Høgskulen på Vestlandet

Bacheloroppgave

ELE350

Predefinert informasjon

Startdato:	08-05-2023 09:00 CEST	**Termin:**	2023 VÅR
Sluttdato:	22-05-2023 14:00 CEST	**Vurderingsform:**	Norsk 6-trinns skala (A-F)
Eksamensform:	Bacheloroppgave		
Flowkode:	203 ELE350 1 O 2023 VÅR		
Intern sensor:	Tom Kjøde		

Deltaker

Navn:	Espen Stornes
Kandidatnr.:	294
HVL-id:	591456@hvl.no

Informasjon fra deltaker

Egenerklæring *:	Ja
Inneholder besvarelsen konfidensielt materiale?:	Nei
Jeg bekrefter at jeg har registrert oppgavetittelen på norsk og engelsk i StudentWeb og vet at denne vil stå på vitnemålet mitt *:	Ja

Gruppe

Gruppenavn:	BO23EB-21 Corvus Energy
Gruppenummer:	13
Andre medlemmer i gruppen:	Tommy Steindal Bürkle Tobiassen

Jeg godkjenner avtalen om publisering av bacheloroppgaven min *

Ja

Er bacheloroppgaven skrevet som del av et større forskningsprosjekt ved HVL? *

Nei

BACHELOROPPGAVE

BO23EB-21 Nettilkobling og styring av testanlegg for brenselcellepakker

Espen Stornes
Tommy Steindal Bürkle Tobiassen

INSTITUTT FOR DATATEKNOLOGI,
ELEKTROTEKNOLOGI OG REALFAG

HØGSKULEN PÅ VESTLANDET
21. mai 2023

Rapportens tittel:	Dato: **21.05.2023**
Nettilkobling og styring av testanlegg for brenselcellepakker	Rapportnr: **BO23EB-21**
Forfattere:	Rev.nr: **0.20**
Tommy S. B. Tobiassen **Espen Stornes**	Antall sider m. vedlegg: **168**
Høyskolens veileder:	Studieretning: **AUTB20**
Tom Kjøde	Gradering: **Åpen**
Merknader: **Vi tillater at oppgaven publiseres**	

Oppdragsgiver:	Oppdragsgivers Referanse:
Corvus Energy AS	**HVL - Grid connection of fuel cell test plant**
Oppdragsgivers kontaktpersoner ink. kontaktinfo:	
Åge Ulvøy	**aulvoy@corvusenergy.com**
Espen Jernquist	**ejernquist@corvusenergy.com**

Revisjon	**Dato**	**Status**	**Utført av**
0.11	03.apr.2023	Første utkast	Tommy T. og Espen S.
0.12	19.apr.2023	Revidert utkast	Tommy T. og Espen S.
0.13	23.apr.2023	Omformerdel	Espen S.
0.14	24.apr.2023	Kapitler om lastbank	Tommy T.
0.15	2.mai.2023	Om lover og regler	Tommy T.
0.16	6.mai.2023	Kapittel for testing	Tommy T.
0.17	10.mai.2023	Diskusjonskapittel	Tommy T. og Espen S.
0.18	16.mai.2023	Konklusjon	Tommy T.
0.19	18.mai.2023	Strukturering av oppgave	Tommy T. og Espen S.
0.20	21.mai.2023	Korrekturlesing og ferdigstilling	Tommy T. og Espen S.

Forord

Dette er den avsluttende oppgaven for studiet; bachelor i ingeniørfag, automatisering med robotikk. Arbeidet med oppgaven har blitt utført på Høgskulen på Vestlandet, Institutt for datateknologi, elektroteknologi og realfag, og i tett samarbeid med Corvus Energy AS. Brenselcellepakken er enda under utvikling, dermed kan informasjon som er relatert til denne være utdatert.

Vi vil gjerne takke våre flinke veiledere for god hjelp. Vi ønsker også å rette en stor takk til Corvus Energy for å være veldig imøtekommende og behjelpelige med alle våre spørsmål. Vi vil også takke for at vi har fått mulighet til å skrive denne oppgaven for dere, og at vi har fått lov til å sitte hos dere å skrive.

Våre nærmeste fortjener også en stor takk for støtte under alle de hektiske dagene i løpet av studie. Tusen takk for støtte under tunge eksamensperioder, og støtte under sluttinnspurten mot endt grad.

Sammendrag

Corvus Energy jobber med å utvikle en brenselcellepakke (FCP) til maritimt bruk. Dette vil gjøre det mulig å benytte hydrogen fra fornybare kilder for å redusere utslipp fra maritim industri. Ved det nåværende testanlegget for brenselcellepakkene brennes all effekt fra testing av i en lastbank. Dette prosjektet ser på muligheten for å legge til en omformer for å føre strømmen inn på strømnettet og vi skal lage et kontrollsystem som regulerer komponentene i testanlegget.

Våre studier viste at omformere for solcellebruk ikke var egnet for oppgaven, men vi fant flere andre omformere som oppfylte kravene. Å regulere lastbanken slik at den dro litt mer eller litt mindre effekt, enn det som ble produsert av brenselcellepakken, ble sett på som den beste løsningen.

Corvus Energy valgte en annen omformer som vi ikke hadde kjennskap til, før vi fikk presentert vår løsning. Vi ville valgt en løsning fra Danfoss ettersom disse omformerne også er godkjent til maritim bruk. Dette ble lagt litt vekt på, ettersom dette er relevant for bruksområdet til brenselcellepakken. Kontrollsystemet ble implementert på en programmerbar logisk styring (PLS), ved bruk av en tilstandsmaskin for sikker oppstart og nedstengning av komponentene. En regulering som er veldig lik av/på-regulering ble brukt for å styre lastbanken.

Den endelige testen viste at kontrollsystemet fungerte, men med noen feil. Feilene ble fikset før endelig levering til Corvus Energy. Grunnet mangel på tid var det noen funksjoner som ikke ble implementert og grundig testing over tid ble aldri utført.

Abstract

As a part of decarbonizing the maritime industry, Corvus Energy is developing a hydrogen fuel cell pack (FCP) for maritime use. This will enable the use of hydrogen generated from renewable energy sources to reduce emissions. At the current FCP test site, all the power produced is burned off in a load bank. This project looks in to the option to add an inverter to the test site that will return power to the grid, and a control system to regulate the components on the test site.

Studies showed that solar inverters were not suited for the task, and several other inverters were found that better suited the required specifications. Regulating the loadbank to draw slightly less or slighly more power (depending on battery state) than the FCP produced was found to be the best solution.

Corvus Energy decided to go with another inverter solution before we could propose ours. We would propose using an inverter from Danfoss, that is also suitable for marine use, as it is the most relevant due to the intended use of the FCP. The control system was implemented on a programmable logic controller (PLC), and we're using a state machine for a safe start up and shutdown sequence. A regulation similar to on/off regulation was implemented to control the loadbank.

The final test showed that our control system worked, albeit with some bugs. The bugs were fixed before the final delivery.

Due to limited time, some features were not implemented, and thorough testing over time was never completed.

Innhold

1 Innledning

På grunn av klimaendringer er det ønskelig å redusere utslipp fra maritim skipsfart. Vi må fremdeles ha energikilder til skip, og derfor er det viktig å ha et bra alternativ til de fossile energikildene vi bruker i dag. Markedet begynner å bli klar for denne teknologien, og det er stor politisk vilje.

Industrien får mange insentiver og støtte til utvikling i en grønn retning. En av bedriftene som har jobbet med elektrifisering av maritime installasjoner lenge, i form av batteripakker, er Corvus Energy. De har allerede kommet svært langt i utviklingen av sin brenselcellepakke for utslippsfrie fartøy, og ønsker å videreutvikle sitt testanlegg. De ønsker å gjøre testoppsett enklere å bruke, og de ønsker å ha mulighet til å levere produsert energi ut på strømnettet. I begynnelsen av denne prosessen, fikk vi være litt med på avklaringene rundt hvorvidt, og hvordan kraft kan leveres ut på nettet.

I løpet av denne prosessen, ble det behov for å endre oppgaven en god del. Det ble bestemt at vi skulle fokusere mer på styring av eksisterende utstyr på testanlegget, ettersom Corvus Energy hadde løst problemstillingen med en av sine leverandører. Endringen i oppgaven ville gi oss en bedre forutsetning for testing og utvikling, og det ville gi Corvus Energy noe de vil få bruk for i denne fasen av produktutviklingen.

Corvus Energy ønsker å ta i bruk programmet vi har utviklet, for testing av første generasjons brenselcellepakker fra Corvus Energy. Programmet vil styre testanlegget, eller bli brukt som grunnlag til å videreutvikle styringen for testanlegget. Det er også mulig at programmet kan bli brukt til å styre testanlegget under produksjonsfasen, hvor brenselcellepakker blir testet før de blir levert til kunde.

I denne oppgaven vil vi gå gjennom og begrunne våre valg gjennom prosessen. Nærmere spesifisert, i forbindelse med testanlegget for brenselceller, og i forbindelse med styringen av testanlegget.

1.1 Om Corvus Energy

Corvus Energy ble startet opp i 2009, og de begynte som et lite privat canadisk selskap som produserte batterimoduler til maritimt bruk. I dag har de flere store eiere som ønsker investering i grønn energi. Corvus Energy har et sterkt maritimt DNA, de har kultur for innovasjon, og har et stort fokus på framtiden. De er pionerer på batteriteknologi, og de jobber nå hardt med å utvikle brenselcellepakker for maritimt bruk.

Corvus Energy har en veldig stor andel av markedet innenfor batteriteknologi i maritim sektor. De har over 200 ansatte over hele verden, og de har et stort globalt salgs- og service-nettverk.

I Norge startet Corvus Energy med 1 ansatt i 2015. I 2019 flyttet Corvus Energy hovedkontoret til Bergen for å være nærmere sitt hovedmarked. De bygde også en helautomatisert fabrikk i Norge for produksjon av sine batterimoduler. I dag er de en plass mellom 100 og 200 ansatte her i Norge.[1]

Corvus Energy har også sterkt fokus på bærekraft. Produktene deres er med på å redusere klimagassutslipp fra maritim sektor, og per dags dato har de bidratt til å redusere verdens utslipp med over 7 mrd. kg CO_2.[2] De jobber også aktivt med andre bærekraftsmål, som for eksempel initiativer for studenter, og at opp til 99% av deres batterisystemer kan gjenvinnes[3]. De er også ISO 14001:2015 sertifisert,[4] som omhandler miljøledelse.

Videre i oppgaven vil vi skrive Corvus, i stedet for Corvus Energy.

Figur 1: Corvus Energy er oppdragsgiver.

1.2 Oppgavebeskrivelse

1.2.1 Opprinnelig oppgavebeskrivelse

I denne oppgaven er det ønskelig å utrede og dimensjonere nettilkobling av et brenselcellesystem. Systemet består i dag av en DC-bus, hvor brenselcellepakke, batteri og lastbank er koblet sammen. Nettilkoblingen (omformeren) som vil erstatte lastbanken, skal utredes i minst to alternativer. En konfigurasjon med, og en uten batteri er ønskelig. Det er også ønskelig å se på kontrollsystem til å integrere nettkoblingen sammen med kontrollsystemer for brenselcelle-testanlegget.

1.2.2 Endret behov for oppdragsgiver

Som nevnt, ble Corvus sitt behov endret under avklaringsfasen i prosjektet, og vi måtte derfor endre en del på vår leveranse. Kravet om omformer og styring av omformer, gikk vekk ettersom Corvus fikk avklart hvordan de skulle løse dette internt. På grunn av dette gikk mye av elkraftdelen ut av oppgavespesifikasjonen, men vi har tatt med arbeidet vi gjorde i denne prosessen som en del av problemanalysen. Vi har lagt mest arbeid i programmeringsdelen.

1.2.3 Endelig Oppgavespesifisering

Dagens testanlegg hos Corvus består av en brenselcellepakke, en batteripakke, en lastbank og støtteelementer til disse (kjøling, sirkulasjonspumper, med mer). Vårt oppdrag er å styre alle komponenter utenom støttesystem. Det må også være tatt hensyn til en fremtidig utvidelse av testanlegget, i form av en DC/AC-omformer som kan kobles til strømnettet via en transformator.

Hoveddelen av oppgaven er at vi skal lage et kontrollsystem for anlegget. Det er ønskelig med et brukergrensesnitt til å kunne visuelt se hva som foregår under testing. Det er også en fordel å ha enkel logging av hendelser, men dette er ikke et krav.

Tilkobling mot strømnettet kan også utløse en del krav fra netteier og myndigheter. Det er ønskelig fra Corvus at vi ser litt på hva dette betyr for dem, og for testanlegget.

1.3 Begreps-, og forkortelsesliste

A - Ampere. Måleenhet for elektrisk strøm.

AC - Alternating Current. Vekselstrøm.

AMS - Automatisk strømmåler. Smart strømmåler, oftest med kommunikasjon.

C - C-rate. Opp- eller utladningshastighet. Angir hvor raskt batteriet kan lades/utlades i løpet av 1 time, i forhold til kapasiteten.

DC - Direct Current. Likestrøm

Distribusjonsnettet - ”Enkelt forklart”, strømnettet mellom veggen til kunden og ”monstermastene”, se figur 5 på side 21

FCP - Fuel Cell Pack. Brenselcellepakke.

FEF - Forskrift for elektriske forsyningsanlegg

FOL - Forskrift om leveringskvalitet i kraftsystemet. Leveringskvalitetsforskriften.

FSE - Forskrift om sikkerhet ved arbeid i elektriske anlegg

HMI - Human-Machine Interface. Også kalt operatørpanel.

Hz - Hertz. Måleenhet for frekvens. Angir antall svingninger/sykluser per sekund.

IP - Internet Protocol. Kommunikasjonsprotokoll som danner grunnlaget for internett.

kWh/kg - kilowattimer per kilo. Måleenhet for energiinnhold per kilo masse.

MPPT - Maximum Powerpoint tracking. Styringsmetode for omformere.

NDA - Non-Disclosure Agreement. Taushetserklæring.

Netteier - Ansvarlig for vedlikehold og drift av strømnettet i regionen.

NVE - Norges vassdrags- og energidirektorat

PLCnext - Phoenix Contacts sitt PLS system.

PLCnext Engineer - Phoenix Contacts sitt program for å programmere PLS.

PLS - Programmerbar Logisk Styring. Programmerbar datamaskin for styring av automasjonsprosesser.

Plusskunde - Strømkunde som også leverer strøm ut på distribusjonsnettet.

s - Sekund. Måleenhet for tid.

S - Siemens - Måleenhet for elektrisk konduktans. Matematisk symbol: G

SOC - State of Charge. Energi tilgjengelig i batteripakke, oppgis i %.

TCP - Transmission Control Protocol. Nettverksprotokoll.

TCP/IP - Transmission Control Protocol/Internet Protocol. Nettverksprotokoll.

TLS - Transport Layer Security. Nettverksprotokoll.

UDP - User Datagram Protocol. Nettverksprotokoll.

UINT - Unsigned Integer. Positivt heltall på 16 bit

UDINT - Unsigned Double Integer. Positivt heltall på 32 bit

USINT - Unsigned Short Integer. Positivt heltall på 8 bit

V - Volt. Måleenhet for spenning. Matematisk symbol: V

W - Watt. Måleenhet. Definert som 1 joule per sekund. Matematisk symbol: P

Word - 16 bit, 2 byte.

Ω - Ohm. Måleenhet for elektrisk motstand.

2 Kravspesifikasjon

2.1 Original kravspesifikasjon

Kravspesifikasjonen ble endret underveis i prosessen, den originale som ikke er gjeldende lenger er som følger:

- Koblingsskjema over det elektriske.
- Koblingsskjema over styringssystem
- Budsjett.
- Komponentliste.
- Eget styresystem som styrer batteripakke, brenselcellepakke og omformer.
- Utredning for mulighet for å koble seg på nettet.
- System uavhengig og avhengig av batteri.
- Helst et system hvor man kan koble ut batteripakken.

2.1.1 Krav til omformer fra Corvus

Følgende krav/anbefalinger ble også anbefalt til spesifikasjoner på omformer i innledende fase av prosjektet:

- Kan styres over Modbus TCP
- Effekt min.: 320 kW
- Anbefalt min. spenning: 400 V
- Anbefalt maks. spenning: 1500 V
- Strøm min.: 500 A
- Anbefalt at omformer er toveis for å lade opp kondensator

2.2 Gjeldende kravspesifikasjon

I denne oppgaven så kan vi dele kravene inn i fire kategorier, hvorav det første kravet var en del av den originale oppgaven:

- Krav til omformer fra Corvus
- Krav fra netteier og myndigheter
- Krav til styringssystem
- Krav til dokumentasjon av styresystem

2.2.1 Krav fra netteier og myndigheter

Det må beskrives litt om hvilke krav som stilles fra netteier i regionen, og hvilke krav som blir stilt fra myndigheter, for en fremtidig tilkobling mot strømnettet.

2.2.2 Krav til styringssystem

Styringssystemet må kunne:

- Styre brenselcellepakke over Modbus TCP.
- Styre batteripakke over Modbus TCP.
- Styre lastbank over Ethernet TCP.
- Mulighet for å koble til nettomformer senere.
- Starte og stoppe brenselcellepakke, batteripakke og lastbank i riktig rekkefølge.
- Håndtere kraftforespørsler fra HMI og Modbus TCP.
- Holde spenning på DC-bus innenfor riktig område.

2.2.3 Krav til dokumentasjon av styresystem

- Oversikt over styresystem og programstruktur
- Kommentarer i koden på engelsk

- Hvordan bruke styresystemet fra HMI
- Hvordan bruke styresystemet fra Modbus, oversikt over registeradresser etc.
- Hvordan endre tilkoblingsparametre (IP-adresser)

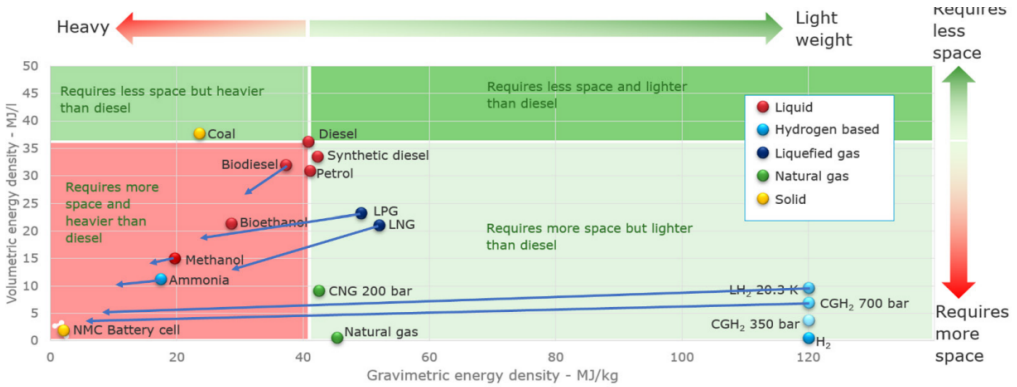
3 Analyse av problemet

Som innledning til analysen begynner vi med å se på det store bildet, deretter vil vi jobbe oss inn mot den tekniske analysen i oppgaven. I slutten av den tekniske delen tar vi en risikovurdering, før vi går inn på styringen av testanlegget.

3.1 Veien mot grønn skipsfart

På grunn av klimaendringer er det ønskelig å redusere utslipp fra maritim skipsfart. Det er derfor ønskelig å fase ut fossile energikilder som blir brukt i dag. Vi må fremdeles ha energikilder til skip, og derfor er det viktig å ha et bra alternativ som kan erstatte de mest brukte løsningene. En av veiene til skipsfart med lavere utslipp, kan være elektrifisering.

En metode dette kan løses på, er at skipets energikilde er batteripakker som kan lades når skipet er i havn. I dette tilfellet, er ladetid og størrelsen til batteripakkene en utfordring for lengre strekninger. Med dagens batteriteknologi, egner ikke batterier seg for mellomlange strekninger, og hydrogen er derfor sett på som den beste kortsiktige erstatningen for fossilt brensel.[5] Skip som går mest på mellomlange strekninger utgjør den største andelen av trafikken i dag. På langdistanse ser man på løsninger der skipene går på ammoniakk/metanol, men i denne oppgaven kommer vi ikke til å gå inn på dette.



Figur 2: Sammenligning mellom forskjellige drivstoff, hentet fra [6]. Pilene angir antatt tetthet når lagringssystemet er tatt i betraktning.

Ved hjelp av hydrogen kan man lagre 33 kWh/kg, mot batterier som kan lagre

0.177 kWh/kg med dagens teknologi. Til sammenligning har diesel 12 kWh/kg.[7]

Løsningen på å benytte hydrogen som drivstoff, kan være i form av brenselcelle-teknologi. Brenselcellen produserer da elektrisitet for å så benyttes til fremdrift via en elektromotor som driver propellene, som igjen skyver skipet.

En svært enkel forklaring av problemet er at høyere energiinnhold per kilo er ønskelig, da dette bidrar til lavere vekt av farkosten. Dette kan igjen bidra til at farkosten er mer energieffektiv, og bruke mindre energi for å flytte samme last over samme distanse. En annen stor fordel med hydrogen, er at det er raskere å fylle hydrogen, enn å lade batterier. Dette betyr mindre tid til kai, og mer driftstid for farkosten.

Det er en del utfordringer med hydrogen også, en utfordring er at det er svært brennbart og det har et stort spenn i konsentrasjoner i luft hvor det er eksplosjons-farlig. Hydrogen har i tillegg svært små atomer/molekyler og er dermed et av de vanskeligste stoffene å lagre, da det er veldig lite som skal til for å få en lekkasje. Hydrogen krever også mye plass for å lagre det på en sikker måte, og dette kan føre til at totalsystemet for lagring blir tungt og stort. Hydrogen som skal lagres flytende, krever også svært lav temperatur, og hydrogen som skal lagres i gassform er under svært høyt trykk.

Corvus har designet et system for å håndtere hydrogen på en trygg måte, dette kaller de ”inherently gas safe”. Dette går ut på at gassen er lagret og transportert via rør-i-rør.[8] Hvis det er en hydrogenlekkasje, vil det lekke ut i ytterrøret. I yt-terrøret blir det sirkulert en inert gass, nitrogen, som blir overvåket for å oppdage en lekkasje før det oppstår farlige situasjoner. Hvis det ikke blir benyttet en slik løsning, skaper det svært mange utfordringer knyttet til sikkerhet og drift av slike anlegg i maritime applikasjoner.

3.1.1 Politisk vilje

Regjeringen sier ofte at Norge må gripe mulighetene i det grønne skiftet. Vi har en krevende omstillingsprosess for å innfri våre klimaforpliktelser. Den maritime verdenen, er en stor sektor som også må være med å bidra. Det er store muligheter

for utslippsbesparelser i denne næringen, og det er derfor viktig å komme med gode løsninger som kan bidra til store utslippskutt.

Den maritime næringen, er en svært viktig sektor for Norge, og vi er historisk sett store på innovasjon innenfor denne sektoren. Dette er også en sektor med mange ansatte, og det er derfor viktig at vi klarer å holde følge med det grønne skiftet, i å skape nye grønne arbeidsplasser i takt med den raske utviklingen i bransjen. Vår oppgave handler om ny teknologi under utvikling, som er et svært viktig bidrag for at verden skal få vridd seg over til mer miljøvennlige løsninger for energikilder i skip. Systemet som denne oppgaven omhandler, er en viktig del av utviklingsprosessen for brenselcellesystemet til Corvus. Produktet som blir utviklet, vil sannsynligvis bidra til å redusere klimautslippet til den globale shippingindustrien. Brenselceller kan også bidra til å at vi kan fortsette vår grønne vekst, og kan bidra til at Norge kan klare å innfri flere av sine klimaforpliktelser.[9]

> *"Regjeringen vil at Norge skal gripe mulighetene i det grønne skiftet. Norge må gjennom en krevende omstilling for å innfri våre klimaforpliktelser. Vårt arbeidsliv må være grønt, smart og nyskapende. Den norske omstillingen må gjennomføres på en kostnadseffektiv måte. Samtidig må den gi vekstmuligheter for norsk økonomi og bidra til norsk eksport av miljøvennlige løsninger. Fornyelse av norsk skipsfart er en viktig motor i omstillingen."* - fra Handlingsplan for grønn skipsfart, utarbeidet av regjeringen.[9]

Etterspørsellen for hydrogenteknologi er stor, og den er økende. Dette er på grunn av at den maritime sektoren har et ambisiøst mål om å redusere sine utslipp med 50% innen 2050.[5] Dette betyr at det allerede er mange rederier som har forpliktet seg til å ta i bruk denne type teknologi, og som må ta del i det grønne skiftet for å kunne også være konkurransedyktige i fremtiden.

Regjeringen har også satt et krav om at alle turistskip og ferger som skal inn i verdensarvfjordene, skal være utslippsfrie senest innen 2026.[9] Norge har ambisiøse mål, og gode rammebetingelser for grønn utvikling av skipsfart.

Corvus leverte batterier til den første batterifergen i Norge, M/S "Ampere".[11] Dette skipet var også verdens første batteridrevne bilferge, som ble satt i rute i 2015. Norge er også et land med høy tilgjengelighet av grønn elektrisitet. Vi tenker derfor også at, i fremtiden kan produksjon av hydrogen være et lurt satsningsområde, gitt at Norge får økt kraftoverskudd.

Figur 3: Batterifergen M/S ”Ampere”, hentet fra [10]

Med alt tatt i betraktning, mener vi at det er et stort marked for denne typen teknologi. Når teknologien modner, vil også tilgjengeligheten og prisene på hydrogen og hydrogenteknologi gå ned, og det vil bli økt tilgjengelighet på hydrogen.

3.1.2 Samarbeid med Toyota

I 2020 inngikk Corvus en avtale med Toyota om tilgang til deres brenselcelleteknologi.[12] De har så utviklet egne pakker som er egnet for skip og deres sikkerhetskrav, der Toyotas moduler er en del av pakken. Planen er at systemet skal være kommersielt tilgjengelig i 2024. For at produktet skal kunne brukes i båter, må de gjennom en rekke tester og godkjenninger, og det er denne fasen pakken er i nå. Corvus har utviklet et eget testområde for dette, hvor de i dag kjører all effekt ut i en lastbank. Corvus har kommet frem til at de vil forsøke å levere kraften ut på strømnettet, som var den opprinnelige oppgavebeskrivelsen. Etter videre oppgaveutredelse og

produktavklaringer med Corvus, har det blitt bestemt, at oppgaven skal fokusere på styring av brenselcellepakken, lastbank og batteripakke, men det skal også se på en videre utvidelse av testanlegget slik at det, i fremtiden, kan levere strøm ut på strømnettet via en omformer og en transformator.

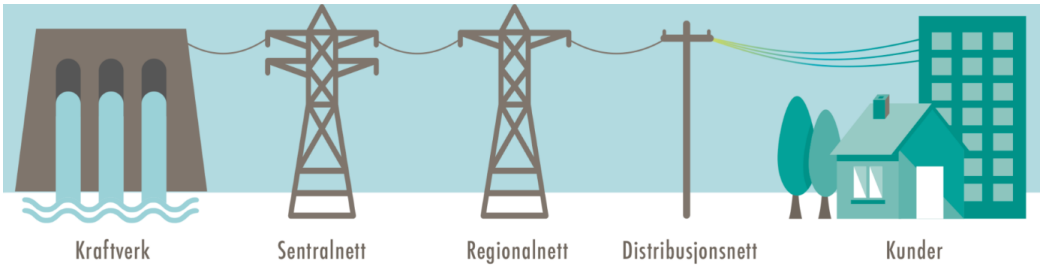
3.2 Tilkobling mot strømnettet

Innledningsvis vil vi nevne litt om hvordan vi tror Corvus er koblet opp mot distribusjonsnettet i dag. Det går en føringsvei mot Corvus sin hovedtavle, hvor inntak fra netteier mest sannsynlig befinner seg. Forsyning til testanlegget kommer fra hovedtavle. AMS-måler (smart strømmåler) mot netteier er i hovedtavle.

Nærmere forklart om strømmålere: De kommer i svært mange forskjellige utførelser, og i dette tilfellet er det mest relevant å ta utgangspunkt i et stort anlegg med høyt forbruk. Her brukes det oftest målere som ikke er direkte tilkoblet, men koblet via en måletransformator. Dette blir gjort for å redusere kostnad, og størrelse, på utstyret. Selve måleren kan da være dimensjonert for mye lavere strøm enn ved når all strøm går direkte gjennom måleren. Det er også forskjell på måleren hos en kunde der kunde bare bruker strøm, og der kunde også leverer strøm (plusskunde). Dette er fordi målekretsene i AMS-målerene vi bruker i dag, bare kan måle i én retning. Alt som går ”feil” vei gjennom en slik måler, vil ikke bli registrert. I tilfeller der kunder også leverer strøm, har målerene dobbelt så mange målekretser for å kunne registrere strømmen som flyter i begge retninger.[13]

Figur 4: Smart strømmåler, ”AMS” måler, hentet fra [14]

Ut fra hva vi har forstått i møter med Corvus, er de ikke plusskunde i dag. All energi som potensielt vil bli levert ut på distribusjonsnettet (se figur 5), vil derfor ”gå tapt”.



Figur 5: Kraftsystemmodellen, hentet fra [15].

Det må også tas inn i beregningene at Corvus bruker en del strøm. Dette betyr at det må produseres ganske mye før det blir levert noe energi ut på distribusjonsnettet, fra Corvus sitt anlegg. Hvis omformer leverer moderate mengder energi inn på Corvus sitt strømnett, er det derfor mulig at brenselcelletesting ikke vil føre til at det blir levert noe strøm ut på distribusjonsnettet. Dette betyr da at alt som blir produsert vil bare gå til eget forbruk, som igjen betyr at det da ikke vil være nødvendig å være plusskunde. I dette tilfellet, bør det også vurderes å måle ”kjøpt” strøm i hovedtavle til Corvus, for å kunne styre omformer slik at det alltid ”kjøpes” litt strøm. Dette vil føre til det ikke blir sendt strøm ut på nettet uten kompensasjon.

3.2.1 Netteier

BKK er netteier i regionen hvor Corvus har sitt testanlegg, og derfor er det BKK som har ansvaret for tilsyn i distribusjonsnettet.[16] BKK har ansvaret for at myndigheters krav til leveringskvalitet blir opprettholdt, og dette er regulert i FOL (Forskrift om leveringskvalitet i kraftsystemet). BKK sitt ansvar er også å sørge for at nærliggende kunder får kraft som opprettholder kravene gitt i FOL. Dette betyr at strøm som omformer leverer ut på strømnettet til Corvus, må levere en gitt, eller bedre, kvalitet, enn det som er definert i FOL.

Figur 6: BKK, netteier i regionen. Hentet fra [17]

3.2.2 Leveringskvalitet

Leveringskvalitet må oppfylle krav ihht. relevante paragrafer i FOL (Leveringskvalitetsforskriften). Dette er delt inn i 3 hoveddeler:

- Spenningens effektivverdi
- Spenningens frekvens
- Spenningens kurveform

Spenningens kurveform er ytterligere definert, med følgende krav gitt av FOL §3-3 til §3-10:

- §3-3 Langsomme variasjoner i spenningens effektivverdi
- §3-4 Kortvarige over- og underspenninger, og spenningssprang
- §3-5 Flimmerintensitet
- §3-6 Usymmetrisk spenning
- §3-7 Overharmoniske spenninger
- §3-8 Interharmoniske spenninger
- §3-9 Signalspenning overlagret forsyningsspenningen
- §3-10 Transiente overspenninger

Kravet til **spenningens effektivverdi**, er at spenningsnivået ikke skal avvike med mer enn 10% fra nominell spenning. For 400 V , vil dette si at spenningen må holdes innenfor 360 V og 440 V, altså 400 V. ±10%[18] Dette er fordi utstyr kan ta skade hvis spenningsnivået avviker mye fra nominell verdi. Spesielt utsatt for spenningsavvik er for eksempel datamaskiner og varme-/kjøleanlegg.[13]

Spenningens frekvens må holdes innenfor 50 Hz $\pm 0,1$ Hz.[18] Norge deler frekvens med Norden, og vi har svært stabil frekvens. Ubalanse i frekvens, blir regulert av nettet.[19] Så lenge omformer har en hvis kvalitet, vil omformer regulere seg inn etter frekvensen på det umiddelbart nærliggende nettet, og levere etter dette.

Spenningens kurveform har litt mer krav knyttet til seg, og er ytterligere definert i FOL. Dette er et svært stort tema og kunne blitt til en hel oppgave i seg selv, så i denne oppgaven kommer vi ikke til å gå så veldig dypt inn i dette. Det eneste vi vil nevne, er dette med støy fra kraftelektronikk. Når vi har en omformer vil det potensielt kunne komme støy som følge av "switching" i kraftelektronikk.[20] Transformator er ønsket for å skape et galvanisk skille mellom testanlegg og Corvus sitt strømnett, og dette kan forsterke støy fra omformer. Ved opptransformering kan støy potensielt bli forsterket, som må tas hensyn til når transformator og omformer blir valgt.[21] Men vi kommer ikke til å gå så veldig mye inn på dette i denne oppgaven.

3.2.3 Konsesjonsplikt

En av de største utfordringene med levering av energi mot distribusjonsnettet, kommer hvis det utløser konsesjonsplikt for anlegget. I dette tilfellet er vi litt usikker på om anlegget leverer nok energi ved full effekt, til å utløse krav om konsesjon. Vi har forsøkt å kontakte NVE (Norges vassdrags- og energidirektorat) for å stille spørsmål om dette, men vi har ikke fått svar. Ut fra hva vi klarer å tolke fra ENL (Energiloven), vil ikke dette anlegget utløse krav om konsesjon ved moderat produksjon som brukes opp internt på Corvus Energy sitt strømnett.

Vi tror at anlegget vil få utfordringer med å overholde kravet om leveringssikkerhet og leveringsplikt, da anlegget bare produserer strøm ved testing. Dette kan skape utfordringer hvis anlegget må ha konsesjon.

Nettsiden til NVE sier at spenninger over 1000 V, utløser konsesjonsplikt for "alle kraftproduksjonsanlegg". I FOL §3-1 er dette videre definert som lavspenningsanlegg, som igjen er definert i FEF (Forskrift for elektriske forsyningsanlegg)

Figur 7: Norges vassdrags- og energidirektorat, NVE, behandler konsesjonssøknader. Hentet fra [22].

som;

> *Med høyspenningsanlegg forstås sterkstrømsanlegg med nominell spenning høyere enn 1000 V vekselstrøm eller høyere enn 1500 V likestrøm.*
> - FEF § 20701

Testanlegget har i dag en øvre nominell spenningsgrense på rundt 750 V DC, som er definert av batteripakken og brenselcellepakken, så nominelt spenningsnivå vil ikke utløse konsesjon for dette anlegget.

Slik vi har forstått ENL, er det behov for å søke om konsesjon hvis anlegget skal levere strøm ut på nettet i vårt tilfelle, da vi ikke kan finne en kategori som direkte eller indirekte gir unntak for konsesjon for brenselceller. Dette betyr at testanlegget kommer inn i kategorien ”annen strømproduksjon”. Her kan vi ikke finne noe spesifikk grense for hvor mye energi som kan leveres uten konsesjon. Den generelle grensen for konsesjon, er når hovedsikringsikring er over 200 A 3-fase og 230 V, gitt av FOL § 3-1. Vi antar at Corvus ikke har 200 A hovedsikring, da dette ville vært for lite for et stort produksjonslokale. Så anlegget vil ikke kunne produsere konsesjonsfritt med grunnlag av dette.

Vi antar derfor at det må søkes om konsesjon, hvis det skal leveres strøm ut på distribusjonsnettet, da anlegget kommer under ”annen produksjon”, og grunnet størrelsen på hovedsikringen. Vi tror også, grunnet anleggets størrelse vil det bli gitt unntak for testanleggets produksjon grunnet relativt lav energimengde. Vi legger da til grunn at energimengde levert av omformeren, er moderate. Vi tror derfor

at det ikke vil være et problem i å få tillatelse til å levere strøm fra omformer på testanlegget ut på distribusjonsnettet, men det må søkes om konsesjon.

3.2.4 Solcellepanel

En mulig vei å komme rundt kravet om konsesjon, hvis det er ønskelig å levere større mengder energi fra testanlegget, er å installere solcellepanel på bygget til Corvus Energy. Vindkraft og solkraft har lov til å levere en god del energi uten konsesjon,[23] gitt det ikke er nødvendig å etablere høyspenningsanlegg for å omsette og levere kraften ut på distribusjonsnettet. Dette kan også gi Corvus en god grunn til å installere solcellepanel. Solcellepaneler vil også bidra positivt til Corvus sin miljøvennlige satsning. Solcellepanelene vil produsere strøm så lenge det er tilstrekkelig med dagslys, og vil indirekte gi brenselcellepakkene en større konsesjonsgrense som resultat av solcellepanelene. Gitt at det er tilstrekkelig dagslys, står solcellene å produserer energi, og all strøm levert fra testanlegg vil da forsvinne i det som blir levert fra solcellepanelene. Utfordringene her er investeringskostnaden, leveringstid på anlegget og at Corvus er leietakere.

3.3 Omformer mot strømnettet

Omformeren mot strømnettet, er den DC/AC-omformeren som skal koble DC-bussen sammen med strømnettet til Corvus. Ettersom rippelspenninger kan degradere katalysatoren til brenselcellene så bør omformeren være av veldig god kvalitet. [24]

3.3.1 Omformer for solceller

Helt i starten av oppgaven så vi en del på solcelleomformere, da vi så på dette som en enkel og god løsning.

Vi så på mange forskjellige modeller, men mest på 2 modeller, grunnet disse var nærmest spesifikasjonene vi hadde behov for;

- Fimer PVS-175-TL[25]
- Growatt MAX 185-253KTL3-X HV [26]

Hvis vi ville valgt å benytte solcelleomformere, hadde dette gjort oppgaven vår enklere på mange måter. Disse er allerede mye brukt i nettet, og de aller fleste al-

ternativene vi har sett på opprettholder kravene gitt i FOL. Solcelleomformerne er billige i innkjøp, og har en veldig enkel styring. De fleste omformerne på markedet vil forsøke å mate ut all tilgjengelig energi på nettet når når spenningen kommer over omformerens "nedre grense". Altså, de aller fleste er styrt med en metode som heter "Maximum PowerPoint Tracking" (MPPT).

