

## Teknisk rapport - tankregulering

<b>Oppgavens tittel:</b> Hovedprosjekt  P2103?!	<b>Dato:</b> 14.05.21
<b>Gruppedeltakere:</b> P2103  Håvard Bonde  +5 andre (Sensurert i tilfelle de ikke vil deles på nett 😊)	<b>Veileder:</b>
<b>Fagkode:</b> IELET2104 Automatiseringsprosjekt	

## Sammendrag

Denne rapporten beskriver hvordan en programmerer og setter i drift nivåregulering av en vanntankrigg. Dette inkluderer kommunikasjonsoppsett, to brukergrensesnitt og en PID-regulator med foroverkobling (et Lead-Lag element), samt alarmaksjoner. Alle programkoder finnes vedlagt.

PID-regulator, foroverkoblingsfilter og ventilfilter er utviklet som programkode inne i PLS-en. PID-regulator og foroverkoblingsfilter er implementert ifra differenslikninger som er utledet fra overføringsfunksjonene for de to regulatortypene.

Brukergrensesnittet er utviklet i samråd med kunden og bransjestandard for HMI, og ved hjelp av dette kan tankriggen styres og overvåkes. I tillegg finnes det et system for feilhåndtering med tilhørende nødhandlinger og alarmer. Dette systemet indikerer feilsituasjoner som oppstår og skiller mellom to alvorligtnivåer: kritisk og ikke-kritisk feil.

Alarmer kan kvitteres både enkeltvis og samlet fra panelet ved riggen, samt via HMI på pc. Alle alarmer og prosessverdier lagres slik at en kan se tilbake på historiske data.

## Innholdsfortegnelse

1	Innledning.....	1
2	Akronymer og forkortelser.....	2
3	2	
4	Teori.....	2
4.1	Diskretisering av kontinuerlige uttrykk .....	3
4.1.1	Z-transformasjon.....	3
4.1.2	Tidsforskyvning i Z-domenet .....	3
4.2	Differenslikninger .....	3
4.2.1	Alternativer avbildninger.....	4
4.2.2	Tustins metode.....	4
4.3	PID-regulator .....	4
4.4	PID - Diskretisering .....	5
4.5	Lead-Lag.....	5
4.6	Lead-Lag – Diskretisering.....	6
4.7	Tracking .....	6
4.8	Ulineært ventilpådrag .....	7
4.9	Kommunikasjon.....	8
4.9.1	Master .....	9
4.9.2	Slave .....	10
4.9.3	GX Configurator.....	11
4.9.4	KEPServer .....	12
4.9.5	InTouch.....	12
4.9.6	IX Developer .....	12
5	Metode .....	13
5.1	Kommunikasjon.....	13
5.1.1	Slave .....	13
5.1.2	Master .....	15
5.1.3	KEPServer .....	21
5.1.4	Intouch.....	21
5.1.5	IX Developer .....	21
5.2	Initieringsprogram.....	22
5.3	Lesing av transmittere.....	22

5.3.1	Skalering .....	22
5.3.2	SMA-filter (Simple Moving Average).....	22
<b>5.4</b>	<b>PID - Implementering .....</b>	<b>24</b>
5.5	Lead-Lag – Implementering.....	26
5.6	PID og Lead-Lag i GX Works.....	28
5.7	Rykkfri overgang.....	30
5.7.1	Tracking .....	30
5.7.2	Rykkfri overgang - implementering .....	31
5.7.3	Rykkfri overgang ved parameterskifte .....	31
5.8	Ulineært ventilpådrag - Ventilfilter .....	33
5.9	Valg av samplingstid .....	35
5.10	Prosessmodell i Matlab (Simulink) .....	35
<b>5.11</b>	<b>Alarmdeteksjoner .....</b>	<b>38</b>
5.11.1	Nivåalarm (HH, H, L, LL) .....	38
5.11.2	Kommunikasjonsbrudd: LT01 og FT01 .....	39
5.11.3	Kommunikasjonsbrudd: Profibus, Master.....	39
5.11.4	Kommunikasjonsbrudd: Profibus, Slave.....	39
5.11.5	Kommunikasjonsbrudd mellom PLS og HMI .....	40
5.11.6	Deteksjon av strømbrudd.....	41
5.11.7	Høy samplingstid .....	41
5.11.8	Stasjonært avvik .....	42
5.12	Alarmaksjoner .....	42
<b>5.13</b>	<b>Brukergrensesnitt.....</b>	<b>45</b>
5.13.1	High Performance HMI .....	45
5.13.2	HMI: InTouch .....	45
5.13.3	Oversikt .....	48
5.13.4	Alarmvisualisering i InTouch.....	49
<b>5.14</b>	<b>HMI: IX Developer .....</b>	<b>53</b>
1.1.1	Oversikt over IX Developer HMI .....	53
5.14.1	Alarmvisualisering: IX Developer.....	56
<b>6</b>	<b>Resultater .....</b>	<b>58</b>
6.1	Kommunikasjon.....	58
6.2	HMI .....	58
6.2.1	Hierarki .....	58
6.2.2	Tilgangsnivåer .....	58

6.3	HMI: Alarmer – InTouch .....	61
6.3.1	Alarmsituasjoner .....	61
6.3.2	Alarmaksjoner .....	62
6.4	HMI: Alarmer – IX Developer .....	66
6.5	Sprangresponser.....	68
6.5.1	Prosessmodell i Simulink .....	68
6.5.2	PI-regulering .....	69
6.5.3	PI + Statisk Foroverkobling .....	70
6.5.4	PID-regulering.....	71
6.5.5	PID + Statisk Foroverkobling.....	72
6.5.6	PID + Lead-Lag .....	73
7	Diskusjon og konklusjon .....	74
7.1	Kommunikasjon.....	74
7.1.1	Kvitteringsutfordringer .....	74
7.2	Sprangresponser.....	74
7.2.1	Prosessmodell i Simulink .....	75
7.2.2	PI-regulering .....	75
7.2.3	PI + Statisk Foroverkobling .....	75
7.3	PID .....	75
7.3.1	PID + Statisk Foroverkobling.....	75
7.3.2	PID + Lead-Lag .....	76
7.4	Regulatorer.....	76
7.5	Rykkfri overgang.....	76
7.5.1	Valg av samplingstid .....	76
7.6	Brukergrensesnitt.....	77
8	Referanser .....	I
9	Vedlegg: Tagname-lister.....	III
9.1	Tabell 1: Oversikt over innholdet i KEPServer-filen.....	III
10	Tabell 2: Oversikt over tags for dataoverføring i Intouch .....	III
10.1	Tabell 3: Oversikt over interne tags i InTouch.....	V
10.2	Tabell 4: Oversikt over tags i IX Developer.....	V
11	Vedlegg: PID-Kode .....	VII
12	Oversikt over eksterne vedlegg.....	IX

## 1 Innledning

Formålet med dette prosjektet er i korte trekk å programmere og sette i drift en tankrigg. Utfyllende informasjon om prosjektoppgaven finnes i «Arbeidsnotat Forprosjekt» (Bonde, et al., 2021).

Løsningen er utviklet i samarbeid med kunden og i tråd med gitte føringer i forprosjektnotatet.

Tankriggen består av to vanntanker med pneumatisk og motorstyrt ventil, pumpe, strømningsmåler og nivåmåler. PLS-riggen består av en master og to slave PLS, i tillegg til et styrepanel. I dette prosjektet skal kun den ene vanntanken og slave PLS-en brukes.

Prosessen skal kunne kontrolleres og overvåkes fra en PC, samt fra styrepanelet på PLS-riggen.

PC-en og styrepanelet skal kommunisere via Ethernet med en PLS av typen Mitsubishi Q00, som fungerer som en masterenhett. Masteren kommuniserer med to slaveenheter via PROFIBUS.

Slaveenheten er koblet til en AD/DA-omformere som brukes for å styre/regulere ventilene inn i en tank, samt lese av tankens nivå.

For å kunne regulere tanken må det også implementeres en PID-regulator og et tilhørende foroverkoblingsfilter.

## 2 Akronymer og forkortelser

HMI	Human Machine Interface. Brukergrensesnitt
SCADA	Supervisory Control and Data Acquisition. Brukergrensesnitt
PLS	Programmerbar Logisk Styring
AD/DA	Analog til Digital /Digital til Analog, tilleggsmodul i slave-PLS
TCP/IP	Transmission Control Protocol/Internet Protocol (kommunikasjonsprotokoller).
Kp	Proporsjonalforsterkning til en regulator.
Ti	Integraltid
Td	Derivasjonstid
KLL	Forsterkningskonstant for Lead-Lag (Fovoverkobling)
Tt	Tidskonstant i teller for Lead-Lag
Tn	Tidskonstant i nevner for Lead-Lag
PV	Prosessverdi
MV	Manipulated value. Pådrag fra regulatoren.
SP	Settpunkt, referanse
PID	(Proporsjonal-Integral-Derivat) Regulator
LL	Lead-Lag, forovekobling for forstyrrelse
CSV	Comma Separated Values
OPC	Open Platform Communications

## 3 Teori

Mye av informasjonen i dette kapitlet er utdrag fra to tidligere arbeidsnotat. For mer detaljert informasjon, se Arbeidsnotat Forprosjekt (Bonde, et al., 2021) og Arbeidsnotat PID og Lead-Lag funksjonsblokk (Halse, et al., 2021).

### 3.1 Diskretisering av kontinuerlige uttrykk

#### 3.1.1 Z-transformasjon

Z-transformasjon benyttes for å enklere kunne jobbe og regne i det diskrete planet. Den komplekse variabelen  $z$  er definert som

$$z = e^{sT}$$

#### 3.1.2 Tidsforskyvning i Z-domenet

Tidsforskyvning i Z-domenet kan enkelt konverteres til tidsforskyvning i en tallsekvens med sammenhengen

$$Z\{f(k - n)\} = z^{-n} \cdot F(z)$$

### 3.2 Differenslikninger

Gitt et generelt diskret system beskrevet av differenslikningen

$$a_0y[k] + a_1y[k - 1] + \dots + a_my[k - m] = b_0u[k] + b_1u[k - 1] + \dots + b_nu[k - n]$$

er den diskrete overføringsfunksjonen gitt av (Anstenrud, 2021)

$$G(z) = \frac{Y(z)}{U(z)} = \frac{b_0 + b_1z^{-1} + \dots + b_nz^{-n}}{a_0 + a_1z^{-1} + \dots + a_mz^{-m}}$$

### 3.2.1 Alternativer avbildninger

Med bakoverdifferanse kan avbildningen mellom s-planet og z-planet uttrykkes som:

$$s = \frac{1 - z^{-1}}{T} \Leftrightarrow z = \frac{1}{1 - Ts}$$

Der T er tastetiden.

### 3.2.2 Tustins metode

Tustins metode, også kalt bilineær transformasjon, gir det beste forholdet i frekvensdomenet mellom kontinuerlig-tid og diskrete systemer. Dette uttrykkes ved:

$$s = \frac{2}{T} \frac{1 - z^{-1}}{1 + z^{-1}} \Leftrightarrow z = \frac{1 + sT/2}{1 - sT/2}$$

## 3.3 PID-regulator

Det er tatt utgangspunkt i regulatorlikningen i boken «Reguleringsteknikk» av K. Bjørvik og P. Hveem (Hveem & Bjørvik, 2014). Videre er det tatt utgangspunkt i sumform da denne anses som mer intuitiv og enklere å programmere enn produktform.

En PID-regulator på sum-form kan uttrykkes som

$$h_{PID}(s) = K_p \left( 1 + \frac{1}{T_i s} + \frac{T_d s}{1 + \frac{T_d}{n} s} \right)$$

### 3.4 PID - Diskretisering

En detaljert utledning av følgende uttrykk og likninger kan ses i Arbeidsnotat PID og Lead-Lag funksjonsblokk (Halse, et al., 2021).

#### **P – leddet**

$$p(z) = e(z) \cdot K_p$$

$$p[k] = K_p e[k]$$

#### **I – leddet**

$$\frac{i(z)}{e(z)} = K_p \frac{1}{T_i \left( \frac{2}{T} \frac{1-z^{-1}}{1+z^{-1}} \right)}$$

$$i[k] = \frac{K_p T}{T_i 2} (e[k] + e[k-1]) + i[k-1]$$

#### **D – leddet**

$$\frac{d(z)}{e(z)} = K_p \frac{\frac{T_d}{n} \left( \frac{2}{T} \frac{1-z^{-1}}{1+z^{-1}} \right)}{1 + \frac{T_d}{n} \left( \frac{2}{T} \frac{1-z^{-1}}{1+z^{-1}} \right)}$$

$$d[k] = \frac{2K_p T_d}{(2\tau + T)} (y[k] - y[k-1]) + \frac{(2\tau - T)}{(2\tau + T)} d[k-1]$$

#### **PID – leddene samlet**

$$u(z) = p(z) + i(z) + d(z)$$

$$u[k] = K_p e[k] + \frac{K_p T}{T_i 2} (e[k] + e[k-1]) + i[k-1] + \frac{2K_p T_d}{(2\tau + T)} (y[k] - y[k-1]) + \frac{(2\tau - T)}{(2\tau + T)} d[k-1]$$

$$\text{der } \tau = \frac{T_d}{n}, \quad k = \text{aktuell programsyklus} \quad \& \quad k-1 = \text{forrige programsyklus}$$

### 3.5 Lead-Lag

Et Lead-Lag-element er en type foroverkobling. Elementet har som oppgave å sende ut et pådrag, beregnet ifra enten settpunkt eller en forstyrrelse, for å motvirke endringer i prosessen. I motsetning til en statisk foroverkobling, reagerer Lead-lag raskere i begynnelsen av et sprang. Foroverkoblingen vil ha problemer med å lukke avviket helt og det er derfor naturlig å bruke Lead-lag elementet sammen med en PID-regulator (Dessen, Forelesningsnotater Foroverkobling, 2021).

Et Lead-Lag element utrykkes slik i s-domænet:

$$h_{LL}(s) = K_{LL} \frac{1 + T_t s}{1 + T_n s}$$

### 3.6 Lead-Lag – Diskretisering

En detaljert utledning av følgende uttrykk og likninger kan ses i Arbeidsnotat PID og Lead-Lag funksjonsblokk (Halse, et al., 2021).

#### Diskret overføringsfunksjon

$$h_{LL}(z) = K_{LL} \cdot \frac{1 + \frac{T_t \cdot (1 - z^{-1})}{T}}{1 + \frac{T_n \cdot (1 - z^{-1})}{T}}$$

#### LeadLag – pådrag

$$u[k] = r[k-1] * K_{LL} \frac{(T - 2T_t)}{(T + 2T_n)} + r[k] * K_{LL} \frac{(T + 2T_t)}{(T + 2T_n)} - u[k-1] \frac{(T - 2T_n)}{(T + 2T_n)}$$

**u[k]:** Pådrag fra LL-elementet

**r[k]:** Nåværende settpunkt

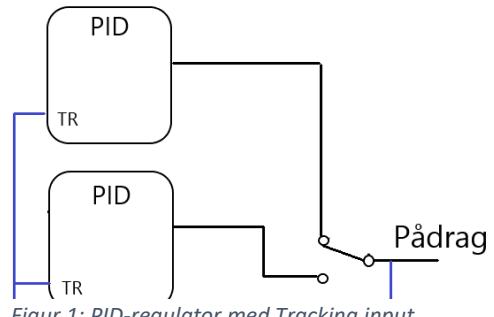
**r[k-1]:** Forrige settpunkt

**u[k-1]:** Forrige pådrag

### 3.7 Tracking

Man kan bruke «Tracking» for å oppnå rykkfri overgang mellom to regulatorer. I figuren ser man hvordan dette normalt vil implementeres mellom to regulatorer.

Regulatoren vil sammenligne den forrige syklusens pådrag med input TR (Tracking). Hvis de er ulike betyr det at en annen regulator har kontroll over pådraget. Da vil I-leddet oppdateres slik at pådraget til regulatorene blir like.



Figur 1: PID-regulator med Tracking input

$$I_{gain} = TR - P_{gain} - D_{gain}$$

### 3.8 Ulineært ventilpådrag

På tankriggen benyttes det en ventil for å kontrollere innstrømmingen i tanken, denne ventilen tilsvarer pådraget. Volumstrømmen ut fra pådragsventilen vil ikke være lineær ved endringer i ventilåpning og/eller pådrag. For å endre dette kan man linearisere pådraget ved å lage et filter som regner om pådraget slik at man får like endringer i volumstrøm gjennom hele skalaen fra 0 – 100 %. De to figurene nedenfor viser det reelle pådraget gitt i volumstrøm[L/s] ut fra ventil, ved et påtrykt pådrag fra 0 – 100%.

0	0
10	0.017
20	0.042
30	0.074
40	0.121
50	0.151
60	0.2
70	0.22
80	0.234
90	0.245
100	0.25

Figur 2: Ulineær pådragskarakteristikk

0	0
10	0.025
20	0.05
30	0.075
40	0.1
50	0.125
60	0.15
70	0.175
80	0.2
90	0.225
100	0.25

Figur 3: Lineær pådragskarakteristikk

Figur 2 viser pådrag fra ventilen uten filter (Dessen, Dataøving\_1 (Vedlegg 9), 2021). Figur 3 viser ønsket pådrag ut fra ventilen. Ved hjelp av målingene fra tabellen til venstre kan man lage en funksjon for et filter som gir ønsket pådrag for en gitt volumstrøm. Eksempelvis ønsker man ideelt sett å ha en volumstrøm på 0.2L/s med et pådrag på 80 %. Ventilen uten filter vil gi en volumstrøm på ca. 0.234L/s med et pådrag på 80 %. Ved å implementere et filter vil pådrag gjennom filteret bli 60 %, som vil gi riktig volumstrøm.

En lineær pådragskarakteristikk vil føre til en mer stabil regulering ved flere ulike pådragsnivåer. Et Lead-Lag elementet vil fungere vesentlig bedre med dette filteret, da det ikke vil fungere like bra for ulike pådragsnivåer med ulineær karakteristikk.

### 3.9 Kommunikasjon

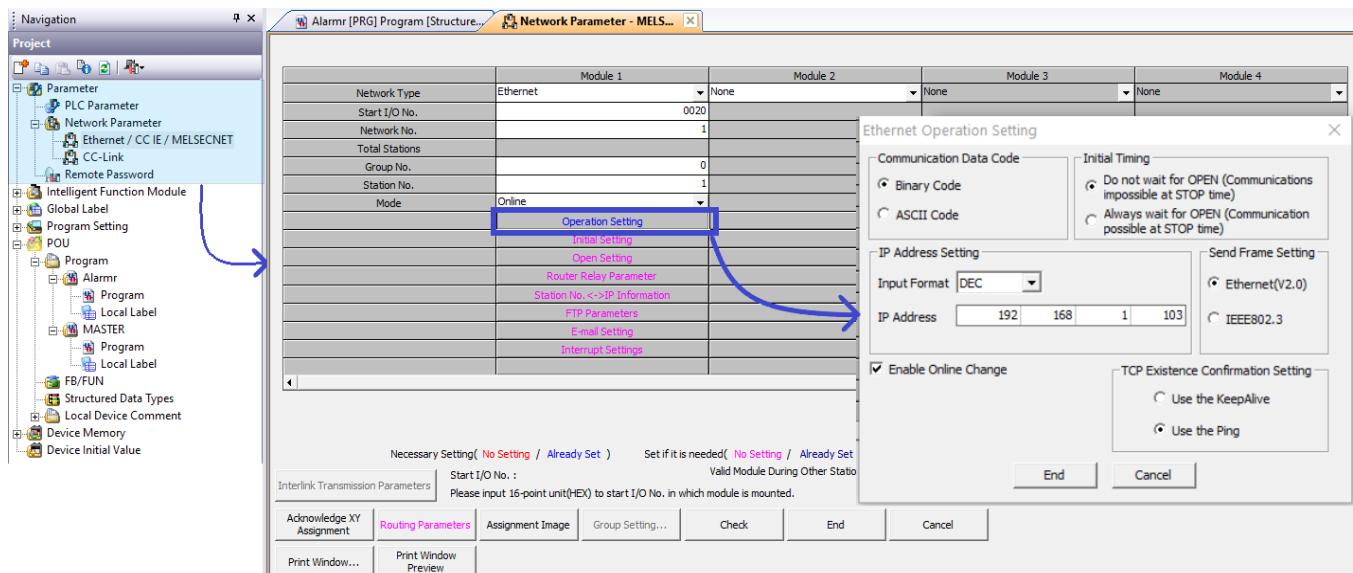
Kommunikasjonen mellom master og slave PLS går via Profibus, og kommunikasjonen fra master PLS til HMI går via Ethernet/WiFi. Kommunikasjonen via Profibus konfigureres i programmet GX-Configurator og kommunikasjonen med HMI via ethernet/WiFi konfigureres i KEPserverEX. Mye av informasjonen om kommunikasjonsoppsettet er hentet fra dokumentet «Guide til kommunikasjonsoppsett» (Arnesen, Sæther Ulvatne, & Engeseth Andersen, 2021).

### 3.9.1 Master

Master PLS-en er bindeleddet mellom slavene og brukergrensesnittene. På et større anlegg vil det være naturlig med mange slaver som kommuniserer med en felles master. Masteren tar for seg handlinger som er felles for alle slavene. I dette anlegget tar masteren hovedsakelig for seg behandling og videreføring av informasjon.

For at masteren skal kunne sende informasjon til og fra brukergrensesnittene må det settes opp en kommunikasjon mellom disse. Dette gjøres i GX Works 2 under «Parameter», «Network Parameter» og i dette tilfellet, «Ethernet». Innstillingene her settes som vist i figur 4, under «Module 1»

I «Operation Setting» kan man sette IP-adressen til masteren. For å kunne kommunisere med brukergrensesnittet må denne adressen være lik IP adressen til den tilhørende Profibus-modulen (Arnesen, Sæther Ulvatne, & Engeseth Andersen, 2021). Andre innstillinger settes lik slik som i figur 4 i «Ethernet Operation Setting» - vinduet.



Figur 4: "Network Parameter" - vindu i GX Works 2

Masteren vil nå være klar til å kommunisere med brukergrensesnittet.

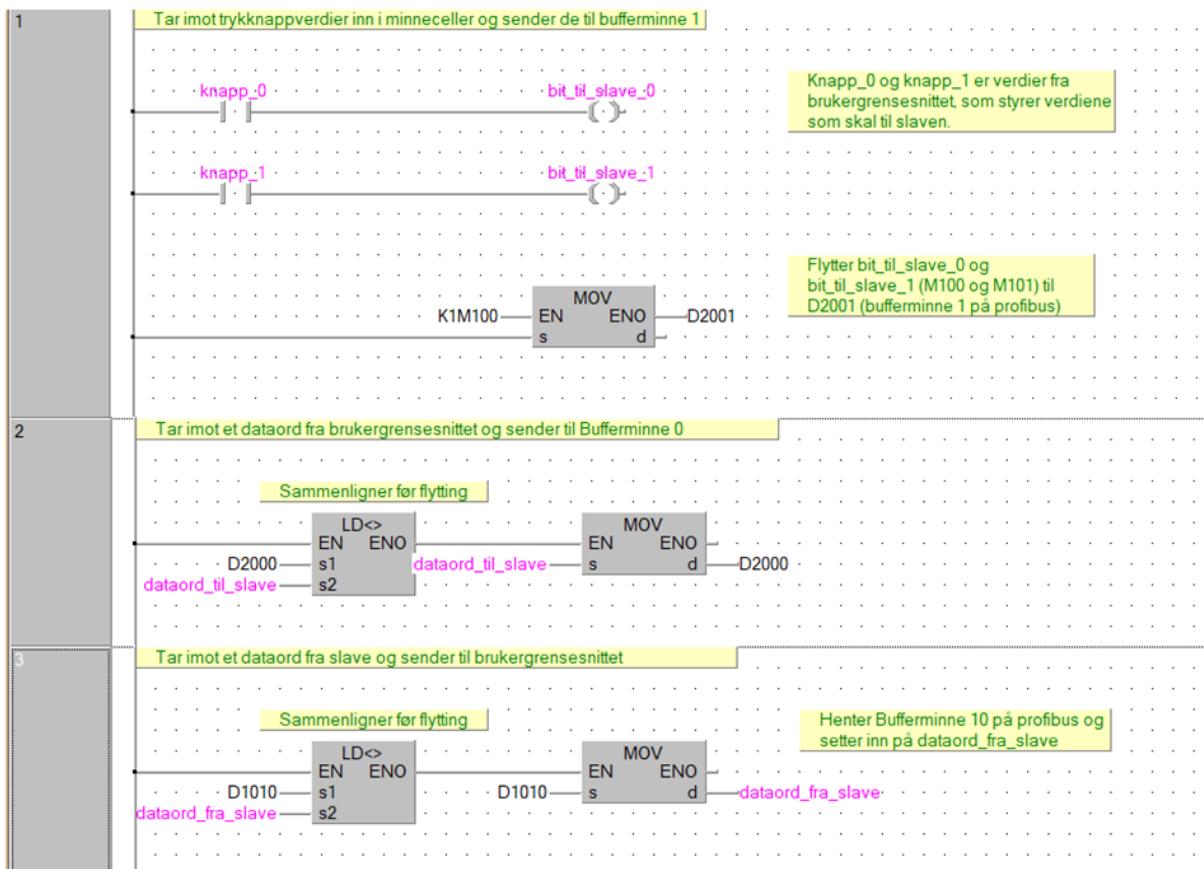
Overføringen av informasjon mellom KEPServer, IX-panelet og masteren fungerer ved hjelp av samsvarende minneceller og dataord. Overføring av informasjon mellom slave og master går gjennom bufferminnet til Profibus-modulene. Logikken som behøves for kommunikasjon må settes opp i et GX Works 2 program. Et eksempel på variabler i en slik kode er vist i figur 5.

	Class	Label Name	Data Type	Constant	Device	Address
1	VAR_GLOBAL	dataord_fra_slave	Word[Signed]	...	D0	%MW0.0
2	VAR_GLOBAL	dataord_til_slave	Word[Signed]	...	D20	%MW0.20
3	VAR_GLOBAL	knapp_0	Bit	...	M20	%MX0.20
4	VAR_GLOBAL	knapp_1	Bit	...	M21	%MX0.21
5	VAR_GLOBAL	bit_til_slave_0	Bit	...	M100	%MX0.100
6	VAR_GLOBAL	bit_til_slave_1	Bit	...	M101	%MX0.101

Figur 5: Oversikt over globale variabler i master-koden

Masteren kommuniserer med brukergrensesnittet via minneceller og dataord. Minneceller kan inneholde trykknappverdier (bits) og masteren skal motta og sende disse verdiene videre til bufferminnet på Profibus (16 bits). Et eksempel på hvordan dette gjøres er vist i nettverk 1 i figur 6.

Dataord som masteren mottar fra brukergrensesnittet må også flyttes over til bufferminne, men tar opp større plass (ett helt bufferminne). Et eksempel på hvordan dette gjøres er vist i nettverk 2 og 3 i figur 6.



Figur 6: Oppsett av hvordan masteren henter bits (Nettverk 1) og dataord (Nettverk 2 og 3), og sender disse videre til slaven.

### 3.9.2 Slave

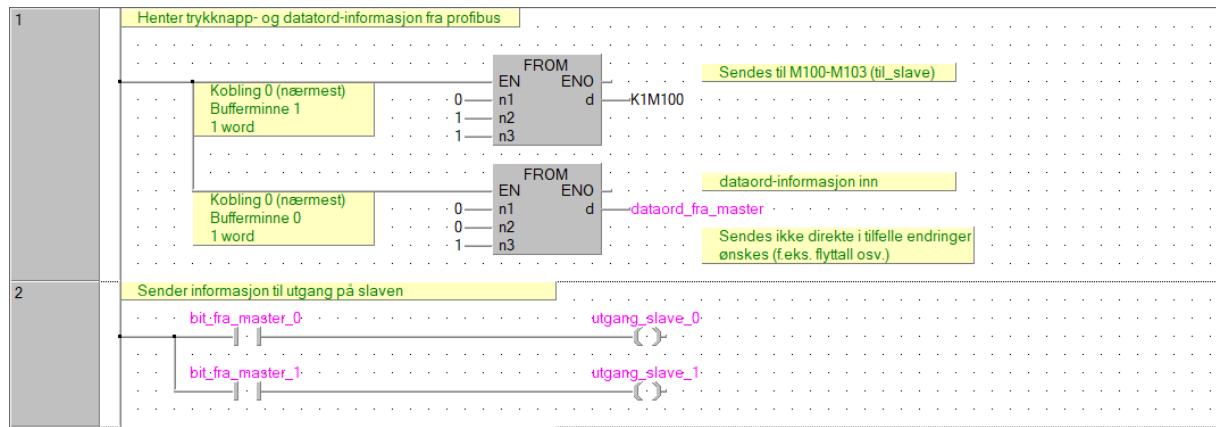
Slave PLS-en er utstyrt med en Profibus-modul og en AD/DA-omformer. Slaven kan kommunisere med Master PLS over Profibus. I tillegg kan den motta måleverdier fra transmittere, eller sende signaler til pådragsorganer via AD/DA-omformeren (analogkortet). I GX Works 2 vil dette tilsvare bits i form av minneceller og dataord.

Slaven benytter det samme bufferminnet som masteren, og må hente verdier derfra. Igjen må en slik logikk implementeres i GX Works 2, og et eksempel på variabler i en slik kode er vist i figur 7.

