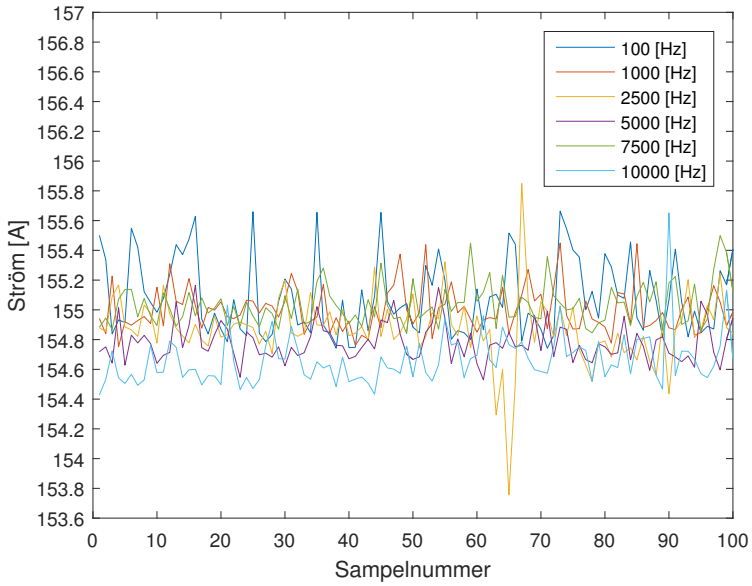


Figur 5.3: *Strömmätning av signal med 50 % pulsbredd och ett antal olika frekvenser över ett shuntmotstånd. Mätning sker differentiellt med 24-bitar AD-omvandlare.*

Vid 80 % pulsbredd förväntas strömmen genom shuntmotståndet vara enligt Ekvation 5.5.

$$I_{shunt,80\%} = 178.3\ [\text{A}] \cdot \frac{4}{5} + 17\ [\text{A}] = 159.64\ [\text{A}] \tag{5.5}$$

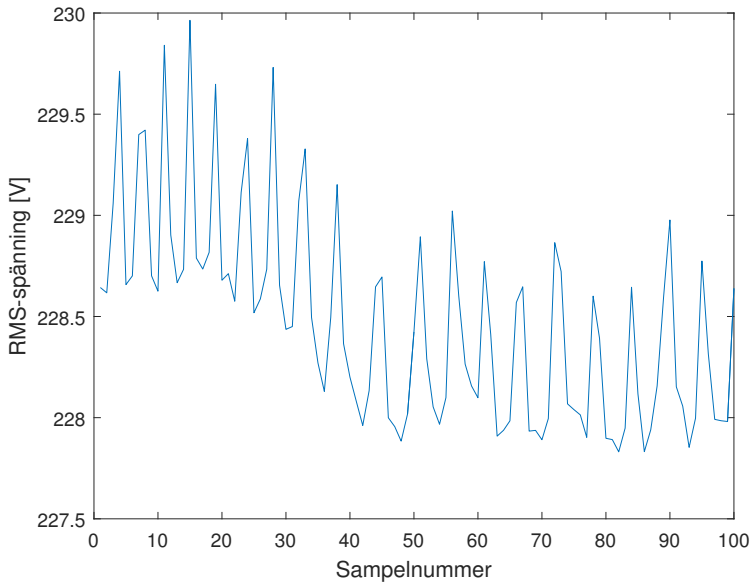
Testdata från shuntmätningen vid några olika frekvenser och 80 % pulsbredd visas i Figur 5.4.



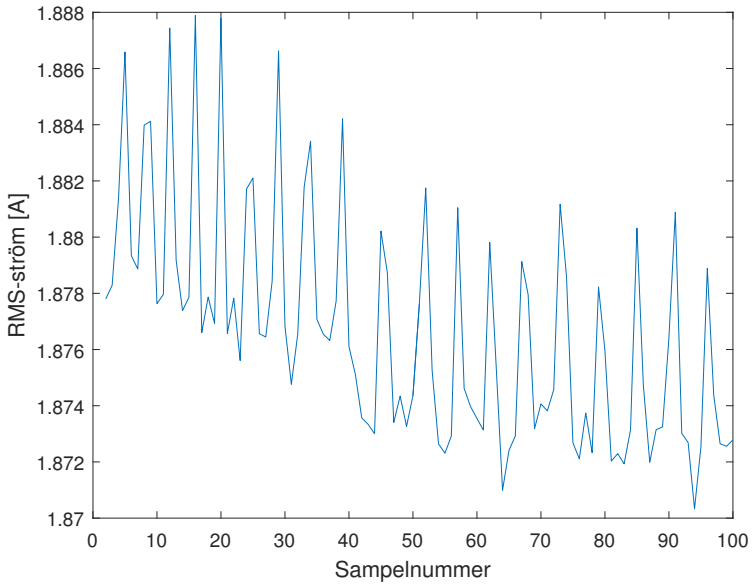
Figur 5.4: *Strömmätning av signal med 80 % pulsbredd och ett antal olika frekvenser över ett shuntmotstånd. Mätning sker differentiellt med 24-bitar AD-omvandlare. En strömspik av okänt ursprung ses mellan sampelnummer 60 och 70.*

5.3 Test av AC-mätning

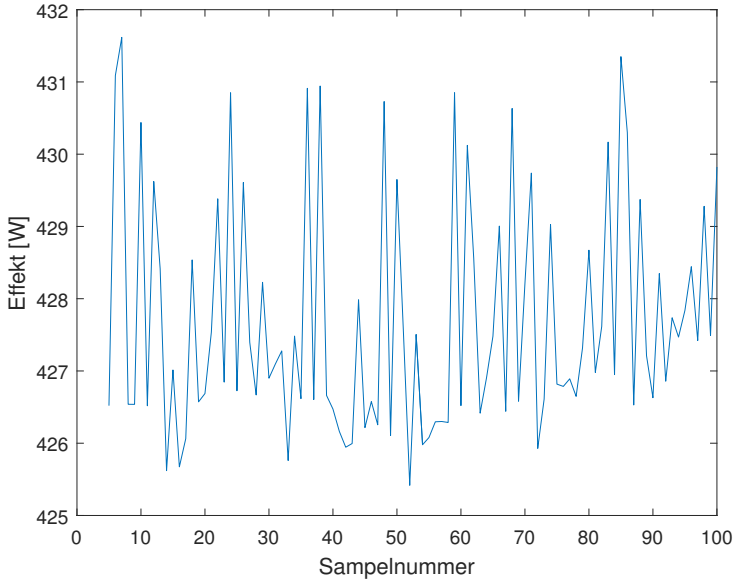
Simultan växelström- och spänningsmätning testades genom att mäta ström, spänning och effekt på en fas med ett referensinstrument [6] för att sedan jämföra resultaten med de beräknade värdena från AD-sampel. Både en rent resistiv last och komplexa laster testades. De komplexa lasterna användes för att verifiera korrekt beräkning av aktiv effekt då ström och spänning inte ligger i fas. En lampa kopplades in som en resistiv last på fasen och de uppmätta och beräknade RMS-värdena för spänning och ström finns att tillgå i Figur 5.5 och Figur 5.6. Beräknad effekt visas i Figur 5.7, uppmätta värden med referensinstrumentet visas i Tabell 5.1.



Figur 5.5: *Beräknad RMS ur spänningsmätning med 12-bitar AD-omvandlare med resistiv last inkopplad.*



Figur 5.6: *Beräknad RMS ur strömmätning med 12-bitar AD-omvandlare med resistiv last inkopplad.*



Figur 5.7: *Beräknad effekt ur ström- och spänningsampel från 12-bitar AD-omvandlare med resistiv last inkopplad.*

Tabell 5.1: Referensinstrumentets mätvärden under samma tid som ström, spänning och effekt uppmättes med 12-bitar AD-omvandlare, resistiv last inkopplad. Listad ström och spänning är effektivvärden.

Uppmätt värde (intervall)	**Enhet**
226.6-228.9	V
1.76-1.77	A
400	W

En kapacitiv last (lysrör) kopplades in på fasen och uppmätta effektivvärden för spänning och ström visas i Figur 5.8 och 5.9. Resulterande effekt visas i Figur 5.10. Referensinstrumentets värden vid kapacitiv last visas i Tabell 5.2. Att lasten nu bidrar till en fasförskjutning mellan ström och spänning och därmed en reducerad aktiv effekt kan ses genom att de uppmätta RMS-värdena för ström och spänning multiplicerade (skenbar effekt) inte resulterar i den uppmätta aktiva effekten 180 W enligt Ekvation 5.6. Effektfaktorn kan beräknas enligt Ekvation 5.7.

$$S = \frac{228.5 + 229.2}{2}[\text{V}] \cdot \frac{0.92 + 0.94}{2}[\text{A}] \approx 213\ [\text{VA}] > P = 180\ [\text{W}] \tag{5.6}$$

P: Aktiv effekt [W]