Strømmen ut av en enkelt solcelle er avhengig av innstrålingen. Ett foton gir ett elektron (ved 100% strømeffektivitet).[27] Det er spenningstap fra solcellen, og spenning fra solcellen er avhengig av materialvalg. Den vil derfor ikke greie å levere høyere spenning enn det den er konstruert for. På samme måte som en hydrogenbrenselcelle sin maksimale spenning er gitt av den kjemiske reaksjonen mellom hydrogen og oksygen, og strømmen avhenger av reaksjonsraten.

I motsetning til en brenselcelle, så vil ikke solcellen bruke mer drivstoff når man kjører på høyere effekt. Derfor er det ønskelig å alltid maksimere effekten ut av en solcelle, hvor man med en brenselcelle må tenkte på drivstofforbruk. Måten man regulerer effekten ut av en solcellemodul er ved å bruke MPPT.

En MPPT er en DC/DC-omformer som regulerer spenningen ut for å få mest mulig effekt. Om en for eksempel har en motstand på 1 Ω, så vil man maks kunne kjøre 1 A gjennom den hvis man har 1 V over den. Så hvis en solcelle leverer maks 1 V, så vil den ikke greie å levere mer enn 1 A over én 1 Ω motstand. Så eventuelle strømmer over dette vil gå til spille. Ved å sette opp spenningen, vil man kunne levere høyere effekt over den samme motstanden. I tilfellet med solcelleomforme så er motstanden en DC/AC-omformer. Strømmen og spenning til en solcelle vil også variere avhengig av forskjellige forhold, og dermed så vil ikke nødvendigvis en høyest mulig spenning føre til høyest mulig effekt. Dermed kontrolleres DC/DC-omformeren med en tilbakekoblingssløyfe slik at den hele tiden leverer høyest mulig effekt. Det er denne kontrollmekanismen som skiller en MPPT-omformer fra en vanlig DC/DC-omformer.[28]

MPPT egner seg derfor svært godt for solceller hvor tilgjengelig sollys, paneltemperatur og last gjør at energiproduksjon fra solcellene varier. Hvis vi skulle latt FCP-en kjøre ut mest mulig effekt, ville dette sannsynligvis fungert bra, men vi er avhengig av å holde riktig spenning på DC-bussen, noe en MPPT ikke tar høyde for.

Hvis vi benytter MMPT vil dette dermed gi oss flere problemer enn det vil forenkle systemet. Brenselcellepakken bør ha en referansespenning som er et stykke over nedre produksjonsverdi for solcelleomformer. Dette kunne blitt løst med batteribank og et DC/DC-ledd mellom brenselcellepakke/batteripakke og omformer, men det vil gjøre anlegget unødig komplisert, og da er hele vitsen med solcelleomformer borte. Det er også ønskelig fra oppdragsgiver med en omformer som kan, til større grad, reguleres uten å måtte komplisere installasjonen unødvendig. Dermed ble omformere for solcellebruk lagt vekk som løsning.

3.3.2 Omformer for nettstabilisering

AEG sin Convert SC Flex HV ble vurdert.[29] Den har et stort spenn for spenning og er lagd for å stabilisere nettspenningen når man har mye fornybare energikilder i energimiksen. Fordelen med denne løsningen er at den vil sannsynligvis være veldig enkel å få godkjent av BKK. Den kan også leveres sammen med transformator, som Corvus også skal ha for å kunne teste jordfeil. Dermed vil det forenkle prosessen med å sette opp systemet. Den er også toveis og kan dermed lade opp kondensatoren. Det er usikkert om man vil trenge ekstra systemer for å begrense ladestrømmen ettersom denne er laget for mye større systemer og for å lade opp store batteribanker.

- Spenningsområde: 700 – 1400 V
- Strøm: 1450 A
- Effekt: 545 kW ved 1400 V, høyere ved lavere spenninger

3.3.3 Omformer for batteri/brenselcelletesting

Elektro-Automatik sin EA-PSB 11500 4U[30] ble vurdert. Den er lagd for batteri og brenselcelletesting. Den har det største spennet på DC-siden, 0-1500 V og det vil dermed være mulig å teste modulene med et stort spenn i spenninger. Dette er den som sannsynligvis er best og mest fleksibel, den er også toveis og kan brukes til å lade opp kondensatorene til FCP-en. Ettersom den er lagd for labmiljøer har den sannsynligvis mulighet for å måle rippelspenninger. Den er lagd for å bruke i egne skap og ville vært enkel å utvide ved behov. I utgangspunktet ble en 30 kW

versjon vurdert, men i kommunikasjon med forhandler ble en ny 60 kW versjon presentert.[31] Den nye versjonen ville gjort systemet billigere men forlenge leveringstiden litt.

- Spenning: 0-1500 V
- Strøm: 60 A/modul
- Effekt: 30 kW/modul, 60 kW/modul for ny type

3.3.4 Flerbruks omformer

Omformeren VACON NXP Grid Converter[32] fra Danfoss er en som er lagd for et vidt spenn av områder, både til bruk i båt og til bruk i fornybar energi (solceller). Dette er den som er mest relevant for Corvus sitt bruksområde, ettersom denne typen kan bli brukt i båter som bruker brenselceller til å lage elektrisk energi. Spenningsområdet er ikke like stort som de andre, og dermed kan det være en utfordring for å teste ut systemet i lavere spenninger. Denne omformertypen har en rekke versjoner med forskjellig spenninger og bruksområder. Dermed så er dataene til denne hentet fra Danfoss sin MyDrive Select, som gir en valg mellom flere versjoner basert på kravene man setter.

- Versjon: NXI_0750_6
- Spenning: 650-1100 V
- Strøm: 590 A
- Maks effekt: 340 kW
- Kan ha høyere effekt og strøm ved andre konfigurasjoner

3.3.5 Andre omformere

Videre vurderte vi brenselcelleomformere fra Prodrive.[33] men de virket bare i spenningsområdet 100-240 V. Det ble vurdert om det var mulig å sette dem i serie, men vi fikk ikke noe svar på vår forespørsel til produsenten om dette.

Batteritestere fra Gustav-Klein[34] ble vurdert, men spenningen var bare opp til 1000 V og den hadde ingen tekniske fordeler fremfor Elektro-Automatik sin.

Andre omformere ble også vurdert, men disse var for langt unna kravene, eller ville krevd for mye støttesystemer, til at vi brukte mer tid på dem. Vi har derfor ikke tatt disse med i oppgaven.

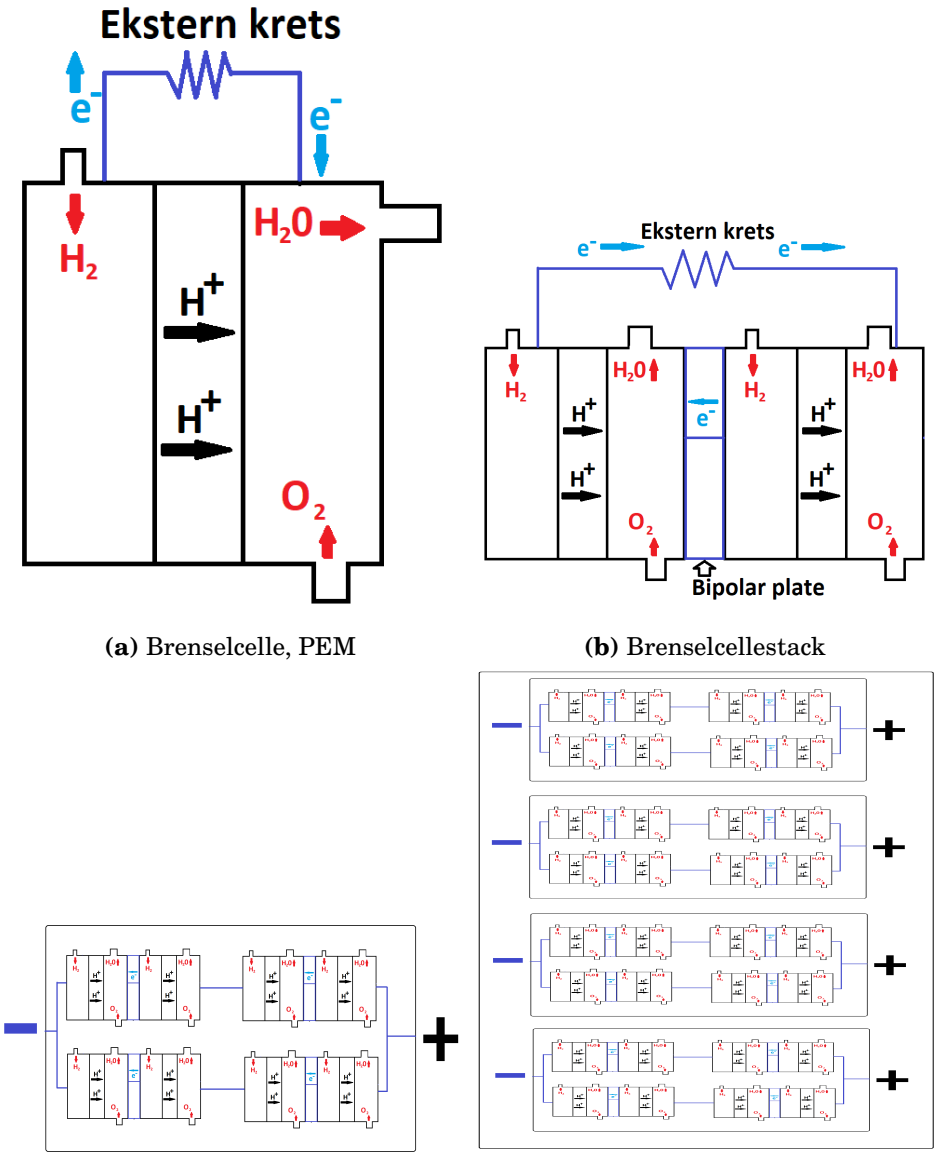
3.4 Brenselcellepakke

Brenselcellene omgjør, i dette tilfellet, hydrogen og oksygen til elektrisitet og vann. Ønsket er å ha en pålitelig og miljøvennlig energikilde, og det er ønsket å ha dette i et enkelt format i form av en skalerbar pakkeløsning. Dette gjør det mye lettere å holde anlegget i drift, og det gjør det mye lettere å erstatte defekte moduler når anlegget får feil og når anlegget nærmer seg slutten av levetiden sin. Kommersielt tilgjengelige og godt utprøvde brenselcellemoduler gjør også pakkene potensielt svært driftsikre og gjør det både billigere og raskere å skaffe reservedeler for pakkene i fremtiden.

3.4.1 Brenselcelleteknologi

En brenselcelle har veldig lik virkemåte som et batteri, men der et batteri har en begrenset mengde energi lagret, så kan en brenselcelle ha kontinuerlig tilgang til drivstoff. Det er flere typer brenselceller som har forskjellige virkemåter og som kan bruke forskjellige drivstoff. Den typen Toyota bruker heter Polymer Electrolyte Membrane/Proton Exchange Membrane (PEM). Som navnene tilsier, så består den av en polymer som fungerer som elektrolytt, denne polymeren gjør at protoner (H^+) lett kan diffundere gjennom membranen, men elektronene må gå rundt i en ekstern elektrisk krets hvor effekt blir hentet ut som vist i figur 8a. Denne brenselcellen fungerer ved relativt lave temperaturer (80°C), men krever dyre katalysatorer som blir dårligere over tid.[35]

Den teoretiske maksspenningen er på 1.23 V for en PEM-celle, men den praktiske spenningen er mye lavere grunnet tap i prosessen. Denne lave spenningen fører til at man som regel har flere celler i en stack, skilt med en bipolar plate for å lede strøm mellom anode og katodesidene. Det er en eller flere brenselcellestacker i Toyota sin brenselcellemodul. Disse modulene har Corvus integrert i sine brenselcellepakker. I figur 8 er det oversikt over de forskjellige oppsettene av brenselceller.



(a) Brenselcelle, PEM

(b) Brenselcellestack

(c) Brenselcellemodul

(d) Brenselcellepakke (FCP)

Figur 8: Forskjell mellom celle, stack, modul og pakke

3.4.2 Corvus sin brenselcellepakke

Brenselcellepakken (Fuel Cell Pack - FCP) er enkelt forklart et kabinett med 4 brenselcellemoduler fra Toyota. Kabinettet fungerer både som en sikkerhetsmekanisme og forenkling ved at den har en intern PLS som styrer alle fire modulene og som håndterer sikkerhetsmekanismer.

Selve modulene leverer en variabel spenning. Spenningen settes ved å lade opp en kondensator som er koblet til DC-bussen. Deretter vil brenselcellene levere en spenning tilsvarende dette. Det eneste som kontrolleres er effekten. Antageligvis bruker modulene en såkalt ”step-up regulator” for å regulere spenningen. Denne antagelsen er basert på litteratur[36], at det er en kondensator som skal lades opp og fordi en enkelt brenselcelle har en teoretisk maksspenning på 1.23 V, og en dermed måtte ha veldig mange brenselceller i serie for å levere spenninger over dette området.

Modulene har følgende karakteristikker[37]:

- Spenning 400-750 V
- Min. effekt: 10 kW
- Maks. effekt: 80 kW

Brenselcellepakken har kommunikasjon over Modbus som skal brukes til å styre den. Dette er enda under utvikling.

3.5 Batteripakke

Batteripakken som blir brukt i testanlegget er fra Corvus sin egen Orca-serie. Den er modulær ved man kan ha flere batteripakker i et såkalt array. Og hver batteripakke kan ha opp til 22 moduler, avhengig av ønsket spenning og energi. Batterisystemet som er installert på testanlegget består av 16 moduler i en pakke. Basert på produktinformasjon på nettsiden så vil det gi[38]:

Figur 9: Batteripakken Orca. Bilde fra [38].

- Energi: 89.6 kWh
- Maks kontinuerlig effekt: 270 kW (3C)
- Maks spenning: 800 V
- Nominell spenning: 713 V
- Minimum spenning: 582 V

Batteripakken kan ta imot store mengder effekt. Nesten nok til å ta i mot alt om FCP kjører på full effekt. Den oppgitte effekten er for kontinuerlig effekt, dermed så vil batteripakken kunne ta imot all effekt fra FCP i korte perioder. Dermed er

ikke batteripakken en begrensning i systemet. Dette gjør arbeidet lettere ved at man ikke trenger å trinne opp effekten fra FCP forsiktig for å spare på batteriet.

Batteripakken har også en forladningsenhet installert, denne vil begrense ladestrømmen til kondensatoren til FCP-en når den kobler til.

Batteripakkene har kommunikasjon over Modbus og CANbus.

3.6 Lastbank

Lastbanken sin oppgave er å ta imot effekten fra FCP-en som omformeren ikke kan ta imot. Lastbanken er bygd opp med motstandselementer, reléer til å legge elementene inn/ut i forskjellige konfigurasjoner, en PLS til å styre reléene, og en PC. De forskjellige konfigurasjonene gir forskjellige motstandsverdier, som igjen brenner av forskjellige mengder energi. Lastbanken er produsert i Kina, og har en dårlig oversatt brukermanual på kinesisk og engelsk, som er en utfordring for brukerne. Ved designing av lastbanken, ble det i tillegg noen misforståelser, og dette har skapt noen utfordringer for Corvus.

Figur 10: Lastbanken i ny tilstand.

Hovedessensen på hvordan denne virker, er; Lastbanken har mange forskjellige motstandselementer som blir lagt inn og ut i forskjellige konfigurasjoner, ved hjelp av reléer. Når lastbanken får beskjed om å sette på last, vil den da sende energien ut i den konfigurasjonen (trinn) av motstandselementer den har fått beskjed

om, og omgjør denne effekten til varme. Lastbanken kan sees på som en gigantisk varmeovn.

Noen av motstandselementene ble designet for 400 V. Ved innledende testing ble disse brukt på for høye spenninger. Resultatet av dette, er derfor at noen av motstandselementer er blitt defekt, og det har ført til at det er færre trinn å velge mellom. Rent praktisk sett, vil ikke dette ha noe å si for vår styring, da vi bare har litt færre trinn å velge mellom.

Lastbanken kan brenne av i spennet mellom 1 kW, og opp til den fulle effekten produsert av brenselcellene. Vi har rundt 30 av 400 trinn å velge mellom, og forholdet mellom avbrent effekt, og trinn er ikke lineært. Avbrent effekt er utregnet ved hjelp av konduktans (S), og spenning (V) inn på lastbank. Altså, ved 8 kW og 750 V vil konduktans for dette motstandselementet være;

$$P = V^2G => G = \frac{P}{V^2} = \frac{80000}{750^2} = \frac{80000}{562500} = 142 \text{ mS} \tag{1}$$

Hvis den samme motstanden brukes ved 700 V, så vil det føre til avbrent effekt på:

$$P = V^2G = 700^2 * 0.142 = 69.58 \text{ kW} \tag{2}$$

Altså avgitt effekt fra lastbank vil være avhengig av spenningen inn på lastbank. En annen utfordring er at konduktansverdiene, er en teoretisk utregning fra dokumentasjon til lastbank. Det kan derfor være avvik på disse verdiene, slik at effekt den brenner av kan være høyere eller lavere. For å få disse verdiene inn i programmet, valgte vi å basere oss på en spenning midt i testområdet på testanlegget. Utregninger og grafer som forklarer dette visuelt, ligger i vedlegg D.

Vi må også tenke på at lastbanken bruker reléer for å velge trinn. Reléer er designet for å veksle et visst antall ganger, og når de har hatt så mange av/på-sykluser, så er komponenten oppbrukt og må byttes ut. Vi må derfor huske å implementere i koden at den ikke bytter trinn alt for ofte, men at den venter tilstrekkelig lenge mellom hver gang den bytter trinn.

3.7 Regulering

I et vanlig kraftsystem for skip, vil man regulere effekten ut av FCP for å holde riktig spenningsnivå på batteripakker og gi effekt til motorer og støttesystemer. I våres tilfelle så må vi regulere den avbrente effekten basert på den produserte effekten og spenningsnivået på batteripakken. Vi vil unngå slitasje på batteripakken og reléene til lastbanken, så vi vil ha en regulering som ikke forandres ofte. Ettersom effekt ut av FCP reguleres utifra testbehov, så vil vi bare regulere lastbanken.

Ettersom lastbanken må reguleres i diskrete steg som fører til en usikker lastproduksjon så må vi ha en regulering som tar dette i betraktning. Vi har valgt å gå for en regulering hvor man enten er i en oppladningssløyfe eller i en utladningssløyfe. Målet er ikke å brenne av samme effekt som blir produsert, men at man brenner av mer effekt enn produsert hvis man er i en oppladningssløyfe, og motsatt hvis man er i en utladningssløyfe.

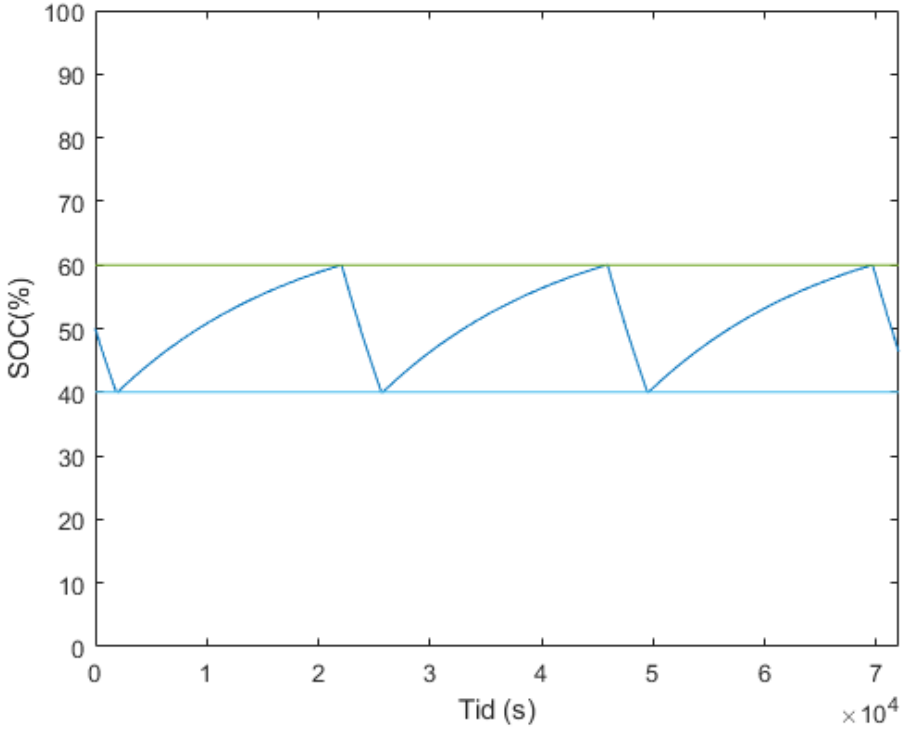
Dermed så vil SOC holde seg mellom settverdiene, og man vil skifte mellom en utladningskurve og oppladningskurve. I figur 11 kan man se hvordan dette utarter seg ved en tenkt effekt på 150 kW. Simulasjonen bruker ikke realistiske tall på batteripakken ettersom spenning ved forskjellige SOC er klassifisert som hemmelig. En lineær sammenheng mellom SOC og spenning er en OK antagelse, men gir litt feil. Se vedlegg A for kode.

3.8 Kommunikasjon over Modbus TCP

Modbus er en åpen protokoll som er veldig populær selv om den er fra 1979. Vi bruker TCP-versjonen som bruker TCP/IP til kommunikasjon mellom tjener og klient. Modbus består av 4 typer registre:[39]

- Coil - 1 bit, som klienten kan lese og skrives til
- Discrete input - 1 bit, som klienten bare kan lese fra
- Input register - 16 bit, som klienten bare kan lese fra
- Holding register - 16 bit, som klienten kan lese og skrives til

Av disse, så bruker Corvus bare ”Holding Register” i batteripakken og FCP. Dermed så trenger vi bare å lese til og fra disse typene registre. Modbus har flere funksjonskoder som beskriver hvilke registre den bruker og om den leser eller



Figur 11: Simulering av opplading og utlading ved 150kW, grensene for opp/utladning er på henholdsvis 40 og 60%.

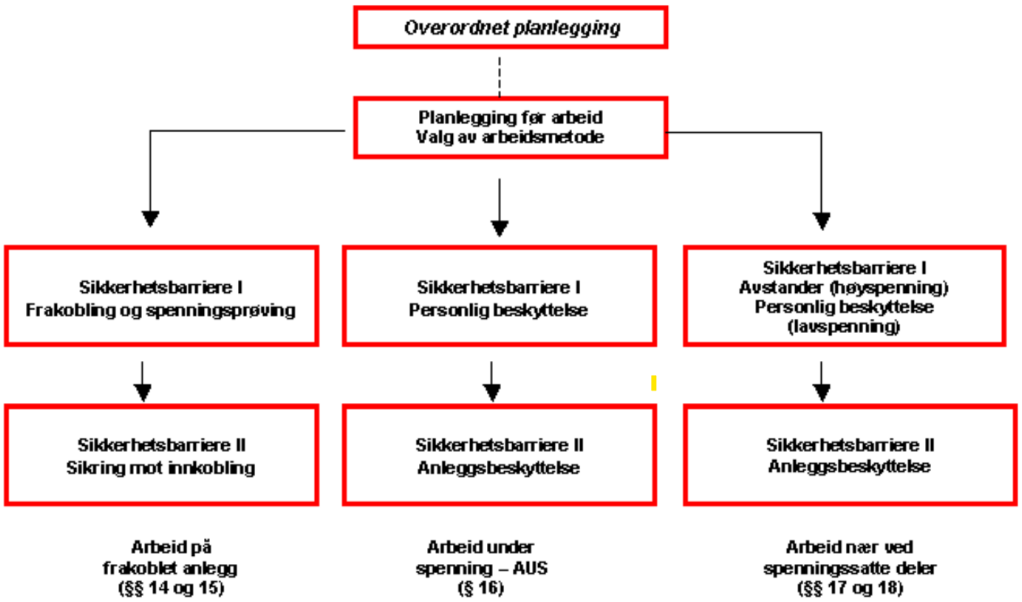
skriver til registre. Vi trenger bare funksjonskode 3 (FC3) for å lese av registre og funksjonskode 16 (FC16) for å skrive til registre. PLCnext har sitt eget Modbus-bibliotek som vi vil bruke. FC3 leser av opptil 125 registre etter hverandre. FC16 skriver til opptil 125 registre i rekkefølge. Det er oppgitt i dokumentasjonen at å lese av og skrive til flere registre går like raskt som med et register. Dermed så bør man tilstrebe å lese av og skrive til så mange registre som mulig. [40]

Hvert holding register inneholder 16 bit med informasjon, det tilsvarer 1 word, så dette må konverteres på riktig måte til heltall eller boolske variabler. Ved å bruke egne funksjoner for det, så vil man få en mer oversiktlig kode.

3.9 Risikovurdering

Arbeidet her er forbundet med to typer risiko, vi jobber med et anlegg som har høye spenninger og som håndterer hydrogen.

3.9.1 Risiko ved elektriske anlegg



Figur 12: Flytskjema for arbeidsmetoder i FSE, hentet fra [41].

For at sikkerheten blir opprettholdt, må FSE (Forskrift for sikkerhet i elektriske anlegg) følges. Hovedessensen her, er at det alltid skal være 2 barrierer til stede ved arbeid i elektriske anlegg. I dette tilfellet er arbeid på frakoblet anlegg som blir mest aktuelt, altså FSE §14-15.

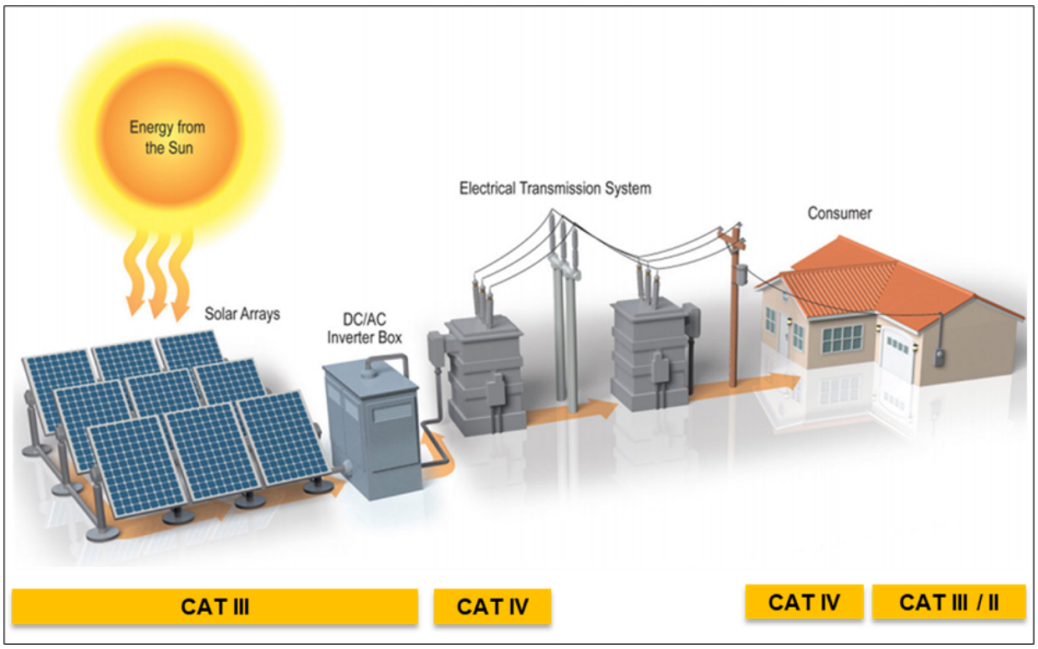
> Ved arbeid på frakoblet anlegg skal følgende sikkerhetstiltak gjennomføres:
>
> a. frakobling
>
> b. sikring mot innkobling
>
> c. kontroll av at anlegget er spenningsløst

d. på bakgrunn av en risikovurdering vurdere behov for og eventuelt etablere nødvendig jord- og kortslutning, og

e. eventuelt beskyttelse mot andre spenningssatte deler nær ved arbeidsstedet (jf. § 17).

- Fra FSE §14

På bakgrunn av dette, har vi lagt inn i vår kode, at lastbanken skal lade ut DC-bussen i nedstengingssekvensen på programmet, og lastbanken er den siste komponenten som kobles fra under nedstenging av programmet. Når FCP ikke går, er batteriet den eneste spenningskilden i anlegget, og DC-bussen blir da ansett som spenningsløs når denne, og FCP, er fysisk adskilt fra DC-bussen. Det er derfor viktig å fysisk koble batteriet fra DC-bussen, og Corvus har allerede gode prosedyrer for dette. Enda spenningen kommer opp i HMI, er det viktig å fysisk måle med egnet måleapparat (minimum CAT III-instrument) at DC-bussen er spenningsløs, da vi mener at måleverdiene i HMI ikke er tilfredstillende pålitelig spenningsindikator grunnet for mange feilkilder.



Figur 13: Kategorier for måleinstrumenter, hentet fra [42].

Når det gjelder styresystemene har disse også egne sikringer. Disse må også

frakobles, sikres mot innkobling og kontrolleres spenningsløse, under arbeid på anlegget. Å koble en ledning fra et tilkoblingspunkt, er en sikring mot innkobling, men sikringen oppstrøms må da sikres mot innkobling (låses) når denne blir frakoblet.

Arbeid under spenning, AUS (§16), anser vi som en unødvendig løsning i dette tilfellet. Det krever mye spesialutstyr og spesialkompetanse for å bli løst på tilfredstillende sikker måte i dette tilfellet.

Arbeid nær ved (§17), anser vi også som lite aktuell i dette anlegget, da arbeid på anlegget skjer når anlegget ikke er under testing. Det er derfor like greit å etablere sikkerhetstiltak for arbeid på spenningsløst anlegg.[41]

3.9.2 Risiko ved hydrogen

Hydrogen er som nevnt både brennbart og eksplosivt, og vi skal styre tilførsel av hydrogen til brenselcellene. Dermed må vi ha kontroll på at det vi gjør ikke utsetter mennesker eller utstyr for fare. På testanlegget har Corvus hydrogensensorer, kaldfakkel og et gasshåndteringssystem som er koblet direkte til brannvesenet. Selve gasshåndteringsanlegget styres av Corvus sin testingeniør uavhengig av FCP, lastbank og batteripakke. Testingeniøren følger egne sikkerhetsprosedyrer før testing. Dermed har vi ikke direkte innvirkning på hydrogenforsyningen. Vi har indirekte innvirkning ved at vi gir et pådrag til FCP. FCP-en har sitt eget sikkerhetssystem som kontroller atmosfæren i FCP og har sine egne forriglinger. Dermed vil FCP og gasshåndteringsanlegget håndtere all risiko forbundet med hydrogen på avveie.

3.10 Styringssystem

Phoenix Contacts sitt PLCnext-system ble valgt ettersom Corvus hadde et startsett med AXC F 2152-kontrolleren liggende og leveringstiden på PLS-er er veldig lang for tiden. Det har den fordelen at selve programmet som man programmerer i, PLCnext Engineer, er gratis, så dermed ville vi ikke trenge egne lisenser.

Styringssystemet skal kontrollere en lastbank, batteripakke og brenselcellepakke. Alle disse har sitt eget styresystem med innebygde sikkerhetsfunksjoner. Dermed vil vi ikke trenge å overvåke parametre som f.eks. temperatur, ettersom de har egne sikkerhetsfunksjoner og håndterer dette selv. Det forenkler sikkerhetsarbeidet

betraktelig. Men vi må ved f.eks. utfall av effekt fra FCP kutte ut lastbanken slik at den ikke tapper batteriet unødvendig. Så våres sikkerhetsfokus er i grensesnittet mellom de forskjellige modulene og innvirkninger de vil ha på hverandre.

Ettersom systemet leveres som en del av en bacheloroppgave, og vi ikke kommer til å vedlikeholde systemet etter innlevering er det veldig viktig at systemet er oversiktlig og godt dokumentert. Hvis ikke risikerer Corvus å måtte bruke veldig lang tid på feilsøking hvis det skulle dukke opp feil i programmet eller hvis de skal utvide det. Dokumentasjonen vil dermed være en viktig del av oppgaven. Dokumentasjonen burde være på engelsk ettersom Corvus har flere ansatte fra andre land, og systemet vil dermed bli lettere tilgjengelig for de.

Kommunikasjonen med batteripakken og FCP må være over Modbus, ettersom dette er spesifisert som krav fra Corvus. Kommunikasjon med lastbanken må gjøres over TCP/IP, ettersom den kun har støtte for det.

3.10.1 To typer styring

Vi har behov to forskjellige styringssystem. Et hvor man har batteri, lastbank og FCP koblet sammen. Det andre har i tillegg en omformer koblet til. For at det skal bli en enkel overgang til systemet med omformer bør vi passe på at styringen av batteri, FCP og lastbank gjøres på en måte som er likt for begge systemer.

3.10.2 Hovedprogram

Hovedprogrammet sin rolle er å ha det overordnete ansvaret for å gi riktig signal til styringen for FCP, batteri og lastbank. Den skal starte opp modulene i riktig rekkefølge og avslutte testingen i riktig rekkefølge. Den vil også regulere effekten ut fra lastbank basert på SOC og produsert effekt fra FCP.

3.10.3 Styring av batteri

Batteriet sin oppgave er å lade opp kondensatoren til FCP-en slik at den holder riktig spenning når man starter opp FCP-en. Den vil også bidra til å holde en stabil spenning på DC-bussen ettersom lastbanken ikke kan finreguleres.

Styringen av batteriet er veldig enkel. Den eneste styringsfunksjonen man har på den er å koble den av og på DC-bussen. Batteriet må passe på å lade opp DC-bussen sakte nok, dette skjer via forladingsenheten som er integrert i batteriet. Den passer på å begrense strømmen ut av batteriet for at kondensatoren og batteriet ikke skal ta skade av for høye strømmer. Virkemåten til den er integrert i batteripakken og vil automatisk måle spenninger og passe på å begrense strømmen ut når batteriet kobler seg til DC-bussen.

Utover styringssignalet inn til batteriet, har styringsprogrammet behov for å vite ladestatus (SOC) for å vite om man skal lade opp eller ut batteriet. Man har behov for å vite strømmen inn/ut av batteriet for å regulere effekten til lastbank og omformer. Man har også behov for å vite om noe er feil ved batteriet for å kunne avslutte forsøket. Man har også behov for å vite om batteriet er koblet til, slik at man kan starte FCP.

Signaler som det er behov for i HMI er alarmer og feil i tillegg til det som er behov i fra styringssystemet. Alarmer er for å varsle testoperatør om at noe er på gang slik at de kan handle ved behov.

Internt så må batteristyringen håndtere de fleste feil som kan oppstå, mest i forbindelse med kommunikasjonen mot batteri.

3.10.4 Styring av FCP

Brenselcellepakken sin oppgave er å levere ønsket effekt fra de modulene operatøren vil kjøre. Denne skal kun reguleres som en del av testingen, dermed så vil den ha behov for en oppstartsprosedyre for å se til at den starter opp skikkelig, en avstengningsprosedyre slik at den blir stengt av skikkelig.

Oppstartsprosedyren må se til at riktige module kobler seg til DC-bussen før batteriet kobler seg til og de ikke starter å levere effekt før batteriet er ferdig tilkoblet. Videre må den gi signal til hovedprogrammet at FCP er klare for at batteri kan kobles til.

Under testing så må styringen til brenselcellepakken videreformidle effektverdier til FCP og se til at kommunikasjonen er OK, den må også lese av verdier fra brenselcellepakken.

Etter testing så må den håndtere nedstengning og signalisere til hovedprogram-

met når den er utkoblet.

3.10.5 Automatisk styring

For å gjøre jobben for operatøren lettere burde vi implementere et automatisk styringssystem som forandrer effektverdiene til FCM-ene etter hvert som tiden går. Dette kan bruke et array, hvor tiden og effektene til de forskjellige modulene er gitt. Vi legger opp til at arrayet går fra 0 til 255. Dette er flere plasser enn det som behøves, men ved å gjøre det slik, så kan man bruke en USINT som index i arrayet, dermed vil man unngå at man prøver å lese eller skrive til et sted i array som ikke eksisterer. Dermed vil man unngå at PLS-en stopper under kjøring.

3.10.6 Styring av lastbank

Lastbanken blir styrt av PLS internt i lastbank, som er satt opp som tjener. Denne kommuniserer over Ethernet TCP, og denne må vi etablere en toveis-kommunikasjon mot med vår PLS, over samme protokoll som Lastbank sender data på. Lastbank sender 19 byte med data, som inneholder diverse variabler. Denne datastrengen inneholder spenning inn på lastbank, strøm inn, effekt inn, trinnverdi, alarmer og hvorvidt den er tilkoblet DC-bussen. For å snakke til lastbanken må det sendes en streng på 3 byte som inneholder ønsket trinn, og om lastbank skal kobles til DC-bussen.

Etter å ha snakket med Corvus Energy om simulering av denne komponenten, og etter vi testet ut forskjellig programvare for TCP og UDP debugging, fant vi ut av at det ikke er noen enkel måte å simulere denne komponenten på. Vi fant derfor ut av at det enkleste er å teste dette direkte opp mot lastbank på testanlegg. Utforming, testing og diskusjon av denne delen av programmet, er derfor knyttet svært tett opp mot kapittelet som omhandler testing. Vi var også veldig heldig med programmeringen av dette, og fikk kommunikasjon på første forsøk. For sending av data, og for mottatte verdier fra lastbank, hadde vi litt mer utfordringer. Sending av data fikk vi til på 2. forsøk, og variablene ble konvertert på rett måte på 2. forsøk. Se kapittel om realisering for utdyping av dette. Litt blir også videre diskutert her, da problemanalyse, testing og diskusjon er veldig sammenflettet når det gjelder lastbanken.

For å få opprette kommunikasjon med lastbank, må det åpnes en tunell mellom PLS i Lastbank (tjener), og vår PLS (klient). Mottatt data må også bli håndtert på en slik måte at ingenting flytter på seg, og slik at ingenting forsvinner. Vi må også knytte inndata opp mot rett variabel i vårt program. Dette gjorde vi ved å hente ut data fra mottatt array, og plassere det inn i UINT/INT-variabler som videre brukes i programmet.