	Class	Label Name	Data Type	Constant	Device	Address
1	VAR_GLOBAL	utgang_slave_0	Bit	...	Y000	%QX0
2	VAR_GLOBAL	utgang_slave_1	Bit	...	Y001	%QX1
3	VAR_GLOBAL	bit_fra_master_0	Bit	...	M100	%MX0.100
4	VAR_GLOBAL	bit_fra_master_1	Bit	...	M101	%MX0.101
5	VAR_GLOBAL	anaDig	Word[Signed]	...	D8260	%MW0.8260
6	VAR_GLOBAL	digAna	Word[Signed]	...	D8262	%MW0.8262
7	VAR_GLOBAL	dataord_fra_master	Word[Signed]	...	D0	%MW0.0
8	VAR_GLOBAL	dataord_til_master	Word[Signed]	...	D1	%MW0.1

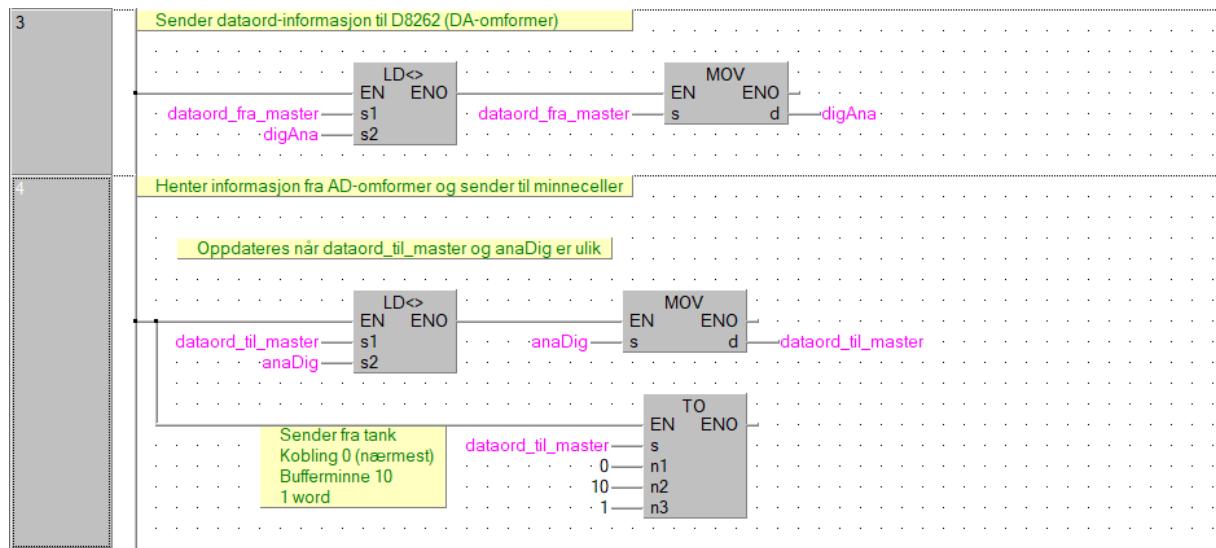
Figur 7: Oversikt over globale variabler i slave-koden.

Måten slaven henter verdier fra masteren er via FROM-blokker, der det bestemmes hvilket bufferminne slaven skal hente fra. Et eksempel på hvordan dette kan gjøres er vist i figur 8.



Figur 8: Oppsett av hvordan slaven henter bits og dataord fra masteren, og hvordan slaven benytter seg av denne informasjonen.

Slaven kan sende en digital verdi som et analogt signal (i form av strøm eller spennin), ved bruk av en DA-omformer. Dette signalet kan så en AD-omformer lese av som en digital verdi (dataord), som nå skal stemme overens med dataordet som først ble sendt ut, fra DA. Merk at DA- og AD-omformeren er i en og samme enhet. Et eksempel på dette er vist i figur 9.



Figur 9: Oppsett av hvordan slaven skriver et dataord (digital verdi) ut på DA-omformeren, og hvordan den leser av AD-omformeren og sender dette tilbake til masteren.

### 3.9.3 GX Configurator

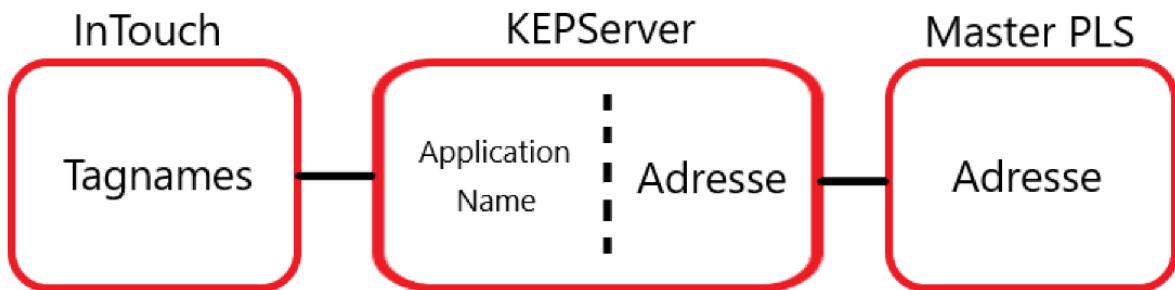
GX Configurator benyttes for å starte opp Profibus-kommunikasjonen mellom slave og master. Programmet konfigurerer blant annet slave-innstillinger, bufferminne-innstillinger og tilkoblingsmuligheter til masteren. Filen fra GX Configurator kan importeres til master PLS-en slik at programmet ikke trenger å bli lastet opp fra en ekstern PC. Dette står det mer om i metode-delen av rapporten.

### 3.9.4 KEPServer

KEPServer er en type OPC server, en server som knytter sammen ulike språk. Programmet kjøres i bakgrunnen på en PC som er koblet opp til WiFi/ethernet på PLS-riggen. KEPServer er bindeleddet mellom InTouch og Master PLS. For at programmene skal kommunisere med hverandre må de settes opp med samsvarende navn og adresser. I KEPServer må det opprettes et program, med tilhørende «Alias name» og «Application name» som senere må angis i InTouch for sammenkobling. Programmet kobles mot Master PLS-en og har tilgang til både lesing og skriving av PLS-en sine minneceller og dataord. Disse hentes ut til KEPServer hvor det nå kan legges til et «Tagname». Tagnames skal brukes i InTouch.

### 3.9.5 InTouch

InTouch er et brukergrensesnitt-program på PC, såkalt SCADA. Programmet kommuniserer med KEPServer gjennom bruk av samsvarende «Tagname». Application name i InTouch må også være likt som i KEPServer programmet.



Figur 10: Kommunikasjon mellom InTouch, KEPServer og Master PLS

### 3.9.6 IX Developer

Programmet brukes for å opprette et brukergrensesnitt til panelet på PLS-riggen. Dette skal også kommunisere med resten av anlegget via Master PLS-en. I IX-Developer trenger man kun adresse til minnecelle eller dataord direkte i Master PLS-en og dermed ikke noe «Tagname» eller tilkobling til KEPServer.

## 4 Metode

### 4.1 Kommunikasjon

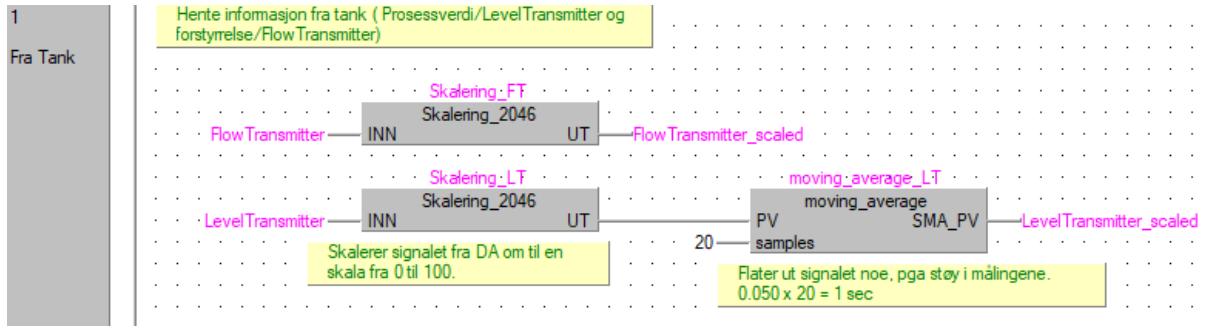
#### 4.1.1 Slave

Slave PLS-en koblet til tank-riggen gjennom et analogkort og til master PLS gjennom Profibus. Figur 11 er en oversikt over de globale variablene i slaven som sendes og mottas.

	Class	Label Name	Data Type	Constan	Device	Address	Comment
1	VAR_GLOBAL	pumpe	Bit	...	Y000	%QX0	Fysisk utganger på PLS
2	VAR_GLOBAL	ventil_1	Bit	...	Y004	%QX4	3 stk utventiler
3	VAR_GLOBAL	ventil_2	Bit	...	Y005	%QX5	
4	VAR_GLOBAL	ventil_3	Bit	...	Y006	%QX6	
5	VAR_GLOBAL	alarm_lampe	Bit	...	Y001	%QX1	
6			...				
7	VAR_GLOBAL	LevelTransmitter	Word[Signed]	...	D8260	%MW0.8260	LT - Range fra 400 - 2046
8	VAR_GLOBAL	FlowTransmitter	Word[Signed]	...	D8261	%MW0.8261	range fra 400 - 2046 (4-20mA)
9	VAR_GLOBAL	LV_Padrag	Word[Signed]	...	D8262	%MW0.8262	Pådrag sendt til D/A, 0 - 4096
10			...				
11			...				
12	VAR_GLOBAL	ventilVar1	Bit	...	M500	%MX0.500	variabel fra HMI om ventier skal være på.
13	VAR_GLOBAL	ventilVar2	Bit	...	M501	%MX0.501	
14	VAR_GLOBAL	ventilVar3	Bit	...	M502	%MX0.502	
15	VAR_GLOBAL	pumpeVar	Bit	...	M503	%MX0.503	fra HMI, om pumpe skal være på.
16	VAR_GLOBAL	sw_reg_param	Bit	...	M504	%MX0.504	
17	VAR_GLOBAL	auto_manuell	Bit	...	M505	%MX0.505	Modus til regulator desse 3 her.
18	VAR_GLOBAL	FF_on	Bit	...	M506	%MX0.506	
19	VAR_GLOBAL	lead_lag_on	Bit	...	M507	%MX0.507	
20			...				
21			...				
22	VAR_GLOBAL	alarm_case_1	Bit	...	M611	%MX0.611	Slå av pumpe, pådrag og åpne ventiler, alarmlys
23	VAR_GLOBAL	alarm_case_2	Bit	...	M612	%MX0.612	Slå av pumpe og pådrag, steng ventiler og
24	VAR_GLOBAL	alarm_case_3	Bit	...	M613	%MX0.613	Slå av FF
25	VAR_GLOBAL	alarm_lampe_ack	Bit	...	M614	%MX0.614	Stopper blinking av lampe
26			...				
27			...				
28	VAR_GLOBAL	Manuelt_Padrag	Word[Signed]	...	D220	%MW0.220	
29	VAR_GLOBAL	Settpunkt	Word[Signed]	...	D221	%MW0.221	Settpunkt, fra profibus
30	VAR_GLOBAL	k_p	FLOAT (Single Precision)	...	D450	%MD0.450	
31	VAR_GLOBAL	t_i	FLOAT (Single Precision)	...	D455	%MD0.455	
32	VAR_GLOBAL	t_d	FLOAT (Single Precision)	...	D460	%MD0.460	
33	VAR_GLOBAL	n_filter2	FLOAT (Single Precision)	...	D465	%MD0.465	
34	VAR_GLOBAL	KLL	FLOAT (Single Precision)	...	D470	%MD0.470	
35	VAR_GLOBAL	Tt	FLOAT (Single Precision)	...	D475	%MD0.475	
36	VAR_GLOBAL	Tn	FLOAT (Single Precision)	...	D480	%MD0.480	
37	VAR_GLOBAL	nominelt_padrag	Word[Signed]	...	D485	%MW0.485	
38			...				
39	VAR_GLOBAL	warning_sampletimeerror	Bit	...	M50	%MX0.50	Alarm detektert av slaaven om for lang tastetid.
40	VAR_GLOBAL	error_LT	Bit	...	M51	%MX0.51	
41	VAR_GLOBAL	error_FT	Bit	...	M52	%MX0.52	
42			...				
43			...				
44	VAR_GLOBAL	LevelTransmitter_scaled	Word[Signed]	...	D10	%MW0.10	Prosessverdi skalert fra 0 til 100
45	VAR_GLOBAL	Padrag	Word[Signed]	...	D11	%MW0.11	Pådrag utregnet, fra 0 - 100
46	VAR_GLOBAL	FlowTransmitter_Scaled	Word[Signed]	...	D12	%MW0.12	Forstymelse skalert fra 0 til 100
47	VAR_GLOBAL	error_bits_from_slave	Word[Signed]	...	D13	%MW0.13	
48	VAR_GLOBAL	watchdog_slave_hmi	Word[Signed]	...	D14	%MW0.14	
49			...				
50	VAR_GLOBAL	profi_connected	Bit	...	M707	%MX0.707	Slaven ikke kommunikasjon med master når lav

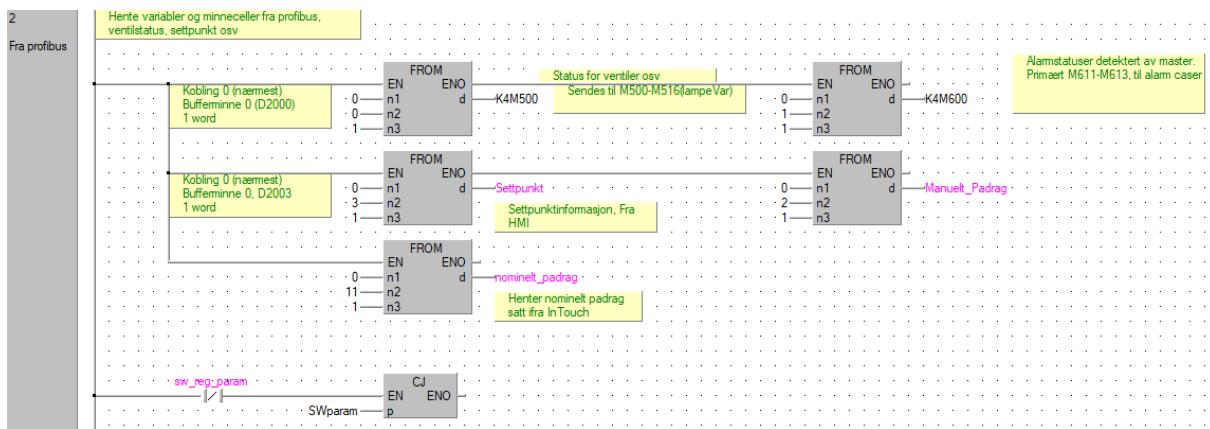
Figur 11: Globale variabler i slave PLS

I figur 12 hentes data inn fra AD-omformeren og skaleres med funksjonsblokker for å få en verdi mellom 0 til 100 %. Nivåmåleren (LT) går også gjennom en ekstra funksjonsblokk for å minske støy, mer om dette senere.



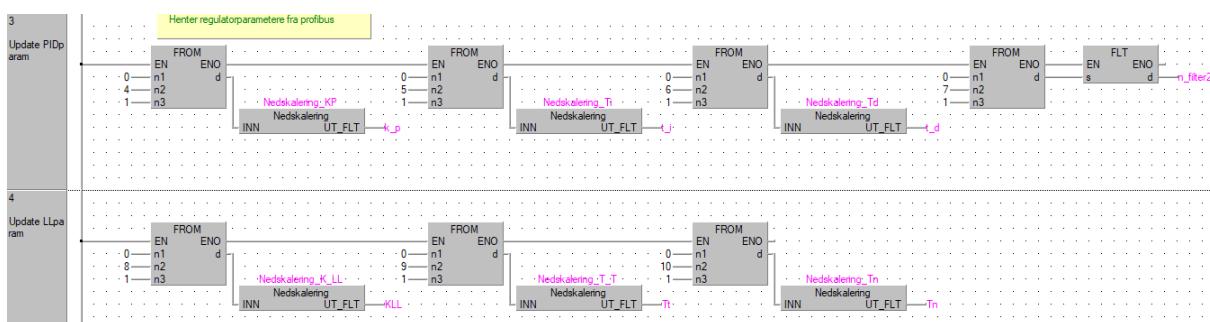
Figur 12: Innhenting av data tankrigg

I figur 13 overfører slaven verdier fra profibusminnet i master til minneceller og dataord internt i slaven.



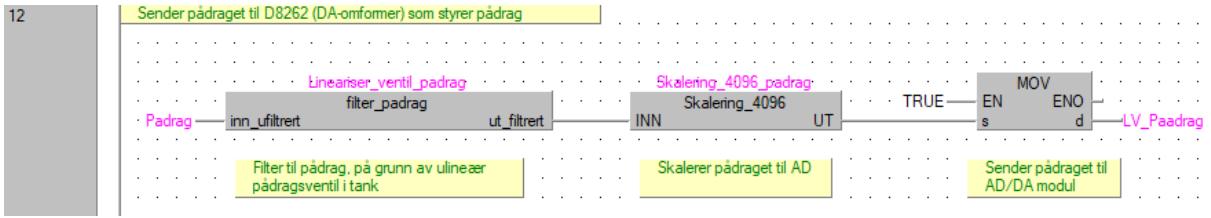
Figur 13: Fra Profibusminne til variabler i slave

I figur 14 skalerer slaven ned parametrene som er sendt fra InTouch. Masteren kan ikke håndtere flyttall så noen verdier må skaleres i slave og InTouch for å få «fullverdig» regulering. I figur 13 er det nederst et CJ – betinget hopp, dette gjør at nettverk 3 og 4 i figur 14 kan hoppes over, fordi operatøren ikke endrer verdiene hvert 50 ms.

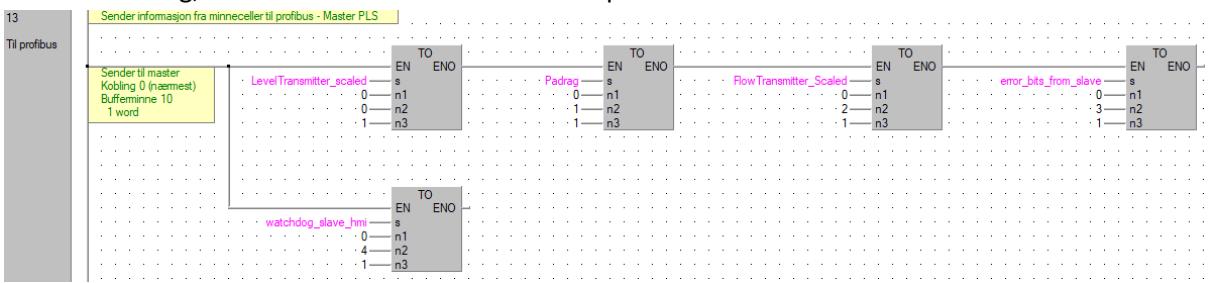


Figur 14: Skalering av parametarer

Det ferdig utregnede pådraget fra regulatoren blir sendt gjennom et ventilfilter og skalering før det går ut til ventilen på tank-riggen via DA omformeren. Dette ser vi i figur 15.



Figur 16 viser pådrag, sensoravlesninger, alarm for samplingstid og watchdog-teller, som skal til HMI for visualisering, sendes videre til master PLS over profibus.



#### 4.1.2 Master

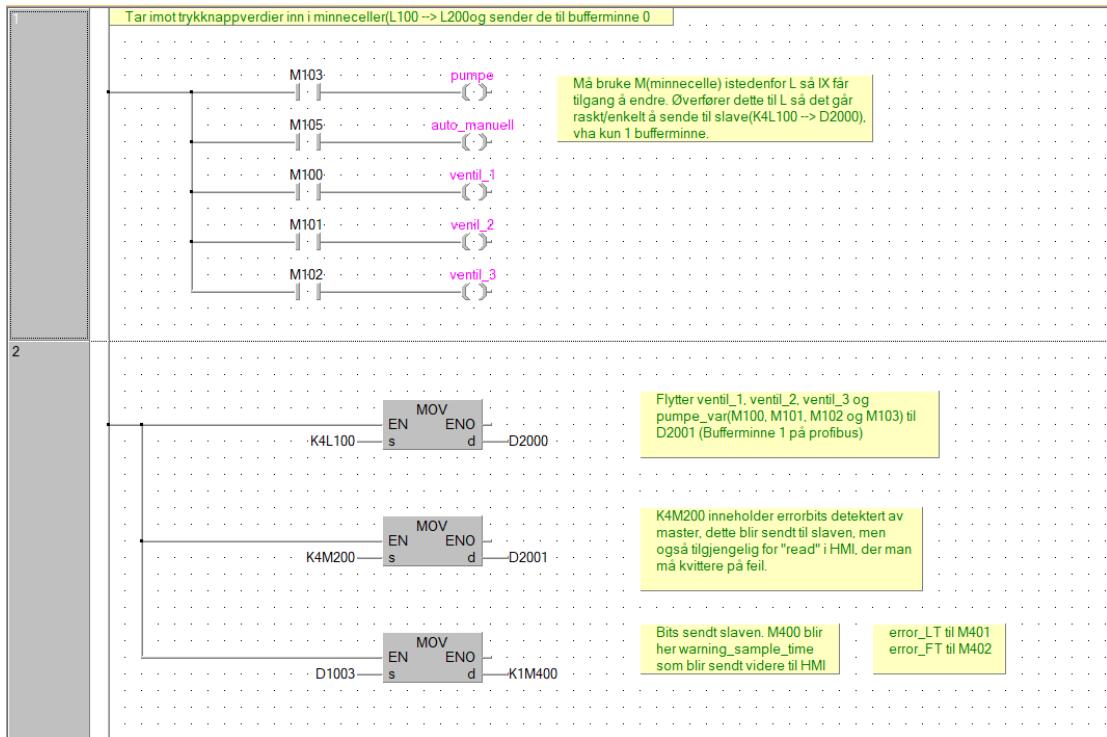
Masteren mottar parameter-verdier, settpunkt, ventilstyring og mer ifra brukergrensesnittene. Samtidig mottar den også informasjon fra slaven som nivået i tanken og utstrømning fra tanken. Alle disse verdiene lagres i variabler. Masteren sine globale variabler er vist i figur 17.

	Class	Label Name	Data Type	Constant	Device	Address
1	VAR_GLOBAL	prosessverdi	Word[Signed]	...	D0	%MW0.0
2	VAR_GLOBAL	padrag	Word[Signed]	...	D1	%MW0.1
3	VAR_GLOBAL	flow_transmitter	Word[Signed]	...	D2	%MW0.2
4	VAR_GLOBAL	first_run	Bit	...	L278	%MX8.278
5				...		
6				...		
7	VAR_GLOBAL	manuelt_padrag	Word[Signed]	...	D20	%MW0.20
8	VAR_GLOBAL	settpunkt	Word[Signed]	...	D21	%MW0.21
9	VAR_GLOBAL	kp	Word[Signed]	...	D22	%MW0.22
10	VAR_GLOBAL	t_i	Word[Signed]	...	D23	%MW0.23
11	VAR_GLOBAL	t_d	Word[Signed]	...	D24	%MW0.24
12	VAR_GLOBAL	n_tf	Word[Signed]	...	D25	%MW0.25
13	VAR_GLOBAL	k_ll	Word[Signed]	...	D26	%MW0.26
14	VAR_GLOBAL	t_t	Word[Signed]	...	D27	%MW0.27
15	VAR_GLOBAL	t_n	Word[Signed]	...	D28	%MW0.28
16	VAR_GLOBAL	nominelt_padrag	Word[Signed]	...	D29	%MW0.29
17				...		
18				...		
19	VAR_GLOBAL	ventil_1	Bit	...	L100	%MX8.100
20	VAR_GLOBAL	venil_2	Bit	...	L101	%MX8.101
21	VAR_GLOBAL	ventil_3	Bit	...	L102	%MX8.102
22	VAR_GLOBAL	pumpe	Bit	...	L103	%MX8.103
23	VAR_GLOBAL	sw_reg_param	Bit	...	L104	%MX8.104
24	VAR_GLOBAL	auto_manuell	Bit	...	L105	%MX8.105
25	VAR_GLOBAL	lead_lag_on	Bit	...	L106	%MX8.106
26	VAR_GLOBAL	lead_lag_static	Bit	...	L107	%MX8.107
27	VAR_GLOBAL	static_and_LL_on	Bit	...	L108	%MX8.108
28				...		
29	VAR_GLOBAL	error_FT_ACK	Bit	...	M300	%MX0.300
30	VAR_GLOBAL	error_LT_ACK	Bit	...	M301	%MX0.301
31	VAR_GLOBAL	error_HH_ACK	Bit	...	M302	%MX0.302
32	VAR_GLOBAL	error_H_ACK	Bit	...	M303	%MX0.303
33	VAR_GLOBAL	error_L_ACK	Bit	...	M304	%MX0.304
34	VAR_GLOBAL	error_LL_ACK	Bit	...	M305	%MX0.305
35	VAR_GLOBAL	error_E_ACK	Bit	...	M306	%MX0.306
36	VAR_GLOBAL	error_PROFACK	Bit	...	M307	%MX0.307
37	VAR_GLOBAL	error_KEP_ACK	Bit	...	M308	%MX0.308
38	VAR_GLOBAL	error_ETH_ACK	Bit	...	M309	%MX0.309
39	VAR_GLOBAL	error_POWER_ACK	Bit	...	M310	%MX0.310
40	VAR_GLOBAL	warning_samplingtime_ACK	Bit	...	M311	%MX0.311
41				...		
42				...		
43	VAR_GLOBAL	error_SAMPLING	Bit	...	M400	%MX0.400
44				...		
45	VAR_GLOBAL	error_FT	Bit	...	M200	%MX0.200
46	VAR_GLOBAL	error_LT	Bit	...	M201	%MX0.201
47	VAR_GLOBAL	error_PROF	Bit	...	M202	%MX0.202
48				...		
49				...		
50	VAR_GLOBAL	error_E	Bit	...	M209	%MX0.209
51	VAR_GLOBAL	error_POWER	Bit	...	M210	%MX0.210
52				...		
53	VAR_GLOBAL	alarm_HH	Bit	...	M205	%MX0.205
54	VAR_GLOBAL	alarm_H	Bit	...	M206	%MX0.206
55	VAR_GLOBAL	alarm_L	Bit	...	M207	%MX0.207
56	VAR_GLOBAL	alarm_LL	Bit	...	M208	%MX0.208
57				...		
58	VAR_GLOBAL	alarm_case_1	Bit	...	M211	%MX0.211
59	VAR_GLOBAL	alarm_case_2	Bit	...	M212	%MX0.212
60	VAR_GLOBAL	alarm_case_3	Bit	...	M213	%MX0.213
61	VAR_GLOBAL	alarm_lampe_ack	Bit	...	M214	%MX0.214
62	VAR_GLOBAL	connection_error_HMI	Bit	...	M215	%MX0.215
63				...		
64	VAR_GLOBAL	limit_HH	Word[Signed]	...	D100	%MW0.100
65	VAR_GLOBAL	limit_H	Word[Signed]	...	D101	%MW0.101
66	VAR_GLOBAL	limit_L	Word[Signed]	...	D102	%MW0.102
67	VAR_GLOBAL	limit_LL	Word[Signed]	...	D103	%MW0.103
68	VAR_GLOBAL	avvik_stasjonært_max	Word[Signed]	...	D105	%MW0.105
69	VAR_GLOBAL	tid_for_avvik	Word[Signed]	...	D106	%MW0.106
70				...		
71	VAR_GLOBAL	watchdog_master_hmi	Word[Unsigned]/Bit String[16-bit]	...	D107	%MW0.107
72	VAR_GLOBAL	watchdog_slave_hmi	Word[Signed]	...	D3	%MW0.3

Figur 17: Globale variabler i master PLS

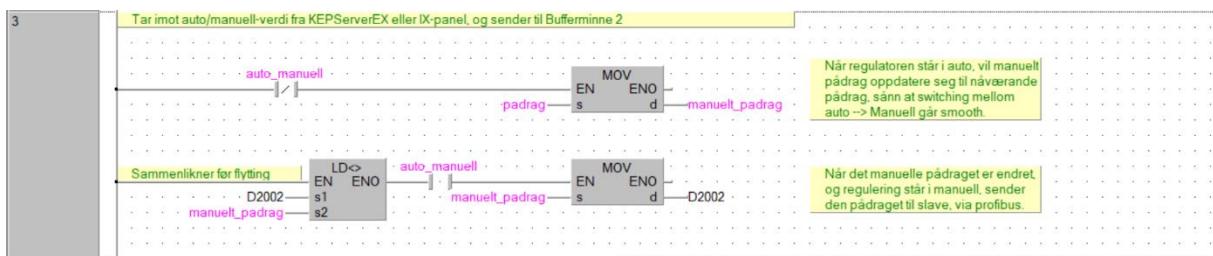
Bit-ene som masteren tar imot lagres direkte i PLS-en. Masteren har egne minneceller som begynner med «L» istedenfor «M», og disse er ikke implementert i IX Developer. De bit-ene som mottas fra IX-

panelet blir dermed behandlet før videresending til slaven. Denne behandlingen av minnecellene er vist i figur 18.

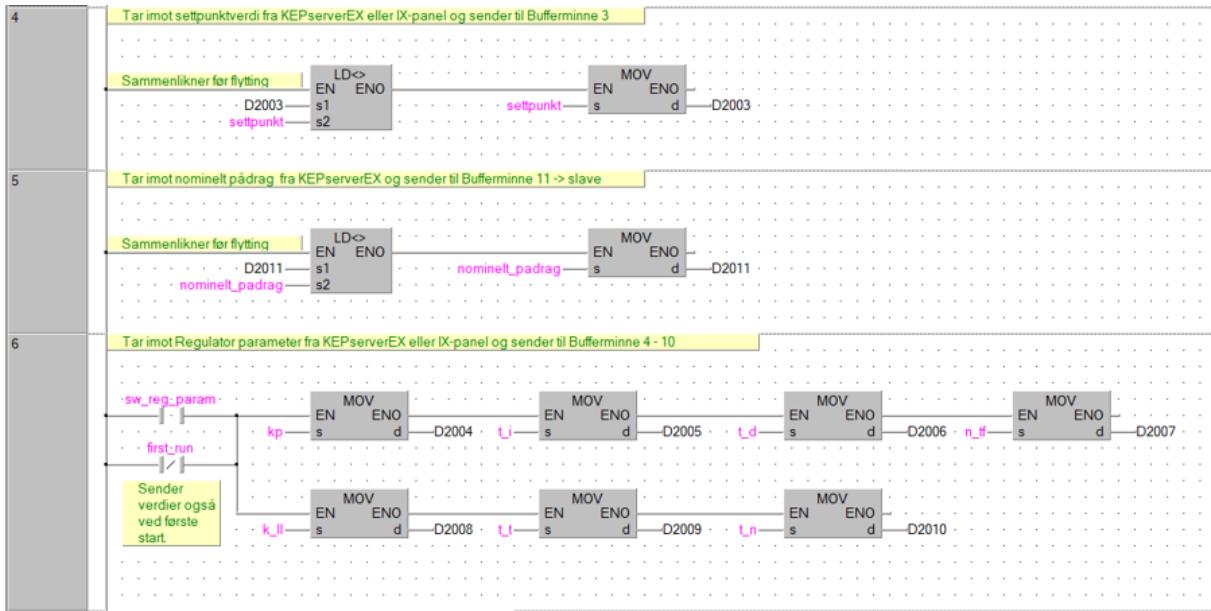


Figur 18: Oversikt over bits som blir hentet fra KEPServerEx eller IX-panelet til masteren, og sendt til bufferminner på Profibus (til slaven).

Fra KEPServer og IX-panelet sendes det verdi for settpunkt, manuelt pådrag, nominelt pådrag og parameterverdier. Masteren sender disse verdiene til slaven som vist i figur 19 og 20.