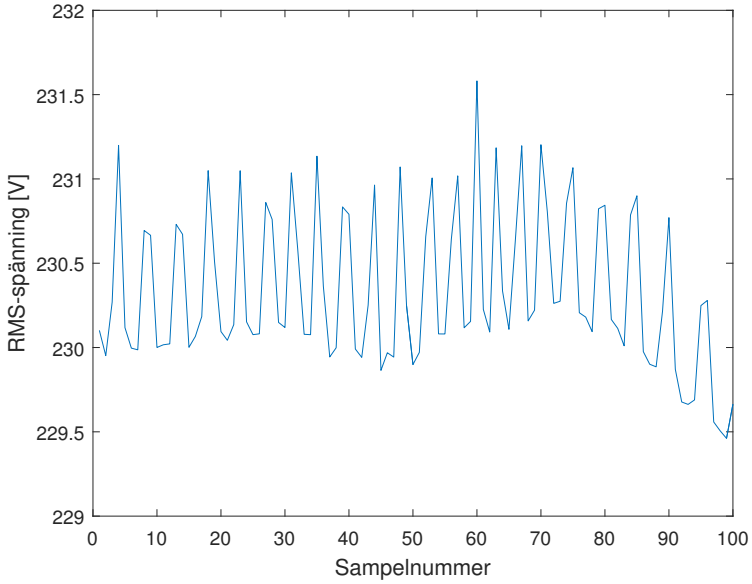
S: Skenbar effekt [VA]

$$\cos\varphi_{\text{kap}} = \frac{P}{S} = \frac{180\ [\text{W}]}{213\ [\text{VA}]} \approx 0.845 \tag{5.7}$$

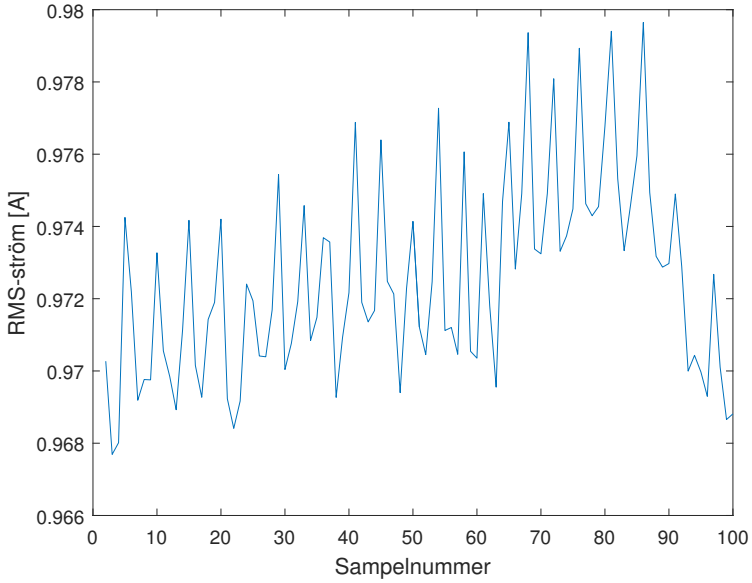
$\cos\varphi_{\text{kap}}$: Effektfaktor med kapacitiv last

P: Aktiv effekt [W]

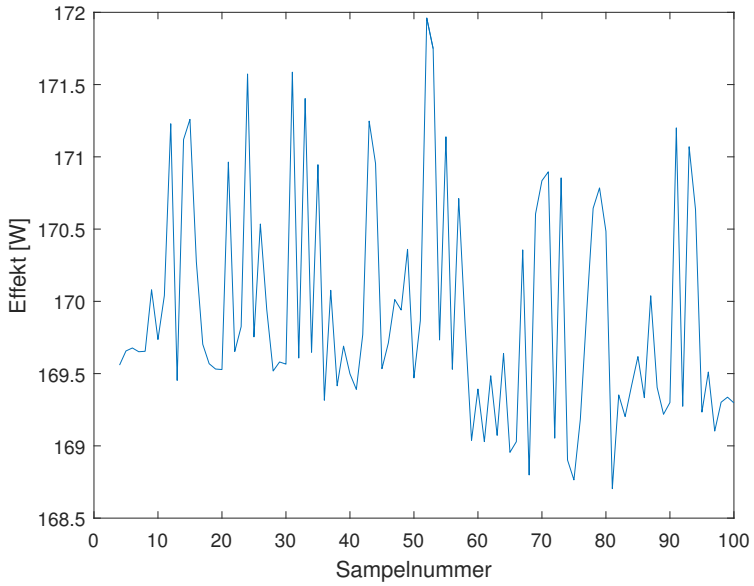
S: Skenbar effekt [VA]



Figur 5.8: *Beräknad RMS-spänning ur spänningsampel från 12-bitar AD-omvandlare med kapacitiv last inkopplad.*



Figur 5.9: *Beräknad RMS-ström ur strömsampel från 12-bitar AD-omvandlare med kapacitiv last inkopplad.*



Figur 5.10: *Beräknad effekt ur ström- och spänningsampel från 12-bitar AD-omvandlare med kapacitiv last inkopplad.*

Tabell 5.2: Referensinstrumentets mätvärden under samma tid som ström, spänning och effekt uppmättes med 12-bitar AD-omvandlare, kapacitiv last inkopplad. Listad ström och spänning är effektivvärden.

Uppmätt värde (intervall)	**Enhet**
228.5-229.2	V
0.92-0.94	A
180	W

Med samma kapacitiva last inkopplad anslöts en induktiv last i form av en fläkt. Detta bör öka den uppmätta aktiva effekten då det induktiva bidraget bör i någon mån motverka den kapacitiva fasförskjutningen (förutsatt att det induktiva bidraget inte är mycket större än bidraget från den kapacitiva lasten). Den aktiva effekten kommer också att öka på grund av att fläkten drar ström. Därför är en direkt jämförelse av uppmätt effekt missvisande och ett bättre mått är jämförelse av effektfaktorerna för båda laster. Effektivvärden för uppmätt ström och spänning med kapacitiv och induktiv last inkopplad visas i Figur 5.11 och Figur 5.12. Beräknad

effekt visas i Figur 5.13. Värden uppmätta med referensinstrumentet visas i Tabell 5.3. Skenbar effekt beräknas på samma sätt som i Ekvation 5.6 och jämförs med uppmätt aktiv effekt i Ekvation 5.8, resulterande effektfaktor beräknas i Ekvation 5.9.

$$S = \frac{228.9 + 229.4}{2}[\text{V}] \cdot \frac{1.13 + 1.14}{2}[\text{A}] \approx 260\ [\text{VA}] > P = 230\ [\text{W}] \tag{5.8}$$

P: Aktiv effekt [W]

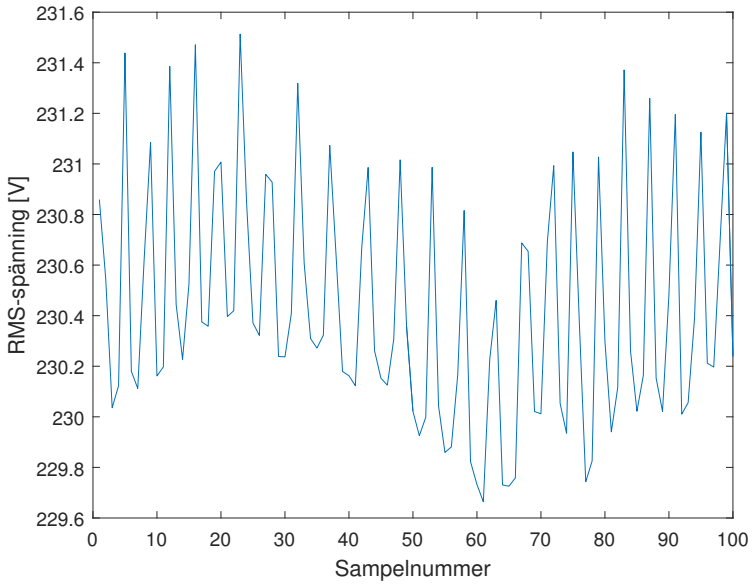
S: Skenbar effekt [VA]

$$\cos\varphi_{\text{kap,ind}} = \frac{P}{S} = \frac{230\ [\text{W}]}{260\ [\text{VA}]} \approx 0.885 \tag{5.9}$$

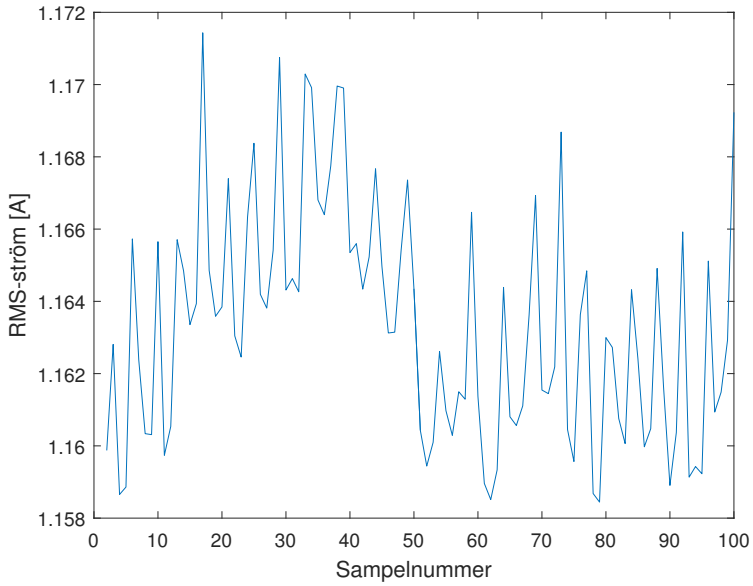
$\cos\varphi_{\text{ind}}$: Effektfaktor med kapacitiv och induktiv last

P: Aktiv effekt [W]

