Telemetrienkoder til bruk i vitenskapelige sonderaketter – dokumentasjon og bruksanvisning

Vidar Killingmo, Alvin Brattli og Terje Angeltveit

Forsvarets forskningsinstitutt/Norwegian Defence Research Establishment (FFI)

25. mai 2009

FFI-rapport 2009/00987

105602

P: ISBN 978-82-464-1598-7
E: ISBN 978-82-464-1599-4

Emneord

Telemetrienkoder

Sonderaketter

Godkjent av

Ulf-Peter Hoppe	Prosjektleder
Stian Løvold	Forskningssjef
Johnny Bardal	Avdelingssjef

Sammendrag

I perioden 2005 til 2007 ble det designet og produsert en ny telemetrienkoder på FFI til bruk i vitenskapelige sonderaketter. Denne rapporten er skrevet som en bruksanvisning og brukerveiledning til telemetrienkoderen (elektriske interface, oppkobling og programmering/konfigurering), og er beregnet på elektronikkingeniører og romingeniører med en viss forkunnskap innen telemetri. Rapporten består av to hoveddeler – en del som stort sett tar for seg det elektriske grensesnittet til enkoderen, og en del som hovedsaklig tar for seg programmering og konfigurering.

English summary

This report is a user guide for the telemetry encoder system for sounding rockets developed at FFI between 2005 and 2007, and describes its electronic interface, assembly, and programming/configuration. The intended audience are engineers with a background in electronics and/or aerospace engineering and with some background in telemetry systems. This report is in two parts – the first part deals with the electronic interface to the encoder, and the second deals with programming and configuration.

Innhold

1	**Innledning**	**7**
2	**STAPPE - Stacked Timer Amon Power Pusek Extendable.**	**7**
2.1.1	Montering og demontering av moduler.	8
2.1.2	Programmering/konfigurering	9
2.2	TIM (Timer)	10
2.2.1	Utganger til pyroteknikk.	10
2.2.2	28V utganger	11
2.2.3	Skru av nyttelasten	11
2.2.4	Analoge monitorer	12
2.2.5	Digitale monitorer	13
2.2.6	Signalbeskrivelser og plugglister for timermodulen (TIM)	13
2.3	AMON	16
2.3.1	Programmering	16
2.3.2	Analoge innganger	16
2.3.3	Sampling av analoge data	18
2.3.4	Signaler og plugglister for AMON	18
2.4	PUSEK	21
2.4.1	Synkord, rammeteller og andre konstanter i formatet	23
2.4.2	Komprimering	24
2.4.3	Formatteller	24
2.4.4	Digitalt grensesnitt	24
2.4.5	Digitale inn- og utganger	26
2.4.6	Datautganger	28
2.4.7	Flaggmonitorer	28
2.4.8	Signaler og plugglister for PUSEK	29
2.5	SLAVEPUSEK	32
2.6	POW (Power)	34
2.6.1	Batteritilkobling og batterilading	34
2.6.2	Tilkobling av barometriske brytere	37
2.6.3	Tilkobling av instrumenter	37
2.6.4	Temperatursensor	38
2.6.5	Plugglister	38
3	**Programmering av STAPPE**	**41**
3.1	C-preprosessoren gpp	41
3.2	Programmering av formatet i STAPPE	43

3.2.1	conv_for.bat	43
3.2.2	conv_format.exe	44
3.2.3	Beskrivelse av formatet (`.for`-filer)	44
3.2.4	Konstanter	48
3.2.5	Kommuteringsrate	48
3.2.6	Deler av et ord	49
3.2.7	Sette sammen ord	49
3.2.8	Sette sammen rammer	50
3.2.9	Synkord	50
3.2.10	Rammeteller og formatteller	51
3.2.11	Merking av formatet med kompileringstidspunkt	51
3.2.12	Programmering av ubrukte deler av formatet	53
3.2.13	Avslutning	53
3.3	Programmering av TIMER	53
3.3.1	conv_tim.bat	53
3.3.2	conv_timer.exe	54
3.3.3	Beskrivelse av timersekvensen (`.tim`-filer)	55
3.4	Installering av programvare	61
3.4.1	Katalogstruktur:	61
3.4.2	Filassosiasjon og path; detaljert beskrivelse	61

1 Innledning

I 2005 til 2007 designet, produserte og testet Romfysikkgruppen ved FFI en ny telemetrienkoder til bruk i housekeeping-seksjonen på vitenskapelige sonderaketter. Enkoderen brukes i vitenskapelige raketter i ECOMA-prosjektet ved FFI og hele eller enkeltmoduler av den har vært brukt i andre raketter skutt opp fra Andøya rakettskytefelt. 10 stk. komplette enkodere ble kjøpt av Andøya rakettskytefelt etter avtale med FFI. Som en del av avtalen får Andøya rakettskytefelt også dokumentasjon for enkoderen i form av en bruksanvisning, i form av denne rapporten. Målgruppen for denne bruksanvisningen er elektronikkingeniører eller romingeniører som allerede er kjent med enkodersystemer og hvordan de brukes. Bruksanvisningen skal være et hjelpemiddel til å koble opp enkoderen uten feil og til å være en veiledning i hvordan programmere den. Rapporten er *ikke* skrevet for å være en lærebok i telemetri eller for å gi mer detaljert kunnskap, som for eksempel hvordan enkoderen fungerer i minste detalj, eller å gi detaljer om designparametre, -løsninger og -prosess. Rapporten er heller ikke ment å være detaljert nok til feilsøking på de enkelte enkodermodulene, reparasjon eller ombygging. Rapporten er delt opp i to hoveddeler – én som stort sett tar for seg det hardwaremessige, elektrisk interface, plugglister, og liknende (kapittel 2) og én del som i hovedsak omhandler programmering/konfigurering av enkoderen (kapittel 3).

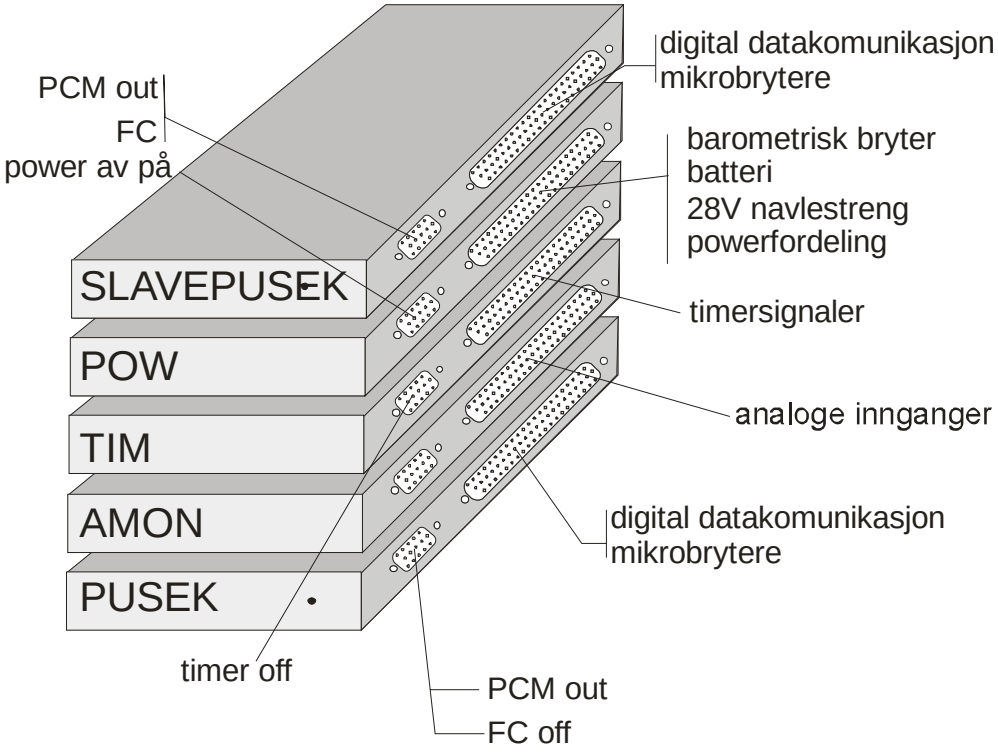
2 STAPPE - Stacked Timer Amon Power Pusek Extendable.

STAPPE (Stacked Timer Amon Power Pusek Extentable) er et system beregnet på sonderaketter bestående av moduler for telemetri, powerfordeling og en timer for fyring av pyroteknikk under flight. Systemet består av ett eller flere kort montert i rammer som stakkes over hverandre (se Figur 2.1). I bakkanten av modulene sitter det en europaplugg som kobler dem sammen elektrisk. Hunndelen av pluggen er på oversiden av kortene og på undersiden er pinnene som plugges inn i kortet under. Et unntak er bunnkortet som er uten pinner og må være nederst. Øvrige kort kan monteres i en vilkårlig rekkefølge. Det er allikevel anbefalt å bruke rekkefølgen vist i Figur 2.2 siden det er den som ble brukt under utviklingen.
STAPPE består av følgende moduler:

- TIM er en timer for fyring av pyroteknikk og for å slå av og på instrumenter under flight og for å slå av nyttelasten ved berging
- AMON utvider enkoderen med analoge innganger
- POW er et power kontrollkort samt monitorering av strømtrekk
- PUSEK er hovedkortet i systemet og er en komplett digital pcm-enkoder
- SLAVEPUSEK en ekstra digital enkoder uten utvidelsesmuligheter

Når STAPPE er satt sammen av mer enn en modul må en av disse styre utlesning av data fra de andre. Dette vil normalt være den samme modulen som leser dataene fra bussen og sender dem ut av boksen. En av modulene må også innholde en spenningsforsyning som lager de interne spenningene. For tiden er det bare PUSEK som inneholder fellesfunksjoner for STAPPE.

Figur 2.1 STAPPE, montert i rammer, klar til å monteres i en nyttelast



Figur 2.2 Skisse av STAPPE, med anbefalt monteringsrekkefølge.

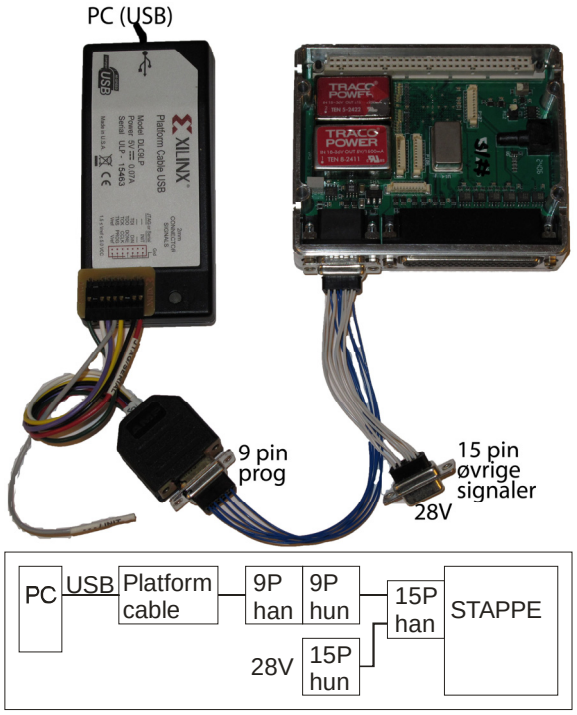
2.1.1 Montering og demontering av moduler.

Når modulene skal settes sammen er det viktig å kontrollere at pinnene for den interne bussen kommer i riktig hull. Videre er det viktig å skyve begge endene av pluggen sammen omtrent like raskt for å ikke bøye pinnene. Ta det med ro og vær forsiktig! Skal rammene tas fra hverandre

kan man stikke en flat skrutrekker forsiktig inn mellom dem og vri varsomt for å få en liten glippe mellom rammene. Skrutrekkeren brukes så til å bende rammene gradvis fra hverandre, vekselvis i den ene og andre enden av pluggen slik at pinnene ikke blir bøyd eller skadet. Pass på at skrutrekkeren ikke skraper opp kortet. Det er meget lett å bøye pinnene hvis en prøver å dra modulene fra hverandre eller bende for fort.

2.1.2 Programmering/konfigurering

Noen av modulene må konfigureres før bruk. Dette gjøres ved å programmere en tabell i en prom. Innholdet i disse tabellene er modulavhengig, og kan inneholde f.eks. oppsett av telemetriformatet eller timersekvensen. Disse tabellene vil variere fra nyttelast til nyttelast, og er noe brukeren må programmere. I tillegg har alle modulene firmware som ligger i en separat prom. Denne programmeres normalt ikke av brukeren.



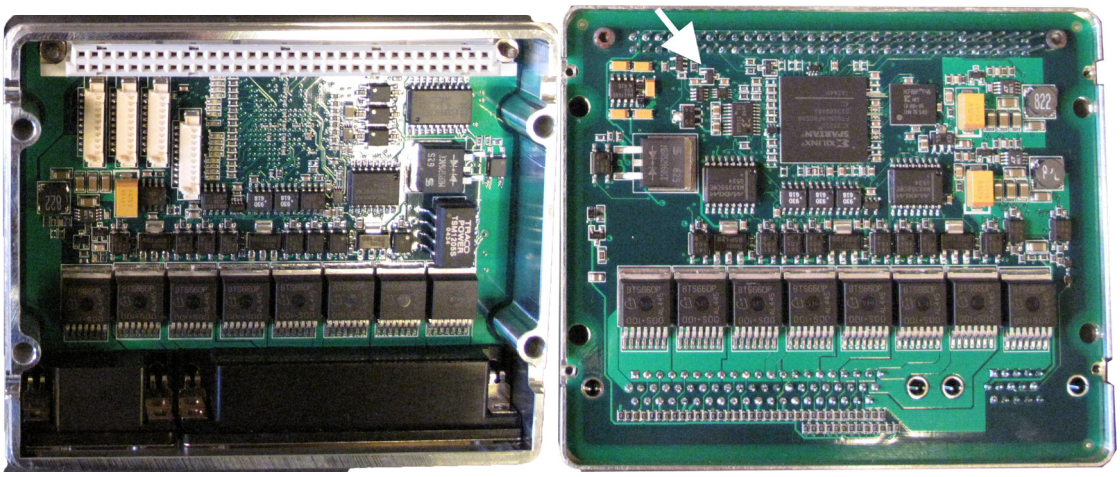
Platform cable	9-pins	15-pins på STAPPE
Vref	1	3
GND	2	12
TCK	3	10
TDO	4	13
TDI	5	9
TMS	6	11
	15-pins	
	28V	5
	28V return	4

Figur 2.3 Oppkobling for programmering av kort.

Alle modulene har en 15 pins high density D sub plugg som blant annet inneholder nødvendige signaler for å programmere modulene. Modulene kobles til en pc via en programmeringsboks fra Xilinx. Xilinx har forskjellige programmeringsbokser, og her omtales bare 'platform cable usb' (Xilinx kaller boksene for cable) som er vist i Figur 2.3 Programmeringsboksen kobles til pcen med en usb-kabel. Kabelen fra programmeringsboksen til STAPPE lager brukeren selv. I Figur 2.3 er det vist et eksempel med en adapterkabel som splitter 15 pins pluggen i en 9 pins D sub med programmeringsignalene og en 15 pins med resten av signalene.
PUSEK har egne DC/DC-omformere og lager dermed selv nødvendige interne spenninger fra 28V. Øvrige moduler er avhengig av å få interne spenninger fra bussen og må derfor sitte sammen med et kort med spenningsforsyning når de skal programmeres. 28V til programmering kan kobles til hvilken som helst av 15 pins-pluggene i stakken eller til de ordinære 28V inngangene på POW.

2.2 TIM (Timer)

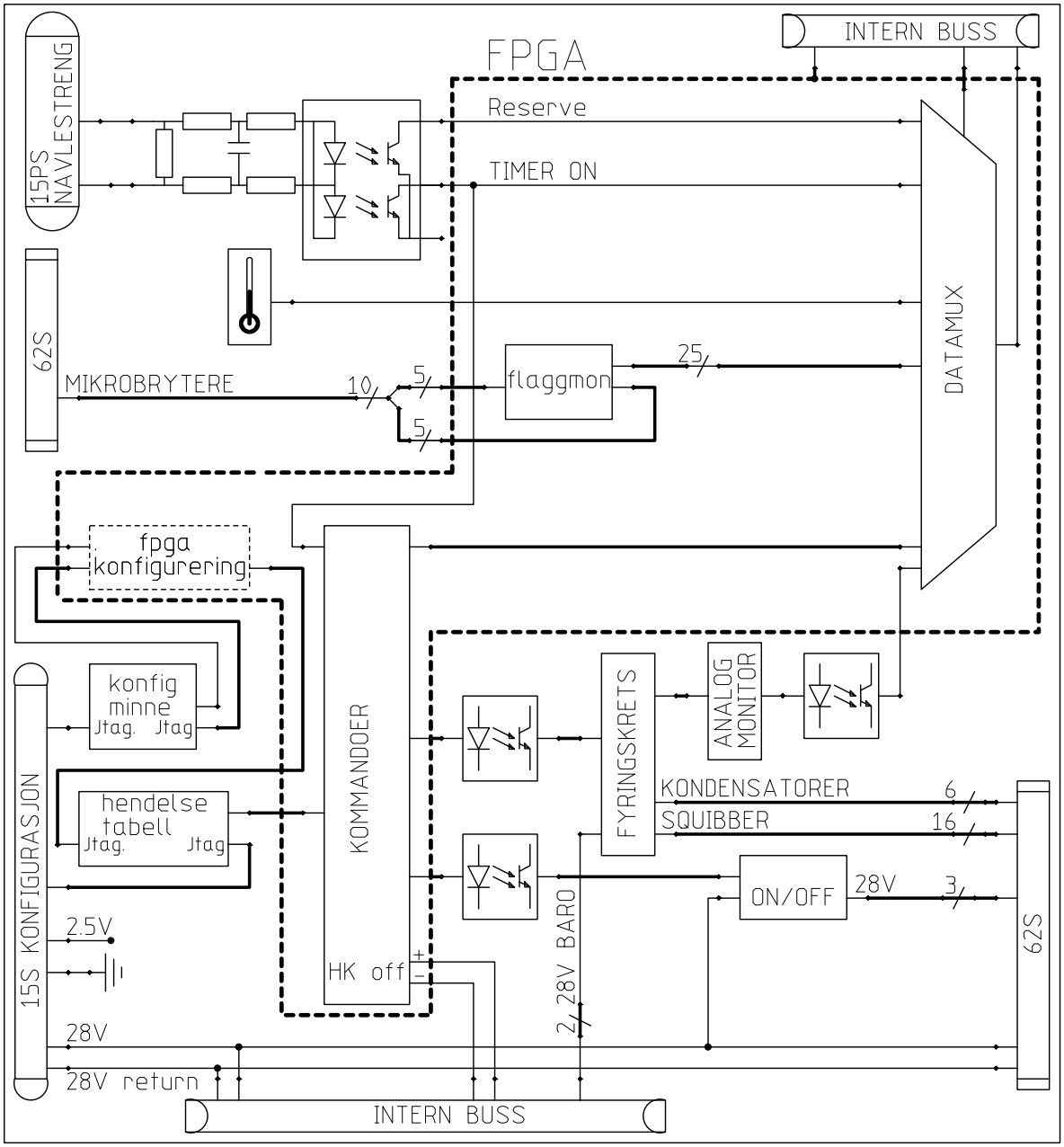
Timermodulen brukes til å generere hendelser gjennom flukten. Den viktigste oppgaven er å fyre av squiber (pyroteknikske aktuatorer) for å kaste av nesekoner, åpne dører, felle ut antenner og lignende. Den har også mulighet for å skru av og på 28V til forbrukere med moderat strømtrekk, og som avslutning skru nyttelasten helt av. Sistnevnte funksjonalitet brukes i forbindelse med nyttelaster som skal berges, så de ikke blir ødelagt av å lande i vann med spenning på. TIM har en egen temperatursensor inne på kortet, se Figur 2.4. Vi vil i det følgende beskrive TIM i mer detalj, med utgangspunkt i blokkskjemaet vist i Figur 2.5.



Figur 2.4 TIM overside (til venstre) og underside (til høyre). Pilen peker på temperatursensoren.

2.2.1 Utganger til pyroteknikk.

Strøm til fyring av squibber kommer fra kondensatorer koblet til 62 pins pluggen på forsiden av timerkortet. Kondensatorene lades til 28V med en strøm på ca.120 mA, og ladingen styres av to separate barometriske brytere som er koblet til bussen via powerkortet (POW). Hver kondensator brukes til to eller tre squibbroer og en må vente til kondensatorene er ladet opp igjen før neste gang de brukes. Med en ladestrøm på 120 mA og en anbefalt kondensator på 4700 uF tar det ca 1 sekund å lade opp kondensatorene, som derfor er absolutt minimum tid før gjenbruk av en kondensator. Kretskortet er *ikke* dimensjonert for samtidig bruk av mer enn én utgang pr. kondensator. Vi anbefaler å bruke en utladningsmotstand på 10 kΩ direkte på hver enkelt kondensator for å lade dem ut når de ikke er koblet til TIM.
Vi fraråder å koble squibbruer i parallell eller serie da en kortslutning eller et brudd i en av bruene vil føre til feilfunksjon i alle squibene tilkoblet squibbroen med feil. Det vil også være vanskelig å fordele strøm likt til squibbene om de parallellkobles, da motstanden varierer fra squib til squib. Tilsvarende gjelder for fordeling av spenning til squibbene om de seriekobles. Prinsippskjema over fyringskretsene er vist i Figur 2.6, med anbefalt tilkobling av kondensatorer og squibber. Totalt kan man koble til 6 kondensatorer og 16 squibbruer. En oversikt over hvilke kondensatorer som leverer strøm til hvilke squibbruer er gitt i Tabell 2.1.



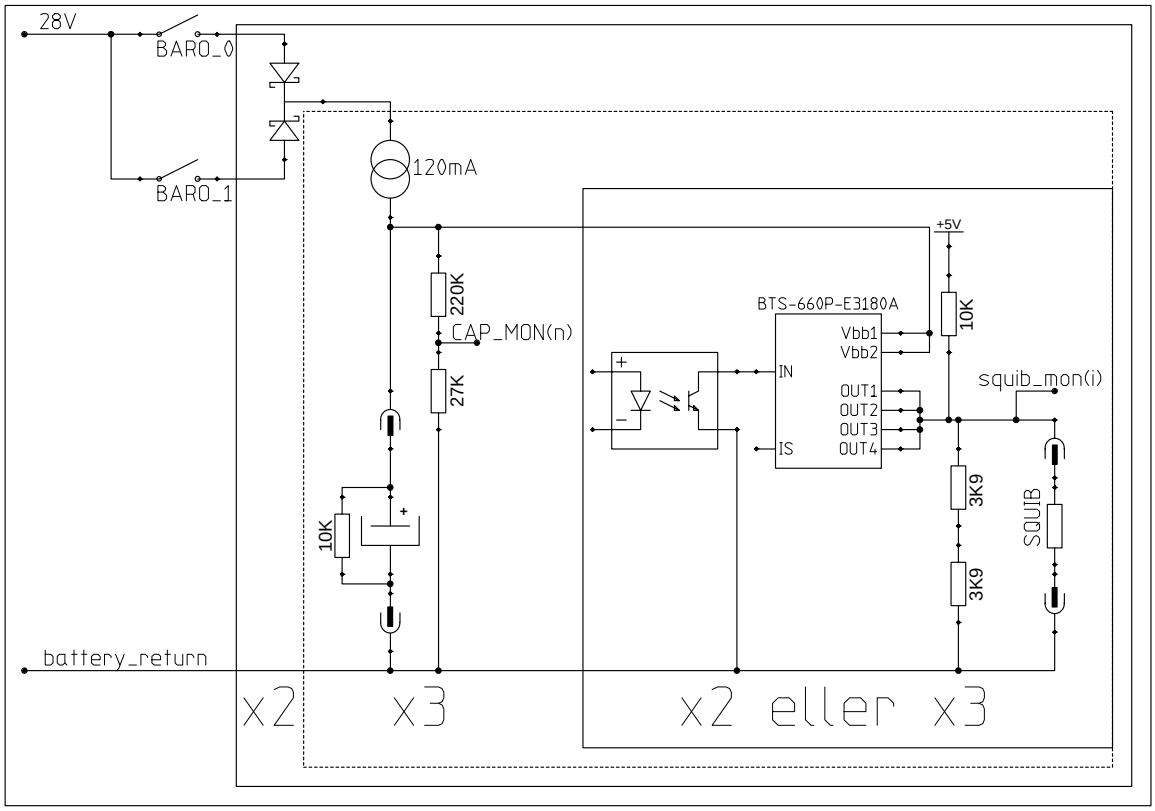
Figur 2.5 Blokkskjema for TIM (Timer-modulen).

2.2.2 28V utganger

TIM kontrollerer 3 stk. 28V utganger kalt pwr. De egner seg for bytte av moder på instrumenter eller til å skru instrumenter av og på. 28V til pwr går ikke via de barometriske bryterne. Verken spenning eller strøm fra utgangene monitoreres men kommandoene som styrer dem kan monitoreres. Den samlede strømmen for de tre pwr utgangene skal begrenses til 1A for å ikke overbelaste den interne bussen.