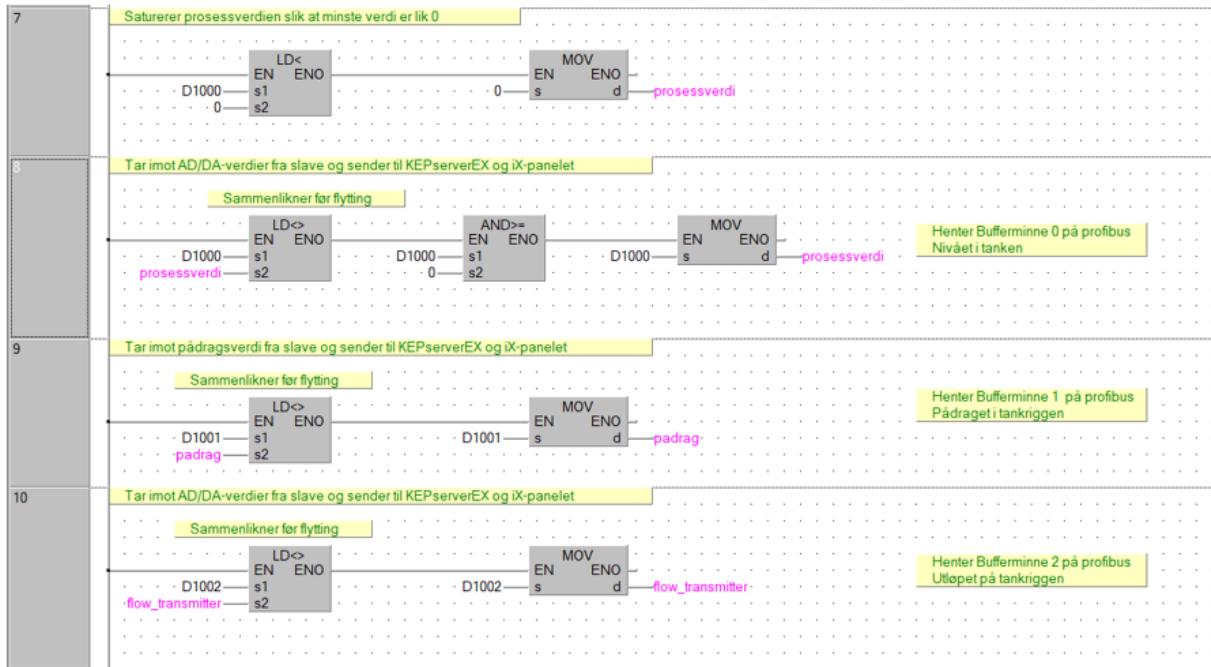


Figur 19: Oversikt over det manuelle pådraget som blir hentet fra KEPServer eller IX-panelet til masteren, og sendt til bufferminner på Profibus (til slaven).



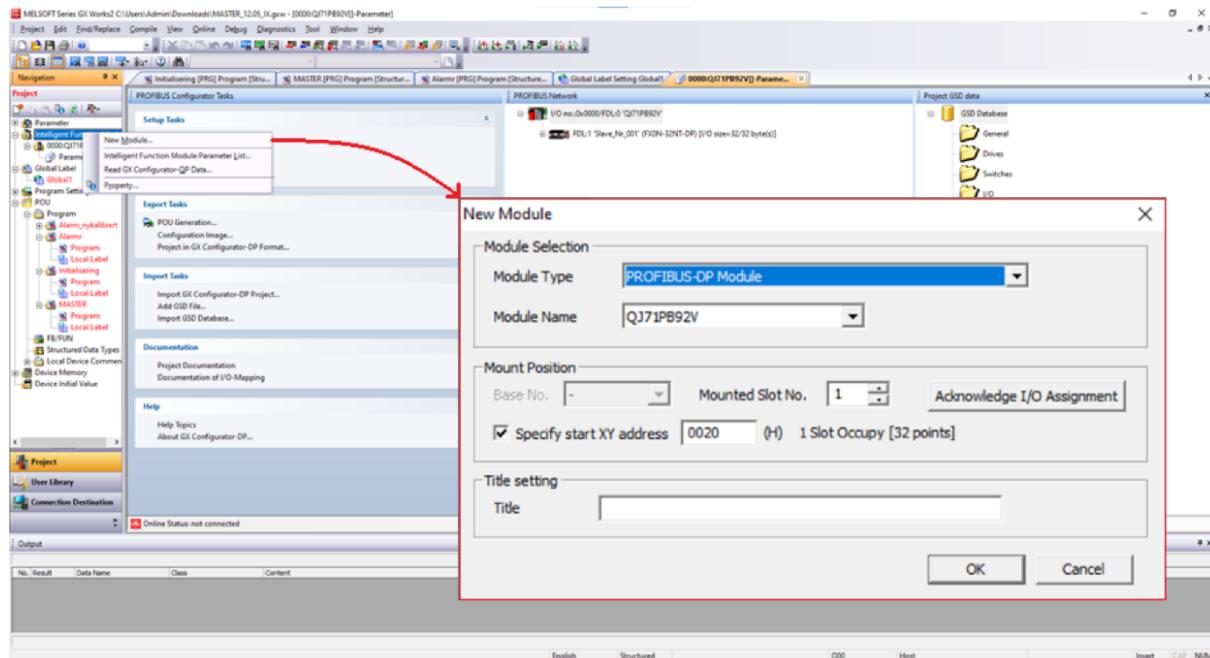
Figur 20: Oversikt over dataord som blir hentet fra KEPServer eller IX-panelet til masteren, og sendt til bufferminner på Profibus (til slaven).

Slave PLS-en sender verdier til masteren, i figur 21 vises det hvordan masteren henter verdiene fra Profibus.



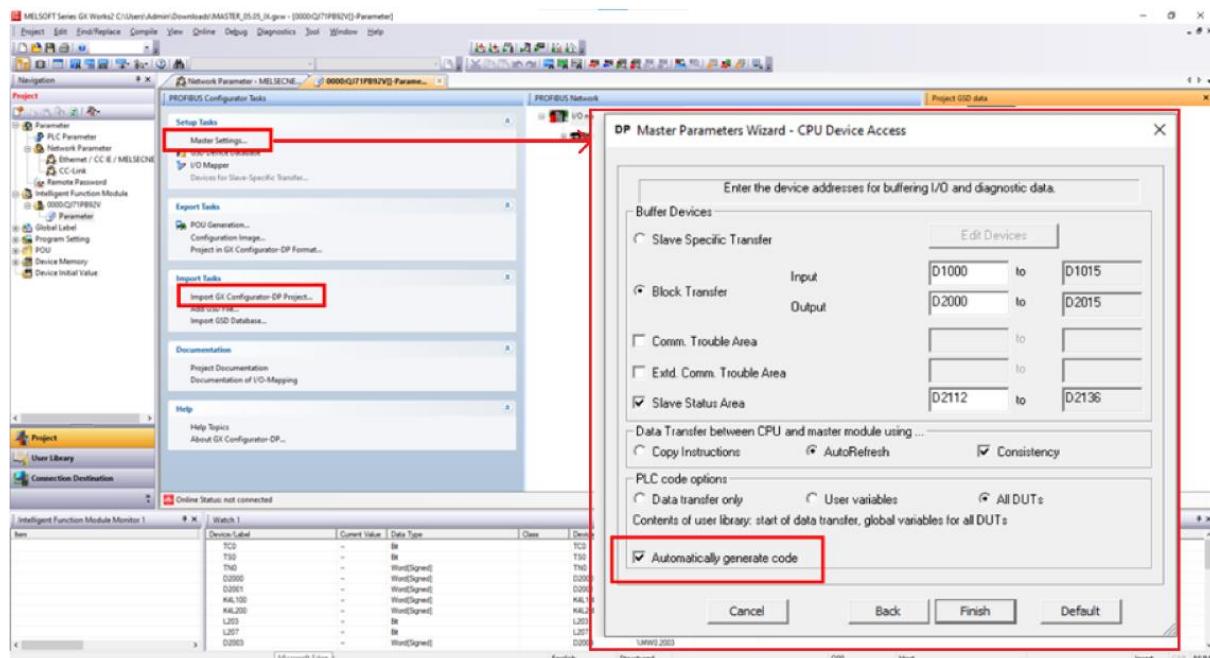
Figur 21: Oversikt over dataord som blir hentet fra bufferminner på Profibus til masteren, og sendt til KEPServer eller IX-panelet.

For at Profibus skal starte kommunikasjonen mellom master og slave automatisk ved oppstart av PLS-ene må det importeres en GX Configurator fil inn i GX Works 2 programmet. Dette gjøres ved å sette opp en «Intelligent Funtion Module» som vist i figur 22.



Figur 22: Konfigurering av profibus på GX Works

I Profibus Configurator Task vinduet kan man importere GX Configurator-filen under «Import Tasks» og «Import GX Configurator-DP Project». Deretter kan man under «Setup Tasks» gå inn på «Master Settings» og huke av for å generere automatisk kode. Dette er vist i figur 23.



Figur 23: «Profibus Configurator Tasks» – vindu i GX Works 2. Til høyre vises «Master Settings» - vindu.

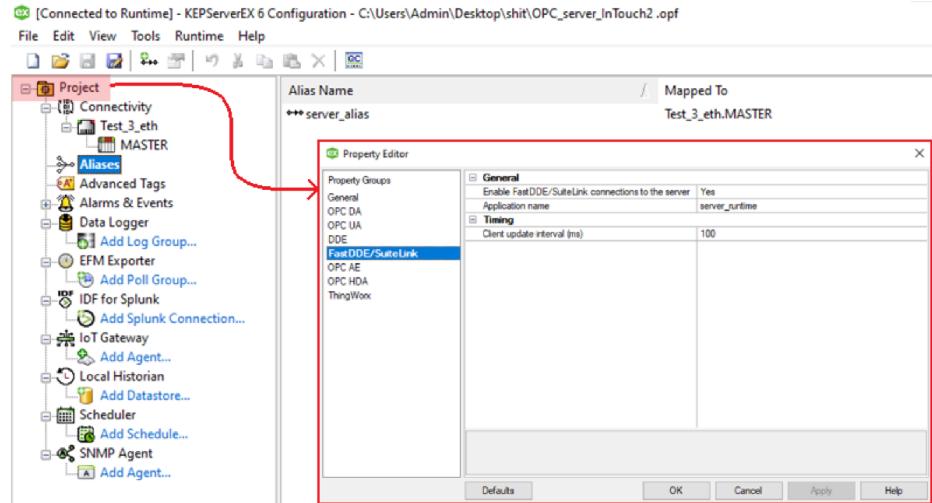
Den genererte koden som kommer etter å ha huket av utfører en rekke aksjoner for å sjekke om det er klart for å opprette kommunikasjon med slaven, og starter deretter opp Profibus-kommunikasjonen automatisk. Koden er vist i figur 24.

```
(* start data transfer *)  
  
LD X1B      (* check communication READY signal *)  
AND X1D      (* check Module READY *)  
OUT_M Y00 (* Exchange start request signal *)  
LD X1B      (* check communication READY signal *)  
AND X1D      (* check Module READY *)  
AND X00      (* Data exchange done *)  
(* if data transfer is not started, jump to end of POU, bypassing the access to slave inputs*)  
JMPCN EndOfPOU  
EndOfPOU:  
(* empty network to provide a label at the end of the POU *)
```

Figur 24: Kodesnuttene over viser den automatisk genererte koden så kommer når man huker av i «Master Parameters Wizard» - vinduet.

#### 4.1.3 KEPServer

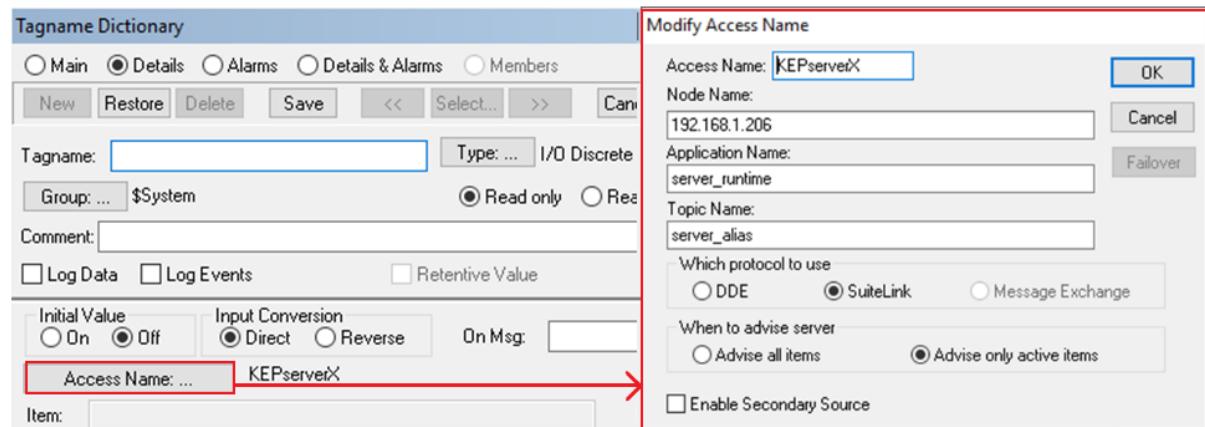
I KEPServer skal det settes opp et alias og applikasjonsnavn som benyttes til kommunikasjon mellom KEPServer og InTouch. Dette gjøres ved å trykke på «Aliases» under «Project». Deretter må det legges til et «Application name» ved å klikke på «Project» og fylle inn som vist på figur 25. For full oversikt over tags for dataoverføring i KEPServer se tabell 1 i vedlegg: Tagname-liste.



Figur 25: «Application name» til KEPServer-filen er «server\_runtime».

#### 4.1.4 Intouch

InTouch programmet må inneholde flere ulike tags. For kommunikasjon er det nødvendig med I/O tags (Input/Output). Grunnen til dette er at disse kan sende og motta verdier til og fra KEPServer, enten lokalt eller eksternt. Dette kan gjøres ved å sette opp et «Access Name» i Tagname Dictionary for KEPServer, vist i figur 26. Full oversikt over tags for dataoverføring og interne tags i InTouch, se tabell 2 og 3, vedlegg: Tagname-liste.



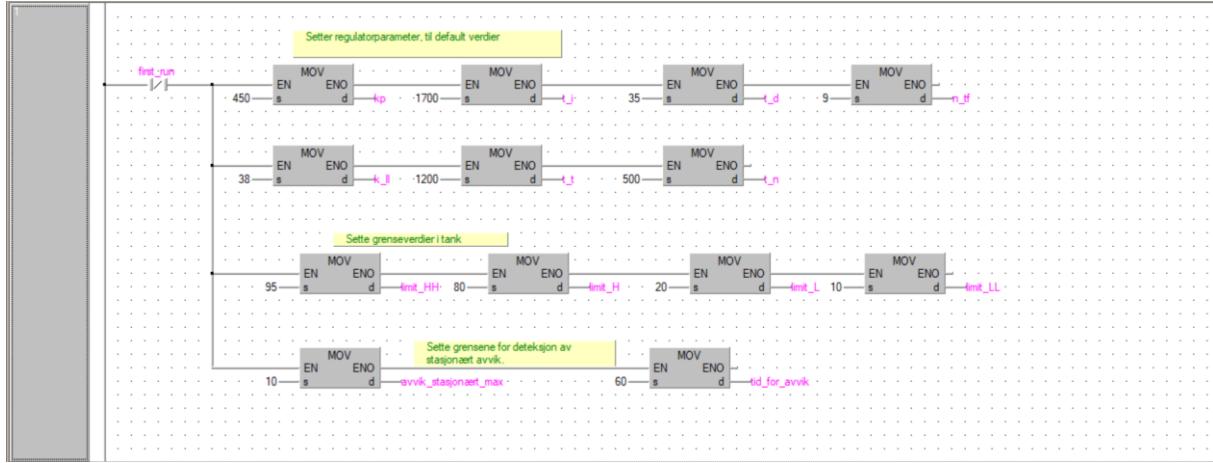
Figur 26: "Tagname Dictionary" - vinduet, med detaljer for I/O - tags. Nederst til venstre kan man se "Access Name".

#### 4.1.5 IX Developer

Brukergrensesnittet på PLS-riggen er koblet opp til masterens variabler. Oversikt over tags i IX Developer finnes i vedlegg for Tagname-liste, tabell 4.

## 4.2 Initieringsprogram

Når PLS-en kjøres for første gang, settes regulatorparametere og grenseverdier til standardverdier. Det er ikke nødvendig å kjøre initieringen etter strømbrudd, da parameterne er lagret i batterimantede dataregistre.



Figur 27: Initieringsprogrammet i masteren som kun kjøres i første scan.

## 4.3 Lesing av transmittere

### 4.3.1 Skalering

Prosessmålingene som sendes til slavens AD/DA, er ønskelig å skalere til en prosentskala for videre bruk i regulator og HMI. I koden under tar man inn verdien lest av fra AD i slave-PLS og skalerer dette til verdier fra 0 til 100%. Dette gjøres når man vet hva som tilsvarer 4mA strøm (0%), og 20mA strøm (100%) i digitale verdier. Blir verdien ut fra funksjonsblokken under 0, tyder dette på en feil i transmitteren, og en alarm vil da bli kalt.

```
(*          Konvertering til flyttall for videre utrekning      *)
FLT(1,INN,INN_FLT);

(*          Skaleringsformel          *)
UT_FLT := (INN_FLT - 400.0)/(1648.0)*100.0;

(*          Konvertering til heltall for utgang i skaleringsblokk *)
INT(1,UT_FLT,UT);
```

### 4.3.2 SMA-filter (Simple Moving Average)

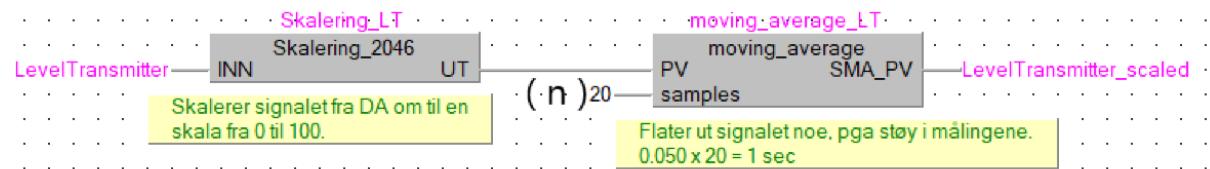
Siden man måler nivået i tanken med en trykktransmitter, og innløpet er i toppen av tanken, blir det en del støy i nivåmålingen. D-leddet i en regulator vil kunne forsterke dette støyet fra inngangen.

Mange transmittere har SMA-filter internt. I dette tilfellet er det ikke ønskelig å justere på transmitterne på anlegget. Det ble derfor laget et eget SMA-filter inne i slaven. Utgangen av filteret blir oppdatert for hver programsyklus. Filteret uttrykkes slik:

$$SMA_{PV} = \frac{PV[k] + PV[k - 1] + \dots + PV[k - n]}{n}, k \in [0, n]$$

Koden under og figur 28 viser hvordan SMA-filteret er implementert i slaven.

```
(* Simple Moving Average Filter *)  
  
INC(1,index);  
IF index >= samples THEN  
    index := 0;  
END_IF;  
  
temp_SMA_PV := temp_SMA_PV + PV - readings[index]; (*Erstatt siste måling med ny*)  
readings[index] := PV;  
SMA_PV := temp_SMA_PV/samples; (*Tar gjennomsnittet*)
```



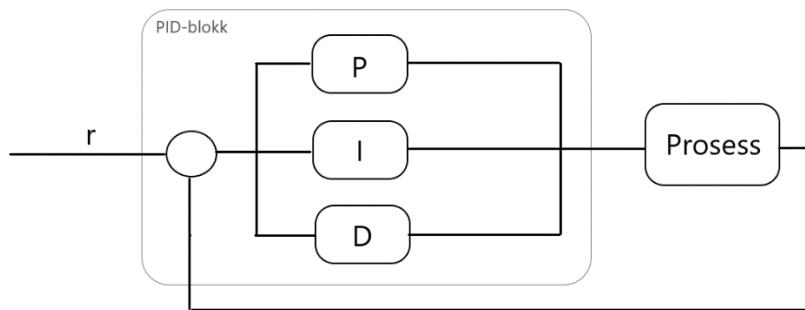
Figur 28: SMA, på bildet måles snittet av 20 målinger

#### 4.4 PID - Implementering

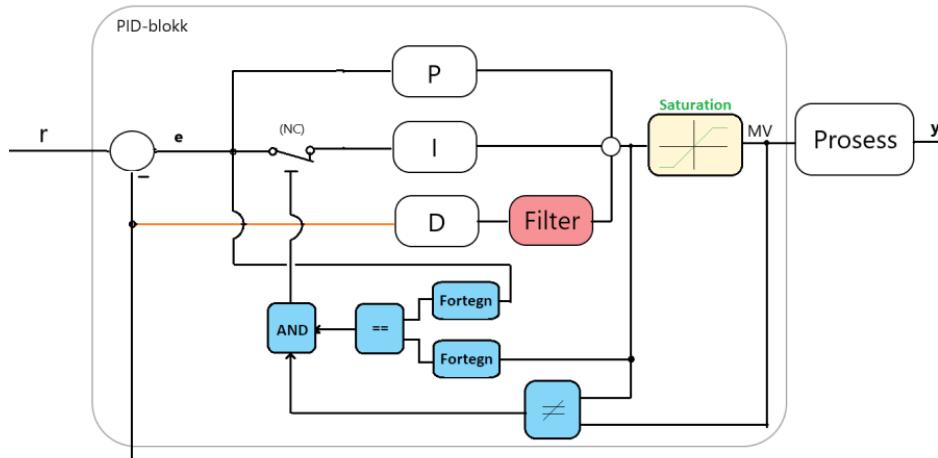
For at regulatoren skal fungere som ønsket er det en del praktiske hensyn man må ta, blant annet:

- Anti-windup
- Derivasjonskick
- Derivasjonsfilter
- Saturation control
- Valg av samplingtid

Mer informasjon om dette finnes i «Arbeidsnotat PID og Lead-Lag funksjonsblokk» (Halse, et al., 2021). Figur 29 viser eksempel på blokediagram for en generell PID regulator og figur 30 viser en PID-regulator med praktiske hensyn: Anti-windup (blå blokker) derivasjonskick (oransje strek), derivajonsfilter(rød blokk), saturation control (gul blokk).



Figur 29: Blokediagram for standard PID regulator.



Figur 30: Blokediagram for PID-blokk med praktiske hensyn

Under ser man en pseudokode av PID-regulatoren. For full kode se vedlegg: PID-kode.

(\* PID Pseudokode \*)

```
IF ENABLE:  
    Nullstill verdier // Starter på nytt  
ELSE:
```

```

//Kjør PID som normal

Saturer Innganger
Kalkuler Avvik
Kalkuler D_gain
Kalkuler P_Gain

// Sjekk for tracking
IF (Utgang == Tracking_inngang): // Denne PIDen er i kontroll
    kalkuler I_gain // Som normalt, se ligning på forrige side
ELSE // Denne PIDen er IKKE i kontroll. Behov for tracking
    Kalkuler I_gain //I-gain = TR - P-Gain - D-gain

kalkuler Utgang
Saturer Utgang

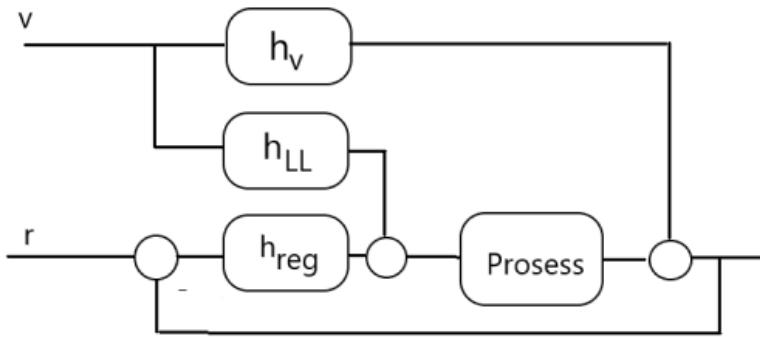
IF Saturerer:
    Antiwindup

Oppdater variabler

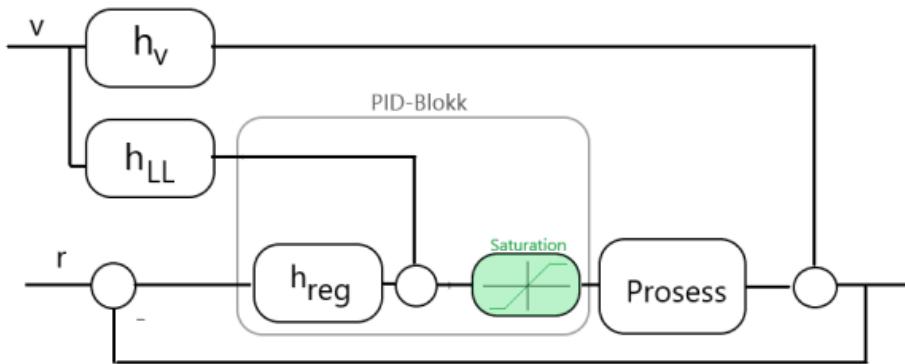
```

## 4.5 Lead-Lag – Implementering

Figur 31 viser en generell Lead-Lag – blokk, og den er relativ enkel i forhold til PID-regulatoren. Den har dermed mindre elementer å ta hensyn til. For å få til rykkfri overgang med PID og Lead-Lag summeres pådraget fra Lead-Lag inne i PID-regulatoren, samtidig løses også problemer med windup. Dette blir fremvist i figur 32.



Figur 31: Blokdiagram av en reguleringssløyfe med et Lead-Lag element, benyttet fra forstyrrelse.



Figur 32: Blokdiagram av en reguleringssløyfe med et Lead-Lag element, benyttet fra forstyrrelse. Her blir Lead-Lag – pådraget sendt inn i regulatoren av praktiske hensyn til integrasjonsdelen i regulatoren.

Koden er fullstendig og under viser hvordan Lead-Lag funksjonsblokken er implementert i slaven i GX Works.

```

(* Lead Lag-Programkode *)

Disturbance_local := Disturbance;

IF ENABLE = 0 THEN
    LL_Output := 0.0;

(* Modus: Statisk *)
ELSIF static THEN
    LL_Output := Disturbance_local*K_LL;

(* Modus: LeadLag *)
ELSE
    diff1 := Disturbance_prev*K_LL*(SampleTime - 2.0*T_t)/(SampleTime+2.0*T_n);
    diff2 := Disturbance_local*K_LL*(SampleTime+2.0*T_t)/(SampleTime+2.0*T_n);
    diff3 := U_prev*(SampleTime-2.0*T_n)/(SampleTime+2.0*T_n);
    LL_Output := diff1 + diff2 - diff3;
END_IF;

(* Saturerer utgangesverdiene *)
IF LL_Output > MAX_OUTPUT THEN
    LL_Output := MAX_OUTPUT;
ELSIF LL_Output < MIN_OUTPUT THEN
    LL_Output := MIN_OUTPUT;
END_IF;

INT(1, LL_Output,Output);          (* Konverterer fra FLOAT til Word, da utgangen av leadlagn er WORD. *)

U_prev := LL_Output;
Disturbance_Prev := Disturbance_local;

```

## 4.6 PID og Lead-Lag i GX Works

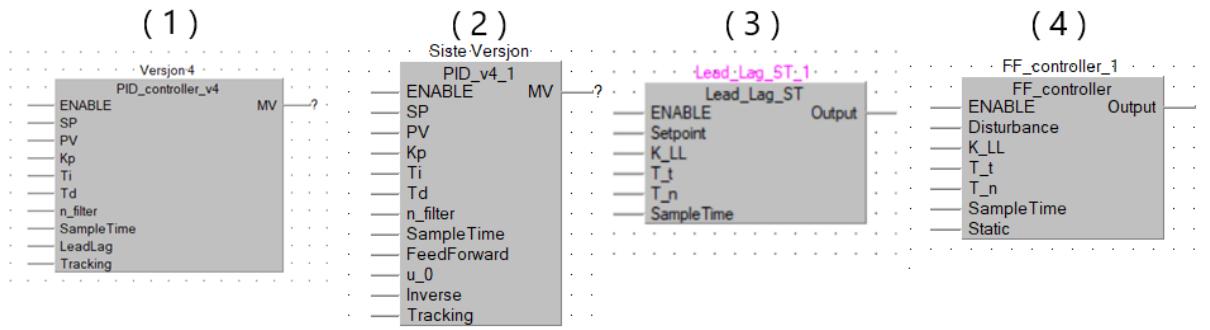
Fullstendig oversikt over hvordan PID og Lead-Lag blokkene er implementert finnes i "Arbeidsnotat PID og Lead-Lag funksjonsblokk" (Halse, et al., 2021). Programkode finnes i vedlegg 3. Endringer og videre implementering av PID og Lead-Lag beskrives her.

Funksjonsblokker merket (1) og (3) i figur 33, er hentet fra arbeidsnotat. Funksjonsblokk (2) og (4) er videreutviklede funksjonsblokker.

Den nyeste versjonen av PID regulatoren har to ekstra innganger: «u\_0» som tar inn nominelt pådrag, og «Inverse» som gjør det mulig å endre retningen på regulatoren. Inngang «LeadLag» har endret navn til «FeedForward». Dette da foroverkoblingen ikke nødvendigvis må være av type Lead-Lag.

Funksjonsblokken «Lead\_Lag\_ST» har endret navn til «FF\_controller», fordi den nå har mulighet for statisk foroverkobling. Hvis inngangen «Static» er høy, kjører filteret som statisk foroverkobling. Hvis ikke kjører den som Lead-Lag.