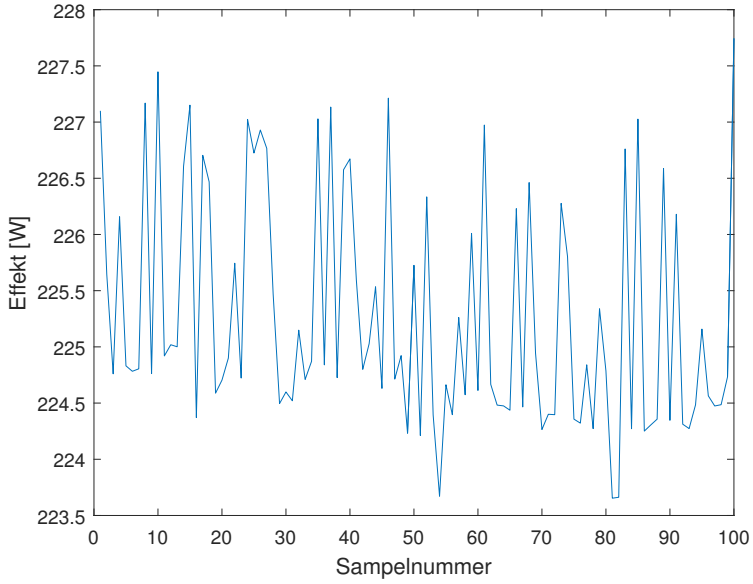
S: Skenbar effekt [VA]



Figur 5.11: *Beräknad RMS-spänning ur spänningsampel från 12-bitar AD-omvandlare med kapacitiv och induktiv last inkopplad.*



Figur 5.12: *Beräknad RMS-ström från ström- och spänningsampel från 12-bitar AD-omvandlare med kapacitiv och induktiv last inkopplad.*



Figur 5.13: *Beräknad effekt från ström- och spänningsampel från 12-bitar AD-omvandlare med kapacitiv och induktiv last inkopplad.*

Tabell 5.3: Referensinstrumentets mätvärden under samma tid som ström, spänning och effekt uppmättes med 12-bitar AD-omvandlare, kapacitiv och induktiv last inkopplad. Listad ström och spänning är effektivvärden.

Uppmätt värde (intervall)	**Enhet**
228.9-229.4	V
1.13-1.14	A
230	W

6 Diskussion

Ett problem som behöver lösas för god funktionalitet har och göra med SPI-kommunikationen. De drivrutiner från HAL-biblioteket som användes för överföringen skulle behöva ses över då den interruptbaserade SPI-överföringen inte verkar fungerar som den ska utan innehåller wait-loopar vilka blockerar processorn tills dess att överföringen är klar. Eftersom SPI-överföring sker ofta vid hög samplingshastighet hos AD7124, resulterar detta i att SPI-kommunikationen tar upp cirka 20-25 % av processorns tid.

Gällande aktiv och passiv effektförbrukning utfördes ingen realistisk utvärdering då det finns bättre komponentval gällande operationsförstärkare etc. om man inte begränsas av vad som är kompatibelt med ett kopplingsbräde, dessa bättre alternativ kommer sannolikt att väljas vid en faktisk implementering. En realistisk utvärdering kräver kännedom om hur stor andel av tiden som prototypen kommer vara aktiv (samla in och utföra beräkningar på data), samt fördelningen av aktivitet mellan olika periferienheter hos mikrokontrollern. Av detta skäl begränsades valet av strömsnåla komponenter till AD-omvandlaren och mikrokontrollern då dessa kräver större ansträngning att byta ut än, till exempel, en operationsförstärkare. Utvärdering av komponenter med avseende på energisnålhet utfördes därför endast genom att läsa datablad och välja komponenter som har stöd för strömsnåla “sleep”-tillstånd. Till exempel drar AD7124 vid den högsta samplingfrekvensen (19200 Sps) och PGA aktiverad 930 μA och når strömnivåer runt 22 μA i “stand-by”-läge. Detta säger dock inte särskilt mycket utan kunskap om hur stor andel av tiden som komponenten används i respektive läge. Data med information om användningsmönster kommer behövas för att närmare utreda hur energieffektiv en slutprodukt kan vara.

Prototypen klarar av att detektera förändringar i likströmsnivån hos insignaler med frekvenser upp till 10 kHz med amplitudskalor motsvarande 10-tals mA. Detektering av förändringar innebär inte nödvändigtvis att precisionen hos den kvantifierade datan är särskilt god vid de högsta frekvenserna då samplingshastigheten på 19200 Sps inte uppnår Nyquist-Shannons samplingsteorem för frekvenser över 9.6 kHz. Detta gäller i synnerhet inte då testningen utfördes med fyrkantsvågor som insignaler vilka, idealt sett, kräver en oändligt hög samplingsfrekvens för att undvika vikning. Eftersom fyrkantsvågen passerar ett filter innan mätning försvann de högsta frekvenserna och därmed även fyrkantsvågkaraktäristiken i någon mån för

de alla högsta frekvenserna vilket sannolikt bidrar till ett mer olinjärt samband mellan spänningsfallet över shunt-motståndet och strömmen genom det vid höga frekvenser. En avvikelse från förväntade värden och prototypens mätvärden fanns i alla mätningar som utfördes och nådde som högst 3 % jämfört med medelvärdet av uppmätt ström (referensinstrument). Ytterligare tester på ett riktigt shuntmotstånd och med en konstruktion på ett prototypkort istället för kopplingsbräde rekommenderas för att kunna fastställa orsaken till denna avvikelse.

Beräkning av aktiv effekt och tillhörande RMS-värden för spänning och ström från 12-bitar AD-omvandlare skiljde ganska mycket från referensinstrumentet i vissa fall. Detta verkade mest ha att göra med avvikelser i det beräknade RMS-värdet hos strömmen. Tjockleken på fasledaren som testades uppgick inte till 9 mm i diameter som rekommenderas i databladet för strömtransformatorn [7]. Transformatorn som användes är dimensionerad för 16 A medan fasen endast lastades med strömmar som nådde cirka 1.9 A som högst. Dessa två faktorer gör det svårt att avgöra huruvida det är något fel i prototypens mätning och/eller beräkningar som ger upphov till dessa avvikelser. Även om de absoluta värdena hade en avvikelse kunde en skillnad i beräknad aktiv effekt detekteras då olika komplexa laster anslöts, vilket förhoppningsvis innebär att den simultana samplingen av ström och spänning kan detektera en skillnad i fasförskjutning. Många fler tester med noggrannare mätinstrument behöver utföras för att hitta orsaken till avvikelsen hos uppmätta värden men även för att med större säkerhet bekräfta att den uppmätta fasförskjutningen speglar verkligheten.

Gällande beräknad frekvens ur sampel från 12-bitar AD-omvandlare fungerade detta bra och gav minst lika bra upplösning som referensinstrumentet (± 0.01 Hz) så länge som prototypen fick mer än 4 beräknade värden för att stabiliseras. Detta gäller vid samtliga testade frekvenser i steg om 10 Hz upp till 100 Hz. Testvärdena för 70 Hz kom av misstag inte med och bör för säkerhets skull också undersökas. En observation som gjordes var att de första få (mindre än 5) beräknade frekvensvärdena hade en större och större avvikelse med ökande frekvens. Orsaken till detta är att startvärdet för den "gissade" frekvensen är 10 vilket innebär att gissningen blir längre och längre från rätt frekvens med ökande frekvens. Detta bör inte vara något problem då fenomenet endast uppstår vid systemstart/reset eller vid kraftiga frekvenssvängningar. Det är rekommenderat att vid kraftig frekvenssvängning till en lägre frekvens uppdatera den "gissade" frekvensen till ursprungsvärdet (eller lägre) för att undvika långsam uppdatering av frekvens. En möjlig lösning skulle kunna vara att automatisk ändra det "gissade" värdet om

det beräknade frekvensvärdet inte har uppdaterats inom en bestämd tid.

Felsökning under projektarbetet tog i vissa fall längre tid än väntat och resulterade i att utdatan från ingångsmodulen inte testades mot en CAN-buss. Istället avlästes beräknade värden i debuggern för mikrokontrollern. Det är lämpligt att utföra ytterligare tester med programvaran för CAN-bussen då denna kommer kräva en del prestanda och skulle i värsta fall kunna påverka den totala prestandan för ingångsmodulen.

Referenser

[1] ST Microelectronics, *STM32F446xx advanced ARM®-based 32-bit MCUs*, RM0390 Reference Manual, Rev. 3, juli 2017.

[2] Linear Technology, *LT1490/LT1491 Dual and Quad Micropower Rail-to-Rail Input and Output Op Amps*, 14901fb LT/LCG 0600 2K, Rev. B, 1996.

[3] ARM Limited, *CMSIS DSP*, version 1.5.2, 2017. URL: `http://www.keil.com/pack/doc/CMSIS/DSP/html/index.html`.

[4] Analog Devices, *Datasheet: 4-Channel, Low Noise, Low Power, 24-Bit, Sigma-Delta ADC with PGA and Reference*, AD7124-4, Rev. C, juli 2016.

[5] Fluke Corporation, *Datasheet: Fluke 83V and 87V Digital Multimeters*, 2161164 D-US-N, Rev. B, april 2005.

[6] Carlo Gavazzi, *Datasheet: Energy Analyzer Type EM111*, EM111 DS 110216.

[7] LEM, *Datasheet: Split core current transformer*, ATO-16-B333-D10, Version 0.

