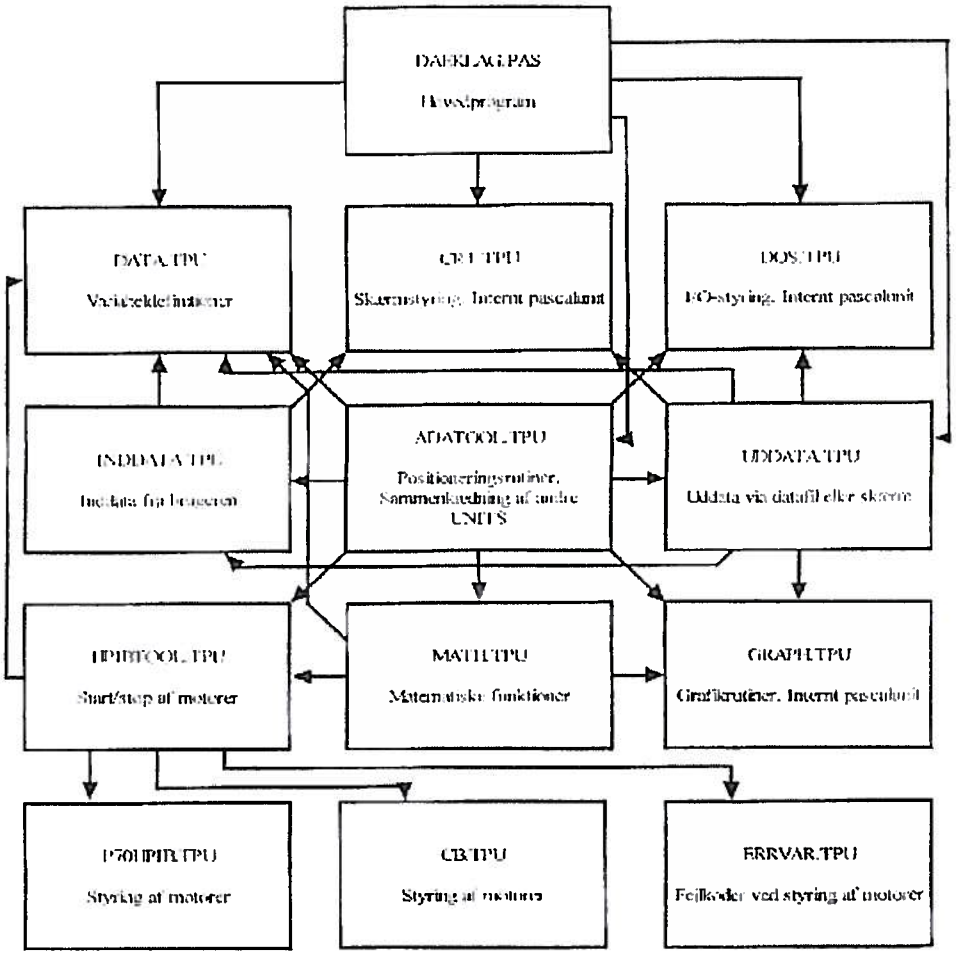
BYG·DTU

Claus Rudbeck

Programdukomentation til Styringsprogram til Solar-tracker



DANMARKS TEKNISKE UNIVERSITET

DTU

SAGSRAPPORT

BYG • DTU SR-01-03

1995

ISSN 1396-402X

Indholdsfortegnelse:

1. Indledning 4

2. Dokumentation af unit *adatool* 7
2.1 Funktions/procedurekald i proceduren *initier* 7
2.2 Funktions/procedurekald i proceduren *set_start_pos* 9
2.3 Funktions/procedurekald i proceduren *tidsstep* 10

3. Procedurekald i transmittansmålingsprogram 18
3.1 Variabelforklaring til procedure *initier* 20
3.2 Variabelforklaring til procedure *set_start_pos* 22
3.3 Variabelforklaring til procedure *tidsstep* 24

4. Implementering af procedurer i solfangerprogram 27

5. Beskrivelse af hardware 30
5.1 Samleboks til pulstæller 30
5.2 Mobil skyggeklods til diffus-strålingssolarimeter 30

6. Litteraturliste 32

Appendix A. Positionering ved tracking med vinkelforskel 33
A.1 Fremgangsmåde for beregninger 33
A.2 Fastlæggelse af planer 34
A.3 Opstilling af planernes ligninger 34
A.4 Rumvinkel mellem solpunktsvektor og fladenormal 38
A.5 Forskydningen af azimut og hældning bestemmes 39

Appendix B. Kildekode til transmittans- og delprogrammer 44
B.1 Kildekode til ADATOOL.TPU 46
B.2 Kildekode til DATA.TPU 68
B.3 Kildekode til DAEKLAG.EXE 71
B.4 Kildekode til HPIBTOOL.TPU 72
B.5 Kildekode til INDDATA.TPU 74
B.6 Kildekode til MATH.TPU 81

B.7 Kildekode til UDDATA.TPU 87

Appendix C. Reserverede navne i styringsprogram 93
C.1 Reserverede UNIT-navne 93
C.2 Reserverede PROCEDURE-navne 93
C.3 Reserverede FUNCTION-navne 94
C.4 Reserverede TYPE-navne 94
C.5 Reserverede VAR-navne 95
C.6 Reserverede CONST-navne 101

1. Indledning:

Denne rapport indeholder programdokumentationen for dels et delprogram til styring af et trackingsystem og dels et program, der benytter dette delprogram til at foretage målinger af transmittans for dæklag. Såvel transmittansmålingsprogrammet som delprogrammet er opbygget i Borland Pascal v7.0, og er compileret i *Real mode* til brug for afvikling på en processor af 80X86-familien. Kildekoden til programmet til transmittansmålinger og delprogrammerne (positioneringsrutinerne) kan findes i Appendix B fra side 44 til side 92.

Positioneringsrutinerne kræver, at der er tilsluttet et HPIB-interfacekort til computeren, hvor der er monteret et pulstællerkort i kortposition 4. Imellem pulsgiverne og pulstællerkortet skal der være indsat en "samleboks" for pulssignalerne. Samleboksens funktion er at samle pulserne fra de tre pulsgivere til et signal, der går videre til pulstællerkortet. Yderligere beskrivelse af samleboksen kan findes på side 30. I kortposition 2 kræves desuden et 20-kanals målekort, som kan måle spændingsforskelle. Spændingsforskellene stammer enten fra solarimetrene eller hvis kontakterne ved endestoppene sluttes. Af de 20 kanaler, der er på kortet i position 2, er kun de første 9 brugt (kanal 40-48 incl.) til positionerings- og dæklagsrutiner, hvorfor kanal nummer 49-59 kan bruges til andre formål, hvis det ønskes. Kortposition 0, 1 og 3 bruges ikke i transmittans-målingsprogrammet eller i positioneringsrutinerne, og er dermed frilagt til andre formål. Dette kunne eksempelvis være temperaturmålinger ved effektivitetsmålinger på solfangere.

Udover HPIB-interfacekortet kræver positioneringsrutinerne at der er tilsluttet en relæ-boks, hvor det er muligt at åbne eller lukke relæer. Disse relæer bruges til angive en rotationsretning for motorerne og om motorerne skal køre eller ej. En oversigt over relænumre for de tre motorer og relæernes funktion fremgår af programudskriften for *data* på side 68.

Delprogrammet *adatool* er opbygget som en *unit* i Borland Pascal v7.0, hvilket betyder, at et hovedprogram kan kalde funktioner og procedurer fra dette delprogram ved en simpel henvisning. Fra delprogrammet *adatool* henvises til andre delprogrammer (procedurer og funktioner). Disse procedurer og funktioner er oprettet for at lette overskueligheden i programmet, og indeholder variabelerklæringer, matematiske hjælpefunktioner og funktioner til styring af motorerne. Delprogrammet *adatool* er ikke noget egentligt program, idet alle kode-linierne i delprogrammet er procedurer eller funktioner. *Adatool* kan kun udføre dets instruktioner, hvis der er et hovedprogram, der kan kalde disse procedurer og funktioner med de relevante parametre. Et eksempel på et hovedprogram, der bruger instruktionerne i *adatool*, er programmet til måling af transmittans gennem dæklag. På tilsvarende måde er det muligt at benytte positioneringsrutinerne

i andre allerede eksisterende programmer: Solfangerprøvningsprogram og Metsetprogram. Eksempler på en sådan implementering i disse programmer kan ses i kapitel 4: Implentering af procedurer i solfangerprogram.

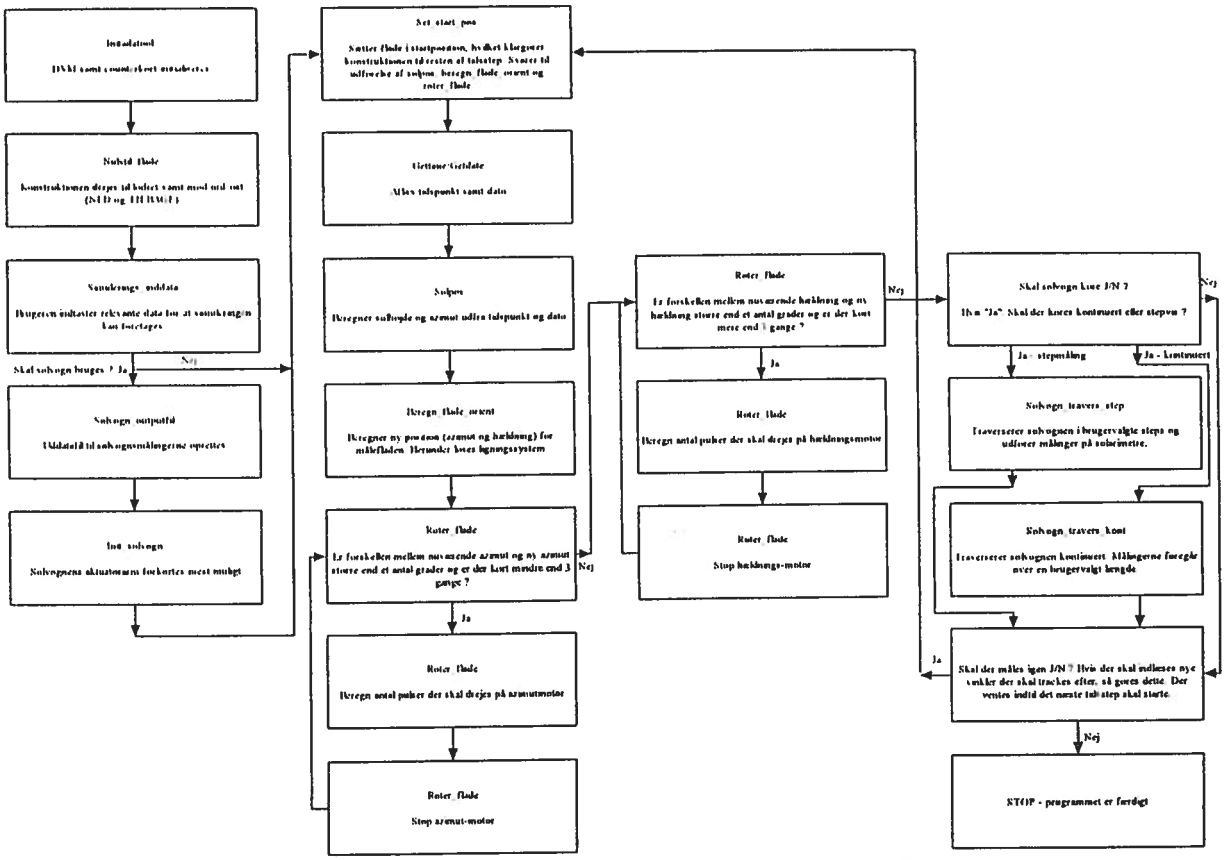
Overordnet kan antallet af kald fra hovedprogrammet til delprogrammet begrænses til to.

I starten af hovedprogrammet kaldes proceduren *initier*. Ved dette kald anføres tillige nogle parametre, der forklares senere i programdokumentationen. Denne procedure giver brugeren mulighed for at indtaste relevante måleparametre, nulstille målekort i PC'en og rotere målefladen til en fast udgangsposition.

Det andet kald fra hovedprogrammet er kaldet af proceduren *tidsstep*. Dette kald gentages med jævne mellemrum i løbet af en måling. Ved procedurekaldet anføres tillige en række parametre. Disse parametre beskrives senere i programdokumentationen. Tiden mellem hvert kald af *tidsstep* er bestemt af brugeren. I *tidsstep* beregnes fladens nye position ud fra solens position, hvorefter målefladen roteres til den ønskede position.

Hvis brugeren af programmet har specificeret, at der skal foretages en transmittansmåling på et dæklag, udføres denne efter, at fladen er positioneret. Transmittansmålingen foregår ved at bestrålingsstyrken måles med pyranometre over og under af dæklaget. For at kunne måle på inhomogene dæklag, er pyranometeret, der måler stråling efter passage af dæklaget, monteret på en lille vogn. I programmet er det tillige muligt at specificere, om denne solvogn skal bevæges i små steps under dæklaget, eller om bevægelsen skal foregå kontinuert.

Af programoversigten på figur 1 kan et forenklet rutediagram af programmet ses. I øverste venstre hjørne af rutediagrammet initialiseres målekort, og konstruktionen drejes til en kendt start-position. I de næste to næste kolonner bestemmes målefladens nye position ud fra kendskab til solens position, hvorefter målefladen drejes. Hvis der skal foretages målinger på dæklag udføres instruktionerne i kolonnen yderst til venstre. Her foretages enten en kontinuert eller stepvis bevægelse af solvognen, hvor der samtidig udføres målinger af den totale, den diffuse og den transmitterede solstråling.



Figur 1. Forenklet rutediagram for program til tracking og dæklagsmålinger

2. Dokumentation af unit *adatool*:

Hovedprocedurerne i unit *adatool* er proceduren *initier*, der er beskrevet i afsnit 2.1, og proceduren *tidsstep*, der er beskrevet i afsnit 2.3. Den første har med initialisering af måleudstyret at gøre, mens den anden procedure bruges til positionering af målefladen. For at kunne positionere målefladen i næsten den rigtige position allerede i første tidsstep, er der oprettet en lille procedure kaldet *set_start_pos*, hvor beskrivelsen af denne kan findes i afsnit 2.2.

2.1 Funktions/procedurekald i proceduren *initier*:

I løbet af et gennemløb af proceduren *initier* kaldes følgende funktioner og procedurer:

initadatool
bruger_inddata
nulstil_flade

Hvis det i *bruger_inddata* bliver anført, at der ønskes foretaget måling på dæklag udføres tillige følgende procedurer:

solvogn_outputfil
init_solvogn

I funktionen *initadatool* opsættes målekort (digitalvoltmeter og pulstællerkort) og relæboks til deres ønskede funktioner og måleområder. Under opsætningen defineres at pulstællerkortet sidder i slot 4 og at pulstællerkortet altid tæller op i step af en.

Input til funktion:	Ingen
Output fra funktion:	Status for målekort. Hvis målekortene bliver initialiseret uden fejl, returnerer funktionen værdien "TRUE", mens en initialisering hvori der er forekommet en fejlmeddelelse medfører, at der returneres værdien "FALSE". Er værdien "FALSE" stopper programudførelsen og der udskrives en fejlmeddelse på skærmen.

I proceduren bruger_inddata, der kan findes i unit *ind_data,* skal brugeren af programmet indtaste relevante inddata. Hvis der ikke udføres dæklagsmålinger skal inddata der henfører til dette ikke indtastes. Variable for disse inddata og deres betydning er følgende:

measure_type	Målingstype (Solfanger, Metset eller Dæklagsmåling). Målingstypen er angivet som et tal mellem 1 og 3, hvor 1 er Solfanger, 2 er Metset og 3 er Dæklagsmåling
tracking_type	Trackingtype (Om der trackes om nul, en eller to akser). Trackingtypen er angivet som et tal mellem 1 og 4, hvor 1 er tracking udelukkende i højderetningen, 2 er tracking udelukkende i azimutretningen, 3 er tracking i både højde- og azimutretning og 4 er en stillestående måleopstilling.
delta_azimut	Vinkel mellem målefladens normal og solens azimutvinkel. Vinklen er angivet i grader
delta_haeld	Vinkel mellem målefladens normal og solhøjde. Vinklen er angivet i grader
locked_azimut	Eventuel fast azimutvinkel for målefladen. Vinklen er angivet i grader
locked_haeld	Eventuel fast hældningsvinkel for målefladen. Vinklen er angivet i grader
h_flade_bred	Højre begrænsning af måleområde for solvogn ved dæklagsmåling. Variablen angiver afstanden fra midten af måleområdet til højre begrænsning i milimeter (incl. fortegn)
v_flade_bred	Venstre begrænsning af måleområde for solvogn ved dæklagsmåling. Variablen angiver afstanden fra midten af måleområdet til venstre begrænsning i milimeter (incl. fortegn)
solvogn_step	Afstand i milimeter mellem målepunkter ved dæklagsmåling
solarimeter_indsving	Pause i milisekunder til indsvingning af solarimetre ved dæklagsmåling
solvogn_maal_type	Kontinuert eller punktvis måling på solarimetrene ved dæklagsmåling. Måletypen afgives som et tal mellem 1 og 2, hvor 1 er en punktvis måling, mens 2 er en kontinuert måling

Input til procedure:	Ingen ekstern input - kun brugerindtastet
Output fra procedure:	De ovenfor nævnte inddata

I proceduren *nulstil_flade* drejes og vippes målefladen til en kendt udgangsposition. Denne position er bestemt af nogle endestop, der er monteret på konstruktionen. Når disse endestop nås gives en spændingsforskel på en forudbestemt kanal, som kan måles via digitalvoltmeteret. Kanalnumre for endestop og relænumre for motorerne er defineret i et unit kaldet *data*. Kildekoden til *data* kan findes på side 68.

Input til procedure: Ingen
Output fra procedure: Målefladens azimut- og hældningsvinkel

Proceduren *solvogn_outputfil* udføres kun hvis der skal foretages målinger på dæklag. I denne procedure oprettes en uddatafil til brug for skrivning af måledata. I toppen af denne fil udskrives tillige de parametre for målingen, der er indtastet af brugeren i proceduren *bruger_inddata*. Hvis der udføres effektivitetsmålinger på en solfanger eller der udføres målinger på Metset-kassen, skal brugeren selv sørge for lagring af disse data i en uddatafil ved modificering af eksisterende programmer.

Input til procedure: Brugerangivne parametre fra proceduren *bruger_inddata*
Output fra procedure: En tekst-fil på computerens harddisk hvor brugerens inddata for målingen er angivet

Proceduren *init_solvogn* bruges til at initialisere solvognen. Initialiseringen indebærer at aktuatorarmen, som solvognen er påspændt, forkortes mest muligt. Efter denne forkortelse køres solvognen ind til det sted på måleplanet, hvor målingen skal starte. Beregningen af denne kørsel foretages ud fra viden om pulstælleren, motoren og aktuatorarmen.

Input til procedure: Yderposition for solvogn når aktuatorarm er forkortet mest muligt. Angivet af brugeren i *bruger_inddata* som værende afstanden fra midten af målefladen til venstre begrænsning (*v_flade_bred*)
Output fra procedure: Solvognens placering på glideskinnen inden målinger startes

2.2 Funktions/procedurekald i proceduren *set_start_pos*:

Proceduren *set_start_pos* bruges til at sætte fladens orientering til en startposition, der er i nærheden af fladens ønskede orientering i første tidsstep. Begrundelsen for at sætte fladens orientering før første tidsstep er, at det første tidsstep ellers ville tage alt for lang tid, hvis konstruktionen eksempelvis skulle dreje en halv omgang eller mere. For at minimere denne drejning i første tidsstep sættes fladens orientering derfor til solens position med forbehold for vinkelforskydninger i de azimut- og hældningsretning. I proceduren *set_start_pos* udføres følgende funktioner og procedurer:

getdate
gettime
slet_linje

solpos
beregn_flade_orient
roter_flade

Beskrivelserne af funktionerne og procedurerne kan findes i kapitel 2.3 *Funktions/procedurekald i proceduren tidsstep*:

2.3 Funktions/procedurekald i proceduren *tidsstep*:

I løbet af et tidsstep gennemløbes proceduren *tidsstep*. Handlingerne i tidssteppet er normalt en positionering eventuelt efterfulgt af en dæklagsmåling. Hvis der foretages en dæklagsmåling, hvor solvognen bevæges i step, kan der dog godt forekomme flere kald af positioneringsrutinen i hvert tidsstep. I proceduren *tidsstep* kaldes følgende funktioner og procedurer:

nulstilcounter kaldes gennem *roter_flade*
getdate
gettime
slet_linje
solpos
beregn_flade_orient
roter_flade
i_vinkel

Hvis der er tale om en dæklagsmåling udføres tillige følgende procedurer:

screen3
solvogn_travers_kont eller *solvogn_travers_step*
closegraph

I proceduren *nulstilcounter* sendes en kommando til pulstællerkortet, om at den nuværende værdi af kortet skal sættes til 0. Hver gang der spørges på værdien, der er lagret i counteren (hvilket gøres i funktionen *gethpcount*) undersøges det, om denne værdi er i nærheden af den maksimalt tilladte værdi for counteren. Dette gøres for at forhindre et eventuelt overflow i pulstællerkortet. Et overflow ville ellers forekomme på et tidspunkt, da pulstællerkortet er sat til altid at tælle opad, men ikke kan tælle højere end 100000 pulser.

Input til procedure: Intet input

Output fra procedure: Intet output

Funktionerne *getdate* og *gettime* læser klokken fra computerens indbyggede ur. Dato og tidspunkt bruges i proceduren *solpos* til at bestemme solens position på himmelbuen. *Getdate* og *gettime* er indbygget i Borland Pascals standard-biblioteker.

Input til funktioner: Intet input
Output fra funktioner: Dato, måned og årstal og tidspunkt på dagen

Proceduren *slet_linje* sletter alt hvad der står i linjen med linjenummeret der angives som input til proceduren. Proceduren bruges til at "rense" den nederste linje på skærmen inden der sendes et nyt output.

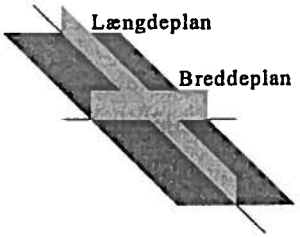
Input til funktion: Linjenummer
Output fra funktion: Sletning af linje

Proceduren *solpos* beregner solens højde og azimutvinkel ud fra kendskab til dato, måned, årstal, tidspunkt på dagen og længde- og breddegrad for måleopstillingen. Beregningsalgoritmerne er implementeret fra [1], hvor teorien bag algoritmerne tillige er forklaret. For at lette overskueligheden i programmet er følgende matematiske funktioner implementeret:

DegreeToRad	omregner en vinkel fra grader til radianer.
RadToDegree	omregner en vinkel fra radianer til grader
tan	beregner tangens til en vinkel
arcsin	beregner arcsin til et tal og returnerer en vinkel
arccos	beregner arccos til et tal og returnerer en vinkel
daynr	beregner dagens nummer i året

Input til procedure: Årstal, måned, dato, klokkeslæt
Output fra procedure: Solhøjde og solens azimut

Proceduren *beregn_flade_orient* bruges til at beregne den ønskede position for målefladen. Efter denne beregning er det muligt at rotere/vippe fladen til denne position. Proceduren er lavet, for at det skal være muligt at foretage tracking et antal grader "ved siden af" solens position. Indfaldsvinklen findes da som vinklen mellem målefladens normal og en vektor, der peger mod solen. Vektoren der peger mod solen kan forekomme i 0, 1 eller 2 planer. De to planer der er tale om, er et



Figur 2. Placering af planer

breddeplan og et længdeplan, hvis placering på målefladen kan ses af figur 2. Planernes matematiske egenskaber forklares yderligere i Appendix A. Går vektoren gennem både længdeplanet og breddeplanet, er målefladen placeret vinkelret på solstrålingen (normalstråling) og indfaldsvinklen er dermed 0 i begge retninger. Hvis vektoren til solen går gennem 1 plan, kan indfaldsvinklen findes, ved at iagttage det spor vektoren mod solen efterlader i planet. Hvis målefladen eksempelvis hældes, så den komplementære vinkel til fladens hældning med vandret er et antal grader større end solhøjden, ses det, at vektoren fra fladens midte mod solen ligger i længdeplanet. Ud fra sporet som vektoren mod solen efterlader kan indfaldsvinklen findes. På tilsvarende måde kan indfaldsvinkler ved azimutændring findes. Hvis vektoren til solen ikke går gennem hverken længdeplanet eller breddeplanet, projiceres vektoren ind på hver af de to planer, hvorefter indfaldsvinklerne i begge akseretninger kan findes. Indfaldsvinklerne i de to akseretninger kan herefter kombineres til en samlet indfaldsvinkel, hvis det ønskes.

Ved rotation omkring den vandrette akse, hældningsaksen, foregår denne tracking ved siden af solens position uden større problemer, idet vipningen foretages omkring en akse, der er i samme plan som målefladen. Ved en forskydning af målefladen i forhold til solen omkring en lodret akse er denne ikke helt så let at beregne. Dette skyldes at den lodrette omdrejningsakse ikke ligger i samme plan som målefladen, undtagen når målefladen står lodret. Ændres fladens azimutvinkel i forhold til solen ses det, at vektoren drejer ud af såvel det vandrette plan såvel som det lodrette. At vektoren mod solen drejer ud af det lodrette plan er ikke overraskende. Vektoren drejer dog også ud af det vandrette plan, hvilket ikke er ønsket, idet der kun var ønsket en "ren" azimutdrejning. For at få vektoren fra fladens midtpunkt mod solen ind i det vandrette plan igen er det nødvendigt med en "kompensationsvipning". Ud fra kendskab til solens højde og azimut, målefladens orientering og de ønskede indfaldsvinkler omkring de to akser er det muligt at beregne den ønskede "kompensationsvipning". Ved enhver positioneringsberegning opstilles og løses følgende ligningssystem:

$$
\begin{aligned}
&\tan(K_1) * \cos(\gamma_s) * \cos(\alpha_s) * \cos(\gamma_f) * \cos(\alpha_f) + \\
&\tan(K_1) * \sin(\gamma_s) * \cos(\alpha_s) * \sin(\gamma_f) * \sin(\alpha_f) + \\
&\tan(K_1) * \qquad \sin(\alpha_s) * \qquad \sin(\alpha_f) - \\
&\qquad \cos(\gamma_f) * \cos(\alpha_s) * \cos(\gamma_s) * \sin(\alpha_f) - \\
&\qquad \sin(\gamma_f) * \cos(\alpha_s) * \sin(\gamma_s) * \sin(\alpha_f) - \\
&\qquad \qquad \sin(\alpha_s) * \cos(\alpha_s) \qquad = 0
\end{aligned}
\tag{2.2.1}
$$

$$
\begin{aligned}
&- \tan(K_2) * \cos(\gamma_s) * \cos(\alpha_s) * \cos(\gamma_f) * \cos(\alpha_f) - \\
&\tan(K_2) * \sin(\gamma_s) * \cos(\alpha_s) * \sin(\gamma_f) * \sin(\alpha_f) - \\
&\tan(K_2) * \sin(\alpha_s) * \sin(\alpha_f) - \\
&\cos(\gamma_f) * \cos(\alpha_s) * \sin(\gamma_s) * - \\
&\sin(\gamma_f) * \cos(\alpha_s) * \sin(\gamma_s) * = 0
\end{aligned}
\qquad (2.2.2)
$$

Teorien bag ligningssystemet, der indeholder formel (2.2.1) og (2.2.2), kan findes bagest i programdokumentationen i Appendix A på side 33.

I ovenstående ligninger er fladens orientering beskrevet ved variablene α_f (fladens hældning) og γ_f (fladens azimut), og solens position på himmelhvælvingen er beskrevet som α_s (solhøjde) og γ_s (solens azimut). K_1 og K_2 er fladens ønskede vinkelforskydninger i henholdsvis højde- og azimut-retning. Ligningssystemet bestående af formel (2.2.1) og (2.2.2) kan ikke løses eksakt, men en numerisk løsning kan findes ved brug af Newton-Raphsons metode for ulineære ligningssystemer. Metoden, som er beskrevet i [2], er velkendt til løsning af ligninger i flere dimensioner. Denne løsningsmetode kræver udregning af de afledede værdier af de to funktioner, der kan opstilles, men disse udregnes nemt.

For at kunne udregne funktionsværdierne er det tillige nødvendigt at indsætte nogle startgæt for iterationen. Et kvalificeret startgæt for målefladens orientering er den position, som fladen allerede står i. Eneste undtagelse er ved det allerførste tidsstep i en måling, hvor fladen står i sin udgangsposition. Her bruges solens position (korrigeret med ønskede vinkelforskydninger). Grunden til at startgættene er nødt til at være i nærheden af den endelige løsning er, at et unøjagtigt startgæt kan medføre en forkert løsning eller i værste fald en instabilitet i konvergensen. Hvis en sådan instabilitet forekommer, kan det medføre (forkerte) resultater, der overstiger de maksimalt tilladelige talstørrelser i computeren. Et sådant overflow medfører at programudførelsen stopper, hvorefter soltrackingen også stopper.

De udregnede vinkler, der tilsammen giver fladens orientering, kan for simple tilfælde kontrolleres ved at indsætte de fundne positioner i udtrykket for beregning af indfaldsvinkel for en flade med vilkårlig orientering. Denne formel er opstillet nedenfor som formel (2.2.3). Ved afprøvning af de to opstillede formler findes det, at der er overensstemmelse mellem regneudtrykkene. En anden mulighed er at opsætte en skyggepind på målefladen. Ved hjælp af denne skyggepind er det muligt at aflæse den resulterende indfaldsvinkel resulterende fra en vilkårlig position af målefladen.

Ved brug af formel (2.2.3) kan indfaldsvinklen på en flade med vilkårlig orientering beregnes.

$$\begin{aligned}\cos\Theta = {} & \sin\delta * \sin\phi * \cos\beta & - \\ & \sin\delta * \cos\phi * \sin\beta * \cos\gamma & + \\ & \cos\delta * \cos\phi * \cos\beta * \cos\omega & + \\ & \cos\delta * \sin\phi * \sin\beta * \cos\gamma * \cos\omega & + \\ & \cos\delta * \sin\beta * \sin\gamma * \sin\omega\end{aligned} \qquad (2.2.3)$$

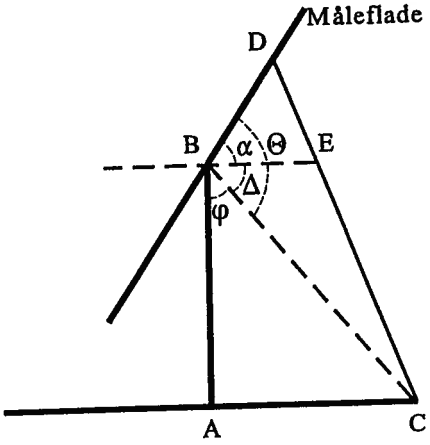
Input til procedure: Målefladens nuværende orientering, trackingtypen (rotation om 0, 1 eller 2 akser) og vinkelforskydningen i henholdsvis azimut og hældning (betegnet som K_1 og K_2).