For etablering av kommunikasjon var det litt leting med å finne ut av hvordan vi skulle løse dette. Vi fant ut av at PLCnext har egne ferdige blokker for kommunikasjon over TCP/TLS. Disse blokkene er delt inn i 3 blokker. En sokkel (TLS Socket2), en blokk for sending (TLS Send2), og en for å motta data (TLS Recieve2). I begynnelsen hadde vi disse inni if-setninger, men vi tok etterhvert sokkel og sendeblokk ut fra disse, da vi så at det var noen utfordringer med nedstenging av kommunikasjon og ved sending av data. Blokkene har også en del variabler som må settes riktig, men med en gang disse var riktig satt opp, etter bruksanvisningen til PLCnext, virket programmet. Disse blokkene kan håndtere kommunikasjon over både TLS og TCP. Siden vi skal snakke over TCP, skal den ene inngangen på sokkelblokken være satt til ”False”.

Største utfordring vi hadde her, var egentlig valg av variabel for mottatt og sendt data. Vi vurderte forskjellige datatyper som streng, UDINT, UINT, med mer. Problemet vi så, var at det kan gi oss utfordring med plassering av data i variablene. Siden det blir mottatt som 1 variabel, var vi redd for at data kunne bli lagt inn i feil variabel, eller at det ville bli unødvendig komplisert å klare å tolke mottatt data. En annen utfordring er også at vi måtte legge sammen trinnverdi og av/på i en streng som skal sendes til lastbank. Etter mye om og men, valgte vi å bruke byte-array med 19 plasser á 1 byte for mottak av data, og 3 plasser á 1 byte for sending av data. Under første test fungerte mottak av data på denne måten. Vi så derfor på dette som den beste løsningen, og videreutviklet programmet rundt dette. For mer om dette, se kapittel for testing.

Lastbanken startes ved at programmet vårt sender en 3 byte streng med data til Lastbanken sin PLS. Denne strengen består av hvilket trinn den skal stå i, og om den skal være aktiv eller ikke. Plassering av byte bestemmer hvilke variabler den

hører til. Når lastbank får beskjed om å brenne av effekt, starter den også viften som kjøler ned elementene i lastbanken.

For valg av trinn, kunne vi valgt å regne ut avbrent effekt ut fra konduktansverdiene vi har, i programmet vårt. Dette ville gitt en mer nøyaktig verdi som også tar hensyn til spenningen inn på lastbanken. Vi diskuterte dette litt, men fant ut av at dette ikke var nødvendig da vi ikke har så veldig stor spenningsvariasjon under testing. Variasjon i avbrent effekt som følge av varierende spenning, vil derfor være veldig liten i dette tilfellet. Dette er et mulig forbedringspotensial i koden slik den er nå. For å finne avgitt effekt i hvert trinn, lagde vi et regneark som regnet dette ut med formelen;

$$P = V^2 G \tag{3}$$

og regnet ut effekten for alle de kjente ledningsevnene, for å finne ut av effekten i hvert trinn. Dette regnearket ligger vedlagt i vedlegg D.

Vi valgte å bruke en fast spenningsverdi som ligger ca. midt i spenningens øvre og nedre verdi i testområdet. Dette gir oss en endring i spenning på ca. 10 volt, som er så lite at avviket ikke vil bety noe. Vi tenker da på stabilitet, regulering osv. Vi valgte også denne løsningen fordi det gir en mer oversiktlig kode, som er lettere å lese og lettere å modifisere senere hvis det viser seg at konduktansverdiene i dokumentasjon avviker så mye at det må gjøres endringer i hvor mye effekt hvert enkelt trinn brenner av. Slik kode er skrevet, er det enkelt å endre disse verdiene, da alle ligger oversiktlig i variabellisten. Hvis en verdi må endres er det bare å endre direkte på variabelen i programmet. Disse effektverdiene ble grunnlaget for reguleringsdelen i programmeringen.

For å velge trinn, vurderte vi å bruke Case-struktur. Case gir veldig kompakt kode, som ville gjort det veldig lett å lese koden senere. Vi brukte Case for manuell kjøring av lastbank. For automatisk kjøring, fant vi ut av at det letteste der og da, var IF-setninger. Den må sammenligne energi som blir brent av i hvert trinn, mot verdien som blir etterspurt, og vi valgte å skrive det som if-setninger, der vi begynner på ”toppen”. Hvis verdien er høyere enn trinnets verdi, velger den det trinnet den sjekker mot. Hvis ikke går den til neste trinn og sjekker. osv. Disse setningene velger altså hvilket trinn vi skal ligge på, ut fra de trinnene vi har tilgjengelig. Vi valgte å skrive dette som 2 separate strukturer, der ene er for oppladning og andre

for utladning. Hvis det ønskes at batteriet skal lades ut, velger den trinnet som er ett trinn over verdien den sammenligner mot, for å få strøm ut av batteriet. Hvis det ønskes at batteriet skal lades, velger den det trinnet som er ett trinn under verdien den sammenligner mot. Vi vurderte å ha én if-setning som sammenligner verdiene, så ha en if-setning inni som bestemmer om den skal velge trinnet over eller under, men vi syntes at koden var lettere å lese med 2 separate sammenligninger. Etter å videre ha tenkt på denne delen av koden, har vi også tenkt på at et mulig forbedringspotensial til koden er å skrive dette som for-løkker, og det er noe vi ville prøvd på neste versjon av koden.

Vi måtte også tenke på at lastbanken bruker reléer for å velge trinn. Reléer er designet for å veksle et visst antall ganger, så vi må bytte trinn så lite som mulig. Dette kan gjøres på forskjellige måter. Det kan brukes en timer for å vente en bestemt tid før det kan sendes data igjen. Vi kan bruke en teller, for å kun muliggjøre sending av data når programmet har telt opp til en hvis verdi. Det er også mulig å pause programmet frem til en viss tid er gått.

Å pause programmet er veldig dårlig praksis. Dette er fordi at da venter ”alt” på denne, og vi valgte å ikke løse det på denne måten.

Ved bruk av timer, fant vi ut av at det låser oss litt, da det ikke er like enkelt å manipulere tiden for å sende en endring til lastbank, for eksempel ved at en variabel er sendt og ikke registrert av lastbank, eller hvis vi ønsker å sende en endring umiddelbart.

Vi valgte å bruke teller i vårt program. Hver syklus i programmet øker verdien med 1, og det gir da en tid mellom hver endring av trinn, som er gitt av syklustiden til vår PLS. Altså, hvis syklustiden er 20Hz, og variabelen er satt til at den kan sende data hver gang verdien er nådd 20, vil data kunne sendes 1 gang hvert sekund. Vi vurderer en tid mellom 2 og 5 minutter, da dette vil gi testen mer stabile forhold men også ta hensyn til reléene i lastbanken. Etter data blir sendt, må da denne verdien settes til 0 igjen slik at den ikke sender neste syklus. Denne løsningen gjør det også enkelt å legge inn kontroll-setninger som sjekker om lastbanken har registrert mottatt data og gjort endring, og hvis lastbank ikke har gjort endring kan data sendes igjen ved å resette teller. Om det også er behov for umiddelbar endring av trinn, er det også mulig å gjøre dette ved å bare øke verdien på telleren slik at

data kan sendes tidligere.

Lastbanken har også 2 moduser. For at lastbank skal fungere slik som tiltenkt, og skal brenne av effekt, må den stå i fjernstyrt modus. Lastbanken sender ikke data, og den kan ikke styres fra vår PLS når den står i manuell modus.

3.10.7 Annet

Andre kritiske komponenter som kjøling og gasshåndteringssystemet vil ikke være styrt av PLS, men er noe som kjøres uavhengig av lasten.

3.11 Operatørkontroll

Vi vil implementere to måter å kontrollere PLS-en, den ene er via et HMI. Den andre er via en Modbus TCP-tjener. At det skal kontrolleres via Modbus TCP er et krav fra Corvus sin side. HMI var i utgangspunktet ikke et krav, men gjør testing mye lettere for oss, og har dermed blitt lagt til.

Begge grensesnittene skal være mulig å bruke, men vi må passe på at bare et kan være i bruk om gangen til å styre systemet, dette for å forhindre uønskede situasjoner.

3.11.1 Brukergrensesnitt

Brukergrensesnittet bør være intuitivt og oversiktlig for operatøren å bruke. Ettersom Corvus er et internasjonalt selskap bør brukergrensesnittet være tilgjengelig på engelsk. Det kunne vært mulig å ha det tilgjengelig på norsk også, men det krever mer arbeid som går utover andre og viktigere funksjoner.

Brukergrensesnittet må kunne håndtere alle situasjoner som kan oppstå under normal drift og gi muligheter for feilsøking. Dermed bør en ha mulighet for å se hvor man er i oppkoblings og nedstengningsprosessen.

Brukergrensesnittet må gi operatøren lett oversikt over feilmeldinger fra de forskjellige komponentene, og mulighet for å tilbakestille alarmer. En alarmlogg som gjør at operatøren kan ha oversikt over nylige hendelser vil være nyttig.

3.11.2 Modbus TCP-tjener

Ettersom Corvus kun bruker holding registre så vil vi også gjøre det slik at det skal være lett å sette seg inn i for de. En utfordring er at man kan lese og skrive til registre. Så hvis vi f.eks. knytter en verdi som skal leses av direkte til variabelen i programmet, så kan man ved et uhell forandre på verdier som man ikke skal forandre på. Dette sikres veldig enkelt med at man ikke knytter verdiene fra PLS til registrene til Modbus-tjeneren. Men oppdaterer registrene med verdiene jevnlig.

Ettersom vi allerede har et HMI som styrer alt, så vil vi knytte de registrene som styrer PLS-en til de samme kommandoene. F.eks. hvis register 1 tilbakestiller alarmer for batteriet, så vil dette registeret leses av, og skrive resultatet til samme variabelen som knappen i HMI som tilbakestiller alarmer, er knyttet til.

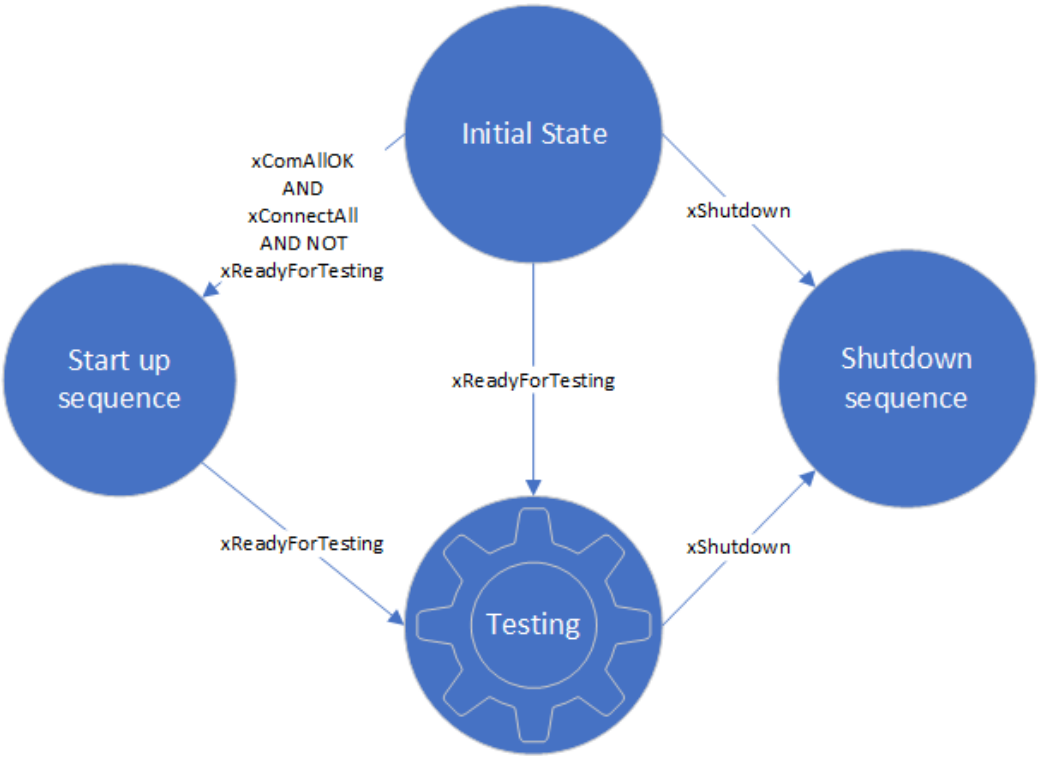
For å gjøre ting lettere for de som skal ordne kommunikasjon mot Modbus-server, så vil vi ha alle registre som leses av på rad. Slik at man kun trenger en avlesning. Vi vil gjøre det samme for kommandoer, men også gjøre slik at man må bruke 2 bits for å utføre en kommando. En bit sier hva som skal gjøres (0 - av, 1 - på), den andre sier om det skal utføres (0 utføres ikke, 1 utføres). Dette gjør at man hvis man sender 0 til registeret som en standardverdi, så vil ingen ting utføres. Dermed kan man utføre flere skrivekommandoer på rad trygt, uten å passe på at man skriver riktig verdi til alle registre hele tiden.

4 Realisering

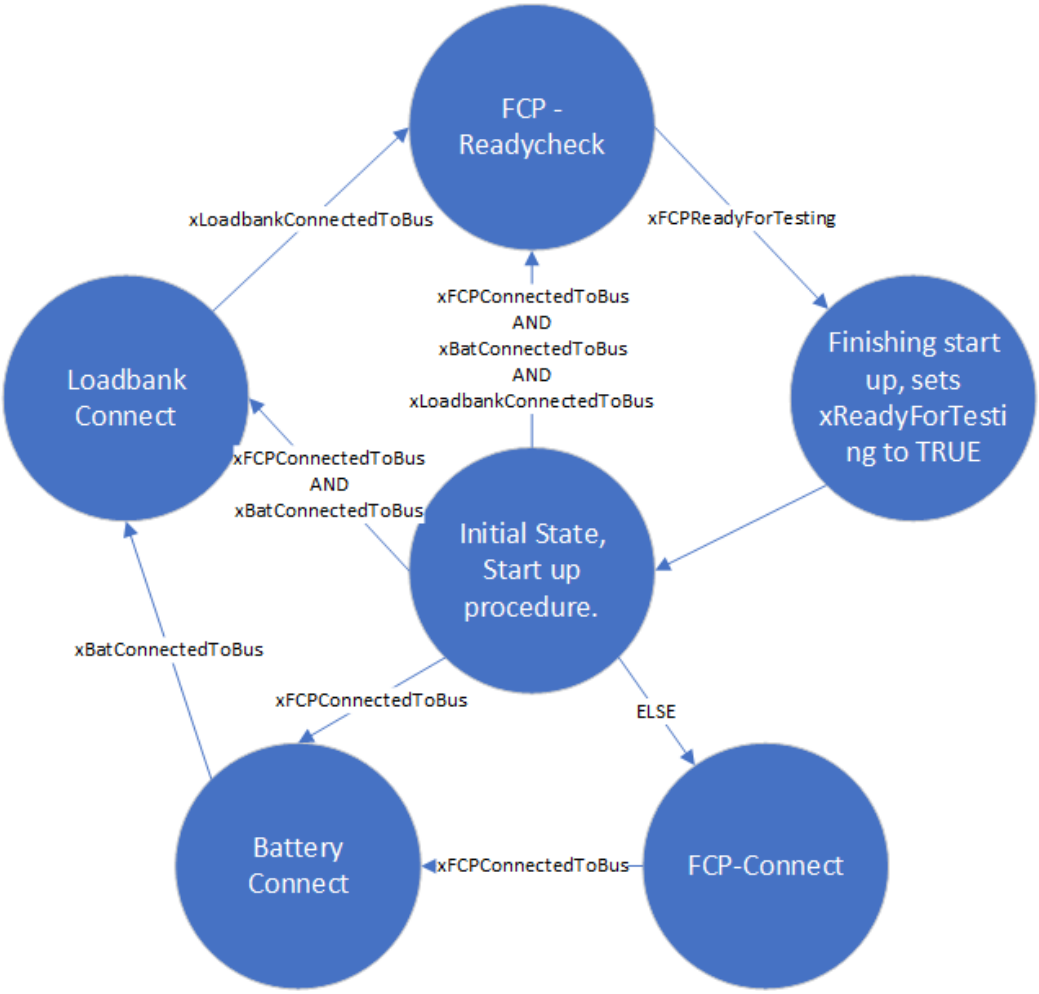
4.1 Realisering av styresystem

Hovedstyresystemet ble laget i en programblokk, pMain, som brukes til å styre underblokker, fbFCP som styrer brenselcellepakke, fbLoadbank som styrer lastbank, fbBattery som styrer batteri. I tillegg er det flere funksjoner og funksjonsblokker som håndterer datakonvertering og kommunikasjon. Det ble også laget en egen funksjonsblokk som simulerte lastbank på en veldig enkel måte, denne kunne vi skrive verdier til når man koblet seg til PLS via PLCnext Engineer.

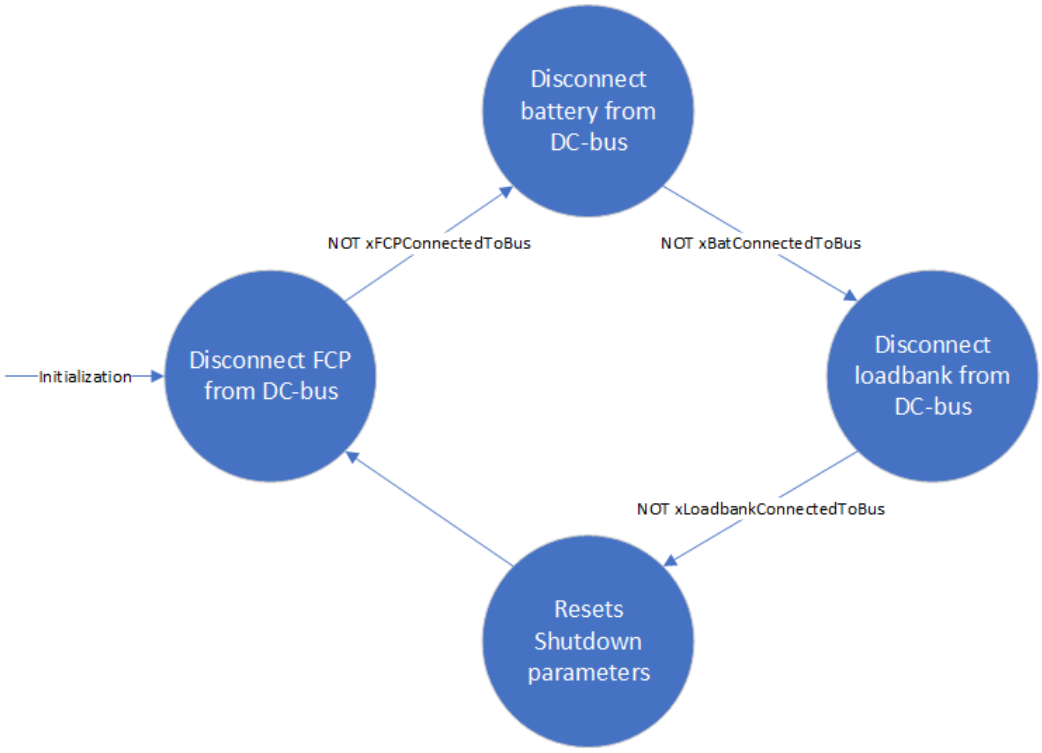
Hovedprogrammet er som en tilstandsmaskin, som vist i figur 14, som igjen har to tilstandsmaskiner under seg, som vist i figur 15 og 16. Dette kunne vært implementert som én stor tilstandsmaskin, men ble implementert som tre forskjellige deler for å gjøre koden mer oversiktlig.



Figur 14: Tilstandsmaskin for hovedprogram. Oppstart og nedstengningsprosedyren referer til henholdsvis figur 15 og 16



Figur 15: Tilstandsmaskin for oppstart

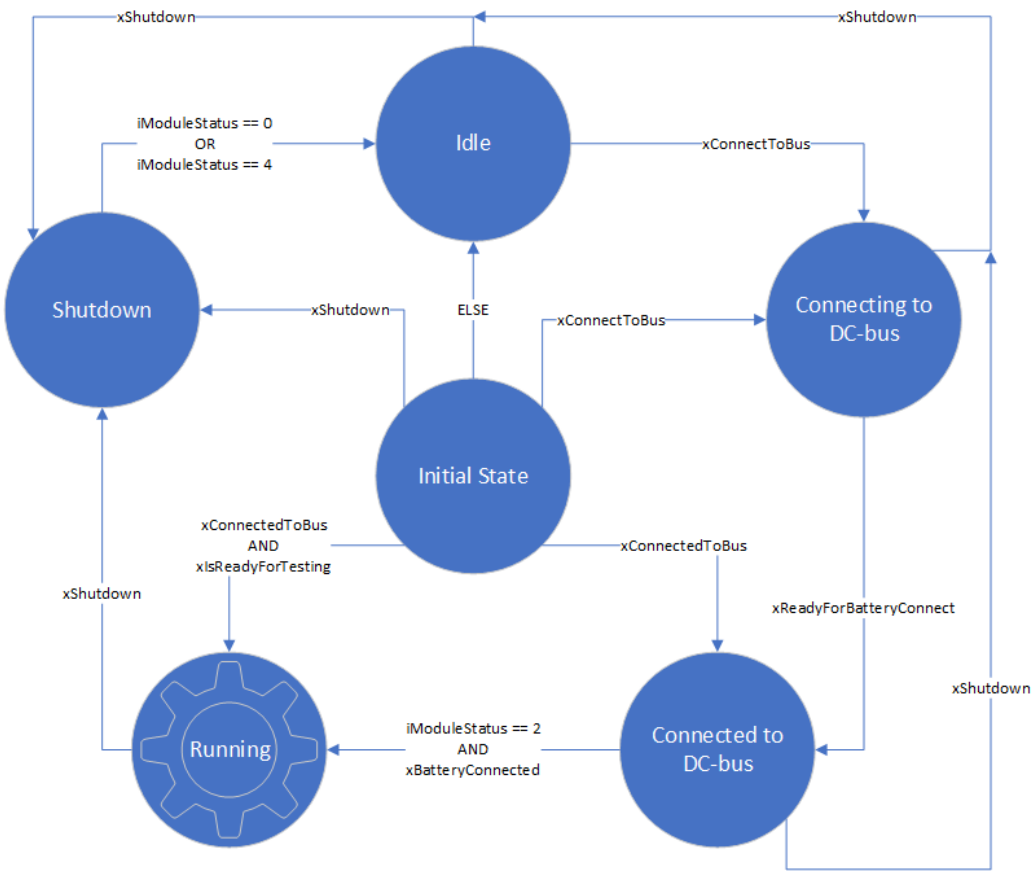


Figur 16: Tilstandsmaskin for nedstengning

4.1.1 Styring av brenselcellepakken

Brenselcellepakken blir styrt over Modbus TCP, den leser av tilgjengelig data ca. hvert sekund og oppdaterer variabler. Alle registrene ligger innenfor et kort nok spenn til at alt kan bli lest av i en operasjon. Noen av registrene man leser av er kommandoregistre og ubrukte registre. Men det er ikke et problem, ettersom det ikke har noen innvirkning på hva som blir skrevet til de. Skriving må gjøres med flere blokker, ettersom registrene er sortert slik at de første er for brenselcellepakken generelt, deretter er det fire bolker, hver bolk har informasjon for én modul. Hvis man skulle skrevet alt i ett, så ville man skrevet over registre som man senere leser av. Dermed kunne det oppstå uforutsette feil grunnet feilavlesninger.

Det er naturlig å styre brenselcellepakken i forskjellige trinn, avhengig av om man er i oppstart, testing eller avslutning. Dermed styres brenselcellepakken som en tilstandsmaskin som vist i figur 17. Det gjør det lettere å forstå programmet for brukeren.



Figur 17: Tilstandsmaskin for styringen av brenselcellepakke. Navn på på piler viser tilstanden variabler må være i på for at programmet skal gå videre i neste tilstand.

4.1.2 Styring av batteri

En funksjonsblokk ble laget for å kommunisere med og styre batteripakken. Funksjonsblokken leser av informasjon fra følgende registergrupper:

- Alarmer: Viser oversikt over advarsler og feil i batteripakke 1.
- Array status: Viser informasjon om strøm, spenninger, tid, SOC. etc. for alle batteripakkene i et ”array”.
- Status batteripakke: Viser om batteripakke 1 er koblet til, har feil, etc..

- Status for tilbakestilling av alarm: Viser når batteripakken sist mottok kommando for å tilbakestille alarmer.

All informasjonen utenom status for tilbakestilling av alarm ble lest av ble lagret i én variabel og lest av/oppdatert ca. hvert sekund, i understrukturene Array_Data og Alarms. Dette ble gjort ettersom det var store mengder data, og vi ikke visste hvilke vi trengte. Status for tilbakestilling av alarm ble brukt i koden for å tilbakestilling av alarm og kun lest ved behov.

Batteripakken trenger bare to styringssignaler, det ene er for å koble den til DC-bussen, det andre er for å tilbakestille alarmer. Disse blir skrevet ved behov.

Informasjon som skal brukes i regulering og/eller Modbus-tjener blir lagt ut som ”output”, informasjon som skal til HMI blir hentet direkte fra funksjonsblokken.

4.1.3 Styring av lastbank

Vi begynte med å løse kommunikasjon opp mot lastbank. Vi startet med å legge inn de predefinerte funksjonsblokkene i programmet vårt, og la inn de nødvendige variablene. Disse variablene er forskjellige BOOL variabler som setter av/på forskjellige funksjoner i blokkene, og nødvendige data som IP-adresser og porter for kommunikasjon.

Med en gang kommunikasjon er opprettet mellom tjener og klient, vil lastbank begynne å sende data. I vårt tilfelle er lastbank satt til tjener, og vår PLS er satt opp som klient. Når kommunikasjon er opprettet, håndterer vår PLS mottak av data, og tolker data slik at verdier blir satt inn i rett variabel. Mottak av data blir løst med den predefinerte funksjonsblokken for mottak av data, og så plasserer den mottatt data inn i de rette variablene.

Vi la så inn setningene for regulering, og la inn kode for konvertering av data og blokk for å sende data. Endringer ble verifisert i tester, og programmet ble videreutviklet.

Vi lagde også noen tilleggsfunksjoner: - Vi programmerte så inn en teller, for å unngå at lastbank får beskjed om å endre trinn for ofte. Vi la inn en funksjon for å kunne overkjøring av denne tiden, slik at man kan gjøre umiddelbare endringer.

- Vi la inn en funksjon for at hvis data ikke blir mottatt av lastbank og lastbank ikke endrer trinn, vil vår PLS gjøre tiltak. Den vil da kjøre en ny regulering og sende data igjen neste syklus. Dette ble løst ved at når array er sendt, blir array lagret i en mellomvariabel. Denne variabelen sier da hvilket trinn lastbank skal stå i. Denne variabelen blir så sammenlignet med hvilket trinn lastbank sier den står i. Hvis disse ikke er lik, blir det regulert igjen og sendt på nytt ved PLS sin neste syklus.

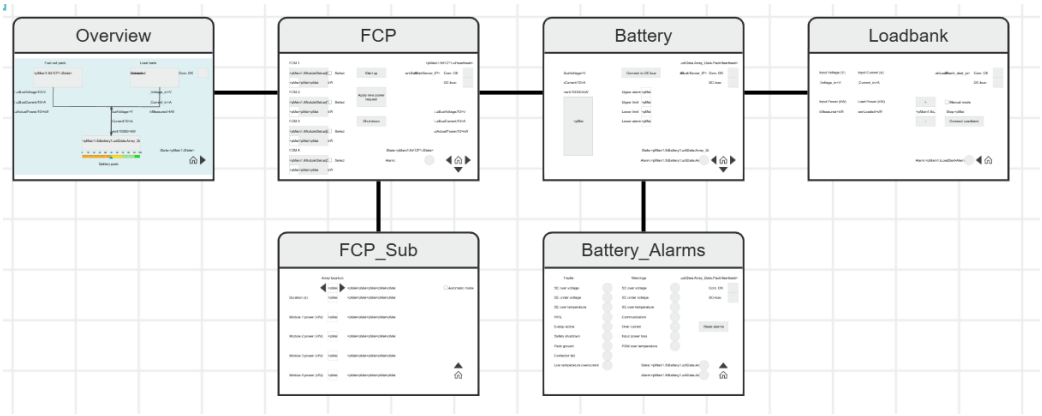
- Det ble lagt inn en Case-struktur som velger trinn manuelt ved hjelp av HMI.

4.1.4 Fysisk Grensesnitt

Vi har laget et HMI (Fysisk grensesnitt) for testprogrammet vårt. Dette er for å forenkle bruk av programmet, og for å gjøre det lettere å lese av verdier.

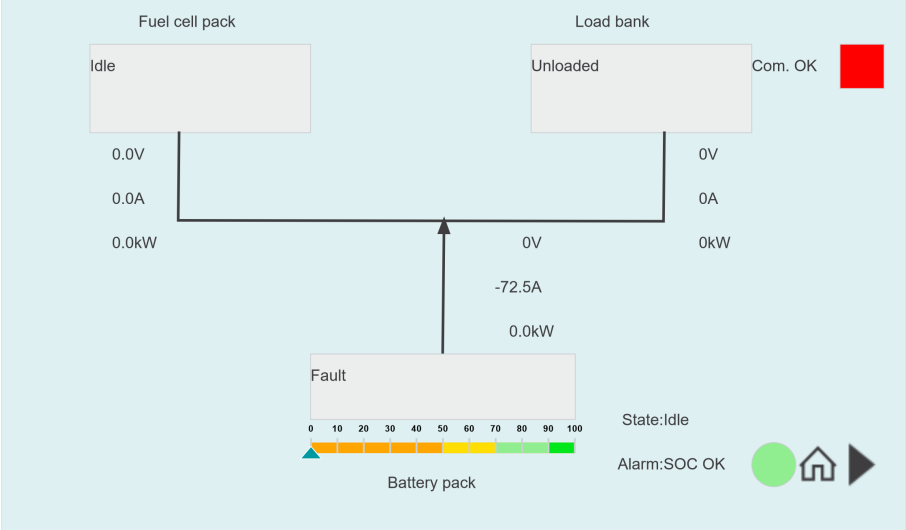
Sidene på skjermen leser verdier fra variabler i programmet. PLCnext Engineer er et veldig enkelt program å lage HMI i, ettersom den har innebygde funksjoner, og den henter variabler direkte fra programmene. Dette gjør at vi enkelt kan sette opp knapper og få ut data fra programmet, som gjør programmet mye lettere å bruke.

Vi har valgt å dele HMI opp i forskjellige sider slikt som vist i figur 18. Først en forside, som viser et overblikk over systemet. Videre kommer det sider for FCP, batteri og lastbank. Alle disse ligger etter hverandre, og noen av komponentene har fått undersider for å gi videre detaljer om komponenten. Alle sidene har status på om det er koblet på DC-bussen og om kommunikasjon er OK. Koblingsstatus og navigasjonsknappene er lagt på samme sted på hver side for lettere oversikt.



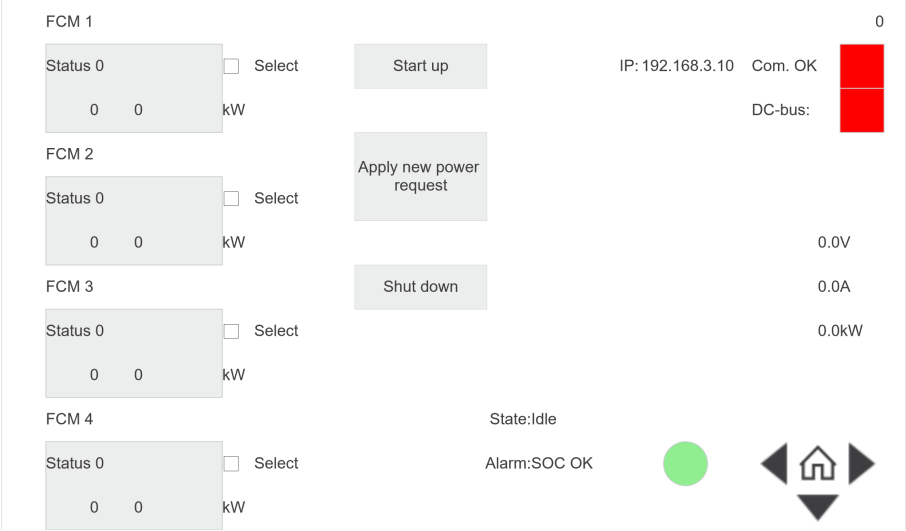
Figur 18: Oversiktsbilde med lokalisering av HMI-sider.

Hovedsiden som vist i figur 19 gir et overblikk over systemet og skal raskt vise status på de forskjellige komponentene. Den viser målt effekt, spenning og strøm for alle komponentene. Den viser også status på kommunikasjon.

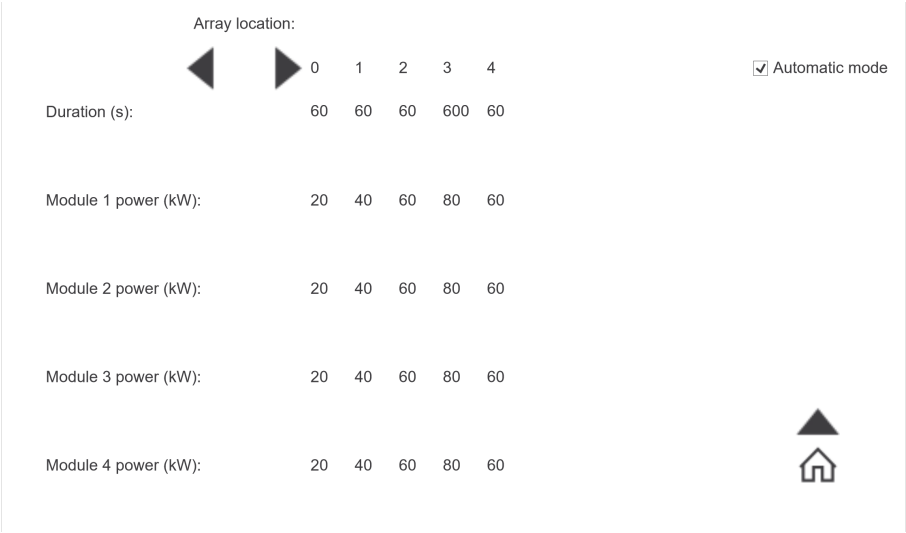


Figur 19: Hovedsiden til HMI

Siden for brenselcellepakken har, som vist i figur 20, mulighet for å styre alle modulene enkeltvis og brukes også for å starte opp alt og skru av alt i riktig rekkefølge. Siden for brenselcellepakken, har også en underside, figur 21, som brukes når man skal sette den over i programmodus, hvor man velge effekter frem i tid, og la systemet styre seg selv.

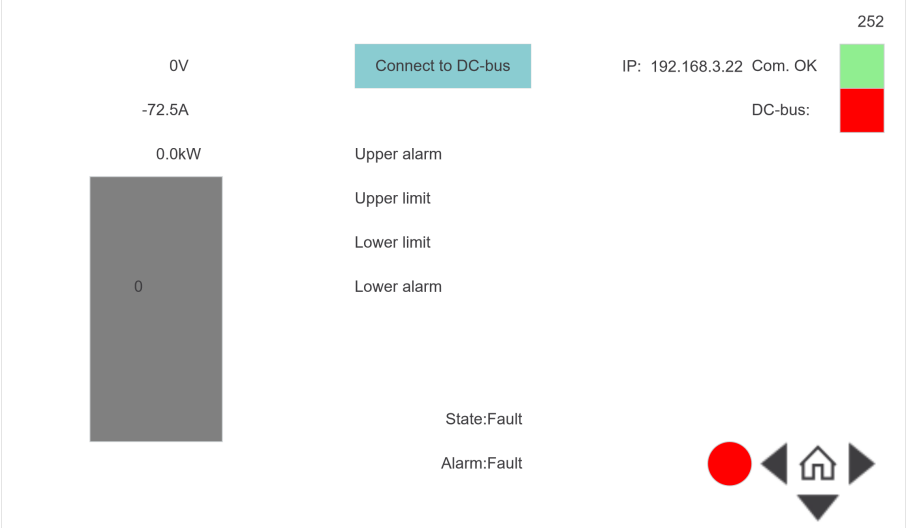


Figur 20: Side for manuell innstilling og start/stopp av FCP.

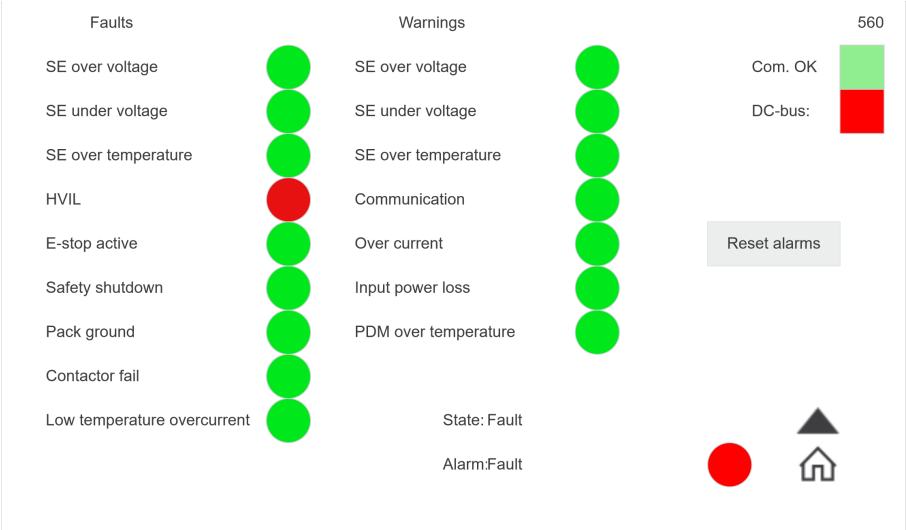


Figur 21: Side for automatisk kjøring.

Siden for batteripakken har, som vist i figur 22, oversikt over batteriets status. Her kan man forandre grensene for hva SOC skal være før programmet snur fra opplading til utlading og motsatt. Siden for batteripakken har også en underside som vist i figur 23 som gir en oversikt over alle alarmene, her kan man også tilbakestille alarmene.