A Appendix

A.1 AD7124 Register

[4, pp. 77]

Addr.	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset	RW
0x00	COMMS	$\overline{WEN}$	R/$\overline{W}$	RS[5:0]						0x00	W
0x00	Status	$\overline{RDY}$	ERROR_FLAG	0	POR_FLAG	CH_ACTIVE				0x00	R
0x01	ADC_CONTROL	0			DOUT_$\overline{RDY}$_DEL	CONT_READ	DATA_STATUS	$\overline{CS}$_EN	REF_EN	0x0000	RW
		POWER_MODE		Mode				CLK_SEL			
0x02	Data	Data [23:16]								0x000000	R
		Data [15:8]									
		Data [7:0]									
0x03	IO_CONTROL_1	GPIO_DAT2	GPIO_DAT1	0	0	GPIO_CTRL2	GPIO_CTRL1	0	0	0x000000	RW
		PDSW	0	IOUT1			IOUT0				
		IOUT1_CH				IOUT0_CH					
0x04	IO_CONTROL_2	VBIAS7	VBIAS6	0	0	VBIAS5	VBIAS4	0	0	0x0000	RW
		0	0	VBIAS3	VBIAS2	0	0	VBIAS1	VBIAS0		
0x05	ID	DEVICE_ID				SILICON_REVISION				0x04	R
0x06	Error	0				LDO_CAP_ERR	ADC_CAL_ERR	ADC_CONV_ERR	ADC_SAT_ERR	0x000000	R
		AINP_OV_ERR	AINP_UV_ERR	AINM_OV_ERR	AINM_UV_ERR	REF_DET_ERR	0	DLDO_PSM_ERR	0		
		ALDO_PSM_ERR	SPI_IGNORE_ERR	SPI_SCLK_CNT_ERR	SPI_READ_ERR	SPI_WRITE_ERR	SPI_CRC_ERR	MM_CRC_ERR	ROM_CRC_ERR		
0x07	ERROR_EN	0	MCLK_CNT_EN	LDO_CAP_CHK_TEST_EN	LDO_CAP_CHK		ADC_CAL_ERR_EN	ADC_CONV_ERR_EN	ADC_SAT_ERR_EN	0x000040	RW
		AINP_OV_ERR_EN	AINP_UV_ERR_EN	AINM_OV_ERR_EN	AINM_UV_ERR_EN	REF_DET_ERR_EN	DLDO_PSM_TRIP_TEST_EN	DLDO_PSM_ERR_EN	ALDO_PSM_TRIP_TEST_EN		
		ALDO_PSM_ERR_EN	SPI_IGNORE_ERR_EN	SPI_SCLK_CNT_ERR_EN	SPI_READ_ERR_EN	SPI_WRITE_ERR_EN	SPI_CRC_ERR_EN	MM_CRC_ERR_EN	ROM_CRC_ERR_EN		
0x08	MCLK_COUNT	MCLK_COUNT								0x00	R
0x09 to 0x18	CHANNEL_0 to CHANNEL_15	Enable	Setup			0		AINP[4:3]		0x8001[1]	RW
		AINP[2:0]			AINM[4:0]						
0x19 to 0x20	CONFIG_0 to CONFIG_7	0				Bipolar	Burnout		REF_BUFP	0x0860	RW
		REF_BUFM	AIN_BUFP	AIN_BUFM	REF_SEL		PGA				
0x21 to 0x28	FILTER_0 to FILTER_7	Filter			REJ60	POST_FILTER			SINGLE_CYCLE	0x060180	RW
		0					FS[10:8]				
		FS[7:0]									
0x29 to 0x30	OFFSET_0 to OFFSET_7	Offset [23:16]								0x800000	RW
		Offset [15:8]									
		Offset [7:0]									
0x31 to 0x38	GAIN_0 to GAIN_7	Gain [23:16]								0x5XXXXX	RW
		Gain [15:8]									
		Gain [7:0]									

B Appendix

B.1 Initieringskod för SPI

```
/**
* @brief SPI1 init function
*/
static void SPI1_Init(void)
{
    // SPI peripheral address
    hspi1.Instance = SPI1;
    // SPI in master mode
    hspi1.Init.Mode = SPI_MODE_MASTER;
    // Enable bidirectional communication
    hspi1.Init.Direction = SPI_DIRECTION_2LINES;
    // Set the data packet size to 1 byte
    hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
    // SCK idles high
    hspi1.Init.CLKPolarity = SPI_POLARITY_HIGH;
    // Sample data on rising edge
    hspi1.Init.CLKPhase = SPI_PHASE_2EDGE;
    // Software chip select if multiple SPI present
    hspi1.Init.NSS = SPI_NSS_SOFT;
    // Set SCK speed
    hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_32;
    // Send most significant bit first
    hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
    // Motorola mode
    hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
    // No cyclic redundancy check
```

```
    hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
    // Not used
    hspi1.Init.CRCPolynomial = 10;
    if (HAL_SPI_Init(&hspi1) != HAL_OK)
    {
    _Error_Handler(__FILE__, __LINE__);
    }
}
```

B.2 DMA inställningar för 12-bitar AD-omvandlare

```
/**
* @brief Initializes the ADC DMA settings
*/
void HAL_ADC_MspInit(ADC_HandleTypeDef* hadc)
{
    GPIO_InitTypeDef GPIO_InitStruct;
    if(hadc->Instance==ADC1)
    {
        /* Peripheral clock enable */
        __HAL_RCC_ADC1_CLK_ENABLE();

        /* ADC1 GPIO Configuration
        PA0-WKUP      ------> ADC1_IN0
        */
        GPIO_InitStruct.Pin = GPIO_PIN_0;
        GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
        // No pull-resistor
        GPIO_InitStruct.Pull = GPIO_NOPULL;
        // Chosen settings to GPIO
        HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

        /* ADC1 DMA Init */
        hdma_adc1.Instance = DMA2_Stream4;
        hdma_adc1.Init.Channel = DMA_CHANNEL_0;
        hdma_adc1.Init.Direction = DMA_PERIPH_TO_MEMORY;
        // Same peripheral address
        hdma_adc1.Init.PeriphInc = DMA_PINC_DISABLE;
        // Increment pointer value after storing
        hdma_adc1.Init.MemInc = DMA_MINC_ENABLE;
```

```
        hdma_adc1.Init.PeriphDataAlignment = DMA_PDATAALIGN_WORD;
        // Store data as word
        hdma_adc1.Init.MemDataAlignment = DMA_MDATAALIGN_WORD;
        // Circular data buffer enabled
        hdma_adc1.Init.Mode = DMA_CIRCULAR;
        hdma_adc1.Init.Priority = DMA_PRIORITY_HIGH;
        hdma_adc1.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
        // Chosen settings to DMA controller
        if (HAL_DMA_Init(&hdma_adc1) != HAL_OK)
        {
            _Error_Handler(__FILE__, __LINE__);
        }

        // Associate DMA with ADC
        __HAL_LINKDMA(hadc,DMA_Handle,hdma_adc1);
    }
}
```

B.3 Initieringskod för 12-bitar AD-omvandlare

```
/**
* @brief ADC1 init function
*/
static void ADC1_Init(void)
{
    ADC_MultiModeTypeDef multimode;
    ADC_ChannelConfTypeDef sConfig;

    /* Configure the global features of the ADC (Clock, Resolution,
    Data Alignment and number of conversion) */
    hadc1.Instance = ADC1; // Select ADC master
    // Select clock speed
    hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
    // Number of data bits
    hadc1.Init.Resolution = ADC_RESOLUTION_12B;
    // Only used with multiple channels
    hadc1.Init.ScanConvMode = DISABLE;
    // Continuous conversion
    hadc1.Init.ContinuousConvMode = ENABLE;
    hadc1.Init.DiscontinuousConvMode = DISABLE;
    hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
    hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
    // Data alignment in peripheral
    hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
    // Only used with multiple channels
    hadc1.Init.NbrOfConversion = 1;
    // ENABLE with circular DMA buffer
    hadc1.Init.DMAContinuousRequests = ENABLE;
    // When to signal conversion complete
```

```
    hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
    if (HAL_ADC_Init(&hadc1) != HAL_OK)
    {
        _Error_Handler(__FILE__, __LINE__);
    }

    /* Configure the ADC multi-mode */
    // Simultaneous conversions, 3 ADC
    multimode.Mode = ADC_TRIPLEMODE_REGSIMULT;
    // Data stored as 1,2,3,1,2,1...
    multimode.DMAAccessMode = ADC_DMAACCESSMODE_1;
    // Cycles between samples
    multimode.TwoSamplingDelay = ADC_TWOSAMPLINGDELAY_5CYCLES;
    if (HAL_ADCEx_MultiModeConfigChannel(&hadc1, &multimode) != HAL_OK)
    {
        _Error_Handler(__FILE__, __LINE__);
    }

    /* Configure for the selected ADC regular channel its corresponding
    rank in the sequencer and its sample time. */
    sConfig.Channel = ADC_CHANNEL_0;
    sConfig.Rank = 1;
    sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
    if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
    {
        _Error_Handler(__FILE__, __LINE__);
    }
}
```