2.2.3 Skru av nyttelasten

Nyttelaster som lander i saltvann med utvendige spenninger på blir skadet, så for å beskytte nyttelaster som skal gjenbrukes kan TIM kommandere POW til å skru av nyttelasten via den



Figur 2.6 Fyringskretsen, med anbefalt tilkobling av fyrekondensatorer og squibber.

Kondensator #	**Squibbru #**
0	0, 6, 12
1	1, 7, 13
2	2, 8, 14
3	3, 9, 15
4	4, 10
5	5, 11

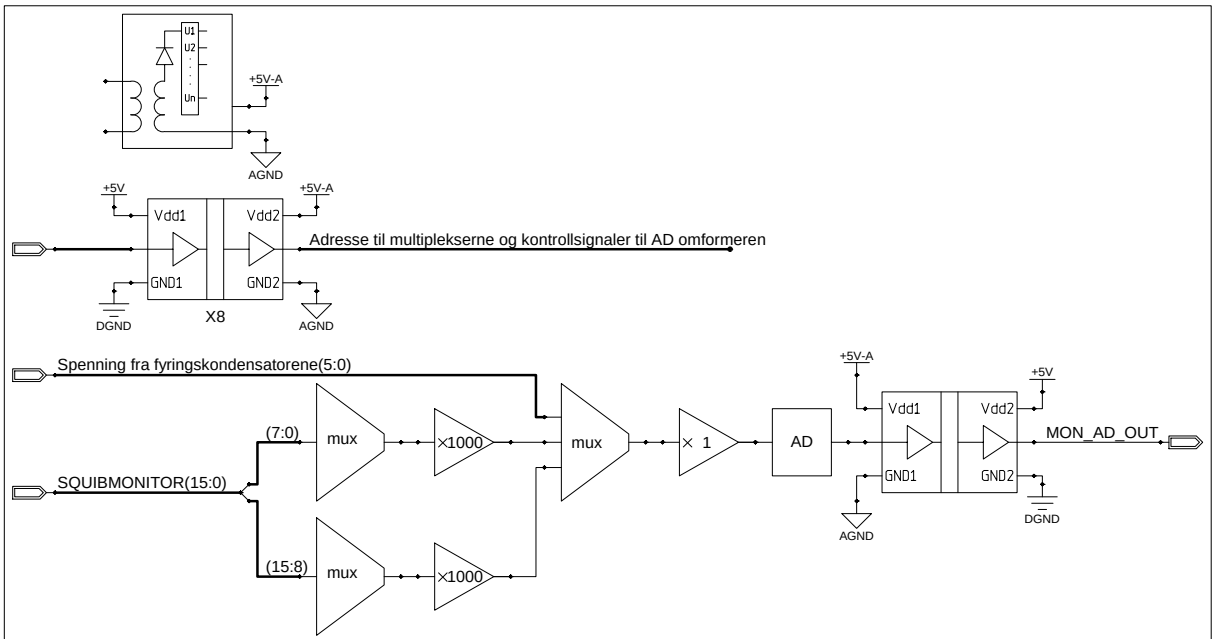
Tabell 2.1 Oversikt over hvilke kondensatorer som leverer strøm til hvilke squibbruer

interne bussen. For å kunne skru nyttelasten av så sent som mulig har TIM en egen kommando som pauser den hvis den får 28V fra de barometriske bryterne. Når lufttrykket stiger på nedtur og de barometriske bryterne åpner vil TIM starte opp igjen, og fortsetter til den kommer til kommandoen som skrur av nyttelasten.

2.2.4 Analoge monitorer

TIM monitorerer spenningen på fyringskondensatorene og motstanden i squibene. Motstanden i en squibbru er vanligvis så lav at motstanden i ledningen frem til den påvirker målingen i betydelig grad. En god måte å bruke squibmonitorene på er å måle en 0Ω motstand gjennom hele ledningsmatta og notere avlesningen, og så gjenta med en motstand med nominelt samme verdi som squibben. Disse to verdiene sammenlignes siden med den verdien monitoren gir når en har

montert squiben. Måleområdet for kondensatorspenningene er 0-45,7V og motstandsområdet for squibmålinger er 0-10Ω. En prinsippskisse for modulen for analog monitorering er gjengitt i Figur 2.7.



Figur 2.7 Analog mux og AD (prinsippskisse)

2.2.5 Digitale monitorer

Det vil ofte være nyttig å overvåke interne tilstander i enkoderen og effekten av kommandoer fra timerenheten. Derfor har TIM noen enbits flaggmonitorer tilgjengelig for slik overvåking. Disse statusbitene kan plasseres fritt og uavhengig av hverandre i formatet. Disse statusbitene er oppsummert i Tabell 2.2. Merk at statusbitene TIM_ON og TIM_RUNNING i praksis gir samme informasjon, så man får ingen ekstra informasjon ved å bruke begge framfor bare én av dem.

2.2.6 Signalbeskrivelser og plugglister for timermodulen (TIM)

På fronten av TIM er det to plugger som er tilgjengelig for tilkobling av elektronikk som skal styres av modulen. Den store pluggen, med 62 pinner, har utganger som brukes for å fyre pyrotekniske enheter (squibber, gassgivere), lade pyro-kondensatorene, slå av og på instrumenter, og for tilgang til intern 28V strømforsyning i STAPPE (enten for å forsyne eksterne brukere eller for å forsyne STAPPE med ekstra 28V). Tabell 2.3 gir en oversikt over de tilgjengelige signalene, mens Tabell 2.4 inneholder plugglista.

For den eksterne mikrobrytermatrisen viser vi til beskrivelsen av PUSEK da disse er lik, bortsett fra at ut- og innganger i flaggmatrisen for TIM har signalnivåer på 0 til 5V mot 0 til 3,3V for PUSEK.

Selv om alle returpinnene på 62 pins pluggen er koblet sammen inne på kortet skal en bruke returpinnene som er spesifisert i plugglista. Dette for å unngå store strømmer inne på kortet.

Navn fra fila timf_std.inc	Status
TIM_BARO_1 = ’1’	28V fra barometrisk bryter 1
TIM_BARO_2 = ’1’	28V fra barometrisk bryter 2
TIM_BARO_1_OR_2 = ’1’	28V fra minst en av de barometriske bryterne
TIM_HK_OFF = ’1’	Nyttelasten skrus av
TIM_STOP = ’1’	Timeren er ferdig og har stoppet
TIM_WAIT_FOR_BARO = ’1’	Timeren er pauset og venter til TIM_BARO_1_OR_2 = ’0’ før den starter opp igjen.
TIM_ON = ’0’	Timer off fra navlestrengen er på og timeren står på starten av timersekvensen.
TIM_ON = ’1’	Timer off-signalet fra navlestrengen er av og timeren går. Forblir høy selv om TIM_STOP sier at timeren har stoppet. Tiden til de forskjellige eventene måles fra det tidspunktet TIM_ON går høy. MERK: *Det er viktig at TIM_ON går fra ’0’ til ’1’ på riktig tidspunkt.*
TIM_HK_OFF_CMD = ’1’	Kommandoen for å slå av nyttelasten er aktiv.
TIM_RUNNING = ’1’	Forsinket utgave av TIM_ON. I praksis det samme som TIM_ON.
SQIB_COMMAND_nn = ’1’	Squibbru nr nn kommanderes til å fyre.
PWR_m = ’1’	Kommandert 28V til pwr utgang nr m.

Tabell 2.2 Interne statusbit for timermodulen (TIM)

Pinnenavn	Funksjon
Squibbru zz	28V til squibbru nr. zz. Bruk Tabell 2.1 for å finne hvilken kondensator som forsyner squibbru zz. Avhengig av de barometriske bryterne.
Squibbru zz return	Retur for squibbru nr. zz.
Kondensator y	Positiv side av kondensator nr y. Husk minimum tid før gjenbruk av kondensatorene.
Kondensator y return	Negativ side av kondensator nr. y.
Flag a	Inngang fra matrisen av mikrobrytere.
Flag group a	Utgang til matrise av mikrobrytere.
Pwr x	28V utgang kontrollert av TIM. Uavhengig av de barometriske bryterne.
Pwr x return	Retur for pwr x.
28V	Tilgang til intern 28V i STAPPE. Brukes normalt ikke. Kan brukes til å forsyne eksterne brukere eller forsyne STAPPE med ekstra 28V. Overbelastning på denne pinnen vil stoppe STAPPE så den er frarådet brukt som utgang.
28V return	Retur for 28V.

Tabell 2.3 Tilgjengelige signaler på 62 pins pluggen

#	Pinnenavn	#	Pinnenavn	#	Pinnenavn
1	Pwr 1	22	Kondensator 1	43	Pwr 1 return
2	Squibbru 01	23	Kondensator 1 return	44	Squibbru 01 return
3	Squibbru 00	24	Flag group 1	45	Squibbru 00 return
4	Kondensator 0	25	Kondensator 00 return	46	Squibbru 06 return
5	Squibbru 06	26	Pwr 0	47	Pwr 0 return
6	Pwr 2	27	Pwr 2 return	48	Squibbru 07 return
7	Squibbru 07	28	Flag 2	49	Flag 4
8	Flag group 0	29	Flag group 2	50	Flag group 4
9	Squibbru 13	30	Flag 3	51	28V
10	Squibbru 12	31	Flag group 3	52	Squibbru 12 return
11	Flag 1	32	Squibbru 13 return	53	28V return
12	Kondensator 3	33	Kondensator 3 return	54	Squibbru 02 return
13	Kondensator 2	34	Kondensator 2 return	55	Squibbru 08 return
14	Squibbru 02	35	Flag 0	56	Squibbru 03 return
15	Squibbru 03	36	Squibbru 09 return	57	Squibbru 15 return
16	Squibbru 08	37	Squibbru 14 return	58	Squibbru 10 return
17	Squibbru 09	38	Squibbru 04 return	59	Kondensator 4 return
18	Squibbru 15	39	Squibbru 11 return	60	Squibbru 05 return
19	Squibbru 14	40	Kondensator 5 return	61	Squibbru 10
20	Kondensator 4	41	Squibbru 04	62	Squibbru 11
21	Kondensator 5	42	Squibbru 05		

Tabell 2.4 Pluggliste for 62-pins high density hunplugg på fronten av timermodulen. Se Tabell 2.1 for oversikt over hvilke kondensatorer som forsyner hvilke squibber.

Pinnenavn	**Programmeringsfunksjon (P) / UMB-funksjon (UMB)**
28V	(P) Power input ved programmering
28V return	(P) Power return
Int 2,5V	(P) Brukes av programkabelen fra Xilinx
Internal return	(P) Brukes av programkabelen fra Xilinx
TCK	(P) Brukes av programkabelen fra Xilinx
TDI	(P) Brukes av programkabelen fra Xilinx
TDO	(P) Brukes av programkabelen fra Xilinx
TMS	(P) Brukes av programkabelen fra Xilinx
UMB power (12V)	(UMB) Alternativ tikobling for UMB power til STAPPE. Brukes vanligvis ikke.
Timer off	(UMB) En positiv spenning på 10-30V i forhold til UMB return stopper og resetter TIM.
UMB return	(UMB) Retur for Timer off og UMB power. Galvanisk skilt fra Return og Internal return.

Tabell 2.5 Tilgjengelige signaler på 15 pins pluggen

#	**Pinnenavn**	#	**Pinnenavn**	#	**Pinnenavn**
1	(ikke i bruk)	6	UMB return	11	TMS
2	(ikke i bruk)	7	UMB power (12V)	12	Internal return
3	Int 2.5V	8	Timer off	13	TDO
4	28V return	9	TDI	14	Internal return
5	28V	10	TCK	15	(ikke i bruk)

Tabell 2.6 Pluggliste for 15-pins high density hunplugg på fronten av timermodulen

Den lille pluggen på fronten av TIM, med 15 pinner, har tre funksjoner for brukeren:

1. For programmering av timertabellen i TIM
2. Signal fra navlestrengen (UMB) for å stoppe og resette timeren
3. Tilførsel av 12V UMB power til STAPPE. Denne funksjonen brukes som regel ikke.

Power return, UMB return og internal return er galvanisk skilt. Tabell 2.5 gir en oversikt over signalene som er tilgjengelig på pluggen med 15 pinner på TIM. Tabell 2.6 gir en pluggliste for denne pluggen.

2.3 AMON

AMON (Analog MONitor) er modulen i STAPPE som alle instrumenter som leverer analoge signaler kobles opp mot. AMON har 21 differensielle og 6 singelendede innganger. Til hver av de singelendede inngangene er det en utgang med 5V til bruk for sensorer med lavt effektforbruk. For å unngå overhøringen man ville ha fått mellom kanaler ved bruk av en analog multiplekser + én AD har alle inngangene blitt utstyrt med hver sin AD. Alle ADene har 12 bit oppløsning. For monitoreringsformål har AMON en egen temperatursensor montert på kortet. Se Figur 2.8.

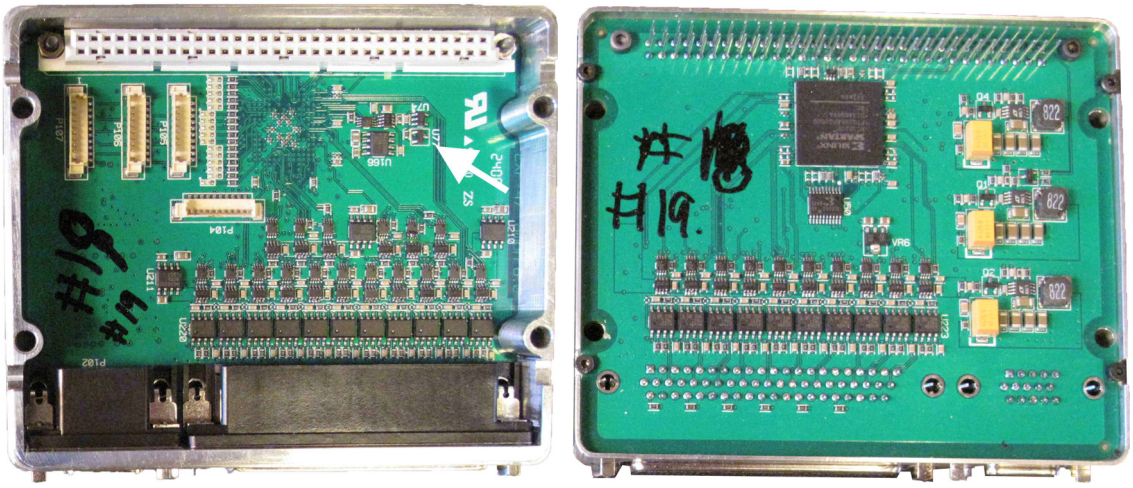
2.3.1 Programmering

AMON programmeres normalt ikke av brukeren, men modulen kan få oppdatert firmware på samme måte som de andre kortene i STAPPE

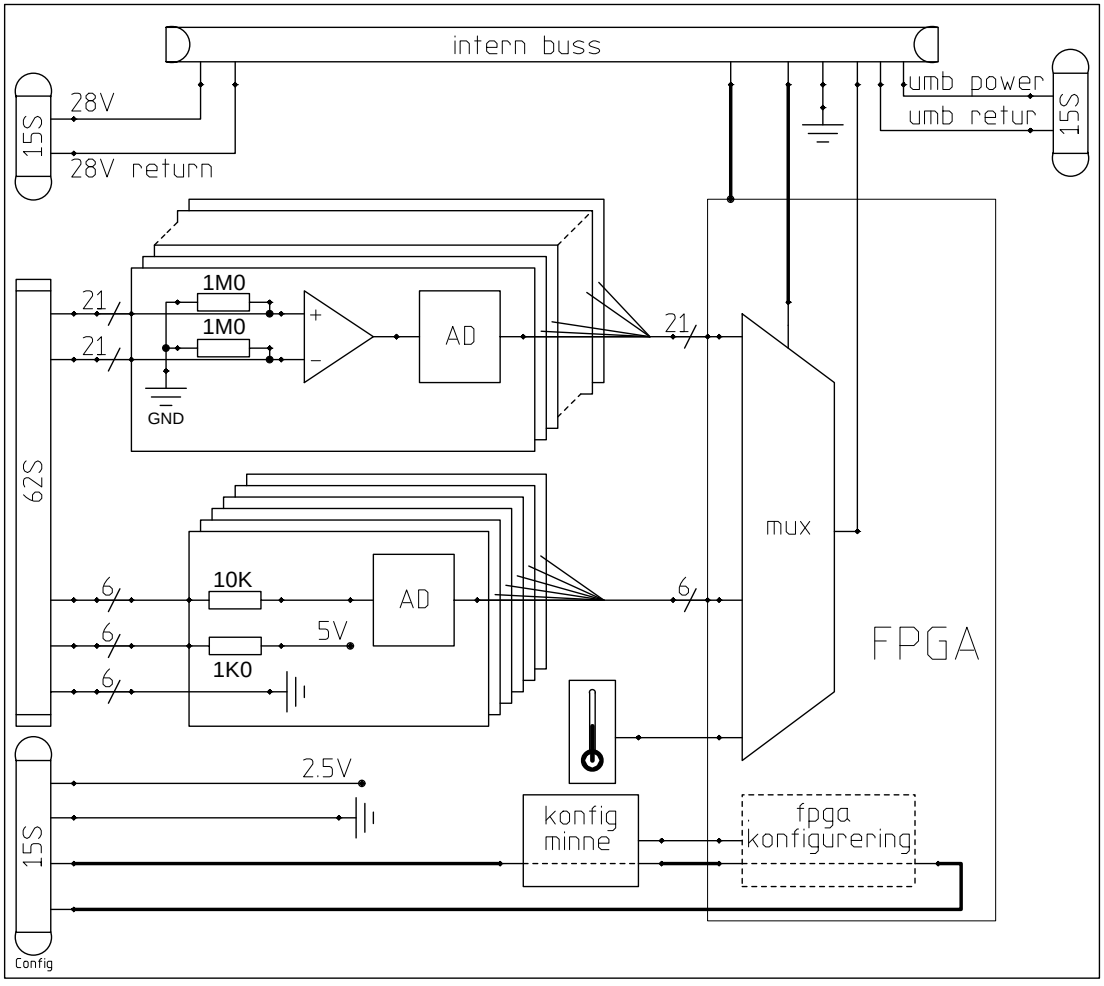
2.3.2 Analoge innganger

AMON har 21 differensielle innganger som vist i Figur 2.10. For full skala på utgangen skal INP+ svinge 0 til 5V i forhold til INP-. For å unngå at inngangsforsterkerne skal gå i metning må begge inngangene holde seg innenfor ±10V i forhold til return. Inngangsimpedans for hver av inngangene er 1MΩ i forhold til return.

For de 6 singleendede inngangene (vist i Figur 2.11) får man full skala ut fra AD ved et dynamisk område på INP = 0V til 5V i forhold til return. Inngangsimpedansen for disse inngangene er 10kΩ. Inngangene bør styres av drivere med utgangsimpedans lavere enn 1kΩ. Utgangene merket ”5 Volt” kan brukes som strømforsyning til kretser som trekker lite strøm, som for eksempel temperatursensorer. Utgangsimpedansen for ”5 Volt” er 1kΩ, og disse utgangene er kortslutningssikre mot return.

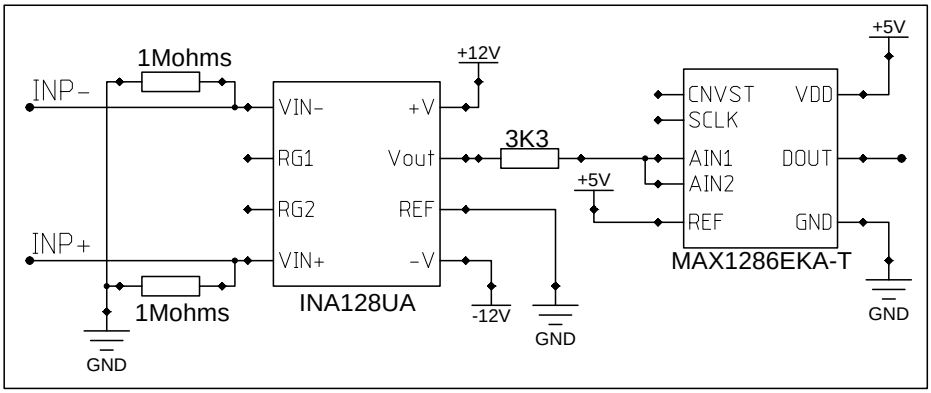


Figur 2.8 AMON overside (til venstre) og underside (til høyre). Pilen peker på temperatursensoren.

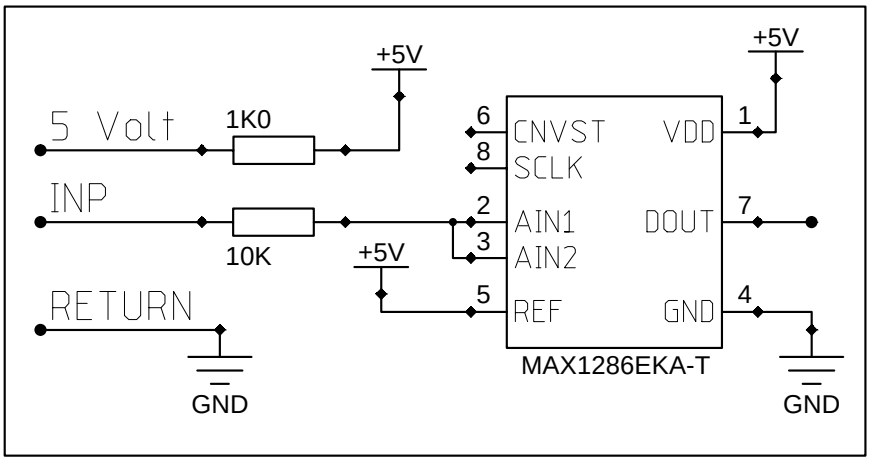


Figur 2.9 Blokkskjema for AMON

Inngangspenningen til de analoge inngangene må ikke overstige ± 40V i forhold til return. Høyere spenninger enn dette kan skade inngangsforsterkeren.



Figur 2.10 Differensiell inngang.



Figur 2.11 Singleended inngang.

2.3.3 Sampling av analoge data

Data fra AD-omformerne leses ut bit for bit, med MSB først. Figur 2.12 illustrerer hvordan sampling og utlesing av en analog inngang skjer. I dette tilfellet blir inngangen samplet, lest ut og sendt ut i telemetriformatet med så kort intervall som mulig. Legg merke til minimumsavstanden i formatet mellom sampler fra samme analoge inngang. Plasseres første bit fra et sample i bit N i telemetriformatet kan neste sample fra samme inngang tidligst samples ved bit N+13, og tidligst plasseres i bit N+24. Dette er uavhengig av hvor mange bit fra samplet som faktisk brukes i formatet. Alle AD-omformerne har 12 bits oppløsning, men en kan velge å bruke færre bit, helt ned til 1, dvs bare MSB. Legg merke til at MSB av hver sample *må* leses, mens de andre bitene kan ignoreres.

2.3.4 Signaler og plugglister for AMON

Tabell 2.7 gir en oversikt over hvilke typer signaler som finnes på den store pluggen (62 pinner) på fronten av AMON, mens Tabell 2.8 gir en pluggliste for denne. Tilsvarende gir Tabell 2.9 oversikt over signalene på den lille pluggen (15 pinner) mens Tabell 2.10 gir plugglista. Merk at bortsett fra Timer off-signalet er funksjoner på og utlegg av 15 pins pluggen identisk med den for timermodulen.

Bit nr.	Kommentar
N	Utlesing av MSB for sample nr. K
N+1	Utlesing av MSB-1 for sample nr. K
N+2	Utlesing av MSB-2 for sample nr. K
…	
N+11	Utlesing av LSB for sample nr. K
N+12	
N+13	AD starter sampling for sample nr. K+1
...	
N+24	Tidligste utlesing av MSB for sample nr. K+1
N+25	Tidligste utlesing av MSB-1 for sample nr. K+1
…	

Figur 2.12 Samplingtidspunkt og minimum samplingavstand. MSB må leses ut, resten av bitene kan ignoreres.

Pinnenavn	Funksjon
+Analog inn XX	Differensiell analog inngang nr XX
-Analog inn XX	Referanse for Differensiell analog inngang nr XX
Analog inn YY	Singleended analog inngang nr YY
return	Referanse for de singelendede analoge inngangene
1kΩ pullup to 5Volt	Spenningsforsyning; lavt effektforbruk.

Tabell 2.7 Tilgjengelige signaler på 62 pins pluggen.