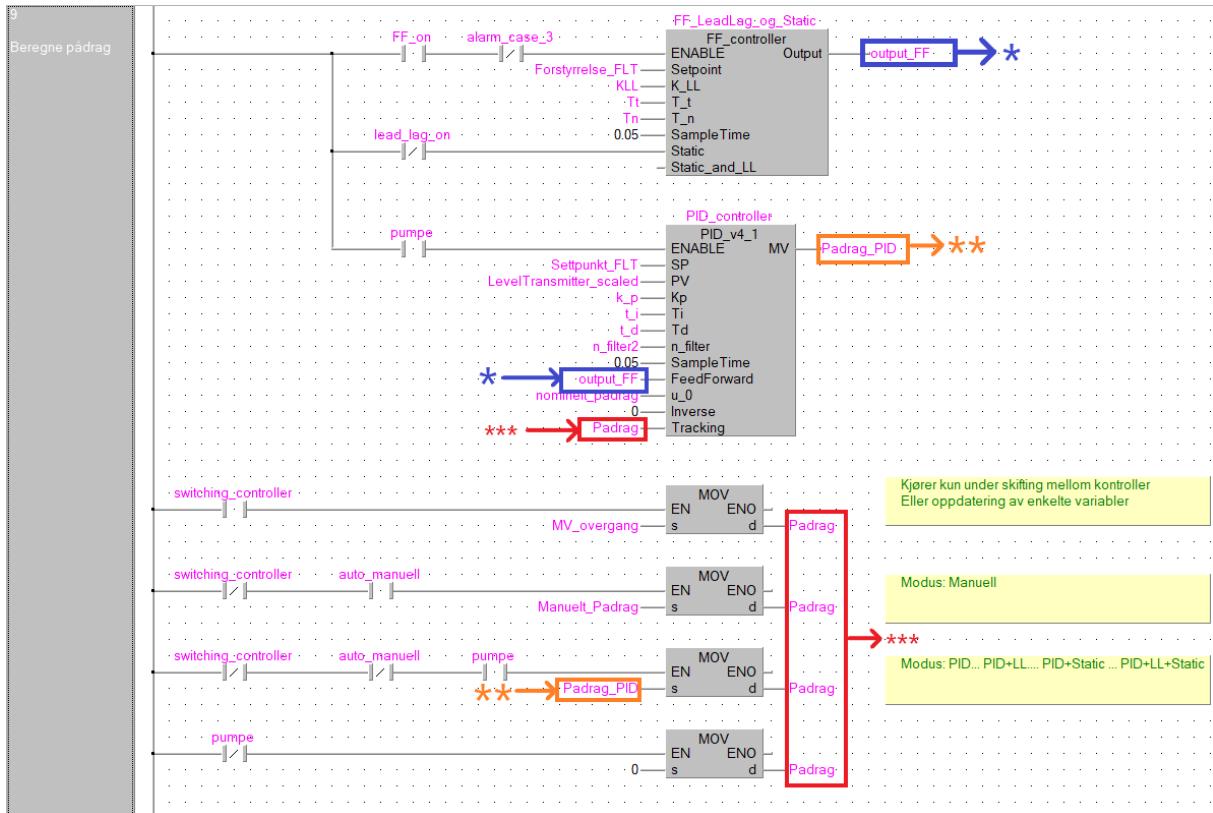
$$Statisk_{gain} = Forstyrrelse \cdot K_{LL}$$



Figur 33: PID og FeedForward Funksjonsblokker. (1) Gammel PID. (2) Ny PID. (3) Gammel FF. (4) Ny FF.

Figur 34 illustrerer hvordan utgangen til foroverkoblingsfilteret går inn på PID-regulatoren. Dette gjør det mulig for rykkfri overgang ved hjelp av inngang «Tracking». Det gjør også at man slipper en ekstra «Saturation»-blokk på utsiden av PID-regulatoren.

Hvis man regulerer ved hjelp av en PI- eller PID-regulator, og man bruker lav  $K_p$  og høy  $T_i$ , kan det oppstå uønskede situasjoner. Selv om I-leddet i utgangspunktet har antiwindup kan man i verste fall komme i en situasjon der I-leddet summerer seg opp helt til utgangen av regulatoren blir 100%. Slike situasjoner kan forekomme hvis pumpe er slått av, eller hvis en alarmaksjon slår av pumpe. For å unngå dette, vil pådraget settes til 0 hvis pumpe er av. Når pumpe er av vil inngang «ENABLE» på PID-blokkene gå lav, og alle interne variabler vil settes til 0. PID-regulatoren vil da starte med blanke ark når pumpe siås på igjen.



Figur 34: Implementering av FF\_controller og PID i GX Works

## 4.7 Rykkfri overgang

### 4.7.1 Tracking

Ved rykkfri overgang mellom PID-regulatorer unngår man et brått sprang i pådraget. Hvordan regulatoren kalkulerer I-leddet hvis pådraget ut ikke er lik inngang for «Tracking» kan uttrykkes ved:

$$I_{gain} = TR - P_{gain} - D_{gain} - FF$$

Merk at regulatoren også har FF (Feed-Forward) i ligningen. Pådraget fra foroverkoblingen går inn i PID-blokken. På denne måten vil rykkfri overgang bli mulig selv om foroverkoblingen slås inn eller ut. I tillegg kan signalet satureres felles ut fra PID-blokken, uten at en trenger en egen blokk for dette på utsiden.

Under ser man hvordan Tracking ble implementert i GX Works. Koden er et utdrag fra PID-funksjonsblokka i slaven.

```
IF MV = Tracking THEN (*Note that MV is from the last cycle and has not been updated YET*)
    Aa := Kp * SampleTime / (2.0 * Ti);
    I_gain := Aa * (Dev + Dev_prev) + I_gain_prev;
ELSE
    (*This controller is not in control. Activate Tracking *)
    tracking_flag := 1;
    FLT(1, Tracking - FeedForward, Aa);
    I_gain := Aa - P_gain - D_gain ;
ENDIF;
```

Det er teoretisk sett ingen begrensninger på hvor mange PID-regulatorer som kan kobles opp i en slik krets, men det er praktiske begrensninger. All programkode til hver regulator vil kjøre hver syklus. Dette fører til to ting:

1. Samplingstiden til programmet øker proporsjonalt med antall regulatorer.
2. Størrelsen på programmet øker proporsjonalt med antall regulatorer.

#### 4.7.2 Rykkfri overgang - implementering

Det er utviklet en alternativ metode for rykkfri overgang. Den bruker samme kode på innsiden av PID-regulatoren. Men, i stedet for å koble uvisst antall PID-regulatorer i parallelle benyttes bare én regulator og en «Switcher»-blokk.

Denne «Switcheren» fryser utgangen til regulatoren i fire programsykler – før den bytter tilbake. Dette gjør at PID-regulatoren kan oppdatere seg selv.

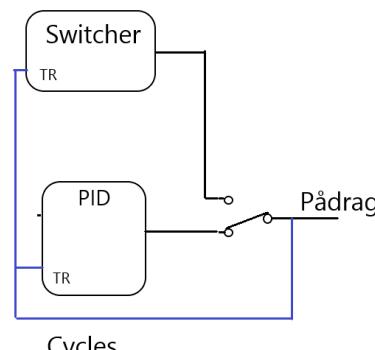
Denne metoden gjør det mulig med rykkfri overgang ved parameterskifte av PID-regulatoren. Dette inkluderer endring av  $K_p$ ,  $T_i$ , og så videre. Samtidig sparer den plass i programmet og minimerer samplingstiden til programmet.

Den eneste ulempen er at regulatoren ikke vil ha kontroll over prosessen under skifte. Dette vil ikke egne seg på prosesser som krever lynraske reaksjoner ved bytte eller for regulatorer med lang samplingtid.

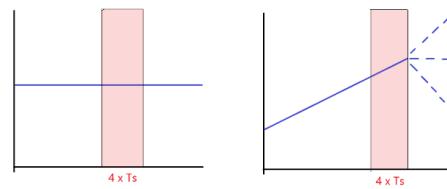
Tid under bytte kan kalkuleres som:

$$\tau_{bytte} = 4 \cdot T_s = 4 \cdot 50ms = 200ms$$

Ved regulering av nivå i en vanntank, med en samplingstid på 50ms, kan tiden som brukes under bytte anses som neglisjerbar.



- Cycles
1. Bryter opp. Bytt paramtere og type regulator
  2. Vent
  3. Vent
  4. Bryter ned.



#### 4.7.3 Rykkfri overgang ved parameterskifte

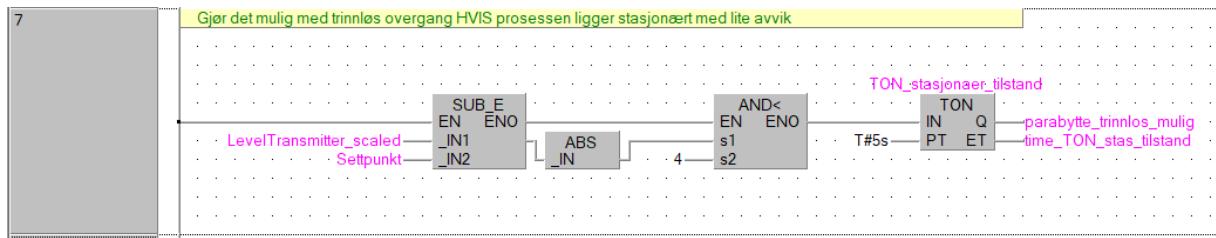
Rykkfri overgang ved parameterskifte kan i enkelte tilfeller føre til uønskede situasjoner. En slik situasjon kan oppstå hvis prosessen er på tur oppover i en rask hastighet og man minsker  $K_p$ , og samtidig øker  $T_i$  slik at integratoren blir mye tregere. Da vil integratoren stå nærmest i ro med et høyt pådrag. Dette kan føre til et høyt dynamisk avvik og i verste fall gjøre at høyhøy alarm slår av prosessen.

I tabell 5 er det illustrert hvordan en slik situasjon kan oppstå ved endring. Legg merke til hoppet i  $I_{gain}$  fra 20 til 50 samtidig som  $T_i$  økes slik at I-leddet blir tregere.

Tabell 1: Rykkfri overgang ved parameterskifte

Før/etter bytte	Parametere	$P_{gain}$	$I_{gain}$	Output
Før	$K_p = 4, T_i = 10$	60	20	80
Etter (Rykkfri)	$K_p = 2, T_i = 1000$	30	50	80
Etter (Ikke rykkfri)	$K_p = 2, T_i = 1000$	30	20	50

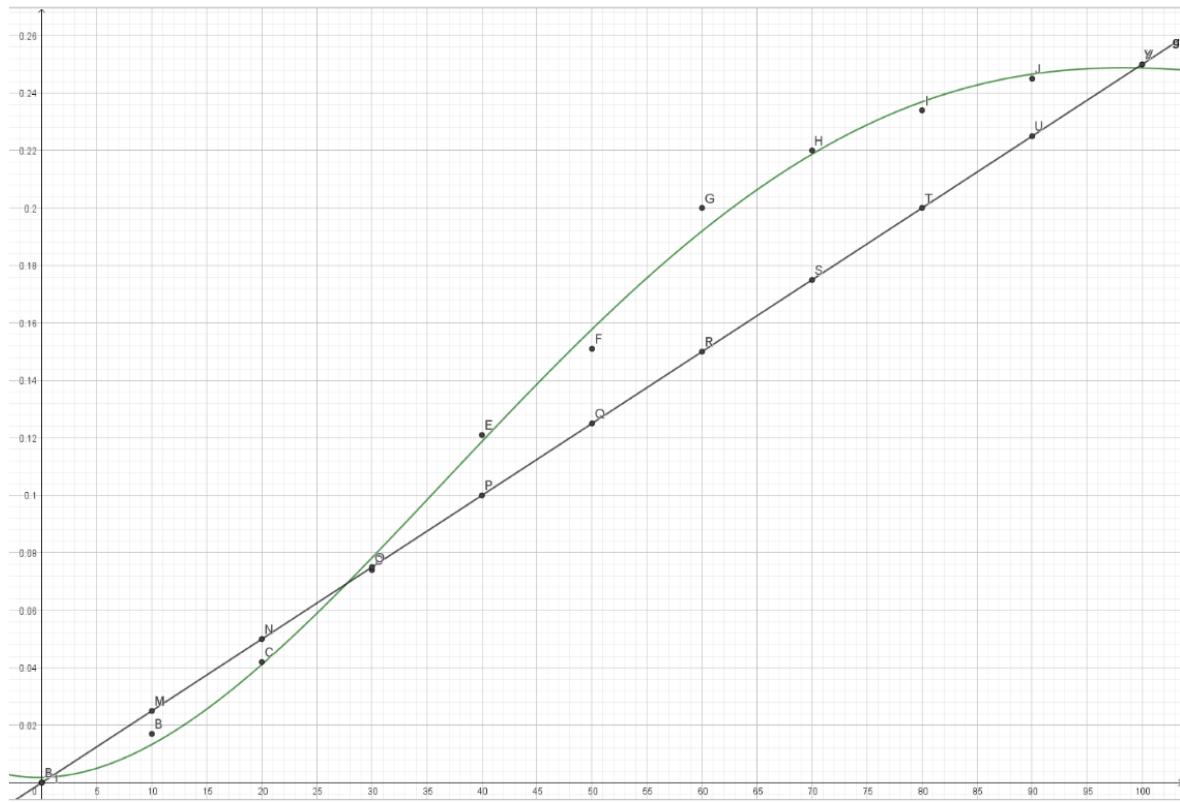
Følgende nettverk hentet fra programkoden til slaven sørger for at rykkfri overgang kun oppstår når prosesser har stått stabilt over lengre tid.



Figur 36: Sjekker om rykkfri overgang kan utføres. Slave POU: "Slave" Nettverk: 7

#### 4.8 Ulineært ventilpådrag - Ventilfilter

Det første steget i å lage et filter for pådraget, er å plotte de to settene med verdier fra figur 2 og 3 i teorikapittelet. Resultatet vil se ut som i figur 37.

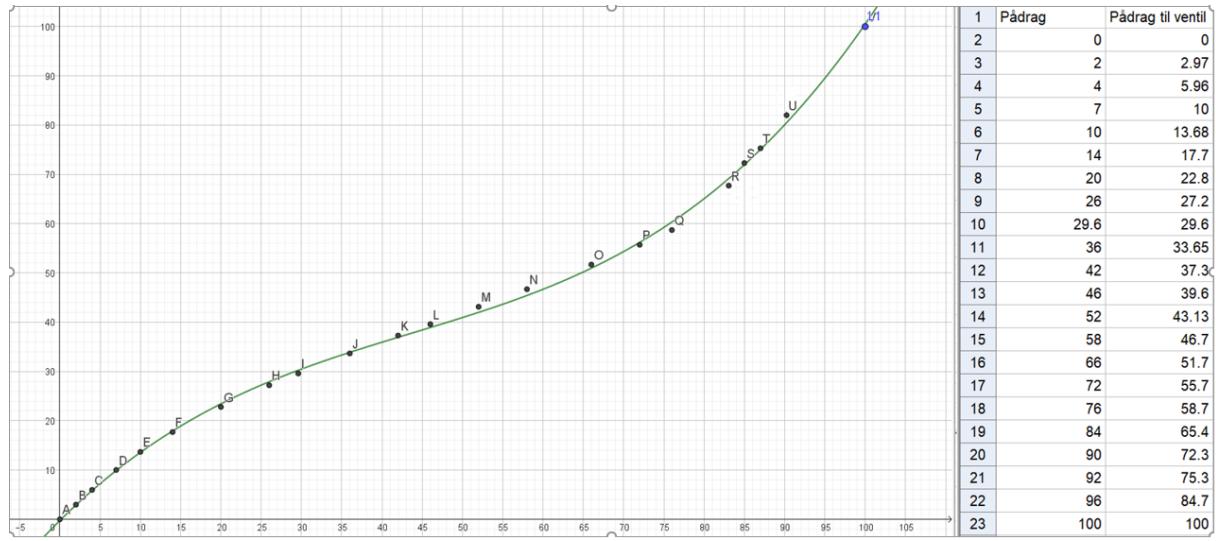


Figur 37: Plotting av de to pådragstabellene (se figur 2 og 3). Den grønne kurven er pådraget uten filter, den svarte kurven er det ønskede pådraget.

Figur 37 viser pådraget (tilnærmet med regresjon i Geogebra) fra ventilen uten filter (grønt) og det «ideelle» pådraget (svart). Her ser man et klart avvik, spesielt når pådraget er rundt  $70 \pm 20$ . Dette vil gjøre reguleringen mer ustabil, spesielt med tanke på foroverkoblingen som er mer avhengig av et lineært pådrag. Dette fordi foroverkoblingen ikke får noen tilbakemelding i form av måling.

Figur 39 viser viser pådraget fra regulatoren i venstre kolonne i tabellen til høyre, sammenliknet med det ønskede pådraget ut til ventilen i høyre kolonne. Pådraget fra regulator i første kolonne må gjennom et filter for å gjøre volumstrømpådraget inn i tanken lineært. Dette er fordi ventilarealet er ulineært ved «lineære» endringer i ventilposisjon. Når regulatoren for eksempel sender ut et pådrag på 66 %, så skal ventilen få et pådrag på 51.7 %. Dette er kun et estimat som ikke vil være helt nøyaktig, men formålet er å eliminere de verste feilmarginene som pådraget hadde.

Ved å plotte pådraget til ventilen som en funksjon av pådraget fra regulatoren, ender man opp med en modell av filteret vist som grafen til venstre i figur 39 som kun er modellert til å fungere mellom [0,100]. Ved små justeringer i tabellen ender man opp med at utgangen blir 0 eller 100 når inngangen til filteret er 0 eller 100.



Figur 38: Modellen av pådragsfilteret.

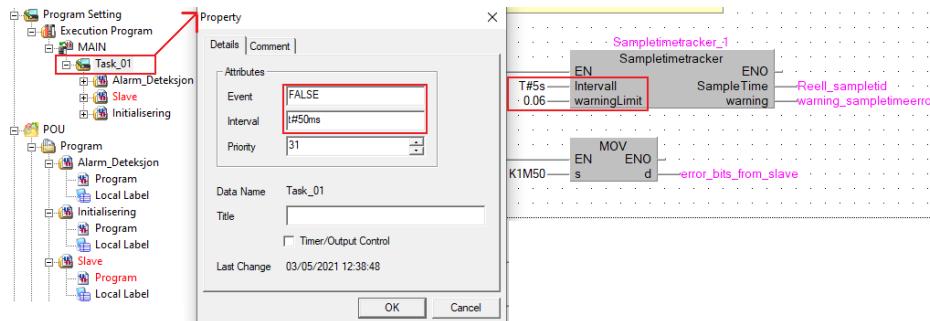
Under vises funksjonen man får ved bruk av regresjon i geogebra. Her er  $P_{\text{filter}}$  lik det pådraget som må sendes til ventilen for å gi et lineært pådrag.  $P$  er pådraget utregnet fra regulator (eller manuell styring).

$$P_{\text{filter}} = 1.9596 \cdot 10^{-4} \cdot P^3 - 2.5722 \cdot 10^{-2} \cdot P^2 + 1.620 \cdot P - 0.2633$$

## 4.9 Valg av samplingstid

For at den diskrete regulatoren skal ligne mest mulig på et kontinuerlig system ønsker man minst mulig samplingstid. Mindre samplingstid fører til en bedre representasjon av signalene.

Slaveprogrammet er utstyrt med en egen funksjonsblokk: «SampletimeTracker». Den har som oppgave å overvåke sampletimeiden. Hvis målt sampletime blir for stor slår denne alarm. Alarmgrensen må bestemmes selv. Det er naturlig at denne er noe høyere enn verdien man låste fast på PLS-en. Man kan stille inn fast samplingstid på GX Works som vist på figur 39.



Figur 39: Innstilling av samplingstid, samt bilde av FB: "SampletimeTracker", alarmgrense på 60ms.

Under ser man programkoden inne i funksjonsblokken: «SampletimeTracker».

```
(*SampleTime programkode*)

DINC(1, counter);
TON_1(IN:=1, PT:=Intervall, Q:=tonFlag, ET:=tonCurrentTlme); (* Start timer *)

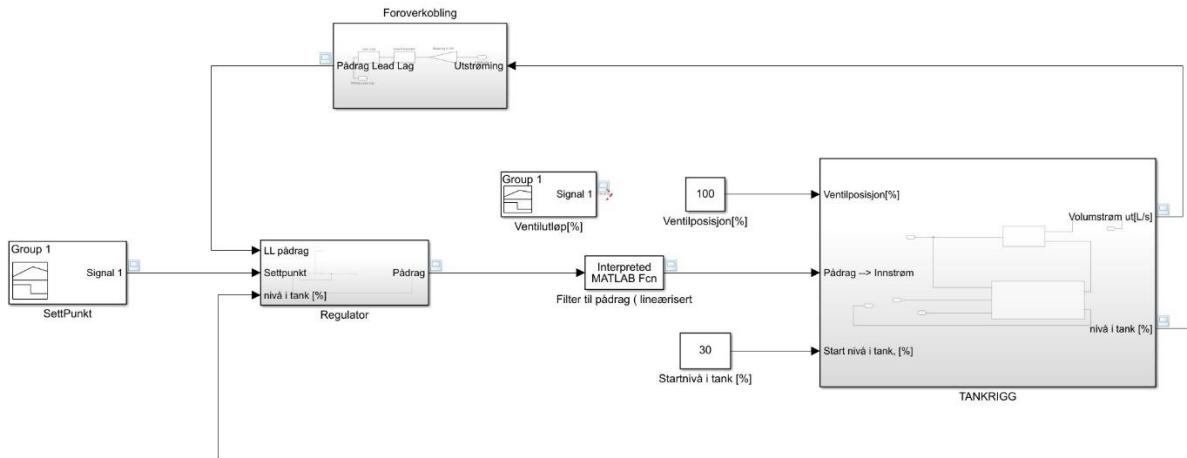
IF tonFlag THEN
    (* Calculate SampleTime *)
    time_DINT := TIME_TO_DINT(tonCurrentTlme); (* Milliseconds *)
    DFLT(1, time_DINT, time_FLT);
    DFLT(1, counter, counter_FLT);
    SampleTime := time_FLT/(counter_FLT*1000.0);

    (* Reset variables *)
    TON_1(IN:=0, PT:=Intervall, Q:=tonFlag , ET:=tonCurrentTlme); (* Reset timer *)
    counter := 0;
END_IF;

(* Warning / Alarm *)
IF SampleTime > warningLimit THEN
    warning := TRUE;
ELSE
    warning := FALSE;
END_IF;
```

## 4.10 Prosessmodell i Matlab (Simulink)

For å enkelt kunne simulere prosessen og teste ulike regulatorparametere ble det laget en modell av prosessen i Simulink. Med denne modellen kan man gjøre sprang i settpunkt og forstyrrelsen, og deretter se hvordan nivået stiller seg inn med valgte regulatorparameter.



Figur 40: Prosessmodell med 3 subsystemer; Tankrigg, Foroverkobling, og Regulator

«Regulator» og «Foroverkobling» fra figur 40 er implementert i PLS rigg, og «TANKRIGG», der selve prosessen foregår, er i tankrigg i den virkelige prosessen. Det blir lest av to verdier fra tankriggen som blir brukt i regulering, nivå og volumstrøm ut (sett på som forstyrrelsen). På grunn av kompleksitet i systemet er denne modellen designet så man kun kan simulere enten med foroverkobling, eller kun med PID, ikke begge deler. Det er ikke helt ideelt, men man kan likevel utføre simuleringer og få brukbare resultater, som kan brukes til justering av parameter.

Denne Simulinkfilen, vist i figuren over, har et tilhørende skript i Matlab. Dette mest på grunn av regulatoren som er i Simulink ikke er likt implementert som den vi har i PLS riggen. Så med dette skriptet gjøres sumform om til formen som PID er implementert i Simulink, så det er mulig å skrive direkte inn de egentlige verdiene. Skriptet er vist under, eller opne vedlegg 8 i Matlab som viser de opprinnelige filene.

```

% Parameter til regulator slave-PLS(SUMFORM)
Kp = 2.8;
Ti = 23;
Td = 0;
n = 10;

%Lead_Lag parametere
K_LL = 0.38;
Tn = 12;
Tt = 5;

% Omgiort parameter til regulatoren som er implementert i Simulink
Tf = Td/n;
N = 1/Tf;
P = Kp;
I = 1/Ti;
D = Td;

T = 0.05 % Tastetid til regulator
sim('Modell_tank');

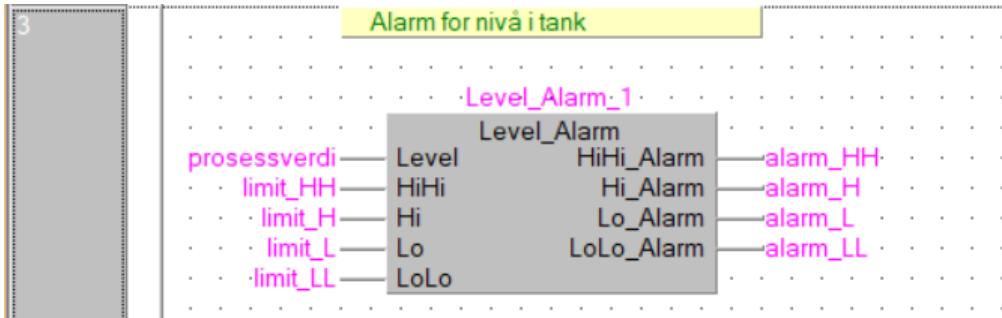
function y = filter(x) ... % Pådragsfilter

```

## 4.11 Alarmdeteksjoner

### 4.11.1 Nivåalarm (HH, H, L, LL)

For håndtering av nivåalarmer til tanken er det laget en enkel funksjonsblokk som sammenligner alarmgrensene med prosessverdien. Alarmgrensene kan man endre i HMI-en.



Figur 41 : MATSER PLS, Funksjonsblokken laget i GX Works 2 for detektering av nivå alarmer.

Når en alarm skal kvitteres settes en bit fra HMI høy («error\_LL\_ACK»), og den vil nullstille alarmsituasjonene som bit-en «error\_LL\_unack» vil aktivere. Hvis feilen fortsatt er til stede vil den bli satt høy igjen (sett-dominans). Logikken, som vist i figur 42, for kvittering av alarmer er gjort likt for alle typer alarmer som master detekterer.



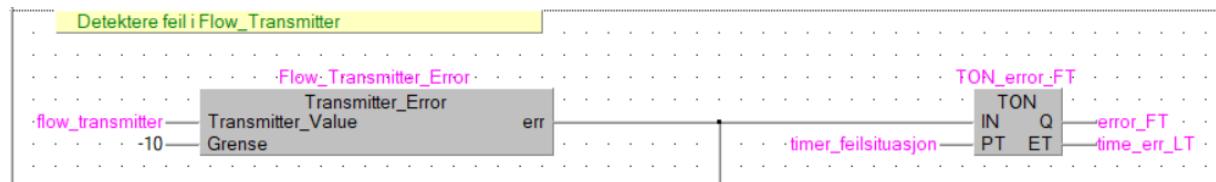
Figur 42: Logikk for kvittering i PLS.

#### 4.11.2 Kommunikasjonsbrudd: LT01 og FT01

Transmitterne LT01 og FT01 har et målesignal på 4-20mA. En verdi godt under 4mA vil være en indikasjon på kommunikasjonsfeil.

Ved kommunikasjonsbrudd på FT01 vil anlegget stoppe, pumpa slås av og utløpsventiler stenges.

Når det er en feil på utstrømningsventil FT01, vil foroverkoblingen i slaven automatisk bli deaktivert. Dette fordi foroverkoblingen er avhengig av målingen til utstrøm for å fungere. Prosessen kan fortsatt kjøre, men ikke med foroverkobling.

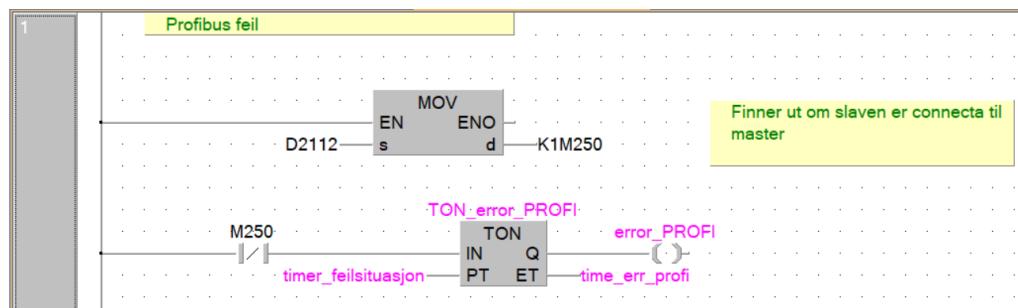


Figur 43: Master PLS, Sjekker verdien til flow\_transmitter, er den under -10, setter error\_FT høy.

#### 4.11.3 Kommunikasjonsbrudd: Profibus, Master

Masteren sjekker om slaven er tilkoblet Profibus. Dette gjøres ved å lese av bufferminne 112, der slavestatuser er lagret. Indeks 0 til dataordet i dette bufferminnet tilhører slaven som blir brukt. Når denne er lav, betyr det at slaven ikke er tilkoblet master PLS. En forsinket innkobling brukes for å unngå triggering av svært kortvarige feil.

Når kommunikasjonen har vært borte over lengre tid går det alarm. Hvis feilen fikses, vil Profibus automatisk kobles til igjen. Operatøren kan kvittere alarmen og starte anlegget igjen uten behov for teknisk hjelp.

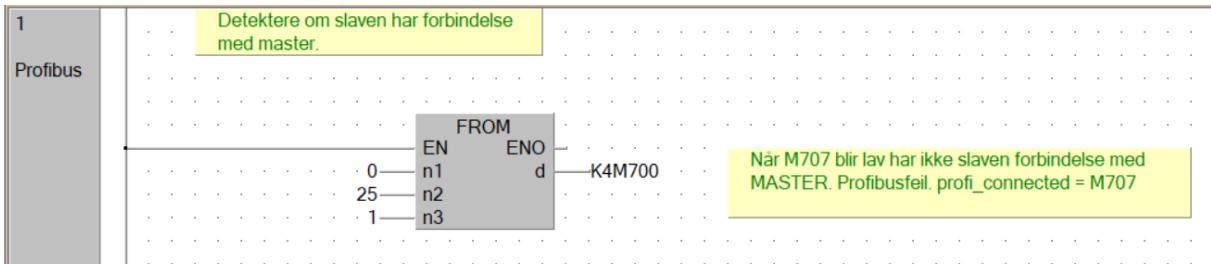


Figur 44: Master PLS, error\_PROF settes høy når slaven ikke er tilkoblet.

#### 4.11.4 Kommunikasjonsbrudd: Profibus, Slave

Slaven sjekker om den har forbindelse til master ved å sjekke bufferminne 25. Indeks 7 i dette dataordet forteller om den har forbindelse til master. Når bit-et er høyt har den forbindelse, er den lav, har den ikke det.

Har ikke slaven forbindelse til master, vil den automatisk sette pådraget til 0, og stenge alle ventiler.



Figur 45: Slave PLS, M707, Profibus\_connected.

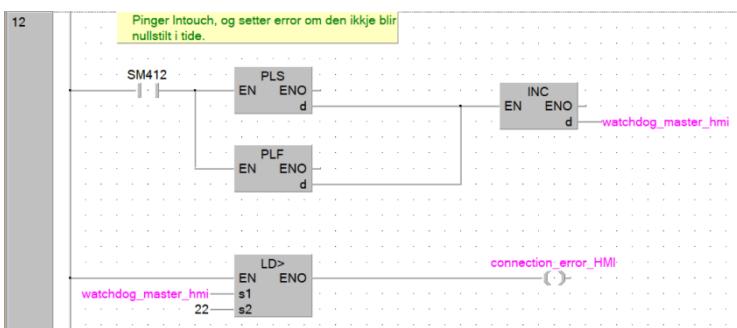
#### 4.11.5 Kommunikasjonsbrudd mellom PLS og HMI

For å detektere kommunikasjonsbrudd mellom PLS og HMI brukes overvåkning i form av en «watchdog timer». «A watchdog timer is a simple countdown timer which is used to reset a microprocessor after a specific interval of time.» (Maxim Integrated Products, Inc., 2001)

I masteren er det satt et heltall som øker to ganger i sekundet. Dette dataordet sendes direkte til InTouch og der sjekkes det hvert andre sekund om det har endret seg siden sist. Om tallet er likt som ved forrige syklus betyr dette at InTouch ikke har forbindelse til master.