B.4 Initieringskod för PWM-modul

```
/**
* @brief PWM init function used for testing frequency calculation and
*   shunt measurements
*/
void pwm_init(void)
{
    /* Timer Output Compare Configuration Structure declaration */
    TIM_OC_InitTypeDef sConfig;
    htim3.Instance = TIM3;
    // 7500 Hz 50%: scaler=9, period=559, pulse=280
    htim3.Init.Prescaler          = 9;
    htim3.Init.Period             = 559;
    // 42 MHz
    htim3.Init.ClockDivision      = TIM_CLOCKDIVISION_DIV1;
    htim3.Init.CounterMode        = TIM_COUNTERMODE_UP;
    htim3.Init.RepetitionCounter = 0;
    if(HAL_TIM_PWM_Init(&htim3) != HAL_OK)
    {
        /* Initialization Error */
        Error_Handler();
    }

    /* Common configuration for all channels */
    sConfig.OCMode       = TIM_OCMODE_PWM1;
    sConfig.OCPolarity   = TIM_OCPOLARITY_HIGH;
    sConfig.OCFastMode   = TIM_OCFAST_ENABLE ;
    sConfig.OCNPolarity  = TIM_OCNPOLARITY_HIGH;
    sConfig.OCNIdleState = TIM_OCNIDLESTATE_RESET;
    sConfig.OCIdleState  = TIM_OCIDLESTATE_RESET;
```

```
    /* Set the pulse value
    or channel 1 */
    sConfig.Pulse = 280;
    if(HAL_TIM_PWM_ConfigChannel(&htim3, &sConfig, TIM_CHANNEL_1)
       != HAL_OK)
    {
        /* Configuration Error */
        Error_Handler();
    }

    /* Start PWM generation */
    if(HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1) != HAL_OK)
    {
        /* PWM Generation Error */
        Error_Handler();
    }
}
```

B.5 Frekvensfunktioner

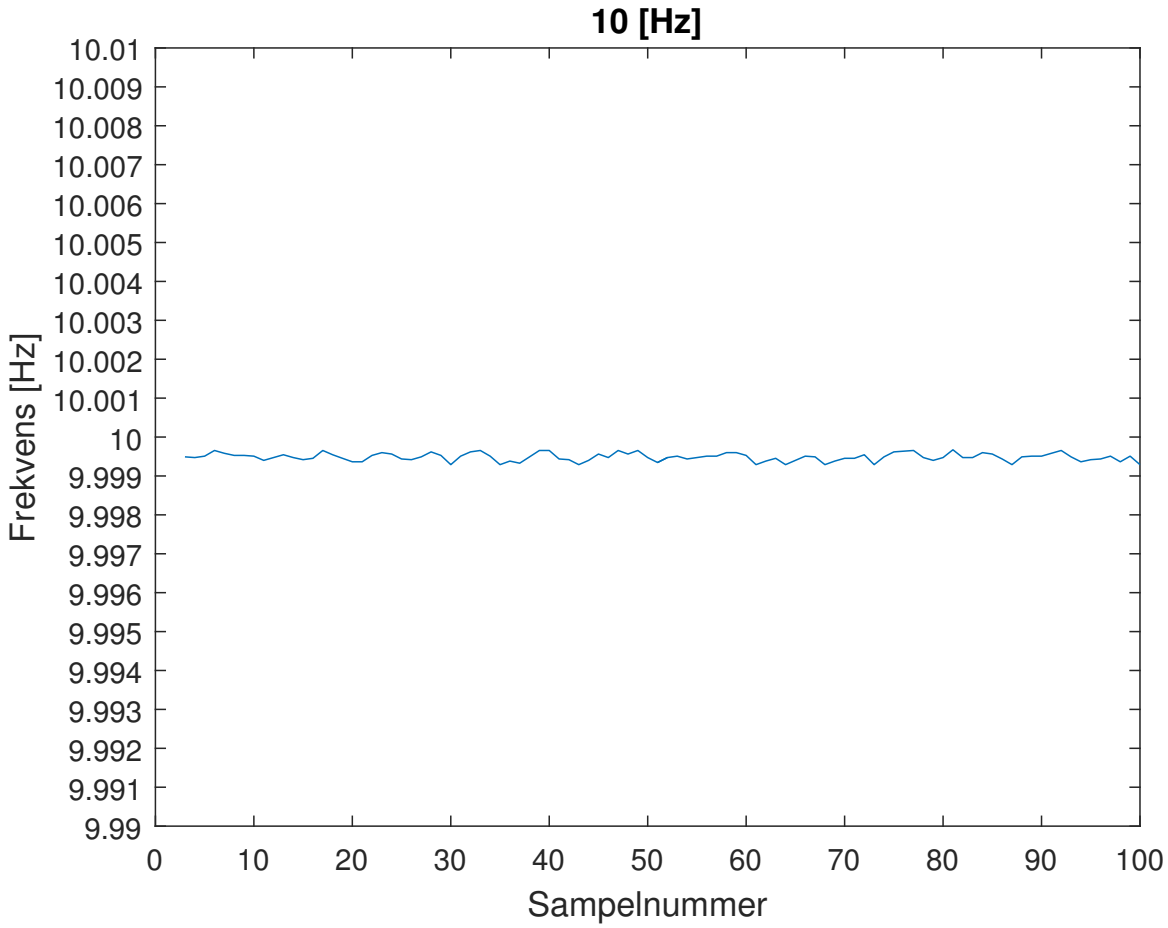
```
/**
* @brief Frequency measurement from ADC voltage samples
*   Takes the ADC source as an argument where the frequency is to
*   be calculated. Measures maximum amplitude to calculate the
*   user defined hysteres. Measures time dt it takes to find N
*   number of periods using peripheral TIM2. Sets flag
*   cycle_flag = 1 when N number of periods is found to signal it
*   is time to calculate the frequency.
*/
void freq_calc(float32_t adc_source)
{
    if(adc_source > max_amp)
    {
        // Measured maximum amplitude
        max_amp = adc_source;
        // Set hysteres high trigger
        hysteres_high = hysteres_high_scaler*max_amp;
        // Set hysteres low trigger
        hysteres_low = hysteres_low_scaler*max_amp;
    }
    if(adc_source >= hysteres_high && freq_flag)
    {
        if(freq_counter == 0)
        {
            // Reset cycle counter
            __HAL_TIM_SET_COUNTER(&htim2, 0);
        }
        freq_counter++;
        if(freq_counter == freq_scaler)
```

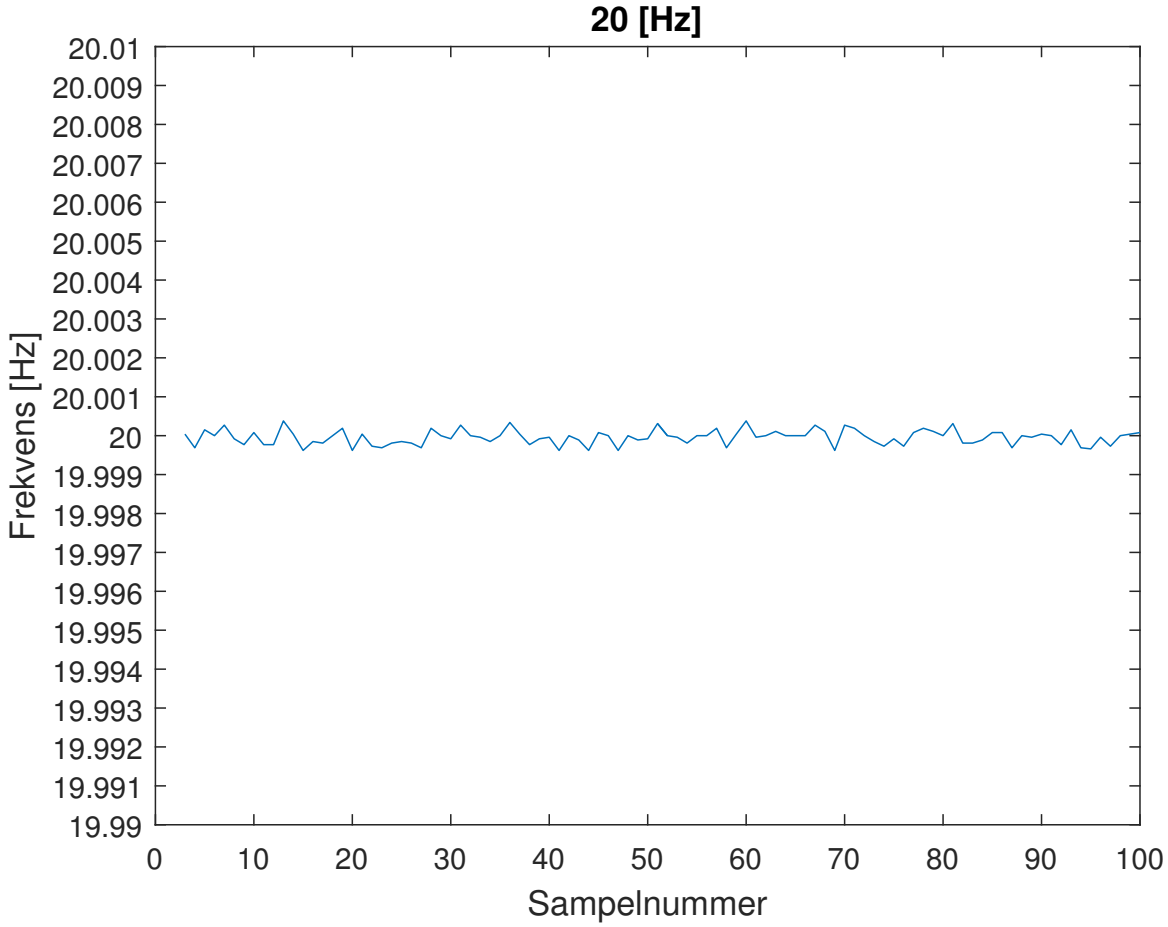
```
        {
            // Save cycle counter value
            cycle_time =
            (float32_t)__HAL_TIM_GET_COUNTER(&htim2);
        }
        // Prevent hysteres high trigger
        freq_flag =! freq_flag;
    }
    else if(adc_source <= hysteres_low && !freq_flag)
    {
        // Prevent hysteres low trigger
        freq_flag =! freq_flag;
        if(freq_counter == freq_scaler)
        {
            freq_counter = 0;
            cycle_flag = 1;
        }
    }
}

/**
* @brief Frequency calculation inside main program loop:
*   Calculate frequency when freq_scaler number of periods
*   have been detected. Set accuracy with count_accuracy in
*   global variables.
*/
if(cycle_flag)
{
    measured_frequency =
    ((float32_t)(freq_scaler-1))*((float32_t)500000/(cycle_time));
    cycle_flag =! cycle_flag;
```