#	Pinnenavn	#	Pinnenavn	#	Pinnenavn
1	+Analog inn 0	22	-Analog inn 0	43	1kΩ pullup to 5 Volt
2	+Analog inn 1	23	-Analog inn 1	44	return
3	+Analog inn 2	24	-Analog inn 2	45	Analog inn 21
4	+Analog inn 3	25	-Analog inn 3	46	1kΩ pullup to 5 Volt
5	+Analog inn 4	26	-Analog inn 4	47	return
6	+Analog inn 5	27	-Analog inn 5	48	Analog inn 22
7	+Analog inn 6	28	-Analog inn 6	49	1kΩ pullup to 5 Volt
8	+Analog inn 7	29	-Analog inn 7	50	return
9	+Analog inn 8	30	-Analog inn 8	51	Analog inn 23
10	+Analog inn 9	31	-Analog inn 9	52	1kΩ pullup to 5 Volt
11	+Analog inn 10	32	-Analog inn 10	53	return
12	+Analog inn 11	33	-Analog inn 11	54	Analog inn 24
13	+Analog inn 12	34	-Analog inn 12	55	1kΩ pullup to 5 Volt
14	+Analog inn 13	35	-Analog inn 13	56	return
15	+Analog inn 14	36	-Analog inn 14	57	Analog inn 25
16	+Analog inn 15	37	-Analog inn 15	58	1kΩ pullup to 5 Volt
17	+Analog inn 16	38	-Analog inn 16	59	return
18	+Analog inn 17	39	-Analog inn 17	60	Analog inn 26
19	+Analog inn 18	40	-Analog inn 18	61	return
20	+Analog inn 19	41	-Analog inn 19	62	return
21	+Analog inn 20	42	-Analog inn 20		

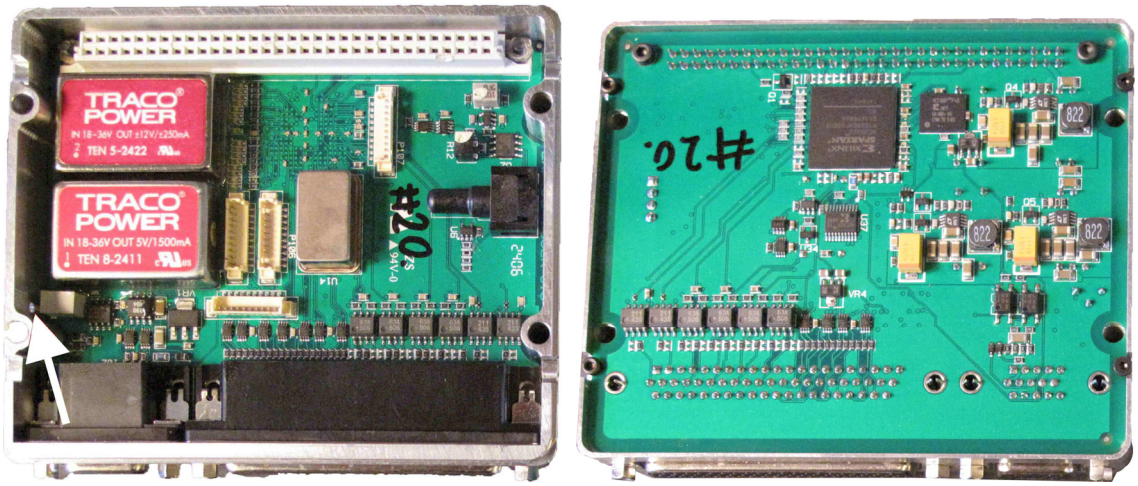
Tabell 2.8 Pluggliste for 62-pins high density hunplugg på fronten av AMON-modulen.

Pinnenavn	**Programmeringsfunksjon (P) / UMB-funksjon (UMB)**
28V	(P) Power input ved programmering
28V return	(P) Power return
Int 2,5V	(P) Brukes av programkabelen fra Xilinx
Internal return	(P) Brukes av programkabelen fra Xilinx
TCK	(P) Brukes av programkabelen fra Xilinx
TDI	(P) Brukes av programkabelen fra Xilinx
TDO	(P) Brukes av programkabelen fra Xilinx
TMS	(P) Brukes av programkabelen fra Xilinx
UMB power (12V)	(UMB) Alternativ tikobling for UMB power til STAPPE. Brukes vanligvis ikke.
UMB return	(UMB) Retur for Timer off og UMB power. Galvanisk skilt fra Return og Internal return.

Tabell 2.9 Tilgjengelige signaler på 15 pins pluggen

#	Pinnenavn	#	Pinnenavn	#	Pinnenavn
1	(ikke i bruk)	6	UMB return	11	TMS
2	(ikke i bruk)	7	UMB power (12V)	12	Internal return
3	Int 2.5V	8	(ikke i bruk)	13	TDO
4	28V return	9	TDI	14	Internal return
5	28V	10	TCK	15	(ikke i bruk)

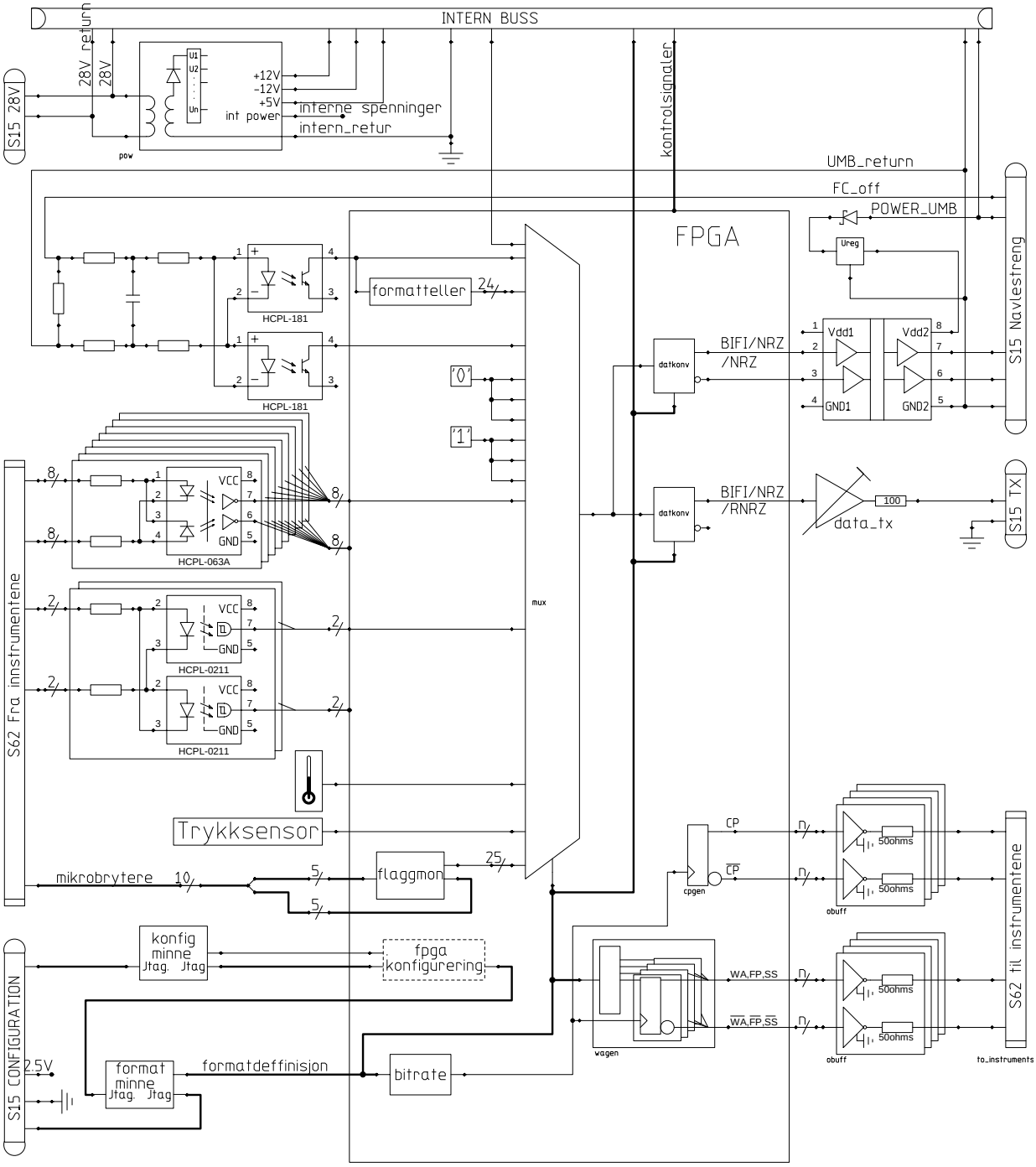
Tabell 2.10 Pluggliste for 15-pins high density hunplugg på fronten av AMON-modulen



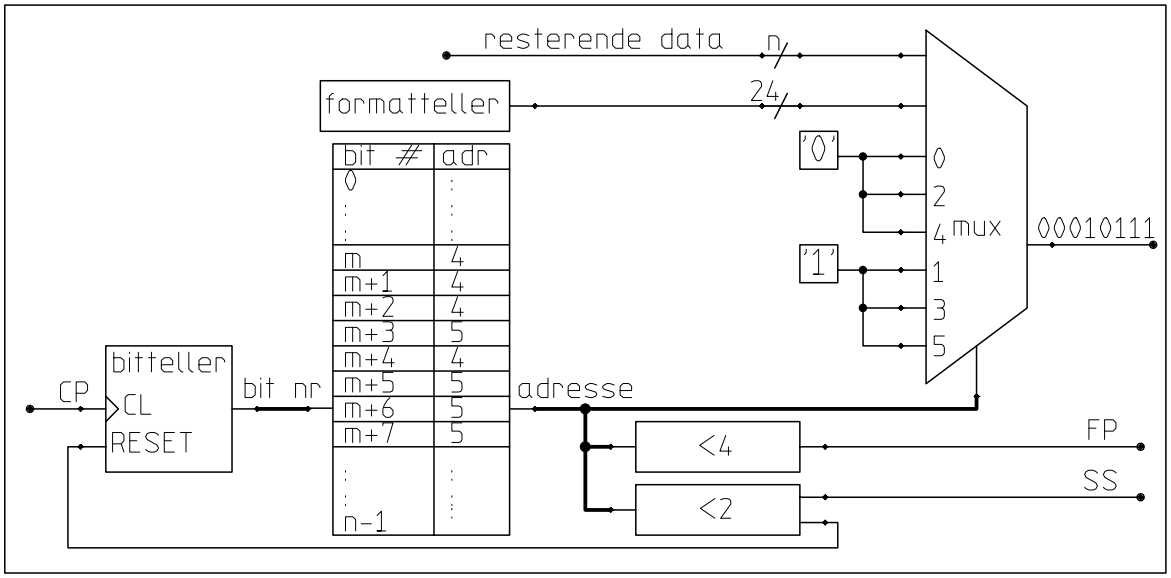
Figur 2.13 PUSEK overside (til venstre) og underside (til høyre). Pilen peker på et hull i ramma som gir tilgang til et potensiometer for å justere amplituden på signalet til radiosenderen.

2.4 PUSEK

PUSEK er hovedkortet i STAPPE og er en komplett enkoder med PCM-utgang og med digitalt grensesnitt til måleinstrumenter. Kortet har DC/DC-omformere som konverterer 28V til ±12V og +5V. Disse spenningene gjøres tilgjengelig for de andre kortene i stakken via den interne bussen. PUSEK bruker ikke selv ±12V internt men lager flere andre interne spenninger for eget bruk, disse er bare tilgjengelig eksternt via en testplugg inne på kortet. PUSEK har en temperaturmonitor som måler temperaturen inne på kortet like ved den ovnsstabiliserte oscillatoren (se Figur 2.13). Temperaturmonitoren har et temperaturområde på 0 til 150˚C, men sensoren er beregnet på 0 til 100˚C, så målinger over 100˚C er unøyaktige. PUSEK har også en trykksensor beregnet for bruk ved test og overvåking av luftlekkasjer i lufttette nyttelaster. Måleområdet på denne er 0 til 2 atmosfærer. Alle logiske funksjoner i PUSEK utføres i en Xilinx FPGA, mens buffring av eksterne signaler går via dedikerte buffere. Digitale inngangssignaler går via optokoblere for å eliminere jordsløyfer. Et blokkskjema av PUSEK er vist i Figur 2.14.



Figur 2.14 Blokkskjema PUSEK



Figur 2.15 Generering av synkord, rammeteller og lesing av formateller.

Adresse	Inngang mulitplekser	Funksjon	Plassering i formatet
0	’0’	FP, SS, resetter bitteller	Første bit i første ramme
1	’1’	FP, SS, resetter bitteller	Første bit i første ramme
2	’0’	FP	Første bit i øvrige rammer
3	’1’	FP	Første bit i øvrige rammer
4	’0’	-	Alle andre steder
5	’1’	-	Alle andre steder

Tabell 2.11 Adressering av ‘0’ og ‘1’ i PUSEK.

2.4.1 Synkord, rammeteller og andre konstanter i formatet

Formatbeskrivelsen ligger lagret som en tabell i et flashminne med én adresse for hvert bit i formatet. Utlesningen av minnet styres av en teller som klokkes av bitklokka CP. Multiplekseren har flere innganger med de konstante verdiene ’0’ og ’1’, hver av disse brukes til forskjellige formål se Figur 2.15 og Tabell 2.11. I eksemplet i Figur 2.15 ligger adressene 4,4,4,5,4,5,5,5 fra bit nr m til m+1. Siden 4 adresserer ’0’ og 5 adresserer ’1’ får en konstanten ’0’, ’0’, ’0’, ’1’, ’0’, ’1’, ’1’, ’1’ eller 17 heksadesimalt i datastrømmen. Dette er måten PUSEK bruker for å lage synkord og for å emulere rammeteller. Andre alternativer til rammeteller som invertering av synkordet i en av rammene eller et ekstra synkord ei ramme kan også brukes. Konstanter kan også brukes til andre formål, som for eksempel å merke formatet med skuddnummer, og tid for generering av formatet. Dessuten brukes konstanter til å fylle ledige tidsluker i formatet.
I tillegg til de 2 adressene beskrevet ovenfor er det 4 til som adresserer ’0’ og ’1’. De plasseres først i rammene og brukes til å lage SS og FP, samt å resette bittelleren, se Tabell 2.11. Dette medfører at alle rammer starter med ’0’ eller ’1’. Det er derfor gunstig å plassere synkordet først i ramma for å benytte de faste bitene til noe fornuftig

Bit 15, 14	Bit 13	Bit 12, 11, ..., 1, 0
Ubrukt	Komprimering	Adresse

Figur 2.16 Oppbygning av 16-bits ord i PUSEKs formatminne. Om bit 13 (Komprimering) er 1 vil det gjeldende ordet gjelde for de 4 neste bitene i telemetriformatet. Om bit 13 er 0 gjelder det bare for én bit i formatet.

Bit #	15	14	...	5	4	3	2	1	0
Innhold	MSB	MSB-1	...	LSB+1	LSB	0	0	0	0

Figur 2.17 Eksempel på 12-bits ord som sendes i et 16-bits ord Bit 15 sendes først, bit 0 sist.

2.4.2 Komprimering

PUSEK er bygd rundt en multiplekser som styres med 13-bits adresser fra formatminnet (se Figur 2.14). Disse adresserer datakanaler og statusbit i enkoderen. For hver bit i datastrømmen blir en ny adresse hentet fram fra formatminnet. Hver lokasjon i formatminnet har 16 bit, og er bygd opp som vist i Figur 2.16. Data i formatminnet kan til en viss grad komprimeres; om bit 13 (kompresjonsbitet) er satt til 1 vil dette bli ekspandert til 4 identiske adresser (men med kompresjonsbitet satt til 0). For å illustrere dette kan vi se på hvordan et 12-bits ord fra en AD kan sendes i et 16-bits ord. La oss bruke notasjonen (C,A) for ord i formatminnet til PUSEK, hvor C er komprimeringsbiten (1 for kompresjon, 0 for ingen kompresjon) og A er en 13-bits adresse for en inngang. La oss videre anta at adressen man bruker for å lese MSB fra en gitt AD-inngang er 42, at inngangen man leser resten av bitene fra har adresse 43 og at adressen man leser ”0” fra er 4. Den ukomprimerte sekvensen i formatminnet blir da:

(0,42) (0,43) (0,43) (0,43) (0,43) (0,43) (0,43) (0,43)
(0,43) (0,43) (0,43) (0,43) (0,4) (0,4) (0,4) (0,4)

mens den komprimerte sekvensen blir:

(0,42) (1,43) (1,43) (0,43) (0,43) (0,43) (1,4)

Begge disse sekvensene med adresser i formatminnet i PUSEK vil gi den samme bitstrømmen med data ut til senderen. For brukeren er den eneste praktiske forskjellen mellom komprimert og ukomprimert formatminne at det tar kortere tid å overføre de komprimerte dataene til PUSEK når den programmeres.

2.4.3 Formatteller

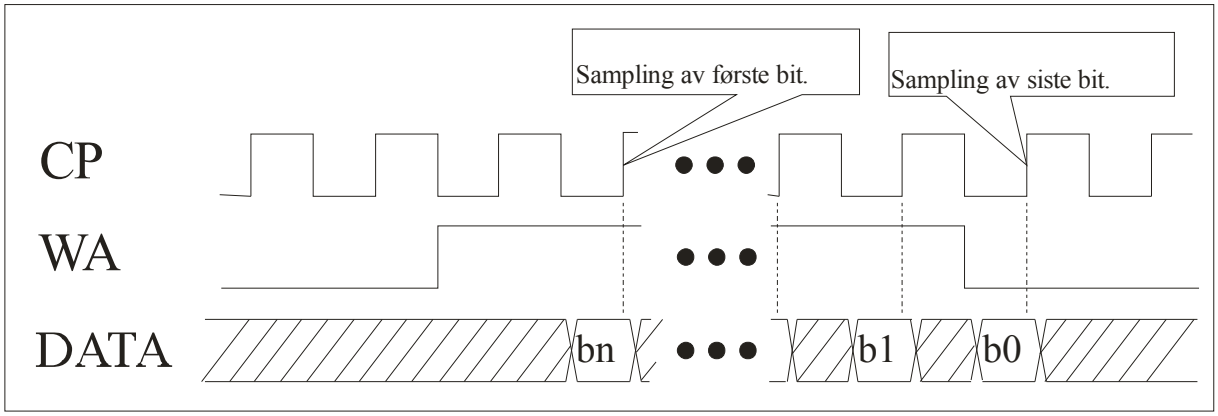
PUSEK har en 24-bits formatteller som øker med én i begynnelsen av hvert format, se Figur 2.15. Hvert enkelt bit fra telleren har sin egen inngang på multiplekseren slik at telleren kan plasseres fritt i formatet, med MSB eller LSB først. Den kan stoppes og resettes ved å sette en spenning mellom FC_off og UMB_return gjennom navlestrengen. Hvis FC_off er på frem til skuddet starter telleren i det raketten starter og navlestrengen trekkes ut.

2.4.4 Digitalt grensesnitt

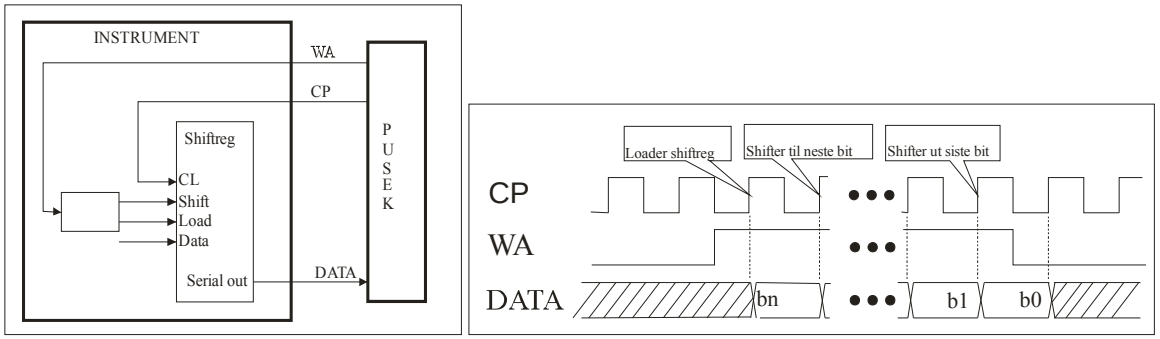
Som illustrert i Figur 2.18 leses dataene av enkoderen på positiv flanke av CP. WA markerer når enkoderen leser data, den er høy i like mange bitintervaller som enkoderen forventer bit. Første

bit leses av enkoderen på andre positive flanke etter at WA går høy. Den siste biten samples på første flanke etter at WA går lav igjen. Figur 2.19 viser et forenklet skjema av et instrument hvor dataene ligger klar på inngangen av skiftregisteret og lastes opp på første positive flanke av CP etter at WA har gått høy. Dataene skiftes deretter ut på de neste positive flankene av CP. Figur 2.20 viser en annen variant med en AD hvor dataene lastes over i skifteregisteret så snart AD-konverteren er klar med en konvertering. *I dette tilfellet er det viktig å huske på at første bit allerede ligger klar på utgangen og at instrumentet må vente til den andre flanken med å skifte ut data.* Dette kan i praksis ordnes med å forsinke datautgangen eller WA et bitintervall, for eksempel med å ha et ekstra bit i skiftregisteret eller ved å forsinke WA med en D-flipflop klokket av CP.

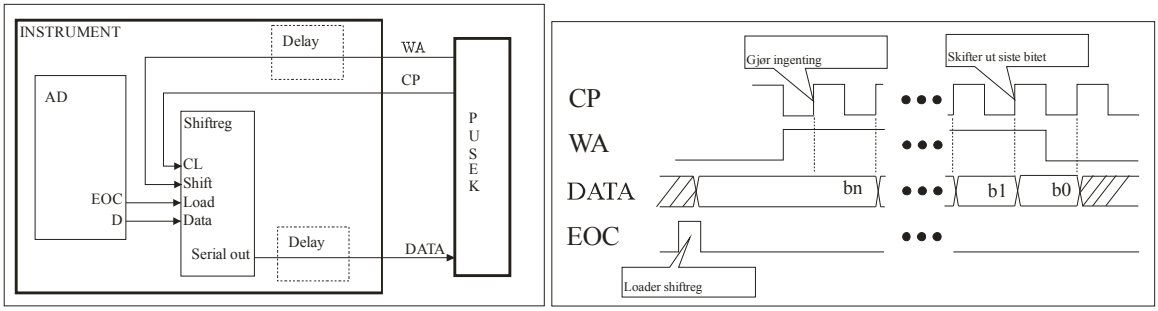
Starten på formatet markeres med en positiv flanke av CP med SS høy. Første bit i formatet samples på *neste* positive flanke av CP, som vist til venstre i Figur 2.21. Starten på rammene markeres tilsvarende med FP (til høyre i Figur 2.21). Det kan være verdt å legge merke til at SS er det samme som WA for det første bitet i ord 0 ramme 0, og tilsvarende for FP men for alle rammene i formatet.



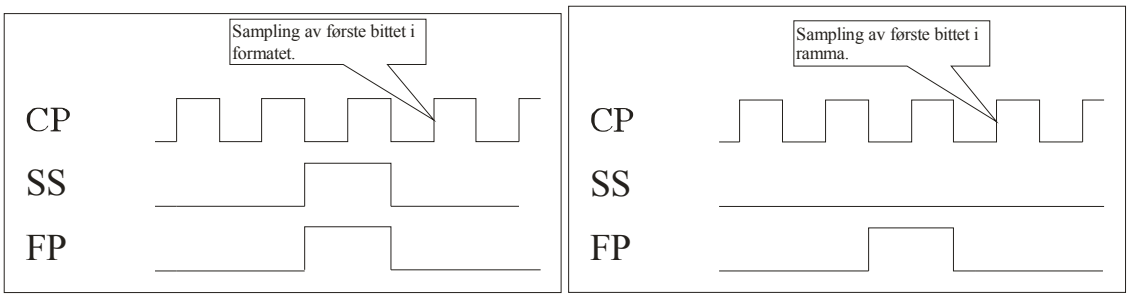
Figur 2.18 Enkoderen leser n+1 bit.



Figur 2.19 Opplasting av skiftregister styrt av WA. Prinsippskisse for instrument (venstre) og timingsdiagram (høyre).



Figur 2.20 Opplasting av skiftregister styrt av AD. Prinsippskisse for instrument (venstre) og timingsdiagram (høyre).



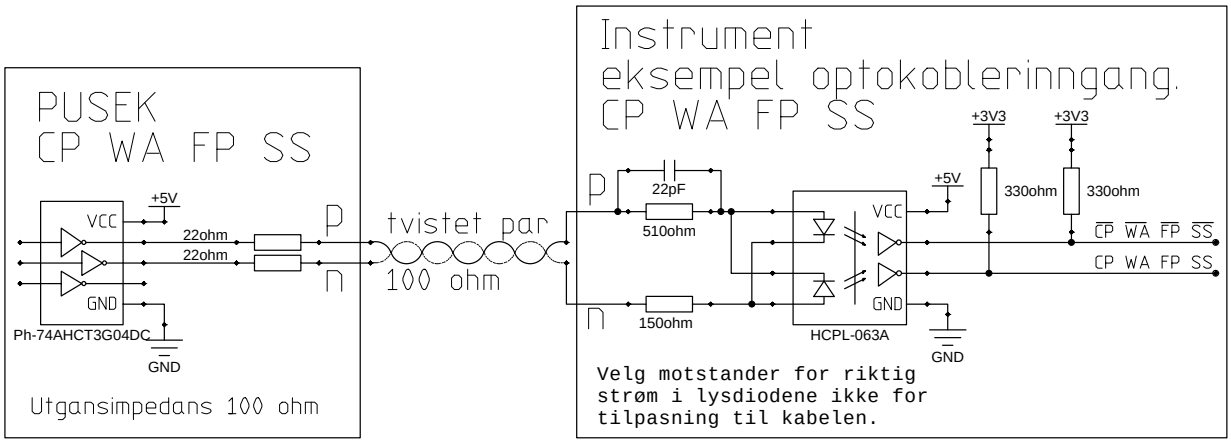
Figur 2.21 Starten av første ramme (venstre) og starten av øvrige rammer (høyre).