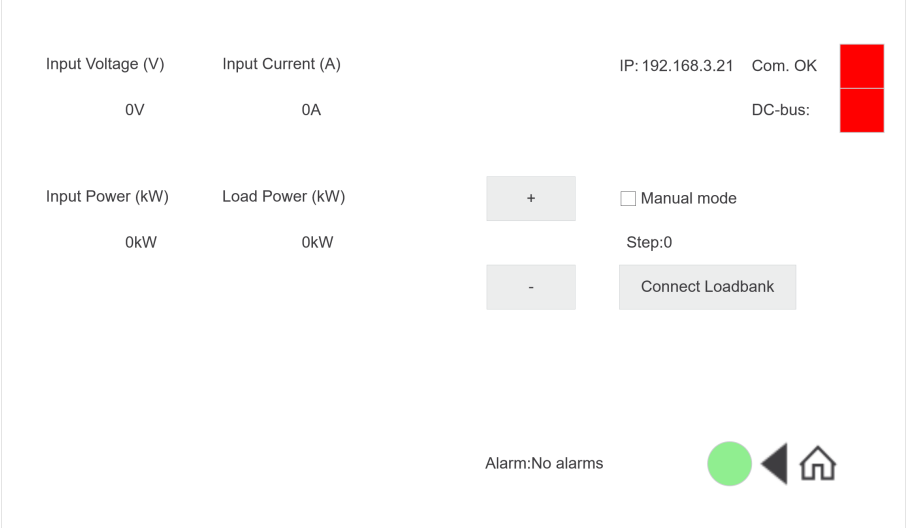


Figur 22: Oversiktside for batteriverdier, med mulighet for å sette grenser for SOC og koble til batteripakke manuelt



Figur 23: Oversikt over batterialarmer, med mulighet for tilbakestilling

Lastbank sin side, som vist i figur 24, viser alle variablene mottatt fra lastbanken. Den har også mulighet til manuell styring via HMI, hvor man setter trinnverdi eller trinner den opp og ned ett trinn. Dette må ikke forveksles med manuell modus fysisk på lastbank.



Figur 24: Oversikt over lastbankverdier med mulighet for manuell kontroll.

4.2 Urealiserte deler

Grunnet mangel på tid, så var det flere planlagte elementer som ikke ble realisert. Vi skulle prøve å integrere omformeren i styresystemet, men grunnet usikkerhet rundt hvordan det elektriske systemet kommer til å bli, fikk vi ikke gjort dette. Den nye koblingen vil ha flere DC/DC-omformere og vi var usikker på hvordan omformeren regulerer effekten ut, om man setter en effekt ut, eller et spenningsnivå inn eller en annen metode. Det ville vært tidkrevende å sette seg inn i dette. Men med måten programmet er bygd opp, med lite interaksjon mellom de forskjellige delen, så vil man sannsynligvis bare trenge ett til trinn i oppstarts- og nedstengningsprosedyren i tillegg til forandring i reguleringsdelen for at man skal kunne gi effekt til omformer i stedet for lastbank.

Modbus-tjeneren ble ikke implementert grunnet dårlig tid, men det ble lagt til en liste over variabler i dokumentasjonen som vil gjøre det lettere å implementere Modbus for Corvus.

Det var enkelte ting som ikke ble tatt med i HMI grunnet mangel på tid og problemer under testing, dette var alarmlogg, som vi ikke fikk tid til. Vi fjernet også en graf som skulle vise effektene til de forskjellige modulene, ettersom den kræsjet HMI-et.

Koden som styrer lastbank kan også forbedres ved at avbrent effekt blir utreg-

net i programmet og ikke er basert på en fast spenning. Valg av trinn kan også mulig skrives mer oversiktlig i koden som for-løkker og med verdiene i et array.

5 Testing

Testene er beskrevet etter rekkefølge.

5.1 Innledende simulering

I starten av programmeringsfasen ble det gjort en del innledende testing for å se at Modbus koblet seg mot serveren og leste fra riktige registre. Til å simulere dette, brukte vi et program som heter Ananas. Dette er en Modbus-simulator, hvor man kan forandre og lese av registerverdier manuelt. Vi har i tillegg brukt et batteri-simuleringsverktøy fra Corvus Energy, som simulerer forskjellige scenarioer fra batteripakkeserien Orca. Denne simulatoren kan også sende feilkoder og annet som kan brukes til feilsøking og testing.

5.2 Oppkobling mot HMI-grensesnitt

Vi har også gjort noe testing med HMI-grensesnitt. Det vi brukte til dette, var en berøringsskjerm som kobles direkte til HMI via en nettleser.

5.3 Første test på testanlegget

Under den første testen var ikke brenselcellepakken koblet opp. Vi testet derfor bare opp mot lastbank og opp mot batteripakke, og i denne fasen var det der behovet var. Hovedmålet vårt med denne testen var å få kontakt med lastbank over Ethernet TCP, teste ut hypotese om dataplassering fra lastbank, og å prøve ut kode som snakker med batteripakken.

Testen startet med at vi koblet vår PLS opp mot switch på testanlegget, ordnet riktige IP-adresser og lastet programmet vårt opp til PLS.

Med en gang programmet startet, begynte lastbank å sende data. Dette så vi ved at funksjonsblokken til sokkelen sendte tilbakemelding om at kommunikasjon er OK. Vi så ikke noe endring på data, ettersom det bare ble sendt ”0” på alle verdiene. ”0” på alle verdiene var også riktig ihht. status på lastbank.

Vi gikk så videre i test, for å sjekke om data fra lastbank ville bli plassert slik vi håpte.

For å teste dette, ble batteripakken lagt inn slik at vi ville få en endring i spenning inn på lastbank. Når batteripakke ble lagt inn, ble det dermed observert at de rette plassene i mottatt data ble endret. Etter en rask utregning så vi at dette stemte med batterispenningen, og vi kunne derfor konstatere at mottatt data fra lastbanken blir mottatt slik som ønsket. Altså, data mottatt blir plassert på rett plass i byte-array-variabelen for mottatt data. Etter dette resultatet, valgte vi derfor å anta at resten av mottatt data vil ligge der det skal i byte-array.

På denne testen fikk vi også testet ut at vi mottok variabler fra batteripakken, og vi fant ut av at vi må endre hvilken variabel vi brukte for å finne SOC. Vi brukte en verdi for tilkoblede batteripakker, noe som gjorde at vi ikke fikk vite SOC før vi koblet til batteriet.

5.4 Simuleringer mellom første og andre test

Ut fra resultater fra første test, videreutviklet vi koden. Under denne prosessen, testet vi også med Ananas og simulator for batteripakker for å se at endringene våre hadde utrettet det de skulle. Vi lagde også et veldig enkelt simuleringsprogram for lastbanken internt i PLS, slik at vi kunne få sett at variablene endret seg sånn de skulle.

Det viste seg også at vi hadde noen utfordringer med å skrive til brenselcellepakken over Modbus som ble ordnet.

Vi fikk også frem noen andre småfeil, som vi rettet underveis.

5.5 Andre test på testanlegget

Målet med den andre testen på testanlegget, var å snakke med batteripakke, og å snakke med lastbank.

Vi fikk noen utfordringer med sending av data mot lastbank, og dette viste seg å være grunnet litt feilprogrammering. Det resulterte i at sendeblokk ikke lukket kommunikasjon når den sendte data, og da ville den ikke sende ny data, ettersom den ikke fikk avsluttet sending. Vi manipulerte verdiene i programmet manuelt, for å finne ut av virkemåten til funksjonene. Etter litt eksperimentering, fikk vi et bilde over hvordan dette må programmeres for at vi får kommunisert med lastbank

slik som ønsket.

5.6 Simuleringer mellom andre og tredje test

Her simulerte vi for å få vekk diverse feil som var avdekket under test på testanlegget. Vi fikk også fjernet noen feil som ble avdekket under simuleringer, og vi fikk videreutviklet programmet noe. Vi fikk også testet at programmet følgte oppstartsprosedyren som var spesifisert av Corvus. Vi testet også ut en graf som skulle vise effekten ut fra de forskjellige modulene over tid, men dette førte til at HMI kræsjet og dermed ble dette tatt vekk.

5.7 Tredje test på testanlegget

På denne testen var det komplett testanlegg med brenselcellepakke. Vi skulle teste batteripakke og lastbank som et komplett system, med drift på brenselcellepakken. Først simulerte vi drift på brenselcellepakken ved at vår veileder fra Corvus Energy manuelt sendte vår PLS de variablene vi trengte for at vårt program skulle tro at brenselcellepakken var i drift. Programmet vårt gjorde som det skulle, og vi kunne teste om lastbank og batteripakke gjorde som det skulle under drift av brenselcellepakken. Programmet gjorde som forventet, men vi hadde noe småfeil i HMI, og det var noen småting som måtte fikses før programmet var klar til bruk.

Brenselcellepakken ble så lagt inn og satt på en lav effekt, for å se om det virker. Programmet gjorde det som det skulle, lastbank og batteripakke regulerte som det skulle ved endret effekt på brenselcellepakken. Vi så også behovet for en funksjon som manuelt kan regulere lastbank (som ble lagt inn etter test), og vi fant noen feil i innlest data, fra lastbank. Vi fant også noen små feil i styringen til batteripakken.

5.8 Fjerde test på testanlegget

I den fjerde testen skulle vi teste om vi greide å få programmet til å kjøre hele prosessen. En del styring internt i brenselcellepakken måtte manuelt styres av veileder ettersom dette ikke var helt ferdigstilt.

Programmet virket som det skulle, men vi fant ut at hvis man prøvde å avbryte oppstarten, så gikk ikke programmet tilbake i riktig innstilling og man kunne ende opp med f.eks. at batteripakken eller brenselcellepakken hadde fått signalet for å

koble seg til, men at dette signalet ikke hadde blitt tilbakestilt når man prøvde å skru av. Vi så også et behov for en manuell knapp for å koble til lastbank. Det ble også bestemt at man under avslutning ville sette lastbanken på laveste trinn for å lade ut spenningen på DC-bussen.

6 Diskusjon

6.1 Omformer

Corvus fikk presentert de tre valgene vi mente var mest relevante for dem, presentasjonen ligger i vedlegg B. De kjenner systemet best og derfor er det naturlig at de velger hvilken løsning de ønsker. Ettersom de valgte en annen løsning før vi fikk presentert, vil vi her presentere hvilken omformer vi syntes ville egnet seg best for Corvus sitt tiltenkte formål.

Vi mener at den beste løsningen fra våre alternativer, så ville Corvus vært mest fornøyd med Danfoss sine omformere. Dette er hovedsakelig fordi de blir i stor grad benyttet i skip i dag. Det ville dermed gjort testsystemet så realistisk som mulig, for å gi brenselcellepakkene best kriterier for realistiske tester. Valget av Danfoss vil være avhengig av at spenningområdet som den opererer i, gir Corvus stort nok spenningsområde.

Ut fra dokumentasjon på Elektro-Automatik sin omformer, kan denne være et bedre valg. Problemet her er at vi ikke har nok innsikt i hva som skal utføres av tester, til å si dette med sikkerhet. Elektro-Automatik sin omformer er mer relevant for labmiljøer som tester brenselcellestacker og brenselcellemoduler, men ikke et komplett system for skip. Elektro-Automatik sin omformer vil sannsynligvis kunne måle rippelspenninger på DC-bussen, ettersom den er tiltenkt labmiljø. Dette kan være relevant for levetiden til brenselcellepakkene ettersom rippelspenninger kan ødelegge katalysatoren i PEM-celler. Vi antar at Toyota har gjort inngående tester på modulene, og at Corvus ikke har behov for så detaljerte tester. Elektro-Automatik sin omformer er det beste andrevalget basert på at de har bedre nedre spenningsområde enn AEG. Kostnadsmessig er Elektro-Automatik i en mye høyere prisklasse enn AEG og Danfoss.

6.2 Behovet for omformer

Dersom dette anlegget skal brukes til testing av et nytt produkt, vil ikke omformer være et økonomisk lønnsomt grep. Men, dersom alle fremtidige brenselcellepakkene som skal produseres, skal gjennom testing før de blir levert til kunde, vil omformeren gi en mye større økonomisk gevinst. Hydrogenet vil kunne delvis bli finansiert

med produsert elektrisitet, men omformeren vil nok ikke lønne seg økonomisk, da produsert effekt mest sannsynlig ikke vil gi nok gevinst, til å tjene inn igjen selve omformeren.

Hvis det er snakk om at alle pakker skal gjennom testen, og at det skal produseres et stort antall brenselcellepakker som skal testes over lengre perioder før leveranse, vil det bli betydelige mengder energi som går tapt. Hvis alt skal brennes av i lastbank, vil det bli en god del energi som bare blir omgjort til varme. Ved å ha en omformer kan dette mates ut på nettet, og da kan den produserte energien bli brukt til noe annet en varme. Så, hvis det er snakk om en stor mengde brenselcellepakker som skal produseres, kan omformeren potensielt gi en inntjening, men vi antar at det må produseres ganske mange brenselcellepakker før vi kommer til et punkt hvor omformeren vil gi økonomisk gevinst.

Grunnen til å ha en omformer i anlegget, bør derfor ikke baseres på kostnad, men for å gi muligheter for flere forskjellige typer tester, som for eksempel jordfeil på AC-siden av omformeren.

Det burde kanskje vurderes å endre litt på testprofiler på brenselcellepakkene, hvis det er ønskelig å levere mesteparten av produsert energi ut på nettet. Endrede testprofiler kan gi anlegget bedre forutsetninger for å ta opp all produsert energi, ved at det blir mindre store topper over lang tid. Da kan det kanskje være mulig å levere all energi produsert av brenselcellepakkene ut på nettet. Når testen går, kan da omformer gå hele tiden, og overskytende effekt fra hva omformer klarer å levere ut på nettet, kan lagres på batteriet, og batteriet kan da tappes ned til ønsket nivå, over tid etter endt test.

Vi har ikke lagd programmet vårt slik, da vi har programmert rundt det eksisterende anlegget.

6.3 Styringssystem

Testingen vi gjorde viste at vi greide å styre både lastbank, brenselcellepakkee og batteripakke, dermed tilfredsstiller vi kravene til styring av disse. Alle komponenter ble startet og stoppet i riktig rekkefølge under testing. HMI fungerer for å håndtere forespørsler, men Modbus-server har ikke blitt implementert. Reguleringen har ikke vært testet over tid, men er laget slik at den skal holde spenningen på DC-bus

innenfor riktige verdier, spesifisert av SoC. Vi har ikke gjort tester for å teste hvor robust styresystemet er. Altså hvor lett det kan oppstå feil hvis man trykker på knapper utenfor det som er vanlig bruk. Vi har heller ikke gjort tester for å se hvordan systemet fungerer over tid, ettersom testene er begrenset av testplanen til Corvus, og de vil være på den sikre siden og bruke sine ferdigtestede men tungvinte systemer for å unngå unødvendige forsinkelser. Det er noen ting som vi ville gjort annerledes hvis vi skulle begynt på prosjektet på nytt, som er omtalt i kapittel 4.2.

Det er usikkert hvordan en nettomformer kan implementeres senere, men ettersom selve reguleringen i dag kun består av en enkel regulering av lastbank, så vil en lett kunne forandre på reguleringstypen ved å først regulere for omformer, for å så sende overskytende effekt over til lastbank hvis omformer ikke kan ta imot all effekten.

Dokumentasjonen ble levert etter kravene, utenom Modbus-tjener, se vedlegg C for brukerdokumentasjon til Corvus.

6.4 Kobling til strømnettet

Opprinnelig trodde vi at anlegget ikke ville utløse krav om konsesjonsmelding, grunnet lave energimengder. Men etter vi har sett mer på dette, ser vi at dette må søkes om og bestemmes av NVE hvis de produserer mer effekt enn de forbruker. Vi antar at det ikke blir levert noe mot nettet, da energi produsert av omformer, med de energimengdene vi har sett på, vil mest sannsynlig bli oppbrukt internt på Corvus sitt nett. Vi antar da at hvis omformer blir styrt slik at det alltid ”kjøpes” litt kraft, vil det ikke være behov for konsesjonsmelding, da det ikke vil bli levert noe energi til distribusjonsnettet.

6.5 Tidsplan

Vi hadde opprinnelig en tidsplan for prosjektet. Denne fulgte vi til liten grad, og etter hvert så vi at denne ikke var veldig relevant for denne oppgaven. Vi måtte løpende planlegge da vi måtte rette oss etter timeplanen til Corvus for å kunne utføre testing av programmet vårt. Vi kunne derfor ikke lage en konkret tidsplan på fremdrift av oppgaven, i forkant av arbeidet. Mellom testing, jobbet vi med programmet og oppgaven.

7 Konklusjon

Programmet vi lagde, oppnådde en tilfredstillende styring av anlegget. Fra erfaringene vi har fått i denne prosessen, ser vi at det er noen ting vi kunne gjort anderledes for å få til et bedre program, og løsninger som ville gitt en mer oversiktlig kode, og en bedre styring av testanlegget.

Det er også slik at Corvus har besluttet å utvide anlegget med omformer, og da er det ikke sikkert at programmet vil styre anlegget like bra hvis denne blir kodet inn i programmet vi har laget. Det kan i dette tilfellet derfor være bedre å ta lærdom i det vi har funnet ut av, og bruke vårt program som et underlag, når vi tar hensyn til at dette programmet også skal styre en omformer vi ikke har noen data på.

Vi anser ikke anlegget som konsesjonspliktig ut fra våre funn, gitt produsert energi blir brukt opp internt hos Corvus. Vi anbefaler Corvus å avklare dette med relevante myndigheter, da vi ikke fikk noe svar fra NVE. Omformeren Corvus har valgt, leverer også i denne størrelsesordenen at det mest sannsynlig ikke vil bli levert noe effekt ut på nettet. Produsert energi vil derfor gå utelukkende til internt forbruk, og Corvus har derfor ikke behov for å være plusskunde. Vi anbefaler å måle ”kjøpt” strøm for å kunne styre omformer, slik at det ikke blir produsert for mye under testing.

Endelig løsning på hvordan anlegget skal se ut, bør ses i sammenheng med endelig bruksområde, og dette må Corvus bestemme selv.

Referanser

[1] The Corvus Energy Journey;. Available from: https://corvusenergy.com/about/history-2/.

[2] Corvus Energy - Powering a clean future;. Available from: https://corvusenergy.com/.

[3] Recycling;. Available from: https://corvusenergy.com/sustainability/recycling/.

[4] Certificates;. Available from: https://corvusenergy.com/certificates/.

[5] Review of Maritime Transport 2020. UNCTAD;. Available from: https://unctad.org/system/files/official-document/rmt2020ch2_en.pdf.

[6] Ryste JA. Comparison of Alternative Marine Fuels. DNV GL;. Available from: https://sea-lng.org/reports/comparison-of-alternative-marine-fuels/.

[7] Tabeller for omregning fra energivare til kWh - Miljødirektoratet;. Available from: https://www.miljodirektoratet.no/ansvarsomrader/klima/for-myndigheter/kutte-utslipp-av-klimagasser/klima-og-energiplanlegging/tabeller-for-omregning-fra-energivarer-til-kwh/.

[8] Energy C. Corvus Energy Inherently gas safe marine fuel cell system awarded Approval in Principle by DNV; 2022. Available from: https://corvusenergy.com/corvus-energy-inherently-gas-safe-marine-fuel-cell-system-awarded-approval-in-prin

[9] KLD. Handlingsplan for grønn skipsfart [Plan]; 2019. Publisher: regjeringen.no. Available from: https://www.regjeringen.no/no/dokumenter/handlingsplan-for-gronn-skipsfart/id2660877/.

[10] Nå har den elektriske ferjen MF «Ampere» gått 6 x ekvator - Norled;. Available from: https://www.norled.no/nyheter/ampere---6-ganger-rundt-ekvator/.

[11] MF Ampere;. Available from: https://corvusenergy.com/projects/mf-ampere/.

[12] H2NOR Development Project;. Available from: https://corvusenergy.com/products/fuel-cell-systems/h2nor-fuel-cell/.

[13] Lauritzen . Elektriker Vg3. Elforlaget;.

[14] AMS målar;. Available from: https://www.morenett.no/informasjon/maalar.

[15] Kraftsystemmodellen;. Available from: https://norgesnett.no/kraftsystemmodellen-2/.

[16] Om BKK;. Available from: https://www.bkk.no/om-bkk.

[17] BKK.no;. Available from: https://www.bkk.no/.

[18] Forskrift om leveringskvalitet i kraftsystemet - Lovdata;. Available from: https://lovdata.no/dokument/SF/forskrift/2004-11-30-1557.

[19] Forsyningssikkerhet;. Available from: https://energifaktanorge.no/norsk-energiforsyning/forsyningssikkerhet/.

[20] Mohan N, Undeland TM, Robbins WP. Power electronics: converters, applications, and design. John wiley & sons; 2003.

[21] Harlow JH. Electric power transformer engineering. CRC press; 2003.

[22] NVE - Norges vassdrags- og energidirektorat;. Available from: https://nve.no/.

[23] energidepartementet Oo. Ny ordning for deling av eigenprodusert, fornybar straum [Pressemelding]; 2023. Publisher: regjeringen.no. Available from: https://www.regjeringen.no/nn/aktuelt/ny-ordning-for-deling-av-eigenprodusert-fornybar-straum/id2964122/.

[24] Dicks AL, Rand DAJ. 12.2.5 Fuel-Cell Interface and Grid Connection Issues. In: Fuel Cell Systems Explained (3rd Edition). John

Wiley & Sons;. Available from: https://app.knovel.com/hotlink/pdf/id:kt011J1TA1/fuel-cell-systems-explained/fuel-cell-interface-grid.

[25] PVS-175-TL | Fimer Spa;. Available from: https://www.fimer.com/three-phase/pvs-175-tl.

[26] MAX 185-253KTL3-X HV | Utility-Scale PV Inverter | Growatt;. Available from: https://www.ginverter.com/products/max-185-253ktl3-x-hv.

[27] Markvart T. Solar electricity. vol. 6. John Wiley & Sons; 2000.

[28] Maximum power point tracking; 2023. Page Version ID: 1138906198. Available from: https://en.wikipedia.org/w/index.php?title=Maximum_power_point_tracking&oldid=1138906198.

[29] AEG. ConvertSC Flex. AEG;. Available from: https://www.aegps.com/en/products/converters-renewables-integration-systems/convert-sc-flex/.

[30] EA-PUB 10000 4U;. Available from: https://www.eapowered.com/product_series/ea-pub-10000-4u/.

[31] EA-PUB 10000 6U;. Available from: https://www.eapowered.com/product_series/ea-pub-10000-6u/.

[32] VACON® NXP Grid Converter;. Available from: https://www.danfoss.com/en/products/dds/system-modules/vacon-system-modules/vacon-nxp-grid-converter/.

[33] Fuel-cell to grid converter | Prodrive Technologies;. Available from: https://prodrive-technologies.com/products/power-conversion/converters/fuel-cell-to-grid-converter/.

[34] Bidirektional programmable Battery-Tester - Gustav Klein | Stromversorgungslösungen weltweit;. Available from: https://www.gustav-klein.com/en/divisions/testing/bidirektionaler-programmable-battery-tester/.

[35] Dicks AL, Rand DAJ. 4. Proton-Exchange Membrane Fuel Cells. In: Fuel Cell Systems Explained (3rd Edition). John Wiley & Sons;. Available from: https://app.knovel.com/hotlink/pdf/id:kt011J1M63/fuel-cell-systems-explained/proton-exchange-membrane.

[36] Dicks AL, Rand DAJ. 12.2.3 Step-up Regulators. In: Fuel Cell Systems Explained (3rd Edition). John Wiley & Sons;. Available from: https://app.knovel.com/hotlink/pdf/id:kt011J1T64/fuel-cell-systems-explained/step-up-regulators.

[37] Toyota. TOYOTA Gen2 Fuel Cell Module. Toyota;.

[38] Corvus Orca Energy;. Available from: https://corvusenergy.com/products/energy-storage-solutions/corvus-orca-energy/.

[39] An Introduction to the Modbus Communication Protocol;. Available from: https://www.solisplc.com/tutorials/modbus.

[40] Function block library Modbus_TCP_12 for PLCnext Engineer. Phoenix Contact;. Available from: https://www.plcnextstore.com/eu/app/1964.

[41] Forskrift om sikkerhet ved arbeid i og drift av elektriske anlegg - Lovdata;. Available from: https://lovdata.no/dokument/SF/forskrift/2006-04-28-458.

[42] Derfor bør du bruke et måleinstrument klassifisert til CAT III til PV-solenergianlegg | Fluke;. Available from: https://www.fluke.com/no-no/finn-ut-mer/blogg/fornybar-energi/cat3-stromtang.

Vedlegg A - Reguleringssimuleringskode

```
%Illustrasjon av regulering.
%Antar en batteripakke på 100kWh, hvor spenningen er på 695V ved 40% og
%705V ved 60% og lineær sammenheng mellom spenning og SOC.

clc, clear variables, close all

V0 = 700;                   %Startspenning, i Volt
V_SOC_0 = 675;              %Spenning ved SOC = 0%, i Volt
V_SOC_100 = 725;            %Spenning ved SOC = 100%, i Volt
SOClav = 40;                %Nedre energinivå batteri for regulering, i kWh/%
SOChoy = 60;                %Øvre energinivå batteri for regulering, i kWh/%
Batkap = 100;               %Effekt i batteriet ved 100% SOC, i kWh
P_FCP = 150000;             %Effekt produsert av FCP, i W.
G_opp = 0.2980;             %Ledningsevne trinn 1, opplading
G_ut = 0.3474;              %Ledningsevne trinn 2, utlading
G = G_ut;                   %Nåværende ledningsevne, variabel
dt = 60;                    %Steglendge i sekund
tmaks = 72000;              %Kjøretid, i sekund
t = zeros(tmaks/dt+1);      %Lager tabell for alle tidsverdiene
V = zeros(tmaks/dt+1);      %Lager tabell for alle spenningsverdiene
SOC = zeros(tmaks/dt+1);    %Lager tabell for alle SOC-verdiene
t(1)  = 0;                  %Tidspunkt, i sekund, startverdi
V(1)  = V0;                 %Spenning, i Volt, startverdi
SOC(1) = 50;                %Energinivå batteri, i kWh/%
n = 1;                      %Steg i prosessen
kwh_per_joule = 3600000;    %kWh per joule, til konvertering


while t(n) < tmaks
    SOC(n+1) = SOC(n) +(P_FCP - (V(n)^2)*G)*dt/kwh_per_joule; %Oppdaterer SOC basert↙
på effekt
    V(n+1) = SOC(n+1)*(V_SOC_100-V_SOC_0)/Batkap + V_SOC_0; %Oppdaterer spenning

    %Vurdere om man skal lade ut eller opp, setter elementverdi
    if SOC(n+1) > SOChoy
        G = G_ut;
    elseif SOC(n+1) < SOClav
        G = G_opp;
    end

    %Oppdaterer tid og steg
    t(n+1) = t(n) + dt;
    n = n + 1;
end

%Lager figur
figure('Name','Simulering av regulering');
plot(t, SOC, [0, tmaks], [SOChoy, SOChoy], [0, tmaks], [SOClav, SOClav])
axis([0 tmaks 0 100])
xlabel('Tid (s)')
ylabel('SOC(%)')
```

Vedlegg B - Presentasjon av omformere

Valg av omformere

AEG – Convert SC Flex HV

Bruksområde: Batteritilkobling til strømnett for å stabilisere nettet.

DC:

- Spenningsområde: 700 – 1400V
- Strøm: 1450A
- Effekt: 545kW ved 1400V. Mer ved lavere effekter.

AC:

- Fleksibelt grunnet transformator.
- Er lagd for å stabilisere strømnett, bør være lett å få godkjent hos BKK.

Pris: ...€ for modul.

Leveringstid: Ukjent

AEG – Convert SC Flex HV

For:

- Stort spenningsområde.
- Gir muligheter for å teste større systemer.
- Sannsynligvis lett å få innpass hos BKK.
- Kan leveres i kontainer med transformator.

Mot:

- For stort for oss?
- Kan hende BKK ikke vil ha et system som kan levere så store effekter?

EA-PSB 11500 4U

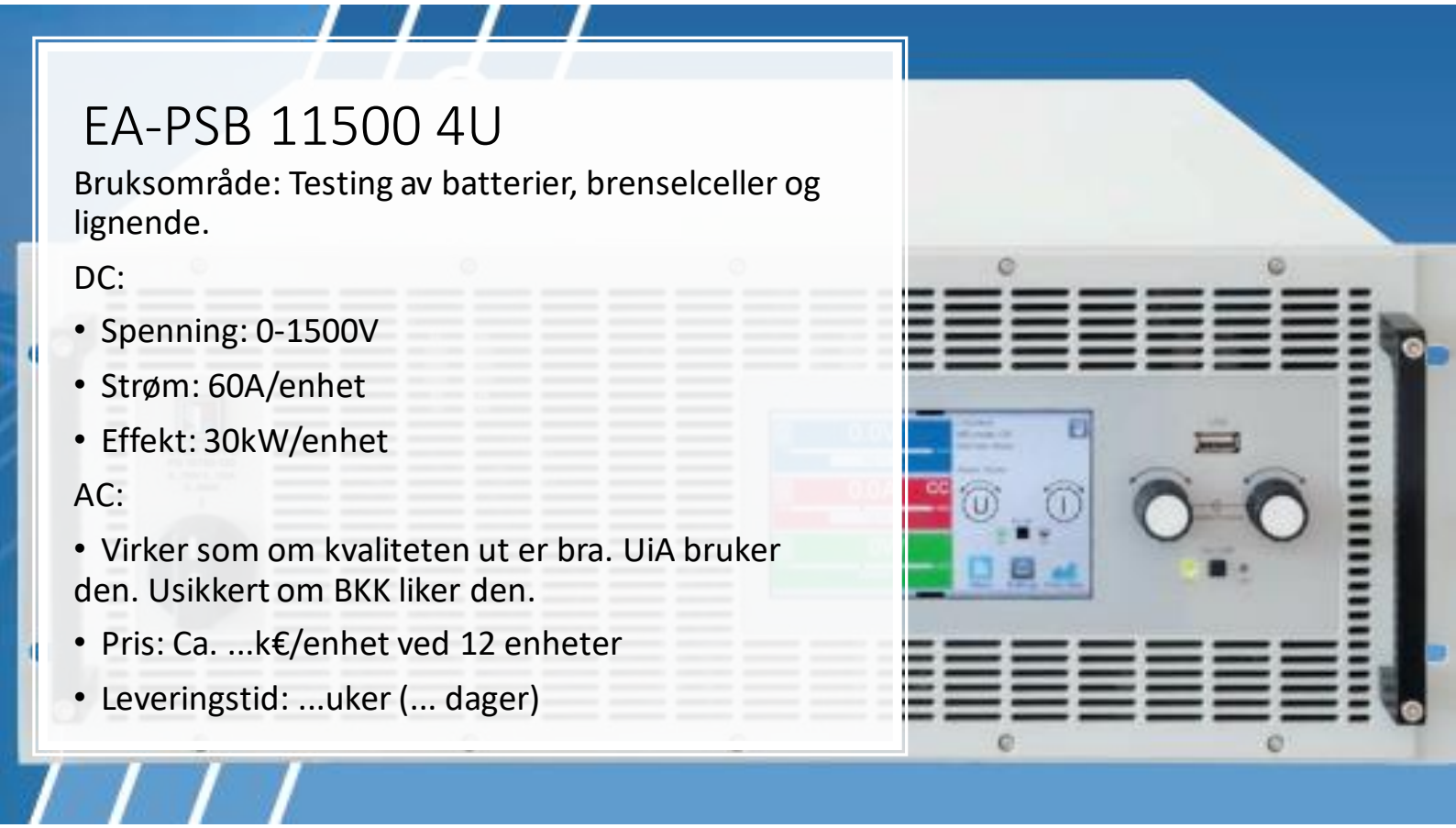
Bruksområde: Testing av batterier, brenselceller og lignende.

DC:

- Spenning: 0-1500V
- Strøm: 60A/enhet
- Effekt: 30kW/enhet

AC:

- Virker som om kvaliteten ut er bra. UiA bruker den. Usikkert om BKK liker den.
- Pris: Ca. ...k€/enhet ved 12 enheter
- Leveringstid: ...uker (... dager)



EA-PSB NY 60kW versjon

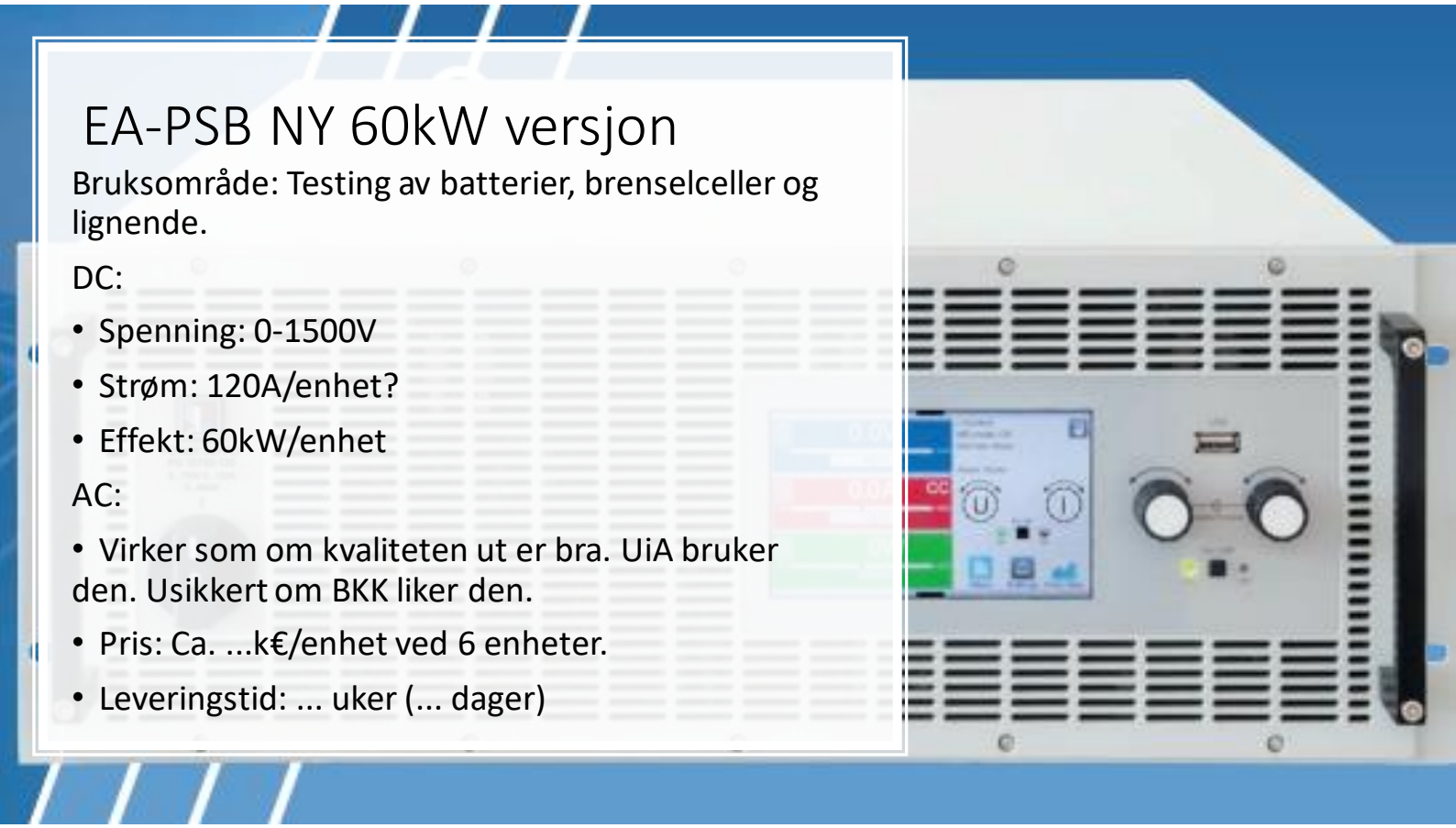
Bruksområde: Testing av batterier, brenselceller og lignende.

DC:

- Spenning: 0-1500V
- Strøm: 120A/enhet?
- Effekt: 60kW/enhet

AC:

- Virker som om kvaliteten ut er bra. UiA bruker den. Usikkert om BKK liker den.
- Pris: Ca. ...k€/enhet ved 6 enheter.
- Leveringstid: ... uker (... dager)



EA-PSB 11500 4U

For:

- Beste spenningsområde på DC-siden
- Er lagd for testing
- Modulær, kan utvides ved behov

Mot:

- Usikkert på AC-siden, kan være vanskeligere å få innpass hos BKK.
- Dyr

Danfoss - VACON® NXP Grid Converter

Bruksområde: Smarte nett, båter.

DC:

- Spenning: 650-1100V
- Strøm: 590A
- Maks effekt: ~340kW
- Kan ha høyere effekt og strøm ved andre valg

AC:

- Støtte for Grid Codes
- Er allerede mye i bruk, bør være greit å få godkjent hos BKK
- Pris: Ca. ...NOK, mulighet for rabatter
- Leveringstid: Ligger på rundt ... dager.

Danfoss - VACON® NXP Grid Converter

For:

- Laget for båter, relevant for Corvus.
- Allerede i bruk hos Corvus
- Støtte for grid codes, burde være lett å få innpass hos BKK

Mot:

- Lav maksspenning relativt til de andre valgene.

Vedlegg C - Brukerdokumentasjon

Documentation for PLC system for Corvus Fuel Cell test site

Preface

This documentation is intended for users of the PLC-program for the FCP test site. The PLC-program is a part of a bachelor thesis by Espen Stornes and Tommy Tobiassen.