Output fra procedure: Målefladens beregnede azimut- og hældningsvinkel

Proceduren *roter_flade* bruges til at rotere og vippe målefladen fra en position til en anden. Kriteriet for at en motor skal starte er, at forskellen mellem den nuværende orientering af målefladen og den beregnede nye orientering af målefladen er større end en grænseværdi, der er fastlagt i unit'et *data*. Overordnet består proceduren *roter_flade* af to dele.

I den første del af proceduren findes beregningerne, der styrer rotationen omkring den lodrette akse (azimut-drejning). Den ønskede drejning oversættes til et antal pulser, som kan tælles på HP-tællerkortet. Da tællerens status kendes inden drejningen påbegyndes, er det muligt at stoppe rotationen på det ønskede tidspunkt.

I den anden del af proceduren findes beregningerne, der styrer vipningen omkring den vandrette akse. Vipningen omkring den vandrette akse er ikke så ligefrem at beskrive som drejningen omkring den lodrette akse. Grundet den valgte udformning af dreje/vippe-mekanismen er det



Figur 3. Mål og vinkler på konstruktion

nødvendigt med en mindre omregning. Da aktuatorarmen har sit holdepunkt et stykke fra rammens omdrejningsakse, er antallet af pulser ved.en vipning ikke ligefrem afhængigt af drejnings-vinklen. Ved udregning af det ønskede antal pulser ved en vipning bruges derfor funktionen *puls_taeller*, som tager hensyn til denne ulinearitet. Bruges betegnelserne på figur 3, kan følgende udregninger foretages.

$$\phi = Arctan \frac{|AC|}{|AB|} \qquad \textbf{(2.2.4)}$$

$$\Delta = \frac{\pi}{2} - \phi \qquad \textbf{(2.2.5)}$$

$$\theta_0 = \alpha_0 + \Delta \qquad \textbf{(2.2.6)}$$

$$\theta_1 = \alpha_1 + \Delta \qquad \textbf{(2.2.7)}$$

$$|BC| = \sqrt{|AB|^2 + |AC|^2} \qquad \textbf{(2.2.8)}$$

$$|CD_0| = \sqrt{|BD|^2 + |BC|^2 - 2 * |BD| * |BC| * \cos(\theta_0)} \qquad \textbf{(2.2.9)}$$

$$|CD_1| = \sqrt{|BD|^2 + |BC|^2 - 2 * |BD| * |BC| * \cos(\theta_1)} \qquad \textbf{(2.2.10)}$$

$$|\Delta CD| = \left| |CD_1| - |CD_0| \right| \qquad \textbf{(2.2.11)}$$

I formel (2.2.6) og (2.2.9) henfører θ_0 og $|CD_0|$ til henholdsvis en vinkel og en længde før vipningen, mens θ_1 og $|CD_1|$ henfører til vinklen og længden efter vipningen.

Når længdeforskellen $|\Delta CD|$ på aktuatorarmen er kendt fra formel (2.2.11), kan det ønskede antal pulser fra pulstællerkortet beregnes, idet der er en lineær sammenhæng mellem aktuatorarmens længdeændring og det registrerede antal af pulser på pulstællerkortet.
Hvis der ønskes en omregning fra et antal pulser til en vinkel for målefladen, kan proceduren *vinkel_taeller* bruges. Princippet i udregningerne i denne procedure er som i proceduren *puls_taeller*, men udregningerne foregår baglæns i forhold til denne.

I det tilfælde at motoren kommer til at dreje/vippe målefladen for meget, er der i programmet indlagt muligheden for at bakke konstruktionen. Dette gøres, hvis målefladen har ramt mere end et vist antal grader forbi den ønskede position. Denne størrelse er indlagt i unit'et *data*. Programmet tillader ikke at motorens retning ændres mere end 3 gange i løbet af et tidsstep. Hvis den ønskede vinkel (på nær den tilladte usikkerhed) ikke er nået indenfor disse 3 korrektioner, forsøges ikke flere korrektioner i løbet af tidssteppet. På grund af den før omtalte ulinearitet mellem antallet af pulser og vinkelændring, er det igen nødvendigt at foretage en beregning så den nøjagtige orientering/hældning af målefladen kan bestemmes. Under disse udregninger bruges blandt andet proceduren *vinkel_taeller*, der er omtalt ovenfor.

Ved en transmittansmåling af dæklag efterfølges positioneringen af dæklagsmålingen. Ved dæklagsmålingen opbygges først en uddata-skærm, hvilket gøres med proceduren *screen3*. Denne procedure detekterer hvilket grafikkort, der sidder i PC-en og vælger derefter en grafiktilstand. På den tilsluttede PC bliver den valgte grafiktilstand 16-farver i en opløsning på 640x400 punkter. Efter at grafiktilstanden er valgt, opbygges grafikskærmen, hvor oplysninger med relation til forsøget anføres (forsøgsansvarliges navn, materialetype og eventuelle bemærkninger til forsøget). Desuden anføres målefladens orientering og solens position på himlen. I den nederste del af grafikskærmen optegnes et diagram, hvor det er muligt at se bestrålingsstyrkerne på de 3 solarimetre. For at kunne vise de 3 bestrålingsstyrker, er der foretaget en skalering af resultaterne, så der kun benyttes en y-akse i koordinatsystemet i stedet for flere.

Traverseringen af solvognen styres af en ud af to procedurer. Proceduren der bevæger solvognen i steps er kaldt *solvogn_travers_step*, mens proceduren der styrer den kontinuerte bevægelse hedder *solvogn_travers_kont*. I begge disse procedurer bevæges solvognen hen under dæklaget mens solarimetrene aflæses. Ved hver aflæsning af solarimetrene skrives de målte bestrålingsstyrker i en datafil på PC-ens harddisk, hvorefter de plottes i et koordinatsystem på skærmen. Dette gør det muligt at iagttage eventuelle uregelmæssigheder i måleserien. Uregelmæssigheder i

måleserierne kunne eksempelvis være skyer, som i en kort periode skyggede for solen. I sådan et tilfælde ville der ikke kunne opnås en konstant strålingsintensitet, hvilket der skulle tages højde for ved analysen af resultaterne. Denne plotning er kun gældende for den aktuelle måleserie, hvorfor det ikke er muligt at sammenligne den nuværende måleserie med en af de foregående.

Hvis der foretages en stepvis dæklagsmåling er der mulighed for at positionere målefladen i løbet af en traversering af solvognen. Denne funktioner er lavet, idet en solvognstraversering kan tage forholdsvis lang tid, hvis pyranometeret skal have tid til at foretage en indsvingning. For hvert step solvognen foretager beregnes solens position på himlen. For at afgøre om fladen skal roteres/vippes bruges samme startkriterier som i proceduren *roter_flade*.

Traverseringen stopper når solvognen når enten en brugerbestemt grænse eller ydergrænsen af sit bevægelsesområde. Når traverseringen stopper flyttes solvognen til startpositionen for næste måling, hvorefter proceduren slutter. I proceduren *tidsstep* udføres dernæst proceduren *Closegraph*, hvorved grafikskærmen lukkes og computeren sættes i tekstskærm-tilstand.

3. Procedurekald i transmittansmålingsprogram:

Det beskrevne unit er lavet, så det kan bruges til positionering af tre forskellige konfigurationer af måleopstillingen: Solfanger, Metset og dæklagsmålinger. Ved de to første er der dog ikke nogen egentlig forskel i positioneringsrutinerne. Disse positioneringsrutiner kan kaldes fra andre programmer, hvis de rigtige parametre anføres. I det følgende kan en skitse af hovedprogrammet til transmittansmålinger ses:

```
                                                                    1
program daeklag;

                                                                    2
uses                                                                3
  adatool,data,crt,dos,math,inddata,uddata;

                                                                    4
var                                                                 5
  MainPointerstatus: pointer;                                       6
  tracktid       : Integer;                                         7
  strttid,stoptid: Longint;                                         8
  t,m,s,h        : Word;

                                                                    9
begin                                                               10
  mark(MainPointerstatus);                                          11
  initier(flade_azimut,flade_haeld,
          Measure_type,Tracking_type,
          Batchparam,
          v_flade_bred,h_flade_bred,
          solvogn_step,solarimeter_indsving,
          solvogn_position,solvogn_maal_type,
          sim_data,tracktid);                                       12
  Batchparam:=TopBP;                                                13
  WHILE Batchparam<>NIL DO                                          14
  begin                                                             15
    antal_step:=Batchparam^.scanning;                               16
    delta_azimut :=batchparam^.delta_azimut;                        17
    delta_haeld  :=batchparam^.delta_haeld;                         18
    locked_azimut:=batchparam^.locked_azimut;                       19
    locked_haeld :=batchparam^.locked_haeld;                        20
    IF (Measure_type=3) AND (Tracking_type=3) THEN
    begin
      gotoxy(1,1);
      Writeln('Skyggeklodsen foran diffusstrålings-solarimeteren skal flyttes');
      Writeln('Ny hældningsforskydning : ',delta_haeld:6:2);
      Writeln('Ny azimutforskydning    : ',delta_azimut:6:2);
      Writeln('Programudførsel pauset ...');
      Write  ('Tryk <RETUR> når skyggeklodsen er sat i rigtig position ');
      readln;
    end;
    set_start_pos(flade_azimut,flade_haeld,delta_azimut,delta_haeld,    21
                  locked_azimut,locked_haeld);
                                                                    22
    gettime(t,m,s,h);                                               23
    stoptid:=3600*t+60*m+s;                                         24
    starttid:=stoptid;                                              25
    REPEAT                                                          26
      stoptid:=stoptid+tracktid;                                    27
      antal_step:=antal_step-1;
```

```
    tidsstep(measure_type,flade_azimut,flade_haeld,                    28
             delta_azimut,delta_haeld,locked_azimut,locked_haeld,solvogn_maal_type,
             v_flade_bred,h_flade_bred,sim_data,scan_retning,solvogn_position);
    vent_tid_diff(starttid,stoptid,antal_step);                        29
  UNTIL (antal_step=0) AND (Batchparam^.scanning<>0);                  30
  Batchparam:=Batchparam^.naeste;                                      31
 end; {WHILE Batchparam}                                               32
 release(MainPointerstatus);                                           33
end.                                                                   34
```

Kommentar til program *daeklag*:

1: Programstart.

2: Afsnit med erklæringer af brug af andre program-biblioteker.

3: Brug af andre program-biblioteker.

4: Variabelerklæringer. Disse foretages i linje 5, 6, 7 og 8.

5: Pointer, der bruges til at frigøre lagerplads brugt af andre pointere. Bruges i linje 10 og 33.

6: Varigheden af et tidsstep i sekunder.

7: Starttid og stoptid for måling udtrykt som antal sekunder siden klokken 00:00:00.

8: Time, minut, sekund og hundrededele af sekund. Bruges ved aflæsning af det indbyggede ur i computeren.

9: Start på selve programteksten.

10: Status af heapen for pointervariable gemmes. Dette gør, at alle pointere efter denne kommando kan slettes ved hjælp af *release*. *Release*-kommandoen udføres i linje 33.

11: Der bedes om brugerinput. Målekortene nulstilles og målefladen drejes til udgangsposition. Proceduren *initier* er yderligere forklaret på side 20.

12: Pointeren sættes til at pege på toppen af pointerstakken, som indeholdende de ønskede vinkler.

13: Løkken fra linje 14 til linje 32 skal køre så længe der er elementer tilbage i pointerstakken, der indeholdende vinkler.

14: Løkken, der er defineret i linje 13, startes.

15: Antallet af tidssteps med de aktuelle vinkler indlæses fra pointer til variabel.

16: Vinkelforskydning i forhold til solen i azimutretning indlæses fra pointer til variabel.

17: Vinkelforskydning i forhold til solen i hældningsretning indlæses fra pointer til variabel.

18: Fast azimutvinkel indlæses fra pointer til variabel.

19: Fast hældningsvinkel indlæses fra pointer til variabel.

20: Hvis der foretages en dæklagsmåling og der trackes om begge akser skal skyggeklodsen flyttes. Programudførelsen standser indtil brugeren har ændret på skyggeklodsen på det solarimeter, der skal måle diffus solstråling. Idet brugeren godkender at denne ændring

er foretaget fortsætter program-udførslen.

21: Fladen drejes til solens position korrigeret med eventuelle vinkelforskydninger i såvel aizmutretning som hældningsretning. Proceduren *set_start_pos* er yderligere forklaret på side 22.

22: Aflæser det interne ur i computeren.

23: Optæller hvor mange sekunder der er gået siden midnat.

24: Sætter starttidspunkt for første tidsstep. Tidspunktet er aflæst i programlinje 22 og er beregnet i linje 23.

25: Linje 26-29 gentages indtil det ønskede antal tidssteps er nået. Hvis der ønskes uendelig mange tidssteps, kører denne løkke indtil brugeren afbryder målingen.

26: Stoptidspunkt øges med et tidsteps længde.

27: Der er påbegyndt et tidsstep mere. Variablen *antal_step* styrer hvor mange tidssteps der mangler. Denne tælles derfor en ned idet der er påbegyndt et tidsstep.

28: Ud fra tidspunktet beregnes solens position. Kombineret med de brugerønskede vinkler, kan fladens nye position beregnes. Målefladen roteres og eventuelt foretages der en dæklagsmåling. Proceduren *tidsstep* er yderligere forklaret på side 24.

29: Proceduren *vent_tid_diff* søger for at computeren venter indtil det næste tidsstep skal starte. I mellemtiden udskrives ventetiden til skærmen.

30: Løkken, som er påbegyndt i linje 25, gentages indtil der ikke er flere vinkelparametre. Hvis antallet af scanninger er valgt til 0, fortsætter løkken uendeligt.

31: Pointeren, der indeholder de brugerbestemte vinkelparametre, sættes til at pege på næste data-sæt.

32: Løkken, som er påbegyndt i linje 13 stopper, hvis stopkriteriet er opfyldt.

33: Lagerplads, der er brugt til at oprette pointer-strukturer siden *mark*-kommandoen i linje 10, frigives ved kommandoen *release*.

34: Programmet er færdigt.

3.1 Variabelforklaring til procedure *initier*:

I proceduren *initier* initialiseres måle- og pulstællerkort i HP-relæboksen. Efter initialiseringen af målekort skal brugeren indtaste de værdier (vinkler mv.) som målingen ønskes foretaget med. Endeligt sættes målefladen i en kendt udgangsposition. Udgangspositionen er bestemt af placeringen af endestoppene på konstruktionen.

```
                                                                    1
procedure initier(                                                  2
            VAR flade_azimut,                                       3
                flade_haeld      : Real;                            4
            VAR Measure_type,                                       5
                Tracking_type    : Shortint;
```

```
            VAR Batchparam       : VinkelPointerType;                    6
            VAR v_flade_bred,                                            7
                h_flade_bred,                                            8
                solvogn_step,                                            9
                solarimeter_indsving,                                    10
                solvogn_position : Real;                                 11
            VAR solvogn_maal_type: Shortint;                             12
            VAR Usr_data         : Usr_dat_t;                            13
            VAR tracktid         : Integer);                             14

begin
  if NOT initadatool THEN                                                15
  begin
    writeln('Fejl i initialisering af måle/tælle-kort');
    halt;
  end;
 bruger_inddata(measure_type,tracking_type,Batchparam,                   16
                      v_flade_bred, h_flade_bred,
                      solvogn_step,solarimeter_indsving,
                      solvogn_maal_type,sim_data,tracktid);
  nulstil_flade(flade_azimut,flade_haeld);                               17
  IF Measure_type=3 THEN                                                 18
  begin
    solvogn_outputfil(measure_type,tracking_type,                        19
                      v_flade_bred,h_flade_bred,
                      solvogn_step,solarimeter_indsving,
                      solvogn_maal_type,Usr_data,tracktid);
    init_solvogn(v_flade_bred,solvogn_position);                         20
 end;
end; {initier}
```

Kommentarer til proceduren *initier*:

1: Procedurenavn.

2: Procedurehoved: Fladens azimutvinkel.

3: Procedurehoved: Fladens hældningsvinkel.

4: Procedurehoved: Måletype (1=Solfanger, 2=Metset, 3=Dæklag).

5: Procedurehoved: Trackingtype (1-4):

1: Der trackes kun omkring hældningsaksen - IKKE omkring azimutakse.

2: Der trackes kun omkring azimutaksen - IKKE omkring hældningsakse.

3: Der trackes både omkring hældningsakse og azimutakse.

4: Der trackes ikke omkring nogle af akserne. Måleopstilligen står stille.

6: Pointerstruktur bestående af de vinkler der skal trackes med og antallet af tidssteps med de enkelte vinkler.

Datastruktur for VinkelPointerType:

```
VinkelPointerType= ^VinkelPointer;
VinkelPointer    = RECORD                    {hvilke vinkler der skal trackes med}
                     naeste        : VinkelPointerType;
                     delta_azimut,           {Indfaldsvinkel i Azimutretning}
                     delta_haeld,            {Indfaldsvinkel i Hældningsretning}
                     locked_azimut,          {Låst vinkel i azimutretning}
                     locked_haeld  : Real;   {Låst vinkel i hældningsretning}
```

```
    scanning      : LongInt;  {Scanningsantal med gældende vinkel}
  end;
```

7: Venstre begrænsning af måleområde ved dæklagsmålinger. Angivet i millimeter. Nulpunkt er midt på glideskinnen.

8: Højre begrænsning af måleområde ved dæklagsmålinger. Angivet i millimeter. Nulpunkt er midt på glideskinnen.

9: Længde af steps ved solvognstraversering. Angivet i millimeter.

10: Tidsrum til indsvingning af solarimetre. Pausen skyldes at solarimetrene har brug for nogen tid til at tilpasse sig til de aktuelle strålingsforhold.

11: Solvognens position på glideskinnen. Angivet i millimeter. Nulpunkt er midt på glideskinnen.

12: Bevægelsesmønster for solvogn: 1=Stepmålinger, 2=Kontinuerte målinger.

13: Data-post bestående af navn på forsøgsansvarlig mv. Denne post bruges når oplysningerne skal udskrives på skærmen eller i en datafil på computerens harddisk.

Datastruktur for usr_dat_t:

```
Usr_dat_t = RECORD
              Navn,
              Materiale,
              Bemaerkning : String[60];
            END;
```

14: Længde af tidsstep i sekunder.

15: Hvis måle/tælle-kort ikke kan initialiseres, afbrydes afviklingen af programmet.

16: Brugerindtastning af nødvendige værdier inden for målingen. Forklaringer til variabelnavne og datastrukturer kan findes i kommentarerne til programlinjerne 2-14.

17: Målefladen bevæges til en kendt udgangsposition. Målefladen stilles i lodret og roteres så langt mod nord-øst som det er muligt.

18: Hvis der er tale om en dæklagsmåling udføres linjerne 19 og 20. Ellers springes til bunden af proceduren, hvorfra kontrollen returneres til hovedprogrammet.

19: Der oprettes en uddatafil på computerens harddisk af brugerbestemt navn hvori alle de brugerindtastede værdier lagres.

20: Solvognen køres til venstre udgangsposition.

3.2 Variabelforklaring til procedure *set_start_pos*:

Proceduren *set_start_pos* bruges til at sætte målefladen i en startposition, hvorfra målingerne kan påbegyndes. Proceduren er lavet for at kunne dreje konstruktionen hen i nærheden af hvor den skal være, inden tidssteppene går i gang. Hvis konstruktionens orientering er for langt væk fra det ønskede tager det for lang tid at dreje den, og dermed er det ikke sikkert at de ønskede tidssteps kan holde. Startposition findes ud fra solens nuværende position på himlen.

```
procedure set_start_pos(                                                    1
            VAR flade_azimut,                                               2
                flade_haeld     : Real;                                     3
            delta_azimut,                                                   4
            delta_haeld,                                                    5
            locked_azimut,                                                  6
            locked_haeld        : Real);                                    7
var                                                                         8
  yy, mon, dat, dow,
  time, min, sek, hund     : word;
  alfa_s_deg,
  azimut_deg               : real;

begin
  getdate(yy,mon,dat,dow);                                                  9
  gettime(time,min,sek,hund);                                               10
  sunheight(yy, mon,dat,time,min,sek,hund,alfa_s_deg,azimut_deg,            11
            timevinkel,deklination_deg);
  beregn_flade_orient(FALSE,delta_azimut,delta_haeld,locked_azimut,locked_haeld,  12
                      alfa_s_deg,azimut_deg,flade_azimut,flade_haeld,
                      tracking_type,ny_flade_azimut,ny_flade_haeld);
  roter_flade(ny_flade_azimut,ny_flade_haeld,flade_azimut,flade_haeld);     13
end; {set_start_pos}
```

Kommentarer til proceduren *set_start_pos*:

1: Procedurestart.

2: Procedurehoved: Fladens azimut.

3: Procedurehoved: Fladens hældning.

4: Procedurehoved: Vinkelforskydning i azimutretning.

5: Procedurehoved: Vinkelforskydning i hældningsretning.

6: Procedurehoved: Eventuel fast azimutvinkel.

7: Procedurehoved: Eventuel fast hældningsvinkel.

8: Lokale variabelerklæringer.

9: Aflæser årstal, måned, dato og dag i ugen fra det indbyggede ur i computeren. Tidspunktet bruges til udregning af solens position i linje 11.

10: Aflæser time, minut, sekund og hundrededele sekund fra det indbyggede ur i computeren. Tidspunktet bruges til udrgning af solens position i linje 11.

11: Beregner solens højde, azimut og deklination ud fra kendskab til tidspunkt (dato, måned, år og tidspunkt på dagen) og den geografiske placering (længde- og breddegraf) af Solar-trackeren.

12: Beregner målefladens nye position ud fra solens position, kendskab til de brugerbestemte vinkelforskydninger i azimut- og hældningsretningen og eventuelle faste orienteringsvinkler for målefladen.

13: Roterer måleflade fra den nuværende orientering til den nye orientering.

3.3 Variabelforklaring til procedure *tidsstep*:

I proceduren *tidsstep* udføres alle de operationer, der skal foregå i løbet af et tidsstep. Først beregnes solens position, og ud fra kendskab til denne kan fladens nye position beregnes. Hvis der skal foretages dæklagsmålinger, skrives fladens orientering i en uddatafil på computerens harddisk efterfulgt af resultaterne fra en scanning af forsøgsmaterialet. Under denne scanning plottes data fra solarimetrene på skærmen samtidig med at de lagres i en uddatafil på harddisken.

```
procedure tidsstep                                                          1
          Measure_type          : Shortint                                  2
          VAR flade_azimut,                                                 3
              flade_haeld       : Real;                                     4
          delta_azimut,                                                     5
          delta_haeld,                                                      6
          locked_azimut,                                                    7
          locked_haeld          : Real;                                     8
          solvogn_maal_type     : Shortint;                                 9
          v_flade_bred,                                                     10
          h_flade_bred          : Real;                                     11
          usr_data              : usr_dat_t;                                12
          VAR scan_retning      : Boolean;                                  13
          VAR solvogn_position: Real);                                      14
                                                                            15
var
  yy, mon, dat, dow,
  time,min,sek,hund: word;
  alfa_s_deg,
  azimut_deg        : real;

begin                                                                       16
  getdate(yy,mon,dat,dow);                                                  17
  gettime(time,min,sek,hund);                                               18
  sunheight(yy, mon,dat,time,min,sek,hund,alfa_s_deg,azimut_deg,
            timevinkel,deklination_deg);
  beregn_flade_orient(FALSE,delta_azimut,delta_haeld,locked_azimut,locked_haeld,  19
                      alfa_s_deg,azimut_deg,flade_azimut,flade_haeld,
                      tracking_type,ny_flade_azimut,ny_flade_haeld);
  roter_flade(ny_flade_azimut,ny_flade_haeld,flade_azimut,flade_haeld);     20
  i_vinkel(deklination_deg, flade_haeld, flade_azimut, timevinkel,indfaldsvinkel);  21
  IF measure_type=3 THEN                                                    22
  begin                                                                     23
    append(out_fil);                                                        24
    writeln(out_fil,'Måling nummer: ',batchparam^.scanning-antal_step,
                    ' af ',batchparam^.scanning);                           25
    CASE tracking_type OF
      1 : begin
            Writeln(out_fil,'Indfaldsvinkel i solhøjde : ',delta_haeld:5:1);
            Writeln(out_fil,'Låst vinkel i azimut      : ',locked_azimut:5:1);
          end;
      2 : begin
            Writeln(out_fil,'Indfaldsvinkel i azimut   : ',delta_azimut:5:1);
            Writeln(out_fil,'Låst vinkel i solhøjde    : ',locked_haeld:5:1);
          end;
      3 : begin
            Writeln(out_fil,'Indfaldsvinkel i azimut   : ',delta_azimut:5:1);
            Writeln(out_fil,'Indfaldsvinkel i hældning : ',delta_haeld:5:1);
```

```
          end;
        4 : begin
              Writeln(out_fil,'Låst vinkel i azimut      : ',locked_azimut:5:1);
              Writeln(out_fil,'Låst vinkel i hældning    : ',locked_haeld:5:1);
            end;
      end; {CASE}
      writeln(out_fil);
      close(out_fil);                                                          26
      screen3(flade_haeld, flade_azimut,alfa_s_deg, azimut_deg,indfaldsvinkel, 27
              v_flade_bred,h_flade_bred,usr_data);
      CASE solvogn_maal_type OF                                                28
        1 : solvogn_travers_step(solvogn_position,delta_azimut,delta_haeld,    29
              locked_azimut,locked_haeld,tracking_type,flade_azimut,flade_haeld,scan_retning);
        2 : solvogn_travers_kont(solvogn_position,scan_retning);               30
      end;
      Closegraph;                                                              31
    end;
  end;
```

Kommentarer til procedure *tidsstep*:

1: Procedurestart.

2: Procedurehoved: Måletype (1=Solfanger, 2=Metset, 3=Dæklagsmåling).

3: Procedurehoved: Målefladens azimut.

4: Procedurehoved: Målefladens hældning.

5: Procedurehoved: Vinkelforskydning i azimutretning.

6: Procedurehoved: Vinkelforskydning i hældningsretning.

7: Procedurehoved: Eventuel fast azimutvinkel.

8: Procedurehoved: Eventuel fast hældningsvinkel.

9: Procedurehoved: Måletype for solvogn (1=step-måling, 2=kontinuert måling).

10: Procedurehoved: Venstre begrænsning af måleflade ved dæklagsmålinger. Angivet i millimeter. Nulpunktet er midt på glideskinnen.

11. Procedurehoved: Højre begrænsning af måleflade ved dæklagsmålinger. Angivet i millimeter. Nulpunktet er midt på glideskinnen.

12. Procedurehoved: Datarecord bestående af navn på forsøgsansvarlig, forsøgsmateriale og bemærkninger til udskrivning på såvel udataskærm som uddatafiler.

Datapost for usr_dat_t:

```
usr_dat_t = RECORD
              Navn,
              Materiale,
              Bemaerkning : String[60];
            END;
```

13: Procedurehoved: Scanningsretning ved solvognsmålinger. (TRUE=venstre mod højde, FALSE=højre mod venstre).

14: Procedurehoved. Solvognens position på glideskinnen. Angivet i milimeter. Nulpunktet er midt på glideskinnen.

15: Lokale variabelerklæringer.
16: Aflæser årstal, måned, dato og dag i ugen fra det indbyggede ur i computeren.
17: Aflæser time, minut, sekund og hundrededele sekund fra det indbyggede ur i computeren.
18: Beregner solens højde, azimut og deklination ud fra tidspunkt og kendskab til længde- og breddegraf for stedet, hvor konstruktionen står.
19: Beregner målefladens nye orientering ud fra solens position, kendskab til vinkelforskydninger og faste orienteringsvinkler.
20: Roterer måleflade fra frn nuværende orientering til den nye orientering.
21: Beregning af indfaldsvinkel ud fra fladens og solens position.
22: Hvis der skal foretages dæklagsmålinger udføres linje 23-31. Ellers springes til bunden af proceduren, hvorfra kontrollen returneres til hovedprogrammet.
23: Uddatafil på computerens harddisk åbnes for skrivning.
24: Tidsstepnummeret skrives i uddatafil.
25: Indfaldsvinkler og faste vinkler skrives i uddadafil.
26 Uddatafil lukkes, hvorved filen opdateres på computerens harddisk.
27: Uddataskærm til dæklagsmålinger opbygges.
28: Det afgøres om der foretages step-målinger eller kontinuerte målinger med solvognen. Hvis der foretages step-målinger, udføres linje 29 ellers udføres linje 30.
29: Step-måling med solvogn.
30: Kontinuert måling med solvogn.
31: Grafisk skærm lukkes efter at have været brugt i enten linje 29 eller 30.

4. Implementering af procedurer i solfangerprogram:

Der eksisterer allerede nu et program, der kan foretage målinger på solfangere. *Adatool* er her tænkt som en udvidelse af såvel dette som andre programmer, hvorved det bliver muligt at foretage udendørs målinger med positionering af målefladen. I det følgende afsnit vil det blive forklaret hvordan de eksisterende programmer skal ændres for at få udbytte af positioneringsalgoritmerne. Der er nedenfor vist en programskitse og givet forslag til hvilke kodelinjer der skal indsættes. Måleprocedurer fra solfangerprogrammet er ikke gengivet korrekt her, men er bare samlet nogle enkelte større procedurekald. På tilsvarende måde er det muligt at opbygge et program til Metset-målinger, hvor det er muligt at positionere målefladen.

Hvis det er ønsket, er det selvfølgelig muligt at kalde underprocedurerne direkte, istedet for gennem proceduren *tidsstep*. Syntaksen for disse procedurekal kan ses på side 24.