2.4.5 Digitale inn- og utganger

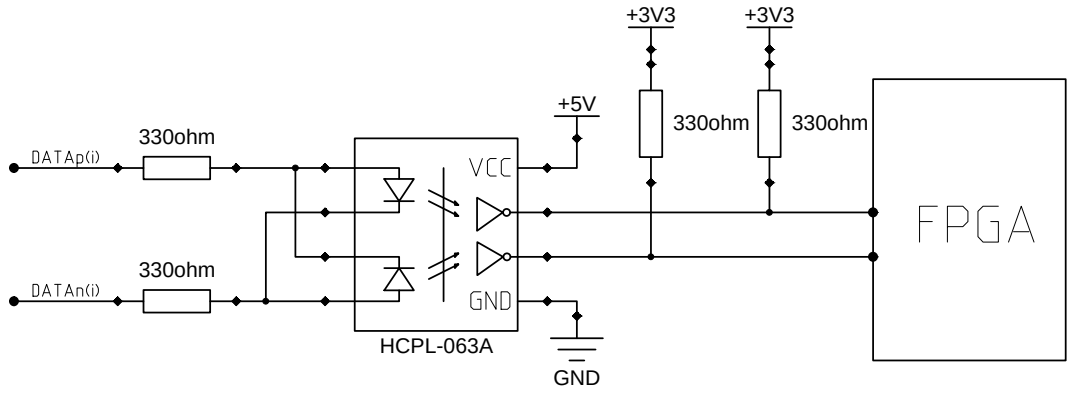
Datainngangene på PUSEK har optokoblere for å begrense problemet med jordsløyfer. For at det skal ha noen effekt må det også brukes optokoblere på inngangene i instrumentet. PUSEK er konstruert for bruk av tvistet signalkabel med en karakteristisk impedans på 100 Ω. Inngangene til PUSEK er beregnet på å drives lavt med 0V og høyt med 5V via en utgangimpedans på 100 Ω. Figur 2.22 viser hvordan utgangsbufrene i PUSEK er konstruert, samt et eksempel på en digital inngang med optokobler i et instrument.

De digitale inngangene på PUSEK er vist i Figur 2.23. Mottakerkretsen i Figur 2.22 er en raskere variant av kretsen i Figur 2.23. Summen av motstandene er i begge mottakerkretsene 660Ω og regulerer den statiske strømmen i optokobleren. I Figur 2.22 er det en kondensator i parallell med en av motstandene for å få en større strøm ved påslag, og dermed en kortere forsinkelse gjennom optokobleren. Motstanden på 150 Ω setter den maksimale strømmen gjennom lysdioden for å unngå overbelastning.

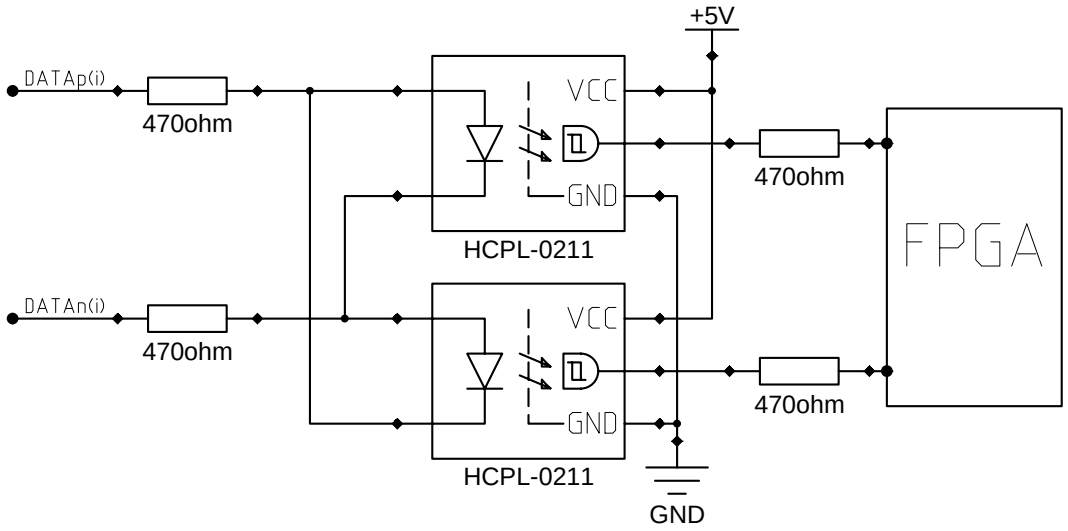
I tillegg til den vanlige datainngangen vist i Figur 2.23 har PUSEK to innganger som er litt annerledes (Figur 2.24). Disse inngangene brukes for å garantere kompatibilitet med instrumenter designet for en tidligere enkoder. Kretsen i Figur 2.24 er noe langsommere enn kretsen i Figur 2.23. I kretsene i Figur 2.22 og Figur 2.23 er utgangen av optokoblerene trukket opp til 3,3V med en motstand på 330 Ω fordi FPGA'en har 3,3V io buffere. Kretsen i Figur 2.24 har seriemotstander på 470 Ω for å begrense strømmen i overspenningsbeskyttelsen internt i FPGAen.



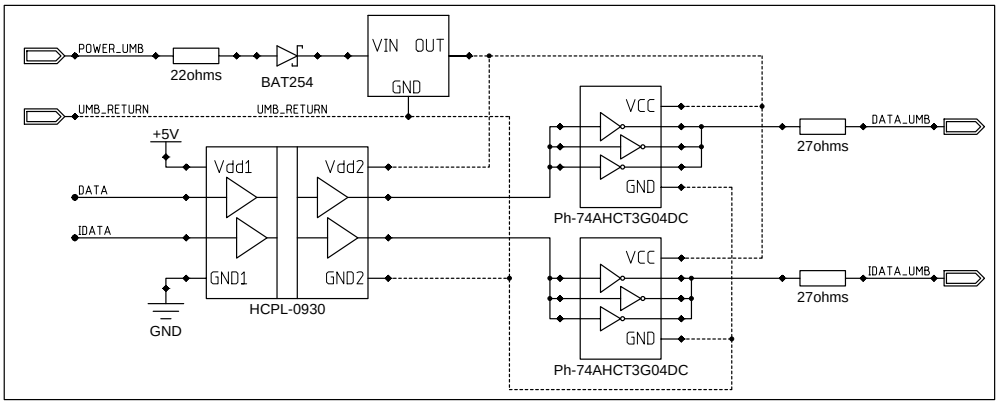
Figur 2.22 Digitalt grensesnitt for kontrollsignaler til instrumentet. Utgang på PUSEK til venstre og eksempel på inngang på instrumentet til høyre.Utgangsinpedans for PUSEK er 100 Ω for å matche transmisjonslinja.



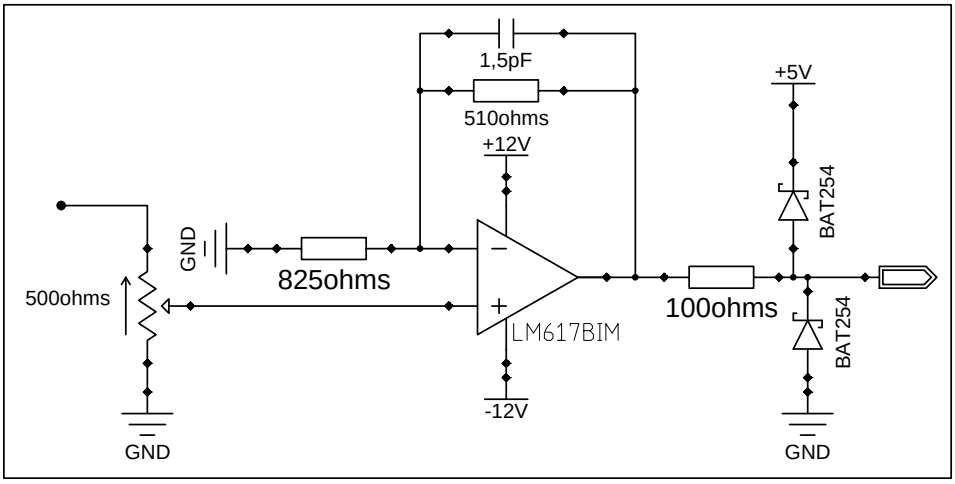
Figur 2.23 Vanlig datainngang på PUSEK.



Figur 2.24 Datainngang PUSEK for kompatibilitet med gamle instrumenter 2 stk.



Figur 2.25 Differensiell utgang for data til navlestrengen.



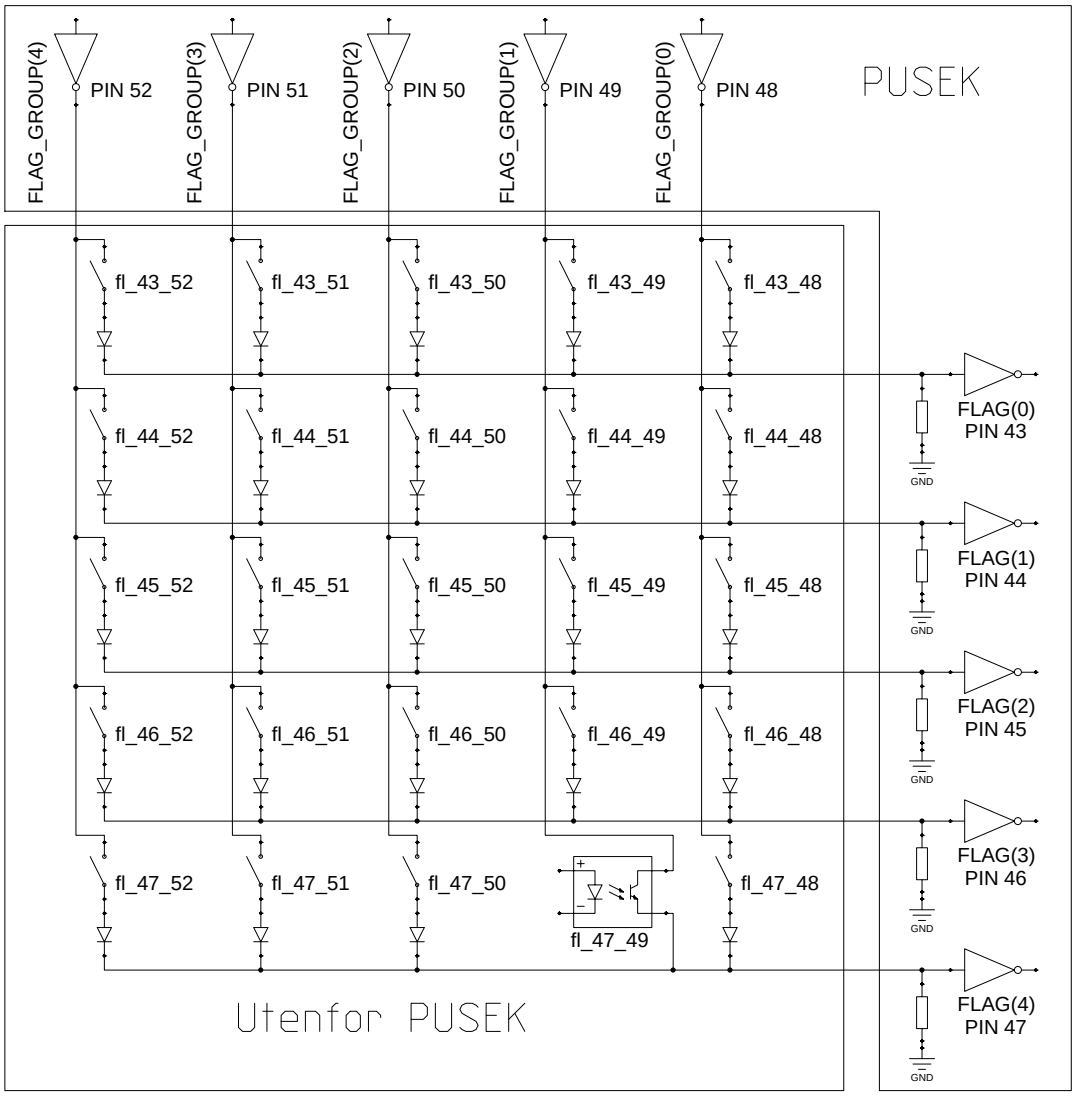
Figur 2.26 Enkeltendet utgang for data til senderen.

2.4.6 Datautganger

PUSEK har to datautganger – én differensiell beregnet på navlestrengen (Figur 2.25), og én 100 Ω enkeltendet med variabel amplitude beregnet på senderen (Figur 2.26). Den differensielle utgangen er galvanisk skilt fra den interne logikken og får spenningen sin fra navlestrengen, noe som garanterer at den ikke støyer på navlestrengspluggen under flukten. Amplituden stilles med et potensiometer som er tilgjengelig gjennom et lite hull på siden av ramma, like ved 15 pins pluggen. Posisjonen til hullet indikeres med en pil i Figur 2.13 på side 21.

2.4.7 Flaggmonitorer

PUSEK har mulighet for å avlese inntil 25 mikrobrytere plassert i en matrise. Hver enkelt av mikrobryterne må kobles i serie med hver sin småsignaldiode, som vist i Figur 2.27. En bryter med diode kan erstattes av en optokobler med en transistorutgang (se figuren for hvordan denne skal kobles). Logikken går på 3,3V. Når mikrobrytermatrisen skal avleses vil en av flag_group-utgangene gå høy, og så vil de 5 tilhørende mikrobryterne bli avlest. Sekvensen har en varighet av 160 klokkepulser (CP), noe som gir en usikkerhet på like lang tid for når mikrobryteren ble samplet.

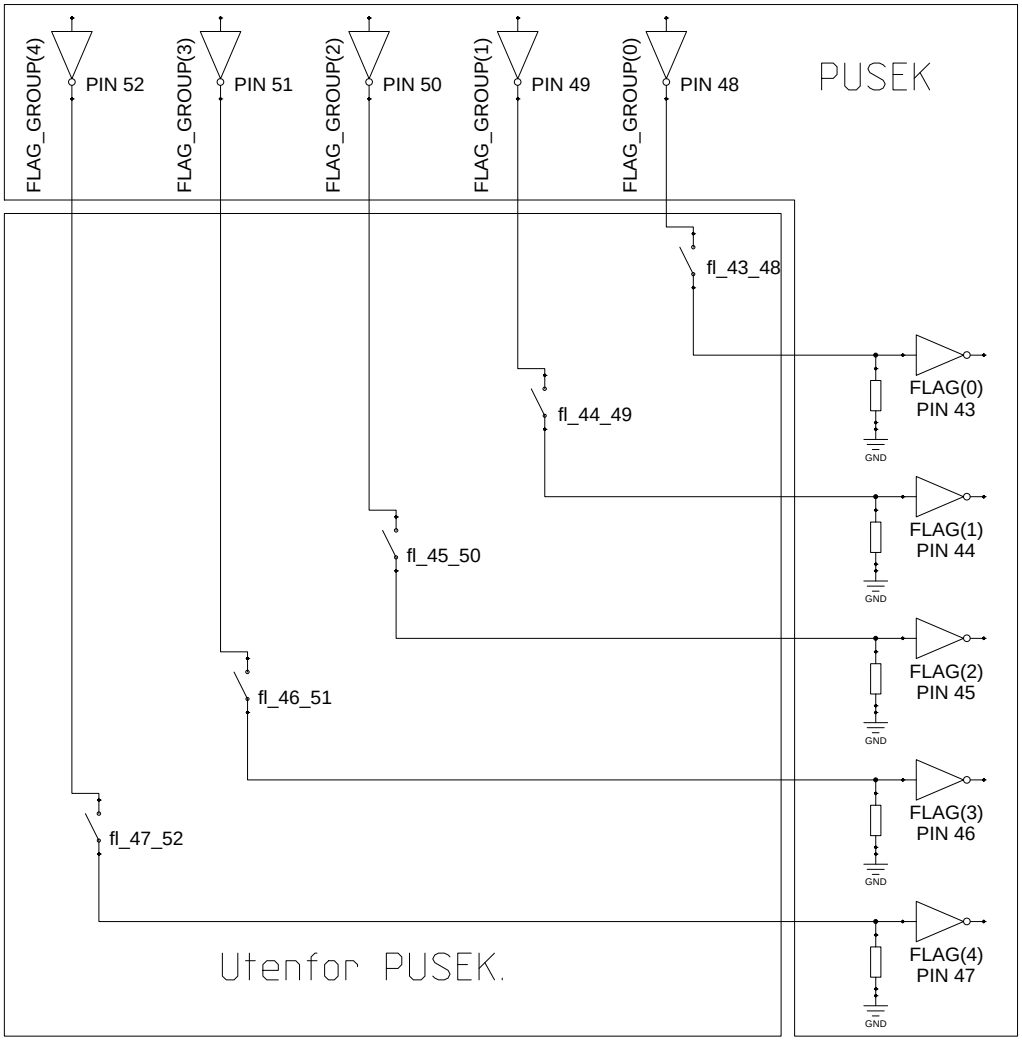


Figur 2.27 Mikrobrytermatrise. En småsignaldiode må kobles i serie med hver mikrobryter. Et alternativ til mikrobrytere er optokoblere med transistorutgang, som antydet i nederste rad (fl_47_49).

Når det er bruk for færre mikrobrytere kan i noen tilfeller noen av diodene sløyfes. Et eksempel på dette er illustrert i Figur 2.28, hvor det er 5 mikrobrytere og ingen dioder.

2.4.8 Signaler og plugglister for PUSEK

Tabell 2.12 gir en oversikt over hvilke typer signaler som finnes på den store pluggen (62 pinner) på fronten av PUSEK, mens Tabell 2.13 gir en pluggliste for denne. Tilsvarende gir Tabell 2.14 oversikt over signalene på den lille pluggen (15 pinner) mens Tabell 2.15 gir plugglista.



Figur 2.28 Forenklet matrise med 5 mikrobrytere. Siden hver mikrobryter er alene på hver sin rad kan diodene i serie med bryterne sløyfes.

Pinnenavn	**Funksjon**
DATAp (MM)	Differensiell datainngang nr. MM (aktiv høy).
DATAn (MM)	Differensiell datainngang nr. MM (aktiv lav).
WAp (MM)	Differensiell Word Address for inngang MM (aktiv høy). Styrer datautlesning.
WAn (MM)	Differensiell Word Address for inngang MM (aktiv lav). Styrer datautlesning.
CPp	Differensiell Systemklokke (aktiv høy).
CPn	Differensiell Systemklokke (aktiv lav).
FPp	Differensiell FramePulse (aktiv høy). Start av ramme.
FPn	Differensiell FramePulse (aktiv lav). Start av ramme.
SSp	Differensiell SubSync (aktiv høy). Start av format.
SSn	Differensiell SubSync (aktiv lav). Start av format.
flag(Y)	Inngang nr. Y fra mikrobrytermatrise.
flag_group(X)	Utgang nr. X til mikrobrytermatrise.

Tabell 2.12 Tilgjengelige signaler på 62 pins pluggen.

#	Pinnenavn	#	Pinnenavn	#	Pinnenavn
1	DATAp(6) *	**22**	DATAn(6) *	**43**	flag(0)
2	DATAp(7) *	**23**	DATAn(7) *	**44**	flag(1)
3	DATAp(8)	**24**	DATAn(8)	**45**	flag(2)
4	DATAp(9)	**25**	DATAn(9)	**46**	flag(3)
5	DATAp(10)	**26**	DATAn(10)	**47**	flag(4)
6	DATAp(11)	**27**	DATAn(11)	**48**	flag_group(0)
7	DATAp(12)	**28**	DATAn(12)	**49**	flag_group(1)
8	DATAp(13)	**29**	DATAn(13)	**50**	flag_group(2)
9	DATAp(14)	**30**	DATAn(14)	**51**	flag_group(3)
10	DATAp(15)	**31**	DATAn(15)	**52**	flag_group(4)
11	WAp (6)	**32**	WAn (6)	**53**	SSp(6)
12	WAp (7)	**33**	WAn (7)	**54**	SSn(6)
13	WAp (8)	**34**	WAn (8)	**55**	CPp(6)
14	WAp (9)	**35**	WAn (9)	**56**	CPn(6)
15	WAp (10)	**36**	WAn (10)	**57**	CPp(7)
16	WAp (11)	**37**	WAn (11)	**58**	CPn(7)
17	WAp (12)	**38**	WAn (12)	**59**	FPp(7)
18	WAp (13)	**39**	WAn (13)	**60**	FPn(7)
19	WAp (14)	**40**	WAn (14)	**61**	SSp(7)
20	WAp (15)	**41**	WAn (15)	**62**	SSn(7)
21	FPp(6)	**42**	FPn(6)		

*Tabell 2.13 Pluggliste for 62-pins high density hunplugg på fronten av PUSEK-modulen. Datainngangene merket med * (nr. 6 og 7) er som i Figur 2.24. Resten av inngangene er som i Figur 2.23. Nummereringen av datainngangene (6-15) reflekterer fysisk adresse i PUSEK, ikke antall innganger.*

Pinnenavn	Programmeringsfunksjon (P) / UMB-funksjon (UMB) / Senderfunksjon (TX)
28V	(P) Power input ved programmering.
28V return	(P) Power return.
Int 2,5V	(P) Brukes av programkabelen fra Xilinx.
Internal return	(P) Brukes av programkabelen fra Xilinx.
TCK	(P) Brukes av programkabelen fra Xilinx.
TDI	(P) Brukes av programkabelen fra Xilinx.
TDO	(P) Brukes av programkabelen fra Xilinx.
TMS	(P) Brukes av programkabelen fra Xilinx.
UMB power (12V)	(UMB) Power til funksjoner som kun brukes før skudd. 10V til 20V.
UMB return	(UMB) Retur for UMB power, Format Count off og DATA umb.
Format count off	(UMB) En positiv spenning på 10V til 30V i forhold til UMB return stopper og resetter formattelleren. Galvanisk adskilt fra resten av PUSEK.
DATA umb	(UMB) Telemetridata til differensiell datautgang. Galvanisk adskilt fra resten av PUSEK. Avhengig av strømforsyning på UMB power for å fungere. Spenningen svinger i forhold til UMB return.
IDATA umb	
DATA TX	(TX) Telemetridata til sender.
DATA TX return	(TX) Retur for telemetridata til sender.

Tabell 2.14 Tilgjengelige signaler på 15 pins pluggen på fronten av PUSEK-modulen.

#	Pinnenavn	#	Pinnenavn	#	Pinnenavn
1	DATA umb	**6**	UMB return	**11**	TMS
2	IDATA umb	**7**	UMB power (12V)	**12**	Internal return
3	Int 2.5V	**8**	Format count off	**13**	TDO
4	28V return	**9**	TDI	**14**	DATA TX return
5	28V	**10**	TCK	**15**	DATA TX

Tabell 2.15 Pluggliste for 15-pins high density hunplugg på fronten av PUSEK-modulen

2.5 SLAVEPUSEK

SLAVEPUSEK er en telemetrienkoder som bruker en oppdatert utgave av kretskortet til PUSEK, og som har en modifisert utgave av firmwaren i PUSEK. Disse endringene overlater noe av styringen til PUSEK. Vi vil i det følgende bare omtale forskjellene mellom PUSEK og SLAVEPUSEK. Om ikke annet er nevnt, vil det som står omtalt om PUSEK også gjelde for SLAVEPUSEK.

Formatet i SLAVEPUSEK må ha en lengde som er et helt antall ganger formatlengden til PUSEK, målt i bit. Bortsett fra denne restriksjonen står en fritt til å velge ordlengde, antall ord per ramme og antall rammer uavhengig av PUSEK. SLAVEPUSEK og PUSEK har samme bitrate og enkoderne er synkronisert slik at hver SS fra slaven har en samtidig SS fra PUSEK. Figur 2.29

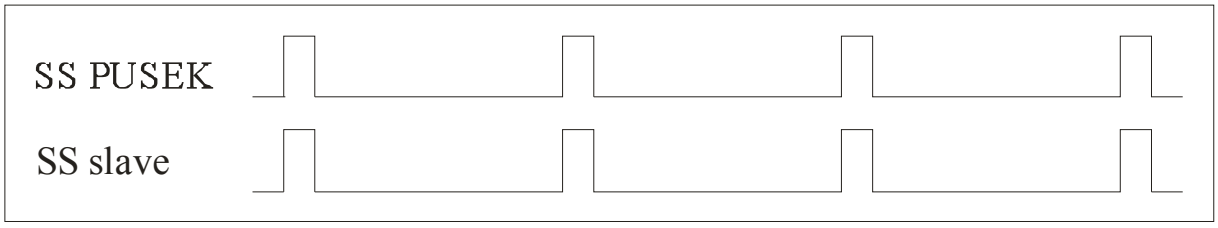
viser SS med like lange formater og Figur 2.30 viser SS når slaven har 3 ganger lengre format enn PUSEK.

SLAVEPUSEK leser data fra bussen, men det er PUSEK som bestemmer hva som leses. Dette kan utnyttes til å få gratis sikkerhetskopier av data som PUSEK leser fra AMON, TIM eller POW. Programmeres slaven til å lese fra bussen samtidig med at PUSEK gjør det vil begge enkoderne lese data bestemt av PUSEK og en får en kopi av PUSEK-data i datastrømmen til slaven. For å få slaven til å lese fra bussen kan en velge en tilfeldig adresse fra AMON, TIM eller POW. Et bedre alternativ kan være å ha en felles fil med dataene begge enkoderne skal lese, og så inkludere fila i begge formatbeskrivelsene. Dette vil dokumentere hva som er likt i formatene og gjøre det lettere å holde oversikten.