Når tallet har passert en grenseverdi skal InTouch nullstille dataordet. Masteren kan da detektere om InTouch er tilkoblet eller ikke ved hjelp av en alarm som sjekker om dataordet blir nullstilt eller ikke. Dette kan ses i figur 46. Samme type løsning er implementert i slaven. Dette gjør at InTouch kan oppdage om det er slave eller master som har kommunikasjonsbrudd.

Dette er satt opp på en slik måte at det ikke er mulig at dataordet er det samme to målinger på rad med mindre det er kommunikasjonsfeil.



Figur 46: Inkrementerer et dataord to ganger i sekundet, setter error om dataordet passerer 22.

Nedenfor ser man koden til watchdogfunksjonen i Intouch. Koden kjører hvert 1500ms

```
{ Watchdog master }
IF local_watchdog_master <> prev_watchdog_master_hmi THEN
    error_com_master = 0;
ELSE
    error_com_master = 1;
```

```

ENDIF;
prev_watchdog_master_hmi = local_watchdog_master;

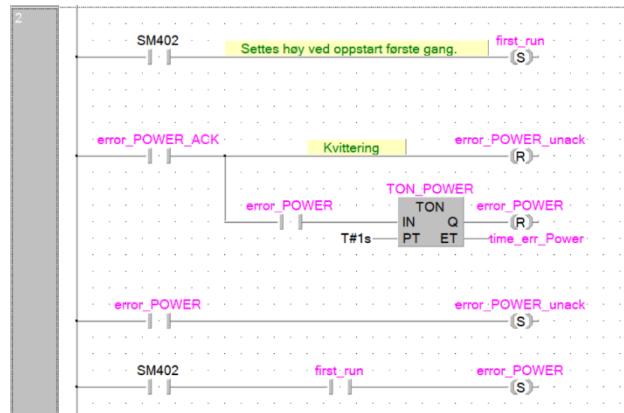
IF local_watchdog_master > 15 THEN
    local_watchdog_master = 0;
ENDIF;

{ Watchdog slave }
IF local_watchdog_slave <> prev_watchdog_slave_hmi THEN
    error_com_slave = 0;
ELSE
    error_com_slave = 1;
ENDIF;
prev_watchdog_slave_hmi = local_watchdog_slave;

```

#### 4.11.6 Deteksjon av strømbrudd

For å detektere strømbrudd settes en bit høy ved første oppstart. Når initieringsminnecelle SM402 i master er høy, samtidig som first run er høy, indikerer dette at PLS har vært skrudd av, for eksempel ved strømbrudd. Om strømmen går og kommer tilbake vil «error\_POWER» bli satt høy, og man vil få en alarm i HMI, når man kvitterer vil feilen forsvinne. All hardware og kommunikasjon starter av seg selv så fort strømmen er tilbake.



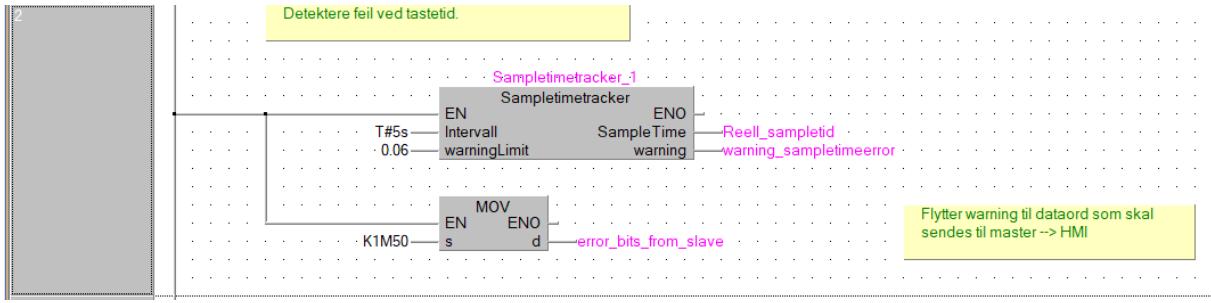
Figur 47: error\_POWER settes når initieringsminnecelle er høy og oppstart er overstått.

#### 4.11.7 Høy samplingstid

Etter man har låst fast samplingstiden kan man stille inn alarmgrensen på funksjonsblokken: «Sampletime tracker», ved å stille på konstanten på inngang: «warningLimit». Det er viktig at denne ligger noe over samplingstiden man har låst fast PLS-en, for å ikke få en uønsket alarm.

Hvis samplingstiden overstiger grensen ved senere tid, kan det skyldes sliten hardware eller tyngre programkoder i PLS. Hvis dette skjer er det naturlig å kontakte en teknikker for å øke samplingstiden inne på PLS-en, samt oppdatere PID-regulatoren deretter. Hvis man kjører prosessen på tidligere tunede parametere etter endring i samplingstid, vil det være aktuelt å tune PID-regulatoren på nytt.

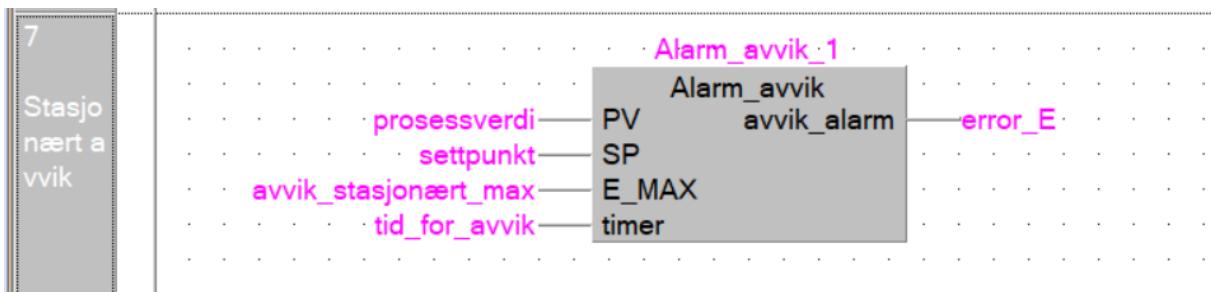
Alarmsignalet blir sendt til master PLS og videre til iX-Developer HMI panel og InTouch.



Figur 48: Slave PLS, POU: Alarm\_Deteksjon, NW: 2

#### 4.11.8 Stasjonært avvik

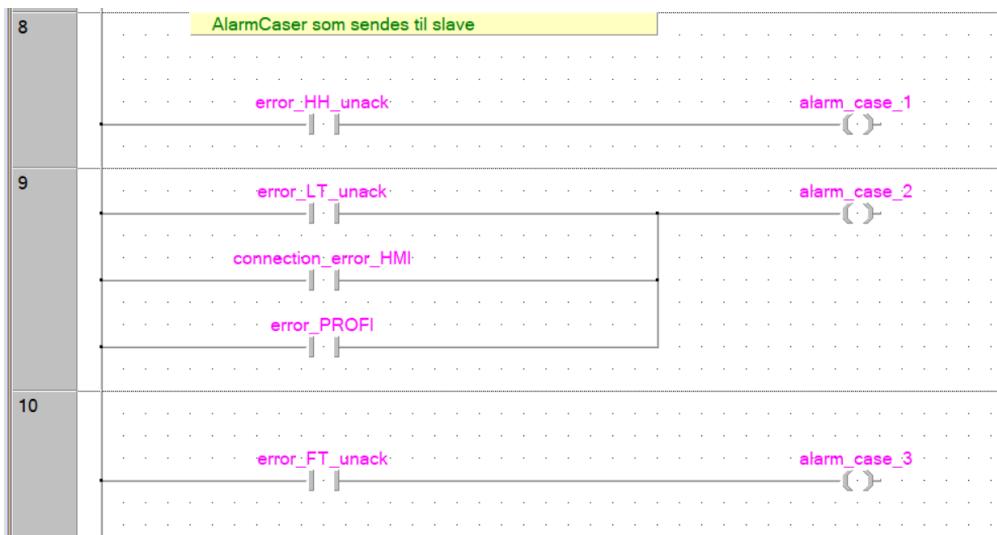
Deteksjon av stasjonært avvik sammenligner prosessverdien og settpunkt. Om differansen blir større enn «avvik\_max» i mer enn «tid\_for\_avvik» sekunder vil bit «error\_E» bli satt høy. Denne alarmen vil ikke føre til noen aksjoner, men er kun en advarsel. Tiden som settes må være lengre enn normal innsvingingstid.



Figur 49: Funksjonsblokk, alarmavvik. Deteksjon av stasjonært avvik

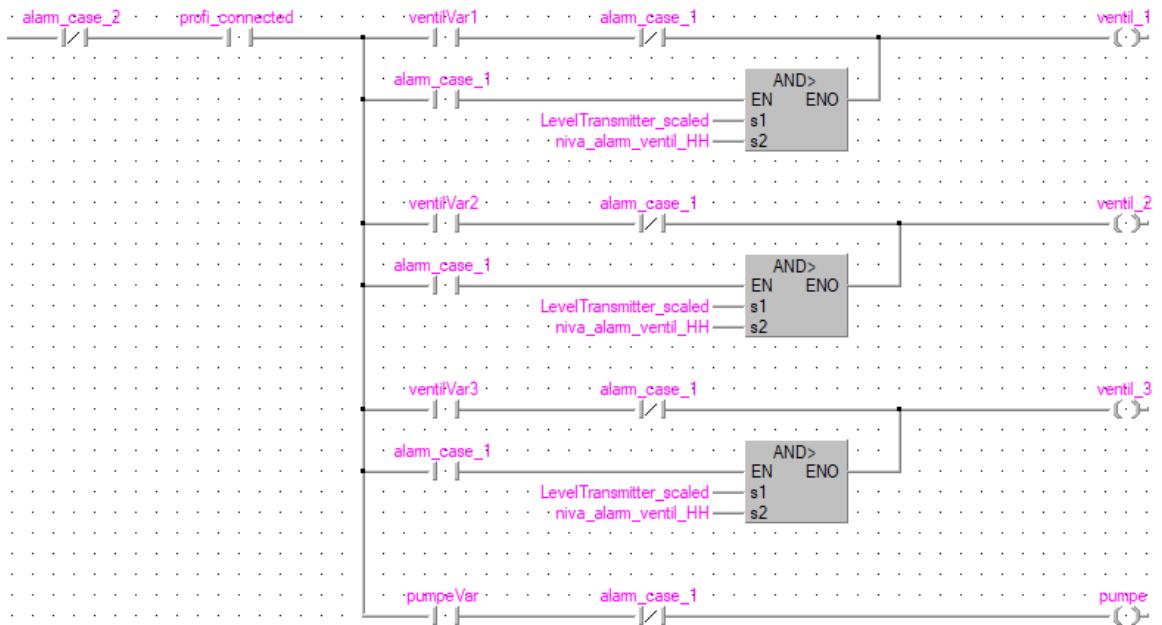
## 4.12 Alarmaksjoner

Når det er ukvitterte alarmer vil dette trigge ulike alarmaksjoner som hovedsakelig brukes i slaven for å utføre nødhandlinger. Ved alarm må feilen forsvinne og kvitteres for at alarmaksjonene skal bli lave. Det holder ikke bare å kvittere alarmen.



Figur 50: Master PLS, alarmaksjoner som sendes til slave

I slaven hentes «alarm\_case\_2» og «alarm\_case\_3» fra Profibus. Ved Profibusfeil har slaven en egen variabel, «profi\_connected», for å aktivere «alarm\_case\_2». Hvis «alarm\_case\_2» aktiveres blir ventilene stengt og pumpen slått av. Ved «alarm\_case\_1» åpnes ventilene til nivået i tanken er senket til 50 %. Pumpen blir slått av og pådraget blir dermed satt lik null. Koden er vist i figur 51 og 52.



Figur 51: Alarmcase 1 og 2 som overstyrer pumpe og ventilér i slave



Figur 52: Pumpen stopper pådraget

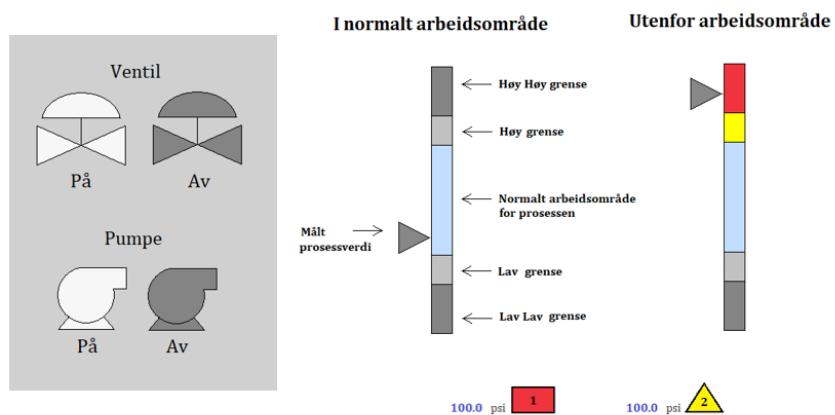
## 4.13 Brukergrensesnitt

### 4.13.1 High Performance HMI

Brukergrensesnittet i InTouch er utviklet ut i tråd med en High Performance HMI stilguide, men med noen modifikasjoner som passer denne prosessen. High Performance, eller ISA101, er en standardisert modell for HMI utviklet av International Society of Automation for å gjøre det enklere for prosessoperatører å navigere ved hjelp av flytskjema-lignende prosessbilder med kun nødvendig informasjon og et fåtall farger. InTouch følger fargeskalaen som er listet opp under, mens IX-Developer har en litt annen utforming på grunn av begrensninger i programvaren. Ved å benytte tabell 2 har man et brukergrensesnitt som tydelig skiller mellom ulike komponenter og tilstander som vist i figur ??

**Tabell 2: Fargebruk i HMI**

Farge	RGB Value	Sample	Bruk
Hvit	255, 255, 255		Eventuelle fremhevinger
Lys grå	240, 240, 240		Indikator for at utstyr står på.
Grå	208, 208, 208		Bakgrunnsfarge for prosessbilder
Mørk grå	174, 170, 170		Fremheving av symbol og bakrunner
Mørkere grå	118, 113, 113		Indikator for at utstyr er av
Svart	0, 0, 0		Tekst, hovedprosesslinjer, outline
Mørk blå	0, 0, 215		Prosessverdier, avlest verdier, prosessverdi trend
Lys blå	187, 224, 227		Ønsket arbeidsområde
Mørk grønn	0, 128, 0		Input fra operator, settpunkt i trend
Rød	255, 43, 43		Prioritet 1 alarm
Gul	255, 255, 0		Prioritet 2 alarm
Oransje	255, 102, 0		Flowtransmitter trend (forstyrrelse)
Mørk magenta	204, 0, 102		Prosessverdi trend



Figur 53: High Performance HMI

### 4.13.2 HMI: InTouch

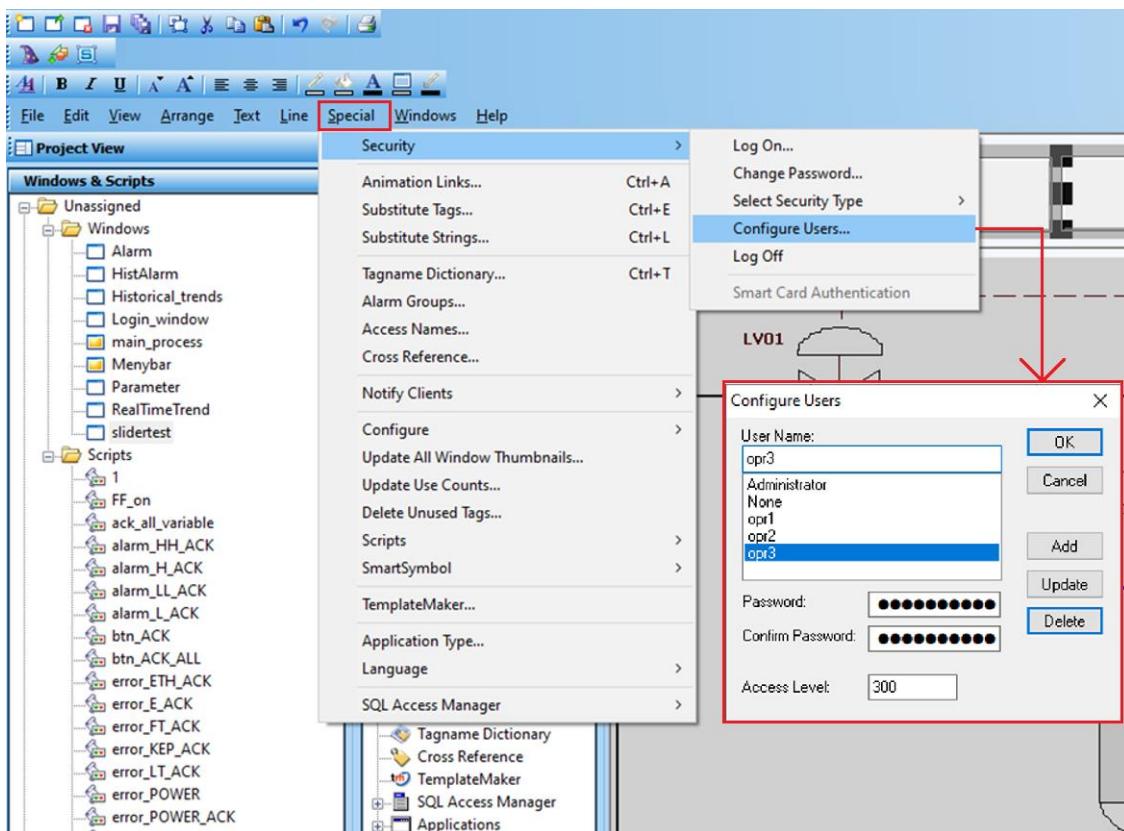
HMI-programmet for PC åpnes i InTouch og startes ved å klikke på «Runtime».

Løsningen er laget slik at enkelte funksjoner krever at operatøren et visst tilgangsnivå. Dette for å begrense muligheten for feilsituasjoner som kan skyldes endringer gjort av operatører uten riktig kompetanse. Brukergrensesnittet på InTouch har tre passordbeskyttede tilgangsnivåer på henholdsvis 100, 200 og 300. I tabell 3 ser man en oversikt innloggingsdata for de tre operatørene.

Tabell 3: Oversikt over ulike operatører med tilhørende brukernavn, passord og tilgangsnivå

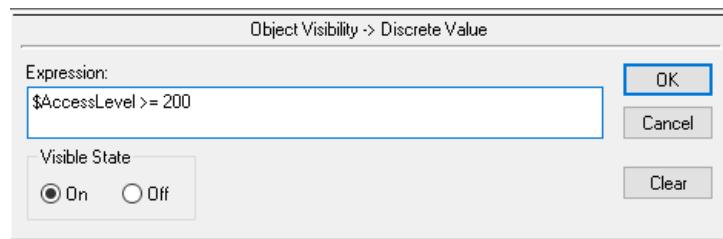
Operatør	brukernavn	passord	tilgangsnivå
1	opr1	opr1	100
2	opr2	opr2	200
3	opr3	opr3	300

Figur 54 viser hvordan man legger til flere operatører. Her opprettes brukernavn og passord, og man definerer hvilket tilgangsnivå operatøren skal ha. En kan også endre tilgangsnivået til en eksisterende operatør på samme sted.



Figur 54: InTouch - Configure Users

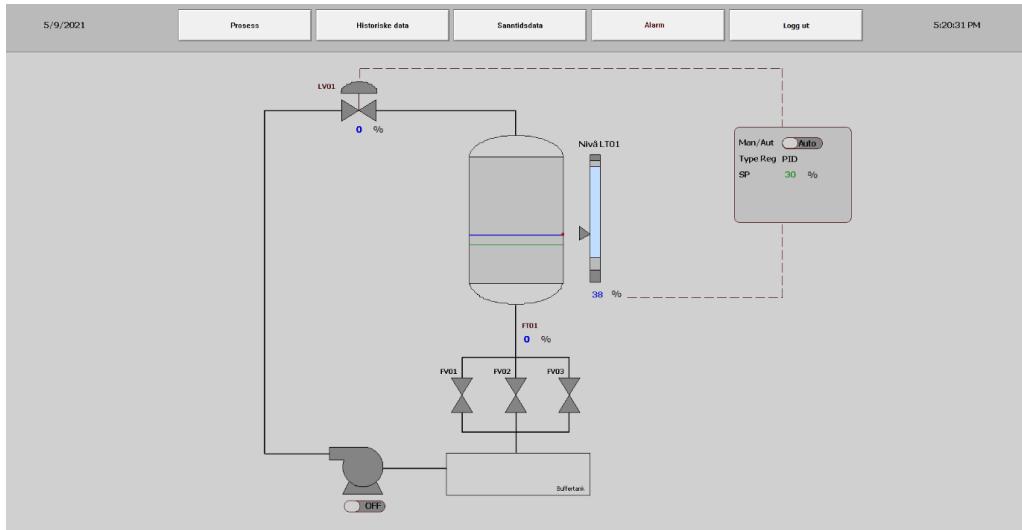
For å begrense hva hvert tilgangsnivå kan se eller gjøre legger man til en kommando under «Expression» når man redigerer funksjonaliteten til et objekt. Figur 55 viser eksempel på hvordan et objekt kan gjøres synlig for en operatør med tilgangsnivå fra og med 200.



Figur 55: InTouch kode som bruker AccessLevel

#### 4.13.3 Oversikt

For å kunne benytte seg av HMI-en må operatører først logge inn med brukernavn og passord. Startsiden til HMI-en på PC viser en oversikt over hele prosessen. Dette inkluderer reguleringssventilen, tanken, tre utløpsventiler, pumpen og eventuelle regulatorparametere. I navigasjonsbaren øverst på siden kan man gå mellom vinduene «Historiske data», «Sanntidsdata» og «Alarmer». Figur 56 viser prosessbildet innlogget som «opr1».



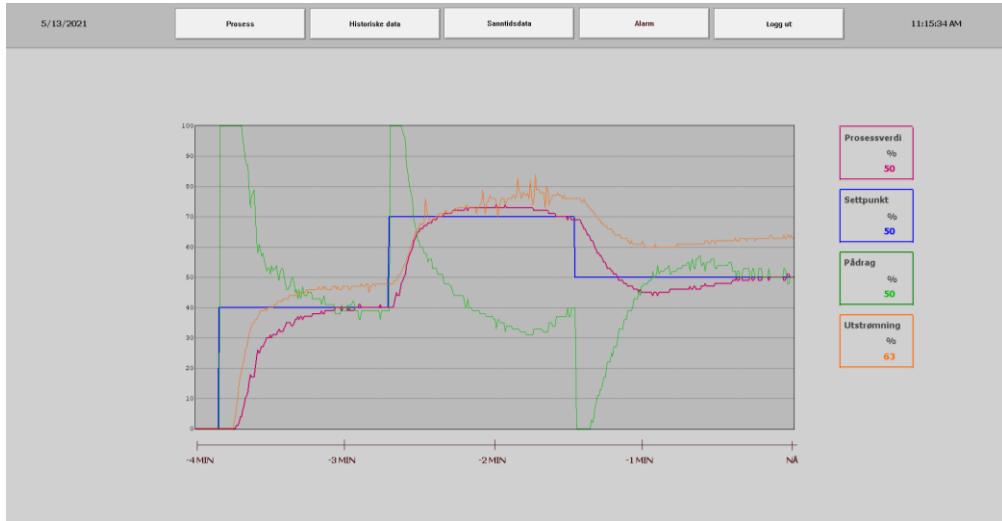
Figur 56: Prosessbilde, innlogget som opr1

Navigasjonsbaren finnes øverst i alle vindu og man kan når som helst enkelt navigere tilbake til prosessbildet ved å klikke på «Prosess». «Historiske data» viser all data som er lagret og man har her mulighet til å velge hvilken periode man ønsker å se data for. I tillegg kan man skalere y-aksen og tidsaksen, flytte på avlesningsmarkører, samt gå tilbake i tid.



Figur 57: InTouch - Historiske data

«Sanntidsdata» viser grafer over prosessverdi, settpunkt, pådrag og utstrømming som oppdateres i sanntid. Her kan man følge med på hvordan enkelte deler av prosessen reagerer ved å gjøre parameterendringer.



Figur 58: InTouch – Sanntidsdata

«Alarm» viser en oversikt over alle alarmer og her kan operatører med et visst tilgangsnivå kvittere en eller flere av disse. Ved å klikke på «Alarmhistorikk» til høyre for alarmlisten får man en oversikt over alarmer tilbake i tid.



Figur 59: InTouch - Alarm

#### 4.13.4 Alarmvisualisering i InTouch

Det skiller mellom kritisk og ikke-kritisk alarm. For å gjøre det enklere å utvide HMI-programmet for flere alarmer er det laget et skript for alarmprioritetene som styrer blinking og synlighet av alarmvisualiseringen. For å legge til nye alarmer må disse skrives inn i dette scriptet under Application Script.

Tabell 4 og 5 viser hvordan alarmer visualiseres som kvittert og ikke kvittert, både enkeltvis og flere alarmer samtidig.

**Tabell 4:** Alarmvisualisering - enkeltvis

Almprioritet	Beskrivelse	Visualisering	
		Ikke kvittert	kvittert
1	Kritisk alarm	Rød varselfirkant blinker ved prosessverdien eller komponenten alarmen tilhører.	Rød varselfirkant synes ved prosessverdien eller komponenten alarmen tilhører.
		Knapp til alarmvindu i menylinjen blinker rødt.	Rød ramme rundt knapp til alarmvindu i menylinjen.
2	Ikke-kritisk alarm	Gul varseltrekant blinker ved prosessverdien eller komponenten alarmen tilhører.	Gul varseltrekant synes ved prosessverdien eller komponenten alarmen tilhører.
		Knapp til alarmvindu i menylinjen blinker gult.	Gul ramme rundt knapp til alarmvindu i menylinjen.

**Tabell 5:** Visualisering av flere alarmer samtidig

<b>Visualisering av flere alarmer samtidig</b>		
<b>Prioritet 1</b>	<b>Prioritet 2</b>	<b>Visualisering</b>
Ikke kvittert	Ikke kvittert	Rød firkant blinker i prosessbildet. Gul trekant synes ikke hvis den tilhører samme komponent.
Ikke kvittert	Kvittert	Knapp til alarmvindu blinker rødt Rød firkant blinker i prosessbildet. Gul trekant synes ikke hvis den tilhører samme komponent.
Kvittert	Ikke kvittert	Knapp til alarmvindu blinker rødt Rød firkant vises i prosessbildet. Gul trekant synes ikke hvis den tilhører samme komponent.
Kvittert	Kvittert	Knapp til alarmvindu blinker gult og rød ramme rundt knapp til alarmvindu. Rød firkant vises i prosessbildet. Gul trekant synes ikke hvis den tilhører samme komponent
		Rød ramme rundt knapp til alarmvindu

## 4.14 HMI: IX Developer

HMI-panelet ved tankriggen er en forenklet utgave av HMI-en på PC. Programmet åpnes i IX Developer og lastes opp til panelet ved å klikke «Download». Merk at en må være tilkoblet tankriggen via WiFi for å kunne laste opp. Etter opplastning vil programmet lagres lokalt på riggen, så om det gjøres endringer må programmet lastes opp på nytt.

Av praktiske hensyn er det ingen innlogging på tankriggens HMI. Tabellen under viser hvilke funksjoner alle operatører har tilgang til.

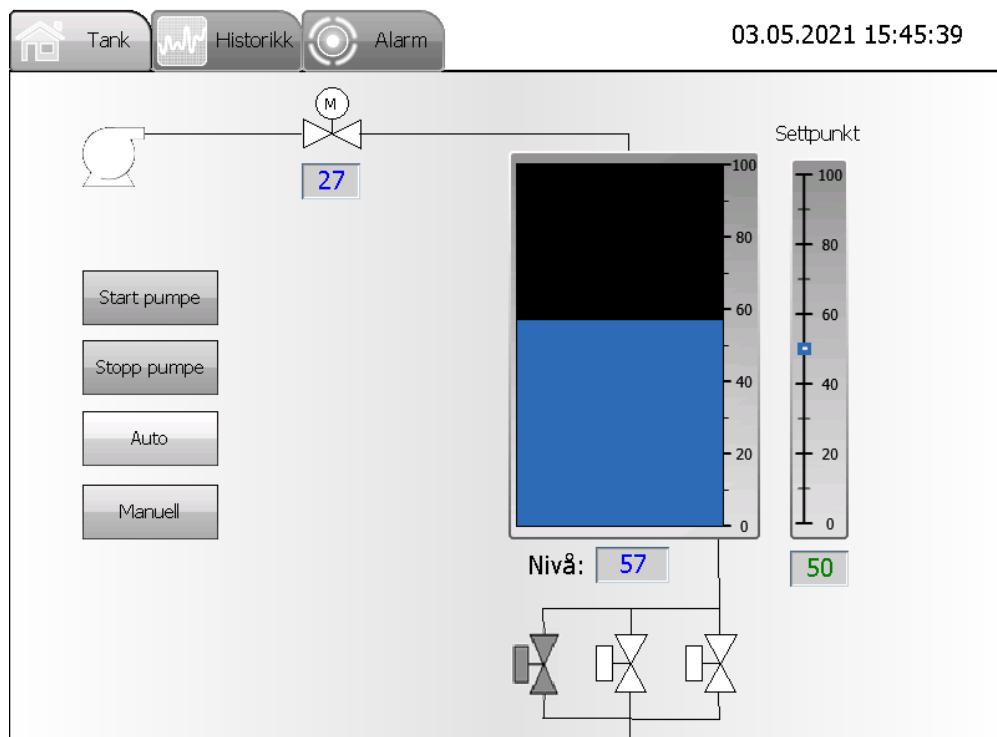
Tabell 6: Oversikt over muligheter på IX-panelet

Variabel	Skrives	Leses
Referanse til nivået	X	X
Nivået i tank		X
Manuelt pådrag	X	X
Auto/manuell tilstand	X	X
Pådrag fra regulatoren		X
Melding om alarmer		X
Kvittering av alarmer	X	
Start/stopp pumpe	X	X
Åpne magnetventiler (3 stk)	X	X

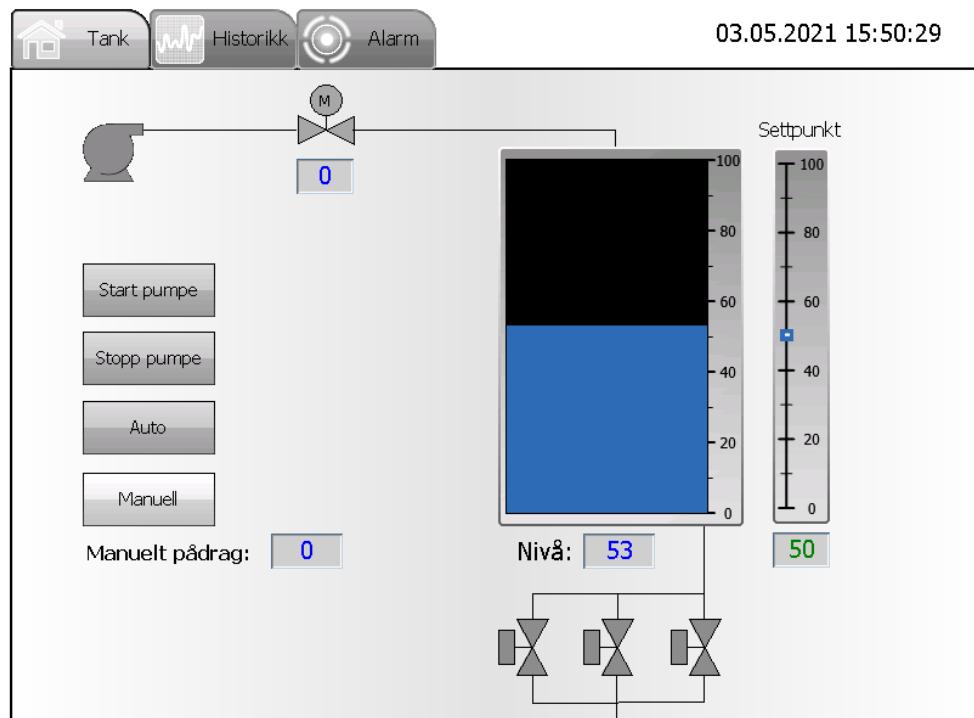
### 1.1.1 Oversikt over IX Developer HMI

«Tank» viser et enkelt prosessbilde av tanken med mulighet til å styre hovedfunksjonene. Man kan velge mellom automatisk eller manuell styring av pådragsventilen ved hjelp av knappene til venstre. Knappen for aktiv tilstand vil bli lysere grå som en indikasjon på dette. Ved automatisk regulering vil man kunne lese og skrive settpunkt ved hjelp av glidebryteren. Mulighet for å lese og endre manuelt pådrag kommer til syne når manuell styring er valgt.