Kodelinjerne markeres med *** er de ekstra linjer, der skal indsættes i solfangerprogrammet for at kunn benytte positioneringsrutinerne.

```
program solfanger;                                                    1

uses                                                                  2
  solfang1, solfang2,                                                 3
  adatool,data,crt,dos,math,inddata,uddata;                           4       ***

begin                                                                 5
  inputsolfangerparametre;                                            6
  initialiser_solfanger_maalestand;                                   7
 bruger_inddata(measure_type,tracking_type,Batchparam,                8       ***
                     v_flade_bred,h_flade_bred,
                     solvogn_step,solarimeter_indsving,
                     solvogn_maal_type,usr_data,tracktid);
  nulstil_flade(flade_azimut,flade_haeld);                            9       ***
  Batchparam:=TopBP;                                                  10      ***
  antal_step    :=Batchparam^.scanning;                               11      ***
  delta_azimut :=Batchparam^.delta_azimut;                            12      ***
  delta_haeld  :=Batchparam^.delta_haeld;                             13      ***
  locked_azimut:=Batchparam^.locked_azimut;                           14      ***
  locked_haeld :=Batchparam^.locked_haeld;                            15      ***
                                        {Solfangerløkkes påbegynde}   16
    IF antal_step=Batchparam^.scanning THEN                           17      ***
      set_start_pos(flade_azimut,flade_haeld,                         18      ***
                    delta_azimut,delta_haeld,
                    locked_azimut,locked_haeld);
    tidsstep(measure_type,flade_azimut,flade_haeld,                   19      ***
             delta_azimut,delta_haeld,
             locked_azimut,locked_haeld,
             solvogn_maal_type,
             v_flade_bred,h_flade_bred,
             usr_data,scan_retning,solvogn_position);
    antal_step:=antal_step-1;                                         20      ***
    IF antal_step=0 THEN                                              21      ***
```

```
  begin                                                          22      ***
    Batchparam:=Batchparam^.naeste;                              23      ***
    IF Batchparam=NIL THEN                                       24
      Programstop;                                               25      ***
    antal_step:=Batchparam^.scanning;                            26      ***
    delta_azimut  :=batchparam^.delta_azimut;                    27      ***
    delta_haeld   :=batchparam^.delta_haeld;                     28      ***
    locked_azimut:=batchparam^.locked_azimut;                    29      ***
    locked_haeld :=batchparam^.locked_haeld;
  end;                                                           30
  Solfanger_maalinger;                                           31
                                        {Her slutter solfangerløkke}
                                                                 32
end.
```

Kommentarer til program *solfanger*:

1: Start af program til måling på solfanger kombineret med positioneringsrutiner

2: Definition af units til brug for programudførsel

3: Programmoduler fra solfangerprogram

4: Programmoduler fra trackingsystem

5: Start af programmets kodedel

6: Modul hvor brugeren har mulighed for at indsætte måleparametre, der relaterer til solfangeren

7: Modul hvor computeren kan initialisere eventuelle målekort mv.

8: Indtastningsprocedure der vedrører positioneringsmekanismerne i programmet

9: Målefladen roteres til en kendt udgangsposition

10: Pointeren sættes til at pege på toppen af pointerstakken, som indeholder de ønskede vinkler

11: Antallet af tidssteps for de nuværende vinkler indsættes i en anden variabel

12: Vinkelforskydning i azimutretning indsættes i en anden variabel

13: Vinkelforskydning i hældningsretning indsættes i en anden variabel

14: Eventuel fast vinkel i azimutretning indsættes i en anden variabel

15: Eventuel fast vinkel i hældningsretning indsættes i en anden variabel

16: Her påbegyndes en løkkestruktur som skal køre indtil målingen er færdig

17: Hvis det er første tidsstep med de nuværende vinkler, skal fladen sættes til en startposition i nærheden af der hvor fladen skal være efter det første tidsstep

18: Fladen sættes i en startposition, der er bestemt af solens position og de vinkelparametre, som brugeren har valgt

19: Rutiner til beregning af fladens position kalde gennem proceduren *tidsstep*. Rutiner til positionering af målefladen kaldes tillige gennem proceduren *tidsstep*. Nærmere forklaring til parametrene i procedurehovedet kan findes på 24 i afsnittet *Variabelforklaring til procedure tidsstep*.

20: Der er påbegyndt et tidsstep mere. Variablen antal_step styrer hvormange tidssteps, der mangler. Denne tælles derfor en ned, idet der er påbegyndt et tidsstep. Hvis der fra brugeren er specificeret at der ønsket uendelig mange tidssteps er antal_step=0 fra starten. Efter at linje 20 er udført er antal_step=-1, hvor løkkens stop-kriterium (antal_step=0) aldrig bliver opfyldt.

21: Hvis tidsstep=0 så er det ønskede antal tidssteps foretaget og der skiftes til næste vinkelsæt

22: Indlæser næste vinkelsæt i variablen *Batchparam*

23: Hvis det nye vinkelsæt, der er indlæst i linje 22, er NIL, så er elementet tomt. Hermed er der ikke flere vinkelsæt, hvorfor programmet skal stoppe

24: Programmet stopper. Datafiler skal lukkes og eventuelt kørende måleudstyr skal lukkes ned

25: Antallet af tidssteps for de nuværende vinkler indsættes i en anden variabel

26: Vinkelforskydning i azimutretning indsættes i en anden variabel

27: Vinkelforskydning i hældningsretning indsættes i en anden variabel

28: Eventuel fast vinkel i azimutretning indsættes i en anden variabel

29: Eventuel fast vinkel i hældningsretning indsættes i en anden variabel

30: Her skal målinger på solfangeren udføres. Dette er allerede implementeret i solfanger-programmet

31: Her slutter løkkestrukturen, der er påbegyndt i linje 16

32: Programudførelse stopper.

5. Beskrivelse af hardware

I forbindelse med opbygningen af Solar-trackeren har det været nødvendigt at lave nogle specielle dele for at få systemet til at fungere. De to ting, der er opbygget, er en samleboks til pulssignaler og en mobil skyggeklods til diffus-strålingssolarimeteret.

5.1 Samleboks til pulstæller

Samleboksen til pulstællerkortene er fremstillet, for at kunne tælle pulser fra alle 3 pulsgivere. Da der kun er plads til 5 indstikskort i HP-boksen, er det ikke muligt at reservere plads til 3 pulstællerkort, der ville være behov for. Dette skyldes, at der også skal være plads til 2 eller 3 målekort til måling af spændingsforskelle.

Grundet ovenstående problematik blev det besluttet at lave en samleboks, hvor signaler fra alle tre pulsgivere kunne analyseres. Som input til samleboksen haves signalerne fra de tre pulsgivere. Det resulterede output fra samleboksen er en kombination af signalerne, der sendes videre til pulstællerkortet. Ulempen ved denne konstruktion er, at det ikke er muligt at køre med mere end en motor af gangen, idet pulserne fra motorerne så vil blandes.

En opvejning af fordele og ulemper ved de to løsningsmetoder har gjort, at det er valgt at benytte samleboksen. Dette gør at der kun køres med en motor af gangen i programmet. Eneste undtagelse fra dette er i de perioder, hvor målefladen nulstilles. Når målefladen nulstilles er pulstællingerne ikke interessante, idet der kun foretages målinger på endestoppene.

5.2 Mobil skyggeklods til diffus-strålingssolarimeter

Ved udførelse af transmittansmålinger på dæklag foreteges målinger på 3 solarimetre. De to af solarimetrene måler henholdsvis total stråling på dæklaget og transmitteret stråling gennem dæklaget, mens det tredje solarimeter måler mængden af diffus stråling på dæklaget. Idet den total solstråling der rammer målefladen er summen af den direkte og den diffuse stråling kan den direkte stråling findes som forskellen mellem den total og den diffuse stråling. Den diffuse stråling findes ved at udelukke den direkte solstråling på solarimeteret, hvilket som regel gøres med en skyggering. Da solarimeteret ikke er stationært og dermed står vinkelret på solstrålingen, er det dog ikke muligt at benytte en skyggering i dette tilfælde.

I stedet for skyggeringen er der fremstillet en skyggeskive, der kan orienteres næsten frit. Idet

solarimeterets ydre glasdome har en ydre diameter på 50 millimeter, er skyggeskiven valgt til 52 millimeter i diameter. Skyggeskivens diameter skal være mindst lige så stor som glasdomens diameter for at modvirke refraktioner i glasset, og diameteren er valgt lidt større for at modvirke evt. usikkerhed ved positionering af Solar-trackeren. For at skygge for solens direkte stråling uden at fjerne for meget af den diffuse stråling er skyggeskiven placeret i en afstand fra solarimeteret, der er 10 gange større end skyggeskivens diameter. Dette forhold gør, at skiven fylder 5.7° (0.1 radian) set fra solarimeteret, hvilket er i overensstemmelse med [3]. For at minimere forstyrrelser i den diffuse stråling, der rammer prøveobjektet (solfanger, dæklag etc.), er føringsbuen, hvorpå skyggeskiven kan bevæges, lavet så lille som mulig (radius = 100 mm, tykkelse=3 mm). Skyggeskiven monteres på et metalrør, der kan bevæges frit, for at bringe den ud i den rette afstand fra solarimeteret. Slutteligt males skyggeskiven hvid på den side, der vender mod solen og sort på den side, der vender mod solarimeteret. Den hvide farve er for at modvirke opvarmning grundet solstråling, og den sorte farve er for at forhindre reflektioner af sollys ned mod solarimeteret.

Under en dæklagmåling stopper programudførelsen midlertidigt, hver gang der skal skiftes indfaldsvinkel om den lodrette eller vandrette akse. Brugeren bliver da bedt om at vippe og/eller dreje skyggeskivekonstruktionen indtil skyggeklodsen står i den ønskede position. Når skyggeklodsen er placeret korrekt på føringsbuen, og brugeren har meddelt computeren dette, fortsætter dæklagsmålingen.

6. Litteraturliste

[1] Jerry Møller Jensen
Solstråling. Undervisningsnotat
Kursus 6407 Passiv Solvarme
Laboratoriet for Varmeisolering, DTU, 1993

[2] William H. Press m. fl.
Numerical Recipes In Pascl. The Art of Scientific Computing
Cambridge University Press, 1989
ISBN 0521-37516-9

[3] A. J. Hunt, D. F. Grether and M. Wahlig
The Measurement og Circumsolar Radiation, An introduction to Meteorological Measurements and Data Handling for Solar Energy Applications
IEA Task IV - Development of an Insolation Handbook and Instrument Package
U.S. Department of Energy, 1980
DOE/ER-0084

[4] Murray R. Spiegel
Schaum's Outline Series. Mathematical Handbook of Formulas and Tables
McGraw-Hill Book Company, 1968

Appendix A. Positionering ved tracking med vinkelforskel

I det følgende er det forklaret hvordan regneudtrykkene, der er benyttet i afsnittet om positionering af tracking-systemet, er fremkommet.

A.1 Fremgangsmåde for beregninger

Fremgangsmåden kan groft set opbygges i 5 punkter:

1 - Først fastlægges et koordinat-system med nulpunkt på midten af den vandrette akse, hvorom vipning af målefladen foregår. De tre akser er vinkelrette på hinanden. X-aksen i koordinatsystemet er peger mod syd, Y-aksen peger mod øst og Z-aksen peger lodret op.

2 - Dernæst fastlægges to planer, som begge står vinkelret på målefladen. De to planer står desuden vinkelret på hinanden og benævnes breddeplanet og længdeplanet. Breddeplanet indeholder den vandrette akse (vippeaksen), mens længdeplanet ligger vinkelret på såvel målefladen (kaldet prøveplanet) som breddeplanet.

3 - Ud fra geometriske betragtninger kan et matematisk udtryk opstilles for de to planer. For at kunne opstilles de matematiske udtryk, der beskriver planerne, benyttes 3 "kendte" punkter for hvert plan: 0-punktet af koordinatsystemet, et punkt beliggende i prøveplanets fladenormal og et tredje punkt i hvert af de to planer. Det tredje punkt er beliggende i prøveplanet.

4 - Ved hjælp af retningsvektoren fra punktet (x,y,z)=(0,0,0) til solen kan et "solpunkt" fastlægges. Solpunktet er beliggende på vektoren fra koordinatsystemets nulpunkt mod solen og ligger i afstanden '1' fra koordinatsystemets nulpunkt.

5 - Vinklen mellem målefladens normal og solens retningsvektor kan herefter bestemmes. Forskydningen i azimut (vinkel mellem fladens normal og retningsvektoren mod solen projeceret på et vandret plan) bestemmes ud fra afstanden mellem solpunktet og længdeplanet. Forskydning (vinkel mellem fladens normal og retningsvektoren mod solen projeceret på et lodret plan) i hældningsvinkel bestemmes ud fra afstanden mellem solpunktet og breddeplanet. Hvis begge disse afstande er 0, er der tale om stråling vinkelret på målefladen. Er en af disse afstande 0, er der tale om forskydning i

enten azimut eller hældningsretningen alene og hvis begge afstande er forskellige fra nul, er der tale om forskydning omkring 2 akser.

A.2 Fastlæggelse af planer

Med betegnelserne 'v' for prøveplanet azimutvinkel (positiv mod vest, syd=0) og 'h' for hældning af prøveplan (vandret=0) fås af punkt 2 og punkt 3:

Breddeplan:

1. Punkt: (0,0,0)	Nulpunkt
2. Punkt: (sin v, cos v,0)	I prøveplanet på vippeaksen
3. Punkt: (cosv * sin h,-sin v * sin h,cos h)	I prøveplanets fladenormal

Længdeplan:

1. Punkt: (0,0,0)	Nulpunkt
2. Punkt: (cos v * cos h,-sin v * cos h,-sin h)	I prøveplanet i afstande 1 fra (0,0,0). Vinkelret på vippeakse
3. Punkt: (cosv * sin h,-sin v * sin h,cos h)	I prøveplanets fladenormal

A.3 Opstilling af planernes ligninger

Fra [4] haves i følge ligning (12.8), at et plans ligning kan skrives som:

$$Ax+By+Cz+D=0 \qquad \textbf{(A.3.1)}$$

Opstilles udtryk for A, B, C og D for breddeplanet fås ved brug af (12.10)

$$A = \begin{bmatrix} y_2 - y_1 & z_2 - z_1 \\ y_3 - y_1 & z_3 - z_1 \end{bmatrix} \qquad \textbf{(A.3.2)}$$

$$= (y_2 - y_1) * (z_3 - z_1) - (y_3 - y_1) * (z_2 - z_1)$$

$$= (\cos v - 0) * (\cos h - 0) - (-\sin v * \sin h - 0) * (0 - 0)$$

$$= \cos h * \cos v$$

$$B = \begin{bmatrix} z_2 - z_1 & x_2 - x_1 \\ z_3 - z_1 & x_3 - x_1 \end{bmatrix} \quad \textbf{(A.3.3)}$$

$$= (z_2 - z_1) * (x_3 - x_1) - (z_3 - z_1) * (x_2 - x_1)$$

$$= (0 - 0) * (\cos v * \sin h - 0) - (\cos h - 0) * (\sin v - 0)$$

$$= -\cos h * \sin v$$

$$C = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{bmatrix} \quad \textbf{(A.3.4)}$$

$$= (x_2 - x_1) * (y_3 - y_1) - (x_3 - x_1) * (y_2 - y_1)$$

$$= (\sin v - 0) * (-\sin v * \sin h - 0) - (\cos v * \sin h - 0) * (\cos v - 0)$$

$$= -\sin^2 v * \sin h - \cos^2 v * \sin h$$

$$= -\sin h \, (\sin^2 v + \cos^2 v)$$

$$= -\sin h$$

$$D = A * x_1 + B * y_1 + C * z_1 \quad \textbf{(A.3.5)}$$

Da (x_1, y_1 og z_1)=(0,0,0) fås D=0.

Ligningen for breddeplanet bliver da:

$$(\cos h * \cos v) * x + (-\cos h * \sin v) * y + (-\sin h) * z = 0 \quad \textbf{(A.3.6)}$$

Ud fra formel (A.3.6) kan afstanden mellem "solpunktet" og breddeplanet bestemmes ved hjælp af formel (12.13) i [4]. Formel (12.13) er vist her som formel (A.3.7).

$$Afstand = \frac{Ax_0 + By_0 + Cz_0 + D}{\pm\sqrt{A^2 + B^2 + C^2}} \quad \textbf{(A.3.7)}$$

$$= \frac{(\cos h * \cos v)x_0 + (-\cos h * \sin v)y_0 + (-\sin h)z_0}{\pm\sqrt{(\cos h * \cos v)^2 + (-\cos h * \sin v)^2 + (-\sin h)^2}}$$

$$= \frac{(\cos h * \cos v)x_0 + (-\cos h * \sin v)y_0 + (-\sin h)z_0}{\pm\sqrt{\cos^2 h * (\cos^2 v \sin^2 v) + \sin^2 h}}$$

$$= \frac{(\cos h * \cos v)x_0 + (-\cos h * \sin v)y_0 + (-\sin h)z_0}{\pm\sqrt{\cos^2 h + \sin^2 h}}$$

$$= (\cos h * \cos v)x_0 + (-\cos h * \sin v)y_0 + (-\sin h)z_0$$

På samme måde kan konstanterne A, B, og C (og D) udregnes for længdeplanet.

$$A = \begin{bmatrix} y_2 - y_1 & z_2 - z_1 \\ y_3 - y_1 & z_3 - z_1 \end{bmatrix} \qquad \textbf{(A.3.8)}$$

$$= (y_2 - y_1) * (z_3 - z_1) - (y_3 - y_1) * (z_2 - z_1)$$

$$= (-\sin v * \sin h - 0) * (\cos h - 0) - (-\sin v * \sin h - 0) * (-\sin h - 0)$$

$$= -\sin v * \cos^2 h - \sin v * \sin^2 h$$

$$= -\sin v$$

$$B = \begin{bmatrix} z_2 - z_1 & x_2 - x_1 \\ z_3 - z_1 & x_3 - x_1 \end{bmatrix} \qquad \textbf{(A.3.9)}$$

$$= (z_2 - z_1) * (x_3 - x_1) - (z_3 - z_1) * (x_2 - x_1)$$

$$= (-\sin h - 0) * (\cos v * \sin h - 0) - (\cos v * \sin h - 0) * (-\sin v * \cos h - 0)$$

$$= -\sin v * \cos^2 h - \sin v * \sin^2 h$$

$$= -\sin v$$

$$C = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{bmatrix} \quad \textbf{(A.3.10)}$$

$$= (x_2 - x_1) * (y_3 - y_1) - (x_3 - x_1) * (y_2 - y_1)$$

$$= (\cos v * \cos h - 0) * (-\sin v * \sin h - 0) - (\cos v * \sin h - 0) * (-\sin v * \cos h - 0)$$

$$= -\cos v * \cos h * \sin v * \sin h + \cos v * \cos h * \sin v * \sin h$$

$$= 0$$

Som før haves x_1, y_1, z_1 lig nul, hvilket giver D=0.

Ligningen for længdeplanet bliver da:

$$(-\sin v) * x + (-\cos v) * y = 0 \quad \textbf{(A.3.11)}$$

$$Afstand = \frac{Ax_0 + By_0 + Cz_0 + D}{\pm\sqrt{A^2 + B^2 + C^2}}$$

$$= \frac{(-\sin v)x_0 + (-\cos v)y_0}{\pm\sqrt{(-\sin v)^2 + (-\cos v)^2}} \quad \textbf{(A.3.12)}$$

$$= (-\sin v)x_0 + (-\cos v)y_0$$

Ud fra ligning (A.3.12) er det muligt at beregnes afstanden mellem solpunktet og længdeplanet. Geometrisk kan det vises, at afstanden fra solpunktet til længdeplanet kan udtrykkes som

$$Afstand_{solpunkt-længdeplan} = -\tan(\Delta\gamma) * \cos(\theta) \quad \textbf{(A.3.13)}$$

Ligeledes kan afstanden fra solpunktet til breddeplanet udtrykkes som

$$Afstand_{solpunkt-breddeplan} = \tan(\Delta\alpha) * \cos(\theta) \quad \textbf{(A.3.14)}$$

I formel (A.3.13) er $\Delta\gamma$ brugt som betegnelse for forskellen i mellem solhøjden og den komplementære vinkel til målefladens hældning. I formel (A.3.14) er $\Delta\alpha$ brugt som forskel i azimutvinkel mellem målefladens normal og solens retningsvektor. θ er rumvinklen mellem prøvefladens fladenormal og solens retningsvektor.

A.4 Rumvinkel mellem solpunktsvektor og fladenormal

For at finde denne rumvinkel opstilles vektorerne for prøvefladens fladenormal og solens retningsvektor.

Retningsvektoren fra punktet (0, 0, 0) til solen kan ved geometriske betragtninger udregnes til:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos\gamma_s * \cos\alpha_s \\ -\sin\gamma_s * \cos\alpha_s \\ \sin\alpha_s \end{bmatrix} \qquad \textbf{(A.4.1)}$$

Længden af denne retningsvektor er '1'.

Ligeledes kan prøveplanets fladenormal udregnes til:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos v * \sin h \\ -\sin v * \sin h \\ \cos h \end{bmatrix} = \begin{bmatrix} \cos\gamma_f * \sin(90 - \alpha_f) \\ -\sin\gamma_f * \sin(90 - \alpha_f) \\ \cos(90 - \alpha_f) \end{bmatrix} = \begin{bmatrix} \cos\gamma_f * \cos\alpha_f \\ -\sin\gamma_f * \cos\alpha_f \\ \sin\alpha_f \end{bmatrix} \qquad \textbf{(A.4.2)}$$

Under udregning af prøveplanets fladenormal er der foretaget substitutioner af 'v' med 'γ_f' og 'h' med '90 - α_f'.

Af [4] fås af formel (12.2) følgende værdier for variablerne l, m og n. Disse skal bruges til udregning af vinklen mellem de to vektorer.

Vektoren mod solpunktet giver:

$l_l = \cos\gamma_s * \cos\alpha_s$
$m_l = -\sin\gamma_s * \cos\alpha_s$
$n_l = \sin\alpha_s$

Vektoren der peger i fladenormalens retning giver:

$l_2 = \cos \gamma_f * \cos \alpha_f$
$m_2 = - \sin \gamma_f * \cos \alpha_f$
$n_2 = \sin \alpha_f$

Bruges formel (12.7) i [4] fås (A.4.3), hvor θ beskriver vinklen mellem de to vektorer.

$$\begin{aligned} \cos \theta &= l_1 * l_2 + m_1 * m_2 + n_1 * n_2 \\ &= \cos \gamma_s * \cos \alpha_s * \cos \gamma_f * \cos \alpha_f + \\ &\quad \sin \gamma_s * \cos \alpha_s * \sin \gamma_f * \cos \alpha_f + \sin \alpha_s * \sin \alpha_f \end{aligned} \qquad \textbf{(A.4.3)}$$

A.5 Forskydningen af azimut og hældning bestemmes

Afstanden fra breddeplanet til solpunktet kan nu bestemmes ved at sætte værdierne fra (A.4.1) ind i ligning (A.3.7). I udregningen substitueres følgende variable: h = 90 - α_f og v = γ_f. Herved fås:

$$\begin{aligned} & Afstand_{solpunkt-breddeplan} \\ &= (\cos h * \cos v) * x_0 + (- \cos h * \sin v) * y_0 + (- \sin h) * z_0 \\ &= (\cos (90 - \alpha_f) * \cos \gamma_f) * \cos \gamma_s * \cos \alpha_s + \\ &\quad (- \cos (90 - \alpha_f) * \sin \gamma_f) * (- \sin \gamma_s * \cos \alpha_s) + \\ &\quad (- \sin (90 - \alpha_f)) * \sin \alpha_s \\ &= \sin \alpha_f * \cos \gamma_f * \cos \gamma_s * \cos \alpha_s + \\ &\quad \sin \alpha_f * \sin \gamma_f * \sin \gamma_s * \cos \alpha_s - \cos \alpha_f * \sin \alpha_s \end{aligned} \qquad \textbf{(A.5.1)}$$

På tilsvarende måde kan afstanden fra længdeplanet til solpunktet bestemmes ved at indsætte værdierne fra (A.4.1) i ligning (A.3.12). I udregningen substitueres følgende variable: h = 90 - α_f og v = γ_f. Herved fås (A.5.2):

$$
\begin{aligned}
&Afstand_{solpunkt\text{-}længdeplan} \\
&= (-\sin v) * x_0 + (-\cos v) * y_0 \\
&= (-\sin \gamma_f) * \cos \gamma_s * \cos \alpha_s + (-\cos \gamma_f) * (-\sin \gamma_s * \cos \alpha_s) \\
&= -\sin \gamma_f * \cos \gamma_s * \cos \alpha_s + \cos \gamma_f * \sin \gamma_s * \cos \alpha_s
\end{aligned}
\qquad \textbf{(A.5.2)}
$$

Kombineres ligningerne (A.5.1), (A.4.3) og (A.3.14) kan den første sammenhæng mellem fladens azimut/hældning og solens position opstilles. I sammenhængen indgår blandt andet en konstant kaldet K_1. Denne konstant er forskydning i forhold til breddeplanet, hvilket vil sige, at den indeholder forskellen mellem komplementærvinklen til fladens hældning og solhøjden.

$$
\begin{aligned}
&(\sin \alpha_f * \cos \gamma_f * \cos \gamma_s * \cos \alpha_s) + (\sin \alpha_f * \sin \gamma_f * \sin \gamma_s * \cos \alpha_s) - \\
&(\cos \alpha_f * \sin \alpha_s) = \tan K_1 * (\cos \alpha_f * \cos \gamma_f * \cos \gamma_s * \cos \alpha_s) + \\
&\tan K_1 * (\cos \alpha_f * \sin \gamma_f * \sin \gamma_s * \cos \alpha_s) + \tan K_1 * (\sin \alpha_s * \sin \alpha_f)
\end{aligned}
\qquad \textbf{(A.5.3)}
$$

På tilsvarende måde kan ligningerne (A.5.2), (A.4.3) og (A.3.13) kombineres til at danne endnu en sammenhæng mellem solens position og fladens orientering. I denne sammenhæng indgår konstanten K_2, som er forskydning i længdeplanet, hvilket vil sige at den indeholder forskellen mellem fladens azimut og solens azimut.

$$
\begin{aligned}
&(\cos \gamma_f * \sin \gamma_s * \cos \alpha_s) - (\sin \gamma_f * \cos \gamma_s * \cos \alpha_s) = \\
&-\tan K_2 * (\cos \alpha_f * \cos \gamma_f * \cos \gamma_s * \cos \alpha_s) - \\
&\tan K_2 * (\cos \alpha_f * \sin \gamma_f * \sin \gamma_s * \cos \alpha_s) - \tan K_2 * (\sin \alpha_s * \sin \alpha_f)
\end{aligned}
\qquad \textbf{(A.5.4)}
$$

Da disse ligninger ikke umiddelbart kan løses analytisk er det nødvendigt at forsøge andre metoder. Den her anvendte metode til løsning af ligningssystemet bestående at to ligninger med to ubekendte er Newton-Raphson metoden i flere dimensioner. Denne metode kræver opstilling af ligningerne på formen $f_n(x_1, x_2, x_3, x_4, ...) = 0$ og udregning af de afledede funktioner for de enkelte ubekendte.

Organiseres (A.5.3) således at ligningen kan skrives på formen $f(\alpha_f, \alpha_s) = 0$ kan de afledede funktioner med hensyn til α_f og α_s beregnes. Disse opstilles som henholdsvis ligning (A.5.5), (A.5.6) og (A.5.7).

$$\begin{aligned} f(\alpha_f, \gamma_f) = {} & \tan K_1 * (\cos \alpha_f * \cos \gamma_f * \cos \gamma_s * \cos \alpha_s) + \\ & \tan K_1 * (\cos \alpha_f * \sin \gamma_f * \sin \gamma_s * \cos \alpha_s) + \\ & \tan K_1 * (\sin \alpha_s * \sin \alpha_f) - \\ & (\sin \alpha_f * \cos \gamma_f * \cos \gamma_s * \cos \alpha_s) - \\ & (\sin \alpha_f * \sin \gamma_f * \sin \gamma_s * \cos \alpha_s) + \\ & (\cos \alpha_f * \sin \alpha_s) = 0 \end{aligned} \qquad \textbf{(A.5.5)}$$

$$\begin{aligned} \frac{\partial f(\alpha_f, \gamma_f)}{\partial \alpha_f} = {} & -\tan K_1 * (\sin \alpha_f * \cos \gamma_f * \cos \gamma_s * \cos \alpha_s) - \\ & \tan K_1 * (\sin \alpha_f * \sin \gamma_f * \sin \gamma_s * \cos \alpha_s) + \\ & \tan K_1 * (\sin \alpha_s * \cos \alpha_f) - \\ & (\cos \alpha_f * \cos \gamma_f * \cos \gamma_s * \cos \alpha_s) - \\ & (\cos \alpha_f * \sin \gamma_f * \sin \gamma_s * \cos \alpha_s) - \\ & (\sin \alpha_f * \sin \alpha_s) \end{aligned} \qquad \textbf{(A.5.6)}$$

$$\begin{aligned} \frac{\partial f(\alpha_f, \gamma_f)}{\partial \gamma_f} = {} & -\tan K_1 * (\cos \alpha_f * \sin \gamma_f * \cos \gamma_s * \cos \alpha_s) + \\ & \tan K_1 * (\cos \alpha_f * \cos \gamma_f * \sin \gamma_s * \cos \alpha_s) + \\ & (\sin \alpha_f * \sin \gamma_f * \cos \gamma_s * \cos \alpha_s) - \\ & (\sin \alpha_f * \cos \gamma_f * \sin \gamma_s * \cos \alpha_s) \end{aligned} \qquad \textbf{(A.5.7)}$$

Organiseres (A.5.4) på formen $g(\alpha_f, \gamma_f) = 0$ kan de afledede funktioner med hensyn til α_f og γ_f beregnes. Disse opstilles som henholdsvis ligning (A.5.8), (A.5.9) og (A.5.10).

$$g(\alpha_f,\gamma_f) = -\tan K_2 * (\cos\alpha_f * \cos\gamma_f * \cos\gamma_s * \cos\alpha_s) - \tan K_2 * (\cos\alpha_f * \sin\gamma_f * \sin\gamma_s * \cos\alpha_s) - \tan K_2 * (\sin\alpha_s * \sin\alpha_f) - (\cos\gamma_f * \sin\gamma_s * \cos\alpha_s) + (\cos\alpha_s * \sin\gamma_f * \cos\gamma_s) = 0 \quad \textbf{(A.5.8)}$$

$$\frac{\partial g(\alpha_f,\gamma_f)}{\partial\alpha_f} = \tan K_2 * (\sin\alpha_f * \cos\gamma_f * \cos\gamma_s * \cos\alpha_s) + \tan K_2 * (\sin\alpha_f * \sin\gamma_f * \sin\gamma_s * \cos\alpha_s) - \tan K_2 * (\cos\alpha_f * \sin\alpha_s) \quad \textbf{(A.5.9)}$$

$$\frac{\partial g(\alpha_f,\gamma_f)}{\partial\gamma_f} = \tan K_2 * (\cos\alpha_f * \sin\gamma_f * \cos\gamma_s * \cos\alpha_s) - \tan K_2 * (\cos\alpha_f * \cos\gamma_f * \sin\gamma_s * \cos\alpha_s) + (\sin\gamma_f * \sin\gamma_s * \cos\alpha_s) + (\cos\gamma_f * \cos\gamma_s * \cos\alpha_s) \quad \textbf{(A.5.10)}$$

Indsættes ligningerne (A.5.5), (A.5.6), (A.5.7), (A.5.8), (A.5.9) og (A.5.10) i en procedure, der bruger Newton-Raphson-løsningsmetode, kan løsningerne aflæses som output fra denne procedure. Udover de ovenfor nævnte ligninger skal proceduren også have nogle andre oplysninger for at kunne gennemføre udregningerne. Da der er tale om en iterativ løsningsmetode, skal der angives et startgæt på en løsning og en nøjagtighed på løsningen.