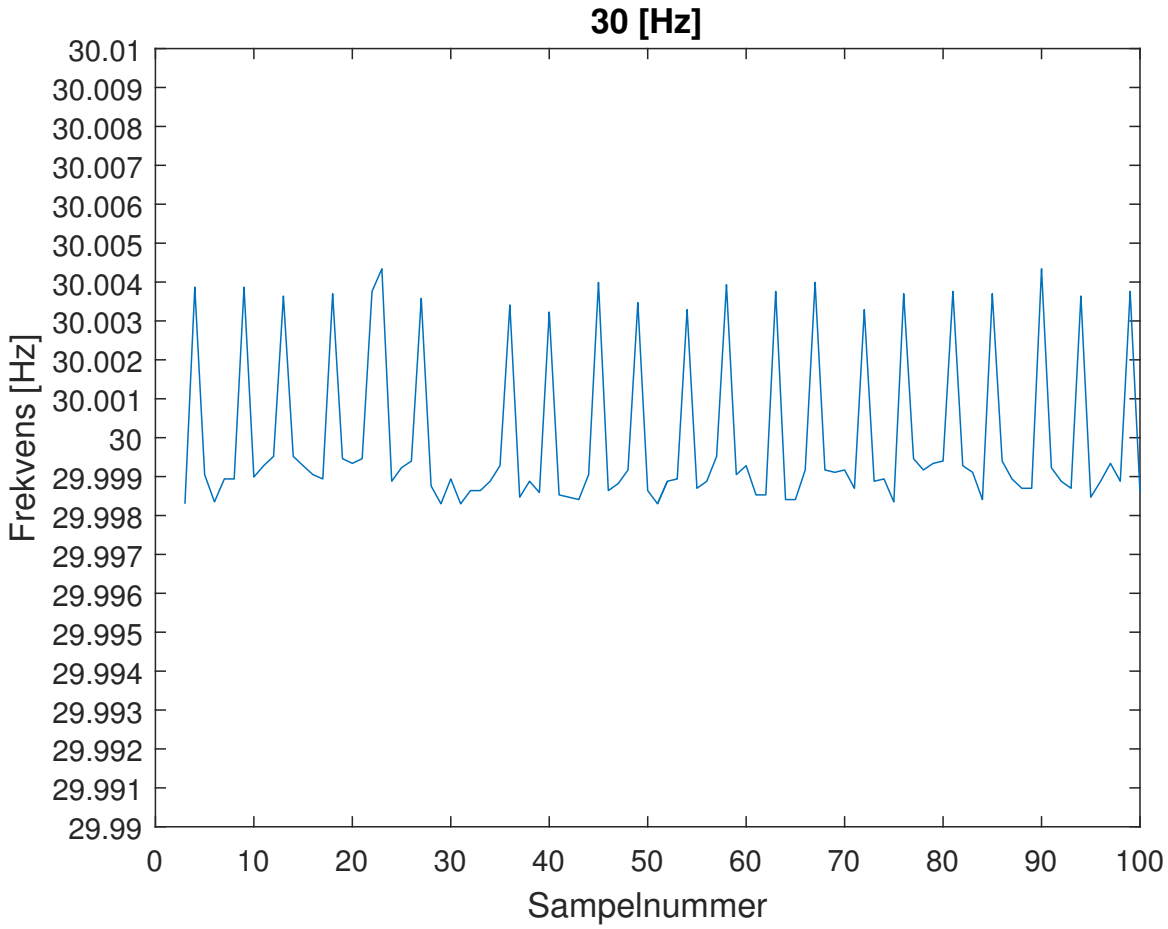
```
    // +2 to avoid div 0 or negative
    freq_scaler =
    (uint8_t)(count_accuracy*floor(ceil(measured_frequency)/10)+2);

}
```