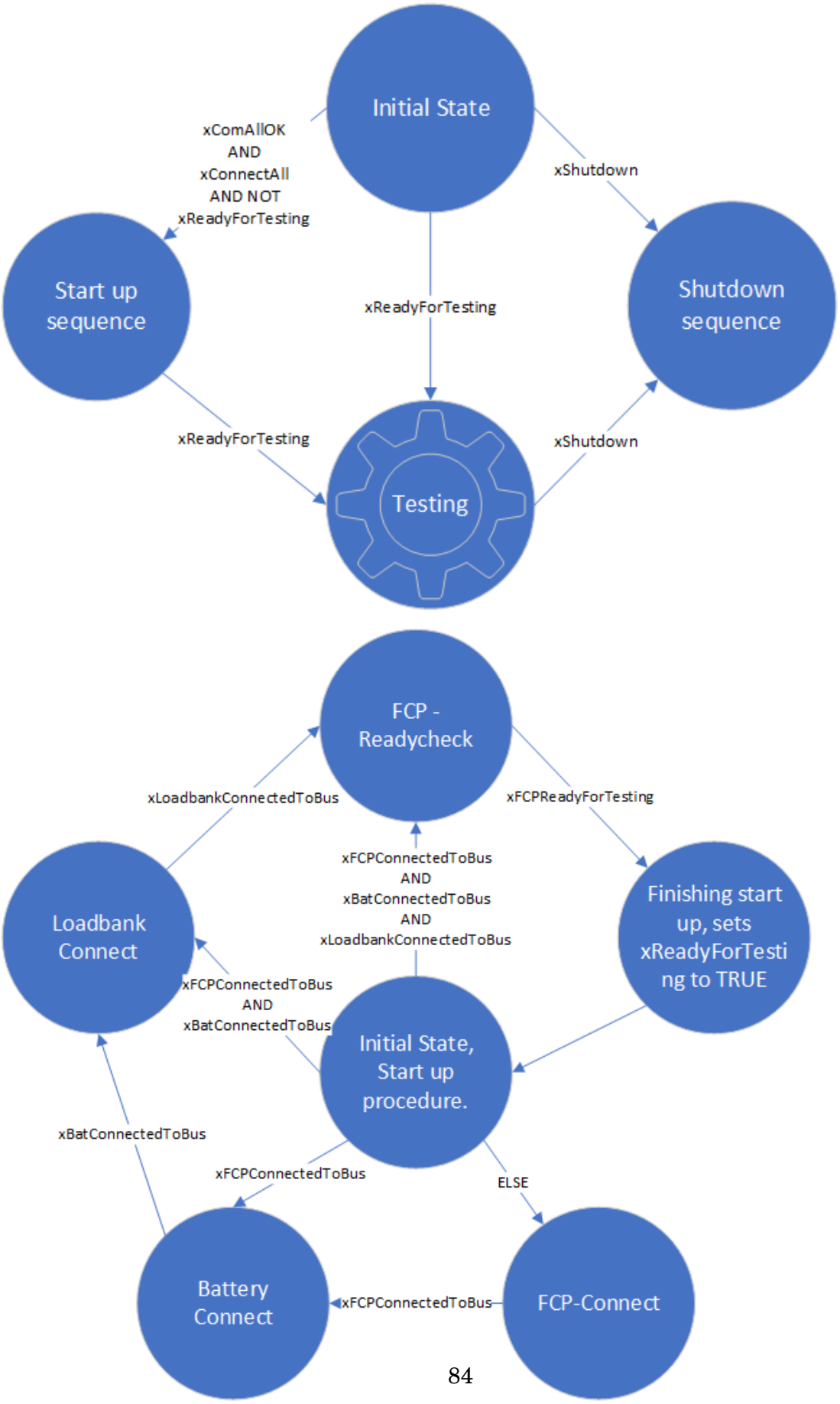
Table of contents

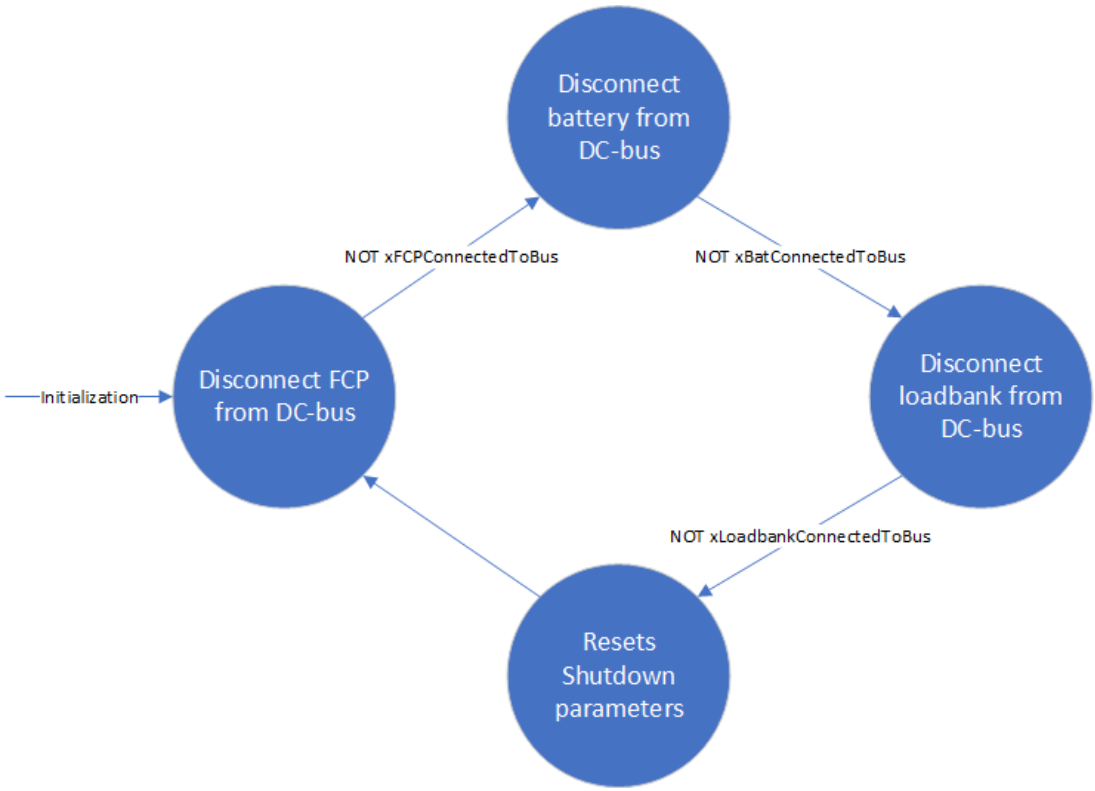
Overview of the system

The system consists of the fuel cell pack, a battery pack and a loadbank. They are all directly connected to a DC-bus. The PLC controlls the FCP and battery over Modbus-TCP. The loadbank is controlled over TCP-IP.

Finite state diagram of main system

The pMain-POU uses a finite state machine-like setup. Where there are 4 main states: Idle, connecting, testing and shutdown. The connecting and shutdown-sequence has several sub-states, where the different components are either connected or disconnected in order.


Initial State
xComAllOK
AND
xConnectAll
AND NOT
xReadyForTesting
xShutdown
Start up sequence
xReadyForTesting
Shutdown sequence
xReadyForTesting
xShutdown
Testing
FCP - Readycheck
xLoadbankConnectedToBus
xFCPReadyForTesting
xFCPConnectedToBus
AND
xBatConnectedToBus
AND
xLoadbankConnectedToBus
Loadbank Connect
Finishing start up, sets xReadyForTesting to TRUE
xFCPConnectedToBus
AND
xBatConnectedToBus
Initial State, Start up procedure.
xBatConnectedToBus
xFCPConnectedToBus
ELSE
Battery Connect
xFCPConnectedToBus
FCP-Connect



The FCP also has several states, which is different from the internal states of the FC modules.

Setting the correct IP-adresses

The IP-adresses are set in the function blocks that controls the communication. All IP-adresses are shown in the HMI-screen for the relevant component.

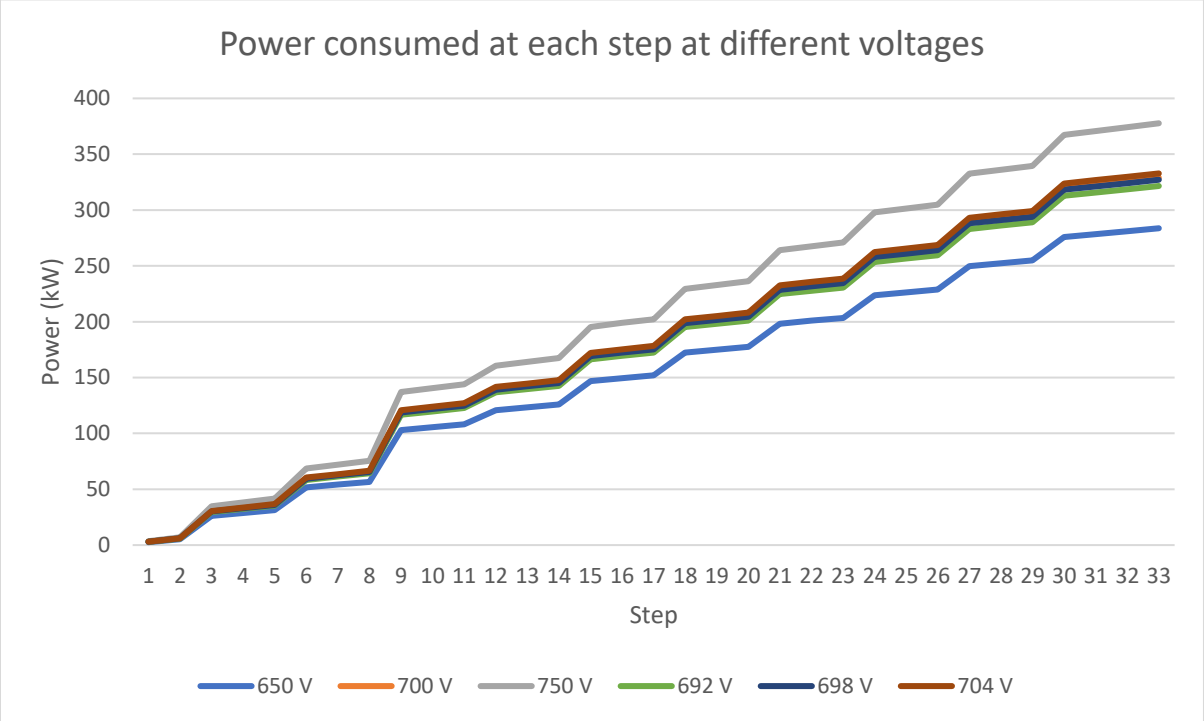
For the FCP, one would need to set the IP-adress in the ClientData.strServer_IP-variable in the fbFCP function block. Default is: 192.168.3.10

For the battery, one would need to set the IP-adress in the ClientData.strServer_IP -variable in the fbBattery function block. Default is: 192.168.3.22

For the loadbank, on would need to set the IP-adress in the strLoadbank_dest_ip -variable in the fbLoadbank function block. Default is: 192.168.3.21

Regulation of the system

The loadbank in the system is controlled in several steps, the estimated power at the different steps at different voltages is shown in the graph below and in the table in the appendix:



It is possible to manually choose the steps for power regulation. If the system is regulating itself, then the system will either be in a discharge loop or charging loop. If the system is in a discharge loop, the loadbank will try to load more power than the FCP is set to produce. As the actual power is a bit uncertain, an offset value has been added. This can be set in PLCnext and is set to 3kW as default. The offset value adjusts how much power the loadbank will be asked to load.

So, if the FCP produced 150kW, and the offset is at 3kW, and the battery is in discharge loop, then the power request to the loadbank will be “more than 153kW”. If the battery is in a charging loop and the offset for charging is 3kW, the request for the loadbank would be “less than 147kW”.

The HMI-panel

The HMI consists of an overview, where the most relevant data is shown, each component has its own screen where their most used functions are. The battery screen has its own subscreen for alarms where one can reset the alarms.

The FCP has its own subscreen where it is possible to program a cycle, where it is possible to use autorun.

Using autorun

In the HMI-panel of the FCP-subscreen, it is possible to set the power of each module and the duration of the sequence. The subscreen will show the current setting and the next 4 settings. When autorun is checked, it will start at the first instance, and run untill a sequence with 0 duration appears. Then it will set all power values to 0 and exit automatic mode. The program will continue to run in manual mode untill shut down or automatic mode is checked again.

The autorun has 256 sequences, the last one (nr. 255) must be set to 0 duration to avoid unknown consequenses.

The timer is based on an RTC-module and used seconds after midnight as reference. If the autorun is used around midnight, the program will not be able to complete the current sequence. If the RTC module has the wrong time, the program might not complete the current sequence.

Further work

Modbus server

The Modbus server was not implemented due to lack of time. The easiest way to implement it would be to link the different variables used by the HMI to the Modbus registers. Care must be taken so that a Modbus register is reset after being read from by the server. As several buttons are automatically reset by the program.

As an example, the “Reset alarms”-button for the battery is connected to the “pMain1.fbBattery1.xResetAlarms” variable. When the button is switched, the variable is set to true, when the alarms are reset, the variable is reset to false and shows up as unpressed in the HMI. If a Modbus server simply wrote the alarm reset register value to the alarm reset variable, the alarm reset variable would never reset. Below is a table of command variables that should be implemented.

Button/input	**Connected variable**
Connect to DC-bus (battery)	pMain1.xBatConnectToBus
Reset alarms (battery)	pMain1.fbBattery1.xResetAlarms
Select (Module 1-4) (FCP)	pmain1.xConnectModule[n] n : 0-3
Power set (FCP)	pMain1.uiPowerHMI[n] n: 0-3
Start up (FCP)	pMain1.xConnectAll
Apply new power request (FCP)	pMain1.xNewPowerValues
Shut down (FCP)	pMain1.xShutdown
Automatic mode	pMain1.xFCPManual
Automatic sequence duration	pMain1.udtTimeAndPowerTable.uliSeqDur[m] m: 0-255
Automatic power request	pMain1.udtTimeAndPowerTable.uiPower[n][m] n: 0-3
Manual mode (loadbank)	pMain1.fbLoadbank1.xManual
Connect Loadbank (loadbank)	pMain1.xLoadbankConnectToBus
Step input (loadbank)	pMain1.fbLoadbank1.uiStep

Unwanted consequenses

If the system looses power, the program should get the connection data from the battery and loadbank, because the connection variable is read from them. The FCP connection data is evaluated in the fbFCP function block. This evaluation happens during the start up sequence. And before the start up sequence can start, the battery and the loadbank must be disconnected. Thus, if the system is up and running and the PLC reboots, a reconnect command will shut down the battery and loadbank before trying to connect the FCP.

The connection should be evaluated again during startup using register 10n9.

Alarm log

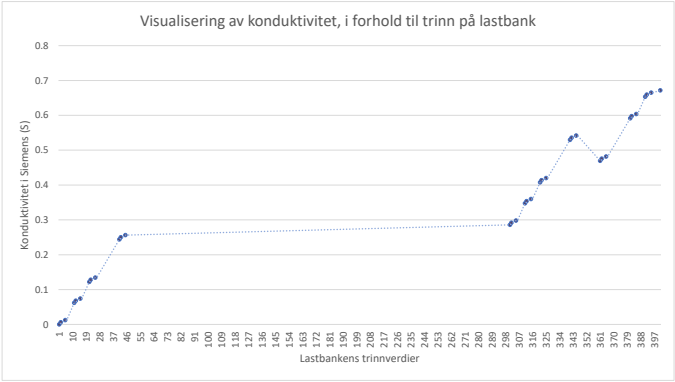
An alarm log for all types of alarms should be implemented.

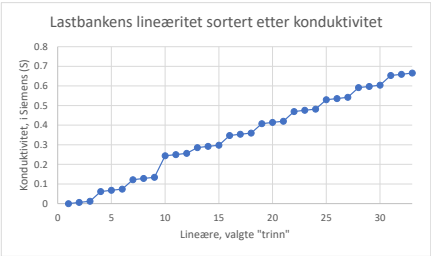
Appendix

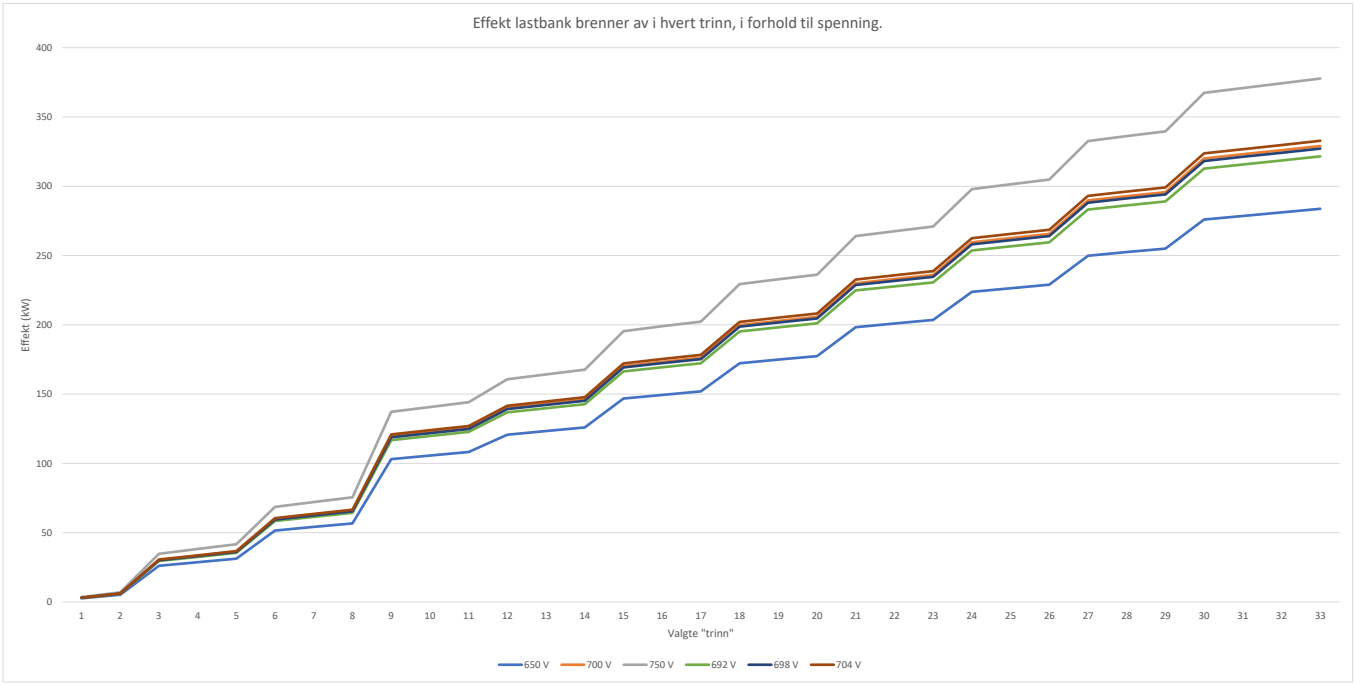
Step	P = G*U/10^3		Voltages (V)		
			695	700	705
	Power value	Conduct-ance (S)	Poew (kW) @ 695V	Power (kW) @ 700V	Power (kW) @ 705V
0	0	0	0	0	0
1	1	0.0062	3	3	3
2	4	0.0123	6	6	6
3	10	0.0617	30	30	31
4	11	0.0679	33	33	34
5	14	0.0740	36	36	37
6	20	0.1220	59	60	61
7	21	0.1282	62	63	64
8	24	0.1342	65	66	67
9	40	0.2439	118	120	121
10	41	0.2501	121	123	124
11	44	0.2562	124	126	127
12	300	0.2857	138	140	142
13	301	0.2919	141	143	145
14	304	0.2980	144	146	148
15	310	0.3474	168	170	173
16	311	0.3537	171	173	176
17	314	0.3597	174	176	179
18	320	0.4077	197	200	203
19	321	0.4139	200	203	206
20	324	0.4199	203	206	209
21	360	0.4694	227	230	233
22	361	0.4756	230	233	236
23	364	0.4816	233	236	239
24	340	0.5296	256	260	263
25	341	0.5358	259	263	266
26	344	0.5419	262	266	269
27	380	0.5913	286	290	294
28	381	0.5976	289	293	297
29	384	0.6036	292	296	300
30	390	0.6531	315	320	325
31	391	0.6593	318	323	328
32	394	0.6653	321	326	331
33	400	0.6715	324	329	334

Vedlegg D - Lastbanktrinn

Valgte "trinn"	Formel: S*U/10^3		Spenninger i testområdet (Volt, U)		
			692	698	704
	Trinn på lastbank	Konduktivitet i Siemens (S)	Effekt (kW) i trinnet, ved 692 V	Effekt (kW) i trinnet, ved 698 V	Effekt (kW) i trinnet, ved 704 V
0	0	0	0	0	0
1	1	0.0062	3	3	3
2	4	0.0123	6	6	6
3	10	0.0617	30	30	31
4	11	0.0679	33	33	34
5	14	0.074	35	36	37
6	20	0.122	58	59	60
7	21	0.1282	61	62	64
8	24	0.1342	64	65	67
9	40	0.2439	117	119	121
10	41	0.2501	120	122	124
11	44	0.2562	123	125	127
12	300	0.2857	137	139	142
13	301	0.2919	140	142	145
14	304	0.298	143	145	148
15	310	0.3474	166	169	172
16	311	0.3537	169	172	175
17	314	0.3597	172	175	178
18	320	0.4077	195	199	202
19	321	0.4139	198	202	205
20	324	0.4199	201	205	208
21	360	0.4694	225	229	233
22	361	0.4756	228	232	236
23	364	0.4816	231	235	239
24	340	0.5296	254	258	262
25	341	0.5358	257	261	266
26	344	0.5419	259	264	269
27	380	0.5913	283	288	293
28	381	0.5976	286	291	296
29	384	0.6036	289	294	299
30	390	0.6531	313	318	324
31	391	0.6593	316	321	327
32	394	0.6653	319	324	330
33	400	0.6715	322	327	333






Effekt lastbank brenner av i hvert trinn, i forhold til spenning.
Effekt (kW)
400
350
300
250
200
150
100
50
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
Valgte "trinn"
650 V
700 V
750 V
692 V
698 V
704 V

Vedlegg E1 - Kode for hovedprogram

FCP

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
fbFCP1	fbFCP	Local		Function block for FCP							
xFCPConnectedToBus	BOOL	Local		Indicates connection from FCP to DC-bus							
xFCPComOK	BOOL	Local		Communication with FCP is OK							
xFCPConnectToBus	BOOL	Local		Command to connect FCP to DC-bus							
iModuleStatus	IntArray3	Local		Aray with status of the different modules							
uiPowerHMI	UintArray3	Local		Array with the power values of the HMI							
uiPowerTotalSet	UINT	Local		Total power set for all modules							
uiPowerSet	UintArray3	Local		Array with the power set of the different modules							
uiMaxModulePower	UINTArray3	Local		Array with the maximum power of each module.	[4 (UINT# 80)]						
xFCPManual	BOOL	Local		Sets the FCP to run in manual mode	TRUE						
xNewPowerValues	BOOL	Local		Indicates new power values from HMI							
xConnectModule	BoolArray3	Local		Indicates which modules that should be connected to DC-bus							
xFCPReadyForTesting	BOOL	Local		Indicates that the FCP is ready for testing							
uiFCPPowerMeasured	UINT	Local		FCP power in hW.							
iModule	INT	Local		Used to specify module i FOR-loops							
xShutdownFCP	BOOL	Local		Command to shut down FCP							
uiModulePowerMeasured	UintArray3	Local		Array with the powered measured by each module							
uiFCPVoltage	UINT	Local		Voltage measured by FCP over DC-bus							

Control limits

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
uiSOCUpperAlarm	UINT	Local		Alarm limit, SOC over this will raise an alarm	UINT#55						
uiSOCUpperLimit	UINT	Local		Control limit, SOC over this will set the system in discharge mode	UINT#45						
uiSOCLowerAlarm	UINT	Local		Alarm limit, SOC under this will raise an alarm	UINT#20						
uiSOCLowerLimit	UINT	Local		Control limit, SOC under this will set the system in charging mode	UINT#25						
uiSOCMidLimit	UINT	Local		SOC over this limit will initialize the program in discharge mode, and vice versa	UINT#35						
xCharging	BOOL	Local		Indicates that the program is in charging mode							
xBatAlarmSOC	INT	Local		Indicates SOC status							

BAT

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
fbBattery1	fbBattery	Local		Function block for battery							
xBatConnectedToBus	BOOL	Local		Battery is connected to DC-Bus							
uiBatSOC	UINT	Local		State of Charge for battery, in %							
xBatComOK	BOOL	Local		Battery communcation status							
xBatConnectToBus	BOOL	Local		Attempting to connect battery to DC-bus							
iBatBusCurrent	INT	Local		Battery current, A negative value indicates packs discharging, and a positive value indicates packs charging.							
uiBatBusVoltage	UINT	Local		Battery voltage on DC-bus							

Loadbank

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
fbLoadbank1	fbLoadbank	Local		Function block for loadbank							
fbLoadbankSim1	fbLoadbankSim	Local		Function block for loadbank sim, used for testing program							
xLBComOK	BOOL	Local		Communication with loadbank is OK							
uiLoadbankPowerRequest	UINT	Local		Power to be burned off by loadbank							
xLoadbankConnectedToBus	BOOL	Local		Indicates that the loadbank is connected to DC-bus							
xLoadbankConnectToBus	BOOL	Local		Command to connect the loadbank to the DC-bus							
iLoadBankAlarms	INT	Local		Indicates alarms in loadbank specified in the text list AlarmsLoadbank							
uiLoadBankLoaded	UINT	Local		The amount of power that the loadbank is trying to load							
uiLoadBankMeasured	UINT	Local		Measured power from loadbank							
xLoadbankSendNow	BOOL	Local		Overwrite in case there is an urgent need for fast switching.							
iLoadbankVoltage	INT	Local		Measured voltage across the loadbank							
uiOffsetCharge	UINT	Local		How much less power the loadbank will operate around during charging	UINT#3						
uiOffsetDischarge	UINT	Local		How much more power the loadbank will operate around during discharging	UINT#3						

Autorun variables

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
RTC	RTC_TYPE	External		Real Time Clock, used to time sequences							
usiArrayLocationHMI	USINT	Local		Indicates the location of the first column in the autorun program HMI overview.							
usiArrayLocationHMI2	USINT	Local		Indicates the location of the second column in the autorun program HMI overview.							
usiArrayLocationHMI3	USINT	Local		Indicates the location of the third column in the autorun program HMI overview.							
usiArrayLocationHMI4	USINT	Local		Indicates the location of the fourth column in the autorun program HMI overview.							
usiArrayLocationHMI5	USINT	Local		Indicates the location of the fifth column in the autorun program HMI overview.							
udtTimeAndPowerTable	udtFCAutoprogram	Local		Array of sequence durations and power settings.							
usiCurrentArrayLocation	USINT	Local		Current location for editing array in HMI							
uliTime	ULINT	Local		Seconds since new day							
uliSeqStartTime	ULINT	Local		Start time of the current sequence.							

Default

Name	Type	Usage	Translate	Comment	Init	Retain	Constant	OPC	HMI	Proficloud	I/Q
xComAllOK	BOOL	Local		Indicates that the communication with all componets is OK							
xReadyForTesting	BOOL	Local		Indicates that all components are connected to the DC-bus.							
xShutdown	BOOL	Local		Command to shut down all components in order.							
iState	INT	Local		Indicates current state of program, used with the text list StatesMain							
xConnectAll	BOOL	Local		Command to connect all modules							
iConnectionState	INT	Local		Current state in the finite state machine for connection							
iShutdownState	INT	Local		Current state in the finite state machine for shutdown							

Components / pMain / Initiation_Communication

```
(*
This code sheet inititates testing procedures and communicates with the different packs
*)


(*Running battery, FCP, GC and LB function blocks*)
fbBattery1(uiSOC                    => uiBatSOC,
           xComOK                   => xBatComOK,
           xConnectedToBus          => xBatConnectedToBus,
           uiBusVoltage             => uiBatBusVoltage,
           iBusCurrent              => iBatBusCurrent,
           xConnectToBus            := xBatConnectToBus);

fbFCP1(uiActualPower                => uiFCPPowerMeasured,
       iModuleStatus                => iModuleStatus,
       xComOK                       => xFCPComOK,
       xConnectedToBus              => xFCPConnectedToBus,
       xIsReadyForTesting           => xFCPReadyForTesting,
       uiModulePowerMeasured        => uiModulePowerMeasured,
       uiBusVoltage                 => uiFCPVoltage,
       uiModulePowerSet             := uiPowerSet,
       xConnectModule               := xConnectModule,
       xConnectToBus                := xFCPConnectToBus,
       xShutdown                    := xShutdownFCP,
       xBatteryConnected            := xBatConnectedToBus);

fbLoadbank1(uiPowerLoaded           => uiLoadBankLoaded,
            uiPowerMeasured         => uiLoadBankMeasured,
            iAlarmLoadbank          => iLoadBankAlarms,
            xLoadbank_Loadstate     => xLoadbankConnectedToBus,
            xComOK                  => xLBComOK,
            iLoadbank_Voltage_in    => iLoadbankVoltage,
            xCharging               := xCharging,
            uiLoadpower             := uiLoadbankPowerRequest,
            xLoad                   := xLoadbankConnectToBus,
            xSendNow                := xLoadbankSendNow);

(*
fbLoadbankSim1(xCharging            := xCharging,
               uiLoadpower          := uiLoadbankPowerRequest,
               xLoad                := xLoadbankConnectToBus,
               xSendNow             := xLoadbankSendNow,
               uiPowerLoaded        => uiLoadBankLoaded,
               uiPowerMeasured      => uiLoadBankMeasured,
               iAlarmLoadbank       => iLoadBankAlarms,
               xLoadbank_Loadstate  => xLoadbankConnectedToBus,
               xComOK               => xLBComOK);
*)


(*Checking if communication is ok*)
IF xBatComOK AND xFCPComOK AND xLBComOK THEN
    xComAllOK := TRUE;
ELSE
    xComAllOK := FALSE;
END_IF;


(*Used for automatic power sequence*)
uliTime := TO_ULINT(RTC.SECONDS) + TO_ULINT(RTC.MINUTES*60) + TO_ULINT(RTC.HOURS*3600);
usiArrayLocationHMI2 := usiArrayLocationHMI + TO_USINT(1);
usiArrayLocationHMI3 := usiArrayLocationHMI + TO_USINT(2);
usiArrayLocationHMI4 := usiArrayLocationHMI + TO_USINT(3);
usiArrayLocationHMI5 := usiArrayLocationHMI + TO_USINT(4);

```

```
(*
This code sheet connects the FCP, battery and loadbank in the correct order before testing.
*)


IF xComAllOK AND xConnectAll AND NOT xReadyForTesting AND NOT xShutdown THEN

CASE iConnectionState OF
    0   : (*This is the initial state and determines the correct state if some connections are established. This will also
    disconnect the loadbank and battery if it is connected before the FCP.*)
        IF NOT xFCPConnectedToBus AND xBatConnectedToBus THEN
            xBatConnectToBus := FALSE;
        END_IF;

        IF NOT xFCPConnectedToBus AND xLoadbankConnectedToBus THEN
            xLoadbankConnectToBus := FALSE;
        END_IF;

        IF NOT xFCPConnectedToBus AND NOT xBatConnectedToBus AND NOT xLoadbankConnectedToBus THEN
            iConnectionState := 1;
        ELSIF xFCPConnectedToBus AND NOT xBatConnectedToBus AND NOT xLoadbankConnectedToBus THEN
            iConnectionState := 2;
        ELSIF xFCPConnectedToBus AND xBatConnectedToBus AND NOT xLoadbankConnectedToBus THEN
            iConnectionState := 3;
        ELSIF xFCPConnectedToBus AND xBatConnectedToBus AND xLoadbankConnectedToBus THEN
            iConnectionState := 4;
        END_IF;

    1   : (*This connects the FCP to the DC-bus*)
        IF NOT xFCPConnectedToBus THEN
            xFCPConnectToBus := TRUE;
        ELSE
            iConnectionState := 2;
        END_IF;

    2   : (*This connects the battery to the DC-bus*)
        IF NOT xBatConnectedToBus THEN
            xBatConnectToBus := TRUE;
        ELSE
            iConnectionState := 3;
        END_IF;

    3   : (*This connects the loadbank to the DC-bus*)
        IF NOT xLoadbankConnectedToBus THEN
            xLoadbankConnectToBus := TRUE;
            uiLoadbankPowerRequest := 0;
        ELSE
            iConnectionState := 4;
        END_IF;

    4   : (*All connected, readying for testing*)
        IF xFCPReadyForTesting THEN
            iConnectionState := 5;
        END_IF;

    5   :
        (*Evaluates initial battery state*)
        IF  uiBatSOC > uiSOCMidLimit THEN
            xCharging := FALSE;
        ELSE
            xCharging := TRUE;
        END_IF;

        xReadyForTesting := TRUE;
        iConnectionState := 0;
    ELSE

END_CASE

iState := 10 + iConnectionState;

ELSIF NOT xConnectAll AND NOT xReadyForTesting AND NOT xShutdown THEN
    iState := 0;

END_IF;

```

77
78
79

```
(*
This code sheet regulates the testing phase.
*)

IF xReadyForTesting THEN
    iState := 20;
    (*Sets new power levels for the FCP if needed.*)
    IF xFCPManual and xNewPowerValues THEN

        (*Limits power in case of too high values from operator, and calculates total power in case of manual mode*)
        FOR iModule := 0 TO 3 BY 1 DO
            IF xConnectModule[iModule] THEN
                uiPowerSet[iModule] := MIN(uiPowerHMI[iModule], uiMaxModulePower[iModule]);
                uiPowerHMI[iModule] := uiPowerSet[iModule];
            END_IF
        END_FOR;

        uiPowerTotalSet := uiPowerSet[0] + uiPowerSet[1] + uiPowerSet[2] + uiPowerSet[3];

        xNewPowerValues := FALSE;

    ELSIF NOT xFCPManual THEN  //Automatic power control
        IF usiCurrentArrayLocation  = 0 AND NOT (uliSeqStartTime = 0) THEN //First run
            uliSeqStartTime := uliTime; //Sets starting time for period
            FOR iModule := 0 TO 3 BY 1 DO
                IF xConnectModule[iModule] THEN
                    uiPowerSet[iModule] := MIN(udtTimeAndPowerTable.uiPower[iModule][usiCurrentArrayLocation], uiMaxModulePower
                    [iModule]);
                    uiPowerHMI[iModule] := uiPowerSet[iModule];
                END_IF;
            END_FOR;

        END_IF;

        IF uliSeqStartTime + udtTimeAndPowerTable.uliSeqDur[usiCurrentArrayLocation] > uliTime THEN //Starts new sequence
            usiCurrentArrayLocation := usiCurrentArrayLocation + TO_USINT(1);

            IF udtTimeAndPowerTable.uliSeqDur[usiCurrentArrayLocation] = 0 THEN //Ends automatic power control
                FOR iModule := 0 TO 3 BY 1 DO
                    IF xConnectModule[iModule] THEN
                        uiPowerSet[iModule] := 0;
                        uiPowerHMI[iModule] := uiPowerSet[iModule];
                    END_IF;
                END_FOR;

                xFCPManual              := TRUE;
                usiCurrentArrayLocation := 0;
                uliSeqStartTime         := 0; //Indicates unstarted automatic power control

            ELSE //Sets new power loads
                FOR iModule := 0 TO 3 BY 1 DO
                    IF xConnectModule[iModule] THEN
                        uiPowerSet[iModule] := MIN(udtTimeAndPowerTable.uiPower[iModule][usiCurrentArrayLocation], uiMaxModulePower
                        [iModule]);
                        uiPowerHMI[iModule] := uiPowerSet[iModule];
                    END_IF;
                END_FOR;

                uliSeqStartTime  := uliTime;
            END_IF;
        END_IF;

        uiPowerTotalSet := uiPowerSet[0] + uiPowerSet[1] + uiPowerSet[2] + uiPowerSet[3];

    END_IF;

    (*Evaluates whether the battery is to charge or discharge*)
    IF uiBatSOC > uiSOCUpperAlarm THEN
        xBatAlarmSOC   := 1;
        xCharging      := FALSE;

    ELSIF uiBatSOC > uiSOCUpperLimit THEN
        xCharging      := FALSE;

    ELSIF uiBatSOC > uiSOCLowerLimit THEN
        xBatAlarmSOC   := 0;
        //Resets alarms and nothing else, within bounds
```

```
76
77      ELSIF uiBatSOC > uiSOCLowerAlarm THEN
78          xCharging       := TRUE;
79
80      ELSE
81          xCharging       := TRUE;
82          xBatAlarmSOC    := 2;
83
84      END_IF;
85
86      (*Sets the loadbank power request*)
87      IF xCharging THEN
88          uiLoadbankPowerRequest := uiPowerTotalSet - uiOffsetCharge;
89      ELSE
90          uiLoadbankPowerRequest := uiPowerTotalSet + uiOffsetDischarge;
91      END_IF;
92
93
94  END_IF;
```

Components / pMain / Shutdown

```
(*
This code sheet conducts the correct shutdown procedure
*)

IF xShutdown THEN
    CASE iShutdownState OF
        0   : (*This is the initial state and signals the FCP to stop and disconnect*)
            xReadyForTesting    := FALSE;
            xConnectAll         := FALSE;
            xFCPConnectToBus    := FALSE;
            xShutdownFCP        := TRUE;
            iConnectionState    := 0;

            FOR iModule := 0 TO 3 BY 1 DO
                uiPowerSet[iModule] := 0;
                uiPowerHMI[iModule] := 0;
            END_FOR;
            uiLoadbankPowerRequest := 0;
            fbLoadbank1.uiStep := 1;
            fbLoadbank1.xManual := TRUE;

            IF NOT xFCPConnectedToBus THEN
                iShutdownState := 1;
            END_IF;

        1   : (*Disconnects battery from DC-bus*)
            xBatConnectToBus := FALSE;

            IF NOT xBatConnectedToBus THEN
                iShutdownState := 2;
            END_IF;

        2   : (*Disconnects loadbank from DC-bus*)
            IF uiBatBusVoltage < 2 AND iLoadbankVoltage < 2 AND uiFCPVoltage < 2 THEN
                xLoadbankConnectToBus := FALSE;
                fbLoadbank1.uiStep := 0;
                fbLoadbank1.xManual := FALSE;
            END_IF;

            IF NOT xLoadbankConnectedToBus THEN
                iShutdownState := 3;
            END_IF;

        3   : (*All disconnected*)
            xShutdown       := FALSE;
            xShutdownFCP    := FALSE;
            iShutdownState  := 0;
    ELSE

END_CASE

iState := 30 + iShutdownState;

END_IF;
```