Som startgæt er brugt målefladens nuværende position, da det må antages at målefladens position efter det nuværende tidsstep er i nærheden af positionen i det nuværende tidsstep. Stopkriteriet for iterationen er, at summen af de numeriske værdier af $f(\alpha_f,\gamma_f)$ og $g(\alpha_f,\gamma_f)$ er mindre end 0.001, eller at antallet af iterationer overstiger en fastsat værdi. Denne værdi er sat til 20. Det andet stopkriterie, der omhandler en maksimalt antal iterationer, er lavet for at forhindre uendelige løkker i programmet. Ved store indfaldsvinkler, hvor der er tale om vinkelforskydninger på 60 grader eller højere, kan det dog vise sig, at antallet af iterationer er utilstrækkeligt. Hvis dette er

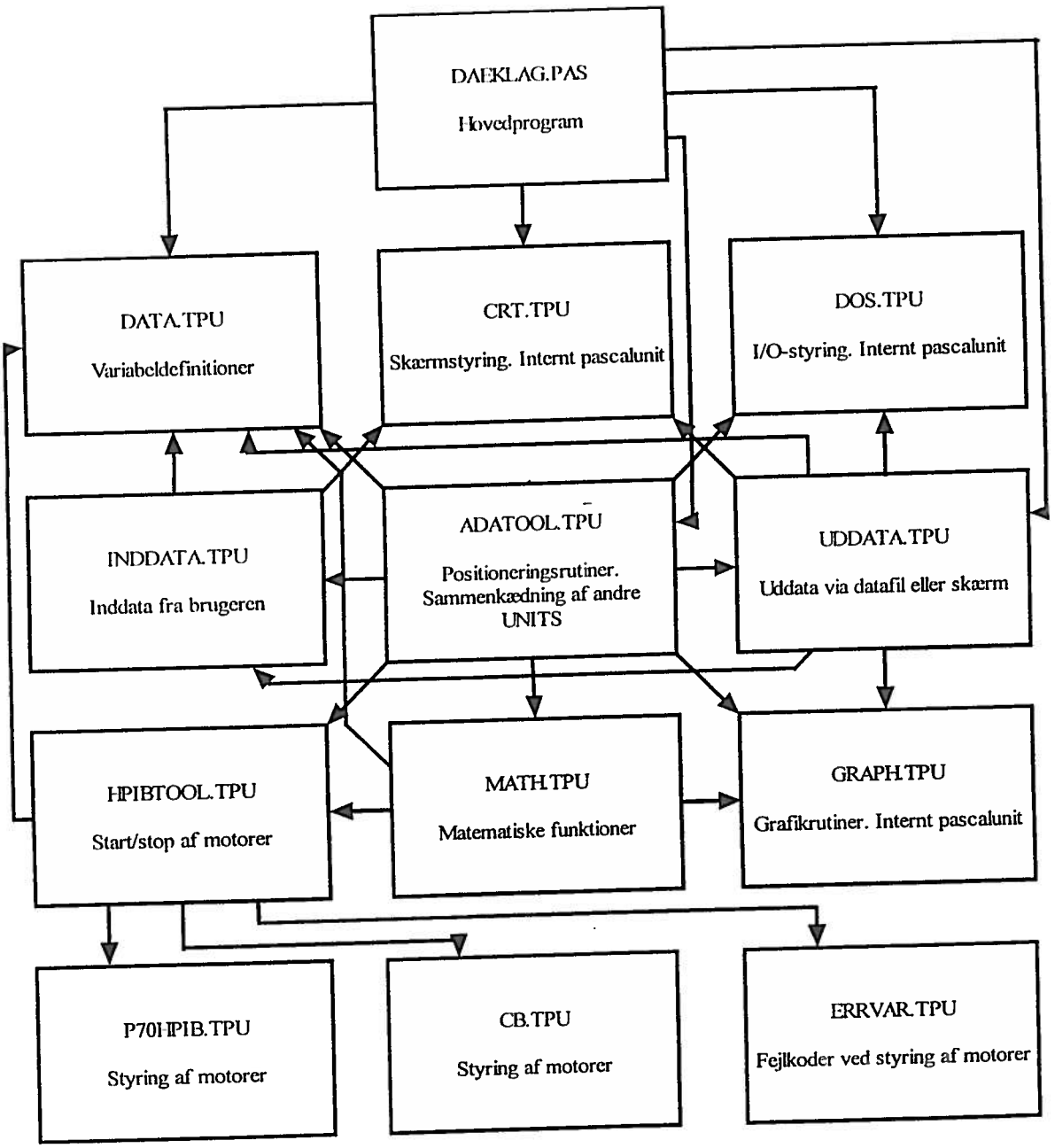
tilfældet, skal det maksimale antal iterationer øges ved kaldet af *mnewt* i proceduren *beregn_flade_orient*.

Ved de første beregninger viste det sig, at det var svært at få proceduren til at konvergere mod en løsning. Forløbet i løsningerne var kraftigt divergerende og uforudseligt. Da Newton-Raphson fremkommer med en ændring til det eksisterende løsningsforslag er disse ændringer gjort mindre, idet løsningsændringen er divideret med en fast værdi. Ved forsøg er denne værdi fastsat til 2. Ulempen ved denne løsningsmetode er, at der skal foretages flere iterationer (i gennemsnit dobbelt så mange) for en given ligningsløsning, og at proceduren dermed tager længere tid at udføre. Idet beregningstiden ikke i forvejen er særlig lang, er dette dog ikke vurderet som værende et stort problem. Ved en senere optimering af programmet kan der formodentlig spares noget beregningstid i denne del af programmet ved at iagttage konvergensmønsteret, men der er ikke brugt yderligere ressourcer på dette i projektet.

Appendix B. Kildekode til transmittans- og delprogrammer

I det følgende er vist kildekoden til programmet, der bruges til at foretage transmittansmålinger af dæklag. Tillige er kildekoden til alle underprogrammer (også kaldet units) vist, idet store dele af beregningsarbejdet er at finde i disse afsnit.

Sammenhængen mellem de enkelte underprogrammer kan ses af nedenstående figur:



Pilene mellem de enkelte underprogrammer angiver hvilke underprogrammer et andet underprogram bruger. Tages underprogrammet HPIBTOOl.TPU som eksempel, kan det ses af dette underprogram bruger P70HPIB.TPU, CB.TPU, ERRVAR.TPU og DATA.TPU. Til gengæld bliver det brugt af MATH.TPU og ADATOOL.TPU.