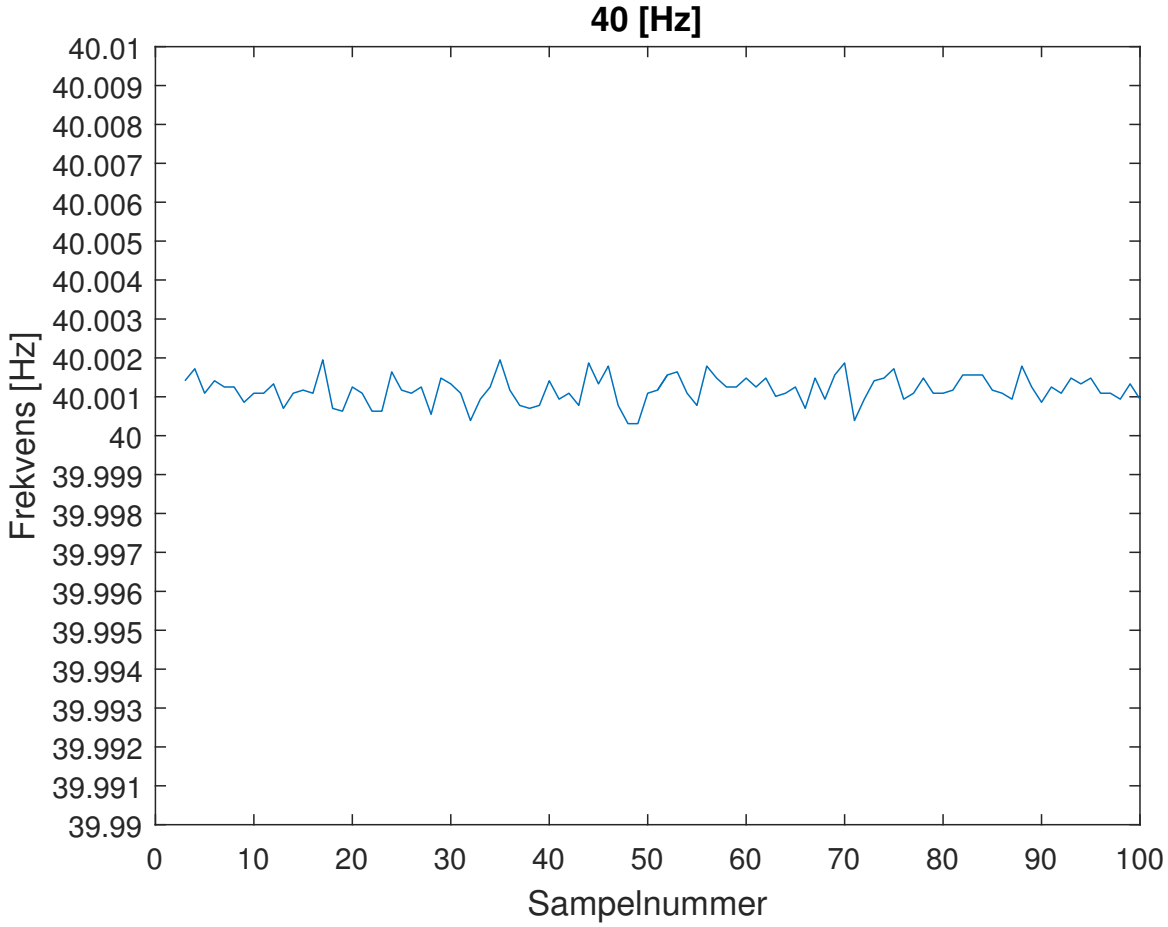
C Appendix

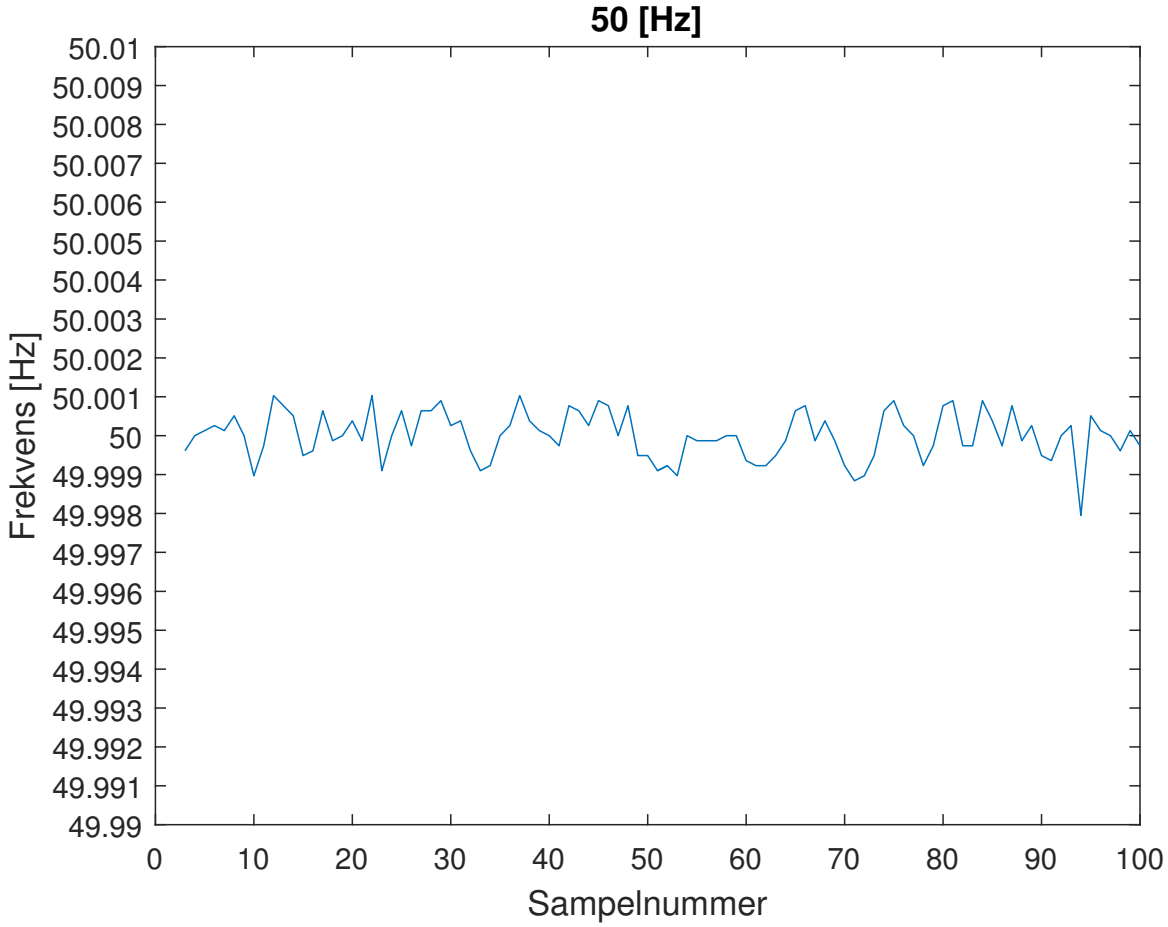
C.1 Frekvensmätningar

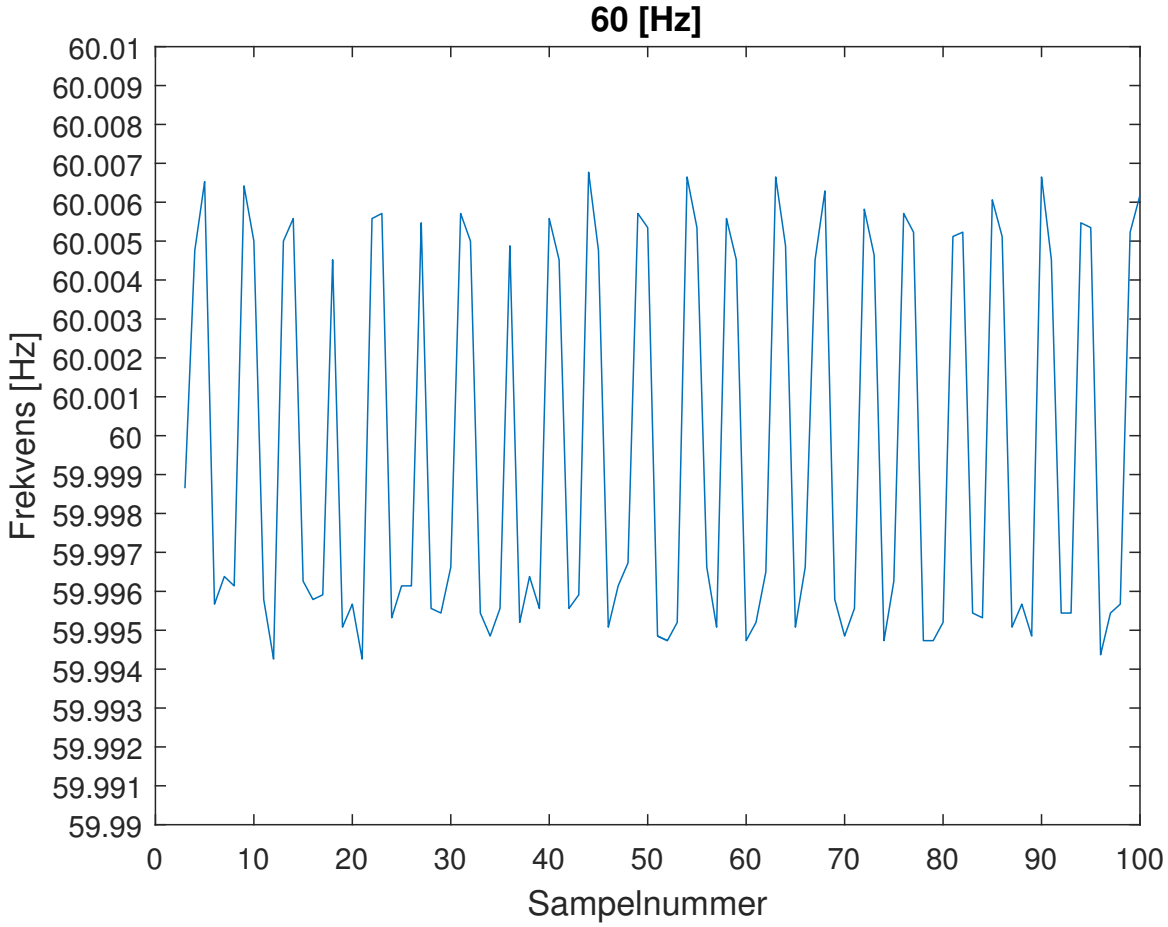


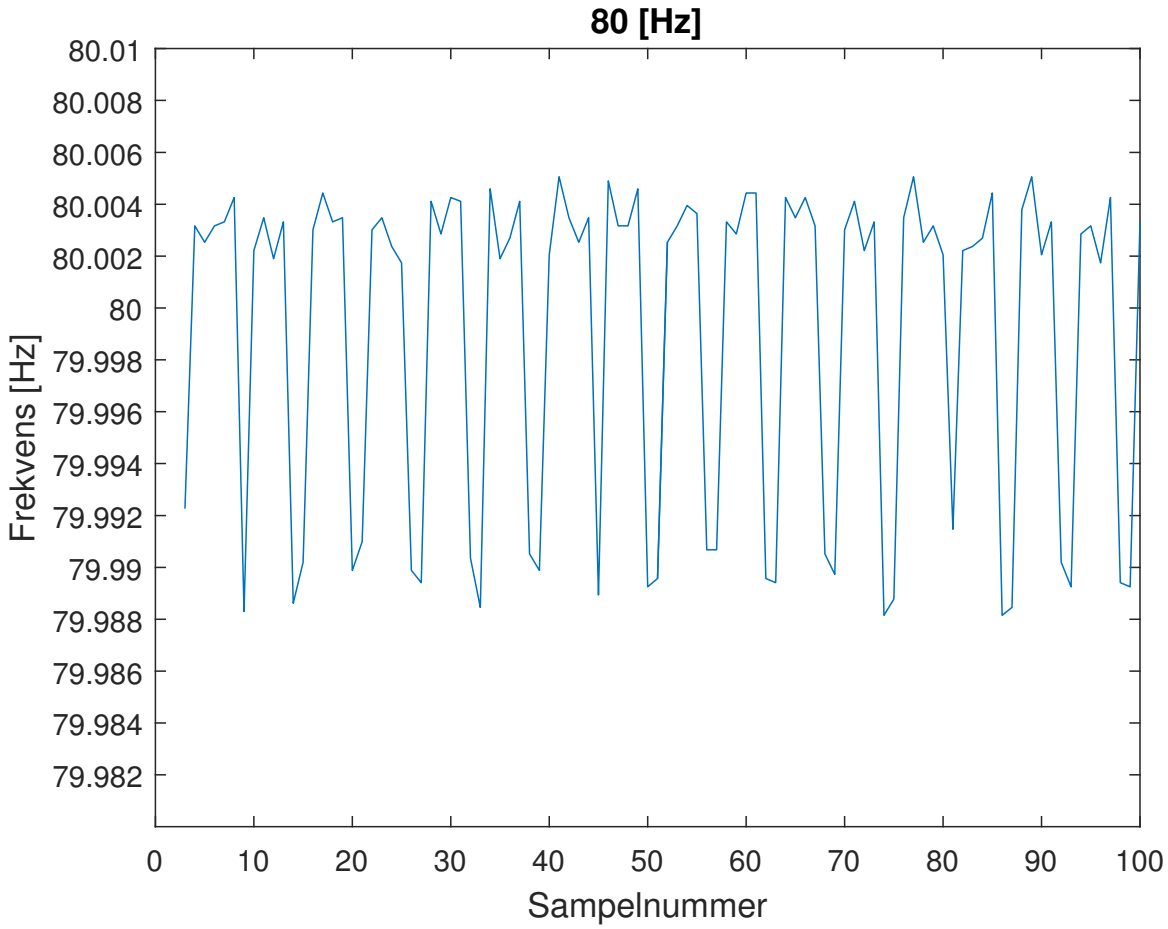

20 [Hz]
Frekvens [Hz]
20.01
20.009
20.008
20.007
20.006
20.005
20.004
20.003
20.002
20.001
20
19.999
19.998
19.997
19.996
19.995
19.994
19.993
19.992
19.991
19.99
0
10
20
30
40
50
60
70
80
90
100
Sampelnummer