Vedlegg E2 - Kode for FCP og støttefunksjoner

Default

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
Input	MB_TCP_ARR_W_1_125	Input										
uiHeartbeat	UINT	Output										
uiPackPower	UINT	Output										
uiBusVoltage	UINT	Output										
uiBusCurrent	UINT	Output										
iModuleStatus	IntArray3	Output										
uiModulePower	UintArray3	Output										
xReadyForBatteryConnect	BOOL	Output										

```
(*
This function block translates word-variables to boolean system variables, making it easier to understand during system
programming.
*)


(*Register 1000 - Heartbeat*)
uiHeartbeat           := TO_UINT(Input[1]);

(*Register 1001-1003 - Electric status*)
uiPackPower           := TO_UINT(Input[2]);
uiBusVoltage          := TO_UINT(Input[3]);
uiBusCurrent          := TO_UINT(Input[4]);

(*Register 1009*)
xReadyForBatteryConnect := Input[10].%X0;

(*Register 1018-1019 - Module 1 power and status*)
uiModulePower[0]        := TO_UINT(Input[19]);
iModuleStatus[0]        := TO_INT(Input[20]);

(*Register 1028-1029 - Module 2 power and status*)
uiModulePower[1]        := TO_UINT(Input[29]);
iModuleStatus[1]        := TO_INT(Input[30]);

(*Register 1038-1039 - Module 3 power and status*)
uiModulePower[2]        := TO_UINT(Input[39]);
iModuleStatus[2]        := TO_INT(Input[40]);

(*Register 1048-1049 - Module 4 power and status*)
uiModulePower[3]        := TO_UINT(Input[49]);
iModuleStatus[3]        := TO_INT(Input[50]);
```

```
(*
This function block controls the FCP and handles everyting related to the FCP.
*)

(*Communication with modbus server*)
MB_Client(xActivate             := ClientData.xActivate,
          xAcknowledge          := ClientData.xAcknowledge,
          xAutoAck              := ClientData.xAutoAck,
          strServer_IP          := ClientData.strServer_IP,
          iPort                 := ClientData.iPort,
          strBindIp             := ClientData.strBindIp,
          uiBindPort            := ClientData.uiBindPort,
          xUDP_Mode             := ClientData.xUDP_Mode,
          tReconnectDelay       := ClientData.tReconnectDelay,
          tTimeout              := ClientData.tTimeout,
          xActive               => ClientData.xActive,
          xReady                => ClientData.xReady,
          xError                => ClientData.xError,
          wDiagCode             => ClientData.wDiagCode,
          wAddDiagCode          => ClientData.wAddDiagCode,
          udtDiag               => ClientData.udtDiag,
          udtTCP_ComData        := ComData);


(*Checks if communication with FCP is OK*)
fbComCheckFCP(ClientData := ClientData, xComOK => xComOK);


(*Updates variables*)

fbReadRegisters1(ComData                    :=  ComData,
                 udtFC3Parameters           :=  udtFC3ParaFCP);

IF fbReadRegisters1.xNewData THEN
    fbDataConvertFCP1(Input                 := udtFC3ParaFCP.arrRegisterValue,
                      uiHeartbeat           => uiHeartbeat,
                      uiPackPower           => uiActualPower,
                      uiBusVoltage          => uiBusVoltage,
                      uiBusCurrent          => uiBusCurrent,
                      iModuleStatus         => iModuleStatus,
                      uiModulePower         => uiModulePowerMeasured,
                      xReadyForBatteryConnect => xReadyForBatteryConnect);
END_IF;



CASE iState OF
    0 : (*Initial state, evaluates correct state*)
        IF xShutdown THEN
            iState := 5;
        ELSIF xConnectedToBus AND xIsReadyForTesting THEN
            iState := 4;
        ELSIF xConnectedToBus THEN
            iState := 3;
        ELSIF xConnectToBus THEN
            iState := 2;
        ELSE
            iState := 1;
        END_IF;

    1 : (*Idle state, disconnected*)
        IF xConnectToBus THEN
            iState := 2;
        END_IF;

        IF xShutdown THEN
            iState := 5;
        END_IF;


    2 : (*Connecting to bus*)
        FOR iModule := 0 TO 3 BY 1 DO
        IF NOT xModuleNotActive[iModule] AND NOT xModuleConnected[iModule] THEN

            IF xConnectModule[iModule] AND NOT xModuleConnecting[iModule] THEN

                udtFC16ParaModule[iModule].arrRegisterValue[0].0 := TRUE; //Register 10n1.0
```

```
78                  udtFC16ParaModule[iModule].arrRegisterValue[1].%X1 := TRUE; //Register 10n1.1
79                  udtFC16ParaModule[iModule].arrRegisterValue[2] := TO_WORD(0); //Register 10n2, power request
80
81                  CASE iModule OF //Uses the correct function block for the module.
82                      0 :
83                          fbWriteToRegisters1(uiHeartbeat   := uiHeartbeat,
84                                              xWriteSuccess       => xWriteSuccess,
85                                              ComData             := ComData,
86                                              udtFC16Parameters   := udtFC16ParaModule[iModule]);
87                      1 :
88                          fbWriteToRegisters2(uiHeartbeat   := uiHeartbeat,
89                                              xWriteSuccess       => xWriteSuccess,
90                                              ComData             := ComData,
91                                              udtFC16Parameters   := udtFC16ParaModule[iModule]);
92                      2 :
93                          fbWriteToRegisters3(uiHeartbeat   := uiHeartbeat,
94                                              xWriteSuccess       => xWriteSuccess,
95                                              ComData             := ComData,
96                                              udtFC16Parameters   := udtFC16ParaModule[iModule]);
97                      3 :
98                          fbWriteToRegisters4(uiHeartbeat   := uiHeartbeat,
99                                              xWriteSuccess       => xWriteSuccess,
100                                             ComData             := ComData,
101                                             udtFC16Parameters   := udtFC16ParaModule[iModule]);
102                 END_CASE;
103
104                 IF xWriteSuccess THEN
105                     uiModuleLastWriteHeartbeat[iModule]     := uiHeartbeat;
106                     xModuleConnecting[iModule]              := TRUE;
107                 END_IF;
108
109             ELSIF NOT xConnectModule[iModule] THEN
110                 xModuleNotActive[iModule]       := TRUE;
111                 xModuleReady[iModule]           := TRUE;
112                 xModuleReadyForTesting[iModule] := TRUE;
113
114             ELSIF xModuleConnecting[iModule] AND xReadyForBatteryConnect THEN
115                 xModuleConnected[iModule]       := TRUE;
116                 xModuleReady[iModule]           := TRUE;
117
118             END_IF;
119         END_IF;
120
121     END_FOR
122
123     IF xModuleReady[0] AND xModuleReady[1] AND xModuleReady[2] AND xModuleReady[3] THEN
124         xConnectedToBus := TRUE;
125         iState          := 3;
126     END_IF;
127
128     IF xShutdown THEN
129         iState := 5;
130     END_IF;
131
132
133     3 : (*Connected to bus*)
134         FOR iModule := 0 TO 3 BY 1 DO
135
136             IF NOT xModuleNotActive[iModule] AND (iModuleStatus[iModule] = 2) AND xBatteryConnected THEN
137                 xModuleReadyForTesting[iModule] := TRUE;
138             END_IF;
139
140         END_FOR;
141
142         IF xModuleReadyForTesting[0] AND xModuleReadyForTesting[1] AND xModuleReadyForTesting[2] AND xModuleReadyForTesting[3]
            THEN
143             xIsReadyForTesting := TRUE;
144             iState := 4;
145         END_IF;
146
147     IF xShutdown THEN
148         iState := 5;
149     END_IF;
150
151
152     4 : (*Running*)
153         FOR iModule := 0 TO 3 BY 1 DO
```

```
154                 IF xModuleConnected[iModule] THEN
155
156                     udtFC16ParaModule[iModule].arrRegisterValue[1].%X0 := TRUE;
157                     udtFC16ParaModule[iModule].arrRegisterValue[1].%X1 := TRUE;
158                     udtFC16ParaModule[iModule].arrRegisterValue[2] := TO_WORD(uiModulePowerSet[iModule]);
159
160                     CASE iModule OF //Uses the correct function block for the module.
161                         0 :
162                             fbWriteToRegisters1(uiHeartbeat   := uiHeartbeat,
163                                                 xWriteSuccess      => xWriteSuccess,
164                                                 ComData            := ComData,
165                                                 udtFC16Parameters  := udtFC16ParaModule[iModule]);
166                         1 :
167                             fbWriteToRegisters2(uiHeartbeat   := uiHeartbeat,
168                                                 xWriteSuccess      => xWriteSuccess,
169                                                 ComData            := ComData,
170                                                 udtFC16Parameters  := udtFC16ParaModule[iModule]);
171                         2 :
172                             fbWriteToRegisters3(uiHeartbeat   := uiHeartbeat,
173                                                 xWriteSuccess      => xWriteSuccess,
174                                                 ComData            := ComData,
175                                                 udtFC16Parameters  := udtFC16ParaModule[iModule]);
176                         3 :
177                             fbWriteToRegisters4(uiHeartbeat   := uiHeartbeat,
178                                                 xWriteSuccess      => xWriteSuccess,
179                                                 ComData            := ComData,
180                                                 udtFC16Parameters  := udtFC16ParaModule[iModule]);
181                     END_CASE;
182
183                 IF xWriteSuccess THEN
184                     uiModuleLastPower[iModule] := uiModulePowerSet[iModule];
185                 END_IF;
186
187             END_IF;
188
189         END_FOR;
190
191         IF xShutdown THEN
192             iState := 5;
193         END_IF;
194
195
196         5 : (*Shutting down*)
197             xIsReadyForTesting := FALSE;
198
199             FOR iModule := 0 TO 3 BY 1 DO
200                 IF xModuleConnected[iModule] AND NOT (uiActualPower = 0) THEN //Sends a power request for 0kW
201
202                     udtFC16ParaModule[iModule].arrRegisterValue[1].%X0  := TRUE;
203                     udtFC16ParaModule[iModule].arrRegisterValue[1].%X1  := TRUE;
204                     udtFC16ParaModule[iModule].arrRegisterValue[2]      := TO_WORD(0);
205
206                     CASE iModule OF //Uses the correct function block for the module.
207                         0 :
208                             fbWriteToRegisters1(uiHeartbeat   := uiHeartbeat,
209                                                 xWriteSuccess      => xWriteSuccess,
210                                                 ComData            := ComData,
211                                                 udtFC16Parameters  := udtFC16ParaModule[iModule]);
212                         1 :
213                             fbWriteToRegisters2(uiHeartbeat   := uiHeartbeat,
214                                                 xWriteSuccess      => xWriteSuccess,
215                                                 ComData            := ComData,
216                                                 udtFC16Parameters  := udtFC16ParaModule[iModule]);
217                         2 :
218                             fbWriteToRegisters3(uiHeartbeat   := uiHeartbeat,
219                                                 xWriteSuccess      => xWriteSuccess,
220                                                 ComData            := ComData,
221                                                 udtFC16Parameters  := udtFC16ParaModule[iModule]);
222                         3 :
223                             fbWriteToRegisters4(uiHeartbeat   := uiHeartbeat,
224                                                 xWriteSuccess      => xWriteSuccess,
225                                                 ComData            := ComData,
226                                                 udtFC16Parameters  := udtFC16ParaModule[iModule]);
227                     END_CASE;
228
229                     IF xWriteSuccess THEN
230                         uiModuleLastPower[iModule] := uiModulePowerSet[iModule];
```

```
231                 END_IF;
232
233             ELSIF uiActualPower = 0 THEN  //0 kW measured will cause the modules to recieve the stop command.
234
235                 udtFC16ParaModule[iModule].arrRegisterValue[1].%X0  := FALSE;
236                 udtFC16ParaModule[iModule].arrRegisterValue[1].%X1  := FALSE;
237                 udtFC16ParaModule[iModule].arrRegisterValue[2]      := TO_WORD(0);
238
239                 CASE iModule OF //Uses the correct function block for the module.
240                     0 :
241                         fbWriteToRegisters1(uiHeartbeat   := uiHeartbeat,
242                                             xWriteSuccess       => xWriteSuccess,
243                                             ComData             := ComData,
244                                             udtFC16Parameters   := udtFC16ParaModule[iModule]);
245                     1 :
246                         fbWriteToRegisters2(uiHeartbeat   := uiHeartbeat,
247                                             xWriteSuccess       => xWriteSuccess,
248                                             ComData             := ComData,
249                                             udtFC16Parameters   := udtFC16ParaModule[iModule]);
250                     2 :
251                         fbWriteToRegisters3(uiHeartbeat   := uiHeartbeat,
252                                             xWriteSuccess       => xWriteSuccess,
253                                             ComData             := ComData,
254                                             udtFC16Parameters   := udtFC16ParaModule[iModule]);
255                     3 :
256                         fbWriteToRegisters4(uiHeartbeat   := uiHeartbeat,
257                                             xWriteSuccess       => xWriteSuccess,
258                                             ComData             := ComData,
259                                             udtFC16Parameters   := udtFC16ParaModule[iModule]);
260                 END_CASE;
261                 IF fcModulesStopped(iModuleStatus) THEN
262
263                     xConnectedToBus := FALSE;
264                     FOR iModule := 0 TO 3 BY 1 DO //Resets variables and makes the function block ready for idle state.
265                         xModuleNotActive[iModule]           := FALSE;
266                         xModuleReady[iModule]               := FALSE;
267                         xModuleReadyForTesting[iModule]     := FALSE;
268                         xModuleConnecting[iModule]          := FALSE;
269                         xModuleConnected[iModule]           := FALSE;
270                     END_FOR;
271
272                     iState := 1;
273                 END_IF;
274             END_IF;
275         END_FOR;
276
277
278
279     (*In case of iState other than 0-5*)
280     ELSE
281         iState := 0;
282
283 END_CASE
284
285
```

Output

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
uiActualPower	UINT	Output		Pack power in hW.								
iModuleStatus	IntArray3	Output		Status for modules								
xComOK	BOOL	Output		Indicates communication with FCP is OK								
xConnectedToBus	BOOL	Output		Indicates that the FCP is connected to the DC-bus								
xIsReadyForTesting	BOOL	Output		Indicates that the FCP is ready for testing								
uiModulePowerMeasured	UintArray3	Output		Measured power from FCM, register 10n8								
uiBusVoltage	UINT	Output		Bus voltage in dV								

Input

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
uiModulePowerSet	UintArray3	Input		Commands the power to the modules								
xConnectModule	BoolArray3	Input		Indicates which modules that are to be connected.								
xConnectToBus	BOOL	Input		Command to connect the FCP to DC-bus								
xShutdown	BOOL	Input		Command to shut down the FCP and disconnect from DC-bus								
xBatteryConnected	BOOL	Input		Indicates that the battery is connected to the DC-bus								

Communication

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
ClientData	udtMB_TCP_CLIENT	Local		Settings for the client	(xAutoAck := TRUE, strServer_IP := STRING#'192.168.3.10', iPort := INT#502, strBindIp := STRING#'0.0.0.0', tReconnectDelay := TIME#500ms, tTimeout := TIME#2s)	Private						
ComData	MB_TCP_UDT_COMMUNICATION	Local		Internal communication data for Modbus function blocks		Private						
MB_Client	MB_TCP_Client_9	Local		Function block for Modbus client		Private						
fbComCheckFCP	fbComCheck	Local		Function block for communication check		Private						
udtFC3ParaFCP	udtMB_TCP_FC3	Local		Parameters for reading from FCP	(iMT_ID := INT#1, tUpdateTime := TIME#1s, wStartRegister := WORD#16#3E8, uiQuantityOfRegisters := UINT#50)	Private						
fbDataConvertFCP1	fbDataConvertFCP	Local		Function block for converting raw data.		Private						
udtFC16ParaModule	udtMB_TCP_FC16Array	Local		Parameters for writing to all 4 modules	[(iMT_ID := INT#2, tUpdateTime := TIME#1s, wStartRegister := WORD#16#3F3, uiQuantityOfRegisters := UINT#2), (iMT_ID := INT#3, tUpdateTime := TIME#1s, wStartRegister := WORD#16#3FD, uiQuantityOfRegisters := UINT#2), (iMT_ID := INT#4, tUpdateTime := TIME#1s, wStartRegister := WORD#16#407, uiQuantityOfRegisters := UINT#2), (iMT_ID := INT#5, tUpdateTime := TIME#1s, wStartRegister := WORD#16#411, uiQuantityOfRegisters := UINT#2)]	Private							

Local Pack variables

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
uiHeartbeat	UINT	Local		Hearbeat of FCP, in seconds		Private						
uiBusCurrent	UINT	Local		Bus current in dA		Private						

Local Module variables

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
uiModuleLastPower	UINTArray3	Local		Indicates last succesfull power request		Private						
uiModuleLastWriteHeartbeat	UINTArray3	Local		Indicates last hearbeat of a succesfull power request		Private						
xModuleConnecting	BoolArray3	Local		Indicates that modules are trying to connect.		Private						
xModuleNotActive	BoolArray3	Local		Indicates which modules are not active.		Private						
xModuleConnected	BoolArray3	Local		Indicates which modules are connected		Private						
xModuleReady	BoolArray3	Local		Indicates which modules are ready for battery connection (inactive are ready)		Private						
xModuleReadyForTesting	BoolArray3	Local		Indicates which modules are ready for FCP testing (inactive are ready)		Private						
iModule	INT	Local		Integer used to indicate which module it is processing in a FOR-loop.		Private						

Default

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
fbReadRegisters1	fbReadFromRegisters	Local		Function block to read from all registers		Private						
fbWriteToRegisters1	fbWriteToRegisters	Local		Function block to write info to module 1		Private						
fbWriteToRegisters2	fbWriteToRegisters	Local		Function block to write info to module 2		Private						
fbWriteToRegisters3	fbWriteToRegisters	Local		Function block to write info to module 3		Private						
fbWriteToRegisters4	fbWriteToRegisters	Local		Function block to write info to module 4		Private						
iState	INT	Local		Current state in the FCP finite state machine		Private						
xReadyForBatteryConnect	BOOL	Local		Indicates the FCP is ready for batteries to be connected.		Private						
xWriteSuccess	BOOL	Local		Indicates that the last write was recieved by the FCP, only valid for one cycle		Private						

Components / Functions / fcEqualArray / Signature

Access Specifier: Public

Return Type: BOOL

Default

Name	Type	Usage	Comment	Init	Constant
In1	MB_TCP_ARR_W_1_125	Input	First array to be evaluated		
In2	MB_TCP_ARR_W_1_125	Input	Second array to be evaluated		
n	INT	Local	Counter used by FOR-loop.		

Components / Functions / fcEqualArray / Code

```
1
2   (*
3   This function compares two arrays of the type MB_TCP_ARR_W_1_125, to see if they are equal or not
4   *)
5   FOR n := 1 TO 125 BY 1 DO
6       IF In1[n] <> In2[n] THEN
7           fcEqualArray := FALSE;
8           RETURN;
9       END_IF;
10  END_FOR
11
12  fcEqualArray := TRUE;
```

Components / Functions / fcModulesStopped / Signature

Access Specifier: Public

Return Type: BOOL

Default

Name	Type	Usage	Comment	Init	Constant
In1	IntArray3	Input	Module status		

Components / Functions / fcModulesStopped / Code

```
(*Checks if all modules are stopped*)

IF (In1[0] = 0 OR In1[0] = 4) AND (In1[1] = 0 OR In1[1] = 4) AND (In1[2] = 0 OR In1[2] = 4) AND (In1[3] = 0 OR In1[3] = 4) THEN
    fcModulesStopped := TRUE;
ELSE
    fcModulesStopped := FALSE;
END_IF;
```

Vedlegg E3 - Kode for batteri og støttefunksjoner

Components / Data conversion / fbDataConvertAlarms / Code

```
(*
This function block translates word-variables to boolean system variables, making it easier to understand during system
programming.
*)


(*Register 400 - Alarm status*)
udtAlarmsData.Pack1AlarmStatus := TO_INT(Input[1]);

(*Register 401 - Faults*)
//First byte
udtAlarmsData.xOverVoltFault            :=  Input[2].%X0;
udtAlarmsData.xUnderVoltFault           :=  Input[2].%X1;
udtAlarmsData.xOverTempFault            :=  Input[2].%X2;
udtAlarmsData.xHVILFault                :=  Input[2].%X3;
udtAlarmsData.xEStopFault               :=  Input[2].%X4;
udtAlarmsData.xSafetyShutdownFault      :=  Input[2].%X5;
udtAlarmsData.xPackGroundFault          :=  Input[2].%X6;
udtAlarmsData.xContactorFailFault       :=  Input[2].%X7;

//Second byte
udtAlarmsData.xLowTempOvercurrentFault  :=  Input[2].%X8;

(*Register 403 - Warnings*)
//First byte
udtAlarmsData.xOverVoltWarning              :=  Input[4].%X0;
udtAlarmsData.xUnderVoltWaring              :=  Input[4].%X1;
udtAlarmsData.xOverTempWarning              :=  Input[4].%X2;
udtAlarmsData.xComWarning                   :=  Input[4].%X3;
udtAlarmsData.xOverCurrentWarning           :=  Input[4].%X4;
udtAlarmsData.xInputPowerLossWarning        :=  Input[4].%X5;
udtAlarmsData.xPdmOverTemperatureWarning    :=  Input[4].%X6;

//Second Byte
udtAlarmsData.xFuseFailureWarning           :=  Input[4].%X8;


```

Default

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
Input	MB_TCP_ARR_W_1_125	Input		Raw data from Modbus								
udtAlarmsData	BAT_Alarms	InOut		Alarms data.								

Components / Data conversion / fbDataConvertArrayStatus / Code

```
(*
This function block translates word-variables to system variables, making it easier to understand during system programming.
*)


(*Register 0 to 3 *)
udtArrayData.NoPacksInArray                 := TO_UINT(Input[1]);
udtArrayData.NoPacksInNetwork               := TO_UINT(Input[2]);
udtArrayData.NoPacksConnected               := TO_UINT(Input[3]);
udtArrayData.NoPacksFaulted                 := TO_UINT(Input[4]);

udtArrayData.BusVoltage                     := TO_UINT(Input[5]);
udtArrayData.BusCurrent                     := TO_INT(Input[6]);
udtArrayData.ArrayOnlineSOC                 := TO_UINT(Input[7]);
udtArrayData.ArrayOnlineSOH                 := TO_UINT(Input[8]);

//Register 8 is reserved

//Register 9 to 16
udtArrayData.ArrayChargeCurrentLimit        := TO_UINT(Input[10]);
udtArrayData.ArrayDischargeCurrentLimit     := TO_UINT(Input[11]);
udtArrayData.MaxCellTemperature             := TO_INT(Input[12]);
udtArrayData.MinCellTemperature             := TO_INT(Input[13]);
udtArrayData.MaxCellVoltage                 := TO_UINT(Input[14]);
udtArrayData.MinCellVoltage                 := TO_UINT(Input[15]);
udtArrayData.MaxUnconnectedPackVoltage      := TO_UINT(Input[16]);
udtArrayData.MinUnconnectedPackVoltage      := TO_UINT(Input[17]);

//Register 17 and 18: Pack heartbeat
udtArrayData.PackHeartbeat                  := TO_UDINT(Input[18]) + TO_UDINT(Input[19])*65536; //65536 = 2^16

//Register 19 to 25
udtArrayData.ServiceState                   := Input[20].%X0;
udtArrayData.BusCurrentUnscaled             := TO_INT(Input[21]);
udtArrayData.MinPackSOC                     := TO_UINT(Input[22]);
udtArrayData.AvgPackSOC                     := TO_UINT(Input[23]);
udtArrayData.MaxPackSOC                     := TO_UINT(Input[24]);
udtArrayData.AvgConnectedCellSOC            := TO_UINT(Input[25]);
udtArrayData.MaxConnectedCellSOC            := TO_UINT(Input[26]);

(*Register 50 - Byte 1 - Pack Status*)
//First byte
udtArrayData.PackOperationMode              := TO_INT(Input[51].%B0);

//Converts to boolean variable
udtArrayData.OpModePowerSave                := FALSE;
udtArrayData.OpModeFault                    := FALSE;
udtArrayData.OpModeReady                    := FALSE;
udtArrayData.OpModeConnecting               := FALSE;
udtArrayData.OpModeConnected                := FALSE;
udtArrayData.OpModeNotReady                 := FALSE;
udtArrayData.OpModeNoData                   := FALSE;

CASE udtArrayData.PackOperationMode OF
    1 : udtArrayData.OpModePowerSave        := TRUE;
    2 : udtArrayData.OpModeFault            := TRUE;
    3 : udtArrayData.OpModeReady            := TRUE;
    4 : udtArrayData.OpModeConnecting       := TRUE;
    5 : udtArrayData.OpModeConnected        := TRUE;
    6 : udtArrayData.OpModeNotReady         := TRUE;
    ELSE
        udtArrayData.OpModeNoData           := TRUE;
END_CASE;

//Second byte
udtArrayData.ServiceRequired                := Input[51].%X8;
```

Default

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
Input	MB_TCP_ARR_W_1_125	Input										
udtArrayData	BAT_Array_Data	InOut										

```
(*This function block controls the battery and handles everything related to the battery pack.*)

(*Battery communication*)
MB_Client(xActivate             := ClientData.xActivate,
          xAcknowledge          := ClientData.xAcknowledge,
          xAutoAck              := ClientData.xAutoAck,
          strServer_IP          := ClientData.strServer_IP,
          iPort                 := ClientData.iPort,
          strBindIp             := ClientData.strBindIp,
          uiBindPort            := ClientData.uiBindPort,
          xUDP_Mode             := ClientData.xUDP_Mode,
          tReconnectDelay       := ClientData.tReconnectDelay,
          tTimeout              := ClientData.tTimeout,
          xActive               => ClientData.xActive,
          xReady                => ClientData.xReady,
          xError                => ClientData.xError,
          wDiagCode             => ClientData.wDiagCode,
          wAddDiagCode          => ClientData.wAddDiagCode,
          udtDiag               => ClientData.udtDiag,
          udtTCP_ComData        := ComData);

(*Checks if communication with battery is OK*)
fbComCheck1(ClientData := ClientData, xComOK => xComOK);

IF xComOK THEN

    //Reads alarms
    fbReadRegistersAlarms(ComData       :=  ComData,
                          udtFC3Parameters    :=  udtAlarmsFC3Para);
    IF fbReadRegistersAlarms.xNewData THEN //Updates system variables for battery alarm status
        fbDataConvertAlarms1(Input := udtAlarmsFC3Para.arrRegisterValue, udtAlarmsData := udtData.Alarms);
    END_IF;

    //Reads array data
    fbReadRegistersArray(ComData        :=  ComData,
                         udtFC3Parameters    :=  udtArrayFC3Para);

    IF fbReadRegistersArray.xNewData THEN //Updates system variables for battery array data
        fbDataConvertArrayStatus1(Input := udtArrayFC3Para.arrRegisterValue, udtArrayData := udtData.Array_Data);
    END_IF;


    //Updates variables used by pMain:
    uiSOC           := udtData.Array_Data.AvgPackSOC;
    uiBusVoltage    := udtData.Array_Data.BusVoltage;
    iBusCurrent     := udtData.Array_Data.BusCurrent;
    xConnectedToBus := udtData.Array_Data.OpModeConnected;


    //Updates other variables:
    xCharging := udtData.Array_Data.BusCurrent > 0;


    //Connects the battery to the bus if xConnectToBus is True, disconnects otherwise
    IF xConnectToBus THEN
        IF NOT xConnectedToBus THEN
            udtCommandsFC16Para.arrRegisterValue[1].%X0 := TRUE;
            udtCommandsFC16Para.arrRegisterValue[1].%X1 := TRUE;

            fbWriteToRegisters1(ComData             := ComData,
                                uiHeartbeat         := TO_UINT(udtData.Array_Data.PackHeartbeat),
                                udtFC16Parameters   := udtCommandsFC16Para);

            udtCommandsFC16Para.arrRegisterValue[1].%X0 := FALSE;
            udtCommandsFC16Para.arrRegisterValue[1].%X1 := FALSE;
        END_IF;

    ELSE
        IF xConnectedToBus THEN
            udtCommandsFC16Para.arrRegisterValue[1].%X0 := FALSE;
            udtCommandsFC16Para.arrRegisterValue[1].%X1 := TRUE;

            fbWriteToRegisters1(ComData             := ComData,
                                uiHeartbeat         := TO_UINT(udtData.Array_Data.PackHeartbeat),
                                udtFC16Parameters   := udtCommandsFC16Para);

            udtCommandsFC16Para.arrRegisterValue[1].%X0 := F
```

```
78                  udtCommandsFC16Para.arrRegisterValue[1].%X1 := FALSE;
79              END_IF;
80
81          END_IF;
82
83
84          //Resets alarms
85          IF xResetAlarms THEN
86              fbReadRegistersAlarmReset(ComData := ComData,
87                  udtFC3Parameters := udtAlarmsFC3ParaReset);
88
89              IF fbReadRegistersAlarmReset.xNewData AND xLastAlarmResetCommand THEN
90                  udiAlarmResetHeartbeatFalse     := TO_UDINT(udtAlarmsFC3ParaReset.arrRegisterValue[1]) + TO_UDINT
                    (udtAlarmsFC3ParaReset.arrRegisterValue[2])*65536;
91
92              ELSIF fbReadRegistersAlarmReset.xNewData AND NOT xLastAlarmResetCommand THEN
93                  udiAlarmResetHeartbeatTrue      := TO_UDINT(udtAlarmsFC3ParaReset.arrRegisterValue[1]) + TO_UDINT
                    (udtAlarmsFC3ParaReset.arrRegisterValue[2])*65536;
94
95              END_IF;
96
97              IF NOT xLastAlarmResetCommand THEN
98                  IF udiAlarmResetHeartbeatFalse < udiAlarmResetHeartbeatTrue THEN
99                      xLastAlarmResetCommand := TRUE;
100                 ELSE
101                     udtCommandsFC16ParaAlarmReset.arrRegisterValue[1].%X0 := TRUE;
102
103                     fbWriteRegistersAlarmReset(uiHeartbeat      := TO_UINT(udtData.Array_Data.PackHeartbeat),
104                                 xWriteSuccess               => xAlarmResetSuccess,
105                                 ComData                     := ComData,
106                                 udtFC16Parameters           := udtCommandsFC16ParaAlarmReset);
107                 END_IF;
108             ELSE
109                 IF udiAlarmResetHeartbeatFalse > udiAlarmResetHeartbeatTrue THEN
110                     xLastAlarmResetCommand  := FALSE;
111                     xResetAlarms            := FALSE;
112                 ELSE
113                     udtCommandsFC16ParaAlarmReset.arrRegisterValue[1].%X0 := FALSE;
114
115                     fbWriteRegistersAlarmReset(uiHeartbeat      := TO_UINT(udtData.Array_Data.PackHeartbeat),
116                                 xWriteSuccess               => xAlarmResetSuccess,
117                                 ComData                     := ComData,
118                                 udtFC16Parameters           := udtCommandsFC16ParaAlarmReset);
119                 END_IF;
120             END_IF;
121         END_IF;
122     END_IF
```

Alarms

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
udtAlarmsFC3Para	udtMB_TCP_FC3	Local		Parameters to read alarm data from battery	(iMT_ID := INT#1, tUpdateTime := TIME#1s, wStartRegister := WORD#16#190, uiQuantityOfRegisters := UINT#5)	Private						
fbReadRegistersAlarms	fbReadFromRegisters	Local		Function block to read alarms		Private						
fbDataConvertAlarms1	fbDataConvertAlarms	Local		Function block to convert raw alarm data		Private						
udiAlarmResetHeartbeatTrue	UDINT	Local		Indicates the last hearbeat when the alarm reset was true		Private						
udiAlarmResetHeartbeatFalse	UDINT	Local		Indicates the last heartbeat when the alarm reset was false		Private						
xLastAlarmResetCommand	BOOL	Local		False: Last write was 0, true: Last write was 1.		Private						
udtAlarmsFC3ParaReset	udtMB_TCP_FC3	Local		Parameters for reading alarm reset heartbeat	(iMT_ID := INT#5, tUpdateTime := TIME#1s, wStartRegister := WORD#16#137, uiQuantityOfRegisters := UINT#2)	Private						
fbWriteRegistersAlarmReset	fbWriteToRegisters	Local		Function block to write alarm reset command		Private						
fbReadRegistersAlarmReset	fbReadFromRegisters	Local		Function block to read alarm reset hearbeat		Private						

ArrayData

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
udtArrayFC3Para	udtMB_TCP_FC3	Local		Parameters to read array data from battery	(iMT_ID := INT#2, tUpdateTime := TIME#1s, uiQuantityOfRegisters := UINT#51)	Private						
fbReadRegistersArray	fbReadFromRegisters	Local		Function block to read array data from library		Private						
fbDataConvertArrayStatus1	fbDataConvertArrayStatus	Local		Function block to convert raw array data		Private						

Commands

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
udtCommandsFC16Para	udtMB_TCP_FC16	Local		Parameters to write connect command to battery	(iMT_ID := INT#3, tUpdateTime := TIME#1s, wStartRegister := WORD#16#168, uiQuantityOfRegisters := UINT#1)	Private						
udtCommandsFC16ParaAlarmReset	udtMB_TCP_FC16	Local		Parameters to write reset alarm command to battery	(iMT_ID := INT#4, tUpdateTime := TIME#1s, wStartRegister := WORD#16#136, uiQuantityOfRegisters := UINT#1)	Private						
xResetAlarms	BOOL	Local		Command from HMI to reset alarms		Private						
xAlarmResetSuccess	BOOL	Local		Indicates that the alarms were successfully reset		Private						

Communication

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
MB_Client	MB_TCP_Client_9	Local		Sets up communication with Modbus server		Private						
fbWriteToRegisters1	fbWriteToRegisters	Local		Function block that handles writing to registers		Private						
ComData	MB_TCP_UDT_COMMUNICATION	Local		Internal communication between Modbus function blocks.		Private						
ClientData	udtMB_TCP_CLIENT	Local		Stores all data related to client	(xAutoAck := TRUE, strServer_IP := STRING#'192.168.3.22', iPort := INT#502, strBindIp := STRING#'0.0.0.0', tReconnectDelay := TIME#500ms, tTimeout := TIME#2s)	Private						
fbComCheck1	fbComCheck	Local		Function block that checks if communication is ok		Private						

Output

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
uiSOC	UINT	Output		State of Charge for array/ pack, in %								
xComOK	BOOL	Output		Communication with battery OK								
xConnectedToBus	BOOL	Output		Battery is connected to DC-Bus								
uiBusVoltage	UINT	Output		Voltage across DC-bus								
iBusCurrent	INT	Output		Current through DC-bus, negative discharging, positive charging								

Input

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
xConnectToBus	BOOL	Input		Command to connect battery to DC-bus								

Default

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
udtData	BAT_Data	Local		Data read from battery		Private						
xCharging	BOOL	Local		Indicates that the battery is charging		Private						

Vedlegg E4 - Kode for lastbank

Default

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
xShutdown	BOOL	Local		Initiate shutdown of socket and turn off loadbank		Private						
uiHMI_stepvar	UINT	Local		Variable to save step on loadbank		Private						
uiHMI_Step	UINT	Local		HMI variable, shows step in HMI		Private						

Constants

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
byVal1	BYTE	Local		Constant	BYTE#16#1	Private						
byVal0	BYTE	Local		Constant		Private						
iVal256	UINT	Local		Constant	UINT#16#256	Private						
iLoadbank_5	INT	Local		Limit value	INT#5	Private						

Variable convert variables

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
iVoltVar1	INT	Local		Variable for incoming voltage		Private						
iVoltVar2	INT	Local		Variable for incoming voltage		Private						
iVoltVar	INT	Local		Variable for incoming voltage		Private						
icurrvar1	INT	Local		Variable for incoming current		Private						
icurrvar2	INT	Local		Variable for incoming current		Private						
icurrvar	INT	Local		Variable for incoming current		Private						
uiPowVar1	UINT	Local		Variable for loaded power		Private						
uiPowVar2	UINT	Local		Variable for loaded power		Private						
uiPowVar	UINT	Local		Variable for loaded power		Private						
uilPowVar1	UINT	Local		Variable for incoming loadbank step		Private						
uilPowVar2	UINT	Local		Variable for incoming loadbank step		Private						
uilPowVar	UINT	Local		Variable for incoming loadbank step		Private						

Output

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
iLoadbank_Voltage_in	INT	Output		Input voltage loadbank								
uiPowerLoaded	UINT	Output		Stepvalue for loadbank								
uiPowerMeasured	UINT	Output		Power loaded by loadbank								
iAlarmLoadbank	INT	Output		Alarm Loadbank. 00 is no alarm								
xLoadbank_Loadstate	BOOL	Output		00 and loadbank is off, 07 is on								
xComOK	BOOL	Output		Variable that tells main program that communication is ok								

Input

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
xCharging	BOOL	Input		Decides discharge or charge of batterypack								
uiLoadpower	UINT	Input		Powerrequest from main program								
xLoad	BOOL	Input		Request from main program to load from main program								
xSendNow	BOOL	Input		Request from main program to instantly change step of loadbank								
xShutdown_Request	BOOL	Local		Request from main program to shutdown loadbank		Private						
xStartLoadbankCom	BOOL	Local		Request from main program to start communication with loadbank		Private						
xManual	BOOL	Input		To set loadbank to manual mode from HMI								
uiStep	UINT	Input		Variable sent from HMI to decide loadbank step								

Stepvalues

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
uiLoadbank_PStep0	UINT	Local		Power at Step 0, which is power value 0		Private						
uiLoadbank_PStep1	UINT	Local		Power at Step 1. which is power value 1	UINT#3	Private						
uiLoadbank_PStep2	UINT	Local		Power at Step 2, which is power value 4	UINT#6	Private						
uiLoadbank_PStep3	UINT	Local		Power at Step 3, which is power value 10	UINT#30	Private						
uiLoadbank_PStep4	UINT	Local		Power at Step 4, which is power value 11	UINT#33	Private						
uiLoadbank_PStep5	UINT	Local		Power at Step 5, which is power value 14	UINT#36	Private						
uiLoadbank_PStep6	UINT	Local		Power at Step 6, which is power value 20	UINT#59	Private						
uiLoadbank_PStep7	UINT	Local		Power at step 7, which is power value 21	UINT#62	Private						
uiLoadbank_PStep8	UINT	Local		Power at step 8, which is power value 24	UINT#65	Private						
uiLoadbank_PStep9	UINT	Local		Power at step 9, which is power value40	UINT#119	Private						
uiLoadbank_PStep10	UINT	Local		Power at step 10, which is power value 41	UINT#122	Private						
uiLoadbank_PStep11	UINT	Local		Power at step 11, which is power value 44	UINT#125	Private						