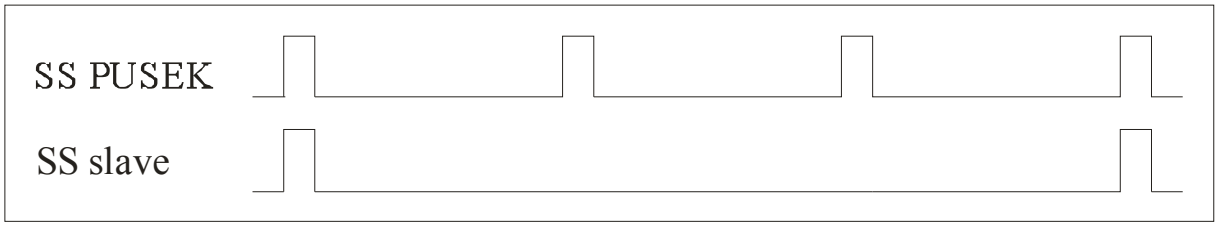
Formattelleren startes og stoppes av PUSEK, og vil telle synkront med formattelleren i PUSEK dersom formatene er like lange. Styresignalene SS, FP og CP kommer så likt at en kan ta signaler fra den ene enkoderen for å styre instrumenter tilhørende den andre enkoderen, men med et par forbehold:

- CP fra den ene enkoderen kan alltid brukes til instrumenter tilhørende den andre.
- FP fra den ene enkoderen kan brukes til instrumenter tilhørende den andre når rammene i begge formatene er like lange.
- SS fra den ene enkoderen kan brukes til instrumenter tilhørende den andre når begge formatene er like lange.

SLAVEPUSEK har ikke egen DC/DC-omformer, oscillator eller trykksensor. Plugglistene til SLAVEPUSEK og PUSEK er identiske. Mikrobrytere leses som i PUSEK, men med 5V signaler som i TIM. De digitale datainngangene på SLAVEPUSEK er som vist i Figur 2.22, og er altså identisk med inngangen som blir anbefalt for bruk i instrumenter.



Figur 2.29 Formatene i PUSEK og SLAVEPUSEK er like lange.



Figur 2.30 SLAVEPUSEK har et format som er tre ganger lengre enn formatet til PUSEK.

2.6 POW (Power)

Power-modulen har følgende funksjoner:

- Av/på batteri med holdefunksjon
- På/av instrumentgrupper (3 grupper med hhv. 4, 5 og 6 utganger) og sender
- Diodekobling mellom ekstern power og batteri
- Batterilading
- Spenningmåling
- Strømmåling
- Temperaturmåling

Som alle de andre modulene har POW inn- og utganger på en 15 pin high density DSUB hun (HD15), en 62 pin high density DSUB hun (HD62), i tillegg til en 64 pin intern Europabuss. Figur 2.31 viser et bilde av Powermodulen. Funksjonell virkemåte vises i blokkskjemaet i Figur 2.32. Merk at bryterfunksjonene er tegnet som mekaniske reléer for enklere lesbarhet. I virkeligheten er det bare ladereléet som er faktisk er implementert som et relé; det er brukt optokoblede halvlederbrytere overalt ellers.

Powermodulen har ingen potmeter eller parametre som skal justeres eller kalibreres av bruker. Firmwaren ligger i et flashminne som kan oppgraderes med JTAG på samme måte som de andre modulene i STAPPE.

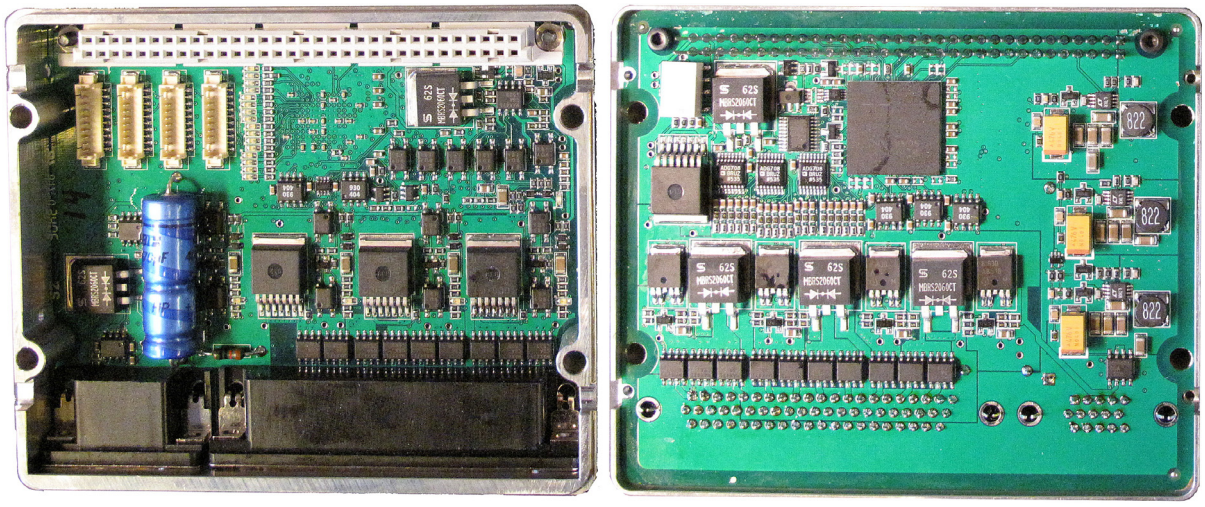
2.6.1 Batteritilkobling og batterilading

Batteripakken bør være sammensatt av 23 NiCd/NiMH-celler. Dette gir en nominell spenning på 23·1.2V=27.6V. Maksimal ladespenning for en NiMH-celle (f.eks. elfa 69-314-55) er 1.53V ved 1C og 1.5V ved 0.5C[†]. Dette gir en batterispenning på hhv 35.2V og 34.5V. Til sammenligning gir 24 celler en nominell spenning på 28.8V og ladespenninger på 36.7V og 36.0V ved ladestrømmer på hhv 1C og 0.5C. Slike ladespenninger er for høye og kan skade elektronikken i STAPPE. Ved ladestrøm over 0.1C må batteriet overvåkes. Slik overvåking er det ikke lagt opp til i powermodulen, så vi anbefaler en maksimal ladestrøm på 0.1C. En så lav ladestrøm kan vanligvis stå på kontinuerlig uten å ødelegge batteriet, men retningslinjer fra batteriprodusent/-leverandør må overholdes med tanke på blant annet kjøling, maks ladetid, etc.
Batteriet tilkobles Powermodulen som vist i Figur 2.33: +28V på batteriet tilkobles pinne 41 og pinne 42, med retur på pinne 32 og 33. Det anbefales på det sterkeste å føre minst to separate ledninger fram til hver batteripol, som vist i figuren. Det er flere grunner til dette, blant annet at

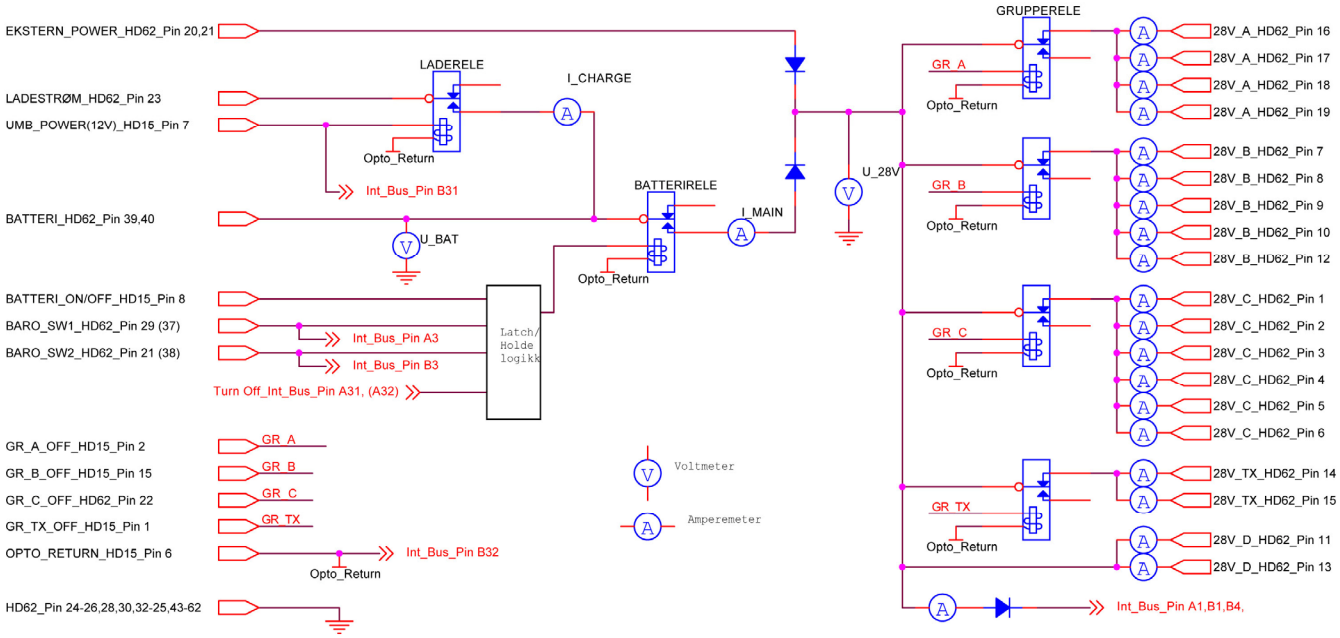
- man ønsker å redusere strømmen gjennom hver pinne i pluggen for å forhindre store strømmer på et lite sted på kortet
- man ønsker å redusere spenningstapet over tilførselsledningene til batteriet
- ved å bruke flere ledningspar innfører man en viss redundans

For ytterligere parallellkobling av forsyningsstrømmen fra batteriet bør +28V fra batteriet også kobles til pinne 39 og 40, med retur på pinne 26 og 30.

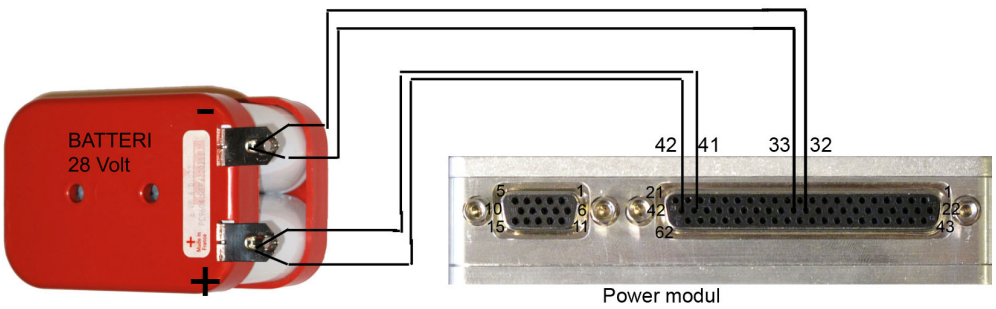
[†] C er et mål på strømstyrke i forhold til batteriets kapasitet. For et batteri med kapasitet på 1800mAh betyr 1C en strømstyrke på 1800mA, 0.1C en strømstyrke på 180mA, osv.



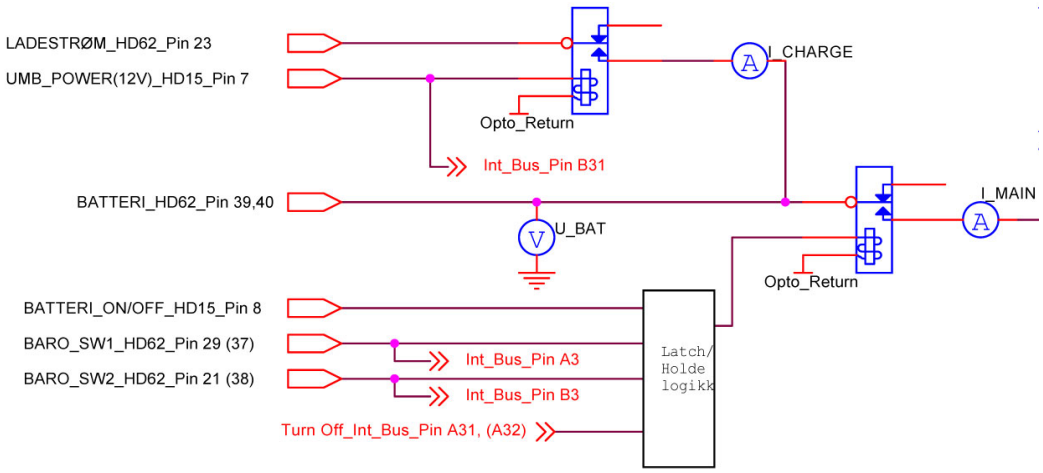
Figur 2.31 Powermodul (POW) overside (til venstre) og underside (til høyre)



Figur 2.32 Blokkskjema for POW. Bryterfunksjoner er tegnet inn som mekaniske reléer, men bortsett fra ladereléet består egentlig alle av optokoblete halvlederbrytere.



Figur 2.33 Tilkobling av batteri til Power-modulen. +28V kobles til pinne 41 og 42, retur til pinne 32 og 33. Ved behov for ytterligere parallellkobling kan +28V kobles til pinne 39 og 40, mens retur kobles til pinne 26 og 30.



Figur 2.34 Styring av ladestrøm for batteriet

Inne på Powerkortet blir batterispenningen målt over en høyohmig spenningsdeler (vist som U_BAT i Figur 2.34), og kan leses fra PUSEK ved hjelp av `adr(POW_U_MON_BAT)`. Batterispenningen U_{BAT} kan beregnes fra den målte 12-bitverdien slik:

$$U_{\mathrm{BAT}}[\mathrm{V}] = (\text{12-bits verdi}) \times \frac{40}{4095}.$$

På samme måte kan strømmen fra batteriet (I_MAIN i Figur 2.34), måles ved hjelp av strømmonitoren innebygget i halvlederbryteren. Denne leses ved hjelp av `adr(POW_I_MAIN)`. I_{MAIN} kan uttrykes i mA ved hjelp av følgende konvertering:

$$I_{\mathrm{MAIN}}[\mathrm{mA}] = (\text{12-bits verdi}) \times \frac{20000}{4095}.$$

Merk at målingen av I_{MAIN} er unøyaktig og ikke helt linær, men den gir allikevel en god indikasjon på hvor mye strøm som trekkes fra batteriet.

Ladestrøm for batteriet tilkobles pinne 23 på 62-pinners pluggen, med retur på pinne 24, se Figur 2.34. Pinne 23 og 24 er beregnet å kobles direkte til navlestrengen (umbilical; UMB). Ladestrømmen går via et laderelé som må være aktivisert, via strømmonitor (POW_I_CHARGE) direkte ut på batteriet. Reléet aktiviseres når pinne 7 på 15-pinnerspluggen (UMB_POWER) har +12V. Hovedfunksjonen til ladereléet er å hindre at det står batterispenning på pinne 23 som går videre til umbilical. Dette er spesielt viktig når nyttelasten skal berges fra sjø (står det spenning på pinne i sjø, vil pluggen bli ødelagt og det kan bli lekkasje). Ladestrømmen kan måles med strømmonitoren I_CHARGE, som adresseres ved hjelp av `adr(POW_MON_I_CHARGE)`. I_{MAIN} kan regnes om til mA ved hjelp av følgende uttrykk:

$$I_{\mathrm{MAIN}}[\mathrm{mA}] = (\text{12-bits verdi}) \times \frac{3278}{4095}.$$

2.6.1.1 Batteri På/Av

Batterireléet har en holde-/latchfunksjon. Det er dermed nødvendig med spenning på pinne 8 i bare et kort øyeblikk og tilstanden blir husket. Følgende regler gjelder:

- +12V på pinne 8 HD15 setter batterireléet på og STAPPE går på batteriforsyning.
- -12V på pinne 8 HD15 setter batterireléet av og STAPPE slås av

- +28V på baroinngangene setter også batterireléet på. Dette er en sikkerhetsfunksjon som er beskrevet nærmere i 2.6.2.
- +5V fra timer på intern bus setter batterireléet av. Denne funksjonen brukes i nyttelaster som skal berges.

2.6.1.2 Ekstern strømforsyning

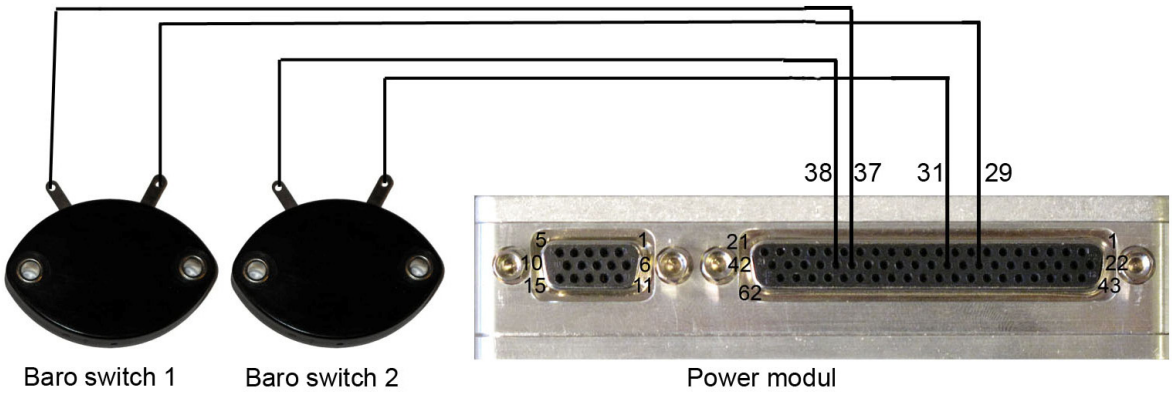
Ekstern strømforsyning kommer fra bakkeutstyret (GSE) via navlestengen (UMB), og tilkobles følgende pinner i powermodulen:

- +28V tilkobles pinne 20 og 21 på HD62
- +28V retur tilkobles pinne 34 og 35 på HD62

Ekstern strømforsyning og batteriforsyning blir diodekoblet sammen som vist i Figur 2.32. STAPPE vil dra strøm fra strømforsyningen som har høyest spenning.

2.6.2 Tilkobling av barometriske brytere

En barometrisk bryter (eller barobryter) er trykkavhengig bryter som slutter kontakt ved lavt trykk. Disse finnes i forskjellige utførelser som trigger/aktiviseres ved ulike høyder (for eksempel ved 3km, 10km, 30km). Barobrytere brukes som sikkerhetsfunksjon for å sikre at nyttelasten alltid vil være slått på over en høyde bestemt av barobryteren. Man bør velge en så lav høyde som mulig, for eksempel 3 km. For redundans er det er mulig å tilkoble to barobrytere. Begge blir monitorert individuelt og legges ut i datastrømmen som flaggbit. Barobryterne adresseres med `adr(TIM_BARO_1)` og `adr(TIM_BARO_2)`.
I powermodulen blir dette signalet brukt som en "alltid på"-funksjon. Barosignalet videreføres til internbussen (pinne A3 og B3) og blir brukt av andre moduler også, som for eksempel som signal til opplading av fyringskondensatorer i timermodulen.
Begge barobryterne kobles til 62-pinnerspluggen. Barobryter 1 kobles mellom pinne 29 og pinne 37, mens barobryter 2 kobles mellom pinne 31 og pinne 38. Se også Figur 2.35.



Figur 2.35 Tilkobling av barometriske brytere på Powermodulen.

2.6.3 Tilkobling av instrumenter

Powermodulen har 19 stk strøm-monitorerte utganger i tillegg til en intern monitor som måler strømforbruket til STAPPE. Disse 19 utgangene er delt opp i 4 grupper/banker, A (4stk), B

(5stk), C (6stk), TX (2stk) og D (2stk). Alle gruppene bortsett fra D kan slås av. Når det er spenning på ekstern strømforsyning eller fra batteri vil utgangene D alltid være på. Da vil intern bus også ha power. Gruppereléene blir kontrollert av optokoblede styringssignaler. Uten tilkoblede styringssignal (GR_x) vil gruppene alltid være på. Når styringsinngangen har +12V vil tilhørende utganger være skrudd av. Se Tabell 2.16 for en oversikt over pinnenummer for tilkobling av instrumentene og Figur 2.36 for prinsippskisse over utgangene.

Strøm-monitoreringskretsene som er brukt er MAX471 fra Maxim. Det er én slik for hver utgang og disse klarer maks 3A. MAX471 vil fungere som en sikring og brenne over ved eventuell kortslutning/overbelastning.

For programmering brukes `adr(POW_I_MON_g#_PIN_xx) (se pow_std.inc)` for å adressere pinne `xx` på 62-pinnerspluggen, mens strømmen til intern bus får man med `adr(POW_I_MON_ENC)`. De målte strømmene kan regnes om til mA ved hjelp av følgende uttrykk:

$$I_{\mathrm{PINxx,\,bus}}[\mathrm{mA}] = (\text{12-bits verdi}) \times \frac{3278}{4095}$$

hvor I_{PINxx} er strømmen ut til pinne `xx` og I_{bus} er strømmen til internbussen.

Funksjon \ Gruppe	**A**	**B**	**C**	**D**	**TX**
Styring (+12V = av)	2 (HD15)	15 (HD15)	22	-	1 (HD15)
Styring retur	6 (HD15)	6 (HD15)	6 (HD15)	-	6 (HD15)
Power (+28V)	16, 17, 18, 19	7, 8, 9, 10, 12	1, 2, 3, 4, 5, 6	11, 13	14, 15
Power retur	58, 59, 60, 61	49, 50, 51, 52, 54	43, 44, 45, 46, 47, 48	53, 55	56, 57

Tabell 2.16 Pinnenummer for tilkobling av instrumenter til powermodulen. Alle pinnenummer er for 62-pinnerspluggen (HD62), bortsett fra pinner merket med (HD15), som er koblet til 15-pinnerspluggen. Gruppe TX er for tilbkobling av sender(e).

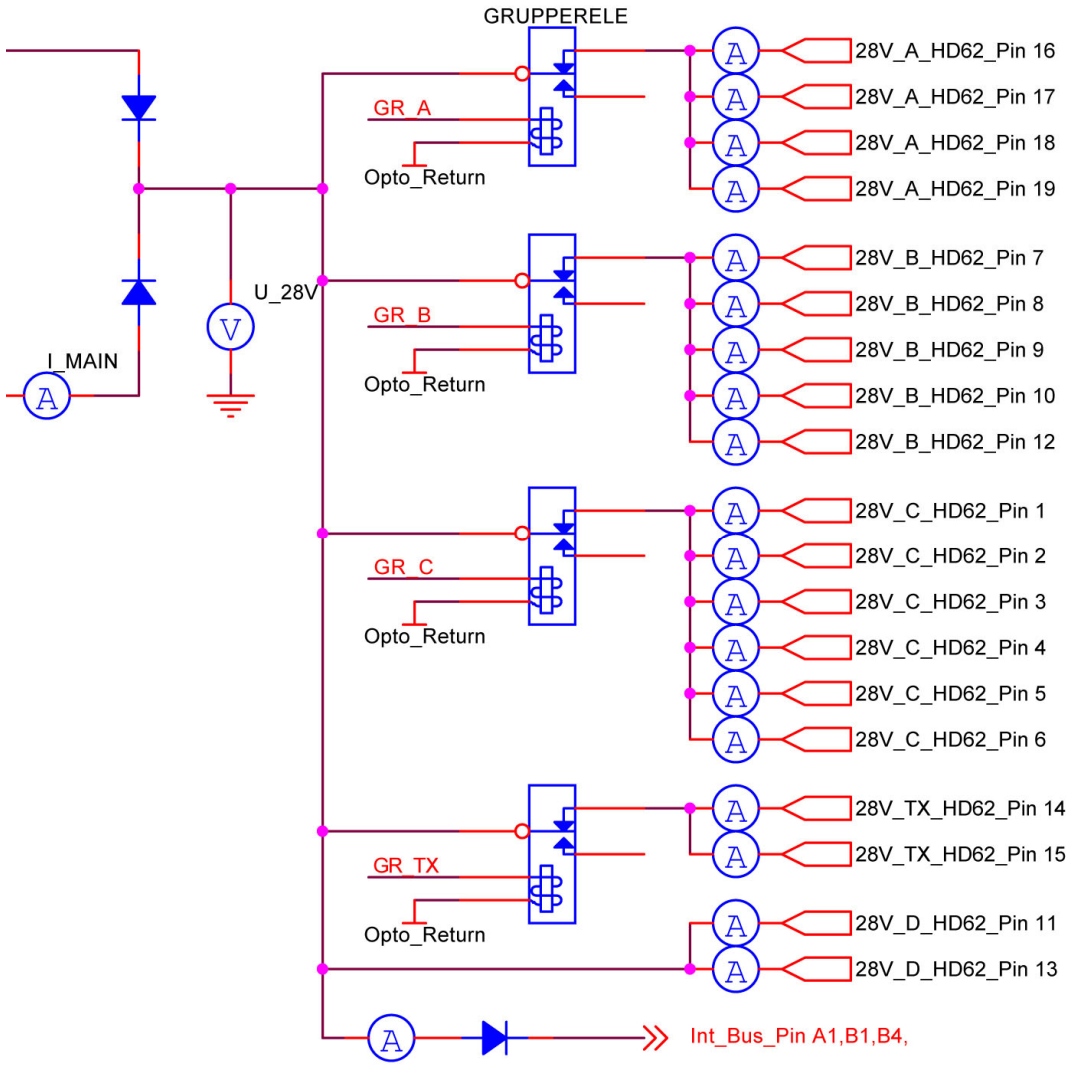
2.6.4 Temperatursensor

Powermodulen har en intern tempersaturensor av typen LM45B som måler i området -50 til +150°C. Temperatursensoren adresseres med `adr(POW_TEMPERATURE)`. Måledata fra temperatursensoren på powerkortet kan regnes om til Celsius ved hjelp av følgende uttrykk:

$$T_{\mathrm{POW}}[^{\circ}\mathrm{C}] = (\text{12-bits verdi}) \times \frac{150}{4095}.$$

2.6.5 Plugglister

Oversikt over signaler og plugglister for Powermodulen finnes i henholdsvis Tabell 2.17 og Tabell 2.18 for 62-pinnerspluggen og i Tabell 2.19 og Tabell 2.20 for 15-pinnerspluggen.