Pumpen startes og stoppes ved å klikke på knappene til venstre. Pumpesymbolet skifter farge i henhold til fargetabellen for High Performance HMI for å indikere om pumpen er av eller på. Dette gjelder også ventilene. Disse åpnes og lukkes ved å klikke direkte på ventilsymbolene.

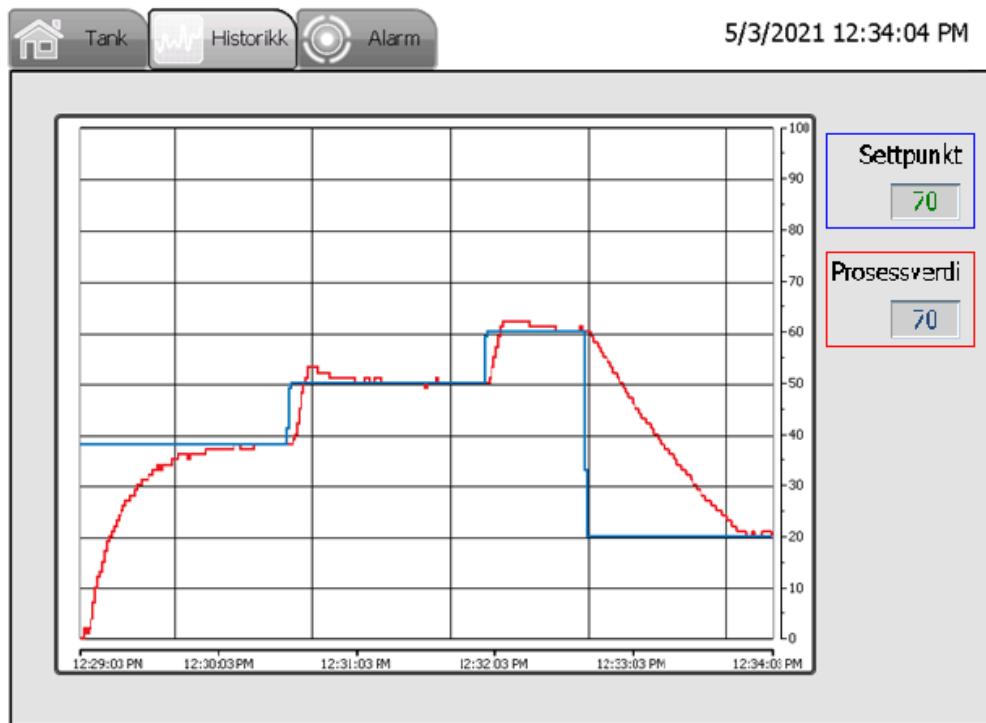


Figur 60: Automatisk justering av tank, to av tre ventiler åpne, pumpe på.



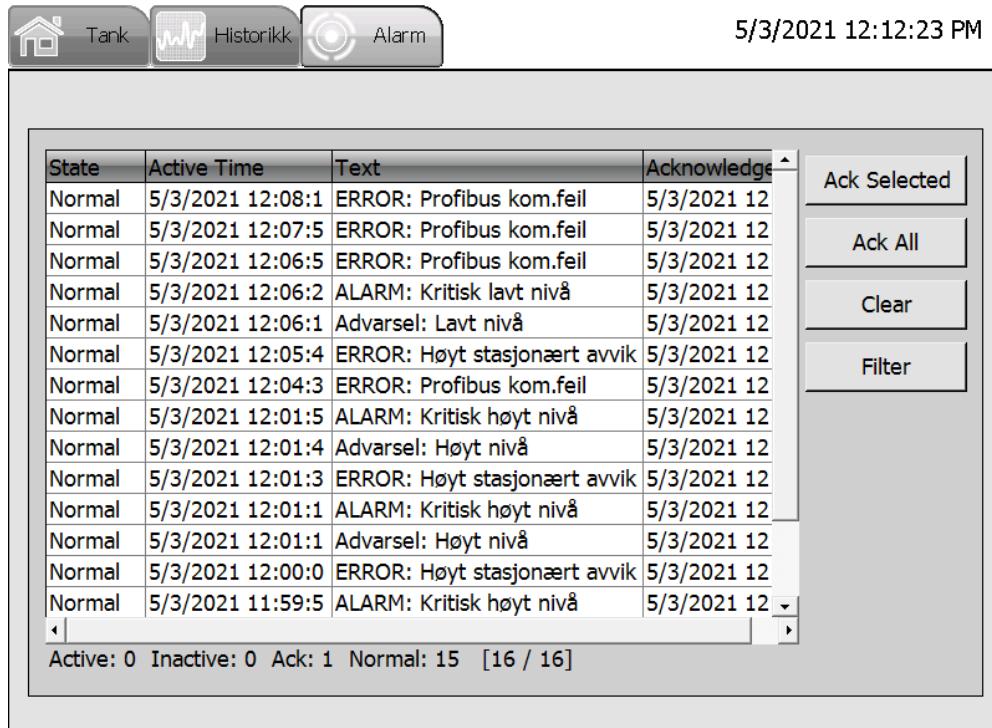
Figur 61: Manuell justering av tank, alle ventiler lukket, pumpe av.

«Historikk»-fanen oppdateres i sanntid og viser grafer for settpunkt og nivået i tanken for de siste fem minuttene. Figur 62 viser historikk fra tom tank til 20 % nivå med sprang i settpunkt underveis for å illustrere hendelsesforløpet.



Figur 62: "Historikk"-vindu

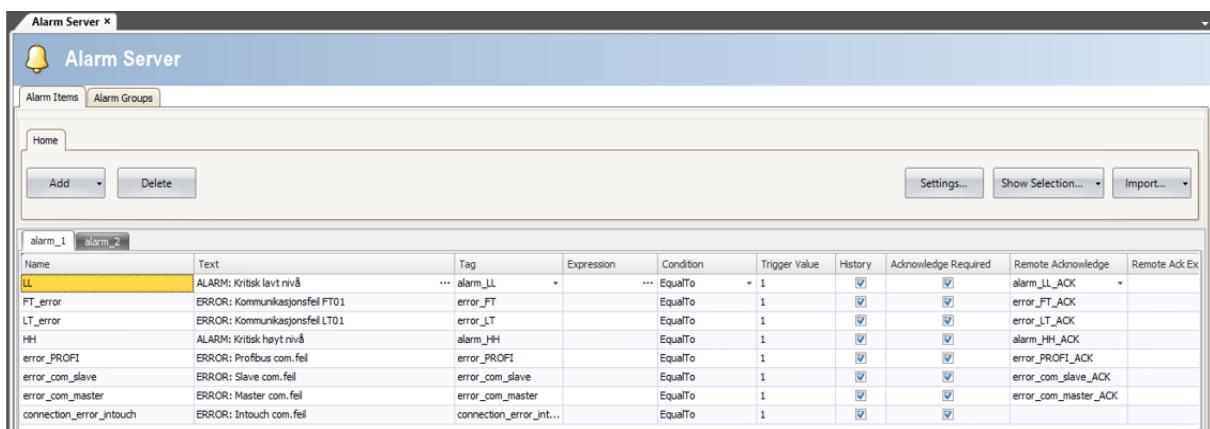
«Alarm» viser en oversikt over alle stående, ikke kvitterte og kvitterte alarmer. Her kvitterer operatøren enten for enkeltalarmer eller alle stående alarmer. Operatøren kan i tillegg velge hvilke alarmer som skal synes ved hjelp av filtreringen, samt fjerne kvitterte alarmer. Ved å trykke på Clear vil alle kvitterte alarmer uten eksisterende feil forsvinne. Figur 63 viser hvordan alarm-vinduet ser ut når alle alarmer er kvittert og anlegget har normal drift uten feilsituasjoner.



Figur 63: «Alarm»-vindu med kvitterte alarmer

#### 4.14.1 Alarmvisualisering: IX Developer

Alarmvisualiseringen på tankriggens HMI er utformet så likt som mulig som HMI for PC. Alarmer kan endres og legges til i «Alarm Server» etter prioritet. Kritisk alarm har prioritet 1 og ikke-kritisk alarm har prioritet 2. Figur 64 og 65 viser oversikt over Alarm Server.



Figur 64: Alarmprioritet 1 i IX-Developer, Alarm server

Alarm Server x

The screenshot shows the 'Alarm Items' tab selected in the IX-Developer Alarm Server interface. The table lists five alarm items with the following details:

Name	Text	Tag	Expression	Condition	Trigger Value	History	Acknowledge Required	Remote Acknowledge	Remote Ack Ex
H	Advarsel: Høyt nivå	... alarm_H	... EqualTo	1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	alarm_H_ACK	
L	Advarsel: Lavt nivå	alarm_L	EqualTo	1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	alarm_L_ACK	
E_error	ERROR: Høyt stasjonært avvik	error_e	EqualTo	1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	error_E_ACK	
error_POWER	ERROR: Strombrudd	error_POWER	EqualTo	1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	error_POWER_ACK	
warning_samplingtid	Advarsel: Fel på samplingstid	warning_samplingtime	EqualTo	1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	warning_samplingtime...	

Figur 65 Alarmprioritet 2 i IX-Developer, Alarm server

## 5 Resultater

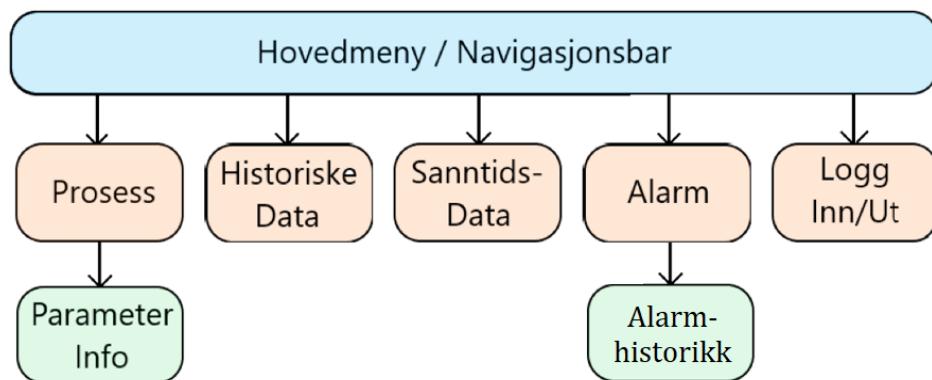
### 5.1 Kommunikasjon

Fullstendig oversikt over kommunikasjonsflyt mellom InTouch/IX-panelet og slaven finnes i vedlegg 12 – kommunikasjonsoversikt.

### 5.2 HMI

#### 5.2.1 Hierarki

Navigeringen er gjort på en måte som gjør det enkelt og oversiktlig, med fokus på å holde antall nivåer til et minimum, samt unngå flere veier til samme vindu.



Figur 66: Hierarki, InTouch

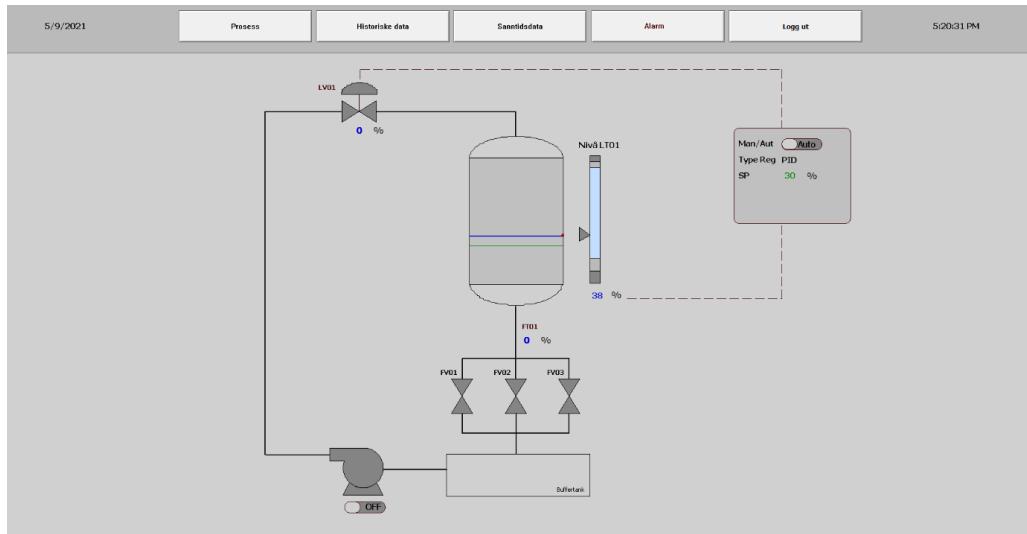
#### 5.2.2 Tilgangsnivåer

Symbolene i InTouch er programmert med tilgangsnivå istedenfor brukernavn for at det skal være enkelt å utvide programmet og legge til nye brukere uten å måtte gjøre omfattende endringer. De tre forhåndsdefinerte brukerne har ulike lese og skrivetilganger. Disse er listet opp i tabell 7.

**Tabell 7:** Oversikt over muligheter hver operatør har i programmet

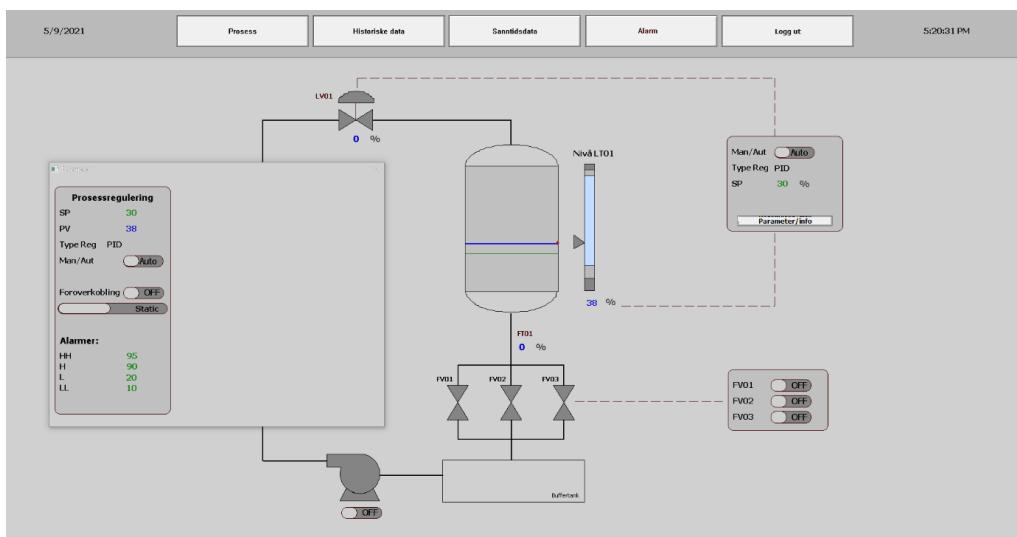
Variabel	Leses			Skrives		
	Operatør 1	Operatør 2	Operatør 3	Operatør 1	Operatør 2	Operatør 3
Referanse til nivået	X	X	X		X	X
Nivået i tank	X	X	X			
Manuelt pådrag		X	X		X	X
Auto/Manuell tilstand	X	X	X		X	X
Valg av regulatortype		X	X			X
Valg av pådragsorgan		X	X			X
Regulatorparameter			X			X
Nominelt pådrag			X			X
Pådrag fra regulatoren	X	X	X			
Foroverkoblingsparameterne			X			X
Valg av foroverkoblingstype		X	X			X
Måling fra strømmingsmåler	X	X	X			
Melding om alarmer	X	X	X			
Kvittering av alarmer					X	X
Start/stopp pumpe	X	X	X		X	X
Alarmgrenser (HH, H, L, LL)	X	X	X		X	X

Ved å logge inn som Operatør 1 vil man kun ha begrenset lesetilgang. Operatør 1 kan overvåke systemet og se informasjon på «Historikk» og «Alarm», men kan ikke gjøre noen endringer eller kvittere eventuelle alarmer. Dette ses i figur 67.



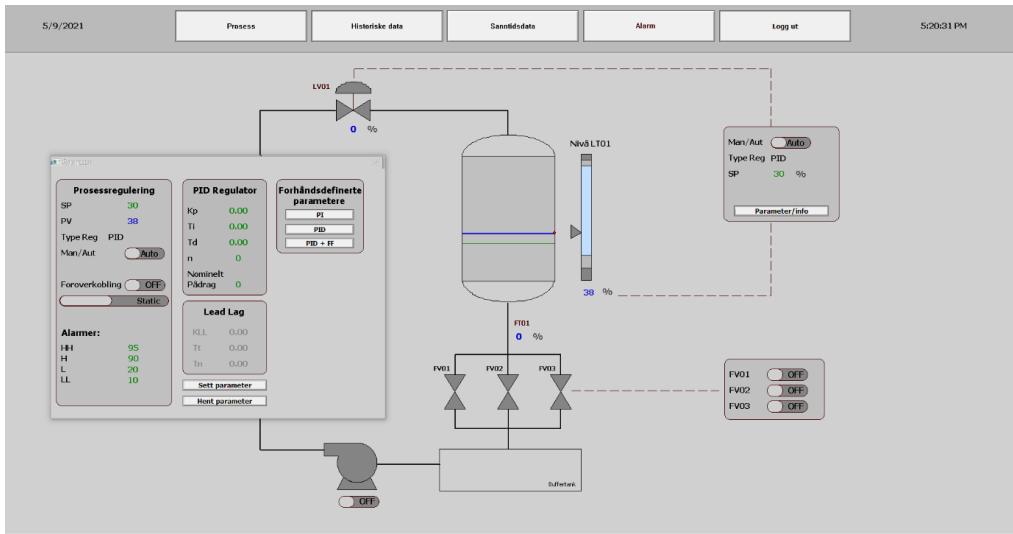
Figur 67: Prosessbilde, logget inn som opr1

Ved å logge inn som operatør 2 får man opp ytterligere informasjon som vist i figur 68. Operatørnivå 2 egner seg godt til prosessarbeideren som skal regulere systemet på daglig basis, men ikke har behov for å endre på reguleringsparameterne til PID og Lead-Lag.



Figur 68: Prosessbilde, logget inn som opr2

Ved å logge inn som Operatør 3 vil man få full lese- og skrivetilgang til all funksjonalitet i InTouch. Figur 69 viser hvordan prosessbildet og «Parameter/Info»-vinduet ser ut for Operatør 3. Det er nå mulig å endre på parameterne til regulatorene, enten ved å skrive inn manuelt eller ved å velge mellom noen forhåndsdefinerte sett for PI, PID og FF. Operatør 3 er tiltenkt teknisk personell som ingeniører og teknisk sjef.



Figur 69: Prosessbilde, logget inn som opr3

## 5.3 HMI: Alarmer – InTouch

### 5.3.1 Alarmsituasjoner

**Tabell 8:** Alarmsituasjoner

Navn	Tagname	Prioritet	Beskrivelse	Alarmaksjon
Nivåalarm: HøyHøy	alarm_HH	Kritisk	Svært høyt nivå i tank.	1
Nivåalarm: Høy	alarm_H	Ikke-kritisk	Høyt nivå i tank.	-
Nivåalarm: Lav	alarm_L	Ikke-kritisk	Lavt nivå i tank.	-
Nivåalarm: LavLav	alarm_LL	Kritisk	Svært lavt nivå i tank.	-
Kommfeil: LT	error_LT	Kritisk	Kommunikasjonsbrudd med LT01	2
Kommfeil: FT	error_FT	Kritisk	Kommunikasjonsbrudd med FT01	3
Kommfeil: Master	error_com_master_hmi	Kritisk	Kommunikasjonsbrudd mellom master PLC og inTouch HMI.	2
Kommfeil: Slave	error_com_slave_hmi	Kritisk	Kommunikasjonsbrudd mellom slave PLC og inTouch HMI.	2
Kommfeil: Profi	error_PROF1	Kritisk	Kommunikasjonsbrudd profibus	2
Strømbrudd	error_POWER	Ikke-kritisk	Det har vært strømbrudd på anlegg. Men strømmen er tilbake.	-
Warning: Sampletime	warning_samplingtime	Ikke-kritisk	For høy samplingstid	-

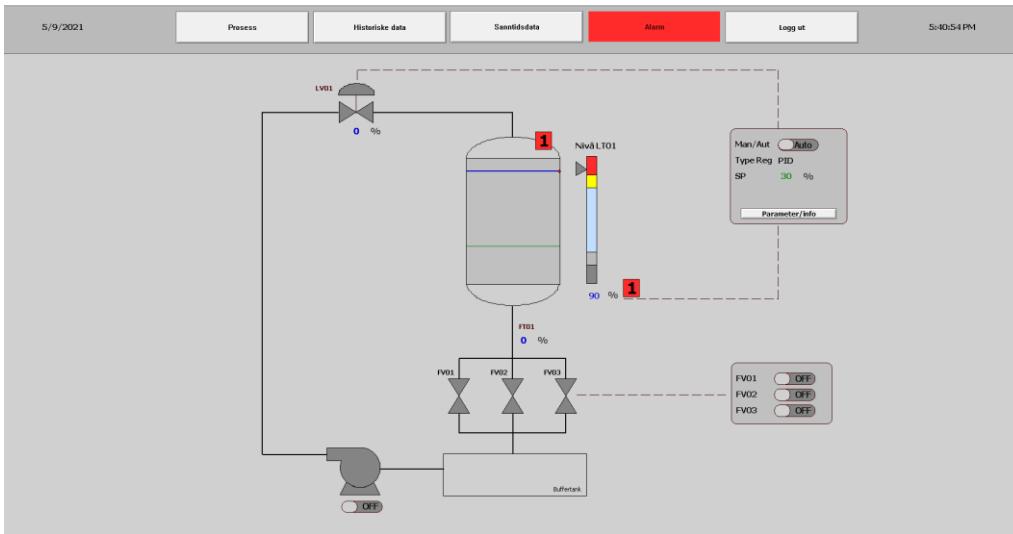
Warning: Stasjonært avvik	error_E	Ikke-kritisk	Prosessverdi har ikke vært nært settpunktet på lenge. Indikerer stasjonært avvik.	-
------------------------------	---------	--------------	---	---

### 5.3.2 Alarmaksjoner

**Tabell 9:** Alarmaksjoner

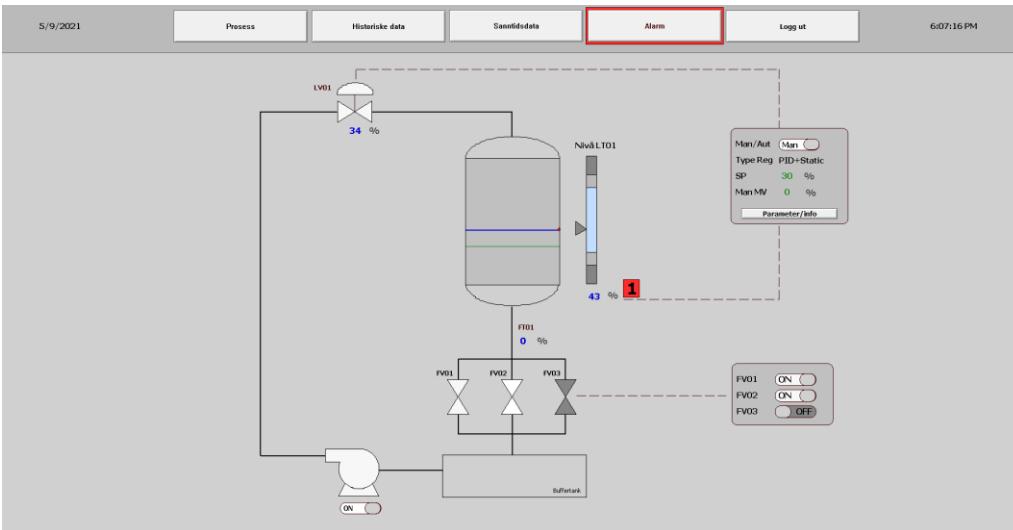
Alarmaksjoner		
Aksjon	Beskrivelse	Gjelder for alarm(er)
Alarmaksjon 1	Tømmer tanken ned til 50%. Slå av pumpe. Slå av pådragsorgan. Åpne utløpsventiler. Vent til prosessnivået når 50%. Steng utløpsventiler. Alarmslys aktiv ved alarm.	Alarm_HH
Alarmaksjon 2	Prosessen skal stå i ro. Slå av pumpe. Slå av pådragsorgan. Steng utløpsventiler. Alarmslys aktiv ved alarm.	Kommfeil: LT Kommfeil: Profi Kommfeil: Master Kommfeil: Slave
Alarmaksjon 3	Slå av foroverkobling. Ingen alarmslys.	Kommfeil: FT

Ved kritisk alarm vil knappen «Alarm» i menylinjen vil blinke kontinuerlig fram til alarmen er kvittert. Et eksempel på dette ses i figur 70 hvor nivået i tanken har passert grenseverdien for Høy/Høy alarm. I tillegg vil det dukke opp et varselsymbol ved komponenten det er feil på.



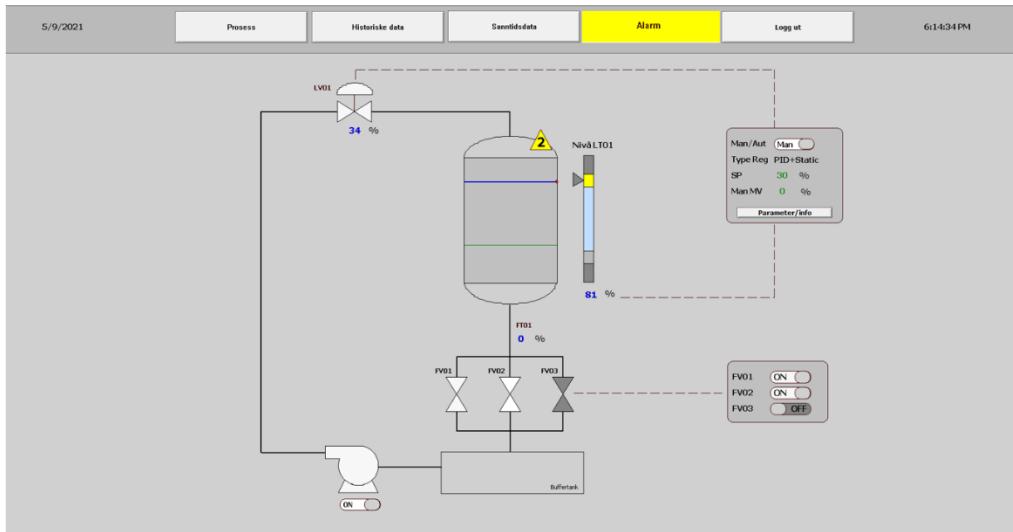
Figur 70: Illustrasjon av kritisk alarmsituasjon

Om alarmen blir kvittert og feilen fortsatt er vedvarende vil «Alarm»-knappen slutte å blinke, men bli stående med en rød ramme. Dette vises i figur 71.

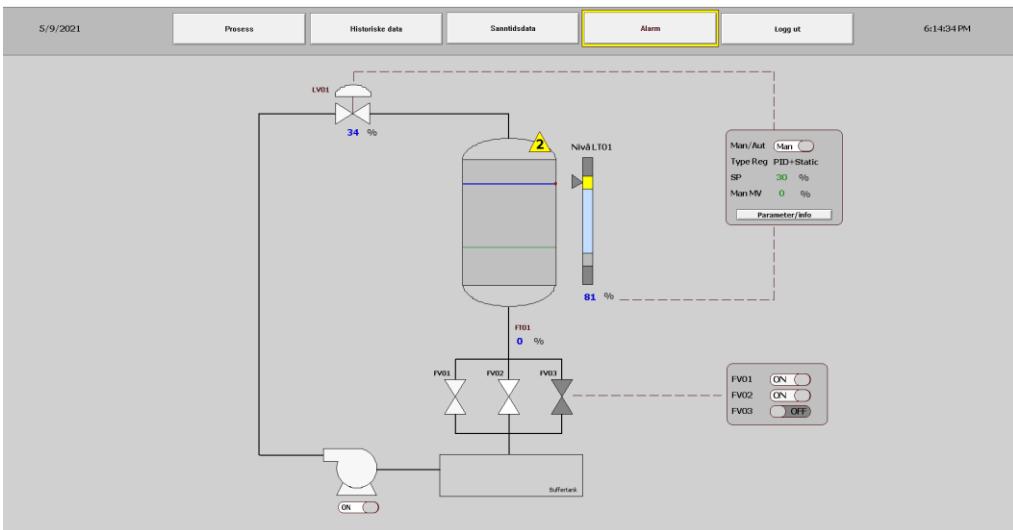


Figur 71: Illustrasjon av kvittert alarm ved kritisk alarmsituasjon

Ved ikke-kritisk alarm vil den samme knappen blinke gult fram til alarmen kvitteres, som vist i figur 72. Hvis feilen kvitteres og fortsatt er vedvarende vil det være en gul ramme, som vist i figur 73.