Stepvalues

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
uiLoadbank_PStep12	UINT	Local		Power at step 12, which is power value 300	UINT#139	Private						
uiLoadbank_PStep13	UINT	Local		Power at step 13, which is power value301	UINT#142	Private						
uiLoadbank_PStep14	UINT	Local		Power at step 14, which is power value 304	UINT#145	Private						
uiLoadbank_PStep15	UINT	Local		Power at step 15, which is power value 310	UINT#169	Private						
uiLoadbank_PStep16	UINT	Local		Power at step 16, which is power value 311	UINT#172	Private						
uiLoadbank_PStep17	UINT	Local		Power at step 17, which is power value 314	UINT#175	Private						
uiLoadbank_PStep18	UINT	Local		Power at step 18, which is power value 320	UINT#199	Private						
uiLoadbank_PStep19	UINT	Local		Power at step 19, which is power value 321	UINT#202	Private						
uiLoadbank_PStep20	UINT	Local		Power at step 20, which is power value 324	UINT#205	Private						
uiLoadbank_PStep21	UINT	Local		Power at step 21, which is power value 360	UINT#229	Private						
uiLoadbank_PStep22	UINT	Local		Power at step 22, which is power value 361	UINT#232	Private						
uiLoadbank_PStep23	UINT	Local		Power at step 23, which is power value 364	UINT#235	Private						
uiLoadbank_PStep24	UINT	Local		Power at step 24, which is power value 340	UINT#258	Private						
uiLoadbank_PStep25	UINT	Local		Power at step 25, which is power value 341	UINT#261	Private						
uiLoadbank_PStep26	UINT	Local		Power at step 26, which is power value 344	UINT#264	Private						
uiLoadbank_PStep27	UINT	Local		Power at step 27, which is power value 380	UINT#288	Private						
uiLoadbank_PStep28	UINT	Local		Power at step 28, which is power value 381	UINT#291	Private						
uiLoadbank_PStep29	UINT	Local		Power at step 29, which is power value 384	UINT#294	Private						
uiLoadbank_PStep30	UINT	Local		Power at step 30, which is power value 390	UINT#318	Private						
uiLoadbank_PStep31	UINT	Local		Power at step 31, which is power value 391	UINT#321	Private						
uiLoadbank_PStep32	UINT	Local		Power at step 32, which is power value 394	UINT#324	Private						
uiLoadbank_PStep33	UINT	Local		Power at step 33, which is power value 400	UINT#327	Private						
uiLoadbank_Stepverdi	UINT	Local		Value sent to Loadbank		Private						
uiLoadbank_PVStep0	UINT	Local		Power value step 0		Private						
uiLoadbank_PVStep1	UINT	Local		Power value step 1	UINT#1	Private						
uiLoadbank_PVStep2	UINT	Local		Power value step 4	UINT#4	Private						
uiLoadbank_PVStep3	UINT	Local		Power value step 10	UINT#10	Private						
uiLoadbank_PVStep4	UINT	Local		Power value step 11	UINT#11	Private						
uiLoadbank_PVStep5	UINT	Local		Power value step 14	UINT#14	Private						
uiLoadbank_PVStep6	UINT	Local		Power value step 20	UINT#20	Private						
uiLoadbank_PVStep7	UINT	Local		Power value step 21	UINT#21	Private						

Stepvalues

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
uiLoadbank_PVStep8	UINT	Local		Power value step 24	UINT#24	Private						
uiLoadbank_PVStep9	UINT	Local		Power value step 40	UINT#40	Private						
uiLoadbank_PVStep10	UINT	Local		Power value step 41	UINT#41	Private						
uiLoadbank_PVStep11	UINT	Local		Power value step 44	UINT#44	Private						
uiLoadbank_PVStep12	UINT	Local		Power value step 300	UINT#300	Private						
uiLoadbank_PVStep13	UINT	Local		Power value step 301	UINT#301	Private						
uiLoadbank_PVStep14	UINT	Local		Power value step 304	UINT#304	Private						
uiLoadbank_PVStep15	UINT	Local		Power value step 310	UINT#310	Private						
uiLoadbank_PVStep16	UINT	Local		Power value step 311	UINT#311	Private						
uiLoadbank_PVStep17	UINT	Local		Power value step 314	UINT#314	Private						
uiLoadbank_PVStep18	UINT	Local		Power value step 320	UINT#320	Private						
uiLoadbank_PVStep19	UINT	Local		Power value step 321	UINT#321	Private						
uiLoadbank_PVStep20	UINT	Local		Power value step 324	UINT#324	Private						
uiLoadbank_PVStep21	UINT	Local		Power value step 360	UINT#360	Private						
uiLoadbank_PVStep22	UINT	Local		Power value step 361	UINT#361	Private						
uiLoadbank_PVStep23	UINT	Local		Power value step 364	UINT#364	Private						
uiLoadbank_PVStep24	UINT	Local		Power value step 340	UINT#340	Private						
uiLoadbank_PVStep25	UINT	Local		Power value step 341	UINT#341	Private						
uiLoadbank_PVStep26	UINT	Local		Power value step 344	UINT#344	Private						
uiLoadbank_PVStep27	UINT	Local		Power value step 380	UINT#380	Private						
uiLoadbank_PVStep28	UINT	Local		Power value step 381	UINT#381	Private						
uiLoadbank_PVStep29	UINT	Local		Power value step 384	UINT#384	Private						
uiLoadbank_PVStep30	UINT	Local		Power value step 390	UINT#390	Private						
uiLoadbank_PVStep31	UINT	Local		Power value step 391	UINT#391	Private						
uiLoadbank_PVStep32	UINT	Local		Power value step 394	UINT#394	Private						
uiLoadbank_PVStep33	UINT	Local		Power value step 400	UINT#400	Private						

Recieved variables from Loadbank

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
iLoadbank_Current_in	INT	Local		Current in to loadbank		Private						
iLoadbank_Loadvariable	INT	Local		Loadvariable to loadbank		Private						
iLoadbank_Backup	INT	Local		Backupvariable. Undefined in documentation.		Private						

Blocks

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
TLS_SOCKET_12	TLS_SOCKET_2	Local		Client		Private						
TLS_SEND_12	TLS_SEND_2	Local		Functionblock, Send to loadbank		Private						
TLS_RECEIVE_12	TLS_RECEIVE_2	Local		Functionblock, recieve from loadbank		Private						

Send/Recieve Outputs

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
xLoadbank_rNDR	BOOL	Local		New data successfully recieved		Private						

Send/Recieve Inputs

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
byLoadbankInData	ByteArray35	Local		Data in		Private						
xLoadbank_SendData	BOOL	Local		Activates Sending Data		Private						
xLoadbank_activate	BOOL	Local		Activates Socket Com	TRUE	Private						

Send/Recieve Adresser

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
strLoadbank_dest_ip	STRING	Local		IP to Loadbank	STRING#'192.168.3.21'	Private						
uiLoadbank_dest_port	UINT	Local		Port to Loadbank	UINT#2000	Private						
strLoadbank_bind_ip	STRING	Local		Bind IP, keep empty		Private						
uiLoadbank_bind_port	UINT	Local		Bind port, keep empty		Private						
strLoadbank_rIP	STRING	Local		IP which data is recieved		Private						
uiLoadbank_rPort	UINT	Local		Port which data is recieved		Private						
udiLoadbank_rDatacnt	UDINT	Local		Recieved byte count		Private						
udiLoadbank_rByte	UDINT	Local		Expected recieved byte count	UDINT#19	Private						
udiLoadbank_sByte	UDINT	Local		Expected byte count sent	UDINT#3	Private						

Send/Recieve Errors

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
xLoadbank_Error	BOOL	Local		Socket error activated		Private						
wLoadbankStatus	WORD	Local		Fault code		Private						
xLoadbank_sError	BOOL	Local		Functionblock sending error activated		Private						
wLoadbank_sStatus	WORD	Local		Fault code		Private						
xLoadbank_rError	BOOL	Local		Functionblock recieve error activated		Private						
wLoadbank_rStatus	WORD	Local		Fault code		Private						

Send/Recieve Blockvariables

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
xLoadbank_is_active	BOOL	Local		Internal variable, communication is active		Private						
xLoadbank_Busy	BOOL	Local		Socket is working on activating communication		Private						
uiLoadbank_UsedPort	UINT	Local		Tells which port is used		Private						
dwLoadbank_Handle	DWORD	Local		Common variable for functionblocks		Private						
xLoadbank_IS_SRV	BOOL	Local		Defines master/slave, keep false		Private						
ciLoadbank_ConnectInfo	ConnectInfo	Local		Predefined data variable for the functionblocks		Private						
xLoadbank_sDone	BOOL	Local		Activated when data finished sending		Private						
xLoadbank_sBusy	BOOL	Local		Activated when busy sending data		Private						

Sendingvariables

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
uliLoadbank_Heartbeat	ULINT	Local		Heartbeat, sending variable.		Private						
xLoadbank_ReadyToSend	BOOL	Local		Tells sendingblock it can send		Private						
uliLoadbank_Heartrate	ULINT	Local		Tells how high heartbeat has to be before it allows sending	ULINT#400	Private						
xLoadbank_Dontsend	BOOL	Local		Resets heartbeat if it should wait more with sending data		Private						

Senddata

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
arrLoadbank_UtData	ByteArray2	Local		Array for data sent to loadbank		Private						
arrLoadbank_LastSent	ByteArray2	Local		Saves last sent data sent to loadbank		Private						
byLoadbankLoad	BYTE	Local		Constant, used to send load to loadbank	BYTE#8#07	Private						
byLoadbankNotLoad	BYTE	Local		Constant, used to send not load to loadbank	BYTE#8#00	Private						
byLoadbank_Step	BYTE	Local		Step that gets converted in to array to send to loadbank		Private						

```
(*Activates socket*)
IF xStartLoadbankCom THEN
    xLoadbank_activate := TRUE;
    xShutdown:=FALSE;
    xStartLoadbankCom := FALSE;

    END_IF


(* Communication-socket to Loadbank *)
TLS_SOCKET_12(
    ACTIVATE        := xLoadbank_activate,
    IS_SRV          := xLoadbank_IS_SRV,
    BIND_IP         := strLoadbank_bind_ip,
    BIND_PORT       := uiLoadbank_bind_port,
    DEST_IP         := strLoadbank_dest_ip,
    DEST_PORT       := uiLoadbank_dest_port,
    CONNECT_INFO    := ciLoadbank_ConnectInfo,
    START_TLS       := FALSE,
    HANDLE          => dwLoadbank_Handle,
    ACTIVE          => xLoadbank_is_active,
    BUSY            => xLoadbank_Busy,
    ERROR           => xLoadbank_Error,
    STATUS          => wLoadbankStatus,
    USED_PORT       => uiLoadbank_UsedPort);

 (* Resets variables when communication is shutdown *)
IF xShutdown OR NOT xLoadbank_is_active THEN
    (*Resets variables*)
    xComOK := FALSE;
    xShutdown := FALSE;
    xShutdown_Request := FALSE;
    END_IF;


(*When communication is established, Loadbank will start sending data*)
(*Data is recieved and converted to variables*)
IF xLoadbank_is_active THEN
        TLS_RECEIVE_12(
            EN_R            := xLoadbank_activate,
            HANDLE          := dwLoadbank_Handle,
            EXP_DATA_CNT    := udiLoadbank_rByte,
            RECEIVE_SECURE  := FALSE,
            NDR             => xLoadbank_rNDR,
            ERROR           => xLoadbank_rError,
            STATUS          => wLoadbank_rStatus,
            SOURCE_IP       => strLoadbank_rIP,
            SOURCE_PORT     => uiLoadbank_rPort,
            DATA_CNT        => udiLoadbank_rDatacnt,
            DATA            := byLoadbankInData);

    xComOK := TRUE;

    (*00 00 00 00 = Input voltage in hexa, convert to i. Place 2, 3*)
    iVoltVar1 := TO_INT(byLoadbankInData[3]);
    iVoltVar2 := TO_INT(byLoadbankInData[2]);
    iVoltVar := iVoltVar1+(iVoltVar2*256);
    iLoadbank_Voltage_in := iVoltVar;

    (*00 00 00 00 = Input current in hexa, convert to i. Place 6, 7*)
    icurrvar1 := TO_INT(byLoadbankInData[7]);
    icurrvar2 := TO_INT(byLoadbankInData[6]);
    icurrvar := icurrvar1+(icurrvar2*256);
    iLoadbank_Current_in := icurrvar;

    (*00 00 00 00 = Input power in hexa, convert to i. Place 8, 9, 10, 11*)
    uiPowVar1 := TO_UINT(byLoadbankInData[11]);
    uiPowVar2 := TO_UINT(byLoadbankInData[10]);
    uiPowVar := uiPowVar1+(uiPowVar2*256);
    uiPowerMeasured := uiPowVar;

    (*00 00 00 00 = loaded power in hexa, convert to i. Place 12, 13, 14, 15*)
    uilPowVar1 := TO_UINT(byLoadbankInData[15]);
    uilPowVar2 := TO_UINT(byLoadbankInData[14]);
    uilPowVar := uilPowVar1+(uilPowVar2*256);
    uiPowerLoaded := uilPowVar;
```

```
78          (*07 = load, 00 = not load -> Convert to bool. True = load, false = not load*)
79          iLoadbank_Loadvariable := TO_INT(byLoadbankInData[16]);
80          IF iLoadbank_Loadvariable > iLoadbank_5 THEN
81              xLoadbank_Loadstate := TRUE;
82          ELSE
83              xLoadbank_Loadstate := FALSE;
84          END_IF;
85
86          (*00 = Backup*)
87          iLoadbank_Backup := TO_INT(byLoadbankInData[17]);
88
89          (*00 = No alarm. 01-07 = misc. alarms ref documentation*)
90          iAlarmLoadbank := TO_INT(byLoadbankInData[18]);
91
92          END_IF;
93
94
95      (* Heartbeat. Controls time between switching *)
96      IF xLoadbank_is_active THEN
97          (* Counts heartbeat up to limit *)
98          uliLoadbank_Heartbeat := uliLoadbank_Heartbeat + 1;
99
100         (* Sets sending allowed to true when counted high enough, or gets signalled to send now *)
101         IF uliLoadbank_Heartbeat > uliLoadbank_Heartrate THEN
102             xLoadbank_ReadyToSend := TRUE;
103         ELSIF xSendNow THEN
104             xLoadbank_ReadyToSend := TRUE;
105         END_IF
106         END_IF
107
108
109     (*Selecting step on Loadbank.
110     Powers Loaded by Loadbank is calculated at 698V and added manually to the variables*)
111     IF xLoadbank_is_active AND xManual THEN
112         CASE uiStep OF
113             0: uiLoadbank_Stepverdi := UiLoadbank_PVStep0;
114                 uiHMI_stepvar := 0;
115             1: uiLoadbank_Stepverdi := UiLoadbank_PVStep1;
116                 uiHMI_stepvar := 1;
117             2: uiLoadbank_Stepverdi := UiLoadbank_PVStep2;
118                 uiHMI_stepvar := 2;
119             3: uiLoadbank_Stepverdi := UiLoadbank_PVStep3;
120                 uiHMI_stepvar := 3;
121             4: uiLoadbank_Stepverdi := UiLoadbank_PVStep4;
122                 uiHMI_stepvar := 4;
123             5: uiLoadbank_Stepverdi := UiLoadbank_PVStep5;
124                 uiHMI_stepvar := 5;
125             6: uiLoadbank_Stepverdi := UiLoadbank_PVStep6;
126                 uiHMI_stepvar := 6;
127             7: uiLoadbank_Stepverdi := UiLoadbank_PVStep7;
128                 uiHMI_stepvar := 7;
129             8: uiLoadbank_Stepverdi := UiLoadbank_PVStep8;
130                 uiHMI_stepvar := 8;
131             9: uiLoadbank_Stepverdi := UiLoadbank_PVStep9;
132                 uiHMI_stepvar := 9;
133             10: uiLoadbank_Stepverdi := UiLoadbank_PVStep10;
134                 uiHMI_stepvar := 10;
135             11: uiLoadbank_Stepverdi := UiLoadbank_PVStep11;
136                 uiHMI_stepvar := 11;
137             12: uiLoadbank_Stepverdi := UiLoadbank_PVStep12;
138                 uiHMI_stepvar := 12;
139             13: uiLoadbank_Stepverdi := UiLoadbank_PVStep13;
140                 uiHMI_stepvar := 13;
141             14: uiLoadbank_Stepverdi := UiLoadbank_PVStep14;
142                 uiHMI_stepvar := 14;
143             15: uiLoadbank_Stepverdi := UiLoadbank_PVStep15;
144                 uiHMI_stepvar := 15;
145             16: uiLoadbank_Stepverdi := UiLoadbank_PVStep16;
146                 uiHMI_stepvar := 16;
147             17: uiLoadbank_Stepverdi := UiLoadbank_PVStep17;
148                 uiHMI_stepvar := 17;
149             18: uiLoadbank_Stepverdi := UiLoadbank_PVStep18;
150                 uiHMI_stepvar := 18;
151             19: uiLoadbank_Stepverdi := UiLoadbank_PVStep19;
152                 uiHMI_stepvar := 19;
153             20: uiLoadbank_Stepverdi := UiLoadbank_PVStep20;
154                 uiHMI_stepvar := 20;
```

```
155             21: uiLoadbank_Stepverdi := UiLoadbank_PVStep21;
156                 uiHMI_stepvar := 21;
157             22: uiLoadbank_Stepverdi := UiLoadbank_PVStep22;
158                 uiHMI_stepvar := 22;
159             23: uiLoadbank_Stepverdi := UiLoadbank_PVStep23;
160                 uiHMI_stepvar := 23;
161             24: uiLoadbank_Stepverdi := UiLoadbank_PVStep24;
162                 uiHMI_stepvar := 24;
163             25: uiLoadbank_Stepverdi := UiLoadbank_PVStep25;
164                 uiHMI_stepvar := 25;
165             26: uiLoadbank_Stepverdi := UiLoadbank_PVStep26;
166                 uiHMI_stepvar := 26;
167             27: uiLoadbank_Stepverdi := UiLoadbank_PVStep27;
168                 uiHMI_stepvar := 27;
169             28: uiLoadbank_Stepverdi := UiLoadbank_PVStep28;
170                 uiHMI_stepvar := 28;
171             29: uiLoadbank_Stepverdi := UiLoadbank_PVStep29;
172                 uiHMI_stepvar := 29;
173             30: uiLoadbank_Stepverdi := UiLoadbank_PVStep30;
174                 uiHMI_stepvar := 30;
175             31: uiLoadbank_Stepverdi := UiLoadbank_PVStep31;
176                 uiHMI_stepvar := 31;
177             32: uiLoadbank_Stepverdi := UiLoadbank_PVStep32;
178                 uiHMI_stepvar := 32;
179             33: uiLoadbank_Stepverdi := UiLoadbank_PVStep33;
180                 uiHMI_stepvar := 33;
181         END_CASE;
182
183
184     ELSIF xLoadbank_is_Active & xCharging THEN
185         (* For when the battery should charge *)
186         IF uiLoadbank_PStep33 < uiLoadPower THEN
187             uiLoadbank_Stepverdi := UiLoadbank_PVStep33; (* Step 33 *)
188             uiHMI_stepvar := 33;
189         ELSIF uiLoadbank_PStep32 < uiLoadPower THEN
190             uiLoadbank_Stepverdi := UiLoadbank_PVStep32; (* Step 32 *)
191             uiHMI_stepvar := 32;
192         ELSIF uiLoadbank_PStep31 < uiLoadPower THEN
193             uiLoadbank_Stepverdi := UiLoadbank_PVStep31; (* Step 31 *)
194             uiHMI_stepvar := 31;
195         ELSIF uiLoadbank_PStep30 < uiLoadPower THEN
196             uiLoadbank_Stepverdi := UiLoadbank_PVStep30; (* Step 30 *)
197             uiHMI_stepvar := 30;
198         ELSIF uiLoadbank_PStep29 < uiLoadPower THEN
199             uiLoadbank_Stepverdi := UiLoadbank_PVStep29; (* Step 29 *)
200             uiHMI_stepvar := 29;
201         ELSIF uiLoadbank_PStep28 < uiLoadPower THEN
202             uiLoadbank_Stepverdi := UiLoadbank_PVStep28; (* Step 28 *)
203             uiHMI_stepvar := 28;
204         ELSIF uiLoadbank_PStep27 < uiLoadPower THEN
205             uiLoadbank_Stepverdi := UiLoadbank_PVStep27; (* Step 27 *)
206             uiHMI_stepvar := 27;
207         ELSIF uiLoadbank_PStep26 < uiLoadPower THEN
208             uiLoadbank_Stepverdi := UiLoadbank_PVStep26; (* Step 26 *)
209             uiHMI_stepvar := 26;
210         ELSIF uiLoadbank_PStep25 < uiLoadPower THEN
211             uiLoadbank_Stepverdi := UiLoadbank_PVStep25; (* Step 25 *)
212             uiHMI_stepvar := 25;
213         ELSIF uiLoadbank_PStep24 < uiLoadPower THEN
214             uiLoadbank_Stepverdi := UiLoadbank_PVStep24; (* Step 24 *)
215             uiHMI_stepvar := 24;
216         ELSIF uiLoadbank_PStep23 < uiLoadPower THEN
217             uiLoadbank_Stepverdi := UiLoadbank_PVStep23; (* Step 23 *)
218             uiHMI_stepvar := 23;
219         ELSIF uiLoadbank_PStep22 < uiLoadPower THEN
220             uiLoadbank_Stepverdi := UiLoadbank_PVStep22; (* Step 22 *)
221             uiHMI_stepvar := 22;
222         ELSIF uiLoadbank_PStep21 < uiLoadPower THEN
223             uiLoadbank_Stepverdi := UiLoadbank_PVStep21; (* Step 21 *)
224             uiHMI_stepvar := 21;
225         ELSIF uiLoadbank_PStep20 < uiLoadPower THEN
226             uiLoadbank_Stepverdi := UiLoadbank_PVStep20; (* Step 20 *)
227             uiHMI_stepvar := 20;
228         ELSIF uiLoadbank_PStep19 < uiLoadPower THEN
229             uiLoadbank_Stepverdi := UiLoadbank_PVStep19; (* Step 19 *)
230             uiHMI_stepvar := 19;
231         ELSIF uiLoadbank_PStep18 < uiLoadPower THEN
```

```
232             uiLoadbank_Stepverdi := UiLoadbank_PVStep18; (* Step 18 *)
233             uiHMI_stepvar := 18;
234         ELSIF uiLoadbank_PStep17 < uiLoadPower THEN
235             uiLoadbank_Stepverdi := UiLoadbank_PVStep17; (* Step 17 *)
236             uiHMI_stepvar := 17;
237         ELSIF uiLoadbank_PStep16 < uiLoadPower THEN
238             uiLoadbank_Stepverdi := UiLoadbank_PVStep16; (* Step 16 *)
239             uiHMI_stepvar := 16;
240         ELSIF uiLoadbank_PStep15 < uiLoadPower THEN
241             uiLoadbank_Stepverdi := UiLoadbank_PVStep15; (* Step 15 *)
242             uiHMI_stepvar := 15;
243         ELSIF uiLoadbank_PStep14 < uiLoadPower THEN
244             uiLoadbank_Stepverdi := UiLoadbank_PVStep14; (* Step 14 *)
245             uiHMI_stepvar := 14;
246         ELSIF uiLoadbank_PStep13 < uiLoadPower THEN
247             uiLoadbank_Stepverdi := UiLoadbank_PVStep13; (* Step 13 *)
248             uiHMI_stepvar := 13;
249         ELSIF uiLoadbank_PStep12 < uiLoadPower THEN
250             uiLoadbank_Stepverdi := UiLoadbank_PVStep12; (* Step 12 *)
251             uiHMI_stepvar := 12;
252         ELSIF uiLoadbank_PStep11 < uiLoadPower THEN
253             uiLoadbank_Stepverdi := UiLoadbank_PVStep11; (* Step 11 *)
254             uiHMI_stepvar := 11;
255         ELSIF uiLoadbank_PStep10 < uiLoadPower THEN
256             uiLoadbank_Stepverdi := UiLoadbank_PVStep10; (* Step 10 *)
257             uiHMI_stepvar := 10;
258         ELSIF uiLoadbank_PStep9 < uiLoadPower THEN
259             uiLoadbank_Stepverdi := UiLoadbank_PVStep9; (* Step 9 *)
260             uiHMI_stepvar := 9;
261         ELSIF uiLoadbank_PStep8 < uiLoadPower THEN
262             uiLoadbank_Stepverdi := UiLoadbank_PVStep8; (* Step 8 *)
263             uiHMI_stepvar := 8;
264         ELSIF uiLoadbank_PStep7 < uiLoadPower THEN
265             uiLoadbank_Stepverdi := UiLoadbank_PVStep7; (* Step 7 *)
266             uiHMI_stepvar := 7;
267         ELSIF uiLoadbank_PStep6 < uiLoadPower THEN
268             uiLoadbank_Stepverdi := UiLoadbank_PVStep6; (* Step 6 *)
269             uiHMI_stepvar := 6;
270         ELSIF uiLoadbank_PStep5 < uiLoadPower THEN
271             uiLoadbank_Stepverdi := UiLoadbank_PVStep5; (* Step 5 *)
272             uiHMI_stepvar := 5;
273         ELSIF uiLoadbank_PStep4 < uiLoadPower THEN
274             uiLoadbank_Stepverdi := UiLoadbank_PVStep4; (* Step 4 *)
275             uiHMI_stepvar := 4;
276         ELSIF uiLoadbank_PStep3 < uiLoadPower THEN
277             uiLoadbank_Stepverdi := UiLoadbank_PVStep3; (* Step 3 *)
278             uiHMI_stepvar := 3;
279         ELSIF uiLoadbank_PStep2 < uiLoadPower THEN
280             uiLoadbank_Stepverdi := UiLoadbank_PVStep2; (* Step 2 *)
281             uiHMI_stepvar := 2;
282         ELSIF uiLoadbank_PStep1 < uiLoadPower THEN
283             uiLoadbank_Stepverdi := UiLoadbank_PVStep1; (* Step 1 *)
284             uiHMI_stepvar := 1;
285         ELSE
286             uiLoadbank_Stepverdi := UiLoadbank_PVStep0; (* Step 0 *)
287             uiHMI_stepvar := 0;
288         END_IF;
289
290         ELSIF xLoadbank_is_Active & NOT xCharging THEN
291         (* For when the battery should discharge *)
292         IF uiLoadbank_PStep32 < uiLoadPower THEN
293             uiLoadbank_Stepverdi := UiLoadbank_PVStep33; (* Step 33 *)
294             uiHMI_stepvar := 33;
295         ELSIF uiLoadbank_PStep31 < uiLoadPower THEN
296             uiLoadbank_Stepverdi := UiLoadbank_PVStep32; (* Step 32 *)
297             uiHMI_stepvar := 32;
298         ELSIF uiLoadbank_PStep30 < uiLoadPower THEN
299             uiLoadbank_Stepverdi := UiLoadbank_PVStep31; (* Step 31 *)
300             uiHMI_stepvar := 31;
301         ELSIF uiLoadbank_PStep29 < uiLoadPower THEN
302             uiLoadbank_Stepverdi := UiLoadbank_PVStep30; (* Step 30 *)
303             uiHMI_stepvar := 30;
304         ELSIF uiLoadbank_PStep28 < uiLoadPower THEN
305             uiLoadbank_Stepverdi := UiLoadbank_PVStep29; (* Step 29 *)
306             uiHMI_stepvar := 29;
307         ELSIF uiLoadbank_PStep27 < uiLoadPower THEN
308             uiLoadbank_Stepverdi := UiLoadbank_PVStep28; (* Step 28 *)
```

```
309             uiHMI_stepvar := 28;
310         ELSIF uiLoadbank_PStep26 < uiLoadPower THEN
311             uiLoadbank_Stepverdi := UiLoadbank_PVStep27; (* Step 27 *)
312             uiHMI_stepvar := 27;
313         ELSIF uiLoadbank_PStep25 < uiLoadPower THEN
314             uiLoadbank_Stepverdi := UiLoadbank_PVStep26; (* Step 26 *)
315             uiHMI_stepvar := 26;
316         ELSIF uiLoadbank_PStep24 < uiLoadPower THEN
317             uiLoadbank_Stepverdi := UiLoadbank_PVStep25; (* Step 25 *)
318             uiHMI_stepvar := 25;
319         ELSIF uiLoadbank_PStep23 < uiLoadPower THEN
320             uiLoadbank_Stepverdi := UiLoadbank_PVStep24; (* Step 24 *)
321             uiHMI_stepvar := 24;
322         ELSIF uiLoadbank_PStep22 < uiLoadPower THEN
323             uiLoadbank_Stepverdi := UiLoadbank_PVStep23; (* Step 23 *)
324             uiHMI_stepvar := 23;
325         ELSIF uiLoadbank_PStep21 < uiLoadPower THEN
326             uiLoadbank_Stepverdi := UiLoadbank_PVStep22; (* Step 22 *)
327             uiHMI_stepvar := 22;
328         ELSIF uiLoadbank_PStep20 < uiLoadPower THEN
329             uiLoadbank_Stepverdi := UiLoadbank_PVStep21; (* Step 21 *)
330             uiHMI_stepvar := 21;
331         ELSIF uiLoadbank_PStep19 < uiLoadPower THEN
332             uiLoadbank_Stepverdi := UiLoadbank_PVStep20; (* Step 20 *)
333             uiHMI_stepvar := 20;
334         ELSIF uiLoadbank_PStep18 < uiLoadPower THEN
335             uiLoadbank_Stepverdi := UiLoadbank_PVStep19; (* Step 19 *)
336             uiHMI_stepvar := 19;
337         ELSIF uiLoadbank_PStep17 < uiLoadPower THEN
338             uiLoadbank_Stepverdi := UiLoadbank_PVStep18; (* Step 18 *)
339             uiHMI_stepvar := 18;
340         ELSIF uiLoadbank_PStep16 < uiLoadPower THEN
341             uiLoadbank_Stepverdi := UiLoadbank_PVStep17; (* Step 17 *)
342             uiHMI_stepvar := 17;
343         ELSIF uiLoadbank_PStep15 < uiLoadPower THEN
344             uiLoadbank_Stepverdi := UiLoadbank_PVStep16; (* Step 16 *)
345             uiHMI_stepvar := 16;
346         ELSIF uiLoadbank_PStep14 < uiLoadPower THEN
347             uiLoadbank_Stepverdi := UiLoadbank_PVStep15; (* Step 15 *)
348             uiHMI_stepvar := 15;
349         ELSIF uiLoadbank_PStep13 < uiLoadPower THEN
350             uiLoadbank_Stepverdi := UiLoadbank_PVStep14; (* Step 14 *)
351             uiHMI_stepvar := 14;
352         ELSIF uiLoadbank_PStep12 < uiLoadPower THEN
353             uiLoadbank_Stepverdi := UiLoadbank_PVStep13; (* Step 13 *)
354             uiHMI_stepvar := 13;
355         ELSIF uiLoadbank_PStep11 < uiLoadPower THEN
356             uiLoadbank_Stepverdi := UiLoadbank_PVStep12; (* Step 12 *)
357             uiHMI_stepvar := 12;
358         ELSIF uiLoadbank_PStep10 < uiLoadPower THEN
359             uiLoadbank_Stepverdi := UiLoadbank_PVStep11; (* Step 11 *)
360             uiHMI_stepvar := 11;
361         ELSIF uiLoadbank_PStep9 < uiLoadPower THEN
362             uiLoadbank_Stepverdi := UiLoadbank_PVStep10; (* Step 10 *)
363             uiHMI_stepvar := 10;
364         ELSIF uiLoadbank_PStep8 < uiLoadPower THEN
365             uiLoadbank_Stepverdi := UiLoadbank_PVStep9; (* Step 9 *)
366             uiHMI_stepvar := 9;
367         ELSIF uiLoadbank_PStep7 < uiLoadPower THEN
368             uiLoadbank_Stepverdi := UiLoadbank_PVStep8; (* Step 8 *)
369             uiHMI_stepvar := 8;
370         ELSIF uiLoadbank_PStep6 < uiLoadPower THEN
371             uiLoadbank_Stepverdi := UiLoadbank_PVStep7; (* Step 7 *)
372             uiHMI_stepvar := 7;
373         ELSIF uiLoadbank_PStep5 < uiLoadPower THEN
374             uiLoadbank_Stepverdi := UiLoadbank_PVStep6; (* Step 6 *)
375             uiHMI_stepvar := 6;
376         ELSIF uiLoadbank_PStep4 < uiLoadPower THEN
377             uiLoadbank_Stepverdi := UiLoadbank_PVStep5; (* Step 5 *)
378             uiHMI_stepvar := 5;
379         ELSIF uiLoadbank_PStep3 < uiLoadPower THEN
380             uiLoadbank_Stepverdi := UiLoadbank_PVStep4; (* Step 4 *)
381             uiHMI_stepvar := 4;
382         ELSIF uiLoadbank_PStep2 < uiLoadPower THEN
383             uiLoadbank_Stepverdi := UiLoadbank_PVStep3; (* Step 3 *)
384             uiHMI_stepvar := 3;
385         ELSIF uiLoadbank_PStep1 < uiLoadPower THEN
```

```
386             uiLoadbank_Stepverdi := UiLoadbank_PVStep2; (* Step 2 *)
387             uiHMI_stepvar := 2;
388         ELSIF uiLoadbank_PStep0 < uiLoadPower THEN
389             uiLoadbank_Stepverdi := UiLoadbank_PVStep1; (* Step 1 *)
390             uiHMI_stepvar := 1;
391         ELSE
392             uiLoadbank_Stepverdi := UiLoadbank_PVStep0; (* Step 0 *)
393             uiHMI_stepvar := 0;
394         END_IF
395     ELSE
396     END_IF;
397
398     IF uiPowerLoaded = uiLoadbank_Stepverdi THEN
399           uiHMI_Step := uiHMI_stepvar;
400     END_IF
401
402
403
404     (* Convert int value to data that gets sent to Loadbank *)
405     IF xLoadbank_is_active & xLoad THEN
406         IF uiLoadbank_Stepverdi > iVal256 THEN
407             arrLoadbank_UtData[0] := byVal1;
408             uiLoadbank_Stepverdi := uiLoadbank_Stepverdi - iVal256;
409         ELSE
410             arrLoadbank_UtData[0] := byVal0;
411     END_IF;
412
413         byLoadbank_Step := TO_BYTE(uiLoadbank_Stepverdi);
414         arrLoadbank_UtData[1] := byLoadbank_Step;
415         arrLoadbank_UtData[2] := byLoadbankLoad;
416
417         xLoadbank_Dontsend := FALSE; (* So it can send the new variable *)
418
419     ELSIF xLoadbank_is_active& NOT xLoad THEN
420
421         IF uiLoadbank_Stepverdi > iVal256 THEN
422             arrLoadbank_UtData[0] := byVal1;
423             uiLoadbank_Stepverdi := uiLoadbank_Stepverdi - iVal256;
424         ELSE
425             arrLoadbank_UtData[0] := byVal0;
426         END_IF
427
428         byLoadbank_Step := TO_BYTE(uiLoadbank_Stepverdi);
429         arrLoadbank_UtData[1] := byLoadbank_Step;
430         arrLoadbank_UtData[2] := byLoadbankNotLoad;
431
432         xLoadbank_Dontsend := FALSE; (* So it can send the new variable *)
433     ELSE
434     END_IF;
435
436
437     (*Comparing strings recieved from Loadbank. Does not send if Loadbank is already loading the right value.
438       Sends next cycle if last sent is not the same as current state of Loadbank.*)
439     IF byLoadbankInData[14] = arrLoadbank_UtData[0]
440     AND byLoadbankInData[15] = arrLoadbank_UtData[1]
441     AND byLoadbankInData[16] = arrLoadbank_UtData[2] THEN
442             xLoadbank_Dontsend := TRUE;
443
444     ELSIF byLoadbankInData[14] =NOT arrLoadbank_LastSent[0]
445     AND byLoadbankInData[15] =NOT arrLoadbank_LastSent[1] THEN
446             xLoadbank_Dontsend := FALSE;
447             uliLoadbank_Heartbeat := uliLoadbank_Heartrate;
448
449     ELSIF byLoadbankInData[16] =NOT arrLoadbank_LastSent [2] THEN
450         xLoadbank_Dontsend := FALSE;
451         uliLoadbank_Heartbeat := uliLoadbank_Heartrate;
452     END_IF;
453
454     (*If shutdown is wanted, changes variables to 00 00 00, sends it and then shuts down socket*)
455     If xShutdown_Request then
456         arrLoadbank_UtData[0] := 00;
457         arrLoadbank_UtData[1] := 00;
458         arrLoadbank_UtData[2] := 00;
459
460         xShutdown := TRUE;
461         xLoadbank_Activate := FALSE;
462         xLoadbank_ReadyToSend := TRUE;
```

```
463         xLoadbank_Dontsend := FALSE;
464     END_IF;
465
466
467     (* Sending array to Loadbank *)
468     IF xLoadbank_is_active & xLoadbank_ReadyToSend & NOT xLoadbank_Dontsend THEN
469         xLoadbank_SendData:= TRUE;
470     END_IF;
471
472     (* TLS SENDING BLOCK *)
473     TLS_SEND_12(
474         REQ := xLoadbank_SendData, (*Initiate sending data signal*)
475         HANDLE := dwLoadbank_Handle,
476         DATA_CNT := udiLoadbank_sByte, (*3 byte*)
477         SEND_SECURE := FALSE, (*TCP = False, TLS = True*)
478         DONE => xLoadbank_sDone,
479         BUSY => xLoadbank_sBusy,
480         ERROR => xLoadbank_sError,
481         STATUS => wLoadbank_sStatus,
482         DATA := arrLoadbank_UtData);
483
484     (* Resetting variables that allows sending *)
485     IF xLoadbank_sDone THEN
486         xLoadbank_SendData := FALSE;
487         xLoadbank_Readytosend := FALSE;
488         uliLoadbank_Heartbeat := 0;
489         xLoadbank_Dontsend := FALSE;
490         arrLoadbank_LastSent[0] := arrLoadbank_UtData[0];
491         arrLoadbank_LastSent[1] := arrLoadbank_UtData[1];
492         arrLoadbank_LastSent[2] := arrLoadbank_UtData[2];
493     END_IF;
494
```