Figur 2.36 Prinsippskisse for strømmonitorerte utganger på powermodulen

Pinnenavn	**Funksjon**
28V XY power	+28V power output, gruppe X, utgang Y. X = A, B, C, D, TX.
XY power return	Power output return, gruppe X, utgang Y. X = A, B, C, D, TX.
28V UMB power	+28V power input fra umbilical
UMB power return	Power input return fra umbilical
BAT+	Tilkobling til batteri, positiv pol
BAT-	Tilkobling til batteri, negativ pol
Charge input	Ladestrøm for batteri fra umbilical (max 3A)
Charge input return	Ladestrøm for batteri fra umbilical (max 3A) return
Baro N	Tilkobling av barometrisk bryter N. N = 1, 2
Baro N 28V	Tilkobling av barometrisk bryter N mot batteri (+28V). N = 1, 2
UMB Exp 3 off	Fra umbilical. Slå av instrumentgruppe C når inngang er +12V.

Tabell 2.17 Tilgjengelige signaler på 62-pins pluggen på fronten av POW-modulen

#	Pinnenavn	#	Pinnenavn	#	Pinnenavn
1	28V C5 power	**22**	UMB Exp 3 off	**43**	C5 power return
2	28V C4 power	**23**	Charge input	**44**	C4 power return
3	28V C1 power	**24**	Charge input return	**45**	C1 power return
4	28V C6 power	**25**	Return	**46**	C6 power return
5	28V C3 power	**26**	BAT-	**47**	C3 power return
6	28V C2 power	**27**	28V UMB power	**48**	C2 power return
7	28V B5 power	**28**	UMB power return	**49**	B5 power return
8	28V B4 power	**29**	Baro 1	**50**	B4 power return
9	28V B2 power	**30**	BAT-	**51**	B2 power return
10	28V B1 power	**31**	Baro 2	**52**	B1 power return
11	28V D1 power	**32**	BAT-	**53**	D1 power return
12	28V B3 power	**33**	BAT-	**54**	B3 power return
13	28V D2 power	**34**	UMB power return	**55**	D2 power return
14	28V TX1 power	**35**	UMB power return	**56**	TX1 power return
15	28V TX2 power	**36**	28V UMB power	**57**	TX2 power return
16	28V A2 power	**37**	Baro 1 28V	**58**	A2 power return
17	28V A4 power	**38**	Baro 2 28V	**59**	A4 power return
18	28V A1 power	**39**	BAT+	**60**	A1 power return
19	28V A3 power	**40**	BAT+	**61**	A3 power return
20	28V UMB power	**41**	BAT+	**62**	UMB power return
21	28V UMB power	**42**	BAT+		

Tabell 2.18 Pluggliste for 62-pins high density hunplugg på fronten av POW-modulen

Pinnenavn	Programmeringsfunksjon (P) / UMB-funksjon (UMB)
28V	(P) Power input ved programmering.
28V return	(P) Power return.
Int 2,5V	(P) Brukes av programkabelen fra Xilinx.
Internal return	(P) Brukes av programkabelen fra Xilinx.
TCK	(P) Brukes av programkabelen fra Xilinx.
TDI	(P) Brukes av programkabelen fra Xilinx.
TDO	(P) Brukes av programkabelen fra Xilinx.
TMS	(P) Brukes av programkabelen fra Xilinx.
UMB power (12V)	(UMB) Power til funksjoner som kun brukes før skudd. 10V til 20V.
UMB return	(UMB) Retur for UMB power, Format Count off og DATA umb.
UMB Int on/off	(UMB) +12V skrur på internt power, -12V skrur av internt power.
UMB Tx on/off	(UMB) Fra umbilical. Slå av Tx når inngang er +12V.
UMB Exp1 off	(UMB) Fra umbilical. Slå av instrumentgruppe A når inngang er +12V.
UMB Exp2 off	(UMB) Fra umbilical. Slå av instrumentgruppe B når inngang er +12V.

Tabell 2.19 Tilgjengelige signaler på 15 pins pluggen på fronten av POW-modulen.

#	Pinnenavn	#	Pinnenavn	#	Pinnenavn
1	UMB Tx on/off	**6**	UMB return	**11**	TMS
2	UMB Exp1 off	**7**	UMB power (12V)	**12**	Internal return
3	Int 2,5V	**8**	UMB Int on/off	**13**	TDO
4	28V return	**9**	TDI	**14**	Internal return
5	28V	**10**	TCK	**15**	UMB Exp2 off

Tabell 2.20 Pluggliste for 15-pins high density hunplugg på fronten av POW-modulen

3 Programmering av STAPPE

Før enkoderen kan brukes, må den programmeres. Alle modulene (PUSEK, AMON, TIMER, POWER, SLAVEPUSEK) kan i utgangspunktet programmeres til en viss grad. To av modulene *må* programmeres før bruk i en rakettnyttelast, nemlig PUSEK og TIMER, og det er i hovedsak disse to brukeren vil ha noe med å gjøre.

I det følgende er filnavn begrenset til 29+3 tegn, noe som burde være langt nok til de fleste praktiske formål (8+3 om programvaren kjøres under MS-DOS). Som vanlig i MS-DOS og forskjellige varianter av MS Windows angir * et fritt valgt filnavn. Filnavn kan ikke inneholde mellomrom. For eksemplene gitt her vil vi gjennomgående angi med kursivskrift (*eksempel*) den delen av filnavnet som velges av brukeren.

Se Appendix A for detaljer om installasjon av den medfølgende programvaren.

3.1 C-preprosessoren gpp

For å gjøre jobben med å beskrive telemetriformatet (i .for-filer) og timerformatet (i .tim-filer) enklere, mer fleksibel og mer oversiktlig brukes enkelte syntaktiske konsepter hentet fra programmeringsspråket C, nærmere bestemt inkludering av filer, bruk av makroer og kommentarer. Dette prosesseres av programmet gpp.exe, som er en preprosessor laget for programmeringsspråket C. Den leser gjennom fila som angis på kommandolinja og utfører preprosessordirektiver (sette inn .inc-filer på rett plass i formatspesifikasjonen, betinget kompilering), fjerner kommentarer og ekspanderer makroer. Andre preprosessorer eller makro-ekspansjonsprogram kan brukes isteden, for eksempel om man ønsker støtte for lengere filnavn. Batchfila conv_for.bat vil i så tilfelle trenge en oppdatering.

Som i programmeringsspråket C starter kommentarer med /* og slutter med */. Merk at en kommentar ikke kan inneholde strengen */, da dette vil tolkes som at kommentaren avsluttes. Dette innebærer også at man ikke kan ha kommentarer i flere nivåer (kommentarer inne i kommentarer). Kommentarer kan gå over flere linjer. For eksempel er

```
/* Dette er en lang kommentar som ikke får plass på
   en enkelt linje og som derfor har blitt fordelt ut
   over flere linjer */
```

en gyldig kommentar, mens

```
/* Man kan ikke ha /* kommentarer inne i */ kommentarer */
```

er ugyldig.
Linjer som starter med # er såkalte *direktiver* til preprosessoren, og behandles spesielt. De to mest aktuelle er #include og #define. Bruken av disse to vil bli kort beskrevet her.

Når direktivet #include leses blir linja det står på byttet ut med innholdet i en angitt fil. For eksempel vil

```
#include "filnavn.inc"
```

bytte ut den aktuelle linja med hele innholdet av fila filnavn.inc. Inkludering av filer vil skje rekursivt, slik at en .inc-fil kan inkludere andre filer, osv. Alle .inc-filer skal ligge i samme katalog som .for og .tim.

Det andre direktivet, #define, brukes for å definere makroer. Som et enkelt eksempel vil

```
#define IDENTIFIKATOR tekst
```

føre til at senere forekomster av strengen IDENTIFIKATOR blir erstattet av strengen tekst. Som i C er det god praksis å bruke store bokstaver på identifikator.

Det er viktig å legge merke til at preprosessoren skiller mellom små og store bokstaver. Derfor vil direktivene

```
#define identifikator tekst
#define Identifikator Tekst
#define IDENTIFIKATOR TEKST
```

alle være ulike. Makroer vil ikke ekspanderes inne i kommentarer.
For mer grundige beskrivelser av hvordan direktivene fungerer, samt andre ting preprosessoren kan gjøre henvises det til lærebøker i C, f.eks. Kernighan & Ritchie: "The C programming language". Denne finnes også i norsk oversettelse, med tittelen "Programmeringsspråket C". Preprosessoren `gpp.exe` er en del av DJGPP, som er en port av kompilatoren GCC til Microsoft-plattformer. DJGPP lisensieres under GPL (GNU General Public License) versjon 2, og kan distribueres fritt. GPL-lisensen er tilgjengelig fra http://www.gnu.org/licenses/old-licenses/gpl-2.0.html. Mer informasjon om DJGPP finnes på http://www.delorie.com/djgpp/. Herfra kan man også laste ned binærdistribusjonen og kildekoden til DJGPP. En oversikt over andre C-kompilatorer som kan være aktuelle å bruke istedenfor DJGPP kan finnes på http://thefreecountry.com/compilers/cpp.shtml. Nevnte URLer er korrekte og fungerer pr. mai 2009.

3.2 Programmering av formatet i STAPPE

Noe forenklet kan man si at programmeringen av telemetriformatet består i å sette opp en tabell som inneholder informasjon om hvordan data fra de enkelte instrumentene i nyttelasten skal pakkes sammen i formatet. Denne tabellen lastes så ned til en programmerbar krets på PUSEK. Når enkoderen senere startes opp vil PUSEK-hardwaren lese denne tabellen, og sørge for at riktig instrument samples til riktig tid, for deretter å sende telemetrisignalet ut til telemetri-senderen og til UMB.

Framgangsmåten for å programmere PUSEK med et PCM-format kan kort oppsummeres slik:

1. Rediger formatfilen *eksempel*.for og tilhørende .inc-filer med en teksteditor.
2. Kjør kommandoen conv_for *eksempel* (eller dobbelklikk på *eksempel*.for)
3. Koble PUSEK til pc-en via en Xilinx programmeringskabel.
4. Kjør kommandoen prog_tab *eksempel* (eller dobbelklikk på *eksempel*.exo)

Vi vil her komme litt mer inn på hva hvert av disse programmene gjør og hvordan man spesifiserer formatet.

3.2.1 conv_for.bat

Kommandoen conv_for.bat er en batchfil som kaller opp andre programmer, som beskrevet i følgende tabell:

Program	Beskrivelse	Leser filer med endelse	Skriver filer med endelse
`gpp.exe`	Preprosessor for programmeringsspråket C.	`.for` `.inc`	`.i`

`conv_format.exe`	Lager en tabell som beskriver PCM-formatet	`.i`	`.vhd`
			`_compressed.vhd`
			`.rpt`

`vhdl2exo.exe`	Konverterer fra VHDL til Motorola exormacs	`.vhd` `_compressed.vhd`	`.exo` `_compressed.exo`

Etter å ha kjørt kommandoen `conv_for` sitter man altså igjen med noen nye filer. De viktigste av disse er *`eksempel`*`.exo` og *`eksempel`*`_compressed.exo`, som er de filene som lastes ned til PUSEK. De andre filene kan gi nyttig informasjon til avanserte brukere og kan være nyttige hvis man må lete etter feil i formatspesifikasjonen. Resultatfilenes funksjon er beskrevet i følgende tabell:

Filtype	Funksjon
`.i`	Utdata fra C-preprosessoren. Alle `.inc`-filer spesifisert i `.for`-filen er inkludert, alle makroer er ekspandert og alle kommentarer fjernet. Denne filen kan være nyttig å studere for å se hva preprosessoren gjør og for å lete etter feil i formatbeskrivelsen (.for og .inc-filene).

.vhd	En beskrivelse av PCM-formatet i VHDL. Kan gi nyttig informasjon til avanserte brukere.
_compressed.vhd	Samme som .vhd, men i en komprimert form som er vanskeligere å tolke enn .vhd.
.rpt	Rapportfil.
.exo	Beskrivelse av PCM-formatet i Exormacs-format . Lastes ned til enkoderen.
_compressed.exo	Samme som .exo, men komprimert. Brukes til å programmere slaveenkoderen. Kan også brukes på nyere enkodere.

Etter bruk kan alle filer som representerer mellomledd i prosesseringen slettes (*.i, *.vhd, *_compressed.vhd, *.rpt).

3.2.2 conv_format.exe

Etter at preprosessoren har gjort jobben sin med å inkludere alle nødvendige filer, ekspandert makroer og fjernet kommentarer gjør programmet `conv_format.exe` den videre prosesseringen for å konvertere formatbeskrivelsen til noe som ligger nærmere en hardwarebeskrivelse av formattabellen. Litt forenklet oversetter `conv_format.exe` brukerens formatbeskrivelse (prosessert av C-preprosessoren) til VHDL.

Legg her merke til at `conv_format.exe` ikke skiller mellom store og små bokstaver og heller ikke mellom linjeskift, tabulator eller mellomrom, i motsetning til C-preprosessoren i forrige steg.

Det er videre viktig å legge merke til at alle tall behandles som positive heltall. Negative tall er ikke gyldige og gjenkjennes ikke av `conv_format.exe`. For alle tall kommer MSB (Most Significant Bit) først i datastrømmen. Tallene kan oppgis i titallssystemet (hvor første siffer *må* være forskjellig fra `0`, bortsett fra selve tallet null) eller heksadesimale (som starter med enten `0x` eller `0X`). Eksempelvis er `0`, `5`, `10`, `15`, `16` og `255` gyldige tall i titallssystemet, mens `016` er ugyldig. Tilsvarende er `0x0`, `0x5`, `0xA`, `0Xf`, `0x10` og `0xFF` de samme tallene i gyldig heksadesimal representasjon.

Summering av tall skjer på vanlig måte, og uttrykk kan fritt blande tall i titallsystemet med heksadesimale tall. Eksempel: `0x5+32` og `11+43+0xA3` er begge gyldige uttrykk. Systemet er begrenset til behandling av maks. 16-bits tall, slik at det største tallet som kan behandles er 2^{16}-1 = 65535 = 0xFFFF heksadesimalt).

3.2.3 Beskrivelse av formatet (`.for`-filer)

Brukerens beskrivelse av telemetriformatet gjøres i form av en tekstfil som spesifiserer nødvendige parametre, som f.eks. gir formatstørrelse, PCM-kode, bitrate, osv., samt en beskrivelse av hvor i formatet data fra hvert enkelt instrument og housekeepingkanal skal legges. Denne filen redigeres ved hjelp av en fritt valgt teksteditor som kan lagre teksten i rent ASCII-

format. Merk at om f.eks. Microsoft Word benyttes må man lagre dokumentet som ren tekst (plain text), og *ikke* som `.doc`, `.rtf` eller andre format som lagrer noe mer enn bare ren ASCII. Filene må lagres som type `.for`, evt omdøpes fra `.txt` til `.for` før `conv_for.bat` kjøres.

Strukturen i `.for`-filene kan kort oppsummeres slik:

```
/* Inkludering av filer med makroer */
#include ”fil1.inc”
#include ”fil2.inc”
[...]
```

```
/* Definering av egne makroer (konstanter) */
#define KONSTANT1 foo
#define KONSTANT2 bar
[...]

/* Spesifikasjon av bitrate, formatstørrelse, pcmkode */
format
[...]

/* Del formatet opp i kanaler og koble opp mot dataene. */
channel_assignments
[...]

end

Egne kommentarer kan skrives her.
```

Egne konstanter definert ved hjelp av `#define`-direktivet kan i prinsippet legges hvor som helst i fila, men det blir som regel mer strukturert om man legger dem etter `#include`-direktivene og før spesifikasjon av bitrate etc., som antydet i eksemplet ovenfor.

Vi vil nå beskrive oppbygningen av `.for`-filene litt mer grundig. Vanligvis begynner en `.for`-fil med å inkludere filer som beskriver de kortene som skal være med. Disse filene må kopieres fra `encoderprog\bin` til den samme folderen som `.for`-filen legges i. Deretter kan en inkludere egne filer som kobler pinnenavn i enkoderen mot navn som bedre beskriver bruken av pinnene. Disse filene skriver brukeren selv. En typisk `#include`-seksjon av `.for`-fila kan f.eks. se slik ut:

```
/* Filer kopiert fra encoderprog\bin: */
#include ”pus_std.inc”   /* Beskrivelse av PUSEK-kortet */
#include ”amon_std.inc”  /* Beskrivelse av AMON-kortet */

/* Brukerens egne definisjoner av pinnenavn i PUSEK+AMON: */
#include ”pus_proj.inc”
#include ”amon_pro.inc”
```

3.2.3.1 Størrelse, hastighet, m.m.

Dernest kommer definisjonen av formatets størrelse og hastighet m.m. Denne delen starter med linjen

```
format
```

Videre kommerer en del parametre som spesifiserer størrelse på dataformatet, hastighet, etc. Det er viktig at parametrene kommer i nøyaktig den samme rekkefølgen som er angitt her.

```
bitrate_div      = rate;
```

Først ut er spesifikasjon av bitrate. Bitraten angis som et heltall, og fysisk bitrate regnes ut slik: Bitrate = 20/(rate+1) MHz, hvor rate er et heltall 1..127. 4 MHz er største hastighet enkoderen er testet med. Se Tabell 3.1 for mulige bitrater.

Valg av modulering til sender (tx) og navlestreng (umb). Kode kan være: `null`, `nrz`, `rnrz` eller `bifi`. Velges `null` legges den angitte utgangen død. `Output_tx` er enkeltendet, variabel amplitude, og brukes normalt til senderen. `Output_umb` er differensiell, og brukes normalt til navlestrengen.

```
output_tx        = kode;
output_umb       = kode;
```

Videre spesifiseres formatstørrelse:

```
maxbit   = nr_msb_bit;
maxword  = nr_maksord;
maxramme = nr_maksramme;
```

Dette gir ord med ordlengde (`nr_msb_bit`+1) bit, rammer med (`nr_maksord`+1) ord og format med (`nr_maksrammer`+1) rammer. Legg merke til at formatet starter med ord 0, ramme 0, og at ordene er organisert med MSB først.

3.2.3.2 Tilordning av kanaler

Denne delen av `.for`-fila starter med linja

```
channel_assignments
```

Dette starter tilordning av kanaler til formatet.

3.2.3.3 Adresselister

Tilordningene er på formen "Plasser i formatet = adresseliste", hvor "Plasser i formatet" angir hvilke bit i hvilket ord og med hvilken kommutering tilordningen gjelder, på formen `word(k)com(m/n)bit(p)`. Her angir `word(k)` hvilket ord i rammen dette gjelder, mens `com(m/n)` angir super/subkommutering, og `bit(p)` angir hvilke bit. "Adresseliste" angir hvor dataene skal leses fra (som for eksempel intern status fra modulene eller fra spesifiserte pinner på moduolene) og angis på formen `adr(liste)`. Liste er ei liste av adresser `adr1,adr2,adr3,...,adrN` og adressene plukkes fra venstre mot høyre. Om det angis færre plasser i formatet på venstresiden av `=`-tegnet enn det er adresser i lista (altså at adresselista er for lang) vil bare det nødvendige antall adresser brukes, mens resten ignoreres. Omvendt, om det er flere plasser i formatet enn det er adresser på høyresiden, vil adresselista

automatisk forlenges med siste element. Merk at det ikke gis tilbakemelding om adresselista er for kort eller for lang.

Noen eksempler:

```
word(2)com(1/1) = adr(55)
```

Her tilordnes ord 2 i hver ramme til pinne med adresse 55. Merk at kanaltildelningen fra en kommando alltid er ekvidistant. I det neste eksemplet er adresselista for lang:

```
word(3)com(1/1)bit(2 to 0) = adr(7,8,9,10,11)
```

Ifølge reglene beskrevet over er dette dermed det samme som

```
word(3)com(1/1)bit(2) = adr(7)
word(3)com(1/1)bit(1) = adr(8)
word(3)com(1/1)bit(0) = adr(9)
```

I det omvendte tilfellet, når adresselista blir for kort, som for eksempel med tilordningen

```
word(3)com(1/1)bit(6 to 0) = adr(7,8,9,10,11)
```

blir adresselista forlenget med siste element:

```
word(3)com(1/1)bit(6) = adr(7)
word(3)com(1/1)bit(5) = adr(8)
word(3)com(1/1)bit(4) = adr(9)
word(3)com(1/1)bit(3) = adr(10)
word(3)com(1/1)bit(2) = adr(11)  /* Siste adresse repeteres  */
word(3)com(1/1)bit(1) = adr(11)  /* siden lista var for kort */
word(3)com(1/1)bit(0) = adr(11)
```

For å forbedre lesbarheten kan man bruke makroer til blant annet å definere adresser og adresselister. I katalogen `encoderprog\bin` ligger det flere `.inc`-filer, ett for hvert kort, med mange makrodefinisjoner som gir adresser i de forskjellige kortene med mer lettleste navn. Vi vil her kort illustrere bruken av disse.

Enkleste variant av kanaltildeling er på formen

```
word(2)com(1/1) = adr(9)
```

Denne tilordningen plasserer data fra adresse 9 i ord 2 alle rammer. I fila `pus_std.inc` finner vi blant annet makrodefinisjonen

```
#define PUSEK_PIN_4_25  9
```

Dermed kan man isteden skrive

```
word(2)com(1/1) = adr(PUSEK_PIN_4_25)
```

Dette bedrer lesbarheten betydelig. Et annet alternativ er å plassere følgende i en egen inkluderingsfil `pus_min.inc`:

```
#define INSTRUMENTNAVN  PUSEK_PIN_4_25
```

og så skrive

```
#include ”pus_std.inc”
[...]
#include ”pus_min.inc”
[...]
word(2)com(1/1) = adr(INSTRUMENTNAVN)
```

3.2.4 Konstanter

Det er to måter å angi konstanter på. Den ene er å bygge opp konstantene ved hjelp av adresselister og de to makroene `NOP_0` og `NOP_1`, som adresserer henholdsvis ’0’ og ’1’. Den andre er å bruke den noe mer brukervennlige `const(tall)`. Disse to metodene er ekvivalente, da `const(tall)` lager en adresseliste som representerer verdien ”`tall`”. Konstanter kan for eksempel brukes til å legge in synkord, rammeteller og andre faste verdier i formatet.

Eksempel: Vi ønsker å legge et 8-bits synkord (0xEB, binært 11101011) i første ord i hver ramme. Ved bruk av `NOP_0` og `NOP_1` kan dette gjøres slik:

```
word(0)sub(1/1)=adr(NOP_1,NOP_1,NOP_1,NOP_0,NOP_1,NOP_0,NOP_1,NOP
_1)
```

Alternativt (og ekvivalent) kan man gjøre det med den mere brukervennlige

```
word(0)sub(1/1) = const(0xEB)
```

Vi har her antatt at det brukes 8-bits ord, altså at `maxbit = 7` er angitt i format-delen, som beskrevet tidligere.

OBS! Det frarådes å bruke adresseringa `word(A to B)` sammen med `const()`, da den gir et resultat som ikke er helt intuitivt. Anta at man skriver `word(A to B)=const(K)`, hvor `K` er et tall som får plass i et ord. Da vil konstanten `K` legges i ord nr. `A`, mens alle bitene i ordene `A+1` til `B` vil fylles opp med den minst signifikante biten i `K`. En mye bedre måte å gjøre dette på vil være å dele det opp i to deler, `word(A)=const(K)` og `word(A+1 to B)=adr(NOP_0)` eller `adr(NOP_1)`, noe som er både mer lettlest og mer intuitivt.