B.1 Kildekode til ADATOOL.TPU

```
unit adatool;

Interface

Uses
  hpibtool,crt,dos,graph,                    {Standard og HP biblioteker}
  data,                                      {Egen bibliotek til definition af globale variable}
  inddata,                                   {Egen bibliotek til inddata}
  uddata,                                    {Egen bibliotek til opbygning af uddata-skærme}
  math;                                      {Egen bibliotek til matematiske hjælpefunktioner}

procedure esc_maaling(                       {Afbryder tryk på 'ESC'-tasten under kørsel med motor}
            m_nummer1,m_nummer2 : ShortInt; {m_nummer1, m_nummer2 = relænummer for kørende
motorer}
          graphicstate        : Boolean); {Graphicstate: TRUE=grafikskærm, FALSE=txtskærm}

procedure solpos(                            {Beregner solhøjde og azimut ud fra tidspunkt}
                                             {samt stedlige koordinater}
          VAR alfa_s_deg,                    {Solhøjde}
              azimut_deg,                    {azimutvinkel i grader}
              timevinkel,                    {timevinkel}
              deklination_deg : Real);       {solens deklination i grader}

procedure i_vinkel(                          {Udregner indfaldsvinkel for en flade med}
            deklination_deg,                 {vilkårlig orientering}
            flade_haeld,                     {Fladens hældning}
            flade_azimut,                    {Fladens azimut}
            timevinkel          : Real;      {timevinkel}
            VAR indfaldsvinkel  : Real);     {Indfaldsvinkel}

procedure nulstil_flade(                     {Nulstiller målefladen. Fladen lodret-stilles og}
            VAR flade_azimut,                {drejes i negativ omdrejningsretning}
                flade_haeld     : Real);     {Fladens azimut og fladens hældning}

procedure puls_taeller(                      {Beregner antal pulser der skal tælles ved drejning}
            alfa_0,alfa_1       : Real;      {omkring den vandrette akse dvs. motor 1}
            VAR Vandret_puls    : LongInt;   {Fladehældning ved start og slut. Antal pulser der}
            VAR length_cd_0     : Real);     {skal køres. Længde af liniestykke CD}

procedure vinkel_taeller(                    {Beregner resulterende vinkel efter drejning på et}
            old_hpcount,                     {antal pulser}
            new_hpcount         : Real;      {Counts før/efter bevægelse}
            retning             : Byte;      {0=sammentrækning; 1=udstrækning}
            length_cd_0         : Real;      {Længden af liniestykke CD}
            VAR alfa_1          : Real);     {Resulterende fladehældning}

procedure roter_flade(                       {Roterer fladen til ny azimut/hældningsvinkler}
            ny_flade_azimut,                 {Ønsket fladeazimut}
            ny_flade_haeld      : Real;      {Ønsket fladehældning}
            VAR flade_azimut,                {Nuværende fladeazimut}
                flade_haeld     : Real);     {Nuværende fladehældning}

procedure beregn_flade_orient(               {Beregner hvor fladen skal orienteres ved næste mål.}
            graphicstate        : Boolean;   {Graphicstate: TRUE=grafikskærm, FALSE=txtskærm}
            alfa_s_deg,                      {Solhøjde}
            azimut_deg,                      {Solens azimutvinkel}
            flade_azimut,                    {Fladens azimut}
            flade_haeld         : Real;      {Fladens hældning}
            tracking_type       : Shortint; {Hvilken trackingtype der foretages}
```

```
            VAR ny_flade_azimut,            {Fladens azimut efter tidsstep}
                ny_flade_haeld  : Real);    {Fladens hµldning efter tidsstep}

procedure solvogn_outputfil(                {Opretter uddatafil til brug for solvognsmÕlinger}
            Measure_type,                   {MÕletype: Solfanger, Metset eller dµklag}
            Tracking_type       : Shortint; {Hvilken tackingtype der foretages}
            v_flade_bred,                   {MÕlefladens venstre begrµnsning}
            h_flade_bred,                   {MÕlefladens h°jre begrµnsning}
            solvogn_step,                   {Solvognens step mellem mÕlinger}
            solarimeter_indsving: Real;     {Tid for solarimetrene til indsvingning}
            solvogn_maal_type   : Shortint; {Kontinuert eller step-mÕling}
            Usr_data            : Usr_dat_t;{Navn pÕ fors°gsansvarlig mv.}
            tracktid            : Integer); {Tid mellem mÕleserier}

procedure init_solvogn(                     {Forkorter aktuatorstangen til solvogn mest muligt}
            v_flade_bred        : Real;     {MÕlefladens venstre begrµnsning}
            VAR solvogn_position: Real);    {Solvognens position efter initialisering}

procedure solvogn_travers_step(             {Steptraversering af solvogn + mÕling pÕ dµklag}
            VAR solvogn_position: Real;     {Solvognens position pÕ mÕlefladen}
            delta_azimut,                   {Vinkel-forskydning i azimut-retning}
            delta_haeld,                    {Vinkel-forskydning i hµldning-retning}
            locked_azimut,                  {Evt. lÕst azimut-vinkel}
            locked_haeld        : Real;     {Evt. lÕst hµldningsvinkel}
            tracking_type       : Shortint; {Hvilken trackingtype der foretages}
            VAR flade_azimut,               {MÕlefladens azimut}
                flade_haeld     : Real;     {MÕlefladens hµldning}
            VAR scan_retning    : Boolean); {Scanningsretning. TRUE = venstre mod h°jre}

procedure solvogn_travers_kont(             {Kont. traversering af solvogn + mÕling pÕ dµklag}
            VAR solvogn_position: Real;     {Solvognens position pÕ mÕlefladen}
            VAR scan_retning    : Boolean); {Scanningsretning. TRUE = venstre mod h°jre}

procedure initier(                          {Initierer HP-udstyr og drejer fladen til start}
            VAR flade_azimut,               {MÕlefladena azimut}
                flade_haeld     : Real;     {MÕlefladens hµldning}
            VAR Measure_type,               {MÕletype: Solfanger, Metset eller dµklag}
                Tracking_type   : Shortint; {Hvilken trackingtype der benyttes ved mÕlinger}
            VAR Batchparam      : VinkelPointerType; {delta/fast vinkler, antal scanninger}
            VAR v_flade_bred,               {Venstre begrµnsning af mÕleflade}
                h_flade_bred,               {H°jre   begrµnsning af mÕleflade}
                solvogn_step,               {Solvognens step mellem mÕlinger}
                solarimeter_indsving,       {Tid til indsvingning af solarimetre}
                solvogn_position : Real;    {Solvognens position}
            VAR solvogn_maal_type: Shortint;{Kontinuert eller step-mÕling}
            VAR Usr_data         : Usr_dat_t;{Navn pÕ fors°gsansvarlig mv.}
            VAR tracktid         : Word);   {Tid mellem mÕleserier}

procedure set_start_pos(                    {Sµtterfladensposition inden f°rste tidsstep}
            VAR flade_azimut,               {MÕlefladens azimut}
                flade_haeld     : Real;     {MÕlefladens hµldning}
            delta_azimut,                   {Vinkelforskydning i azimutretning}
            delta_haeld,                    {Vinkelforskydnin i hµldningsretning}
            locked_azimut,                  {Evt. lÕst azimutvinkel}
            locked_haeld        : Real);    {Evt. lÕst hµldningsvinkel}

procedure tidsstep(Measure_type : Shortint; {Handlinger i l°bet af 1 tidsstep}
            VAR flade_azimut,               {MÕlefladens azimut}
                flade_haeld     : Real;     {MÕlefladens hµldning}
            delta_azimut,                   {Vinkel-forskydning i azimut-retning}
            delta_haeld,                    {Vinkel-forskydning i hµldning-retning}
            locked_azimut,                  {Evt. lÕst azimut-vinkel}
```

```
            locked_haeld          : Real;      {Evt. lÕst hµldningsvinkel}
            solvogn_maal_type     : Shortint;  {Kontinuert eller step-mÕling}
            v_flade_bred,                      {Venstre begrµsning af mÕleflade}
            h_flade_bred          : Real;      {H°jre   begrµsning af mÕleflade}
            Usr_data              : Usr_dat_t;{Navn pÕ fors°gsansvarlig}
            VAR scan_retning      : Boolean;   {Scanningsretning. TRUE = venstre mod h"jre}
            VAR solvogn_position: Real);       {Solvognens position pÕ mÕlefladen}

implementation

procedure esc_maaling(                         {Aflµser tryk pÕ 'ESC'-tasten under k°rsel med motor}
            m_nummer1,m_nummer2 : Shortint; {m_nummer1, m_nummer2 = relµnummer for k°rende motor}
            graphicstate        : Boolean); {Graphicstate: TRUE=grafikskµrm, FALSE=txtskµrm}

var
  ch               : Char;                   {Aflµsning af Readkey}
  relae_count      : Shortint;               {Tµller af relµ-numre - Til nedlukning af relµboks}

begin
  IF Keypressed THEN                         {Hvis keyboardbufferen indeholder et tegn}
  begin
    ch:=Readkey;                             {Aflµs tegn i keyboardbuffer}
    IF ch=#27 THEN                           {Tegnet i keyboardbuffer er 'ESC'}
    begin
      setdio(m_nummer1,0);                   {Stop motor 'm_nummer1' mens bruger sp°rges}
      IF m_nummer2<>0 THEN                   {Er der ogsÕ en motor 'm_nummer2' der k°rer ?}
        setdio(m_nummer2,0);                 {Stopper motor 'm_nummer2}
      IF graphicstate THEN                   {SÕ er maskinen i grafikmode}
      begin
        TextColor(0);
        OuttextXY(0,0,'ESC: Afslut');
        TextColor(4);
        OuttextXY(0,0,'Ünskes mÕlingen stoppet? (J/N): ')
      end
      else                                   {SÕ er maskinen i textmode}
      begin
        gotoxy(1,1);
        Write('Ünskes mÕlingen stoppet? (J/N): ');
      end; {IF .. ELSE ..}
      repeat
        ch:=Readkey;                         {Omformer tastetryk til kapitalt bogstav}
        ch:=Upcase(ch);
      until ((ch='J') OR (ch='N'));          {Gentager aflµsening indtil der tastes 'J' eller 'N'}
      IF ch='J' THEN                         {SÕ skal mÕlingen stoppes}
      begin
        Closegraph;                          {Grafikskµrm lukkes - hvis i tekstmode sker intet}
        Stop_maaling;                        {Afslutter programmet}
      end
      else
      begin                                  {Fjerner skrevne tekst pÕ grafikskµrm}
        IF graphicstate THEN
        begin
          TextColor(0);
          OuttextXY(0,0,'Ünskes mÕlingen stoppet? (J/N) : ');
          TextColor(4);
          OuttextXY(0,0,'ESC: Afslut');
        end
        else                                 {Fjerner skrevne tekst pÕ tekstskµrm}
        begin
          gotoxy(1,1);
          Write('                                        ');
        end;
```

```
        setdio(m_nummer1,1);                {Her skal motor 'm_nummer1' startes igen}
        IF m_nummer2<>0 THEN
          setdio(m_nummer2,1);              {Her skal motor 'm_nummer2' startes igen}
      end;
    end; {IF ch=#27}
  end; {If keypressed}
end; {esc_maaling}

procedure solpos(                           {Beregner solh°jde og azimut ud fra tidspunkt samt}
                                            {stedlige koordinater}
            VAR alfa_s_deg,                 {Solh°jde}
                azimut_deg,                 {Solazimut}
                timevinkel,                 {Timevinkel=vinkel mellem meridianplan og sollinje}
                deklination_deg : real);    {Solens deklination i grader (3.1.2)}

const
  l_meridian=-15;                           {Longitude of time-meridian}

var
  k,                                        {Lokalkonstanten}
  fi,                                       {Steddets breddegrad i grader}
  w,u,w_deg,u_deg,
  w_u_deg, w_u,                             {Hjµlpevariable til (3.1.2)}
  deklination,                              {Solens deklination i radianer (3.1.2)}
  W_ew,                                     {Timevinkel nÕr solen er i stik vest (3.9.5)}
  B,                                        {Hjµlpest°rrelse til (3.2.1)}
  T_j,                                      {Tidsekvation i timer (3.2.1)}
  T_z,                                      {Lokaltid (zonetid)}
  T_s,                                      {Soltid (3.4.2)}
  alfa_s,                                   {Solh°jde (3.9.1)}
  Teta_z,                                   {Zenit-vinkel (3.8.1)}
  gamma_s_deg      : Real;                  {Solens azimut (3.9.2)}
  C1, C2, C3       : Shortint;              {Konstanter pÕ enten 0 eller 1 (3.9.4)-(3.9.7)}

begin
  fi:=DegreeToRad(latt);
  w_deg:=360/365.2422;
  w:=DegreeToRad(w);                        {Omformes til radianer}
  u_deg:=daynr(aar,mnd,dat)-79.301-0.2422*(aar-1969)+INT((aar-1969)/4);
  u:=DegreeToRad(u);                        {Omformes til radianer}
  w_u_deg:=w_deg*u_deg;
  w_u:=DegreeToRad(w_u_deg);
  deklination_deg:=0.3723+23.2567*sin(w_u)+0.1149*sin(2*w_u)-0.1712*sin(3*w_u)
                       - 0.7580*cos(w_u)+0.3656*cos(2*w_u)+0.0201*cos(3*w_u); {(3.1.2)}
  deklination:=DegreeToRad(deklination_deg);
  W_ew:=arccos(tan(deklination)/tan(fi));                                     {(3.9.5)}
  k:=4*(l_meridian-long)/60;                                                  {(3.4.1)}
  B:=DegreeToRad(daynr(aar,mnd,dat)-1)*360/365;
  T_j:=(229.2*(0.000075 + 0.001868*cos(  B)-0.030277*sin(  B)                 {(3.2.1)}
                      - 0.014615*cos(2*B)-0.04089 *sin(2*B)))/60;
  T_z:=tim+(min/60)+(sek/sqr(60))+(hun/(100*sqr(60)));
  T_s:=T_z+T_j+k;                                                             {(3.4.2)}
  Timevinkel:=DegreeToRad(15*(T_s-12));
  alfa_s:=arcsin((cos(fi)*cos(deklination)*cos(Timevinkel)+
                 sin(fi)*sin(deklination)));
  alfa_s_deg:=RadToDegree(alfa_s);
  Teta_z:=(pi/2)-alfa_s;
  gamma_s_deg:=RadToDegree(arcsin(sin(Timevinkel)*cos(deklination)/sin(Teta_z)));
  IF abs(Timevinkel)<=W_ew THEN C1:=1 else C1:=-1;
  IF (fi-deklination)>=0   THEN C2:=1 else C2:=-1;
  IF Timevinkel>=0         THEN C3:=1 else C3:=-1;
  azimut_deg:=C1*C2*gamma_s_deg+C3*((1-C1*C2)/2)*180;
```

```
end; {solpos}

procedure i_vinkel(                          {Udregner indfaldsvinkel. Vilkårlig orienteret flade}
            deklination_deg,                  {Solens deklination}
            flade_haeld,                      {Fladens hældning}
            flade_azimut,                     {Fladens azimut}
            timevinkel          : Real;       {Solens timevinkel}
            VAR indfaldsvinkel  : Real);      {Vinkel mellem fladenormal og solens retningsvektor}

var
  d, b, h,                                    {d=deklination, b=breddegrad, h=fladehældning}
  a, t              : Real;                   {a=fladeazimut, t=timevinkel}

begin
  d:=DegreeToRad(deklination_deg);            {Omregning fra grader til radianer}
  b:=DegreeToRad(latt);                       {Omregning fra grader til radianer. Fra unit 'data'}
  h:=DegreeToRad(flade_haeld);                {Omregning fra grader til radianer}
  a:=DegreeToRad(flade_azimut);               {Omregning fra grader til radianer}
  t:=timevinkel;                              {Er allerede omregnet til radianer}
  indfaldsvinkel:=Arccos(sin(d)*sin(b)*cos(h)-
                         sin(d)*cos(b)*sin(h)*cos(a)+
                         cos(d)*cos(b)*cos(h)*cos(t)+
                         cos(d)*sin(b)*sin(h)*cos(a)*cos(t)+
                         cos(d)*sin(h)*sin(a)*sin(t));
                                              {Udregning af indfaldsvinkel (3.7.1)}
  indfaldsvinkel:=RadToDegree(indfaldsvinkel);{Omregning fra radianer til grader}
end; {i_vinkel}

procedure nulstil_flade(                      {Mølefladen lodret-stilles samt drejes i TILBAGE}
            VAR flade_azimut,                 {Fladens azimut}
                flade_haeld     : Real);      {Fladens hældning}

var
  dvm_value         : Real;                   {Aflæst værdi fra Digitalvoltmeter}

begin
  dvm_taeller:=0;                             {Antallet af l°kkegennemgange nulstilles}
  dvm_value:=0;                               {Nulstiller værdi fra digitalvoltmeter}
  setdio(R4_M2,1);                            {Sætter motor 2 til at dreje TILBAGE}
  setdio(R3_M2,1);                            {Starter motor 2}
  setdio(R1_M1,1);                            {Sætter motor 1 til at dreje NED}
  setdio(R2_M1,1);                            {Starter motor 1}
  WHILE abs(dvm_value)<5 DO                   {Hvis voltmeter<5V så er endestoppet IKKE nået}
  begin
    dvm_taeller:=dvm_taeller+1;               {Antallet af l°kkegennemgange °ges med 1}
    IF dvm_taeller>=max_dvm_taeller THEN      {Der måles hvert max_dvm_taeller l°kkel°b}
    begin                                     {på DVM - slider ikke så meget på DVM}
      dvm_taeller:=0;                         {Nulstiller l°kkegennemgange siden sidste DVM-måling}
      dvm_value:=getadcchannel(stop_neg_1);   {Aflæser om neg. endestop er nået på motor 1}
    end;
    esc_maaling(R2_M1,R3_M2,FALSE);           {Aflæser ESC-tast}
  end; {WHILE dvm_value ...}
  setdio(R2_M1,0);                            {Endestop for motor 1 nået. Stopper motor 1}
  setdio(R1_M1,0);                            {Sætter motor 1 til at dreje OP}
  dvm_taeller:=0;                             {Antallet af l°kkegennemgange nulstilles}
  dvm_value:=0;                               {Nulstiller værdi fra digitalvoltmeter}
  WHILE abs(dvm_value)<5 DO                   {Hvis voltmeter<5V så er endestoppet IKKE nået}
  begin
    dvm_taeller:=dvm_taeller+1;               {Antallet af l°kkegennemgange °ges med 1}
    IF dvm_taeller>=max_dvm_taeller THEN      {Der måles hvert max_dvm_taeller l°kkel°b}
    begin                                     {på DVM - slider ikke så meget på DVM}
      dvm_taeller:=0;                         {Nulstiller l°kkegennemgange siden sidste DVM-måling}
```

```
      dvm_value:=getadcchannel(stop_neg_2); {Aflæser om neg. endestop er nået på motor 2}
    end;
    esc_maaling(R3_M2,0,FALSE);             {Aflæser ESC-tast}
  end; {WHILE dvm_value ...}
  setdio(R3_M2,0);                          {Stopper motor 2}
  setdio(R4_M2,0);                          {Sætter motor 2 til at dreje FREM}
  delay(motor_pause);
  flade_azimut:=min_azimut;                 {I forhold til referencepunkt}
  flade_haeld:=max_haeldning;               {Lodretstående flade}
end; {nulstil_flade}

procedure puls_taeller(                     {Beregner antal pulser ved drejning omkring v. akse}
            alfa_0,alfa_1       : Real;
            VAR Vandret_puls    : LongInt;
            VAR length_cd_0     : Real);

const                                       {Grafisk forklaring kan findes på tegning}
  length_ab = 1.66;                         {Længden af liniestykket AB}
  length_ac = 1.32;                         {Længden af liniestykket AC}
  length_bd = 1.13;                         {Længden af liniestykket BD}

var
  phi,                                      {Vinkel mellem BA og BC}
  delta,                                    {Vinkel mellem BC og BE}
  teta_0,                                   {Vinkel mellem BD og BC - situation 0 - radian}
  teta_1,                                   {Vinkel mellem BD og BC - situation 1 - radian}
  alfa_0_rad,                               {Vinkel mellem BD og BE - situation 0 - radian}
  alfa_1_rad,                               {Vinkel mellem BD og BE - situation 1 - radian}
  length_bc,                                {Længden af liniestykket BC}
  length_cd_1,                              {Længden af liniestykket CD - situation 1}
  length_cd_01      : Real;                 {Længdeforskel af liniestykket CD situation 0<->1}

begin
  alfa_0_rad:=DegreeToRad(alfa_0);          {Omregning til radianer}
  alfa_1_rad:=DegreeToRad(alfa_1);          {Omregning til radianer}
  phi:=arctan(length_ac/length_ab);         {Vinkel mellem BA og BC}
  delta:=(pi/2)-phi;                        {Vinkel mellem BC og BE}
  teta_0:=alfa_0_rad+delta;                 {Vinkel mellem BD og BC - situation 0 - radian}
  teta_1:=alfa_1_rad+delta;                 {Vinkel mellem BD og BC - situation 1 - radian}
  length_bc:=sqrt(sqr(length_ab)+sqr(length_ac));
  length_cd_0:=sqrt(sqr(length_bd)+sqr(length_bc)-2*length_bd*length_bc*cos(teta_0));
  length_cd_1:=sqrt(sqr(length_bd)+sqr(length_bc)-2*length_bd*length_bc*cos(teta_1));
  length_cd_01:=abs(length_cd_0-length_cd_1);{Længdeforskel af liniestykket CD situation 0<->1}
  vandret_puls:=round(length_cd_01*1000*M1_puls_length);
                                            {Antal pulser der skal drejes ved hældningsændring}
end; {puls_taeller_vandret}

procedure vinkel_taeller(                   {Beregner vinkel efter drejning på et antal pulser}
            old_hpcount,                    {Status på HP-counter inden drejning}
            new_hpcount         : Real;     {Status på HP-counter efter drejning}
            retning             : Byte;     {0=sammentrækning; 1=udstrækning}
            length_cd_0         : Real;     {Længden af liniestykke CD i situation 0}
            VAR alfa_1          : Real);    {Resulterende hældning af måleflade}

const                                       {Grafisk forklaring kan findes på tegning}
  length_ab = 1.66;                         {Længden af liniestykket AB}
  length_ac = 1.32;                         {Længden af liniestykket AC}
  length_bd = 1.13;                         {Længden af liniestykket BD}

var
  phi,                                      {Vinkel mellem BA og BC - radian}
  delta,                                    {Vinkel mellem BC og BE - radian}
```

```
  teta_1,                                   {Vinkel mellem BD og BC - situation 1 - radian}
  alfa_1_rad,                               {Vinkel mellem BD og BE - situation 0 - radian}
  length_bc,                                {Lµngden af liniestykket BC}
  length_cd_1,                              {Lµngden af liniestykket CD - situation 1}
  length_cd_01      : real;                 {Lµngdeforskel af liniestykket CD situation 0<->1}

begin
  CASE retning OF
    0 : length_cd_01:=(old_hpcount-new_hpcount)/(M1_puls_length*1000);
    1 : length_cd_01:=(new_hpcount-old_hpcount)/(M1_puls_length*1000);
                                            {Beregner lµngdeforskel mellem sit. 0 og 1 for CD}

    ELSE
    begin
      Clrscr;
      writeln('Fejl i variabel retning i funktion vinkel_taeller_vandret');
      writeln;
      writeln('MÕlingen stopper ...');
      Stop_maaling;                         {Stopper programudf°relse}
    end;
  end; {CASE}
  length_cd_1:=length_cd_0+length_cd_01;    {Beregner lµngde af CD i situation 1}
  length_bc:=sqrt(sqr(length_ab)+
                  sqr(length_ac));          {Beregner lµngde af BC}
  teta_1:=Arccos((sqr(length_bd)+sqr(length_bc)-sqr(length_cd_1))/(2*length_bd*length_bc));
  phi:=arctan(length_ac/length_ab);         {Vinkel mellem BA og BC - radian}
  delta:=pi/2-phi;                          {Vinkel mellem BC og BE - radian}
  alfa_1_rad:=teta_1-delta;                 {Vinkel mellem BD og BE - situation 1 - radian}
  alfa_1:=RadtoDegree(alfa_1_rad);          {Vinkel mellem BD og BE - situation 1 - grader}
end; {vinkel_taeller_vandret}

procedure roter_flade(                      {MÕleflade retter sig ind efter solposition}
            ny_flade_azimut,                {Fladens °nskede nye azimut}
            ny_flade_haeld      : Real;     {Fladens °nskede nye hµldning}
            VAR flade_azimut,               {Fladens nuvµrende azimut   - returnerer azimut}
                flade_haeld     : Real);    {Fladens nuvµrende hµldning - returnerer hµldning}

var
  d_count,                                  {Antal counts inden fladen nÕr rette position}
  old_hpcount,                              {Status af counteren inden drejningen foretages}
  new_hpcount       : LongInt;              {Nuvµrende status af HP-counteren}
  length_cd_0,                              {Lµngden af liniestykket fra C til D}
  dvm_value         : Real;                 {Aflµst vµrdi af digitalvoltmeteret}
  motorstart        : Byte;                 {Antal af motorstarter indenfor tidsstep}

procedure stop_tekst;
begin                                       {Udskriver fejltekst pÕ skµrm}
  Closegraph;                               {Lukker evt. grafisk skµrm}
  Clrscr;
  writeln('Der er ramt et endestop som ikke var forventet. Det tyder pÕ at der');
  writeln('er en usikkerhed ved positioneringen af fladen');
  writeln('Kontroller venligst at endestoppene sidder hvor de skal');
  writeln;
  writeln('MÕling stopper ...');
  writeln;
  stop_maaling;                             {Lukker relµer mv.}
end; {stop_tekst}

begin
  Motorstart:=0;                            {Antallet af motorstarter er 0}
  WHiLE ((abs(flade_azimut-ny_flade_azimut)>=Graddiff_azimut) AND (Motorstart<3)) DO
                                            {Startbetingelse for at azimutmotor mÕ starte}

  begin
```

```
IF (flade_azimut-ny_flade_azimut)>=Graddiff_azimut THEN
                                    {Motor 2 skal dreje TILBAGE}
begin
  d_count:=round(exp(motorstart*ln(0.5))*(flade_azimut-ny_flade_azimut)*M2_puls_length);
                                    {Vinkelforskel omregnes til counts}
                                    {Exp-led giver dµmpning af k°rselsafstand}
  old_hpcount:=round(gethpcount);   {Aflµser counter inden k°rsel}
  dvm_taeller:=0;                   {Nulstiller antallet af l°kkegennemgange}
  dvm_value:=0;                     {Nulstiller vµrdi fra DVM - pga. repeat-l°kke}
  setdio(R4_M2,1);                  {Angiver retning TILBAGE for motor 2}
  setdio(R3_M2,1);                  {Starter motor 2}
  repeat                            {K°r indtil stopbetingelse}
    dvm_taeller:=dvm_taeller+1;     {Antallet af l°kkegange °ges med 1}
    new_hpcount:=round(gethpcount); {Aflµser counter under k°rsel}
    IF dvm_taeller>=max_dvm_taeller THEN{Der mÕles hvert max_dvm_taeller l°kkel°b}
    begin
      dvm_taeller:=0;
      dvm_value:=getadcchannel(stop_neg_2);{Aflµser endestop i negativ retning for motor 2}
      IF abs(dvm_value)>5 THEN      {SÕ er der ramt et endestop}
        stop_tekst;                 {Stopper mÕling grundet fejl}
    end; {if dvm_taeller}
    esc_maaling(R3_M2,0,FALSE);     {Aflµser om der er trykket pÕ 'ESC'-tast}
  until (new_hpcount>=old_hpcount+d_count);{Stopbetingelse: Nok counts}
  setdio(R3_M2,0);                  {Stopper motor 2}
  setdio(R4_M2,0);                  {Angiver retning FREM for motor 2}
  delay(motor_pause);               {Pause til at bremse motoren}
  new_hpcount:=round(gethpcount);   {Sp°rger om status pÕ counteren efter motorstop}
  flade_azimut:=flade_azimut-(new_hpcount-old_hpcount)/M2_puls_length;
                                    {Beregning af ny azimut-vinkel}
  motorstart:=motorstart+1;         {Motor 2 har k°rt 'motorstart' gange i tidsstep}
end; {if-then}

IF (ny_flade_azimut-flade_azimut)>=Graddiff_azimut THEN
                                    {Motor 2 skal dreje FREM}
begin
  d_count:=round(exp(motorstart*ln(0.5))*(ny_flade_azimut-flade_azimut)*M2_puls_length);
                                    {Vinkelforskel omregnes til counts}
                                    {Exp-led giver dµmpning af k°rselsafstand}
  old_hpcount:=round(gethpcount);   {Aflµser counter inden k°rsel}
  dvm_taeller:=0;                   {Nustiller l°kkegennemgangstµller}
  dvm_value:=0;                     {Nulstiller vµrdi fra DVM pga. repeatl°kke}
  setdio(R4_M2,0);                  {Angiver retning FREM for motor 2}
  setdio(R3_M2,1);                  {Starter motor 2}
  repeat                            {K°r indtil stopbetingelse}
    dvm_taeller:=dvm_taeller+1;     {Antallet af l°kkegennemgange °ges med 1}
    new_hpcount:=round(gethpcount); {Aflµser counter under k°rsel}
    IF dvm_taeller>=max_dvm_taeller THEN{Der mÕles hvert max_dvm_taeller l°kkel°b}
    begin
      dvm_value:=getadcchannel(stop_pos_2);{Aflµser endestop i positiv retning for motor 2}
      dvm_taeller:=0;               {Nulstiller counter}
      IF abs(dvm_value)>5 THEN      {SÕ er der ramt et endestop}
        stop_tekst;                 {Stopper mÕling grundet fejl}
    end; {IF dvm_taeller}
    esc_maaling(R3_M2,0,FALSE);     {Aflµser om der er trykket pÕ 'ESC'-tast}
  until (new_hpcount>=old_hpcount+d_count);{Stopbetingelse: Nok counts}
  setdio(R3_M2,0);                  {Stopper motor 2}
  delay(motor_pause);               {Pause til at bremse motoren}
  new_hpcount:=round(gethpcount);   {Sp°rger om status pÕ counteren efter motorstop}
  flade_azimut:=flade_azimut+(new_hpcount-old_hpcount)/M2_puls_length;
                                    {Beregning af ny azimut-vinkel}
  motorstart:=motorstart+1;         {Motor 2 har h°rt 'motorstart' gange i tidsstep}
end; {if-then}
```

```
end; {While (abs(flade ......)

motorstart:=0;                          {Antallet af motorstarter i tidsstep nulstilles}
WHILE ((abs(flade_haeld-ny_flade_haeld)>=Graddiff_haeld) AND (motorstart<3)) DO
                                        {Startbetingelse for at hµldningsmotor mÕ starte}

begin
  IF (flade_haeld-ny_flade_haeld)>=Graddiff_haeld THEN
                                        {Motor 1 skal drejes OP}

  begin
    puls_taeller(flade_haeld,ny_flade_haeld,d_count,length_cd_0);
                                        {Vinkelforskel omregnes til counts}
    d_count:=round(d_count*exp(motorstart*ln(0.5)));
                                        {Exp-led giver dµmpning af k°rselsafstanden}
    old_hpcount:=round(gethpcount);     {Aflµser counter inden k°rsel med motor}
    dvm_taeller:=0;                     {Nulstiller l°kkegennemgange}
    dvm_value:=0;                       {Nulstiller vµrdi fra DVM pga. repeatl°kke}
    setdio(R1_M1,0);                    {Angiver retning OP for motor 1}
    setdio(R2_M1,1);                    {Starter motor 1}
    repeat                              {K°r indtil stopbetingelse}
      dvm_taeller:=dvm_taeller+1;       {Antallet l°kkegennemgange °ges med 1}
      new_hpcount:=round(gethpcount);   {Aflµs counter under k°rsel}
      IF dvm_taeller>=max_dvm_taeller THEN{Der mÕles hvert max_dvm_taeller l°kkel°b}
      begin
        dvm_value:=getadcchannel(stop_pos_1);{Aflµser positiv endestop for motor 1}
        dvm_taeller:=0;
        IF abs(dvm_value)>5 THEN        {SÕ er der ramt et endestop}
          stop_tekst;                   {Stopper mÕling grundet fejl}
      end; {IF dvm_taeller}
      esc_maaling(R2_M1,0,FALSE);     {Aflµser om der er trykket pÕ 'ESC'-tast}
    until (new_hpcount>=old_hpcount+d_count);{Stopbetingelse: Nok counts}
    setdio(R2_M1,0);                    {Stopper motor 1}
    delay(motor_pause);                 {Pause til at bremse motoren}
    new_hpcount:=round(gethpcount);     {Sp°rger om status pÕ counteren efter motorstop}
    vinkel_taeller(old_hpcount,new_hpcount,0,length_cd_0,flade_haeld);
                                        {Beregner fladens hµldning efter drejning}
    motorstart:=motorstart+1;           {Motor 1 har k°rt 'motorstart' i tidsstep}
  end; {if-then}

  if (ny_flade_haeld-flade_haeld)>=Graddiff_haeld THEN
                                        {Motor 1 skal drejes NED}

  begin
    puls_taeller(flade_haeld,ny_flade_haeld,d_count,length_cd_0);
                                        {Vinkelforskel omregnes til counts}
    d_count:=round(d_count*exp(motorstart*ln(0.5)));
                                        {Exp-led giver dµmpning af k°rselsafstand}
    old_hpcount:=round(gethpcount);     {Aflµser counter inden k°rsel}
    dvm_taeller:=0;                     {Nulstiller l°kkegennemgange}
    dvm_value:=0;                       {Nulstiller vµrdi fra DVM pga. repeatl°kke}
    setdio(R1_M1,1);                    {Angiver retning NED for motor 1}
    setdio(R2_M1,1);                    {Starter motor 1}
    repeat                              {K°r indtil stopbetingelse}
      dvm_taeller:=dvm_taeller+1;       {Antallet af l°kkegennemgange °ges med 1}
      new_hpcount:=round(gethpcount);   {Aflµs counter under k°rsel}
      IF dvm_taeller>=max_dvm_taeller THEN{Der mÕles hvert max_dvm_taeller l°kkel°b}
      begin
        dvm_taeller:=0;
        dvm_value:=getadcchannel(stop_neg_1);{Aflµser negativ endestop for motor 1}
        IF abs(dvm_value)>5 THEN        {SÕ er der ramt et endestop}
          stop_maaling;                 {Stopper mÕling grundet fejl}
      end; {IF dvm_taeller}
      esc_maaling(R2_M1,0,FALSE);       {Aflµser om der er trykket pÕ 'ESC'-tast}
    until (new_hpcount>=old_hpcount+d_count);{Stopbetingelse: Nok counts}
```

```
      setdio(R2_M1,0);                          {Stopper motor 1}
      setdio(R1_M1,0);                          {Angiver retning OP for motor 1}
      delay(motor_pause);                       {Pause til at bremse motoren}
      new_hpcount:=round(gethpcount);           {Sp°rger om status pÕ counteren efter motorstop}
      vinkel_taeller(old_hpcount,new_hpcount,1,length_cd_0,flade_haeld);
                                                {Beregner fladens hµldning efter drejning}
      motorstart:=motorstart+1;                 {Motor 1 har k"rt 'motorstart' i tidsstep}
    end; {if-then}
  end; {WHILE ((abs(flade ...)
end; {roter_flade}

procedure beregn_flade_orient(                {Beregner hvor fladen skal orienteres ved nµste mÕl.}
            graphicstate        : Boolean;    {Graphicstate: TRUE=grafikskµrm, FALSE=txtskµrm}
            alfa_s_deg,                       {Solh°jde}
            azimut_deg,                       {Solens azimutvinkel}
            flade_azimut,                     {Fladens azimut}
            flade_haeld         : Real;       {Fladens hµldning}
            tracking_type       : Shortint;   {Hvilken tracking-type der foretages mÕlinger med}
            VAR ny_flade_azimut,              {Fladens azimut efter tidsstep}
                ny_flade_haeld  : Real);      {Fladens hµldning efter tidsstep}

var
  az,                                         {Solazimut i radianer}
  alfa_s,                                     {solh°jde i radianer}
  d_azimut_rad,                               {Fladens azimutforskydning i radianer}
  d_haeld_rad       : Real;                   {Fladens hµldningsforskydninger i radianer}
  flade_orient      : RealArrayNP;            {Angiver fladens orientering (hµldning,azimut)}

begin
  az:=DegreeToRad(azimut_deg);                {Omregner fra grader til radianer}
  alfa_s:=DegreeToRad(alfa_s_deg);            {Omregner fra grader til radianer}
  IF Tracking_type=1 THEN                     {Hvis der kun trackes omkring vandret akse}
    delta_azimut:=locked_azimut-azimut_deg;   {Korrigerer da bevµgelse IKKE foregÕr omkring lodr.}
  IF Tracking_type=2 THEN                     {Hvis der kun trackes omkring lodret akse}
    delta_haeld:=locked_haeld-alfa_s_deg;     {Korrigerer da bevµgelse IKKE foregÕr omkring vandr.}
  IF (Tracking_type>=1) OR (Tracking_type<=3) THEN
  begin                                       {Tracking_type=1,2,3}
    d_azimut_rad:=DegreeToRad(delta_azimut);{Omregning af delta-vinkel fra grader til radian}
    d_haeld_rad :=DegreeToRad(delta_haeld); {Omregning af delta-vinkel fra grader til radian}
    flade_orient[1]:=DegreeToRad(flade_haeld); {Som startgµt pÕ fladens nye orientering bruges}
    flade_orient[2]:=DegreeToRad(flade_azimut);{fladens tidligere orientering}
    IF (flade_haeld=max_haeldning) AND (flade_azimut=min_azimut) THEN
    begin                                     {SÕ er det nµsten helt sikkert f°rste tidsstep}
      flade_orient[1]:=DegreeToRad((90-alfa_s_deg)+delta_haeld);
      flade_orient[2]:=DegreeToRad(azimut_deg+delta_azimut);
    end;                                      {Et startgµt pÕ fladens nye position er solens
                                               nuvµrende position korrigeret med delta_vinklerne}
    mnewt(20,flade_orient,2,0.001,0.001,d_haeld_rad,d_azimut_rad,alfa_s,az,graphicstate);
                                              {De opstillede ligninger l°ses vha. Newton-Raphson
                                               i multiple dimensioner med f°lgende param.:
                                               Maksimalt antal iterationer = 20,
                                              Gµt pÕ fladens orientering = fladens tidl. orient.,
                                               Antal dimensioner der regnes i = 2
                                               Korrektionsvinkel (radianer) for hµldning,
                                               Korrektionsvinkel (radianer) for azimut,
                                               Solens "h°jde" og azimutvinkel (radianer)}
    ny_flade_haeld :=RadToDegree(flade_orient[1]);
    ny_flade_azimut:=RadToDegree(flade_orient[2]);
    IF (ny_flade_haeld<min_haeldning) OR (ny_flade_haeld>max_haeldning) THEN
    begin                                     {Hµldningsvinkel er udenfor lovligt interval}
      Clrscr;
      writeln('Der er fors°gt en vipning udenfor det lovlige interval for hµldningsvinklen');
```

```
    writeln('Dette interval er fra ',min_haeldning,' til ',max_haeldning,' grader,');
    writeln('hvor programmet "nskede en vinkel pÕ: ',ny_flade_haeld:6:2);
    writeln('Problemet kan enten l"ses ved at starte mÕlingen tidligt pÕ morgenen/');
    writeln('sent om eftermiddagen, hvor solh"jden er lavere');
    writeln('eller ved at vµlge en mindre indfaldsvinkel');
    writeln;
    writeln('Programafvikling stopper');
    stop_maaling;                          {Stopper programudf"rsel}
  end;
  IF (ny_flade_azimut>max_azimut) OR (ny_flade_azimut<min_azimut) THEN
                                           {Azimutvinkel er udenfor lovligt interval}
  begin
    Clrscr;
    writeln('Der er fors"gt en drejning udenfor det lovlige interval for azimutvinklen');
    writeln('Dette interval er fra ',min_azimut,' til ',max_azimut,' grader,');
    writeln('hvor programmet "nskede en vinkel pÕ: ',ny_flade_azimut:6:2);
    writeln('Problemet kan enten l"ses ved at starte mÕlingen tidligt pÕ morgenen/');
    writeln('sent om eftermiddagen, hvor solh°jden er lavere');
    writeln('eller ved at vµlge en mindre indfaldsvinkel');
    writeln;
    writeln('Programafvikling stopper');
    stop_maaling;                          {Stopper programudf°rsel}
   end;
 end; {IF Tracking_type>=1 OR Tracking_type<=3}
 IF Tracking_type=4 THEN                   {StillestÕende konstruktion}
 begin
   ny_flade_azimut:=locked_azimut;         {Positionen er fastlÕst}
   ny_flade_haeld:=locked_haeld;           {Positionen er fastlÕst}
 end; {IF Tracking_type=4}
end; {beregn_flade_orient}

procedure solvogn_outputfil(               {Opretter uddatafil til brug for solvognsmÕlinger}
            Measure_type,                  {MÕletype: Solfanger, Metset eller dµklag}
            Tracking_type       : Shortint; {Hvilken trackingtype der foretages mÕlinger med}
            v_flade_bred,                  {MÕlefladens venstre begrµsning}
            h_flade_bred,                  {MÕlefladens h°jre   begrµsning}
            solvogn_step,                  {Solvognens step mellem mÕlinger}
            solarimeter_indsving: Real;    {Tid for solarimetre til indsvingning}
            solvogn_maal_type   : Shortint; {Kontinuert eller step-mÕling}
            Usr_data            : Usr_dat_t;{Navn pÕ fors°gsansvarlig mv.}
            tracktid            : Integer); {Tid mellem mÕleserier}

var
  out_navn         : STRING[80];           {Path + navn pÕ outputfil}
begin
  clrscr;
  gotoxy(10,10);Write('Outputfilnavn for solvognsmÕling (incl. path): ');
  repeat
    gotoxy(10,11);Writeln;
    gotoxy(10,11);Write('Filnavn: ');
    readln(out_navn);                      {Indlµser path og filnavn}
  until tjek_fil(out_navn);                {Unders°ger om path+filnavn er gyldigt}
  Assign(out_fil,out_navn);                {Sammenknytter variabel og filnavn}
  Rewrite(out_fil);                        {Opretter output-fil}
  Writeln(out_fil,'Soltracking');
  Writeln(out_fil,'Udf°rt af : ',Usr_data.navn);
  Writeln(out_fil,'Materiale : ',Usr_data.materiale);
  Writeln(out_fil,'Bemµrkning: ',Usr_data.bemaerkning);
  Writeln(out_fil);
  getdate(aar,mnd,dat,dag);                {Indlµser dato fra computerens ur}
  gettime(tim,min,sek,hun);                {Indlµser tidspunkt fra computerens ur}
  writeln(out_fil,'MÕling startet: ',dat,'-',mnd,'-',aar,' kl. ',tim,'.',min,'.',sek);
  writeln(out_fil,'MÕleparametre:');
```

```
  writeln(out_fil);
  write(out_fil,'MÕletype: ');              {Udskriver mÕletype i en fil}
  CASE measure_type OF
    1 : writeln(out_fil,'Solfanger');
    2 : writeln(out_fil,'Metset');
    3 : writeln(out_fil,'Dµklagstransmittans');
  end; {case measure_type}
  writeln(out_fil);
  write(out_fil,'Tracking type: ');
  CASE tracking_type OF                     {Udskriver tracking type i en fil}
    1 : writeln(out_fil,'Om vandret akse');
    2 : writeln(out_fil,'Om lodret akse');
    3 : writeln(out_fil,'Om vandret og lodret akse');
    4 : writeln(out_fil,'Ingen tracking - Fast orientering');
  end; {CASE tracking_type}
  writeln(out_fil);                         {Udskrver en tom linje}
  IF measure_type=3 THEN                    {Hvis der udf°res mÕlinger med solvognen}
  begin
    writeln(out_fil,'Solvognsparametre:');
    Writeln(out_fil,'Afstand fra rammemidte til venstre afgrµnsning = ',v_flade_bred:7:2);
    Writeln(out_fil,'Afstand fra rammemidte til  h°jre  afgrµnsning = ',h_flade_bred:7:2);
    Write(out_fil,'MÕlemetode for solvogn: ');
    IF solvogn_maal_type=1 THEN             {Der udf°res step-mÕlinger pÕ solvogn}
    begin
      writeln(out_fil,'Step-mÕling');
      Writeln(out_fil,'Afstand mellem mÕlepunkter       : ',solvogn_step:6:3,' mm.');
      Writeln(out_fil,'Pause til indsving af solarimetre: ',solarimeter_indsving:6:2,' ms');
      Writeln(out_fil,'Ønsket antal scanninger          : ',antal_step);
    end; {If solvogn_maal=1}
    IF solvogn_maal_type=2 THEN             {Der udf°res kontinuert-mÕlinger pÕ solvogn}
    begin
      writeln(out_fil,'Kontinuert mÕling');
      writeln(out_fil,'Afstand mellem mÕlepunkter       : ',solvogn_step:6:3,' mm.');
    end; {if mÕletype=2}
    writeln(out_fil,'Tidsrum mellem mÕlinger: ',tracktid:6,' sekunder');
  end; {IF measure_type=3...}
  writeln(out_fil,'');                      {Sender en tom linje til outputfilen}
  Close(out_fil);                           {Lukker outputfil}
end; {solvogn_outputfil}

procedure init_solvogn(                     {Forkorter aktuatorstangen til solvogn mest muligt}
            v_flade_bred        : Real;
            VAR solvogn_position: Real);

var
  dvm_value        : Real;                  {MÕlevµrdi fra digitalvoltmeter}
  old_hpcount,                              {Counter f°r motork°rsel}
  new_hpcount,                              {Counter under motork°rsel}
  d_count          : Integer;               {Ønsket forskel mellem new- og oldhpcount}

begin
  setdio(R6_M3,0);                          {Motor 3 skal forkorte aktuatorarm}
  setdio(R5_M3,1);                          {Starter motor 3}
  repeat
    dvm_value:=getadcchannel(stop_neg_3);   {Aflµser negativt endestop for motor 3}
    esc_maaling(R5_M3,0,FALSE);             {Aflµser om der er trykket pÕ 'ESC'-tasten}
  until abs(dvm_value)>5;                   {Gentages indtil venstre endestop (>5 V) nÕs}
  setdio(R5_M3,0);                          {Stopper motor 3}
  delay(motor_pause);                       {Venter til motor er stoppet}
  d_count:=round((v_arm_bred-v_flade_bred)*M3_puls_length);
  old_hpcount:=round(gethpcount);           {Aflµser counter inden motorstart}
  dvm_value:=0;                             {Nulstiller antalet af l°kkegennemgange}
```

```
dvm_taeller:=0;                                  {Nulstiller vµrdi fra DVM - pga. repeat-l°kke}
setdio(R6_M3,1);                                 {Angiver at motor 3 skal forlµnge aktuatorarm}
setdio(R5_M3,1);                                 {Starter motor 3}
repeat
  dvm_taeller:=dvm_taeller+1;                    {Antallet af l°kkeange °ges med 1}
  new_hpcount:=round(gethpcount);                {Aflµser counter under k°rsel}
  IF dvm_taeller>=max_dvm_taeller THEN           {Der mÕles hvert max_dvm_taeller l°kkel°b}
  begin
    dvm_taeller:=0;                              {Nulstiller antallet af l°kkegange}
    dvm_value:=getadcchannel(stop_pos_3);        {Aflµser positivt endestop for motor 3}
    IF abs(dvm_value)>5 THEN                     {SÕ er der ramt et endestop}
    begin
      Closegraph;                                {Lukker evt. grafisk skµrm}
      Clrscr;
      writeln('Fejl i positionering af solvogn');
      writeln('Unders°g venligst placering af endestop pÕ k°reskinnen');
      writeln;
      writeln('MÕling stopper');
      writeln;
      stop_maaling;                              {Stopper mÕling}
    end; {IF abs(dvm_value)}
  end; {If dvm_taeller}
  esc_maaling(R5_M3,0,FALSE);                    {Aflµser om der er trykket pÕ 'ESC'-tasten}
until (new_hpcount>=old_hpcount+d_count);        {K°rer til venstre begrµsning af mÕleflade}
setdio(R5_M3,0);                                 {Stopper motor 3}
setdio(R6_M3,0);                                 {Angiver retning TILBAGE for motoren}
delay(motor_pause);                              {Venter til motor 3 er stoppet}
new_hpcount:=round(gethpcount);                  {Beregner position for solvogn incl. "overl°b"}
solvogn_position:=-v_arm_bred+(new_hpcount-old_hpcount)/M3_puls_length;
end; {procedure init_solvogn}

procedure solvogn_travers_step(                  {Steptraversering af solvogn + mÕling pÕ dµklag}
            VAR solvogn_position: Real;          {Solvognens position pÕ mÕlefladen}
            delta_azimut,                        {Vinkel-forskydning i azimut-retning}
            delta_haeld,                         {Vinkel-forskydning i hµldning-retning}
            locked_azimut,                       {Evt. lÕst azimut-vinkel}
            locked_haeld        : Real;          {Evt. lÕst hµldningsvinkel}
            tracking_type       : Shortint;      {Hvilken trackingtype der foretages mÕlinger med}
            VAR flade_azimut,                    {MÕlefladens azimut}
                flade_haeld     : Real;          {MÕlefladens hµldning}
            VAR scan_retning    : Boolean);      {Scanningsrtn. TRUE=venstre mod h°jre. FALSE=omvendt}

type
  SolPointerType = ^ SolPointer;                 {Opretter pointertype til uddata fra solarimetre}
  SolPointer     = RECORD
                     naeste    : SolPointerType;
                     placering,                  {Solvognens placering}
                     sol_1,                      {Input fra solarimeter 1}
                     sol_2,                      {Input fra solarimeter 2}
                     sol_3     : real;           {Input fra solarimeter 3}
                   end;

var
  Top, Bund, SP    : SolPointerType;             {Indeholder uddata fra solarimetre}
  Pointerstatus    : pointer;                    {Bruges til at frigive pointer-lagerplads igen}
  dvm_value,                                     {Vµrdi aflµst pÕ DVM - bruges til at aflµse endestop}
  dvm_value_1,                                   {Uddata fra DVM mÕlt over solarimeter under dµklag}
  dvm_value_2,                                   {Uddata fra DVM mÕle over solarimeter m. skyggering}
  dvm_value_3,                                   {Uddata fra DVM mÕlt over solarimeter udenfor dµklag}
  nuv_solvogn_position,                          {Nuvµrende position af solvogn}
  alfa_s_deg,                                    {Solh°jde i grader}
  azimut_deg       : real;                       {Solazimut i grader}
```

```
  old_hpcount,                              {Status af HP-counterkort f"r start af motor 3}
  new_hpcount,                              {Nuvµrende status af HP-counterkort}
  d_count,                                  {Antal counts pr. mÕling}
  solvstep,                                 {Antal gange solvognen har kørt i dette tidsstep}
  pause             : Integer;              {Pause mellem mÕlinger}
  yyl,mml,ddl,dowl,                         {Dato efter positioneringstjek}
  timl,minl,sekl,hunl : Word;               {Tid  efter positioneringstjek}
  oldtid            : time_type;            {Tidspunkt ved forrige solvognsstep}
  used_delay        : Longint;              {Hundrededele sekund brugt pÕ positionsberegning}
  outtxt            : string[17];           {Tekststreng der beskriver den nuvµrende handling}

begin
  mark(Pointerstatus);                      {Gemmer status af heapen for pointervariable}
  Top:= NIL; Bund:=NIL;                     {Nulstiller top/bund-element i pointer-liste}
  oldtid.time:=0;oldtid.min:=0;             {Nulstiller tidspunkt for sidste solvognsstep}
  oldtid.sek:=0;oldtid.hun:=0;
  outtxt:='Scanner mÕleflade';              {Default beskrivende handlingstekst}
  setcolor(7);
  bar(256,120,392,128);                    {Sletter tidligere skµrmindhold pÕ udskrivningssted}
  outtextXY(256,120,outtxt);                {Udskriver handling pÕ skµrmen}
  IF motor_pause>solarimeter_indsving THEN  {Find lµngste pause}
    pause:=motor_pause
  ELSE
    pause:=round(solarimeter_indsving);
  nuv_solvogn_position:=solvogn_position;   {Sµtter den nuvµrende solvognsposition}
  d_count:=round(solvogn_step*M3_puls_length);
  delay(round(solarimeter_indsving));       {Pause til indsvingning af solarimetre}
  solvstep:=0;                              {Nulstiller antallet af solvognsstep}
  old_hpcount:=round(gethpcount);           {Aflµser counteren inden solvognen skal k°re}
  repeat
    dvm_taeller:=0;                         {Nulstiller antallet af l°kkegennemgange}
    dvm_value:=0;                           {Nulstiller vµrdi fra DVM - pga. repeat-l°kke}
    solvstep:=solvstep+1;                   {Tµller solvstep en op}
    if scan_retning then
      setdio(R6_M3,1)                       {Angiver at motor 3 skal forlµnge aktuatorarm}
    else
      setdio(R6_M3,0);                      {Angiver at motor 3 skal forkorte aktuatorarm}
    setdio(R5_M3,1);                        {Starter motor 3}
    repeat
      dvm_taeller:=dvm_taeller+1;           {Antallet af l°kkegange °ges med 1}
      new_hpcount:=round(gethpcount);       {Aflµser counter under k°rsel med solvogn}
      IF dvm_taeller>=max_dvm_taeller THEN  {Der mÕles hvert max_dvm_taeller l°kkel°b}
      begin
        dvm_taeller:=0;                     {Nulstiller antallet af l°kkegennemgange}
        IF scan_retning THEN                {Hvis der scannes i positiov retning, sÕ skal der}
          dvm_value:=getadcchannel(stop_pos_3) {mÕles pÕ positivt endestop}
        else                                {Hvis der scannes i negativ retning, sÕ skal der}
          dvm_value:=getadcchannel(stop_neg_3);{mÕles pÕ negativt endestop}
        IF abs(dvm_value)>5 THEN            {SÕ er der ramt et endestop}
        begin
          Closegraph;
          Clrscr;
          writeln('Der er ramt et endestop pÕ traverseringsskinnen som ikke var forventet');
          writeln('Kontroller venligst at endestoppene sidder hvor de skal');
          writeln;
          writeln('MÕling stopper');
          writeln;
          stop_maaling;                     {MÕling stopper}
        end; {if abs(dvm_value)}
      end; {if dvm_taeller}

      esc_maaling(R5_M3,0,TRUE);            {Aflµser om der er trykket pÕ 'ESC'-tasten}
```

```
until new_hpcount>=old_hpcount+(d_count*solvstep);
setdio(R5_M3,0);                        {Stopper motor 3}
getdate(aar,mnd,dat,dag);               {Aflæser dato      inden positioneringstjek}
gettime(tim,min,sek,hun);               {Aflæser tidspunkt inden positioneringstjek}
                                        {Det unders"øges om der er behov for positionering}
solpos(alfa_s_deg,azimut_deg,timevinkel,deklination_deg);
                                        {beregner solens position pÅ himlen}
beregn_flade_orient(TRUE,alfa_s_deg,azimut_deg,flade_azimut,flade_haeld,tracking_type,
                    ny_flade_azimut,ny_flade_haeld);
                                        {beregner ny orientering af mÅlefladen}
IF (abs(ny_flade_azimut-flade_azimut)>=Graddiff_azimut) OR
   (abs(ny_flade_haeld -flade_haeld )>=Graddiff_haeld ) THEN
                                        {SÅ skal der drejes - brugeren informeres}
begin
  outtxt:='Roterer mÅleflade';          {Udskifter handlingstekst}
  Bar(256,120,392,128);                 {Sletter gamle tekst}
  OuttextXY(256,120,outtxt);            {Udskriver nye tekst pÅ skærmen}
  roter_flade(ny_flade_azimut,ny_flade_haeld,flade_azimut,flade_haeld);
  outtxt:='Scanner mÅleflade';          {Reetablerer gamle tekst}
  Bar(256,120,392,128);                 {Sletter nuværende tekst}
  OuttextXY(256,120,outtxt);            {Udskriver gamle tekst pÅ skærmen}
end;                                    {Positioneringstjek er færdig}
getdate(yy1,mm1,dd1,dow1);              {Aflæser dato      efter positioneringstjek}
gettime(tim1,min1,sek1,hun1);           {Aflæser tidspunkt efter positioneringstjek}
used_delay:=hund_diff(yy1,mm1,dd1,tim1,min1,sek1,hun1,
                      aar,mnd,dat,tim ,min ,sek ,hun);
                                        {Beregner brugt tid pÅ positionering}
                                        {Pause minus rotations/beregningstid}
delay(pause-used_delay);
dvm_value_1:=getadcchannel(Best_sol_1); {MÅling pÅ solarimeter nr. 1}
dvm_value_2:=getadcchannel(Best_sol_2); {MÅling pÅ solarimeter nr. 2}
dvm_value_3:=getadcchannel(Best_sol_3); {MÅling pÅ solarimeter nr. 3}
dvm_value_1:=dvm_value_1*sol_1;         {Omregning fra volt til watt for solarimeter 1}
dvm_value_2:=dvm_value_2*sol_2;         {Omregning fra volt til watt for solarimeter 2}
dvm_value_3:=dvm_value_3*sol_3;         {Omregning fra volt til watt for solarimeter 3}
new(SP);                                {Opretter nyt element i Pointer-struktur}
SP^.placering := nuv_solvogn_position;  {Indlæser solvognsposition i Pointer-struktur}
SP^.sol_1     := dvm_value_1;           {Indlæser solarimeter 1 i Pointer-struktur}
SP^.sol_2     := dvm_value_2;           {Indlæser solarimeter 2 i Pointer-struktur}
SP^.sol_3     := dvm_value_3;           {Indlæser solarimeter 3 i Pointer-struktur}
IF scan_retning THEN                    {Danner FIFO-struktur}
begin
  SP^.naeste:=NIL;                      {Lader næste element pege pÅ tom lagerplads}
  IF Top=NIL THEN                       {Første gang lØkken kører}
    Top:=SP
  else
    Bund^.naeste:=SP;                   {Nederste element sættes til nuværende mÅling}
  Bund:=SP;
end
ELSE                                    {Danner LIFO-struktur}
begin
  SP^.naeste:= Top;
  Top:=SP;                              {Øverste element sættes til nuværende mÅling}
end;
screen3_data(nuv_solvogn_position,dvm_value_1,dvm_value_2,dvm_value_3,oldtid);
                                        {Indsætter variable værdier i uddata-skærm 3 - tid
                                         og bestrÅlingsstyrker}
gettime(oldtid.time,oldtid.min,oldtid.sek,oldtid.hun);
                                        {Aflæser ur efter solvognsstep}
new_hpcount:=round(gethpcount);         {Aflæser counter efter pause}
nuv_solvogn_position:=solvogn_position+(new_hpcount-old_hpcount)/M3_puls_length;
until ((nuv_solvogn_position>=h_flade_bred) OR (abs(nuv_solvogn_position)>=v_flade_bred));
                                        {Stop: Aktuatorarmen er udstrakt eller forkortet}
setdio(R6_M3,0);                        {Sætter retning TILBAGE for solvogn}
```

```
                                        {Indholdet af pointer-struktur lagres}
  append(out_fil);                      {Åbner outputfilen}
  WHILE SP<>NIL DO                      {SÅ er der stadig data i pointer-strukturen}
  begin                                 {Udskriver data fra tidsstep}
    Write(out_fil,SP^.placering:6:2);Write(out_fil,' ');
    Write(out_fil,SP^.sol_1:6:2);Write(out_fil,' ');
    Write(out_fil,SP^.sol_2:6:2);Write(out_fil,' ');
    Write(out_fil,SP^.sol_3:6:2);Writeln(out_fil,'');
  end;
  writeln(out_fil,'');                  {Skifter til ny linie inden næste MÅleserie}
  close(out_fil);                       {Lukker output-filen}
  release(Pointerstatus);               {Sletter oprettede pointer-strukturer siden}
                                        {mark-kommando}
                                       {Der skal traverseres mod startpunkt for næste step}
  old_hpcount:=round(gethpcount);       {Aflæser counter inden solvognskØrsel}
  new_hpcount:=round(gethpcount);       {Startværdi for counter under kØrsel}
  dvm_taeller:=0;                       {Nulstiller antallet af lØkkegennemgange}
  dvm_value:=0;                         {Nulstiller værdi fra DVM - pga. repeatlØkke}
  IF scan_retning THEN                  {Der skal kØres mod hØjre startpunkt}
  begin
    d_count:=round((h_flade_bred-nuv_solvogn_position)*M3_puls_length);
    setdio(R6_M3,1);                    {Forlænger aktuatorarm}
  end
  else                                  {Der kØres mod venstre startpunkt}
  begin
    d_count:=round((v_flade_bred+nuv_solvogn_position)*M3_puls_length);
    setdio(R6_M3,0);                    {Forkorter aktuatorarm}
  end;
  setdio(R5_M3,1);                      {Starter motor 3}
  repeat                                {KØr indtil stopbetingelse}
    dvm_taeller:=dvm_taeller+1;         {Antallet af lØkkegange Øges med 1}
    new_hpcount:=round(gethpcount);     {Aflæser hpcounter}
    IF dvm_taeller>=max_dvm_taeller THEN {Der MÅles hvert max_dvm_taeller lØkkel°b}
    begin
      dvm_taeller:=0;                   {Nulstiller antallert af lØkkegange}
      IF scan_retning THEN              {Bestemmer endestop udfra scannings-retning}
        dvm_value:=getadcchannel(stop_pos_3){Aflæser endestop i positiv retning for motor 3}
      ELSE
        dvm_value:=getadcchannel(stop_neg_3);{Aflæser endestop i negativ retning for motor 3}
      IF abs(dvm_value)>5 THEN          {SÅ er der ramt et endestop}
      begin
        Closegraph;                     {Lukker evt. grafisk skærm}
        Clrscr;
        writeln('Der er ramt et endestop pÅ traverseringesskinne som ikke var');
        writeln('forventet. Kontroller venligst at endestoppene er korrekt placeret');
        writeln;
        writeln('MÅling stopper ...');
        writeln;
        stop_maaling;                   {Afbryder relæer og stopper programudfØrelse}
      end; {if bs(dvm_value)}
    end; {IF dvm_taeller}
    esc_maaling(R5_M3,0,FALSE);         {Tjekker for tryk pÅ ESC under traversering}
  until (new_hpcount>=old_hpcount+d_count); {Stopbetingelse: Nok counts}
  setdio(R5_M3,0);                      {Stopper motor 3}
  IF scan_retning THEN                 {Sætter solvognens nye position afh. af scan-retning}
    solvogn_position:=h_flade_bred      {Sætter solvognens nye position til h. begrænsning}
  else
    solvogn_position:=-v_flade_bred;   {Sætter solvognens nye position til -v. begrænsning}
  scan_retning:=NOT(scan_retning);      {Vender scanningsretning til næste scanning}
end; {solvogn_travers_step}

procedure solvogn_travers_kont(         {Kont. traversering af solvogn + mÅling pÅ dæklag}
```

```
          VAR solvogn_position: Real;      {Solvognens postion pÕ mÖlefladen}
          VAR scan_retning    : Boolean); {Scanningsrtn. TRUE=venstre mod h"jre. FALSE=omvendt}

type
  SolPointerType = ^ SolPointer;              {Opretter pointertype til uddata fra solarimetre}
  SolPointer     = RECORD
                     naeste    : SolPointerType;
                     placering : Real;        {Solvognsposition}
                     sol_1     : Real;        {Input fra solarimeter 1}
                     sol_2     : Real;        {Input fra solarimeter 2}
                     sol_3     : Real;        {Input fra solarimeter 3}
                   end;

var
  Top, Bund, SP     : SolPointerType;         {Indeholder uddata fra solarimetre}
  Pointerstatus     : pointer;                {Bruges til at frigive pointer-lagerplads igen}
  dvm_value,                                  {Vµrdi aflµst pÕ DVM - bruges til at aflµse endestop}
  dvm_value_1,                                {Uddata fra DVM mÖlt over solarimeter under dµklag}
  dvm_value_2,                                {Uddata fra DVM mÖle over solarimeter m. skyggering}
  dvm_value_3,                                {Uddata fra DVM mÖlt over solarimeter udenfor dµklag}
  nuv_solvogn_position : Real;                {Nuværende position af solvogn}
  old_hpcount,                                {Status af HP-counterkort f°r start af motor 3}
  new_hpcount,                                {Nuvµrende status af HP-counterkort}
  d_count,                                    {Antal counts pr. mÖling}
  solvstep,                                   {Antal gange solvognen har k°rt i dette tidsstep}
  pause             : Integer;                {Pause mellem mÖlinger}
  oldtid            : Time_type;              {Tidspunkt for sidste solvognsstep}

begin
  mark(Pointerstatus);                        {Gemmer status af heapen for pointervariable}
  Top:= NIL; Bund:=NIL;                       {Nulstiller top/bund-element i pointer-liste}
  solvstep:=0;                                {Solvognen har k°rt 0 gange i dette tidsstep}
  oldtid.time:=0;oldtid.min:=0;               {Nulstiller tidspunkt for sidste solvognsstep}
  oldtid.sek:=0;oldtid.hun:=0;
  delay(2*round(solarimeter_indsving));       {Venter 2 solarimeterindsvingspauser}
  nuv_solvogn_position:=solvogn_position;     {Sµtter den nuvµrende solvognsposition}
  d_count:=round(solvogn_step*M3_puls_length);
  old_hpcount:=round(gethpcount);             {Aflµser counteren inden solvognen skal k°re}
  new_hpcount:=round(gethpcount);             {Startvµrdi for HP-counter}
  dvm_taeller:=0;                             {Nulstiller antallet af l°kkegennemgange}
  dvm_value:=0;                               {Nulstiller vµrdi fra DVM - pga. repeat-l°kke}
  if scan_retning then
    setdio(R6_M3,1)                           {Angiver at motor 3 skal forlµnge aktuatorarm}
  else
    setdio(R6_M3,0);                          {Angiver at motor 3 skal forkorte aktuatorarm}
  setdio(R5_M3,1);                            {Starter motor 3}
  repeat                                      {K°rer indtil solvogn er ved stop-punkt}
    repeat                                    {L°kke k°rer indtil der skal mÖles pÕ DVM}
      dvm_taeller:=dvm_taeller+1;             {Antallet af l°kkegennemgange °ges med 1}
      new_hpcount:=round(gethpcount);         {Lµser countervµrdi under k°rsel}
      IF dvm_taeller>=max_dvm_taeller THEN    {Der mÖles hvert max_dvm_taeller l°kkel°b}
      begin
        dvm_taeller:=0;                       {L°kketµller nulstilles}
        IF scan_retning THEN
          dvm_value:=getadcchannel(stop_pos_3){Aflµser endestop i positiv retning for motor 3}
        ELSE
          dvm_value:=getadcchannel(stop_neg_3);{Aflµser endestop i negativ retning for motor 3}
        IF abs(dvm_value)>5 THEN              {SÕ er der ramt et endestop}
        begin
          Closegraph;                         {Lukker evt. grafisk skµrm}
          Clrscr;
          writeln('Der er ramt et endestop pÕ traverseringsskinne der ikke var');
```

```
        writeln('forventet. Kontroller venligst at endestoppene er placeret rigtigt');
        writeln;
        writeln('MÕling stopper');
        writeln;
        stop_maaling;                       {Stopper programudf"relse}
      end; {if abs(dvm_value)}
    end; {if svm_taeller}
    esc_maaling(R5_M3,0,TRUE);              {Tjekker om de er trykket pÕ 'ESC'-tasten}
  until (new_hpcount>=old_hpcount+(d_count*solvstep));
  solvstep:=solvstep+1;                     {Solvognen har k°rt et step mere}
  if scan_retning then                      {Beregner ny posotion for solvognen}
    nuv_solvogn_position:=solvogn_position+(new_hpcount-old_hpcount)/M3_puls_length
  else
    nuv_solvogn_position:=solvogn_position-(new_hpcount-old_hpcount)/M3_puls_length;
  dvm_value_1:=getadcchannel(Best_sol_1); {MÕling pÕ solarimeter nr. 1}
  dvm_value_2:=getadcchannel(Best_sol_2); {MÕling pÕ solarimeter nr. 2}
  dvm_value_3:=getadcchannel(Best_sol_3); {MÕling pÕ solarimeter nr. 3}
  dvm_value_1:=dvm_value_1*sol_1;         {Omregning fra volt til watt for solarimeter 1}
  dvm_value_2:=dvm_value_2*sol_2;         {Omregning fra volt til watt for solarimeter 2}
  dvm_value_3:=dvm_value_3*sol_3;         {Omregning fra volt til watt for solarimeter 3}
  new(SP);                                {Opretter nyt element i Pointer-struktur}
  SP^.placering := nuv_solvogn_position;  {Indlµser solvognsposition i Pointer-struktur}
  SP^.sol_1     := dvm_value_1;           {Indlµser solarimeter 1 i Pointer-struktur}
  SP^.sol_2     := dvm_value_2;           {Indlµser solarimeter 2 i Pointer-struktur}
  SP^.sol_3     := dvm_value_3;           {Indlµser solarimeter 3 i Pointer-struktur}
  IF scan_retning THEN                    {Danner FIFO-struktur}
  begin
    SP^.naeste:=NIL;                      {Lader nµste element pege pÕ tom lagerplads}
    IF Top=NIL THEN                       {F°rste gang l°kken k°rer}
      Top:=SP
    else
      Bund^.naeste:=SP;
    Bund:=SP;                             {Indsµtter data pÕ nederste plads}
  end
  ELSE                                    {Danner LIFO-struktur}
  begin
    SP^.naeste:= Top;
    Top:=SP;                              {Indsµtter data pÕ °verste plads}
  end;
  screen3_data(nuv_solvogn_position,dvm_value_1,dvm_value_2,dvm_value_3,oldtid);
                                          {Indsµtter variable vµrdier i uddata-skµrm 3 - tid
                                           og bestrÕlingsstyrker}
  gettime(oldtid.time,oldtid.min,oldtid.sek,oldtid.hun);
                                          {Aflµser ur efter solvognsstep}
  new_hpcount:=round(gethpcount);         {Aflµser counter efter pause}
  nuv_solvogn_position:=solvogn_position+(new_hpcount-old_hpcount)/M3_puls_length;
until ((nuv_solvogn_position>=h_flade_bred) OR (abs(nuv_solvogn_position)>=v_flade_bred));
                                        {Stopkriterier: Aktuator er udstrakt eller forkortet}
setdio(R6_M3,0);                          {Sµtter retning TILBAGE for solvogn}
                                          {Indholdet af pointer-struktur lagres pÕ disken}
append(out_fil);                          {Ôbner outputfilen}
WHILE SP<>NIL DO                          {SÕ er der stadig data i pointer-strukturen}
begin                                     {Udskriver mÕledata}
  Write(out_fil,SP^.placering:6:2);Write(out_fil,' ');
  Write(out_fil,SP^.sol_1:6:2);Write(out_fil,' ');
  Write(out_fil,SP^.sol_2:6:2);Write(out_fil,' ');
  Write(out_fil,SP^.sol_3:6:2);Writeln(out_fil,'');
end;
writeln(out_fil,'');                      {Skifter til ny linie inden nµste mÕleserie}
close(out_fil);                           {Lukker output-filen}
release(Pointerstatus);                   {Sletter oprettede pointer-strukturer siden}
                                          {mark-kommando}
```

```
                                    {Der skal traverseres mod startpunkt for nøste step}
old_hpcount:=round(gethpcount);     {Aflæser counter inden traversering}
new_hpcount:=round(gethpcount);     {Sætter startværdi for counter under kørsel}
dvm_taeller:=0;                     {Nulstiller antallet af løkkegennemgange}
dvm_value:=0;                       {Nulstiller værdi fra DVM - pga. repeatløkke}
IF scan_retning then
begin
  d_count:=round((h_flade_bred-nuv_solvogn_position)*M3_puls_length);
  setdio(R6_M3,1);                  {Forlænger aktuatorarm}
end
else
begin
  d_count:=round((v_flade_bred+nuv_solvogn_position)*M3_puls_length);
  setdio(R6_M3,0);                  {Forkorter aktuatorarm}
end;
setdio(R5_M3,1);                    {Starter motor 3}
repeat                              {Kør indtil stopbetingelse}
  dvm_taeller:=dvm_taeller+1;       {Antallet af løkkegange øges med 1}
  new_hpcount:=round(gethpcount);   {Aflæser hpcounter}
  IF dvm_taeller>=max_dvm_taeller THEN  {Der måles hvert max_dvm_taeller løkkeløb}
  begin
    dvm_taeller:=0;                 {Nulstiller antallet af løkkegange}
    IF scan_retning THEN            {Bestemmer endestop udfra scanningsretning}
      dvm_value:=getadcchannel(stop_pos_3){Aflæser endestop i positiv retning for motor 3}
    ELSE
      dvm_value:=getadcchannel(stop_neg_3);{Aflæser endestop i negativ retning for motor 3}
    IF abs(dvm_value)>5 THEN        {Så er der ramt et endestop}
    begin
      Closegraph;                   {Lukker evt. grafisk skærm}
      Clrscr;
      writeln('Der er ramt et endestop på traverseringsskinne som ikke var');
      writeln('forventet. Kontroller venligst at endestoppene er korrekt placeret');
      writeln;
      writeln('Måling stopper ...');
      writeln;
      stop_maaling;                 {Afbryder relæer og stopper programudførelse}
    end; {if abs(dvm_value)}
  end; {if dvm_taeller}
  esc_maaling(R5_M3,0,FALSE);       {Tjekker for tryk på ESC under traversering}
until (new_hpcount>=old_hpcount+d_count);
setdio(R5_M3,0);                    {Stopper motor 3}
IF scan_retning THEN
  solvogn_position:=h_flade_bred    {Sætter solvognens position til h. begrænsning}
ELSE
  solvogn_position:=-v_flade_bred;  {Sætter solvognens position til -v. begrænsning}
scan_retning:=NOT(scan_retning);    {Vender scanningsretning til næste scanning}
end; {solvogn_travers_kont}

procedure initier(                  {Initierer HP-udstyr og drejer fladen til start}
          VAR flade_azimut,         {Målefladena azimut}
              flade_haeld     : Real;   {Målefladens hældning}
          VAR Measure_type,         {Måletype: Solfanger, Metset eller dæklag}
              Tracking_type   : Shortint;{Hvilken trackingtype der benyttes ved målinger}
          VAR Batchparam      : VinkelPointerType; {delta/fast vinkler, antal scanninger}
          VAR v_flade_bred,         {Venstre begrænsning af måleflade}
              h_flade_bred,         {Højre  begrænsning af måleflade}
              solvogn_step,         {Solvognens step mellem målinger}
              solarimeter_indsving, {Tid til indsvingning af solarimetre}
              solvogn_position : Real;  {Solvognens position}
          VAR solvogn_maal_type: Shortint;{Kontinuert eller step-måling}
          VAR Usr_data        : Usr_dat_t;{Navn på forsøgsansvarlig mv.}
          VAR tracktid        : Word);  {Tid mellem måleserier}
```

```
begin
  if NOT initadatool THEN                    {SÖ er der fejl i initialiseringen}
  begin
    writeln('Fejl i initialisering af mÖle/tµlle-kort');
    writeln;
    writeln('Unders"q venligst om alle kort er sat rigtigt i og om alle enheder er tµndt');
    writeln('Tryk en <RETUR>');
    readln;                                  {Venter pÖ brugerinput - skal vµre <return>}
    halt;                                    {Stop program}
  end;
  bruger_inddata(measure_type,tracking_type,Batchparam,v_flade_bred, h_flade_bred,
                 solvogn_step,solarimeter_indsving,solvogn_maal_type,Usr_data,tracktid);
                                             {Indlµser data fra brugeren}
  Clrscr;
  slet_linje(25);                            {Sletter nederste linje pÖ skµrmen}
  gotoxy(1,25);write('Nulstiller mÖleflade. Dette kan tage noget tid ...');
  nulstil_flade(flade_azimut,flade_haeld);
  IF Measure_type=3 THEN                     {Hvis der skal foretages dµklagsmÖlinger}
  BEGIN
    solvogn_outputfil(measure_type,tracking_type,v_flade_bred, h_flade_bred,
                      solvogn_step,solarimeter_indsving,solvogn_maal_type,Usr_data,tracktid);
                                             {Uddatafil oprettes}
    init_solvogn(v_flade_bred,solvogn_position);
                                             {Solvognen placeres i startposition}
  END;
end; {initier}

procedure set_start_pos(                     {Sµtter fladens position inden f°rste tidsstep}
            VAR flade_azimut,
                flade_haeld     : Real;
            delta_azimut,
            delta_haeld,
            locked_azimut,
            locked_haeld        : Real);
var
  alfa_s_deg,
  azimut_deg              : real;

begin
  getdate(aar,mnd,dat,dag);                  {Aflµser dato i computer}
  gettime(tim,min,sek,hun);                  {Aflµser tidspunkt i computer}
  slet_linje(25);
  gotoxy(1,25);
  write('Beregner solh°jde');
  solpos(alfa_s_deg,azimut_deg,timevinkel,deklination_deg);
                                             {Beregner solh°jde og azimut}
  slet_linje(25);
  gotoxy(1,25);
  write('Beregner mÖlefladens position');
  beregn_flade_orient(FALSE,alfa_s_deg, azimut_deg,flade_azimut,flade_haeld,tracking_type,
    ny_flade_azimut,ny_flade_haeld);         {Beregner fladens orientering}
  slet_linje(25);
  gotoxy(1,25);
  write('Roterer mÖleflade til start. Azi ',flade_azimut:6:2,' til ',ny_flade_azimut:6:2,' Hld
',
        flade_haeld:6:2,' til ',ny_flade_haeld:6:2);
  roter_flade(ny_flade_azimut,ny_flade_haeld,flade_azimut,flade_haeld);
                                             {Orientering af fladen: flade_azimut og flade_haeld}
  slet_linje(25);
end; {set_start_pos}

procedure tidsstep(                          {Handlinger i l°bet af et tidsstep}
```

```
            Measure_type        : Shortint; {Mõletype: Solfanger, Metset eller dµklag}
            VAR flade_azimut,               {Fladens azimut}
                flade_haeld     : Real;     {Fladens hµldning}
            delta_azimut,                   {Vinkel-forskydning i azimut-retning}
            delta_haeld,                    {Vinkel-forskydning i hµldning-retning}
            locked_azimut,                  {Evt. lõst azimut-vinkel}
            locked_haeld        : Real;     {Evt. lõst hµldningsvinkel}
            solvogn_maal_type   : Shortint; {Kontinuert eller step-mõling}
            v_flade_bred,                   {Venstre begrµnsning af mõleflade}
            h_flade_bred        : Real;     {H°jre   begrµnsning af mõleflade}
            Usr_data            : Usr_dat_t;{Navn põ fors°gsansvarlig mv.}
            VAR scan_retning    : Boolean;  {Scanningsrtn. TRUE=venstre mod h°jre, FALSE=omvendt}
            VAR solvogn_position: Real);    {Solvognens position põ mõlefladen}

var
  alfa_s_deg,                               {Solh°jde i grader}
  azimut_deg        : real;                 {Solazimut i grader}

begin
  getdate(aar,mnd,dat,dag);                 {Aflµser dato i computer}
  gettime(tim,min,sek,hun);                 {Aflµser tidspunkt i computer}
  slet_linje(25);
  gotoxy(1,25);
  write('Beregner solh°jde. Step: ',batchparam^.scanning-antal_step,'/');
  IF batchparam^.scanning=0 THEN
    write(chr(236))                         {Skriver uendeligt-tegn, ved uendeig mõling}
  else
    write(batchparam^.scanning);            {Skriver antal tidssteps for dette vinkelsµt}
  solpos(alfa_s_deg,azimut_deg,timevinkel,deklination_deg);
                                            {Beregner solh°jde og azimut}

  slet_linje(25);
  gotoxy(1,25);
  write('Beregner mõlefladens position. Step: ',batchparam^.scanning-antal_step,'/');
  IF batchparam^.scanning=0 THEN
    write(chr(236))                         {Skriver uendeligt-tegn, ved uendeig mõling}
  else
    write(batchparam^.scanning);            {Skriver antal tidssteps for dette vinkelsµt}
  beregn_flade_orient(FALSE,alfa_s_deg, azimut_deg,flade_azimut,flade_haeld,tracking_type,
    ny_flade_azimut,ny_flade_haeld);        {Beregner fladens orientering}
  slet_linje(25);
  gotoxy(1,25);
  write('Roterer mõleflade. Azi ',flade_azimut:6:2,' til ',ny_flade_azimut:6:2,' Hld ',
    flade_haeld:6:2,' til ',ny_flade_haeld:6:2,' Step: ',batchparam^.scanning-antal_step,'/');
  IF batchparam^.scanning=0 THEN
    write(chr(236))                         {Skriver uendeligt-tegn, ved uendeig mõling}
  else
    write(batchparam^.scanning);            {Skriver antal tidssteps for dette vinkelsµt}
  roter_flade(ny_flade_azimut,ny_flade_haeld,flade_azimut,flade_haeld);
                                            {Orientering af fladen: flade_azimut og flade_haeld}
  i_vinkel(deklination_deg, flade_haeld, flade_azimut, timevinkel,indfaldsvinkel);
  IF measure_type=3 THEN                    {Sõ skal der foretages transmittansmõlinger}
  begin
    append(out_fil);                        {Skriver mõlingsnummer, delta/lõste vinkler}
    writeln(out_fil,'Mõling nummer: ',batchparam^.scanning-antal_step,
                    ' af ',batchparam^.scanning);
    CASE tracking_type OF
      1 : begin
            Writeln(out_fil,'Indfaldsvinkel i solh°jde: ',delta_haeld:5:1);
            Writeln(out_fil,'Lõst vinkel i azimut     : ',locked_azimut:5:1);
          end;
      2 : begin
            Writeln(out_fil,'Indfaldsvinkel i azimut  : ',delta_azimut:5:1);
```

```
          Writeln(out_fil,'LÕst vinkel i solh"jde   : ',locked_haeld:5:1);
        end;
    3 : begin
          Writeln(out_fil,'Indfaldsvinkel i azimut  : ',delta_azimut:5:1);
          Writeln(out_fil,'Indfaldsvinkel i hµldning: ',delta_haeld:5:1);
        end;
    4 : begin
          Writeln(out_fil,'LÕst vinkel i azimut     : ',locked_azimut:5:1);
          Writeln(out_fil,'LÕst vinkel i hµldning   : ',locked_haeld:5:1);
        end;
  end; {CASE}
  writeln(out_fil,'');                    {Indsµtter tom linie i uddatafilen}
  close(out_fil);                         {Lukker uddatafil}
  screen3(flade_haeld, flade_azimut,alfa_s_deg, azimut_deg,indfaldsvinkel,
          v_flade_bred,h_flade_bred,Usr_data);
                                          {Opbygger grafikbaseret uddata-skµrm}
  CASE solvogn_maal_type OF
    1 : {Der foretages stepmÕlinger}
        solvogn_travers_step(solvogn_position,delta_azimut,delta_haeld,
          locked_azimut,locked_haeld,tracking_type,flade_azimut,flade_haeld,scan_retning);
    2 : {Der foretages kontinuerte mÕlinger}
        solvogn_travers_kont(solvogn_position,scan_retning);
  end; {CASE solvogn_maal_type}
  Closegraph;                            {Lukker grafikskµrm ned - sµtter maskinen i txt-mode}
 end; {IF measuretype=3}
end; {tidsstep}

begin
end.
```