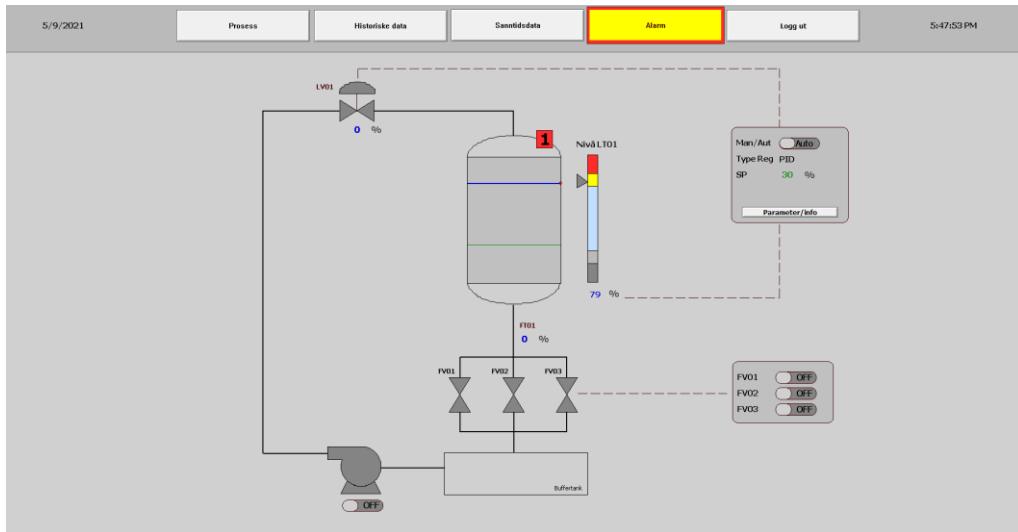


Figur 72: Illustrasjon av ikke-kritisk alarmsituasjon



Figur 73: Illustrasjon av kvittert alarm ved ikke-kritisk alarmsituasjon

I tilfeller hvor en kritisk alarm går over til å bli en ikke-kritisk alarm, eksempelvis om nivået i tanken har vært for høyt og gradvis synker, vil man se både rød ramme og blinkende gul knapp. Dette vises i figur 74.



Figur 74: Illustrasjon av kritisk og ikke-kritisk alarmsituasjon simultant

I oversikten over alarmer vil man tydelig kunne se hvilke alarmer som er kritiske, ikke-kritiske og kvitterte ved hjelp av de samme fargekodene som nevnt i tabell for ISA-101 tidligere i dette dokumentet. Figur 75 viser eksempel på stående og kvitterte alarmer, både kritisk og ikke-kritisk.

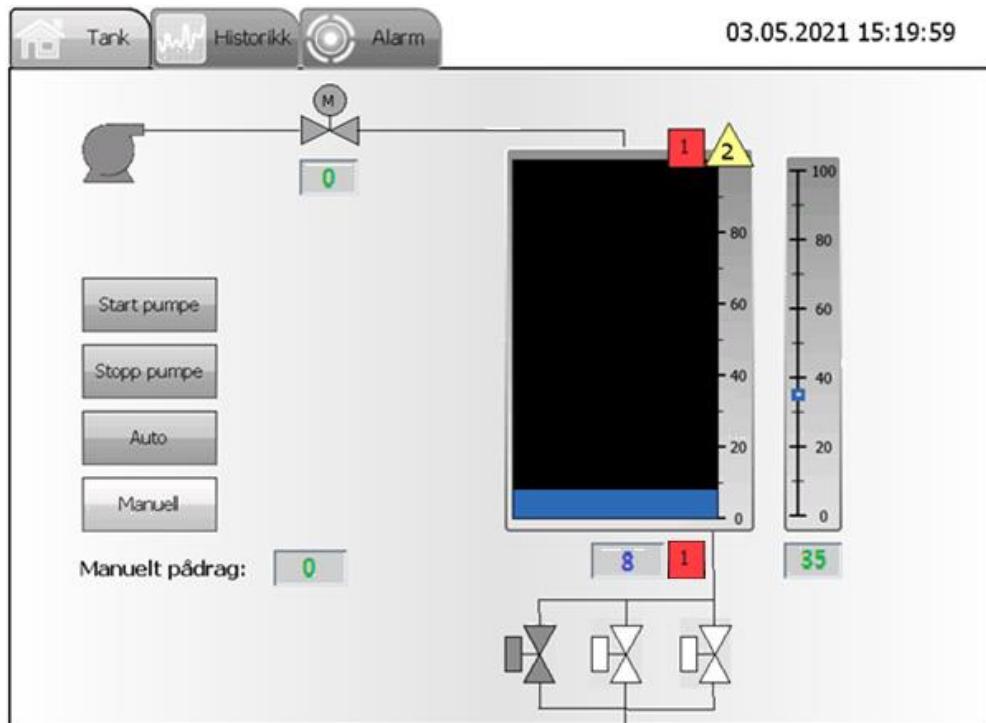
This screenshot shows the alarm history overview screen. At the top, it displays navigation tabs: Prosess, Historiske data, Sannitiddata, Alarm (highlighted in red), and Logg ut. The timestamp is 5/9/2021 5:41:27 PM. A large table below lists alarms from 09:00 to 17:18. The table columns include Date, Time, State, Class, Type, Priority, Name, Group, Provider, Value, and Limit. Several rows are highlighted in yellow, indicating active or critical alarms. To the right of the table are three buttons: 'Alarmhistorikk', 'Kvittere valgt alarm', and 'Kvittere alle alarmer'. At the bottom of the table, there are messages 'Update Successful' and 'Default Query'.

Date	Time	State	Class	Type	Priority	Name	Group	Provider	Value	Limit
09:00	11:40	UNACK	09C	09C	1	alarm_FT	System	Unisucht	OFF	ON
09:00	11:40	UNACK	09C	09C	1	alarm_FT	System	Unisucht	ON	ON
09:01	11:40	UNACK	09C	09C	1	alarm_FT	System	Unisucht	ON	ON
09:01	11:40	UNACK	09C	09C	1	alarm_FT	System	Unisucht	ON	ON
09:05	17:10	UNACK	09C	09C	1	alarm_FT	System	Unisucht	ON	ON
09:05	17:10	ACK	09C	09C	1	alarm_FT	System	Unisucht	ON	ON
09:05	17:10	ACK	09C	09C	1	alarm_FT	System	Unisucht	ON	ON
09:05	17:10	ACK	09C	09C	1	alarm_FT	System	Unisucht	ON	ON
09:05	17:10	ACK	09C	09C	1	alarm_FT	System	Unisucht	ON	ON

Figur 75: Alarmoversikt

## 5.4 HMI: Alarmer – IX Developer

Ved alarmsituasjoner vil et varselsymbol som viser alvorligetsgraden av situasjonen komme til synet på tank-fanen til panelet. HMI-en på tankriggen skiller mellom kritisk (rød) og ikke-kritisk (gul) alarm. Figur 76 viser en simulert alarmsituasjon hvor tanken snart er tom for å illustrere dette.



Figur 76: Eksempel på alarmvarsling

Figur 77 viser alarmvinduet etter en simulert alarmsituasjon. Alarmer markert i grønt er kvitterte alarmer. Den oransje alarmen er «Inactive», det vil si at årsaken til alarmen har forsvunnet, men den har enda ikke blitt kvittert. I dette tilfelle er det fordi nivået har gått ned til ikke-kritisk lav alarm og opp igjen til kritisk høy uten kvittering imellom.

The screenshot shows a control room interface with a top navigation bar containing three buttons: 'Tank' (with a house icon), 'Historikk' (with a waveform icon), and 'Alarm' (with a circular icon). The date and time '5/3/2021 12:14:21 PM' are displayed to the right of the navigation bar.

The main area displays a table of alarms:

State	Active Time	Text	Acknowledged
Active	5/3/2021 12:14:2	ALARM: Kritisk lavt nivå	
Inactive	5/3/2021 12:13:3	ALARM: Kritisk lavt nivå	
Acknowle	5/3/2021 12:13:1	Advarsel: Lavt nivå	5/3/2021 12:1
Acknowle	5/3/2021 12:13:1	ERROR: Høyt stasjonært avvik	5/3/2021 12:1
Acknowle	5/3/2021 10:32:3	Advarsel: Feil på samplingstid	5/3/2021 10:3

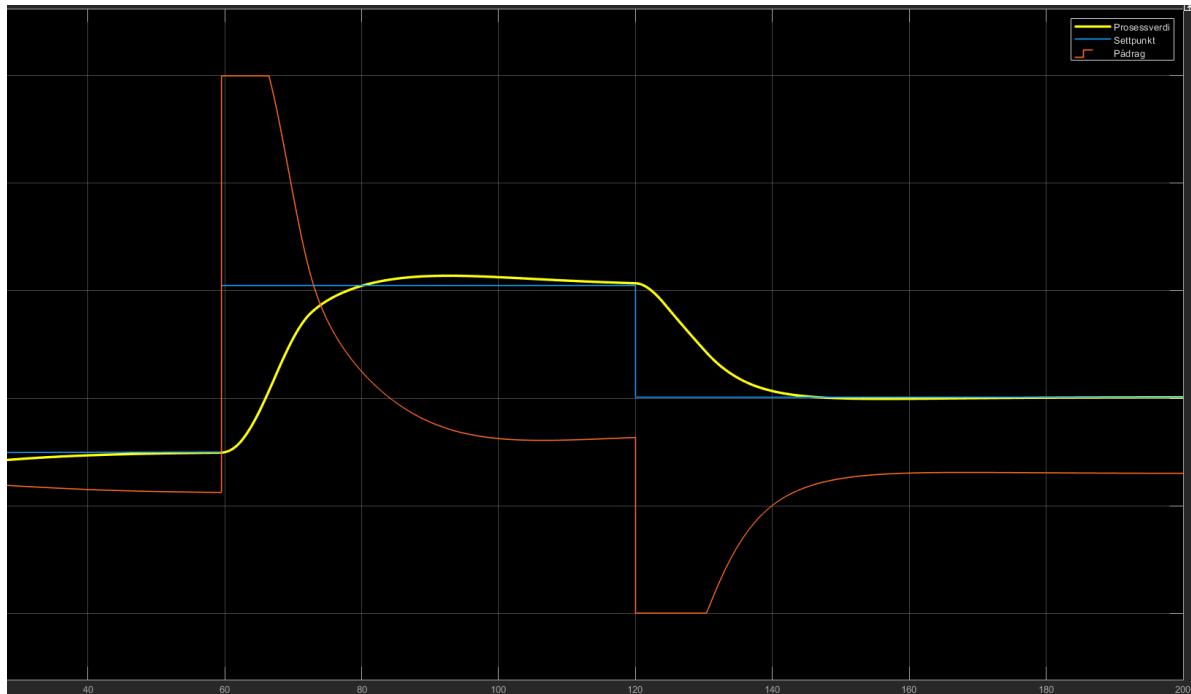
Below the table, a status message reads: 'Active: 1 Inactive: 1 Ack: 3 Normal: 0 [5 / 5]'. To the right of the table is a vertical toolbar with four buttons: 'Ack Selected', 'Ack All', 'Clear', and 'Filter'.

Figur 77: Simulert alarmsituasjon

## 5.5 Sprangresponser

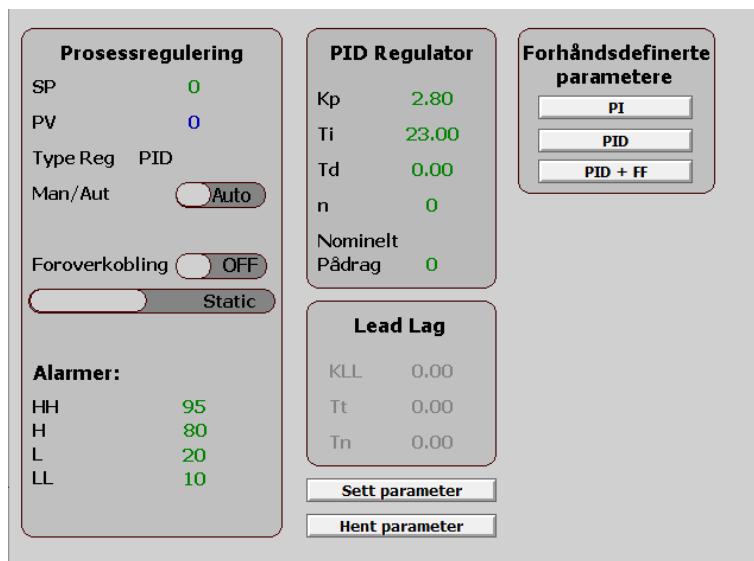
Under vises flere eksempler av sprangresponser, med bruk av ulike reguleringstyper. Det er eksempler med sprang i både settpunkt og forstyrrelser. De ulike sprangresponsene er regulert med parameterne som er vist i figuren i forkant.

### 5.5.1 Prosessmodell i Simulink

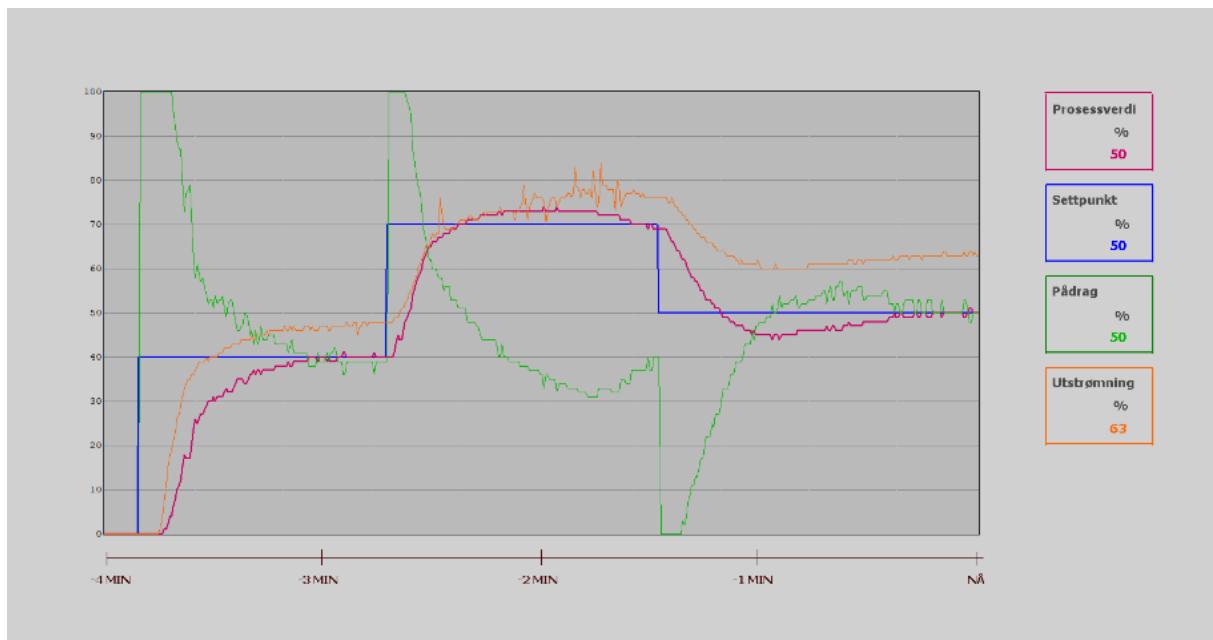


Figur 78: Eksempel på sprangrespons i Simulink; PI regulering

## 5.5.2 PI-regulering

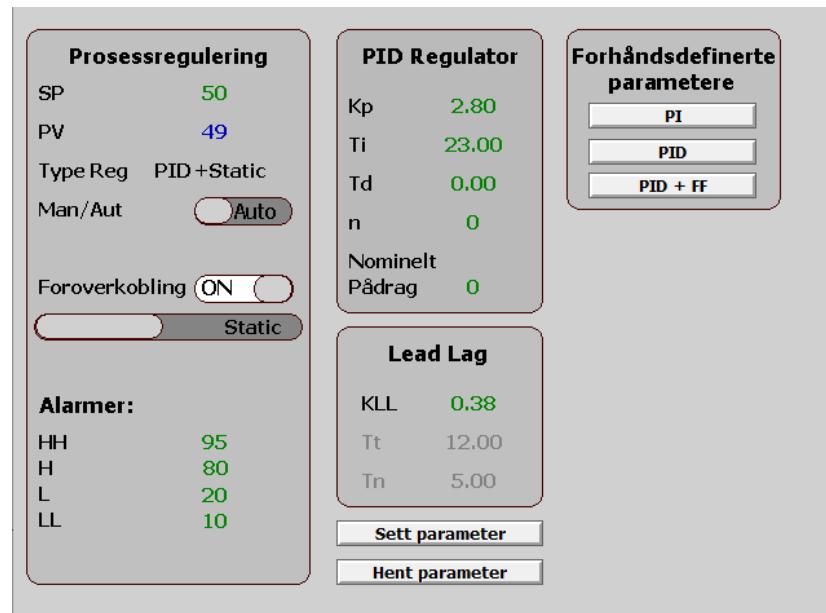


Figur 79: Paramterliste ved PI-regulering

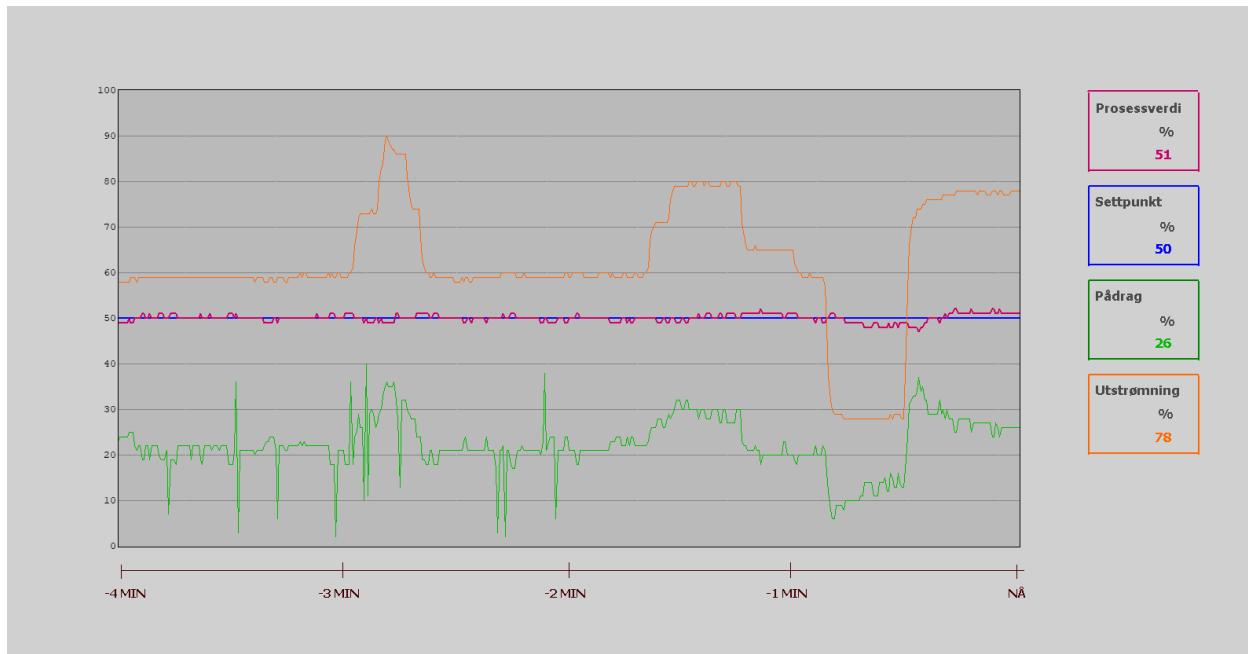


Figur 80: Sprangrespons med PI-regulering

### 5.5.3 PI + Statisk Foroverkobling

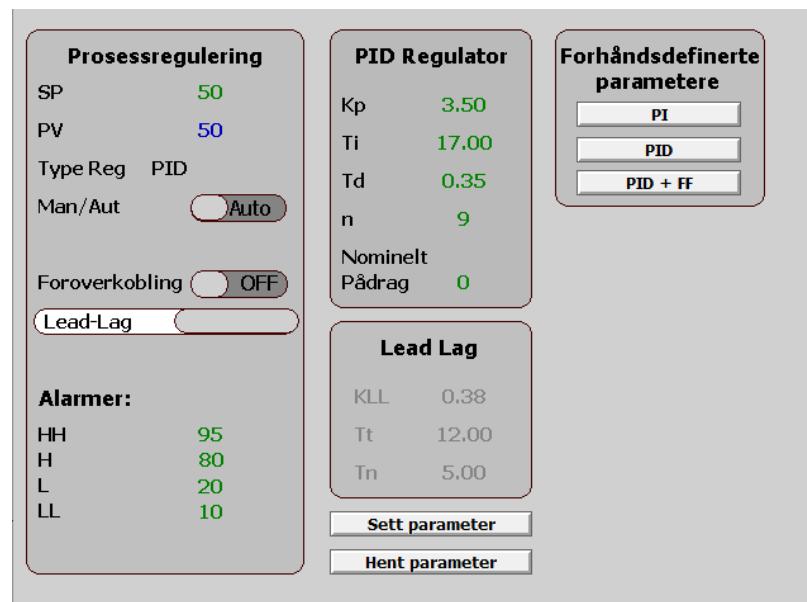


Figur 81: Parameterliste ved PI + Statisk regulering.

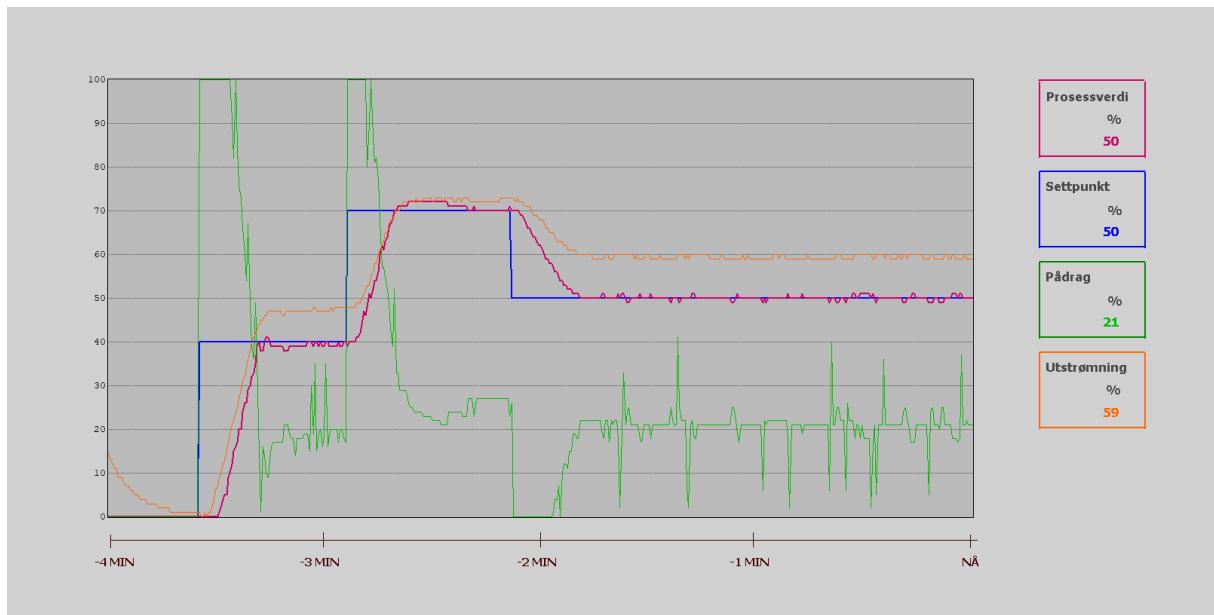


Figur 82: Sprangrespons ved PI (Første halvdel med PID) + Statisk foroverkobling. Sprang i forstyrrelse.

#### 5.5.4 PID-regulering

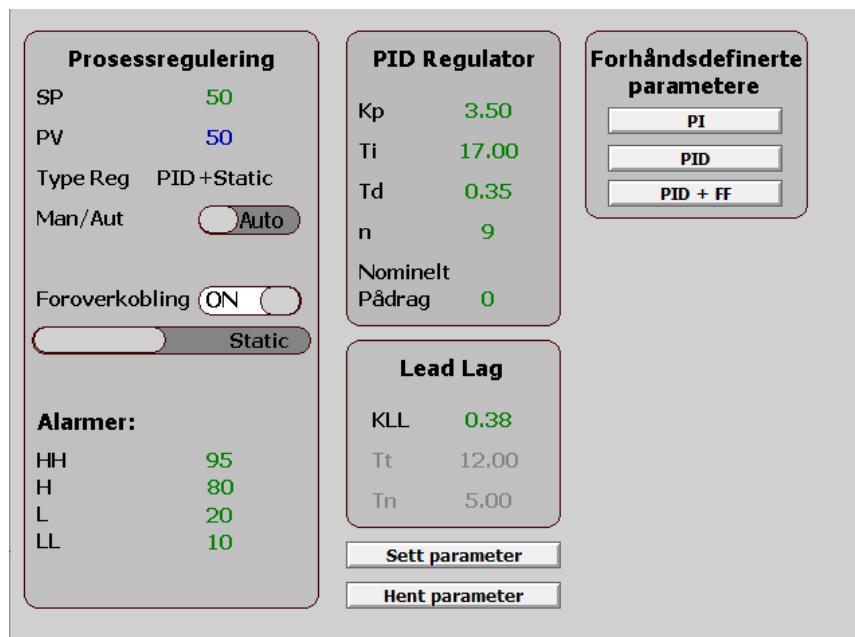


Figur 83: Paramterliste ved PID-regulering

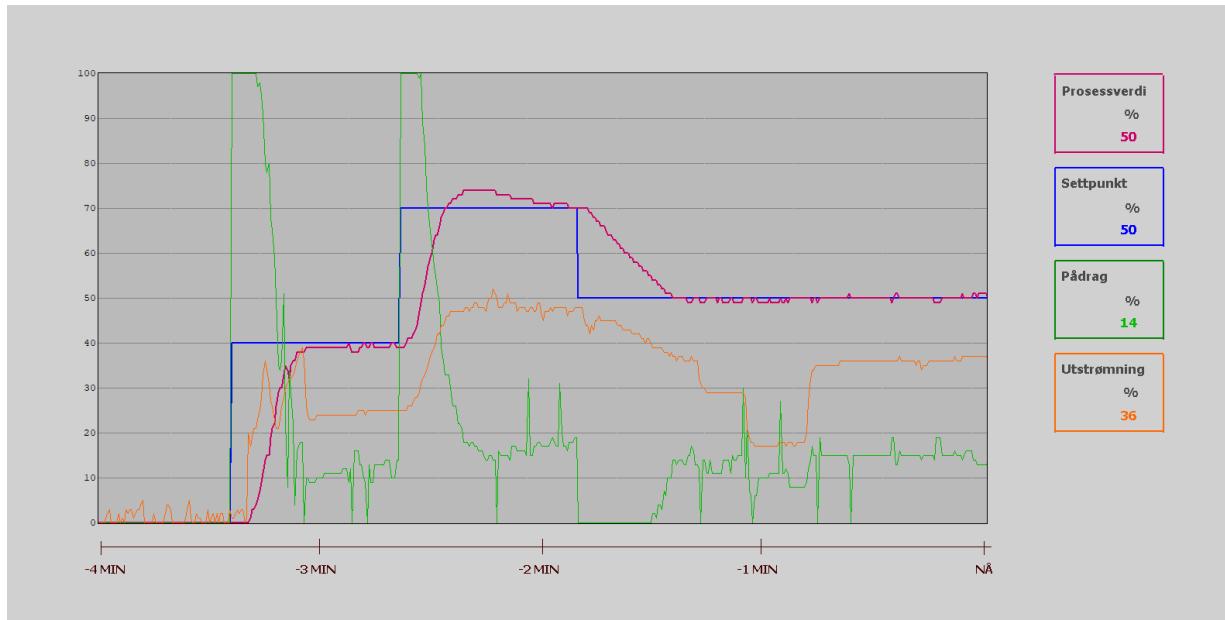


Figur 84: Sprangrespons ved PID-regulering

### 5.5.5 PID + Statisk Foroverkobling



Figur 85: Parameterliste ved PID med Statisk foroverkobling

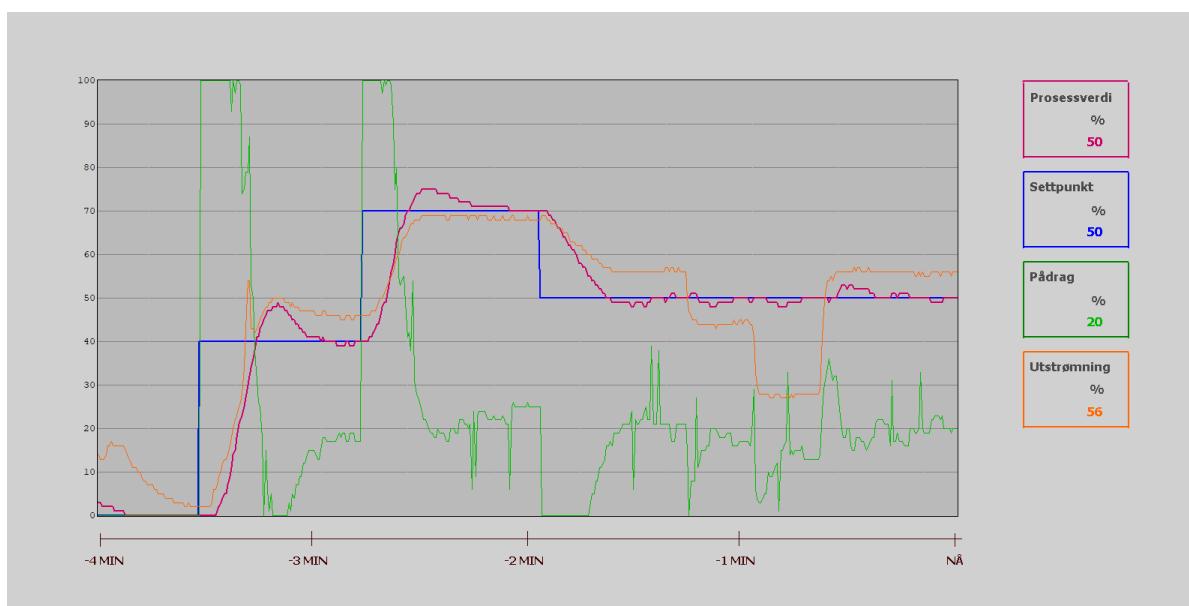


Figur 86: Sprangrespons (settpunkt først, så forstyrrelse) ved PID med Statisk foroverkobling

### 5.5.6 PID + Lead-Lag

<b>Prosessregulering</b> SP      50 PV      48 Type Reg    PID +Lead Lag Man/Aut <input checked="" type="button"/> Auto  Foroverkobling <input checked="" type="button"/> ON <input type="button"/> Lead-Lag	<b>PID Regulator</b> Kp      3.50 Ti      17.00 Td      0.35 n      9 Nominelt Pådrag      0	<b>Forhåndsdefinerte parametere</b> <input type="button"/> PI <input type="button"/> PID <input type="button"/> PID + FF
<b>Lead Lag</b> KLL      0.38 Tt      12.00 Tn      5.00 <input type="button"/> Sett parameter <input type="button"/> Hent parameter		

Figur 87: Parameterliste ved PID med Lead-Lag foroverkobling



Figur 88: Sprangrespons med PID og Lead-Lag Foroverkobling. I første halvdel: Sprang i settpunkt. Andre halvdel: Sprang i forstyrrelse.