3.2.5 Kommuteringsrate

3.2.5.1 Superkommutering

Hvis data fra et instrument skal opptre mer enn en gang i hver ramme, og at ordene plasseres ekvidistant, bruker man *superkommutering*. For eksempel, hvis et instrument skal samples 3 ganger i hver ramme, og dataene skal legges i ordene `x`, `y` og `z`, kan man skrive

```
word(x)com(1/1) = adr(INSTRUMENTNAVN)
word(y)com(1/1) = adr(INSTRUMENTNAVN)
word(z)com(1/1) = adr(INSTRUMENTNAVN)
```

Dette kan erstattes med

```
/* Superkommutering */
word(x)com(3/1) = adr(INSTRUMENTNAVN)
```

For *subkommutering* gjelder tilsvarende, og kan illustreres med følgende eksempel:

```
/* Subkommutering */
word(w_nr)com(1/N)frame(f_nr) = adr(INSTRUMENTNAVN)
```

Her blir data fra `INSTRUMENTNAVN` lagt i ord `w_nr`, i hver `N`te ramme, regnet fra ramme `f_nr` i formatet. Rent konkret, for

- `N`=2, dvs. `com(1/2)`, blir kanalen lagt i annenhver ramme med start i ramme `f_nr`
- `N`=3, dvs. `com(1/3)`, blir kanalen lagt i hver tredje ramme med start i ramme `f_nr`
- `N`=4, dvs. `com(1/4)`, blir kanalen lagt i hver fjerde ramme med start i ramme `f_nr`

og så videre.

Mikset kommutering, dvs. `com(super/sub)`, hvor både `super` og `sub` > 1, er det ingen støtte for.

3.2.6 Deler av et ord

I tillegg til ord, kommuteringsrate og startramme kan man også spesifisere enkeltbit i formatet. Dette skjer ved å føye til bit(b) i ordadresseringen:

```
word(w_nr)com(1/1)bit(b) = adr(adresse)
```

Dette programmerer bit `b` i ord `w_nr`. Flere bit kan adresseres samtidig ved bruk av `bit(a to b)`. Resultatet vil bli identisk uavhengig av om `a>b` eller `b>a`. Et eksempel på adressering av deler av et ord:

```
word(w_nr)com(1/1)bit(5) = adr(adresse_1)
word(w_nr)com(1/1)bit(4) = adr(adresse_2)
word(w_nr)com(1/1)bit(3) = adr(adresse_3)
```

Dette er det samme som

```
word(w_nr)com(1/1)bit(3 to 5) = adr(adresse_1, adresse_2,
adresse_3)
```

eller

```
word(w_nr)com(1/1)bit(5 to 3) = adr(adresse_1, adresse_2,
adresse_3)
```

3.2.7 Sette sammen ord

Man kan også tilordne en adresseliste til en sammenhengende sekvens av ord:

```
word(n to n+m)com(1/1) = adr(adresseliste)
```

Dette er det samme som

```
word(n+m to n)com(1/1) = adr(adresseliste)
```

Ordene fra og med `ord(n)` til og med `ord(n+m)` tildeles adresser fra adresselista. Tilordningen av adresser fra adresselista skjer i stigende rekkefølge til `ord(n)`, `ord(n+1)`, og så videre opp til `ord(n+m)`. Man kan også gjøre dette med bare deler av ord, dvs.

```
word(n to n+m)com(1/1)bit(5 to 2) = adr(adresseliste)
```

Dette fungerer akkurat som ovenfor, bortsett fra at de her bare er bit 5, 4, 3 og 2 i hvert av ordene n til n+m som brukes.

Om adresselista bare har ett element blir

```
word(2)com(1/1) = adr(adresse_n)
word(3)com(1/1) = adr(adresse_n)
word(4)com(1/1) = adr(adresse_n)
```

det samme som

```
word(2 to 4)com(1/1) = adr(adresse_n)
```

eller

```
word(4 to 2)com(1/1) = adr(adresse_n)
```

3.2.8 Sette sammen rammer

Man kan gjøre tilsvarende ting med rammer ved å bruke `frame(a to b)`:

```
word(w_nr) com(1/x) frame(3) = adr(adresseliste)
word(w_nr) com(1/x) frame(4) = adr(adresseliste)
word(w_nr) com(1/x) frame(5) = adr(adresseliste)
word(w_nr) com(1/x) frame(6) = adr(adresseliste)
```

Dette er det samme som

```
word(w_nr) com(1/x) frame(3 to 6) = adr(adresseliste)
```

eller

```
word(w_nr) com(1/x) frame(6 to 3) = adr(adresseliste)
```

3.2.9 Synkord

Synkordet lages ved å legge inn en konstant verdi i en sekvens av bit, for eksempel slik (forutsatt 8-bits ord):

```
word(0)com(1/1)  = const(0xEB)
word(1)com(1/1)  = const(0x90)
```

Dette legger inn et 16 bits synkord i starten av hver ramme.

Msb-bitet i ord 0 må være konstant ’0’ eller ’1’ (`NOP_0`, `NOP_1`) og brukes vanligvis som første bit i synkordet. Det er normalt, men ikke nødvendig, å legge synkordet helt i starten av hver ramme; synkordet kan plasseres på et vilkårlig sted i formatet. Formatet kan inneholde så mange synkord som det er plass til.

Husk at `word(a to b)` ikke fungerer sammen med `const()`. Av denne grunn blir konstruksjonen `word(0 to 1)com(1/1) = const(0xEB90)` ugyldig.

3.2.10 Rammeteller og formatteller

I et format med N rammer starter *rammetelleren* på 0 i begynnelsen av hvert format, og teller opp til N-1. Plasseringen av rammetelleren i formatet spesifiseres med

```
framecounter word(W)msb_bit(B)
```

hvor `W` er ordet rammetelleren skal plasseres i og `B` angir hvor det mest signifikante bitet skal plasseres. Rammetelleren blir da plassert i ord `W`, i bitene `B`, `B-1`, ...

I et format med 4 rammer kan rammeteller legges i ord 3 ved å skrive

```
framecounter word(3)msb_bit(1)
```

som er nøyaktig det samme som å skrive

```
word(3)com(1/4)frame(0)bit(1 to 0) = const(0)
word(3)com(1/4)frame(1)bit(1 to 0) = const(1)
word(3)com(1/4)frame(2)bit(1 to 0) = const(2)
word(3)com(1/4)frame(3)bit(1 to 0) = const(3)
```

Formattelleren er 24 bit lang og teller formater. Hvert enkelt bit i formattelleren har sin egen adresse. LSB har adresse `PUSEK_FC_LSB` og MSB har adresse `PUSEK_FC_MSB`. Adresselister for hele og deler av formattelleren er definert i fila `pus_std.inc` som `PUSEK_FC_23, PUSEK_FC _22, ... , PUSEK_FC_1` og `PUSEK_FC_0`. Alle `PUSEK_FC_n` er definert som adresselister som adresserer fra og med bit `n` til og med bit 0 i formattelleren: formatteller bit n, formatteller bit n-1, ..., formatteller bit 1, formatteller bit 0. Eksempler: Om vi antar ord på 4 bit vil

```
word(6)com(1/1) = adr( PUSEK_FC_LSB+3, PUSEK_FC_LSB+2,
                       PUSEK_FC_LSB+1, PUSEK_FC_LSB+0 )
```

plassere bitene 3 til 0 av formattelleren i ord 6. Man vil oppnå det samme med

```
word(6)com(1/1) = adr(PUSEK_FC_3)
```

Plassering av bit 7 til 4 i ord 5 kan gjøres med

```
word(5)com(1/1) = adr(PUSEK_FC_7)
```

(fremdeles forutsatt 4 bit ordlengde).
Eksempel med 8-bits ordlengde:

```
word(6 to 8)com(1/1) = adr(PUSEK_FC_23)
```

Dette plasserer hele formattelleren i ordene 6, 7 og 8.

3.2.11 Merking av formatet med kompileringstidspunkt

Plasseres adressen `TID_GEN_FORMAT` eksakt 32 ganger i formatet byttes den ut med en sekvens av `NOP_0` og `NOP_1` slik at en får antall sekunder siden 1. januar 1970.

For eksempel kan kompileringstidspunktet legges inn i ordene 4 til 7 i den fjerde ramma:

```
/* forutsatt 8-bits ord */
word(4 to 7)com(1/X)frame(3) = adr(TID_GEN_FORMAT)
```

Her er X antall rammer i formatet (laveste subkommutering). En annen måte kan være å legge ett eller bare noen få bit i hver ramme, for eksempel slik:

```
/* forutsatt 32 rammer per format */
word(4)com(1/1)bit(3) = adr(TID_GEN_FORMAT)
```

Her brukes bare ett bit i hver ramme (i bit 3, ord 4).

Rate	Bitrate [MHz]	Rate	Bitrate [MHz]	Rate	Bitrate [MHz]	Rate	Bitrate [MHz]
1	10,0000 *	33	0,5882	65	0,3030	97	0,2041
2	6,6667 *	34	0,5714	66	0,2985	98	0,2020
3	5,0000 *	35	0,5556	67	0,2941	99	0,2000
4	4,0000	36	0,5405	68	0,2899	100	0,1980
5	3,3333	37	0,5263	69	0,2857	101	0,1961
6	2,8571	38	0,5128	70	0,2817	102	0,1942
7	2,5000	39	0,5000	71	0,2778	103	0,1923
8	2,2222	40	0,4878	72	0,2740	104	0,1905
9	2,0000	41	0,4762	73	0,2703	105	0,1887
10	1,8182	42	0,4651	74	0,2667	106	0,1869
11	1,6667	43	0,4545	75	0,2632	107	0,1852
12	1,5385	44	0,4444	76	0,2597	108	0,1835
13	1,4286	45	0,4348	77	0,2564	109	0,1818
14	1,3333	46	0,4255	78	0,2532	110	0,1802
15	1,2500	47	0,4167	79	0,2500	111	0,1786
16	1,1765	48	0,4082	80	0,2469	112	0,1770
17	1,1111	49	0,4000	81	0,2439	113	0,1754
18	1,0526	50	0,3922	82	0,2410	114	0,1739
19	1,0000	51	0,3846	83	0,2381	115	0,1724
20	0,9524	52	0,3774	84	0,2353	116	0,1709
21	0,9091	53	0,3704	85	0,2326	117	0,1695
22	0,8696	54	0,3636	86	0,2299	118	0,1681
23	0,8333	55	0,3571	87	0,2273	119	0,1667
24	0,8000	56	0,3509	88	0,2247	120	0,1653
25	0,7692	57	0,3448	89	0,2222	121	0,1639
26	0,7407	58	0,3390	90	0,2198	122	0,1626
27	0,7143	59	0,3333	91	0,2174	123	0,1613
28	0,6897	60	0,3279	92	0,2151	124	0,1600
29	0,6667	61	0,3226	93	0,2128	125	0,1587
30	0,6452	62	0,3175	94	0,2105	126	0,1575
31	0,6250	63	0,3125	95	0,2083	127	0,1563
32	0,6061	64	0,3077	96	0,2062		

*Tabell 3.1 Mulige bitrater. * STAPPE er ikke testet med høyere bitrate enn 4.00 MHz.*

3.2.12 Programmering av ubrukte deler av formatet

Alle posisjoner i formatet *må* programmeres. Som siste kanaltildelning i `.for`-fila kan man avslutte med

```
default = adr(NOP_0)
```

eller

```
default = adr(NOP_1)
```

som fyller alle ledige posisjoner som hittil ikke har blitt tilordnet med `NOP_0` eller `NOP_1`.

3.2.13 Avslutning

Det hele avsluttes med:

```
end
```

All tekst etter `end` ignoreres og kan brukes til kommentarer. Selv om det strengt tatt ikke er nødvendig, anbefales det å omslutte kommentarer her med kommentarmarkørene `/*` og `*/`. Grunnen er at C-preprosessoren potensielt kan bli forvirret om den støter på linjer som starter med `#` (og som den oppfatter som preprosessordirektiv), samt at tekststrenger identiske med navn på makroer under visse omstendigheter kan forårsake problemer.

3.3 Programmering av TIMER

Timeren er en modul som styrer tidssekvensen av hendelser i rakettnyttelasten under flight, f.eks. fyring av squibber og å slå av og på instrumenter til riktig tid. Programmering av timeren består derfor i å lage en liste med informasjon om hva som skal skje ved hvilket tidspunkt, eller mer konkret, hvilken logisk tilstand utgangene fra TIMER-kortet skal ha til enhver tid.
Framgangsmåten for å programmere timeren kan kort oppsummeres slik:

1. Rediger den aktuelle timerfilen *`eksempel`*`.tim` og tilhørende `.inc`-filer med en teksteditor.
2. Kjør kommandoen `conv_tim` *`eksempel`* (eller dobbelklikk på *`eksempel`*`.tim`)
3. Koble TIMER-kortet til pc-en via en Xilinx programmeringskabel.
4. Kjør kommandoen `prog_tab` *`eksempel`* (eller dobbelklikk på *`eksempel`*`.exo`)

En nærmere beskrivelse av hva hvert av programmene gjør og hvordan man spesifiserer timer-sekvensen følger.

3.3.1 conv_tim.bat

Kommandoen conv_tim.bat er en batchfil som kaller opp andre programmer, som beskrevet i følgende tabell:

`conv_tim.bat` kaller på:	Beskrivelse	Leser filer med navn som slutter på	Skriver filer med navn som slutter på
`gpp.exe`	En preprosessor for programmeringsspråket C.	`.tim` `.inc`	.i

conv_timer.exe	Lager en tabell som beskriver timersekvensen, i både VHDL og Motorola exormacs format.	.i	.vhd .exo

Etter å ha kjørt kommandoen conv_tim sitter man altså igjen med en del nye filer. Den viktigste av disse er *eksempel*.exo, som er fila som lastes ned til TIMER. De andre filene kan gi nyttig informasjon til avanserte brukere og kan være nyttige hvis man må lete etter feil i spesifikasjonen av timersekvensen. Resultatfilenes funksjon er beskrevet i følgende tabell:

Filtype	Funksjon
.i	Utdata fra C-preprosessoren. Alle .inc-filer spesifisert i .tim-filen er inkludert, alle makroer er ekspandert og alle kommentarer fjernet. Denne filen kan være nyttig å studere for å se hva preprosessoren gjør og for å lete etter feil i formatbeskrivelsen (.tim og .inc-filene).
.vhd	En beskrivelse av timersekvensen i VHDL. Er ment brukt til simulering. Kan gi nyttig informasjon til avanserte brukere.
.exo	Beskrivelse av timersekvensen i Exormacs-format . Lastes ned til enkoderen.

Etter bruk kan alle filer som representerer mellomledd i prosesseringen slettes (*.i og *.vhd).

3.3.2 conv_timer.exe

Etter at preprosessoren har gjort jobben sin med å inkludere alle nødvendige filer, ekspandert makroer og fjernet kommentarer gjør programmet conv_timer.exe den videre prosesseringen for å konvertere beskrivelsen av timersekvensen til noe som ligger nærmere en hardwarebeskrivelse av timertabellen. Litt forenklet oversetter conv_timer.exe brukerens timersekvens (prosessert av C-preprosessoren) til VHDL og til Exormacs-format. Legg her merke til at conv_timer.exe (i likhet med conv_format.exe) ikke skiller mellom store og små bokstaver og heller ikke mellom linjeskift, tabulator eller mellomrom, i motsetning til C-preprosessoren i forrige steg.

Det er videre viktig å legge merke til at alle tall behandles som enten positive heltall eller relle tall. Negative tall er ikke gyldige og gjenkjennes ikke av programmet conv_timer.exe. For alle tall kommer MSB (Most Significant Bit) først. Hvordan conv_timer.exe tolker et tall (som heltall eller reellt tall) er avhengig av kontekst (hva tallet brukes til), ikke hvordan tallet skrives.

Heltall oppgis i titallssystemet (hvor første siffer *må* være forskjellig fra 0, bortsett fra selve tallet null) eller heksadesimale (som starter med enten 0x eller 0X). Eksempelvis er 0, 5, 10, 15, 16 og 255 gyldige tall i titallssystemet, mens 016 er ugyldig. Tilsvarende er 0x0, 0x5, 0xA, 0Xf, 0x10 og 0xFF de samme tallene i gyldig heksadesimal representasjon.

Reelle tall brukes for tidsangivelser og kan enten skrives som et heltall (som beskrevet over) eller som (`heltall.heltall`). Merk at om man bruker desimalskille (`.`) *må* det oppgis siffer både før og etter punktum. For eksempel er `42`, `3.14`, `0.25` og `75.0` alle gyldige relle tall, mens `.25` og `17.` er ugyldige.

Summering av tall skjer på vanlig måte, og uttrykk kan fritt blande tall i titallsystemet med heksadesimale tall og reelle tall. Eksempel: `12.3+32.2` og `11.2+43` er begge gyldige uttrykk, mens `33+11` er et gyldig reellt uttrykk når det brukes i en sammenheng hvor det forventes en tidsangivelse.

3.3.3 Beskrivelse av timersekvensen (`.tim`-filer)

Brukerens beskrivelse av timersekvensen gjøres i form av en tekstfil som spesifiserer nødvendige parametre samt en beskrivelse av hvilken tilstand timerens kommandoord skal ha til enhver tid. Denne filen redigeres ved hjelp av en fritt valgt teksteditor som kan lagre teksten i rent ASCII-format. Merk at om f.eks. Microsoft Word benyttes må man lagre dokumentet som ren tekst (plain text), og *ikke* som `.doc`, `.rtf` eller andre format som lagrer noe mer enn bare ren ASCII. Filene må lagres som type `.tim`, evt omdøpes fra `.txt` til `.tim` før `conv_tim.bat` kjøres.

Strukturen i `.tim`-filene kan kort oppsummeres slik:

```
/* Inkludering av filer med makroer */
#include ”fil1.inc”
#include ”fil2.inc”
[...]

/* Definering av egne makroer (konstanter) */
#define EVENT1 65
#define EVENT2 123.456
[...]

/* Informasjon om timeren */
hardware_info
[...]

/* Timerkommandoer */
sekvens
[...]

end

Egne kommentarer kan skrives her.
```

Egne konstanter definert ved hjelp av `#define`-direktivet kan i prinsippet legges hvor som helst i fila, men det blir som regel mer strukturert om man legger dem etter `#include`-direktivene, som antydet ovenfor.

Vi vil nå beskrive oppbygningen av `.tim`-filene litt mer grundig. Vanligvis begynner en `.tim`-fil med å inkludere fila `timt_std.inc`, som må kopieres fra katalogen `encoderprog\inc_timer` til den samme katalogen som `.tim`-fila ligger i. Deretter inkluderes filer som kobler makronavn med navn som beskriver funksjoner i nyttelasten. Disse filene skriver brukeren selv. Vanligvis er det nok med bare én fil. Etter `#include`-direktivene kan man definere egne makroer og konstanter. En typisk start på `.tim`-fila kan se slik ut:

```
/* Standardfil for programmering av timer */
/* (kopiert fra encoderprog\inc_timer)    */
#include ”timt_std.inc”

/* Inkludering av filer som beskriver funksjoner i nyttelasten.*/
/* Disse skrives av brukeren selv (vanligvis nok med én fil.   */
#include ”timt_pro.inc”

/* Brukerens egne makroer og konstanter */
#define TID1 55.0 /* sekunder */
#define TID2 72.5 /* sekunder */
[...etc...]

/* Spesifisering av tidsoppløsning */
hardware_info
time_resolution = 0.1 /* sekunder */
```

Merk at den foreløpig eneste tilgjengelige tidsoppløsningen er 0.1 s.

3.3.3.1 Timersekvens

Neste del av `.tim`-fila er en liste med beskrivelser av hva som skal skje under flighten, og når det skal skje. Starten av timersekvenslista angis med stikkordet `sequence`:

```
sequence
```

Denne lista må være kronologisk og kan bare inneholde ett element for hvert tidspunkt.

Timeren inneholder en tabell med 24-bits kommandoord. Utgangene fra timeren er styrt av hvert sitt bit i ordet. Når timeren er resatt er det ordet fra tid 0 som brukes. Når timeren starter henter den fram neste ord med en periode på 0.1 sekund. Tabellen er lang nok for en sekvens på én time. Beskrivelsen av hver enkelt hendelse starter med angivelse av tidspunktet, etterfulgt av en liste med kommandoord (rent teknisk er dette en spesifikasjon av hvilke bit i timeren som skal settes til logisk 1 eller 0 for dette tidspunktet. Skjematisk sett ser hver enkelte timerhendelse slik ut:

```
time = tidspunkt /* i sekunder */
{
    bitsekvens = heltall
    bitsekvens = heltall
    [...]
}
```

Bitsekvensene angis her på formen `(bit[N])` eller `(bit[M..N])`, hvor `M` og `N` er heltall og angir bitposisjon i timerens kommandoord, og dermed må være i intervallet [0, 23]. Merk at om man bruker et intervall, altså formen `(bit[M..N])`, kan `M` og `N` kun være

konstante tall i titallssystemet, og ikke makroer eller heksadesimale tall, og det må ikke brukes mellomrom før og etter "`..`". Altså vil `(bit[17..5])` og `(bit[23..0])` være gyldige, mens `(bit[14 .. 9])`, `(bit[BIT_A..BIT_B])`, `(bit[FOO .. BAR])` og `(bit[0x0F..0])` alle være ugyldige.

Første element i listen må være for 0 sekunder og spesifisere alle bit i kommandoordet:

```
time = 0
{
        /* Alle utganger av og TIMER_MODE = NORMAL */
        (bit[23..0]) = 0
}
```

Om et bit spesifiseres flere ganger vil det være den sist brukte som gjelder. Det kan benyttes til å først sette alle utganger til default av og så skru på enkeltutganger, som i dette eksemplet:

```
time = 0
{
        /* Sett alle bit til 0, bortsett fra 5 og 9 */
        (bit[23..0])     = 0  /* default */
        (bit[9])         = 1
        (bit[5])         = 1
}
```

Dette er det samme som

```
time = 0
{
        (bit[23..10])    = 0
        (bit[9])         = 1
        (bit[8..6])      = 0
        (bit[5])         = 1
        (bit[4..0])      = 0
}
```

For å lage en puls med varighet 0.1 sekunder for fyring av en squib med 2 bruer styrt av bitene 2 og 3 vil man typisk gjøre det slik:

```
/* Send puls på 0.1 s til squibber */
time = 50
{
        (bit[2])         = 1
        (bit[3])         = 1
}
time = 50.1
{
        (bit[2])         = 0
        (bit[3])         = 0
}
```

I fila timt_std.inc finnes det makroer som gjør at en isteden kan skrive:

```
time = 50
{
        SQUIB_02 = 1
        SQUIB_03 = 1
}
time = 50.1
{
        SQUIB_02 = 0
        SQUIB_03 = 0
}
```

eller

```
time = 50
{
        SQIB_COMMAND_02_PIN_14_54 = 1
        SQIB_COMMAND_03_PIN_15_56 = 1
}
time = 50.1
{
        SQIB_COMMAND_02_PIN_14_54 = 0
        SQIB_COMMAND_03_PIN_15_56 = 0
}
```

Et bedre alternativ er å ha egne makroer i en egen inkluderingsfil. For eksempel, om `my_timer.inc` inneholder

```
#define NESEKON_BRO_1              SQIB_COMMAND_02_PIN_14_54
#define NESEKON_BRO_2              SQIB_COMMAND_03_PIN_15_56
```

kan man skrive

```
#include ”my_timer.inc”
[...etc...]
time = 50
{
        NESEKON_BRO_1 = 1
        NESEKON_BRO_2 = 1
}
time = 50.1
{
        NESEKON_BRO_1 = 0
        NESEKON_BRO_2 = 0
}
```

3.3.3.2 Timermode

Timeren har 4 modi styrt av to av bitene i kommandoordet, nærmere bestemt bit 20 og 21, som kan adresseres med `(bit[21..20])`:

1. `NORMAL` – Timeren henter et nytt kommandoord hvert 0.1 sekund.
2. `WAIT_FOR_BARO` – Timeren stopper hvis det er 28V fra barometerbryteren. Ment brukt når ting skal skje i lav høyde på nedtur, typisk ved recovery hvor fallskjermer skal utløses og nyttelasten skal skrus av før den blir våt. *Det er viktig å legge merke til at*

timeren ikke lenger kan lade opp fyrekondensatorene når den starter opp igjen etter å ha ventet på barometerbryteren. Grunnen til dette er at barobryteren skal forhindre squibber i å fyres av mens nyttelasten står på bakken.

3. PAYLOAD_OFF – Skrur av nyttelasten. Ment brukt sammen med recovery.
4. STOP – Timeren stopper. Avslutt alltid med STOP selv om PAYLOAD_OFF er brukt til å skru av nyttelasten. Grunnen til dette er bl.a. for å redusere sjansen for feil under testing av nyttelasten, hvor barobryteren er deaktivert. La det gå minst et sekund mellom PAYLOAD_OFF og STOP.