B.2 Kildekode til DATA.TPU

```
unit data;
{Definerer globale variable for procedurerne adatool og inddata}

interface

const
  Isc             = 7;                {Adresser pÕ indstikskort}
  Dvmadd          = 709;              {Adresse pÕ digitalvoltmeter}
  dacadd          = 709;              {Adresse pÕ digital til analog konverter}
  cntadd          = 709;              {Adresse pÕ tµllerkort}
  dioadd          = 710;              {Adresse pÕ relµboks}
  noerr           = 0;                {Opbevarer fejlkoder}
                                      {Kanaloversigt i HP-Boksen}
  R1_M1           = 1;                {K. 1 µndrer lµngde pÕ aktuator-arm vha. vandret
                                       motor. 0 = OP (mod vandret), 1 = NED (mod lodret)}
  R2_M1           = 2;                {K. 2 angiver k°r/stop af vandret motor
                                       0 = STOP, 1 = KÏRSEL}
  R3_M2           = 3;                {K. 3 angiver k°r/stop for lodret motor
                                       0 = STOP, 1 = KÏRSEL}
  R4_M2           = 4;                {K. 4 dreje omkring lodret akse
                                       0 = FREM (Vest), 1 = TILBAGE (Øst)}
  R5_M3           = 5;                {K. 5 angiver k°r/stop for solmÕlevogn-motor
                                       0 = STOP, 1 = K°rsel}
  R6_M3           = 6;                {K. 6 angiver retning for k°rsel for solmÕlevogn
                                       0 = FORLÄNG, 1 = FORKORT}
                                      {Gearing af motorer}
  M1_puls_length  = 58.72;            {Antal pulser pr. mm for aktuatorstang. Motor 1}
  M2_puls_length  = 118.05;           {Antal pulser pr. grad drejning.       Motor 2}
  M3_puls_length  = 15;               {Antal pulser pr. mm for aktuatorstang. Motor 3}

                                      {Startkriterie for drejning. Hvis forskel mellem
                                      gammel og ny vinkel>Graddiff, sÕ foretages drejning}
  Graddiff_haeld  = 0.25;             {Startkriterie for hµldningsdrejning}
  Graddiff_azimut = 0.25;             {Startkriterie for azimutdrejning}

                                      {Stedlige koordinater}
  long            = -12.5;            {MÕlefladens lµngdegrad}
  latt            = 55.5;             {MÕlefladens breddegrad}

                                      {Def. af kanaler i DVM som bruges under styring}
  Stop_neg_1      = 40;               {Endestop i 'NED' omdrejningsretning for motor 1}
  Stop_pos_1      = 41;               {Endestop i 'OP' omdrejningsretning for motor 1}
  Stop_neg_2      = 42;               {Endestop i 'TILBAGE' omdrejningsretning for motor
2}
  Stop_pos_2      = 43;               {Endestop i 'FREM' omdrejningsretning for motor 2}
  Stop_neg_3      = 44;               {Endestop i negativ omdrejningsretning for motor 3}
  Stop_pos_3      = 45;               {Endestop i positiv omdrejningsretning for motor 3}
  Best_sol_1      = 46;               {BestrÕlingsstyrke pÕ mÕle-solarimeter}
  Best_sol_2      = 47;               {BestrÕlingsstyrke pÕ mÕle-solarimeter m. skyggering}
  Best_sol_3      = 48;               {BestrÕlingsstyrke pÕ reference-solarimeter}

                                      {Solarimeterkonstanter}
  sol_1           = 2000000;          {Konstant for solarimeter 1}
  sol_2           = 2000000;          {Konstant for solarimeter 2}
  sol_3           = 2000000;          {Konstant for solarimeter 3}

  motor_pause     = 1000;             {Pause mellem motorskift. Bruges til at stoppe motor}
  motortid        = 30;               {Tidsrum afsat til at rotere/vippe konstruktionen}
  solvogn_hast    = 4;                {Solvognens hastighed er 4 mm/sekund}
```

```
                                        {Geometriske definitioner}
  v_arm_bred       = 240;               {Afstand fra center af ramme til venstre afgrænsning
                                         af aktuatorarmens bevægelsesområde (i milimeter).
                                         Aktuatorstang er strakt}
  h_arm_bred       = 200;               {Afstand fra center af ramme til h"jre afgrænsning
                                         af aktuatorarmens bevægelsesområde (i milimeter).
                                         Aktuatorstang er kort}

                                        {Nedenstående vinkeler er bestemt af
                                         endestoppositioner}
  max_haeldning    =  87.7;             {Maximal hældning for mØlefladen}
  min_haeldning    =   0;               {Minimal hældning for mØlefladen}
  max_azimut       = 170;               {Maximal azimutvinkel for mØlefladen}
  min_azimut       =-130;               {Minimal azimutvinkel for mØlefladen}

  max_dvm_taeller  = 500;               {Antal gennemlØbne lØkker mellem der tjekkes pÅ
                                         digitalvoltmeter for om der skulle være ramt nogle
                                         endestop}

  ugedag : array [0..6] of String[7]    {Ugedag i ugen}
         = ('SØndag','Mandag','Tirsdag',
            'Onsdag','Torsdag','Fredag',
            'LØrdag');
  maxsolstraal     = 1200;              {Maksimal solstrØling pÅ solarimeter}

type
  real_filtype     = TEXT;              {Bruges til uddata fra solvognsmØlinger}
  Usr_dat_t        =  RECORD            {Navn pÅ forsØgsansvarlig, materiale samt bemærkning}
                        Navn,
                        Materiale,
                        Bemaerkning : String[60];
                      END;
  Time_type        = RECORD             {Opbevarer tidspunkt}
                       time,
                       min,
                       sek,
                       hun      : Word;
                     end;
  Str80            = String[80];        {Tekststreng med længden 80 tegn}
  VinkelPointerType= ^VinkelPointer;    {Pointertype til opbevaring af antal scanninger med}
  VinkelPointer    = RECORD             {hvilke vinkler}
                       naeste   : VinkelPointerType;
                       delta_azimut,    {Ønsket indfaldsvinkel i Azimutretning}
                       delta_haeld,     {Ønsket indfaldsvinkel i Hældningsretning}
                       locked_azimut,   {LÅst vinkel i azimutretning}
                       locked_haeld : Real; {LÅst vinkel i hældningsretning}
                       scanning : LongInt;    {Antal scanninger med pÅgældende vinkel}
                     end;


var
  Batchparam,TopBP,
  BundBP           : VinkelPointerType;    {Opbevarer antal scanninger og vinkler for tidssteps}
  adc              : array[1..100] of real;
  Usr_data         : Usr_dat_t;
  out_fil          : real_filtype;        {Uddatafil fra solvognsmØlingerne}
  hpoldcount       : Longint;             {Countertal}
  Measure_type,                           {MØlingstype ved mØlinger}
  Tracking_type,                          {Trackingtype ved mØlinger}
  Solvogn_maal_type: Shortint;            {MØletype for solvogn; 1=StepmØling, 2 = Kontinuert}
  delta_haeld,                            {Reservation af lagerplads til 10 mØlinger}
  delta_azimut,                           {Tracking-vinkler i forhold til solens position}
```

```
  locked_azimut,                          {Fast vµrdi for opstillingens orientering}
  locked_haeld,                           {Fast vµrdi for opstillingens hµldning}
  hpcount,
  flade_azimut,                           {MÕlefladens azimut}
  flade_haeld,                            {MÕlefladens hµldning}
  ny_flade_azimut,                        {Planlagt position for mÕleflade efter drejning}
  ny_flade_haeld,                         {MÕlefladens venstre begrµsning}
  v_flade_bred,                           {MÕlefladens h°jre begrµnsning}
  h_flade_bred,                           {Afstand i mm mellem solvognsmÕlinger}
  solvogn_step,                           {Indsvingningstid for solarimetre i milisekunder}
  solarimeter_indsving,                  {Position af solvogn i forhold til midten af fladen}
  solvogn_position,                      {vinkel mellem stedets meridianplan og linie til sol}
  timevinkel,                             {Solens deklination i radianer (3.1.2)}
  deklination,                            {Solens deklination i grader (3.1.2)}
  deklination_deg,
  indfaldsvinkel    : Real;               {Indfaldsvinkel pÕ en vilkÕrlig flade}
  antal_step        : Word;               {Antallet af tidssteps i mÕlingen}
  mellemrum         : String[6];          {En tekststreng kun indeholdende mellemtrum}
  scan_retning      : Boolean;            {Angivelse af retning for scanning vha. solvogn}
  tracktid          : Word;               {Antal sekunder mellem tidssteps}
  dvm_taeller       : Integer;            {Antallet af l°kkegennemgange}
  aar, mnd,
  dat, dag,
  tim, min,
  sek, hun          : word;               {+r, mÕned, dato, dag, time, minut, sekund, hund-del}

IMPLEMENTATION

begin
end.
```