## 6 Diskusjon og konklusjon

Det har blitt lagt vekt på brukervennlighet og mange automatiske handlinger som skal forhindre at feil oppstår. Vi har tatt utgangspunkt i at tankriggen kan være en del av et større anlegg og tatt avgjørelser basert på hvilke situasjoner man kommer ut for i arbeidslivet.

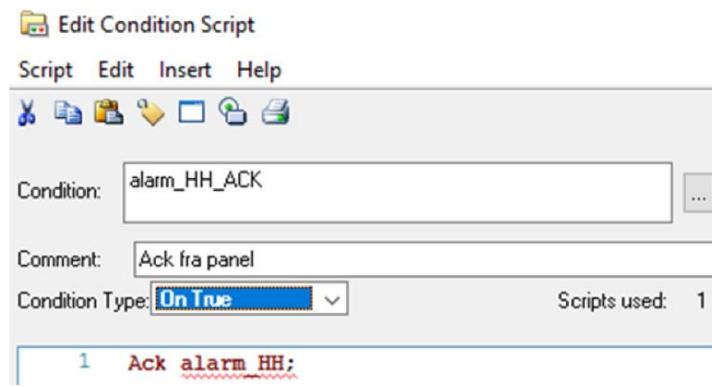
### 6.1 Kommunikasjon

Oppsettet av kommunikasjon for å sende bits mellom de forskjellige delene av anlegget gikk nokså feilfritt, og når kommunikasjonen var oppe var det bare å begynne og øke antallet verdier som overføres. Master PLS-en kan ikke operere med flyttall, og siden denne PLS-en fungerte som et bindeledd så ble det dermed logisk å ha mye utregninger og skalering på Slave PLS-en.

#### 6.1.1 Kvitteringsutfordringer

Underveis i prosjektet oppsto det problemer med å få brukergrensesnittene til å kvittere mellom hverandre. Løsningen vi landet på ble en ekstra kvitterings-bit til hver alarm som både settes når alarmen er kvittert og som ekstern kvittering for alarmen. Det ble det tatt utgangspunkt i hvordan kvitteringen fungerte internt i InTouch. Hvis det ikke er alarm eller alarmen er blitt kvittert vil kvitterings-bit til alarmen være høy. Hvis det er en ukvittert alarm vil den være lav. Denne verdien ble skrevet over til en egen bit som sendes mellom grensesnittene via master PLS. I IX-panelet ble det lagt til dynamikk for hver alarm som gjør at kvitterings-bit fungerer på samme måte her. Når en alarm oppstår settes tilhørende alarm-bit til 0, og når alarmen blir kvittert settes den tilbake til 1. I tillegg ble det lagt til ekstern kvittering på begge brukergrensesnittene. I IX-Developer er dette en egen funksjon i «Alarm Server» som heter «remote acknowledge», mens i InTouch ble det laget ett Condition Script for hver alarm, som vist i eksempelet i figur 89.

### 6.2 Sprangresponser



Figur 89: Condition Script Intouch, ekstern alarmkvittering

På flere av sprangresponsene vi fikk (se figurer under Resultater: Sprangresponser i dette dokumentet) kan man se at sprangresponsen fra settpunkt ikke blir helt like ved ulike nivåer i tanken, dette fordi forstyrrelsen/utstrømming fra tank øker ved høgre nivå i tank, og dermed vil innsvingingen bli noe annerledes.

### 6.2.1 Prosessmodell i Simulink

Med modellen som vi designet i Matlab (Simulink) testet vi prosessen med flere ulike parametersett. De parametersettene vi fant der, ble lagt til grunn når vi begynte å teste og finjustere på den virkelige prosessen. I resultatdelen er det lagt ved figur 78, en sprangrespons simulert i Matlab av prosessen, ved bruk av PI regulator, uten foroverkobling. Sammenligner man den responsen, med figur 80 (sprangrespons fra fysisk prosess ved PI regulering), ser man store likhetstrekk. Dette tyder på at modellen i Matlab stemmer greit overens med virkelig prosess.

Med små justeringer ble innsvingningsforløpene gode også der. Diskusjon rundt de ulike typene regulering av fysisk prosess beskrives under.

### 6.2.2 PI-regulering

Sprangresponsen vi fikk med PI-regulator var noe treg i starten. Den har lite til null dynamisk oversving og lukker stasjonært avvik relativt raskt. Dette er typisk oppførsel til en PI-regulator. Dette tyder på at integralleddet i regulatoren fungerer som den skal

### 6.2.3 PI + Statisk Foroverkobling

Med sprang i forstyrrelsen ved bruk av PI + statisk foroverkobling kan vi se at foroverkoblingen gjør jobben sin. Man kan nesten ikke se endring i prosessverdien, selv om forstyrrelsen endrer seg mye på kort tid. Dette tyder på at foroverkoblingen funker som den skal.

## 6.3 PID

Sprangresponsen vi fikk med PID-regulator var mye raskere sammenlignet med PI-regulatoren. Den har noe dynamisk oversving, selv om denne var overraskende liten. Dette er typisk oppførsel til en PID-regulator. Dette tyder på at derivasjonsleddet fungerer. Vi kan også se tydelig virkningen av derivatleddet ved å studere hvor hakkete pådraget er på plottet.

En utfordring til derivatvirkninger er at den forsterker støy. Når man ser hvor fin sprangresponsen ble kan man legge merke til at derivatvirkningen ikke har store problemer med støy. Dette går tilbake til at vi har et SMA («Simple Moving Average») filter på avlesning av prosessverdien.

### 6.3.1 PID + Statisk Foroverkobling

Denne sprangresponsen fikk noe mer dynamisk oversving ved sprang i settpunkt. Man kan minske det dynamiske oversvinget ved å redusere  $K_p$ . Med likhet med tidligere sørger integralleddet for at avviket lukkes over tid.

Det ble også gjort et sprang i utstrømning (forstyrrelse). Man ser ingen endring i prosessverdien når sprangen blir påført. Dette tyder på at foroverkoblingen fungerer, og takler sprangen svært godt.

### 6.3.2 PID + Lead-Lag

Det ble brukt samme parametere som *PID + Statisk* regulering. Her ble det ett større dynamisk oversving. Man ser også at prosessverdien er noe mer urolig når den ligger rundt settpunktet. Til ettertanke kan dette tyde på at  $K_p$  på PID-regulatoren er litt for høy eller at tidskonstantene på Lead-Lag elementet bør fininnstilles.

Det ble også gjort et sprang i utstrømning (forstyrrelse). Man ser en liten endring i prosessverdi. Igjen kan dette skyldes litt for aggressiv PID-regulator eller at tidskonstantene på Lead-Lag elementet bør fininnstilles.

## 6.4 Regulatorer

PID-regulatoren ble kontinuerlig oppdatert over tid da det stadig dukket opp større og mindre problemer.

Det ble gjort en del endringer av foroverkoblingsfilteret og PID-regulatoren i hovedprosjektet.

Et problem vi fikk var at integralleddet i PID-regulatoren summerte seg for høyt opp ved bruk av foroverkobling sammen med PID-regulator. Dette til tross for at PID-regulatoren er utstyrt med anti-windup. Det viste seg at hvis anti-windup skal fungere optimalt må pådraget fra foroverkoblingen gå inn på PID-regulatoren slik at anti-windup kunne ta den med i betraktningen. Dette løste samtidig problemet med rykkfri overgang når man slår inn og ut foroverkobling.

I tillegg viste det seg at det var en del elementer som ikke var lagt inn i PID-regulatoren og foroverkoblingen. Når arbeidsnotat 2 ble skrevet visste vi ikke at foroverkoblingen skulle være utstyrt med statisk foroverkobling. Vi antok i stedet at det bare skulle være Lead-Lag. Dette ble lagt inn i ettertid. Det manglet også innganger for nominelt pådrag og mulighet for å kjøre PID i revers. Dette ble også lagt inn i ettertid.

## 6.5 Rykkfri overgang

Som beskrevet tidligere ble «Tracking» implementert på vanlig måte, ved å oppdatere integraltiden i PID-regulatoren. Når det kommer til hvordan rykkfri overgang er implementert på utsiden av PID-regulatoren, er programmet vårt noe annerledes enn hva som er normalt.

Normalt sett vil man parallellkoble et gitt antall PID-regulatorer. Vi hadde først tenkt å gjøre det slik. Men det viste seg etter hvert at det ble en god del forskjellige regulatorprogrammer, noe som har innvirkning på samplingstiden og størrelsen til programmet. Det ble derfor utviklet en egen blokk som mellomlagret pådraget under skifte, mens PID-regulatoren oppdaterer alle verdier. Denne løsningen så ut til å fungere tilfredsstillende.

### 6.5.1 Valg av samplingstid

For at den diskrete regulatoren skal ligne mest mulig på et kontinuerlig system ønsker man minst mulig samplingstid. Mindre samplingstid fører til en bedre representasjon av signalene.

Regulatoren tar samplingstiden som input. Det kan være fristende å la samplingstiden gå fritt, altså så raskt som mulig, og dermed bruke en variabel som input på regulatoren. Problemet med dette er at når man tuner regulatoren, har man tunet den for den spesielle samplingstiden. Selv om samplingtiden blir brukt i likningene inne i regulatoren vil oppførselen til I- og D-leddet variere ut fra verdien på samplingstiden. Derfor ønsker man å stille inn en fast samplingtid.

For å løse dette problemet ble det utviklet en funksjonsblokk: «Samplimetatracker», som oppdaterer gjennomsnittlig samplingtid over et satt intervall. For å finne ut hvor lavt vi kunne sette samplingstiden, ble programmet gående fritt, uten å låse samplingstiden. Programmet brukte  $\approx 25$  ms på en syklus. Vi tok utgangspunkt i at samplingstiden kan endre seg over tid. Enten pga. hardware, eller ved senere utvidelser av programmet. Det ble konkludert med at 50 ms var mer enn godt nok for denne prosessen. Dermed ble samplingstiden låst fast til 50 ms på GX-Works. Samplingstiden ble satt som en konstant inn på PID-blokken.

## 6.6 Brukergrensesnitt

Ved utarbeiding av HMI har vi lagt stor vekt på strukturert og enkel navigering, minimalistisk design av komponenter, og lite bruk av farger for å gjøre brukergrensesnittet lettere å lese og framheve de få stedene farger blir brukt. Dette fordi det skal være intuitivt for nye brukere, samt kreve lite opplæring. I tillegg vil et så enkelt brukergrensesnitt som mulig bidra til høyere sikkerhet ved anlegget og lavere risiko for feil.

I utviklingen av HMI-en for panelet ved tankriggen ble det lagt vekt på effektivisering og brukervennlighet. HMI-en har derfor ikke noen form for innlogging eller tilgangsnivå. Den er også relativt begrenset sammenlignet med HMI for PC. Årsaken til dette er at det ikke ble ansett som nødvendig med like mye informasjon og endringsmuligheter når man står i anlegget som når man er på et eventuelt kontrollrom. Det ble også konkludert med at en eventuell innlogging ville være ineffektivt for operatører som står ute ved riggen. Skal man gjøre avanserte endringer i regulering og styring kan man eventuelt kontakte kontrollrom hvis man eksempelvis tenker seg at dette er en bit av ett større anlegg.

Panelet har omtrent samme lese- og skrivemuligheter som operatør 2 har ved innlogging på HMI-en på PC, med unntak av mulighet for manuellstyring og åpning av ventiler. Tanken bak dette er at disse funksjonene kan være nødvendige å ha tilgang til for noen som står ute i anlegget ved prosessen og må endre noe raskt ut ifra forholdene, samtidig som det ikke vil være nødvendig for operatør 1 å gjøre disse endringene fra avstand.

## 7 Referanser

- Anstenrud, T. (2021, Januar 21). Forelesning 3 Z-transformasjon. Trondheim, Trøndelag, Norge.
- Arnesen, H. M., Sæther Ulvatne, M., & Engeseth Andersen, R. (2021, Februar). Guide til kommunikasjonsoppsett (Vedlegg 11). Trondheim, Trøndelag, Norge.
- Beijer Electronics . (2017, April). BE IX Developer users guide. Malmø, Sverige.
- Beijer Electronics. (2004, Januar). Melsec Q Kom i gang. Drammen, Viken, Norge.
- Beijer Electronics AB. (2003, Februar). FXON-32NT-DP Kom i gang. Malmø, Sverige.
- Bonde, Håvard, H. K., Sæther Grasbakken, A., Bonde, M., Berdahl, H. W., & Grøterud, M. (2021). *Arbeidsnotat Forprosjekt (Vedlegg 1)*. Trondheim.
- Dessen, F. (2021, Januar 18). Dataøving\_1 (Vedlegg 9). Norge.
- Dessen, F. (2021, Februar 3). Forelesningsnotater Foroverkobling. Trondheim , Trøndelag, Norge.
- Halse, K., Sæther Grasbakken, A., Bonde, H., Bonde, M., Berdahl, H. W., & Grøterud, M. (2021). *Arbeidsnotat PID og Lead-Lag funksjonsblokk (Vedlegg 2)*. Trondheim.
- Hveem, P., & Bjørvik, K. (2014). *Reguleringsteknikk*. Trondheim: Kybernetes forlag.
- Inductive Automation. (2018, September 12). *Webområde for Inductive Automation*. Hentet fra Webområde for Inductive Automation:  
[https://www.inductiveautomation.com/resources/article/what-is-scada#:~:text=Supervisory%20control%20and%20data%20acquisition%20\(SCADA\)%20is%20a%20system%20of,locally%20or%20at%20remote%20locations](https://www.inductiveautomation.com/resources/article/what-is-scada#:~:text=Supervisory%20control%20and%20data%20acquisition%20(SCADA)%20is%20a%20system%20of,locally%20or%20at%20remote%20locations)
- Maxim Integrated Products, Inc. (2001, 08 10). *Maxim Integrated*. Hentet fra Using The Secure Microcontroller Watchdog Timer: <https://www.maximintegrated.com/en/design/technical-documents/app-notes/1/101.html>
- Mitsubishi Electric Corporation. (2012, Mars). FX3U Users manual. Tokyo, Japan.
- Mitsubishi Electric Corporation. (2002, November). Melsec Q Data book. Tokyo, Japan.
- Mitsubishi Electric Corporation. (2015, April). FXON-32NT-DP Users manual. Tokyo, Japan.
- Mitsubishi Electric Corporation. (2015, April). FXON-3A Users manual. Tokyo, Japan.
- Mitsubishi Electric Corporation. (2016, Juli). FX3U Programming manual. Tokyo, Japan.
- Mitsubishi Electric Corporation. (2017, November). FX3U Communication users manual. Tokyo, Japan.
- Mitsubishi Electric Corporation. (2018, Mars). FX3U Hardware manual. Tokyo, Japan.
- Mitsubishi Electric Corporation. (2018, April). GX Configurator DP software manual. Tokyo, Japan.
- Mitsubishi Electric Europe BV. (2007, August). Melsec Q Beginners manual. Ratingen, Tyskland.
- Mitsubishi Electric Europe BV. (2007, Mai 1). QJ71PB92D Users manual. Ratingen, Tyskland.
- PTC Inc. (2007, Juni). Kepware client connectivity guide. Boston, Massachusetts, USA.

- PTC Inc. (2018). Kepware InTouch client driver. Boston, Massachusetts, USA.
- PTC Inc. (2019). Kepware Configuring Mitsubishi FX3U for KEPServerEX. Boston, Massachusetts, USA.
- PTC Inc. (2019). Kepware kepserverex manual. Boston, Massachusetts, USA.
- Spesifikasjoner for SCADA HMI. (2021). Trondheim, Trøndelag, Norge.
- Strøm, K. (2021). 1.1 Tegninger og koblingsskjema. Trondheim, Trøndelag, Norge.
- Taylor, D. (2017, 06 23). *Erdos Miller: Engineering solutions*. Hentet fra Blog: PID Anti-windup Techniques: <https://info.erdosmiller.com/blog/pid-anti-windup-techniques>
- Utstyrts. og programvareoversikt. (2021). Trondheim, Trøndelag, Norge.
- Åström, K. J., & Hägglund, T. (2006). *Advanced PIC Control*.

**<https://www.ntnu.no/viko/harvard-eksempler>**

## 8 Vedlegg: Tagname-lister

8.1 Tabell 10: Oversikt over innholdet i KEPServer-filen

TAGNAME	ADRESSE
PROSESSVERDI	D0
PADRAG	D1
FLOW_TRANSMITTER	D2
WATCHDOG_SLAVE_HMI	D3
MANUELT_PADRAG	D20
SETTPUNKT	D21
K_P	D22
T_I	D23
T_D	D24
N_FILTER	D25
K_LL	D26
T_T	D27
T_N	D28
NOMINELT_PADRAG	D29
LIMIT_HH	D100
LIMIT_H	D101
LIMIT_L	D102
LIMIT_LL	D103
WATCHDOG_MASTER_HMI	D107
VENTIL_1	L100
VENTIL_2	L101
VENTIL_3	L102
SW_REG_PARAM	L104
FF_ON	L106
LEAD_LAG_ON	L107

Tagname	Adresse
pumpe	M103
auto_manuell	M105
error_FT	M200
error_LT	M201
error_PROF1	M202
alarm_HH	M205
alarm_H	M206
alarm_I	M207
alarm_LL	M208
error_E	M209
error_POWER	M210
alarm_case_1	M211
alarm_case_2	M212
alarm_case_3	M213
error_FT_ACK	M300
error_LT_ACK	M301
alarm_HH_ACK	M302
alarm_H_ACK	M303
alarm_L_ACK	M304
alarm_LL_ACK	M305
error_E_ACK	M306
error_PROF1_ACK	M307
error_POWER_ACK	M310
warning_samplingtime_ACK	M311
warning_samplingtime	M400

9 Tabell 11: Oversikt over tags for dataoverføring i Intouch

Tagname	Access Rights	Type	Kommentar
alarm_case_2	KEPserverX	I/O Discrete	Alarmreaksjon 2
alarm_case_3	KEPserverX	I/O Discrete	Alarmreaksjon 3
alarm_H	KEPserverX	I/O Discrete	Alarm H
alarm_H_ACK	KEPserverX	I/O Discrete	Alarm H Ack
alarm_HH	KEPserverX	I/O Discrete	Alarm Høy
alarm_HH_ACK	KEPserverX	I/O Discrete	Alarm HH Ack
alarm_L	KEPserverX	I/O Discrete	Alarm L
alarm_L_ACK	KEPserverX	I/O Discrete	Alarm L Ack
alarm_LL	KEPserverX	I/O Discrete	Alarm LL
alarm_LL_ACK	KEPserverX	I/O Discrete	Alarm LL Ack
auto_manuell	KEPserverX	I/O Discrete	Auto/manuell styring

error_com_master_ACK	KEPserverX	I/O Discrete	Kom.feil master Ack
error_com_slave_ACK	KEPserverX	I/O Discrete	Kom.feil slave Ack
error_E	KEPserverX	I/O Discrete	Høyt stasjonært avvik
error_E_ACK	KEPserverX	I/O Discrete	Kvittering høyt stasjonært avvik
error_ETH	KEPserverX	I/O Discrete	Kom.feil Ethernet
error_ETH_ACK	KEPserverX	I/O Discrete	Kom.feil Ethernet Ack
error_KEP	KEPserverX	I/O Discrete	Kom.feil KepServer
error_KEP_ACK	KEPserverX	I/O Discrete	Kom.feil KepServer Ack
error_LT	KEPserverX	I/O Discrete	Kom.feil level transmitter
error_LT_ACK	KEPserverX	I/O Discrete	Kom.feil level transmitter Ack
error_POWER	KEPserverX	I/O Discrete	Alarm etter strømbrudd
error_POWER_ACK	KEPserverX	I/O Discrete	Alarm etter strømbrudd Ack
error_PROFIBUS	KEPserverX	I/O Discrete	Kom.feil profibus
error_PROFIBUS_ACK	KEPserverX	I/O Discrete	Kom.feil profibus Ack
FF_on	KEPserverX	I/O Discrete	Foroverkobling av/på
lead_lag_on	KEPserverX	I/O Discrete	Lead Lag av/på
lead_lag_static	KEPserverX	I/O Discrete	Statisk Led Lag av/på
pumpe	KEPserverX	I/O Discrete	Pumpe av/på
sw_reg_param	KEPserverX	I/O Discrete	Sender regulator parameter
ventil_1	KEPserverX	I/O Discrete	Ventil 01: forstyrrelse
ventil_2	KEPserverX	I/O Discrete	Ventil 02: forstyrrelse
ventil_3	KEPserverX	I/O Discrete	Ventil 03: forstyrrelse
warning_samplingtime	KEPserverX	I/O Discrete	Varsel høy samplingstid
warning_samplingtime_ACK	KEPserverX	I/O Discrete	Varsel høy samplingstid Ack
flow_transmitter	KEPserverX	I/O Integer	Flowtransmitter utløp
k_ll	KEPserverX	I/O Integer	KLL * 100
k_p	KEPserverX	I/O Integer	Kp * 100
limit_H	KEPserverX	I/O Integer	Grense H
limit_HH	KEPserverX	I/O Integer	Grense HH
limit_L	KEPserverX	I/O Integer	Grense L
limit_LL	KEPserverX	I/O Integer	Grense LL
manuelt_padrag	KEPserverX	I/O Integer	Manuelt pådrag LV
n_filter	KEPserverX	I/O Integer	Filterkonstant, heltall
nominelt_padrag	KEPserverX	I/O Integer	Nominelt pådrag til PID
padrag	KEPserverX	I/O Integer	Pådrag fra regulator
prosessverdi	KEPserverX	I/O Integer	Prosessverdi. Nivå fra LT
Tagname	Access Rights	Type	Kommentar
sett punkt	KEPserverX	I/O Integer	Settpunkt
t_d	KEPserverX	I/O Integer	Td * 100
t_i	KEPserverX	I/O Integer	Ti * 100
t_n	KEPserverX	I/O Integer	TN * 100
t_t	KEPserverX	I/O Integer	TT * 100
watchdog_master_hmi	KEPserverX	I/O Integer	Watchdog master komunikasjon
watchdog_slave_hmi	KEPserverX	I/O Integer	Watchdog slave komunikasjon

9.1 Tabell 12: Oversikt over interne tags i InTouch

Interne tags			
Tagname	Access Rights	Type	Kommentar
blink_prio_1	-	Memory Discrete	Alarm blink, prioritet 1
blink_prio_2	-	Memory Discrete	Alarm blink, prioritet 2
fill_prio_1	-	Memory Discrete	Alarm indikasjon prioritet 1
fill_prio_2	-	Memory Discrete	Alarm indikasjon prioritet 2
sw_ferdige_PI	-	Memory Discrete	Hent ferdigsatte parameter PI
sw_ferdige_PID	-	Memory Discrete	Hent ferdigsatte parameter PID
sw_ferdige_PID_FF	-	Memory Discrete	Hent ferdigsatte parameter PID + foroverkobling
sw_get_reg_param	-	Memory Discrete	Hent parameter fra master PLS
n_lokal	-	Memory Interger	Filterkonstant lagret lokalt
Td_lokal	-	Memory Interger	Lokal Td før skalering
Ti_lokal	-	Memory Interger	Lokal Ti før skalering
Tn_lokal	-	Memory Interger	Lokal Tn før skalering
Tt_lokal	-	Memory Interger	Lokal Tt før skalering
KII_lokal	-	Memory Interger	Lokal KII før skalering
Kp_lokal	-	Memory Interger	Lokal Kp før skalering
Local_watchdog_master	-	Memory Interger	
Local_watchdog_slave	-	Memory Interger	
prev_watchdog_master_hmi	-	Memory Interger	
prev_watchdog_slave_hmi	-	Memory Interger	

9.2 Tabell 13: Oversikt over tags i IX Developer

Tagname	Access Rights	Type	Adresse
prosessverdi	Read	INT16	D0
padrag	Read	INT16	D1
manuelt_padrag	ReadWrite	INT16	D20
settpunkt	ReadWrite	INT16	D21
ventil_1	ReadWrite	BIT	M100
ventil_2	ReadWrite	BIT	M101
ventil_3	ReadWrite	BIT	M102
pumpe	ReadWrite	BIT	M103
auto_manuell	ReadWrite	BIT	M105
error_FT	Read	BIT	M200
error_LT	Read	BIT	M201

error_PROF1	Read	BIT	M202
alarm_HH	Read	BIT	M205
alarm_H	Read	BIT	M206
alarm_L	Read	BIT	M207
alarm_LL	Read	BIT	M208
error_e	Read	BIT	M209
error_POWER	Read	BIT	M210
connection_error_intouch	Read	BIT	M215
warning_samplingtime	Read	BIT	M400
error_FT_ACK	ReadWrite	BIT	M300
error_LT_ACK	ReadWrite	BIT	M301
alarm_HH_ACK	ReadWrite	BIT	M302
alarm_H_ACK	ReadWrite	BIT	M303
alarm_L_ACK	ReadWrite	BIT	M304
alarm_LL_ACK	ReadWrite	BIT	M305
error_E_ACK	ReadWrite	BIT	M306
error_PROF1_ACK	ReadWrite	BIT	M307
error_POWER_ACK	ReadWrite	BIT	M310
warning_samplingtime_ACK	ReadWrite	BIT	M311

## 10 Vedlegg: PID-Kode

```

(*
Had to rename variables in GX-works in order to avoid errors.

Inputs:
  SP      - reference / Setpoint
  PV      - Process Value
  Kp      - Propotional Gain
  Ti      - Integration time
  Td      - Derivation time
  n_filter - Adjusting the filter constant tau = Td / n
  Tracking - Tracking input.

Outputs:
  MV      - Modulated Value

Equations:
  MV      = P_gain + I_gain + D_gain
  P_gain = Kp * e[k]
  I_gain = KpT/(Ti*2)*(e[k] + e[k-1])+i[k-1]
  D_gain = 2KpTd/(2)*(y[k]-y[k-1]) + (2tau - T)/(2tau + T) * d[k-1]

Tracking:
  I_gain on the controller can be deactivated by setting Ti = 0.0.
  In order for Tracking to work - Ti needs to be higher than 0.0.
*)

IF ENABLE = 0 THEN (* When enable has been 0 -> Start with fresh values. *)
  MV := 0;
  I_gain := 0.0;
  I_gain_prev := 0.0;
  Dev_prev := Dev;
  PV_prev := PV_float;
  D_gain_prev := D_gain;
  tracking_flag:= 0;
ELSE
  IF Kp = 0.0 THEN (* PID-controller is Deactivated -> Write LeadLag directly to output *)
    MV := FeedForward;
  ELSE
    (* Saturate input *)
    Aa := SP;
    FLT(1, PV, PV_float);
    IF Aa < MIN_INPUT THEN
      Aa := MIN_INPUT;
    ELSIF Aa > MAX_INPUT THEN
      Aa := MAX_INPUT;
    END_IF;

    IF PV_float < MIN_INPUT THEN
      PV_float := MIN_INPUT;
    ELSIF PV_float > MAX_INPUT THEN
      PV_float := MAX_INPUT;
    END_IF;

    (* Calculate deviation (avvik) *)
    Dev := Aa - PV_float; (*SP - PV *)
  END_IF;
END_IF;

```

(\* Calculate D \*)

```

IF Td = 0.0 THEN
    D_gain := 0.0;
ELSE
    Aa:= 2.0*Kp*Td / (2.0*(Td/n_filter)+ SampleTime);
    Bb:= (2.0*(Td/n_filter)- SampleTime) / (2.0*(Td/n_filter)+ SampleTime);
    D_gain := Aa * (PV_float - PV_prev) + Bb * D_gain_prev;
END_IF;

(* Calculate P *)
P_gain := Dev * Kp;

(* Calculate I *)
IF Ti = 0.0 THEN (* I_gain can be deactivated by setting Ti to 0. *)
    I_gain := 0.0;
ELSE
    IF MV = Tracking THEN (* Note that this MV is from the last cycle *)
        Aa := Kp * SampleTime / (2.0 * Ti);
        I_gain := Aa * (Dev + Dev_prev) + I_gain_prev;
    ELSE
        (*This controller is not in control. Activate Tracking *)
        tracking_flag := 1;
        FLT(1, Tracking - FeedForward, Aa);
        I_gain := Aa - P_gain - D_gain ;
    END_IF;
END_IF;

(* Write to output*)
INT(1, (P_gain + I_gain + D_gain), PID_Padrag);
IF NOT Inverse THEN
    MV := PID_Padrag+FeedForward+u_0; (* Normal output *)
ELSE
    MV := -1*(PID_padrag+FeedForward+u_0) + MAX_OUTPUT; (* Inverse output *)
END_IF;
END_IF;

(* Saturate output and check for windup *)
IF MV <= MIN_OUTPUT THEN          (* Negative Saturation *)
    IF Dev < 0.0 THEN             (* Antiwindup *)
        I_gain := I_gain_prev;
    END_IF;
    MV := MIN_OUTPUT;
ELSIF MV >= MAX_OUTPUT THEN      (* Positive Saturation *)
    IF Dev > 0.0 THEN             (* Antiwindup *)
        I_gain := I_gain_prev;
    END_IF;
    MV := MAX_OUTPUT;
END_IF;

(* Update previous values *)
Dev_prev       := Dev;
PV_prev        := PV_float;
I_gain_prev    := I_gain;
D_gain_prev    := D_gain;
tracking_flag  := 0;

END_IF;

```

## **11   Oversikt over eksterne vedlegg**

Vedlegg 1 – Arbeidsnotat Forprosjekt

Vedlegg 2 – Arbeidsnotat PID og Lead-Lag funksjonsblokk

Vedlegg 3 – Slave\_PLS (GX Works)

Vedlegg 4 – Master\_PLS (GX Works)

Vedlegg 5 – HMI\_IX\_Developer

Vedlegg 6 – HMI\_InTouch

Vedlegg 7 - KepServerEX

Vedlegg 8 – ProsessModell\_Matlab\_Simulink

Vedlegg 9 – Dataøving 1, Reguleringsteknikk 2

Vedlegg 10 – Forelesning Z-transformasjon

Vedlegg 11 – Guide til kommunikasjonsoppsett

Vedlegg 12 – Kommunikasjonsoversikt