Timermodus kan settes ved hjelp av de forhåndsdefinerte makroene TIMER_MODE, NORMAL, WAIT_FOR_BARO, PAYLOAD_OFF og STOP. Eksempel:

```
time = 0
{
        TIMER_MODE = NORMAL
}
```

3.3.3.3 Eksempel på en timersekvens

```
#include ”timt_std.inc”
#include ”my_timer.inc”
#define BAROTIME 300

time = 0
{
        /* Alle utganger av (impliserer TIMER_MODE = NORMAL) */
        (bit[23..0]) = 0
        /* Det er god praksis å eksplisitt sette  */
        /* TIMER_MODE = NORMAL selv om det gjøres */
        /* implisitt i kommandoen over            */
        TIMER_MODE = NORMAL
}

/* Fyr squibber til nesekon med en 0.1 sek puls ved 50 sek */
time = 50.0
{
        NESEKON_BRO_1 = 1
        NESEKON_BRO_2 = 1
}
time = 50.1
{
        NESEKON_BRO_1 = 0
        NESEKON_BRO_2 = 0
}

/* Stopper og venter på baroswitchen */
time = BAROTIME
{
        TIMER_MODE = WAIT_FOR_BARO
}
/* Skrur av nyttelasten 10s etter at baroen har skrudd av 28V */
time = BAROTIME+10
{
        TIMER_MODE = PAYLOAD_OFF
}
/* Stopper timeren 11,5s etter at baroen har skrudd av 28V.  */
/* Avslutt alltid med TIMER_MODE = STOP, selv om nyttelasten */
/* ikke skal berges.  Dette for å unngå at TIM prøver å lese */
/* ut over enden av timersekvenslista, og dermed utføre      */
/* ugyldige og/eller uønskede instruksjoner under testing.   */
time = BAROTIME+11.5
{
        TIMER_MODE = STOP
}
```

3.4 Installering av programvare

Følgende installasjonsbeskrivelse tar utgangspunkt i Windows XP, men vil være svært lik for andre varianter av MS Windows. Softwaren kommer pakket sammen i fila `encoderprog.zip`. Pakk ut fila med `winzip` eller tilsvarende i en folder etter fritt valg. Folderen `encoderprog\bin` må ligge i environmentvariabelen `path`. Programmet `encoderprog\startbat.exe` skal assosieres med filtypene `.for`, `.tim`, `.exo`, `.mcs` og `.bit`. I tillegg anbefales det at filtypene assosieres med `.ico`-filene fra folderen `encoderprog`. Filene som slutter med `.ico` har navn tilsvarende filtypen de hører sammen med.

3.4.1 Katalogstruktur:

`encoderprog` (folder)

`startbat.exe`		
`exomcs.ico`	`for.ico`	`mcs.ico`
`exo.ico`	`tim.ico`	`bit.ico`

`encoderprog\bin` (folder) Må ligge i path. Inneholder programmer.

`bell.bat`	`conv_for.bat`	`prog_tab.bat`
`prog_flash.bat`	`conv_tim.bat`	`mo.bat`
`impactport.bat`	`conv_format.exe`	
`gpp.exe`	`vhdl2exo.exe`	
`conv_timer.exe`		

`encoderprog\bin\pre` (folder) Inneholder diverse filer som brukes av preprosessoren.

`encoderprog\inc_format` (folder) Standard .inc-filer for enkoderprogrammering.

`amo_xtr1.inc`	`pipepstd.inc`	`pow_std.inc`
`pus_std.inc`	`amon_std.inc`	

`encoderprog\inc_timer` (folder) *Standard `.inc`-filer for timeren.*

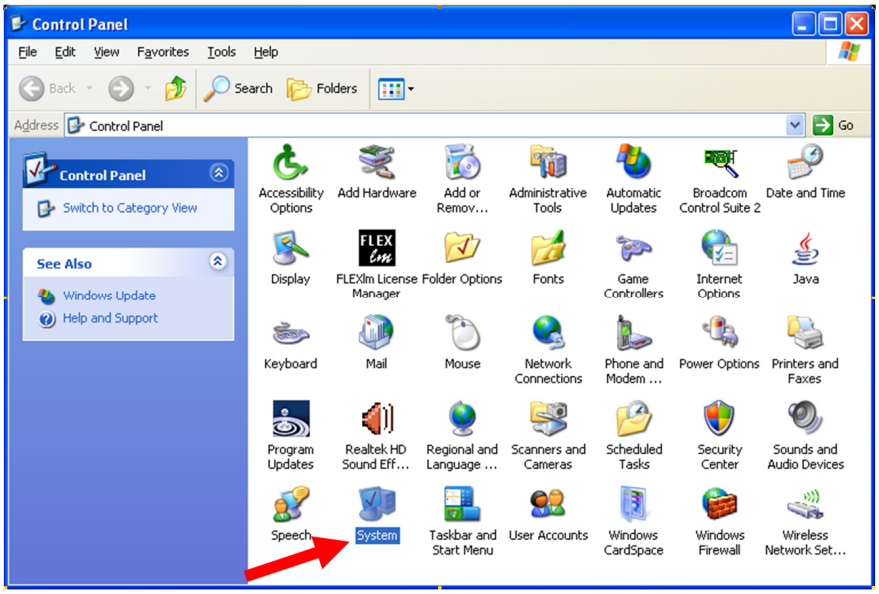
`timf_std.inc`

3.4.2 Filassosiasjon og path; detaljert beskrivelse

Filen `startbat.exe` må legges inn i path. Dette gjøres via System Properties i Control Panel.

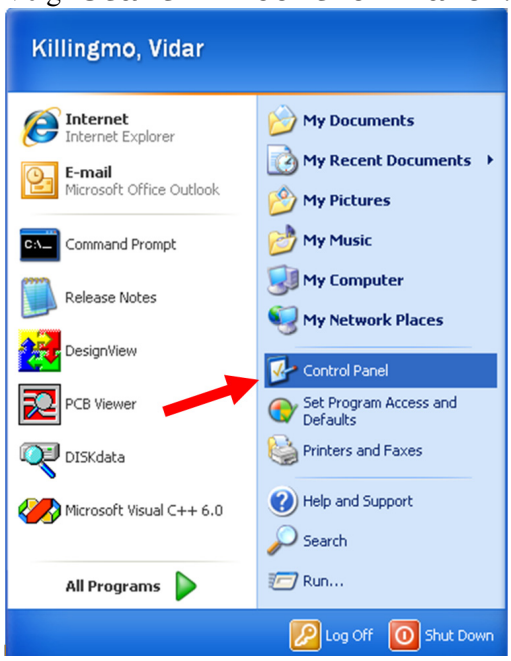
3.4.2.1 For klassisk startmeny

Klikk på `Start -> Settings -> Control Panel`.
Velg `System` og klikk.

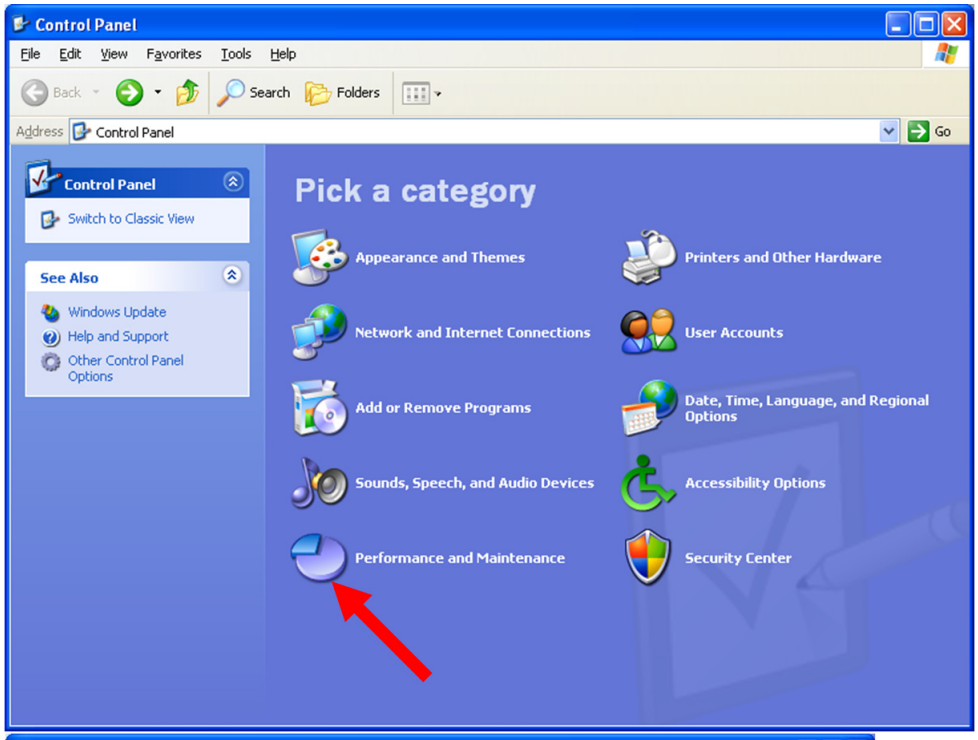


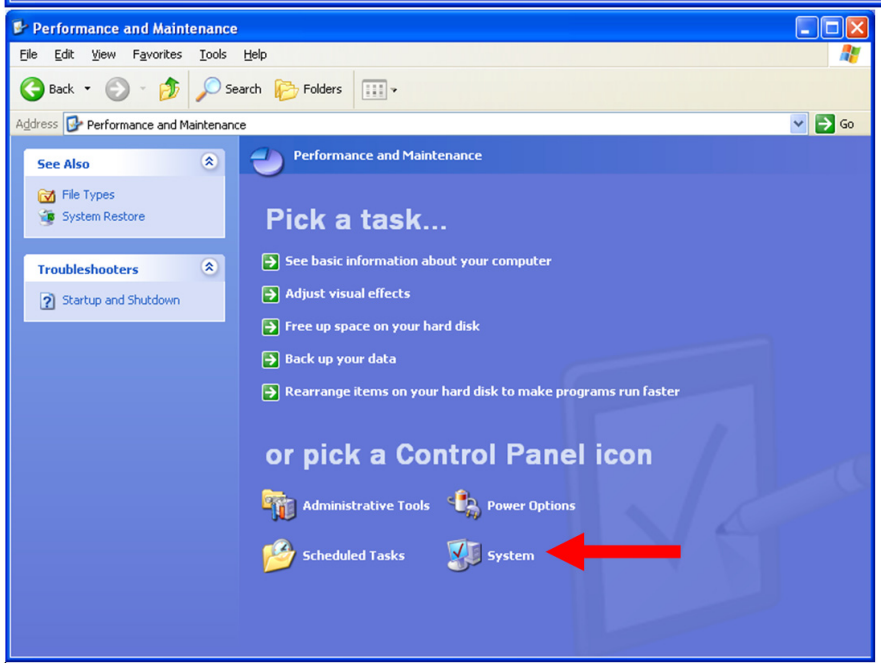
3.4.2.2 For moderne startmeny

Velg `Start -> Control Panel`:



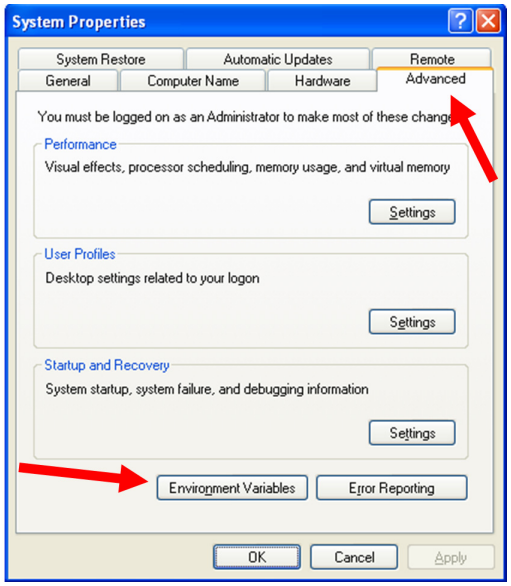
Velg `Performance and Maintenance` og deretter `System`:





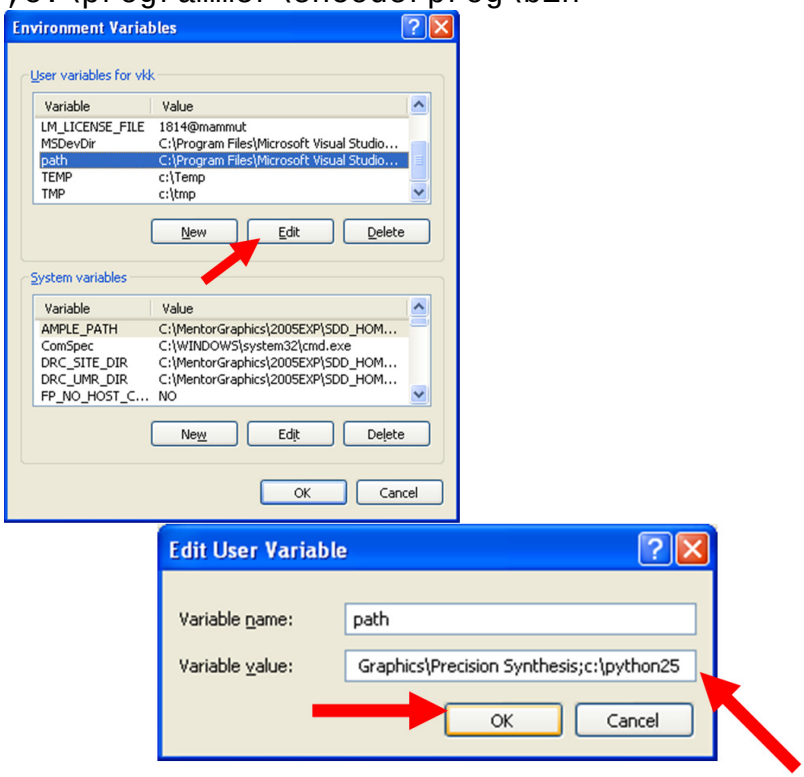
3.4.2.3 Felles for begge typer startmeny

Herfra blir det likt for begge variantene.
Velg arkfanen `Advanced` og klikk `Environment Variables`:



Velg `path` i vinduet for brukervariabler og klikk `Edit`. I vinduet som nå dukker opp, klikk i feltet for variabelverdi og trykk på `end`-tasten for å komme til slutten av linja. Legg til stien for `bin`-folderen. Pass på å få med med et semikolon (`;`) foran. Eksempel:

```
;c:\programmer\encoderprog\bin
```

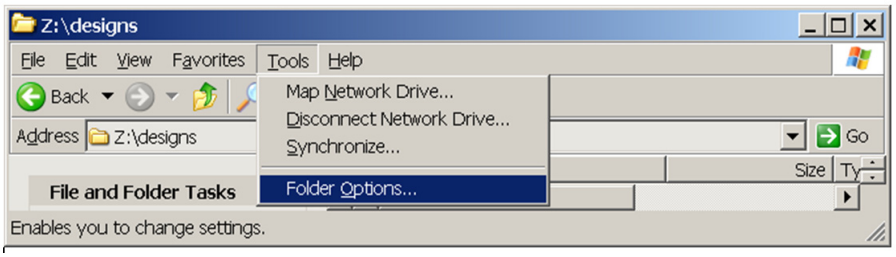


Klikk `OK` i alle tre vinduene.

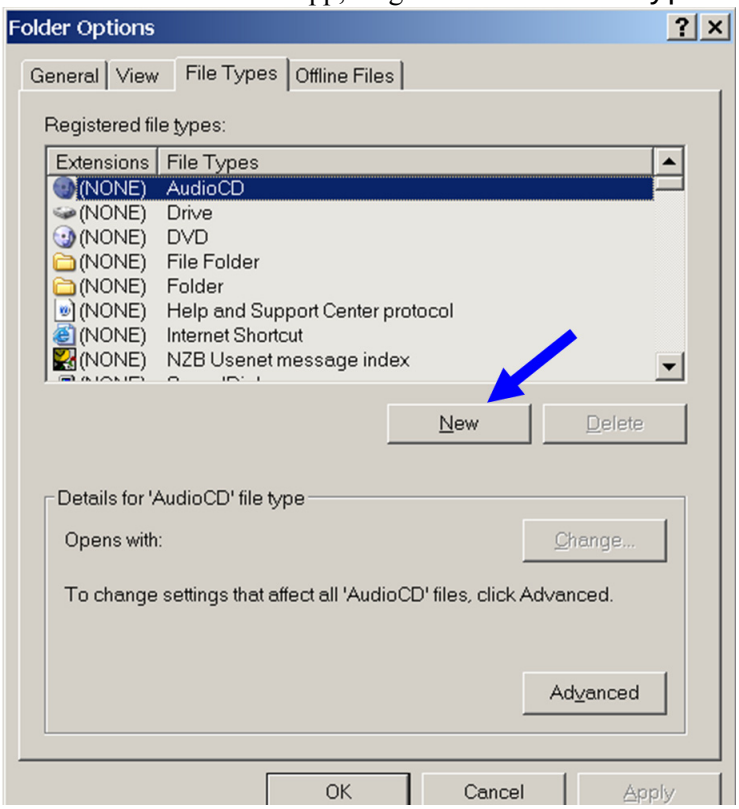
3.4.2.4 Assosiering av filtyper

Det neste steget er å få assosiert filtyper til `startbat.exe` og ikoner til filtypene.

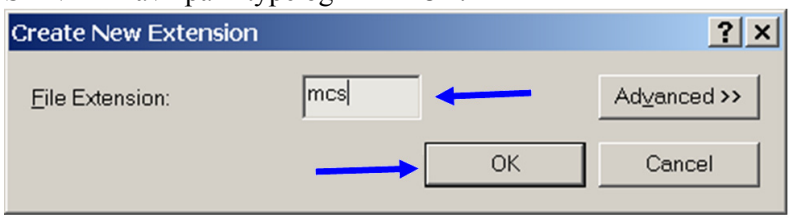
Start `Explorer` (utforsker). Velg menyen `Tools->Folder Options`.



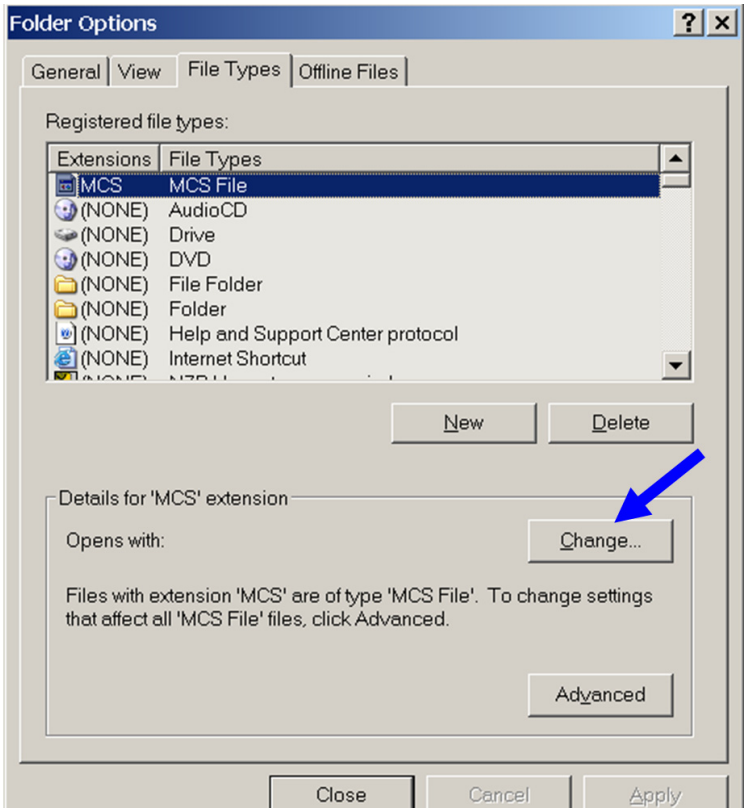
I vinduet som nå dukker opp, velg arkfanen `File Types` og klikk på `New`:



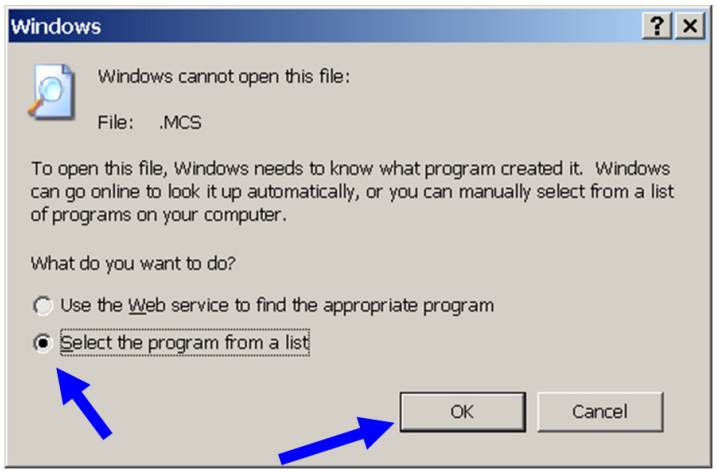
Skriv inn navn på filtype og klikk `OK`:



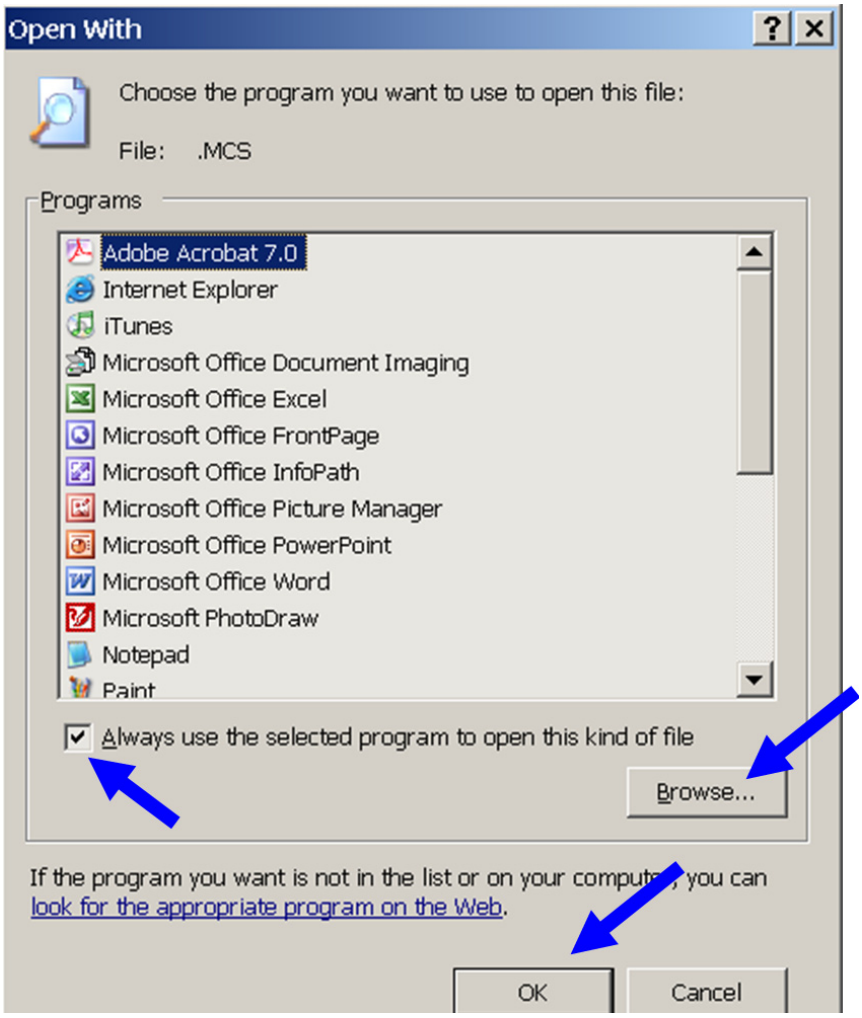
Velg den nye filtypen og klikk på `Change`. Merk at om du ønsker å velge ikon for denne filtypen gjøres dette via `Advanced`-knappen, og at dette i så fall må gjøres *før* du trykker `Change`.



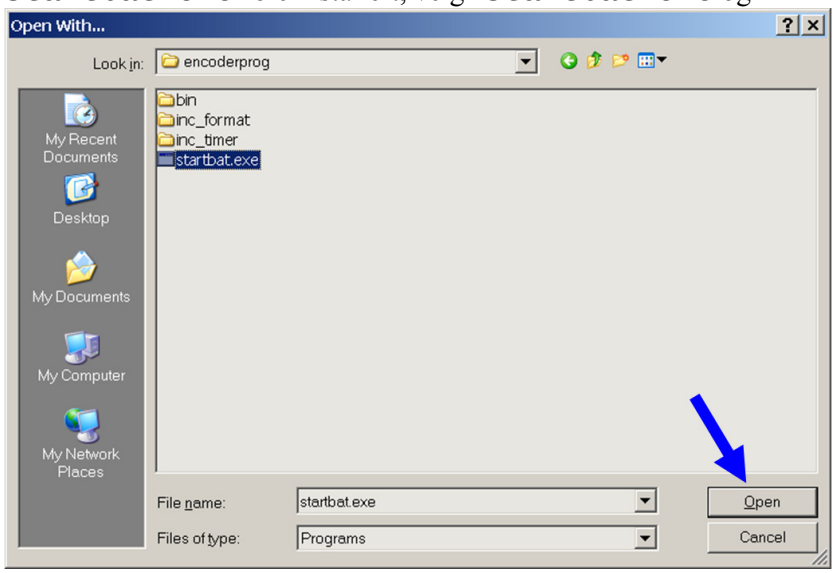
Velg `Select the program from a list` og klikk `OK`.



Hak av ved `Always use the selected program to open this kind of file`. Velg `startbat.exe` fra lista og klikk `OK`. Gjenta operasjonen for de andre filtypene (`for`, `tim`, `exo`, `mcs` og `bit`).



Om `startbat.exe` ikke finnes i lista, klikk `Browse...` og klikk deg fram til der hvor `startbat.exe` ble installert, velg `startbat.exe` og klikk `Open`.



Hvis en ønsker å velge ikon og `Advanced`-knappen er erstattet med en `Restore`-knapp i dette vinduet må en klikke på `Restore` og begynne forfra med assosiering for den valgte filtypen, og passe på å velge ikon *før* man velger programmet filtypen skal åpnes med.