B.3 Kildekode til DAEKLAG.EXE

```
program daeklag;                          {Foretager mÕlinger pÕ dµklag}

uses
  adatool,data,crt,dos,uddata;

var
  MainPointerstatus: pointer;             {Bruges til at frigeive pointer-lagerplads igen}
  starttid,stoptid : Longint;             {Antal sekunder siden klokken 00:00:00}

begin
  mark(MainPointerstatus);                {Gemmer status af heapen for pointervariable}
  initier(flade_azimut,flade_haeld,       {Indlµser brugeredata, nulstiller mÕlekort}
          Measure_type,Tracking_type,
          Batchparam,
          v_flade_bred,h_flade_bred,
          solvogn_step,solarimeter_indsving,
          solvogn_position,solvogn_maal_type,
          Usr_data,tracktid);
  Batchparam:=TopBP;                      {Sµtter top i vinkelpointerstruktur}
  WHILE Batchparam<>NIL DO                {SÕ lµnge der er parametre i batchk°en}
  begin
    antal_step:=Batchparam^.scanning;     {Overf°rer  antal scanning fra pointer til variabel}
    delta_azimut :=batchparam^.delta_azimut;{Overf°rer  delta_azimut   fra pointer til variabel}
    delta_haeld  :=batchparam^.delta_haeld; {Overf°rer  delta_haeld    fra pointer til variabel}
    locked_azimut:=batchparam^.locked_azimut;{Overf°rer locked_azimut  fra pointer til variabel}
    locked_haeld :=batchparam^.locked_haeld;{Overf°rer  locked_haeld   fra pointer til variabel}
    IF (Measure_type=3) AND (Tracking_type=3) THEN
    begin                                 {Dµklagsmõling med tracking omkring 2 akser}
      gotoxy(1,1);
      Writeln('Skyggeklodsen foran diffusstrõlings-solarimeteren skal flyttes');
      Writeln('Ny hµldningsforskydning : ',delta_haeld:6:2);
      Writeln('Ny azimutforskydning    : ',delta_azimut:6:2);
      Writeln('Programudf°rsel pauset ...');
      Write  ('Tryk <RETUR> nÕr skyggeklodsen er sat i rigtig position ');
      readln;                             {Venter indtil brugeren har rykket pÕ skyggeklods}
    end;
    set_start_pos(flade_azimut,flade_haeld,delta_azimut,delta_haeld,locked_azimut,locked_haeld);
                                          {Sµtter startposition for flade}
    gettime(tim,min,sek,hun);             {Afluser PC'ens ur mht. tidspunkt}
    stoptid:=3600*tim+60*min+sek;
    starttid:=stoptid;                    {Startvµrdi for nytid - Starter WHILE-l°kke}
    REPEAT                                {Gentager °nsket antal tidssteps}
      stoptid:=stoptid+tracktid;          {Sµtter stoptidspunkt for WHILE-l°kken}
      antal_step:=antal_step-1;           {Der er pÕbegyndt et tidsstep mere}
      tidsstep(measure_type,flade_azimut,flade_haeld,
               delta_azimut,delta_haeld,locked_azimut,locked_haeld,solvogn_maal_type,
               v_flade_bred,h_flade_bred,Usr_data,scan_retning,solvogn_position);
      vent_tid_diff(starttid,stoptid,antal_step);{Udskriver tidsdifferens i (x,y)=(10,25)}
    UNTIL (antal_step=0) AND (Batchparam^.scanning<>0);
                                          {SÕ er der foretaget det °nskede antal tidssteps
                                           Hvis Batchparam^.scanning fra starten var nul sÕ
                                           fortsµtter l°kken uendeligt eller til brugerstop}
    Batchparam:=Batchparam^.naeste;       {Indlµser nµste element med antal scanning og vinkel}
  end; {WHILE Batchparam}
  release(MainPointerstatus);             {Sletter oprettede pointere siden mark-kommando}
end.
```

B.4 Kildekode til HPIBTOOL.TPU

```
unit hpibtool;

Interface

Uses
  p70hpib,cb,errvar,                          {HP biblioteker}
  data;                                       {Egne variabel-erklæringer}

function  initadatool:boolean;                {Initialisering}
function  getadcchannel(chan:integer)
                                : real;       {Mål spændingsforskel på kanal "chan"}
procedure setdio(       channel : byte;       {Relæ-boks - tænder/slukker for kanal "channel"}
                        io      : byte);
function  gethpcount:real;                    {Pulstæller-status/tal}

implementation

                                              {Definition af "globale" variable i 'HPIBTOOL'}
var
  err             : integer;
  command         : string;

function initadatool:boolean;                 {Initialiserer hp-interface. Udarbejdet af Lasse}
var
  i               : integer;
begin
  errorcode[1]:=0;
  err:=IoReset(Isc);                          {Set Interface to default}
  if err<>noerr then errorcode[1]:=1;         {konfiguration}
  err:=IoTimeOut(Isc,5);                      {Set timeout}
  if err<>noerr then errorcode[1]:=errorcode[1] and 1;
  err:=IoClear(Isc);                          {Clear IO-port}
  if err<>noerr then errorcode[1]:=errorcode[1] and 2;
  err:=IoClear(Dvmadd);                       {Set devices to known state}
  if err<>noerr then errorcode[1]:=errorcode[1] and 4;
  command:='CF4,1';                           {Pulstællerkort i slot 4 - Tællerkort tæller op}
  err:=IoOutputs(cntadd,command,length(command));
  if err<>noerr then errorcode[1]:=errorcode[1] and err and 4;
  command:='?%';                              {Initialisering af hp-digital portene - Relæboks}
  err:=IoOutputs(dioadd,command,length(command));
  if err<>noerr then errorcode[1]:=errorcode[1] and err and 4;
  command:='B1B2B3B4B5B6';
  err:=IoOutputs(dioadd,command,length(command));
  if err<>noerr then errorcode[1]:=errorcode[1] and err and 4;
  if errorcode[1]=0 then initadatool:=true
  else initadatool:=false;
end; {initadatool}

function getadcchannel(chan:integer):real;    {Aflæser digitalvoltmeter på 'chan'. Lavet af Lasse}
var
                  nr,                         {Tekst-værdi af kanalnummer}
                  command :string;            {Kommando som sendes til DVM}
begin
  if (chan>=0) and (chan<100) then begin
    str(chan,nr);
    command:='AC'+nr;                         {Luk kanal 'nr'}
    err:=IoOutputs(Dvmadd,command,length(command));
    if err<>noerr then errorcode[1]:=errorcode[1]+err+8;
    command:='VR5 VN1 VA1 VF1 VD5 VC0 VS0 VW0 VT3 SD0';
    err:=IoOutputs(Dvmadd,command,length(command));
```

```
    if err<>noerr then errorcode[1]:=errorcode[1] and 16;
    err:=IoEnter(Dvmadd,adc[chan]);          {Read value from channel 'chan'}
    if err<>noerr then errorcode[1]:=errorcode[1] and 32;
  end;
  getadcchannel:=adc[chan];
end; {getadcchannel}

procedure setdio(channel:byte; io:byte);     {Åbner eller lukker kanal 'chan'}
var
  command,                                   {Kommando som sendes til relæ}
  nr               : string;                 {Tekst-værdi af kanalnummer}
begin
  if (channel>0) and (channel<=6) then begin
    if (io=0) or (io=1) then begin
      str(channel,nr);
      if io=1 then command:='A '+nr          {Strøm gennem relæ nummer 'nr'}
      else command:='B '+nr;                 {Ingen strøm gennem relæ nummer 'nr'}
      err:=IoOutputs(Dioadd,command,length(command));
      if err<>noerr then errorcode[1]:=errorcode[1] and 128;
    end
    else errorcode[1]:=errorcode[1] and 256;
  end
end; {setdio}

function gethpcount:real;                    {Aflæser status på HP-counter}

procedure nulstilcounter;
begin
  command:='CS4,0';                          {Indlæser værdi '0' i counter i slot 4}
  err:=IoOutputs(Cntadd,command,length(command));
  command:='CF4,1';                          {Sætter counter i slot 4 til at tælle op}
  err:=IoOutputs(Cntadd,command,length(command));
end; {nulstilcounter}

begin
  command:='CR4,1';                          {Aflæs status på counter}
  err:=IoOutputs(Cntadd,command,length(command));
  if err<>noerr then errorcode[1]:=errorcode[1]+err and 16284;
  err:=IoEnter(Cntadd,hpcount);              {Spørger om counterens værdi}
  if err<>noerr then
    errorcode[1]:=errorcode[1] and 16284
  else
    if hpcount>99500 then
      nulstilcounter;   {Hvis counteren er ved at løbe over maxgrænse}
  gethpcount:=hpcount;
end; {gethpcount}

begin
end.
```

B.5 Kildekode til INDDATA.TPU

```
unit inddata;

Interface

Uses
  data,crt;                                  {Standard og HP biblioteker}

procedure tjek_esc;                          {Undersøger om der er trykket på 'ESC'-tasten = STOP}

procedure bruger_inddata(                    {Indtastning af inddata til brug for målingen}
            VAR Measure_type,                {Måletype: Solfanger, Metset eller døgklag}
                Tracking_type   : Shortint; {Hvilken trackingtype der foretages målinger med}
            VAR Batchparam      : VinkelPointerType; {Delta/låst vinkler og antal af scanninger}
            VAR v_flade_bred,                {Målefladens venstre begrænsningsflade}
                h_flade_bred,                {Målefladens  højre  begrænsningsflade}
                solvogn_step,                {Solvognens step mellem målinger}
                solarimeter_indsving: Real; {Tid for solarimetrene til indsvingning}
            VAR solvogn_maal_type   : Shortint;{Kontinuert eller step-måling}
            VAR Usr_data        : Usr_dat_t;{Navn på forsøgsansvarlig mv.}
            VAR tracktid        : Word);     {Tid mellem måleserier}

implementation

procedure tjek_esc;                          {Undersøger om der er trykket på 'ESC'-tasten = STOP}

var
  ch              : Char;

begin
  IF Keypressed THEN                         {Så findes der et tastetryk i keyboardbufferen}
  begin
    ch:=Readkey;                             {Afløser tastetryk fra keyboardbufferen}
    IF ch=#27 THEN                           {Så er der trykket ESC}
    begin
      gotoxy(1,1);
      Write('Ønskes målingen stoppet ? (J/N) : ');
      repeat
        ch:=Readkey;                         {Afløser brugerens tastetryk}
        ch:=Upcase(ch);
      until ((ch='J') OR (ch='N'));          {Indtil der er svaret enten Ja eller Nej}
      IF ch='J' THEN                         {Så skal målingen stoppes}
      begin
        Writeln('Programmet er afslutttet ...');
        halt;                                {Afslutter programmet}
      end;
    end; {IF ch=#27}
  end; {If keypressed}
end; {tjek_esc}

procedure bruger_inddata(                    {Indtastning af inddata til brug for målingen}
            VAR Measure_type,                {Måletype: Solfanger, Metset eller døgklag}
                Tracking_type   : Shortint; {Hvilken trackingtype der foretages målinger med}
            VAR Batchparam : VinkelPointerType; {Delta/låst vinkler samt antal af scanninger}
            VAR v_flade_bred,                {Målefladens venstre begrænsningsflade}
                h_flade_bred,                {Målefladens  højre  begrænsningsflade}
                solvogn_step,                {Solvognens step mellem målinger}
                solarimeter_indsving: Real; {Tid for solarimetrene til indsvingning}
            VAR solvogn_maal_type   : Shortint;{Kontinuert eller step-måling}
            VAR Usr_data        : Usr_dat_t;{Navn på forsøgsansvarlig mv.}
```

```
            VAR tracktid        : Word);     {Tid mellem mÕleserier}

{Er afpr"vet 14/6 - virker tilfredsstillende. - Evt. skal der laves om i GUI'en}

var
  OK_svar           : Boolean; {Om de angivne mÕleparametre er godkendt af brugeren}
  Indlaes_tast      : Char;
  code              : Integer;
  get_real,                                {Til input af reelle tal}
  l_a,l_h,d_a,d_h,                         {Lagring af reelle tal til efter input}
  min_solvogn_step,
  tracktid_min,
  solvognstid,
  indsvingtid       : Real;                {Vµrdien af et indlµst tal}
  batchnummer       : Shortint;            {Hvilket batchnummer der indtastes for}
  mereinput         : Boolean;             {Angiver om der skal foretages flere inputs}
  get_streng        : string[15];          {Til input af strenge - konverteres til tal}
  get_char          : char;                {Til input af karakterer}

FUNCTION streng_til_real(
           min_graense,max_graense:Real;
           outputkolonne          :Integer;
           decimal:Boolean):real;          {TRUE=Decimal, FALSE=Ingen decimal}

var
  x_taeller         : Shortint;            {Bruges til styring af udskrevne mellemrum}

begin
  repeat
    code:=0;                               {Nulstiller fejlkode}
    gotoxy(14+6*batchnummer,outputkolonne);
    Writeln;
    gotoxy(14+6*batchnummer,outputkolonne);
    get_char:=Readkey;                     {Aflµser tastaturbuffer}
    Write(get_char);                       {Udskriver tastetryk}
    IF get_char=#0 THEN                    {Der er trykket pÕ specialtast}
    begin
      get_char:=Readkey;                   {Aflµser tastaturbuffer}
      IF get_char=#107 THEN                {Der er tastet Alt+F4}
        mereinput:=FALSE;                  {Sµtter flag der stopper indtastning}
    end
    else
    begin
      readln(get_streng);                  {Indlµser vinkel som strengvµrdi}
      get_streng:=get_char+get_streng;
                                           {Lµgger den f°rst indtastede karakter til}
      val(get_streng,get_real,code);
                                           {Omformer streng til talvµrdi}
    end; {IF ... else}
  until ((get_real>=min_graense) AND (get_real<=max_graense) AND (code=0)) OR (mereinput=FALSE);
  IF mereinput THEN                        {Der skal tastes mere data ind}
  begin
    gotoxy(14+6*batchnummer,outputkolonne);
    for x_taeller:=WhereX to 79 do
    write(' ');                            {Sletter tidligere udkrifter pÕ resten af linjen}
    gotoxy(14+6*batchnummer,outputkolonne);
    IF decimal THEN
      write(get_real:5:1)
    else
      write(get_real:5:0);
    streng_til_real:=get_real;             {Tildeler vµrdi til funktionen}
  end {IF mereinput}
```

```
  else
    streng_til_real:=1000;                    {Data skal ignoreres}
end; {streng_til_real}

begin
  OK_svar:=FALSE;
  locked_azimut:=0;                           {Lõst azimutvinkel}
  locked_haeld:=0;                            {Lõst hµldningsvinkel}
  delta_azimut:=0;                            {Azimutforskydning}
  delta_haeld:=0;                             {Solh"jdeforskydning}
  d_a:=0;d_h:=0;l_a:=0;l_h:=0;                {Nulstilling af inputvinkler}
  min_solvogn_step:=1/M3_puls_length;         {Den minimale step-afstand for solvogn - 1 puls}
  min_solvogn_step:=(trunc(min_solvogn_step*1000)+1)/1000;
                                              {Afskµrer decimaler efter 3. decimal}

  TopBP:=NIL;
  BundBP:=NIL;
  clrscr;
  gotoxy(1,1);write('Fors"gsansvarlig  : ');Readln(Usr_data.navn);
  gotoxy(1,2);write('Fors"gsmateriale  : ');Readln(Usr_data.materiale);
  gotoxy(1,3);write('Fors"gsbemµrkning : ');Readln(Usr_data.bemaerkning);

  WHILE OK_svar<>TRUE DO
  begin
    gotoxy(1,4);                              {MÕletype}
    write('MÕletype (1 : Solfanger, 2 : Metset, 3 : Dµklag)');
    gotoxy(68,4);write(':');
    repeat                                    {Gentager lµsning indtil en gyldig vµrdi er opnÕet}
      Indlaes_tast:=Readkey;
    until ((Indlaes_tast='1') OR (Indlaes_tast='2') OR (Indlaes_tast='3'));
    val(Indlaes_tast,Measure_type,code);      {Den numeriske vµrdi af indtastningen findes}
    if code<>0 then                           {SÕ er der en forkert karakter}
      Writeln('Fejl i position: ',Code);
    gotoxy(70,4);
    CASE Measure_type OF
      1 : write('Solfanger');
      2 : write('Metset');
      3 : write('Dµklag');
    end; {CASE Measure_type}

    gotoxy(1, 5);                             {Tracking-type}
    write('Tracking (1=H°jde, 2=Azimut, 3=H°jde og Azimut, 4=Fast position)');
    gotoxy(68,5);write(':');
    repeat                                    {Gentager lµsning indtil en gyldig vµrdi er opnÕet}
      Indlaes_tast:=Readkey;
    until ((Indlaes_tast='1') OR (Indlaes_tast='2') OR (Indlaes_tast='3') OR (Indlaes_tast='4'));
    val(Indlaes_tast,Tracking_type,code);     {Den numeriske vµrdi af indtastningen findes}
    if code<>0 then                           {SÕ er der en forkert karakter}
      Writeln('Fejl i position: ',Code);
    gotoxy(70,5);
    CASE Tracking_type OF
      1 : write('H°jde');
      2 : write('Azimut');
      3 : write('H°jde/Az');
      4 : write('Fast pos.');
    end; {CASE Tracking_type}

    gotoxy(1,7);
    Write('Angivelse af indfaldsvinkler i azimut/hµldningsretning samt evt. faste vinkler');
    gotoxy(1,8);
    Write('Batchk°rsel nummer     1     2     3     4     5     6     7     8     9     10');
    gotoxy(1,9);
    Textcolor(Yellow);                        {Sµtter tekstfarce til Gul}
```

```
{ 9}Writeln('Indfaldsvinkel');
    Textcolor(LightGray);                    {Sætter tekstfarve til Grå}
{10}Writeln('Azimut');
{11}Writeln('Hældning');
    Textcolor(Yellow);                       {Sætter tekstfarve til Gul}
{12}Writeln('Faste vinkler');
    Textcolor(LightGray);                    {Sætter tekstfarve til Grå}
{13}Writeln('Fast azimut');
{14}Writeln('Fast hældning');
{15}Write('Antal scanninger');
    gotoxy(1,17);
{17}Writeln('Ovenstående værdier skal være indenfor følgende intervaller');
{18}Writeln('Indfaldsvinkler : min    0 grader. Max   90 grader');
{19}Writeln('Fast azimut     : min ',min_azimut:4,' grader. Max ',max_azimut:4,' grader');
{20}Writeln('Fast hældning   : min ',min_haeldning:4,' grader. Max ',max_haeldning:4,' grader');
{21}Writeln('Hvis der ønskes uendelig mange scanninger anføres "0" som inddata');
{22}Writeln('For at afslutte indtastning af vinkler tastes Alt+F4 et vilkårligt sted');
    TopBP:=NIL;
    mereinput:=TRUE;
    batchnummer:=0;
    REPEAT
      batchnummer:=batchnummer+1;
      CASE tracking_type OF
        1 : begin                            {Solhøjdetracking}
              gotoxy(14+6*batchnummer,10);Write('-----');
              gotoxy(14+6*batchnummer,14);Write('-----');
              d_h:=streng_til_real(-90,90,11,TRUE);{Indlæst værdi overføres til hjælpevariabel}
              IF d_h<>1000 THEN      {SÅ er der ikke tastet Alt+F4 i sidste indtastning}
              begin
                l_a:=streng_til_real(min_azimut,max_azimut,13,TRUE);
                                     {Indlæst værdi overføres til hjælpevariabel}
                if l_a<>1000 THEN
                  antal_step:=round(streng_til_real(0,100,15,FALSE));
                                     {Indlæst værdi overføres til hjælpevariabel}
              end;
            end; {1}
        2 : begin
              gotoxy(14+6*batchnummer,11);Write('-----');
              gotoxy(14+6*batchnummer,13);Write('-----');
              d_a:=streng_til_real(-90,90,10,TRUE);{Indlæst værdi overføres til hjælpevariabel}
              IF d_a<>1000 THEN      {SÅ er der ikke tastet Alt+F4 i sidste indtastning}
              begin
                l_h:=streng_til_real(min_haeldning,max_haeldning,14,TRUE);
                                     {Indlæst værdi overføres til hjælpevariabel}
                IF l_h<>1000 THEN
                  antal_step:=round(streng_til_real(0,100,15,FALSE));
                                     {Indlæst værdi overføres til hjælpevariabel}
              end;
            end; {2}
        3 : begin
              gotoxy(14+6*batchnummer,13);Write('-----');
              gotoxy(14+6*batchnummer,14);Write('-----');
              d_a:=streng_til_real(-90,90,10,TRUE);{Indlæst værdi overføres til hjælpevariabel}
              IF d_a<>1000 THEN      {SÅ er der ikke tastet Alt+F4 i sidste indtastning}
              begin
                d_h:=streng_til_real(-90,90,11,TRUE);
                                     {Indlæst værdi overføres til hjælpevariabel}
                IF d_h<>1000 THEN
                  antal_step:=round(streng_til_real(0,100,15,FALSE));
                                     {Indlæst værdi overføres til hjælpevariabel}
              end;
            end; {3}
```

```
        4 : begin
              gotoxy(14+6*batchnummer,10);Write('-----');
              gotoxy(14+6*batchnummer,11);Write('-----');
              l_a:=streng_til_real(min_azimut,max_azimut,13,TRUE);{Indlæst værdi overf°res til
hjælpevariabel}
              IF l_a<>1000 THEN      {SÕ er der ikke tastet Alt+F4 i sidste indtastning}
              begin
                l_h:=streng_til_real(min_haeldning,max_haeldning,14,TRUE);
                                      {Indlæst værdi overf°res til hjælpevariabel}
                IF l_h<>1000 THEN
                  antal_step:=round(streng_til_real(0,100,15,FALSE));
                                      {Indlæst værdi overf°res til hjælpevariabel}
              end;
            end; {4}
      end; {CASE tracking_type}
      IF ((d_a<>1000) AND (d_h<>1000) AND (l_a<>1000) AND (l_h<>1000)) THEN
      begin
        new(batchparam);              {Reserverer plads til pointer}
        batchparam^.delta_azimut :=d_a;     {Indlæste værdier overf°res til pointerstruktur}
        batchparam^.delta_haeld  :=d_h;
        batchparam^.locked_azimut:=l_a;
        batchparam^.locked_haeld :=l_h;
        batchparam^.scanning:=antal_step;
        batchparam^.naeste:=NIL;
        IF TopBP=NIL THEN             {F°rste gang l°kken k°rer}
          TopBP:=batchparam
        else
          BundBP^.naeste:=batchparam;
        BundBP:=batchparam;           {Indsætter data pÕ nederste plads}
      end
      else
        mereinput:=FALSE;             {SÕ skal der ikke indlæses mere inddata}

    UNTIL ((mereinput=FALSE) AND (batchnummer>1)) OR (batchnummer=10);
                                      {Der er indtastet de n°dvendige data}

    Batchparam:=TopBP;

    Window(1,17,80,25);
    Clrscr;

    Window(1,1,80,25);

    IF measure_type=3 THEN            {Udf°res hvis der skal mÕles med solvogn}
    begin
      gotoxy(1,17);                   {Angivelse af mÕlefladens afgrænsning}
      Write('Afstand fra midt til venstre afgrænsning af mÕleflade (< ',v_arm_bred,' mm)');
      gotoxy(68,17);Write(':');
      repeat
        gotoxy(70,17);Writeln;        {Sletter inddatalinje}
        gotoxy(70,17);readln(v_flade_bred);
      until (v_flade_bred>0) AND (v_flade_bred<=v_arm_bred);
      gotoxy(1,18);
      Write('Afstand fra midt til  h°jre  afgrænsning af mÕleflade (< ',h_arm_bred,' mm)');
      gotoxy(68,18);Write(':');
      repeat
        gotoxy(70,18);Writeln;        {Sletter inddatalinje}
        gotoxy(70,18);readln(h_flade_bred);
      until (h_flade_bred>0) AND (h_flade_bred<=h_arm_bred);

      gotoxy(1,20);
      Write('Styring af bevægelse for solvogn (1=Step, 2=Kontinuert)');
      gotoxy(68,20);Write(':');
```

```
  repeat                                  {Gentager indtil gyldigt tastetryk}
    Indlaes_tast:=Readkey;
  until ((Indlaes_tast='1') OR (Indlaes_tast='2'));
  val(Indlaes_tast,solvogn_maal_type,code);
  if code<>0 then                         {SÕ er der en forkert karakter}
    Writeln('Fejl i position: ',Code);
  gotoxy(70,20);
  CASE solvogn_maal_type OF
    1 : begin {Step-mÕling}
          write('Step');
          gotoxy(1,21);
          write('Afstand mellem mÕlepunkter (min. ',min_solvogn_step:6:4,' mm)');
          gotoxy(68,21);Write(':');
          repeat
            gotoxy(70,21);Writeln;
            gotoxy(70,21);
            readln(solvogn_step);
          until solvogn_step>=min_solvogn_step;
          gotoxy(1,22);
          write('Pause (ms) til indsvingning af solarimetre inden mÕling');
          gotoxy(68,22);Write(':');
          repeat
            gotoxy(70,22);Writeln;
            gotoxy(70,22);
            readln(solarimeter_indsving);
          until solarimeter_indsving>=0;
        end; {1}
    2 : begin {Kontinuert mÕling}
          write('Kontinuert');
          gotoxy(1,21);
          Write('Afstand mellem mÕlepunkter (min. ',min_solvogn_step:6:4,' mm)');
          gotoxy(68,21);Write(':');
          repeat
            gotoxy(70,21);Writeln;
            gotoxy(70,21);
            readln(solvogn_step);
          until solvogn_step>=min_solvogn_step;
        end; {2}
  end; {CASE solvogn_maal_type OF}
end; {IF measure_type=3}

                                          {Tid mellem drejning/vipning konstruktionen}
IF measure_type=3 THEN                    {Beregning af tid til k°rsel med solvogn}
begin
  IF solarimeter_indsving>motor_pause THEN
    indsvingtid:=solarimeter_indsving
  else
    indsvingtid:=motor_pause;
  solvognstid:=((v_flade_bred+h_flade_bred)/solvogn_step)*indsvingtid +
               (v_flade_bred+h_flade_bred)*solvogn_hast;
end
else
  solvognstid:=0;
tracktid_min:=(motortid+solvognstid)/1000;{Omregner fra milisekunder til sekunder}
gotoxy(1,23);Write('Antal sekunder mellem mÕleserier (min. ',tracktid_min:6:2,' sekunder');
repeat
  gotoxy(70,23);Writeln;                  {Sletter indtastningslinjen}
  gotoxy(70,23);readln(tracktid);
until tracktid>=tracktid_min;             {SÕ er der den tid der skal bruges til drejning}

gotoxy(1,25);Write('OK at mÕle med ovenstÕende parametre ? (J/N)');
gotoxy(68,25);Write(':');
```

```
    repeat
      Indlaes_tast:=Readkey;
    until ((Indlaes_tast='j') OR (Indlaes_tast='J') OR (Indlaes_tast='n') OR (Indlaes_tast='N'));
    IF (Indlaes_tast='j') OR (Indlaes_tast='J') THEN
    begin
      OK_Svar:=TRUE;
      gotoxy(70,25);
      write('J');
    end
    ELSE
    begin
      OK_Svar:=FALSE;
      gotoxy(70,25);
      write('N');
    end; {ELSE}
  end; {While}
end; {bruger_inddata}

begin
end.
```