Input

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
xCharging	BOOL	Input										
uiLoadpower	UINT	Input										
xLoad	BOOL	Input										
xSendNow	BOOL	Input										

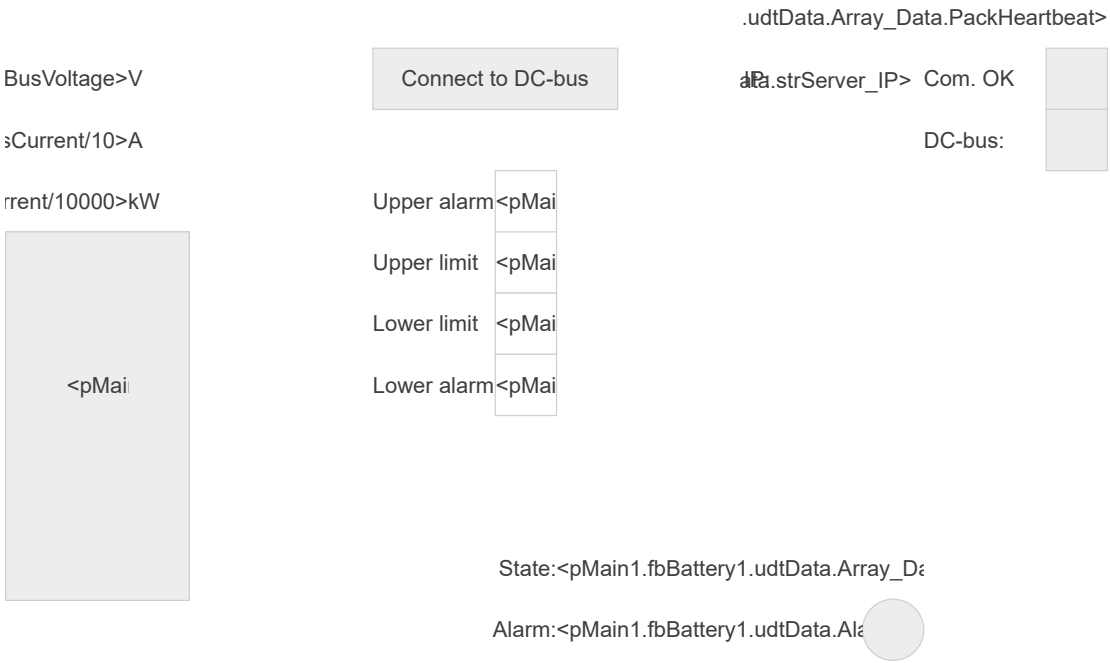
Output

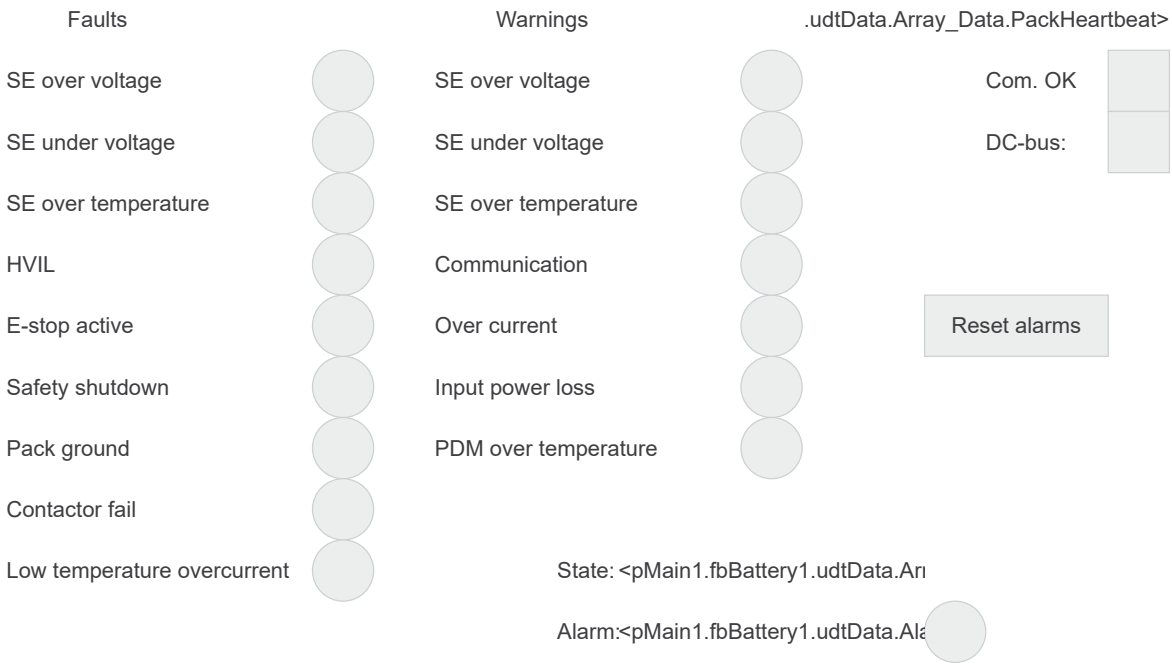
Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
uiPowerLoaded	UINT	Output										
uiPowerMeasured	UINT	Output										
iAlarmLoadbank	INT	Output										
xLoadbank_Loadstate	BOOL	Output										
xComOK	BOOL	Output										
uiPowerLoaded1	UINT	Local				Private						
uiPowerMeasured1	UINT	Local				Private						
iAlarmLoadbank1	INT	Local				Private						
xLoadbank_Loadstate1	BOOL	Local				Private						
xComOK1	BOOL	Local				Private						

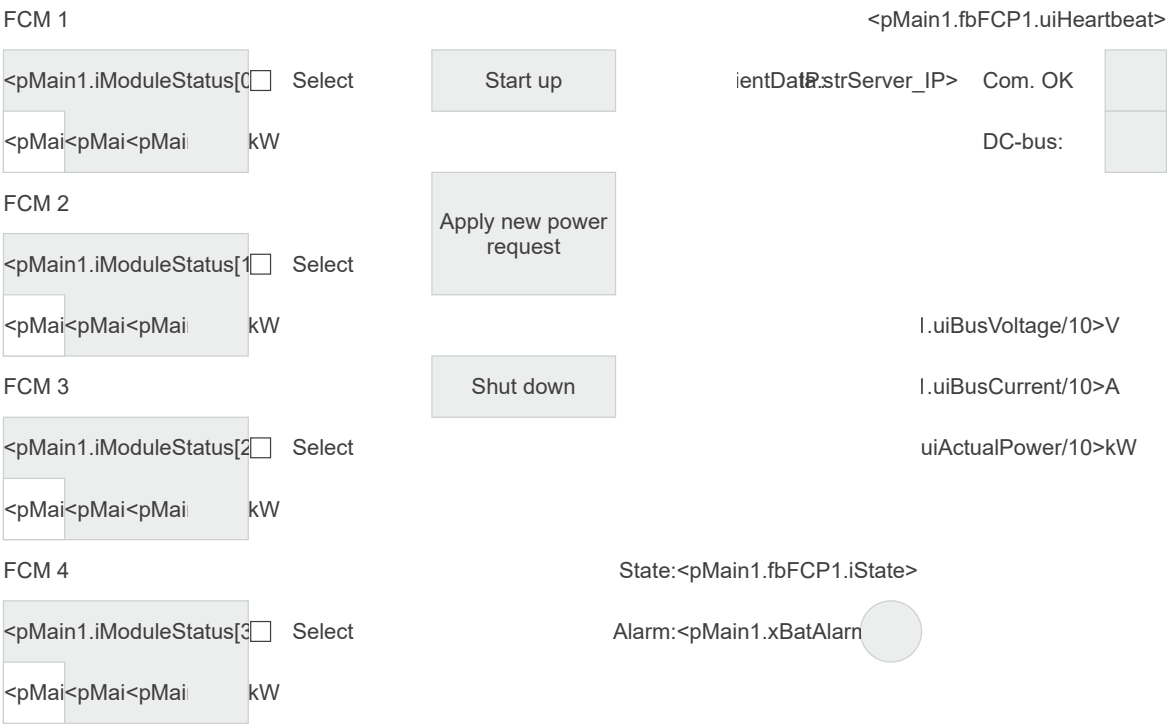
Components / fbLoadbankSim / Code

```
1   (*Simple simulator to give values to and from loadbank function block, used by attaching to the PLCruntime*)
2
3   uiPowerLoaded := uiPowerLoaded1;
4   uiPowerMeasured := uiPowerMeasured1;
5   xComOK := xComOK1;
6   iAlarmLoadbank := iAlarmLoadbank1;
7   xLoadbank_Loadstate := xLoadbank_Loadstate1;
```

Vedlegg E5 - Kode for HMI

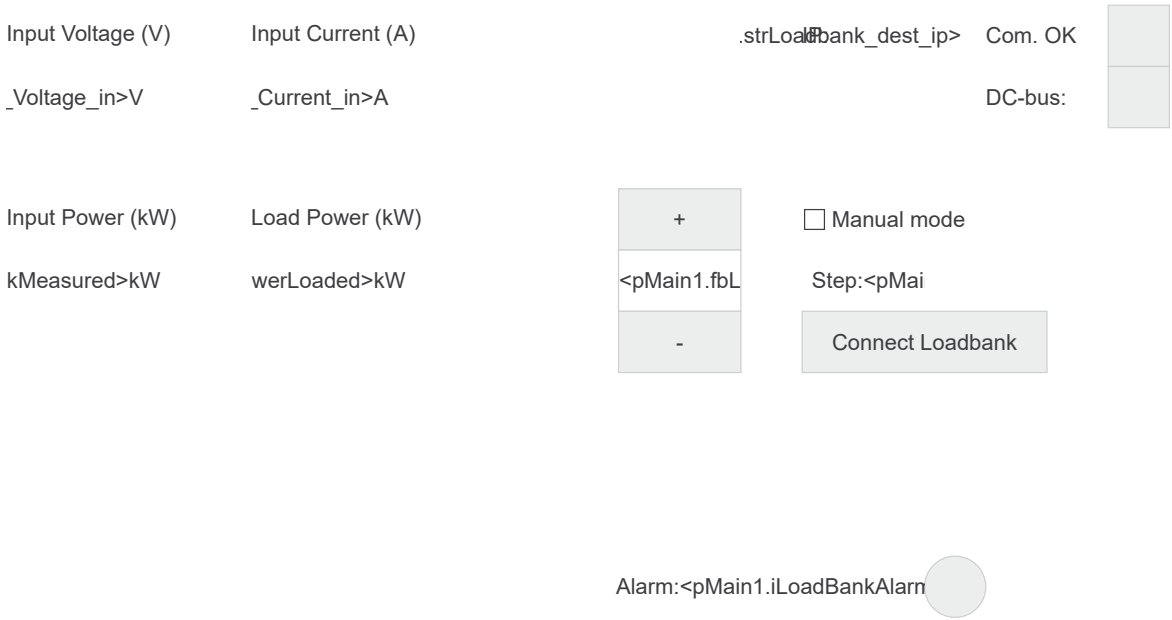

.udtData.Array_Data.PackHeartbeat>
BusVoltage>V
Connect to DC-bus
ata.strServer_IP>
Com. OK
sCurrent/10>A
DC-bus:
rrent/10000>kW
Upper alarm
<pMai
Upper limit
<pMai
Lower limit
<pMai
<pMai
Lower alarm
<pMai
State:<pMain1.fbBattery1.udtData.Array_Da
Alarm:<pMain1.fbBattery1.udtData.Ala

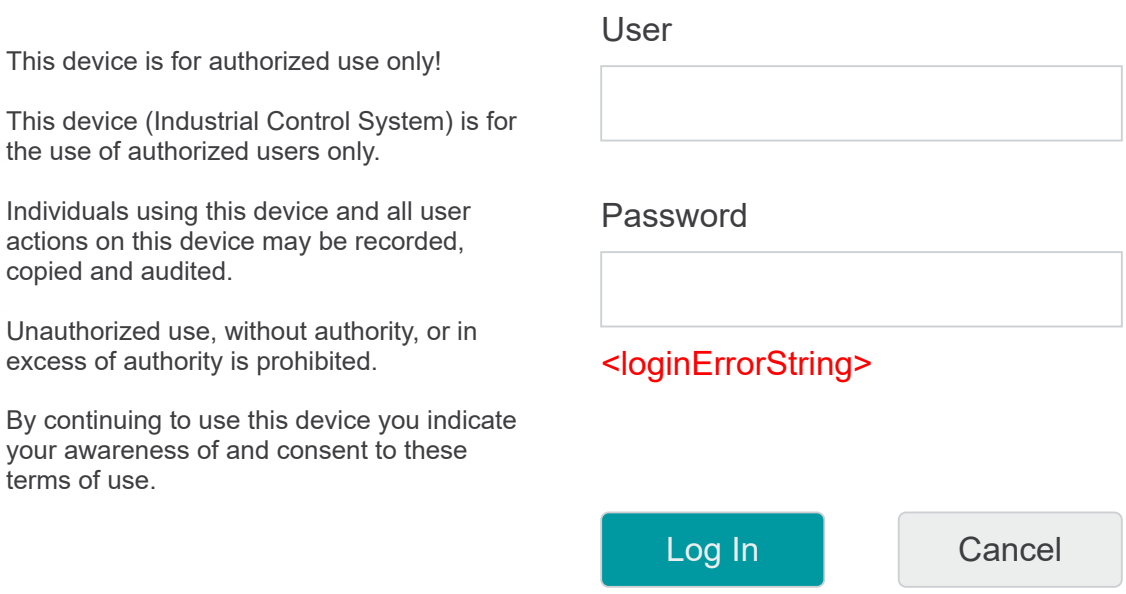

Faults
Warnings
.udtData.Array_Data.PackHeartbeat>
SE over voltage
SE under voltage
SE over temperature
HVIL
E-stop active
Safety shutdown
Pack ground
Contactor fail
Low temperature overcurrent
SE over voltage
SE under voltage
SE over temperature
Communication
Over current
Input power loss
PDM over temperature
Com. OK
DC-bus:
Reset alarms
State: <pMain1.fbBattery1.udtData.Arı
Alarm:<pMain1.fbBattery1.udtData.Ala

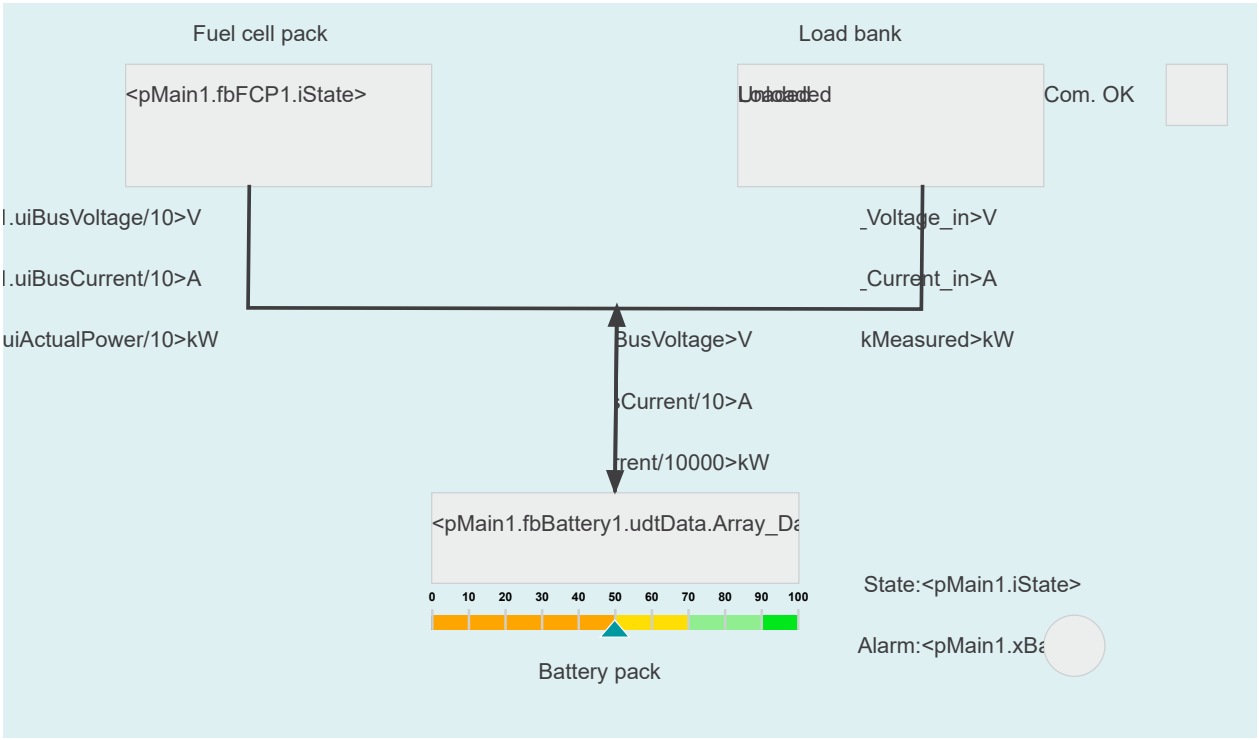

FCM 1
<pMain1.fbFCP1.uiHeartbeat>
<pMain1.iModuleStatus[0
Select
Start up
ientDataP.strServer_IP>
Com. OK
<pMai<pMai<pMai
kW
DC-bus:
FCM 2
Apply new power request
<pMain1.iModuleStatus[1
Select
<pMai<pMai<pMai
kW
l.uiBusVoltage/10>V
Shut down
FCM 3
l.uiBusCurrent/10>A
<pMain1.iModuleStatus[2
Select
uiActualPower/10>kW
<pMai<pMai<pMai
kW
FCM 4
State:<pMain1.fbFCP1.iState>
<pMain1.iModuleStatus[3
Select
Alarm:<pMain1.xBatAlarm
<pMai<pMai<pMai
kW

Array location:

	<pMai	<pMai<pMai<pMai<pMai<pMai	☐ Automatic mode
Duration (s):	<pMai	<pMai<pMai<pMai<pMai<pMai	
Module 1 power (kW):	<pMai	<pMai<pMai<pMai<pMai<pMai	
Module 2 power (kW):	<pMai	<pMai<pMai<pMai<pMai<pMai	
Module 3 power (kW):	<pMai	<pMai<pMai<pMai<pMai<pMai	
Module 4 power (kW):	<pMai	<pMai<pMai<pMai<pMai<pMai	


Input Voltage (V)
Input Current (A)
.strLoadBbank_dest_ip>
Com. OK
_Voltage_in>V
_Current_in>A
DC-bus:
Input Power (kW)
Load Power (kW)
+
Manual mode
kMeasured>kW
werLoaded>kW
<pMain1.fbL
Step:<pMai
-
Connect Loadbank
Alarm:<pMain1.iLoadBankAlarm


This device is for authorized use only!
This device (Industrial Control System) is for the use of authorized users only.
Individuals using this device and all user actions on this device may be recorded, copied and audited.
Unauthorized use, without authority, or in excess of authority is prohibited.
By continuing to use this device you indicate your awareness of and consent to these terms of use.
User
Password
<loginErrorString>
Log In
Cancel


Fuel cell pack
<pMain1.fbFCP1.iState>
Load bank
Com. OK
.uiBusVoltage/10>V
.uiBusCurrent/10>A
uiActualPower/10>kW
_Voltage_in>V
_Current_in>A
kMeasured>kW
BusVoltage>V
Current/10>A
rent/10000>kW
<pMain1.fbBattery1.udtData.Array_Da
0 10 20 30 40 50 60 70 80 90 100
State:<pMain1.iState>
Alarm:<pMain1.xBa
Battery pack

Vedlegg E6 - Kode til Modbus-kommunikasjon

Components / Functions / fbReadFromRegisters / Code

```
(*
This function block reads from registers defined in its inputs and returns data while handling errors it encounters.
*)

xNewData := FALSE;

IF  NOT udtFC3Parameters.xActive THEN //First run
    IF udtFC3Parameters.xActivate = FALSE THEN
        udtFC3Parameters.xActivate := TRUE;

    ELSIF udtFC3Parameters.xError THEN //Handles errors
        udtFC3Parameters.xAcknowledge := TRUE;

    END_IF;

END_IF;

(*Runs the Modbus FC3 function block*)
ReadFC3(xActivate             :=  udtFC3Parameters.xActivate,
            xAcknowledge            :=  udtFC3Parameters.xAcknowledge,
            iMT_ID                  :=  udtFC3Parameters.iMT_ID,
            tUpdateTime             :=  udtFC3Parameters.tUpdateTime,
            uiUnitIdentifier        :=  udtFC3Parameters.uiUnitIdentifier,
            wStartRegister          :=  udtFC3Parameters.wStartRegister,
            uiQuantityOfRegisters   :=  udtFC3Parameters.uiQuantityOfRegisters,
            xActive                 =>  udtFC3Parameters.xActive,
            xNDR                    =>  udtFC3Parameters.xNDR,
            uiByteCount             =>  udtFC3Parameters.uiByteCount,
            arrRegisterValue        =>  udtFC3Parameters.arrRegisterValue,
            xError                  =>  udtFC3Parameters.xError,
            wDiagCode               =>  udtFC3Parameters.wDiagCode,
            wAddDiagCode            =>  udtFC3Parameters.wAddDiagCode,
            udtTCP_ComData          :=  ComData);


udtFC3Parameters.xAcknowledge := FALSE;

(*Checks if it has recieved the correct amount of bytes*)
IF udtFC3Parameters.xNDR THEN
    IF udtFC3Parameters.uiByteCount = udtFC3Parameters.uiQuantityOfRegisters*2 THEN
        xNewData := TRUE;

    ELSE
        //Unexpected error

    END_IF;

ELSE
    //No new data

END_IF;
```

Default

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
ComData	MB_TCP_UDT_COMMUNICATION	InOut		Communication data used by Modbus functions								
udtFC3Parameters	udtMB_TCP_FC3	InOut		Parameters for connection to client								
xNewData	BOOL	Output		Indicates new data.								
ReadFC3	MB_TCP_FC3_3	Local		Function block that reads Modbus holding registers		Private						

Default

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
WriteFC16	MB_TCP_FC16_1	Local		Function block used to write Modbus holding registers		Private						
uiHeartbeat	UINT	Input		Last heartbeat								
xWriteSuccess	BOOL	Output		Indicates write command recieved by Modbus server								
ComData	MB_TCP_UDT_COMMUNICATION	InOut		Communication data used by Modbus functions								
udtFC16Parameters	udtMB_TCP_FC16	InOut		Parameters for connection to client								
xRecentWrite	BOOL	Local		Indicates that a recent write was performed		Private						
uiHeartbeatLastWrite	UINT	Local		Heartbeat when last write was performed		Private						
uiWaitTime	UINT	Local		Wait time between trying to send the same write command.	UINT#10	Private						

```
(*
This function block writes to registers defined in its inputs while handling errors it encounters.
It will not resend already sent data, and will wait for WriteFC16.xDone to returned as true, or waituntil a certain time has passed, as indicated by uiWaitTime
*)

WriteFC16(xActivate         := udtFC16Parameters.xActivate,
        udtTCP_ComData      := ComData);

WriteFC16.xAcknowledge  := FALSE;
xWriteSuccess           := FALSE;

IF  NOT WriteFC16.xActive AND ComData.ClientReady THEN //First run
    IF udtFC16Parameters.xActivate = FALSE THEN
        udtFC16Parameters.xActivate := TRUE;

        (*Initialization of function block*)
        WriteFC16(xActivate         := udtFC16Parameters.xActivate,
            iMT_ID                  := udtFC16Parameters.iMT_ID,
            tUpdateTime             := udtFC16Parameters.tUpdateTime,
            uiUnitIdentifier        := udtFC16Parameters.uiUnitIdentifier,
            wStartRegister          := udtFC16Parameters.wStartRegister,
            uiQuantityOfRegisters   := udtFC16Parameters.uiQuantityOfRegisters,
            udtTCP_ComData          := ComData);

    ELSIF WriteFC16.xError THEN //Handles errors
        WriteFC16.xAcknowledge := TRUE;

    END_IF;

ELSIF WriteFC16.xActive THEN    (*Active and running*)
    IF WriteFC16.xDone THEN
        xRecentWrite := FALSE;
        xWriteSuccess := TRUE;
    END_IF;

    IF NOT fcEqualArray(udtFC16Parameters.arrRegisterValue, WriteFC16.arrRegisterValue) THEN
        IF NOT xRecentWrite THEN
            WriteFC16(xActivate             := udtFC16Parameters.xActivate,
                      arrRegisterValue      := udtFC16Parameters.arrRegisterValue,
                      udtTCP_ComData        := ComData);
            xRecentWrite := TRUE;
            uiHeartbeatLastWrite := uiHeartbeat;
        ELSIF (uiHeartbeatLastWrite + uiWaitTime) > uiHeartbeat THEN
            xRecentWrite := FALSE;
        END_IF;
    END_IF;


END_IF;
```

Components / Functions / fbComCheck / Code

```
(*
This method checks if the Modbus communication between server and client is working and resolves errors
*)

IF ClientData.xActive THEN
    IF ClientData.xReady THEN
        xComOK := TRUE;
    ELSIF ClientData.xError THEN
        xComOK := FALSE;
    ELSE
        xComOK := FALSE;
    END_IF;
ELSE
    xComOK := FALSE;
    IF NOT ClientData.xActivate THEN
        ClientData.xActivate := TRUE;
    END_IF;

END_IF;
```

Default

Name	Type	Usage	Translate	Comment	Init	Access	Retain	Constant	OPC	HMI	Proficloud	I/Q
ClientData	udtMB_TCP_CLIENT	InOut		Communication data used by Modbus functions								
xComOK	BOOL	Output		Indicates succesfull communication with server								

Vedlegg E7 - Datatyper

Components / Arrays

```
TYPE
     ByteArray2     : ARRAY[0..2] OF BYTE;
     ByteArray18    : ARRAY[0..18] OF BYTE;
     BoolArray3     : ARRAY[0..3] OF BOOL;
     IntArray3      : ARRAY[0..3] OF INT;
     UintArray3     : ARRAY[0..3] OF UINT;
END_TYPE

```

```
TYPE
    uiPower2        : ARRAY[0..255] OF UINT;

udtFCAutoprogram : STRUCT

    uliSeqDur       : ARRAY[0..255] OF ULINT;
    uiPower         : ARRAY[0..3] OF uiPower2;

END_STRUCT

END_TYPE
```

Components / Battery command data

```
TYPE

BAT_Commandssss : STRUCT
    ArrayAlarmReset : BOOL;
    Field2 : BOOL;
END_STRUCT

END_TYPE
```

```
TYPE


BAT_Array_Data : STRUCT

    //Regiser 0 to 3
    NoPacksInArray      : UINT;
    NoPacksInNetwork    : UINT;
    NoPacksConnected    : UINT;
    NoPacksFaulted      : UINT;

    //Register 4 to 7
    BusVoltage          : UINT;
    BusCurrent          : INT;
    ArrayOnlineSOC      : UINT;
    ArrayOnlineSOH      : UINT;

    //Register 8 is reserved

    //Register 9 to 16
    ArrayChargeCurrentLimit     : UINT;
    ArrayDischargeCurrentLimit  : UINT;
    MaxCellTemperature          : INT;
    MinCellTemperature          : INT;
    MaxCellVoltage              : UINT;
    MinCellVoltage              : UINT;
    MaxUnconnectedPackVoltage   : UINT;
    MinUnconnectedPackVoltage   : UINT;

    //Register 17 and 18: Pack heartbeat
    PackHeartbeat       : UDINT;

    //Register 19 to 25
    ServiceState        : BOOL;
    BusCurrentUnscaled  : INT;
    MinPackSOC          : UINT;
    AvgPackSOC          : UINT;
    MaxPackSOC          : UINT;
    AvgConnectedCellSOC : UINT;
    MaxConnectedCellSOC : UINT;

    //Register 50: Pack Status
    PackOperationMode   : INT;
    ServiceRequired     : BOOL;

    //PackOperationMode in bool.
    OpModeNoData        : BOOL;
    OpModePowerSave     : BOOL;
    OpModeFault         : BOOL;
    OpModeReady         : BOOL;
    OpModeConnecting    : BOOL;
    OpModeConnected     : BOOL;
    OpModeNotReady      : BOOL;

END_STRUCT;

BAT_Commands : STRUCT
    //Register 360
    Pack1Connect        : BOOL;
    Pack1Select         : BOOL;

END_STRUCT

BAT_Alarms : STRUCT

    //Register 400: Alarm status
    Pack1AlarmStatus            : INT; //0: No alarms; 1: Fault; 2: Warning

    //Register 401: Faults
    xOverVoltFault              : BOOL;
    xUnderVoltFault             : BOOL;
    xOverTempFault              : BOOL;
    xHVILFault                  : BOOL;
    xEStopFault                 : BOOL;
    xSafetyShutdownFault        : BOOL;
    xPackGroundFault            : BOOL;
    xContactorFailFault         : BOOL;
```

```
78
79          xLowTempOvercurrentFault    : BOOL;
80          //xRepeatedOverChargeLockoutFault : BOOL; //Unimplemented
81          //xDeepOverDischargeLockoutFault : BOOL; //Unimplemented
82
83          //Register 403: Warnings
84          xOverVoltWarning            : BOOL;
85          xUnderVoltWaring            : BOOL;
86          xOverTempWarning            : BOOL;
87          xComWarning                 : BOOL;
88          xOverCurrentWarning         : BOOL;
89          xInputPowerLossWarning      : BOOL;
90          xPdmOverTemperatureWarning  : BOOL;
91
92          xFuseFailureWarning         : BOOL;
93
94
95      END_STRUCT
96
97      BAT_Data : STRUCT
98          Alarms      : BAT_Alarms;
99          Array_Data  : BAT_Array_Data;
100     END_STRUCT
101
102     END_TYPE
```

```
TYPE
    ByteArray35 : ARRAY[0..35] OF BYTE;
END_TYPE
TYPE
    IP_ADDRESS_ARRAY : ARRAY[0..3] OF BYTE;
END_TYPE
TYPE
    EIPD_IO_ARRAY : ARRAY[0..127] OF WORD;
END_TYPE
TYPE
HMI_STATUS_STRUCT2 : STRUCT
    SESSION_ID : STRING;
    STATION_ID : STRING;
    LAST_REQ : LINT;
    IP_ADDRESS : IP_ADDRESS_ARRAY;
    STATION_NUM : UINT;
END_STRUCT
END_TYPE
TYPE
HMI_CONTROL_STRUCT : STRUCT
    DISABLE : BOOL;
END_STRUCT
END_TYPE
TYPE
    HMI_STATUS_ARRAY2 : ARRAY[1..256] OF HMI_STATUS_STRUCT2;
END_TYPE
TYPE
    HMI_CONTROL_ARRAY : ARRAY[1..256] OF HMI_CONTROL_STRUCT;
END_TYPE
TYPE
HMI_STATUS_TYPE2 : STRUCT
    CLIENT_COUNT : UINT;
    CLIENTS : HMI_STATUS_ARRAY2;
END_STRUCT
END_TYPE
TYPE
HMI_CONTROL_TYPE : STRUCT
    CLIENTS : HMI_CONTROL_ARRAY;
END_STRUCT
END_TYPE
TYPE
TASK_INFO : STRUCT
    INTERVAL : LINT;
    PRIORITY : INT;
    WATCHDOG : LINT;
    LAST_EXEC_DURATION : LINT;
    MIN_EXEC_DURATION : LINT;
    MAX_EXEC_DURATION : LINT;
    LAST_ACTIVATION_DELAY : LINT;
    MIN_ACTIVATION_DELAY : LINT;
    MAX_ACTIVATION_DELAY : LINT;
    EXEC_TIME_THRESHOLD : LINT;
    EXEC_TIME_THRESHOLD_CNT : UDINT;
    NAME : STRING;
END_STRUCT
END_TYPE
TYPE
    TASK_INFO_ARRAY : ARRAY[1..16] OF TASK_INFO;
END_TYPE
TYPE
ESM_INFO : STRUCT
    TASK_COUNT : UINT;
    TICK_COUNT : UDINT;
    TICK_INTERVAL : UDINT;
    TASK_INFOS : TASK_INFO_ARRAY;
END_STRUCT
END_TYPE
TYPE
    STRING512 : STRING[512];
END_TYPE
TYPE
    ESM_INFO_ARRAY : ARRAY[1..2] OF ESM_INFO;
END_TYPE
TYPE
ESM_EXCEPTION_INFO : STRUCT
    TYPE_ID : UDINT;
    SUB_TYPE : STRING512;
```

```
78          SUB_TYPE_ID : UDINT;
79          TASK_NAME : STRING;
80          PROGRAM_NAME : STRING512;
81          INFORMATION : STRING512;
82      END_STRUCT
83      END_TYPE
84      TYPE
85          ESM_EXCEPTION_INFO_ARRAY : ARRAY[1..2] OF ESM_EXCEPTION_INFO;
86      END_TYPE
87      TYPE
88      ESM_DAT : STRUCT
89          ESM_COUNT : USINT;
90          ESM_INFOS : ESM_INFO_ARRAY;
91          EXCEPTION_COUNT : USINT;
92          EXCEPTION_INFOS : ESM_EXCEPTION_INFO_ARRAY;
93      END_STRUCT
94      END_TYPE
95      TYPE
96          PND_IO_512 : ARRAY[0..511] OF BYTE;
97      END_TYPE
98      TYPE
99      RTC_TYPE : STRUCT
100         HOURS : USINT;
101         MINUTES : USINT;
102         SECONDS : USINT;
103         DAY : USINT;
104         MONTH : USINT;
105         YEAR : UINT;
106     END_STRUCT
107     END_TYPE
108     TYPE
109         CPU_LOAD_PER_CORE_ARRAY : ARRAY[1..2] OF USINT;
110     END_TYPE
111     TYPE
112     DEVICE_STATE_2152_TYPE : STRUCT
113         BOARD_TEMPERATURE : SINT;
114         reserved1 : BOOL;
115         reserved2 : USINT;
116         CPU_LOAD_ALL_CORES : USINT;
117         CPU_LOAD_PER_CORE : CPU_LOAD_PER_CORE_ARRAY;
118     END_STRUCT
119     END_TYPE
120     TYPE
121     PARTITION_INFO : STRUCT
122         MEM_TOTAL : ULINT;
123         MEM_FREE : ULINT;
124         MEM_USED : ULINT;
125         MEM_USAGE : USINT;
126     END_STRUCT
127     END_TYPE
128
```

```
TYPE
    udtMB_TCP_SERVER    : STRUCT
    (* Inputs *)
        xActivate           : BOOL;
        xAcknowledge        : BOOL;
        xAutoAck            : BOOL;
        strDestIp           : STRING;
        uiDestPort          : UINT;
        uiBindPort          : UINT;
        strBindIp           : STRING;
        xUDP_Mode           : BOOL;
        tReconnectDelay     : TIME;
        tTimeout            : TIME;
    (* Outputs *)
        xActive             : BOOL;
        xConnected          : BOOL;
        xError              : BOOL;
        wDiagCode           : WORD;
        wAddDiagCode        : WORD;
        udtDiag             : MB_TCP_UDT_SER_DIAG;
    (*InOut not included*)
    END_STRUCT;

    udtMB_TCP_CLIENT    : STRUCT
    (* Inputs *)
        xActivate           : BOOL;
        xAcknowledge        : BOOL;
        xAutoAck            : BOOL;
        strServer_IP        : STRING;
        iPort               : INT;
        uiBindPort          : UINT;
        strBindIp           : STRING;
        xUDP_Mode           : BOOL;
        tReconnectDelay     : TIME;
        tTimeout            : TIME;
    (* Outputs *)
        xActive             : BOOL;
        xReady              : BOOL;
        xError              : BOOL;
        wDiagCode           : WORD;
        wAddDiagCode        : WORD;
        udtDiag             : MB_TCP_UDT_CLI_DIAG;
    (*InOut not included*)
    END_STRUCT;

    udtMB_TCP_FC3    : STRUCT
    (* Inputs *)
        xActivate                   : BOOL;
        xAcknowledge                : BOOL;
        iMT_ID                      : INT;
        tUpdateTime                 : TIME;
        uiUnitIdentifier            : UINT;
        wStartRegister              : WORD;
        uiQuantityOfRegisters       : UINT;
    (* Outputs *)
        xActive                     : BOOL;
        xNDR                        : BOOL;
        uiByteCount                 : UINT;
        arrRegisterValue            : MB_TCP_ARR_W_1_125;
        xError                      : BOOL;
        wDiagCode                   : WORD;
        wAddDiagCode                : WORD;
    (*InOut not included*)
    END_STRUCT;

    udtMB_TCP_FC16    : STRUCT
    (* Inputs *)
        xActivate               : BOOL;
        xAcknowledge            : BOOL;
        iMT_ID                  : INT;
        tUpdateTime             : TIME;
        uiUnitIdentifier        : UINT;
        wStartRegister          : WORD;
        uiQuantityOfRegisters   : UINT;
        arrRegisterValue        : MB_TCP_ARR_W_1_125;
    (* Outputs *)
        xActive                 : BOOL;
```

```
78          xDone                   : BOOL;
79          xError                  : BOOL;
80          wDiagCode               : WORD;
81          wAddDiagCode            : WORD;
82      (*InOut not included*)
83      END_STRUCT;
84
85      udtMB_TCP_FC16Array  : ARRAY[0..3] OF udtMB_TCP_FC16;
86
87      MB_ClientFC3FC16    : STRUCT
88          udtTCP_ComData          : MB_TCP_UDT_COMMUNICATION; //InOut on all
89          Client                  : udtMB_TCP_CLIENT;
90          FC3                     : udtMB_TCP_FC3;
91          FC16                    : udtMB_TCP_FC16;
92      END_STRUCT;
93
94  END_TYPE
```