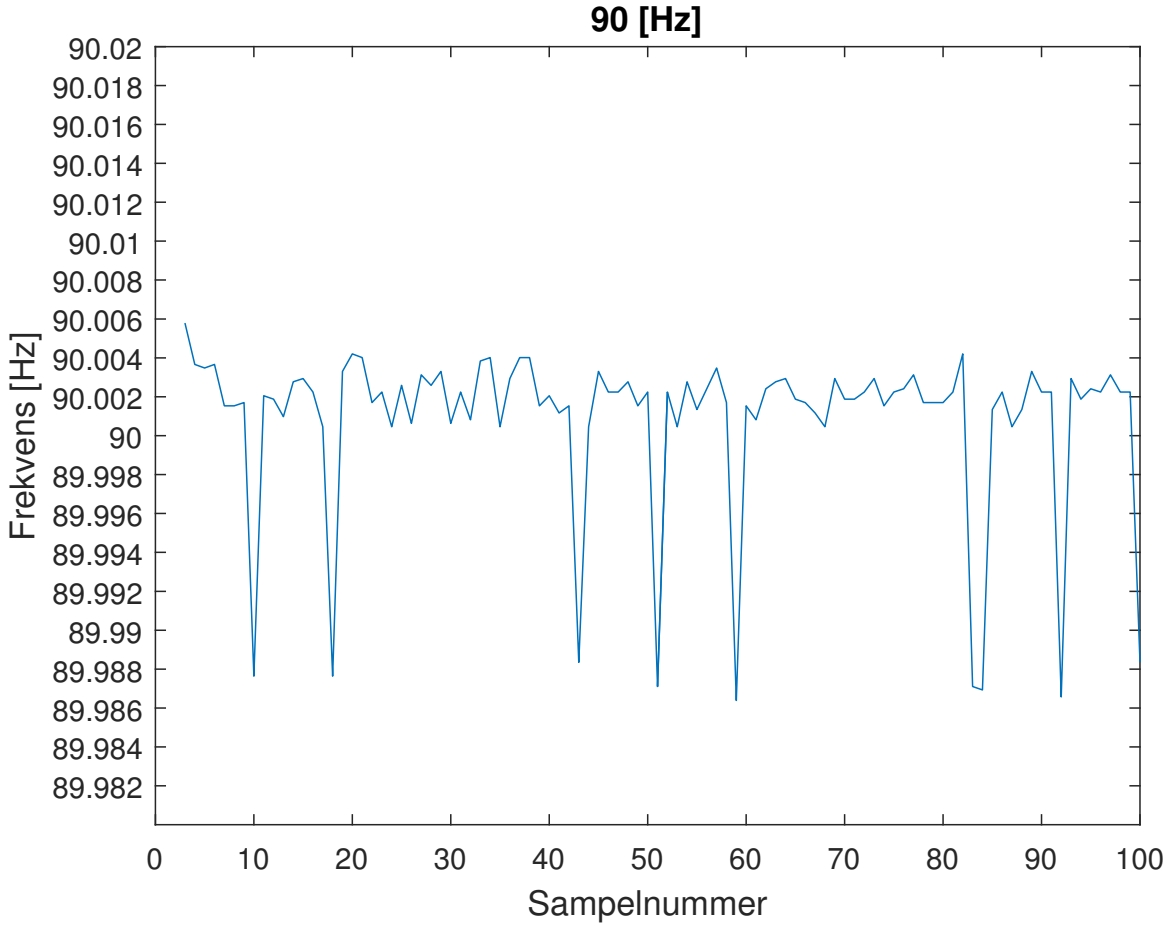

30 [Hz]
Frekvens [Hz]
30.01
30.009
30.008
30.007
30.006
30.005
30.004
30.003
30.002
30.001
30
29.999
29.998
29.997
29.996
29.995
29.994
29.993
29.992
29.991
29.99
0
10
20
30
40
50
60
70
80
90
100
Sampelnummer

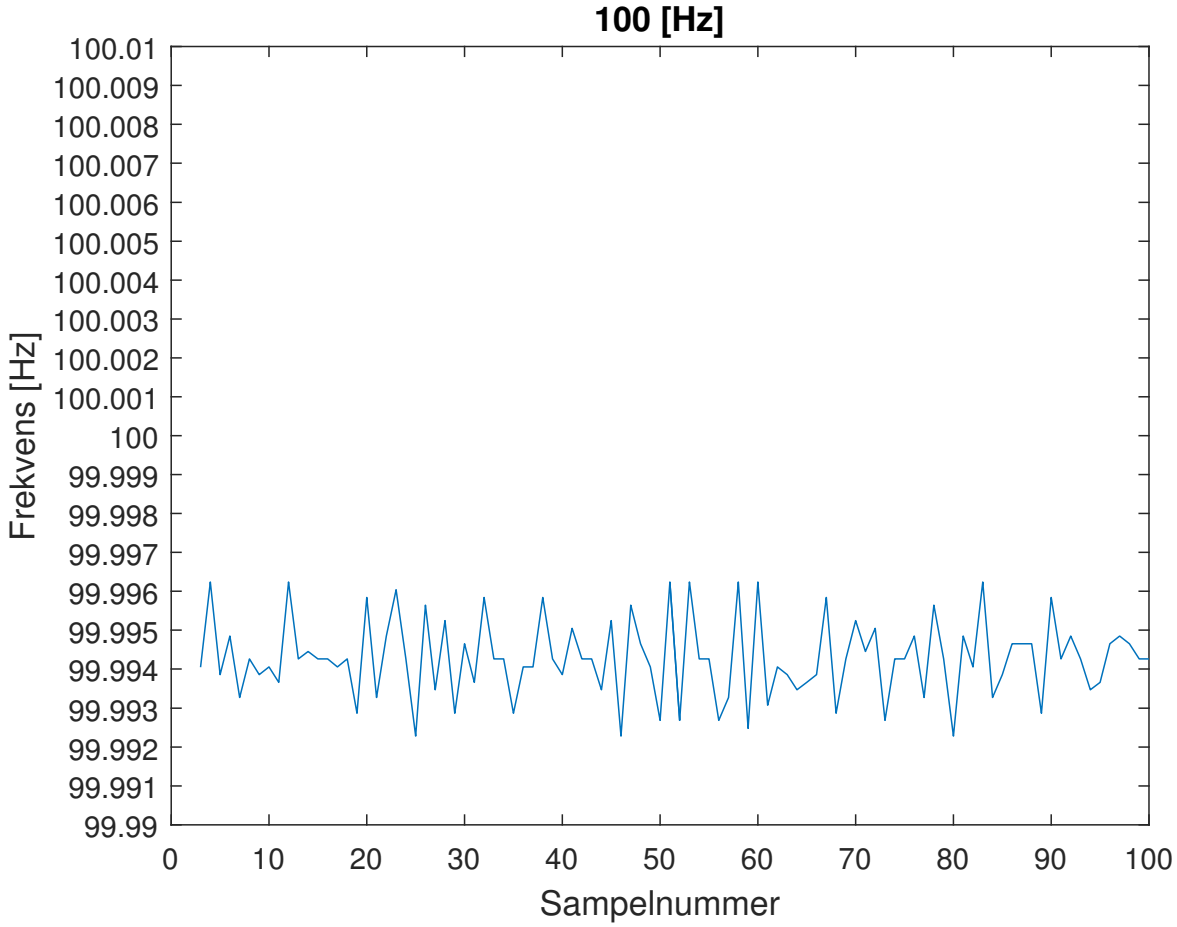

40 [Hz]
Frekvens [Hz]
40.01
40.009
40.008
40.007
40.006
40.005
40.004
40.003
40.002
40.001
40
39.999
39.998
39.997
39.996
39.995
39.994
39.993
39.992
39.991
39.99
0
10
20
30
40
50
60
70
80
90
100
Sampelnummer


50 [Hz]
Frekvens [Hz]
50.01
50.009
50.008
50.007
50.006
50.005
50.004
50.003
50.002
50.001
50
49.999
49.998
49.997
49.996
49.995
49.994
49.993
49.992
49.991
49.99
0
10
20
30
40
50
60
70
80
90
100
Sampelnummer


60 [Hz]
Frekvens [Hz]
60.01
60.009
60.008
60.007
60.006
60.005
60.004
60.003
60.002
60.001
60
59.999
59.998
59.997
59.996
59.995
59.994
59.993
59.992
59.991
59.99
0
10
20
30
40
50
60
70
80
90
100
Sampelnummer


80 [Hz]
80.01
80.008
80.006
80.004
80.002
80
79.998
79.996
79.994
79.992
79.99
79.988
79.986
79.984
79.982
Frekvens [Hz]
0
10
20
30
40
50
60
70
80
90
100
Sampelnummer


90 [Hz]
Frekvens [Hz]
90.02
90.018
90.016
90.014
90.012
90.01
90.008
90.006
90.004
90.002
90
89.998
89.996
89.994
89.992
89.99
89.988
89.986
89.984
89.982
0
10
20
30
40
50
60
70
80
90
100
Sampelnummer


100 [Hz]
Frekvens [Hz]
100.01
100.009
100.008
100.007
100.006
100.005
100.004
100.003
100.002
100.001
100
99.999
99.998
99.997
99.996
99.995
99.994
99.993
99.992
99.991
99.99
0
10
20
30
40
50
60
70
80
90
100
Sampelnummer