B.6 Kildekode til MATH.TPU

```
unit math;
{Definerer matematiske hjælpefunktioner til tracking-system}

interface

uses
  crt,dos,data,graph,hpibtool;              {data er kun med i testdelen af programmet}

CONST
   np = 2;

TYPE
   RealArrayNP     = ARRAY [1..np] OF real;
   RealArrayNPbyNP = ARRAY [1..np,1..np] OF real;
   IntegerArrayNP  = ARRAY [1..np] OF integer;

function  DegreeToRad(x:real)    : Real;     {Translate degrees to radians}
function  RadToDegree(x:real)    : Real;     {Translate radians to degrees}
function  tan(x:real)            : Real;     {Calculates tangens(x) where x is an angle}
function  arcsin(x:real)         : Real;     {Calculates arcsin of angle 'x'}
function  arccos(x:real)         : Real;     {Calculates arccos of angle 'x'}
function  daynr(yy,mon,dat:word): Word;      {Calculates the number of the day}
function  hund_diff(yy ,mm ,dd , tim, min,  {Beregner hundrede-dele sekund mellem to tidspunkter}
            sek ,hun, yy1,mm1, dd1,         {Tidspunkt '0' i årstal, måned, dag etc.}
            tim1,min1,sek1,hun1 : Word)     {Tidspunkt '1' i årstal, måned, dag etc.}
                                : Longint;
PROCEDURE ludcmp(VAR a     : RealArrayNPbyNP;{LU-decomposition - Fra Numerical recipes}
                     n     : Integer;
                 VAR indx  : IntegerArrayNP;
                 VAR d     : real);
PROCEDURE lubksb(VAR a     : RealArrayNPbyNP;{LU-decomposition - Forward/backward substitution}
                     n     : Integer;
                 VAR indx  : IntegerArrayNP;
                 VAR b     : RealArrayNP);
PROCEDURE usrfun(VAR x     : RealArrayNP;    {Udregner funktionsværdi for N-R i 'n' dimensioner}
                     n     : Integer;        {x = initialgæt, n=antal ubekendte}
                 VAR alpha: RealArrayNPbyNP;{alpha = partiel afledede af funktioner}
                 VAR beta : RealArrayNP;    {beta = negative af funktionsværdi}
                 K1, K2,                    {forskydning i breddeplan og længdeplan}
                 al_s,ga_s: Real);          {gamma_s og alfa_s i radianer : Solens position}
PROCEDURE mnewt(ntrial     : integer;       {Newton-Raphson i 'n' dimensioner}
            VAR x          : RealArrayNP;   {ntrial = Antal iterationer}
            n              : integer;       {x=initialgæt}
            tolx,tolf,                      {tolx=tolerance på x-akse, toly=tolerance på y-akse}
            K1, K2,                         {forskydning i breddeplan og længdeplan}
            al_s,ga_s      : Real;          {gamma_s og alfa_s i radianer : Solens position}
            graphicstate   : Boolean);      {Graphicstate: TRUE=grafikskærm, FALSE=txtskærm}

procedure stop_maaling;                     {Afbryder relæer og stopper måling}

IMPLEMENTATION

function DegreeToRad(x:real)     : Real;    {Omregner grader til radianer}
begin
  DegreeToRad:=(x*pi)/180;
end; {DegreeToRad}

function RadToDegree(x:real)     : Real;    {Omregner radianer til grader}
begin
```

```
  RadToDegree:=x*180/pi;
end; {RadToDegree}

function tan(x:real)              : Real;     {Udregner tangens til vinklen 'x'}
begin
  IF cos(x)=0 THEN                            {Hvis cos(x)=0 sÕ er tan(x)=uendeligt}
    x:=Arccos(0.0001);                       {'x' sættes til en vinkel lidt ved siden af 90 grader}
  tan:=sin(x)/cos(x);
end; {tan}

function arcsin(x:real)           : Real;     {Calculates arcsin(x) where x is an angle}
begin
  IF x=1 THEN                                 {G"r at Arcsin kan udregnes for 'x'=1}
    x:=0.9999;
  IF x=-1 THEN                                {G"r at Arcsin kan udregnes for 'x'=-1}
    x:=-0.9999;
  arcsin:=arctan(x/sqrt(1-sqr(x)));
end; {arcsin}

function arccos(x:real)           : Real;     {Udregner Arccos til vinkel 'x'}
var
  arccos_t          : Real;
begin
  arccos_t:=arctan(sqrt(1-sqr(x))/x);
  IF arccos_t<0 THEN
    arccos:=arccos_t+pi                       {Flytter vinklen over i 2. kvadrant}
  ELSE
    arccos:=arccos_t;
end; {arccos}

function daynr(yy,mon,dat:word) : Word;       {Calculates the number of the day}
var
  m_nr, dag         : Integer;                {MÕnednummer og Dagnummer}
begin
  m_nr:=0; dag:=0;
  repeat
    inc(m_nr);
    case m_nr of
      1,3,5,7,8,10,12 : dag:=dag+31;          {MÕneder med 31 dage}
      2               : if (yy div 4) * 4=yy then dag:=dag+29 else dag:=dag+28; {SkudÕr ?}
      4,6,9,11        : dag:=dag+30;          {MÕneder med 30 dage}
    end; {case m_nr ...}
  until m_nr=mon-1;                           {SÕ er dagene for alle mÕneder blevet beregnet}
  daynr:=dag+dat;                             {Udregner dagens nummer}
end; {daynr}

function hund_diff(yy ,mm ,dd ,              {Beregner hundrede-dele sekund mellem to tidspunkter}
           tim ,min ,sek ,hun ,               {Tidspunkt '0' i Õr, mÕned, dag, time etc.}
           yy1,mm1,dd1,                       {Tidspunkt '1' i Õr, mÕned, dag, time etc.}
           tim1,min1,sek1,hun1  : word)
                                : Longint;
begin
  hund_diff:=((((daynr(yy1,mm1,dd1)-1)*24+tim1)*60+min1)*60+sek1)*100+hun1 -
             (((((daynr(yy, mm, dd )-1)*24+tim )*60+min )*60+sek )*100+hun)
end; {hund_diff}

PROCEDURE ludcmp(VAR a    : RealArrayNPbyNP;{LU-decomposition - Fra Numerical recipes}
                     n    : integer;
                 VAR indx : IntegerArrayNP;
                 VAR d    : real);
CONST
   tiny = 1.0e-20;
```

```
VAR
   k,j,imax,i: integer;
   sum,dum,big: real;
   vv: ^RealArrayNP;
BEGIN
   new(vv);
   d := 1.0;
   FOR i := 1 TO n DO BEGIN
      big := 0.0;                          {'big' indeholder st°rste element i matrice}
      FOR j := 1 TO n DO
         IF abs(a[i,j]) > big THEN big := abs(a[i,j]);
      IF big = 0.0 THEN BEGIN              {Alle elementer er negative - matricen er singulær}
         writeln('pause in LUDCMP - singular matrix');
         stop_maaling;
      END;
      vv^[i] := 1.0/big
   END;
   FOR j := 1 TO n DO BEGIN
      FOR i := 1 TO j-1 DO BEGIN
         sum := a[i,j];
         FOR k := 1 TO i-1 DO
            sum := sum-a[i,k]*a[k,j];
         a[i,j] := sum
      END;
      big := 0.0;
      FOR i := j TO n DO BEGIN
         sum := a[i,j];
         FOR k := 1 TO j-1 DO
            sum := sum-a[i,k]*a[k,j];
         a[i,j] := sum;
         dum := vv^[i]*abs(sum);
         IF dum >= big THEN BEGIN
            big := dum;
            imax := i
         END
      END;
      IF j <> imax THEN BEGIN
         FOR k := 1 TO n DO BEGIN
            dum := a[imax,k];
            a[imax,k] := a[j,k];
            a[j,k] := dum
         END;
         d := -d;
         vv^[imax] := vv^[j]
      END;
      indx[j] := imax;
      IF a[j,j] = 0.0 THEN a[j,j] := tiny;
      IF j <> n THEN BEGIN
         dum := 1.0/a[j,j];
         FOR i := j+1 TO n DO
            a[i,j] := a[i,j]*dum
      END
   END;
   dispose(vv)
END;{ludcmp}

PROCEDURE lubksb(VAR a     : RealArrayNPbyNP;{LU-decomposition - Forward/backward substitution}
                     n     : Integer;
                 VAR indx  : IntegerArrayNP;
                 VAR b     : RealArrayNP);
VAR
   j,ip,ii,i       : Integer;
```

```
   sum              : Real;
BEGIN
   ii := 0;
   FOR i := 1 TO n DO BEGIN
      ip := indx[i];
      sum := b[ip];
      b[ip] := b[i];
      IF ii <> 0 THEN
         FOR j := ii TO i-1 DO
            sum := sum-a[i,j]*b[j]
      ELSE IF sum <> 0.0 THEN
         ii := i;
      b[i] := sum
   END;
   FOR i := n DOWNTO 1 DO BEGIN
      sum := b[i];
      FOR j := i+1 TO n DO
         sum := sum-a[i,j]*b[j];
      b[i] := sum/a[i,i]
   END
END;{lukbsb}

PROCEDURE usrfun(VAR x     : RealArrayNP;     {Udregner funktionsvørdi for N-R i 'n' dimensioner}
                     n     : integer;         {x = initialgøt, n=antal ubekendte}
                 VAR alpha: RealArrayNPbyNP;{alpha = partiel afledede af funktioner}
                 VAR beta : RealArrayNP;     {beta = negative af funktionsvørdi}
                 K1, K2   : Real;            {Forskydning i breddeplan og løngdeplan}
                 al_s,ga_s: Real);           {gamma_s og alfa_s i radianer : Solens position}

begin
  beta[1]:=    -(tan(K1)*cos(ga_s)*cos(al_s)*cos(x[2])*cos(x[1]) +
                 tan(K1)*sin(ga_s)*cos(al_s)*sin(x[2])*cos(x[1]) +
                 tan(K1)*          sin(al_s)          *sin(x[1]) -
                         cos(ga_s)*cos(al_s)*cos(x[2])*sin(x[1]) -
                         sin(ga_s)*cos(al_s)*sin(x[2])*sin(x[1]) +
                                   sin(al_s)          *cos(x[1])   );
  beta[2]:=   -(-tan(K2)*cos(ga_s)*cos(al_s)*cos(x[1])*cos(x[2]) -
                 tan(K2)*sin(ga_s)*cos(al_s)*cos(x[1])*sin(x[2]) -
                 tan(K2)          *sin(al_s)*sin(x[1])           -
                         sin(ga_s)*cos(al_s)          *cos(x[2]) +
                         cos(ga_s)*cos(al_s)          *sin(x[2])   );
  alpha[1,1]:= - tan(K1)*cos(ga_s)*cos(al_s)*cos(x[2])*sin(x[1]) -
                 tan(K1)*sin(ga_s)*cos(al_s)*sin(x[2])*sin(x[1]) +
                 tan(K1)*          sin(al_s)          *cos(x[1]) -
                         cos(ga_s)*cos(al_s)*cos(x[2])*cos(x[1]) -
                         sin(ga_s)*cos(al_s)*sin(x[2])*cos(x[1]) -
                                   sin(al_s)          *sin(x[1])    ;
  alpha[1,2]:= - tan(K1)*cos(ga_s)*cos(al_s)*sin(x[2])*cos(x[1]) +
                 tan(K1)*sin(ga_s)*cos(al_s)*cos(x[2])*cos(x[1]) +
                         cos(ga_s)*cos(al_s)*sin(x[2])*sin(x[1]) -
                         sin(ga_s)*cos(al_s)*cos(x[2])*sin(x[1])    ;
  alpha[2,1]:=   tan(K2)*cos(ga_s)*cos(al_s)*sin(x[1])*cos(x[2]) +
                 tan(K2)*sin(ga_s)*cos(al_s)*sin(x[1])*sin(x[2]) -
                 tan(K2)          *sin(al_s)*cos(x[1])           ;
  alpha[2,2]:=   tan(K2)*cos(ga_s)*cos(al_s)*cos(x[1])*sin(x[2]) -
                 tan(K2)*sin(ga_s)*cos(al_s)*cos(x[1])*cos(x[2]) +
                         sin(ga_s)*cos(al_s)          *sin(x[2]) +
                         cos(ga_s)*cos(al_s)          *cos(x[2])    ;
end;{usrfun}

PROCEDURE mnewt(ntrial          : integer;  {Newton-Raphson i 'n' dimensioner}
            VAR x               : RealArrayNP;{ntrial = Antal iterationer}
```

```
            n                    : integer;  {x=initialgµt af fladeorientering}
            tolx,tolf,                       {tolx=tolerance pÕ x-akse, toly=tolerance pÕ y-akse}
            K1, K2,                          {Forskydning i breddeplan og lµngdeplan}
            al_s,ga_s            : Real;     {gamma_s og alfa_s i radianer : Solens position}
            graphicstate         : Boolean); {Graphicstate TRUE=Grafikskµrm, FALSE=Txtskµrm}
LABEL 99;                                    {Definition af labels (hoppepunkter) der bruges}
VAR
  relae_taeller,                             {Tµller til lukning af relµer}
  k,i              : integer;
  errx,errf,d      : real;
  beta             : ^RealArrayNP;           {Indeholder h"jresiden i matricen}
  alpha            : ^RealArrayNPbyNP;       {Indeholder venstresiden af matricen}
  indx             : ^IntegerArrayNP;
BEGIN
  new(beta);                                 {Reservation af lagerplads til pointer}
  new(alpha);                                {Reservation af lagerplads til pointer}
  new(indx);                                 {Reservation af lagerplads til pointer}
  FOR k := 1 TO ntrial DO BEGIN
     usrfun(x,n,alpha^,beta^,K1,K2,al_s,ga_s);{Udregning af funktionerne samt deres afledede}
     errf := 0.0;
     FOR i := 1 TO n DO
        errf := errf+abs(beta^[i]);          {Opsummerer funktionsvµrdier}
     IF errf <= tolf THEN GOTO 99;           {Unders°ger om funktionsvµrdisum er indenfor °nsket
                                              tolerance - Hvis "JA" sÕ afsluttes proceduren}
     ludcmp(alpha^,n,indx^,d);               {LU-dekomposition af den opstillede matrix}
     lubksb(alpha^,n,indx^,beta^);           {LU-Backsubstitution af den opstillede matrix}
     errx := 0.0;
     FOR i := 1 TO n DO BEGIN
        errx := errx+abs(beta^[i]);          {Opsummerer funktionsvµrdier}
        x[i] := x[i]+(beta^[i]/2);           {L°sning sµttes til tidligere l°sning plus
                                              dµmpning af l°sningsµndring. Dµmpning g°r at
                                             iterationen konvergerer ved stor vinkelforskydning,
                                              men iterationen lµngere om at konvergere}
     END;
     IF errx <= tolx THEN GOTO 99;           {Unders°ger om funktionsvµrdisum er indenfor °nsket
                                              tolerance - Hvis "JA" sÕ afsluttes proceduren}
  END;
  IF k=ntrial then                           {Hvis iterationen er stoppet pga. antal iterationer}
  begin
    Closegraph;                              {Sµtter computeren i tekstmode}
    Writeln('Fladepositionsberegninger ikke konvergeret indenfor ',k,' iterationer.');
    Writeln('Fors°g med en mindre indfaldsvinkel eller foretag mÕlingerne ved en lav solh°jde');
    stop_maaling;
  end; {IF k=ntrial}
99:
  dispose(indx);                             {Frigiver lagerplads fra pointer}
  dispose(alpha);                            {Frigiver lagerplads fra pointer}
  dispose(beta);                             {Frigiver lagerplads fra pointer}
  x[1]:=pi/2-x[1];                           {Omregner fra hµldn. med lodret til hµld med vandret}
end;{mnewt}

procedure stop_maaling;                      {Afbryder relµer og stopper mÕling}
var
  relae_taeller   : Shortint;                {Tµller til afbrydning af relµerne 1 til 6}
begin
  Writeln;
  Write('Evt. Õbne motorrelµer lukkes ');
  for relae_taeller:=1 to 6 do
  begin
    setdio(relae_taeller,0);                 {Lukker relµ-nummer 'relae_taeller'}
    write('.');
    delay(250);
```

```
  end; {for relae_taeller}
  writeln;
  writeln('Alle relµer er nu afbrudt - mÖling stopper');
  getdate(aar,mnd,dat,dag);                    {Aflµser dato i computerens ur}
  gettime(tim,min,sek,hun);                    {Aflµser tidspunkt i computerens ur}
  writeln;
  writeln('MÖling stoppet ',ugedag[dag],' d. ',dat,'/',mnd,'-',aar,
          ' kl. ',tim,':',min,':',sek);
  writeln;
  writeln('Tryk pÖ en tast !');
  readln;
  halt;                                        {Stopper program-udf°relse}
end; {stop_maaling}

begin
end.
```

B.7 Kildekode til UDDATA.TPU

```
unit uddata;

Interface

uses
  crt, dos, data, graph, inddata;

procedure screen1;                          {Udvider uddata-skµrm til solfangermÕlinger}

procedure screen2;                          {Udvider uddata-skµrm til Metset-mÕlinger}

procedure screen3(                          {Opbygger uddata-skµrm til dµklagsmÕlinger}
            flade_haeld,                    {MÕelfladens hµldning}
            flade_azimut,                   {MÕlefladens azimut}
            alfa_s_deg,                     {Solh°jden i grader}
            azimut_deg,                     {Solens azimut i grader}
            indfaldsvinkel,                 {Indfaldsvinkel for solstrÕling}
            v_flade_bred,                   {MÕlefladens venstre begrµnsning}
            h_flade_bred        : Real;     {MÕlefladens  h°jre  begrµnsning}
            Usr_data            : Usr_dat_t);{Navn pÕ fors°gsansvarlig mv.}

procedure screen3_data(                     {Opbygger uddata-skµrm 3 med variable vµrdier}
            nuv_solvogn_position,           {Solvognens nuvµrende position}
            dvm_value_1,                    {MÕlevµrdi fra f°rste solarimeter}
            dvm_value_2,                    {MÕlevµrdi for andet  solarimeter}
            dvm_value_3         : Real;     {MÕlevµrdi for tredje solarimeter}
            oldtid              : Time_type);{Tidspunkt for sidste sovognstraversering}

function tjek_fil(out_fil:Str80):Boolean;   {Unders°ger gyldigheden af et brugerangivet filnavn}

procedure vent_tid_diff(                    {Beregner forskel mellem stoptid og nytid og}
            nytid, stoptid : Longint;       {udskriver denne i (x,y)=(10,25)}
            antal_step : Longint);

procedure slet_linje(y_koor:Byte);          {Sletter linje nummer y_koor pÕ skµrmen}

implementation

procedure screen1;                          {Udvider uddata-skµrm til solfangermÕlinger}
begin
  {DENNE PROCEDURE ER IKKE LAVET FÄRDIG ENDNU}
end;

procedure screen2;                          {Udvider uddata-skµrm til Metset-mÕlinger}
begin
  {DENNE PROCEDURE ER IKKE LAVET FÄRDIG ENDNU}
end;

procedure screen3(                          {Opbygger uddata-skµrm til dµklagsmÕlinger}
            flade_haeld,                    {MÕelfladens hµldning}
            flade_azimut,                   {MÕlefladens azimut}
            alfa_s_deg,                     {Solh°jden i grader}
            azimut_deg,                     {Solens azimut i grader}
            indfaldsvinkel,                 {Indfaldsvinkel for solstrÕling}
            v_flade_bred,                   {MÕlefladens venstre begrµnsning}
            h_flade_bred        : Real;     {MÕlefladens  h°jre  begrµnsning}
            Usr_data            : Usr_dat_t);{Navn pÕ fors°gsansvarlig mv.}

var
```

```
  yy_str, mon_str, day_str      : STRING[4];{Strengvørdi af Ór, mÓned og dato}
  grafikdriver,                             {Angiver hvilken grafikdriver der er valgt}
  grafiktilstand,                           {Angiver grafiktilstand}
  fejlkode                      : Integer;  {Angiver fejlkode i forbindelse med kal af driver}
  outtxt                        : String[80];{Til udskrivning af tekst pÓ grafikskørm}
  v1_txt,v2_txt                 : String[8];{Strengvørdi af høldningsvinkler mv}
  delta_x_ind, delta_y_ind      : Real;     {Inddelingsafstande pÓ x- og y-akse}
  delestregnummer               : Shortint; {Tøller antallet af delestreger pÓ akser}
  x_stregkoor, y_stregkoor      : Integer;  {x- og y-koordinat for delestreger}

begin
  v1_txt:='';v2_txt:='';                    {Nulstilling af variable}
  grafikdriver:=VGA; grafiktilstand:=2;     {Definering af grafikkort og grafikdriver}
  initgraph(grafikdriver, grafiktilstand, '');{Initialisering af grafikkort}
  fejlkode:=graphresult;                    {Unders"ger om der er opstÓet fejl ved grafik-kald}
  if fejlkode<>grOK THEN
  BEGIN
    Writeln('Grafikfejl: ',GraphErrorMsg(Fejlkode));
    Halt;
  END;                                      {SÓ er grafikskørmen defineret}

  SetColor(4);
  OuttextXY(  0,  0,'ESC : Afslut');
  Setcolor(15);
  getdate(aar,mnd,dat,dag);                 {Afløser dato fra PC'ers ur}
  Str(aar,yy_str);                          {Omdanner Órstal til streng}
  Str(mnd,mon_str);                         {Omdanner mÓned til streng}
  IF length(mon_str)=1 THEN mon_str:=' '+mon_str;
  Str(dat,day_str);                         {Omdanner dato til streng}
  IF length(day_str)=1 THEN day_str:=' '+day_str;
  outtxt:=ugedag[dag];
  IF length(ugedag[dag])=6 THEN outtxt:=outtxt+' ' ELSE outtxt:=outtxt;
                                            {Justerer løngde af outtxt}
  outtxt:=outtxt+' d. '+day_str+'/'+mon_str+'-'+yy_str;
  OuttextXY(264,0,outtxt);
  OuttextXY(440,0,'kl.    :  :   (Vintertid)');
  OuttextXY( 56, 15,'TransmittansmÓling af døklag. Laboratoriet for Varmeisolering, DTU');
  Setcolor(2);
  Line(56,26,584,26);
  Setcolor(7);
  outtxt:='Fors°gs-ansvarlige   : '+Usr_data.navn;
  OuttextXY( 72, 30,outtxt);
  outtxt:='Materiale            : '+Usr_data.materiale;
  OuttextXY( 72, 45,outtxt);
  outtxt:='Kommentar            : '+Usr_data.bemaerkning;
  OuttextXY( 72, 60,outtxt);
  Setcolor(2);
  Line(56,70,584,70);
  Setcolor(7);
  Str((90-flade_haeld):4:2,v1_txt);         {Fladens h°jde = 90 - fladehøldning}
  IF (90-flade_haeld)<10 THEN v1_txt:=' '+v1_txt;
                                            {Formatterer output}
  Str(alfa_s_deg:4:2,v2_txt);
  IF alfa_s_deg<10 THEN v2_txt:=' '+v2_txt; {Formatterer output}
  outtxt:='MÓlefladens "h°jde"  :   '+v1_txt+#248+' Solh°jde          :   '+v2_txt+#248;
  outtextXY( 72,75,outtxt);
  v1_txt:='';v2_txt:='';                    {Variable nulstilles}
  Str(abs(flade_azimut):5:2,v1_txt);        {Konverterer tal til streng}
  IF abs(flade_azimut)<100 THEN v1_txt:=' '+v1_txt;
  IF flade_azimut>=0 THEN v1_txt:=' '+v1_txt ELSE v1_txt:='-'+v1_txt;
                                            {Formatterer output}
  Str(abs(azimut_deg):5:2,v2_txt);          {Konverterer tal til streng}
```

```
IF abs(azimut_deg)<100 THEN v2_txt:=' '+v2_txt;
IF azimut_deg>=0 THEN v2_txt:=' '+v2_txt ELSE v2_txt:='-'+v2_txt;

outtxt:='Mðlefladens azimut   : '+v1_txt+#248+' Solazimut          : '+v2_txt+#248;
OuttextXY( 72,90,outtxt);
v1_txt:='';v2_txt:='';
Str(indfaldsvinkel:4:2,v1_txt);
IF indfaldsvinkel<10 THEN v1_txt:=' '+v1_txt;
outtxt:='Rumindfaldsvinkel    :   '+v1_txt+#248;
OuttextXY( 72,105,outtxt);
Setcolor(2);
Line(56,115,584,115);
setcolor(7);
outtxt:='Nuvµrende handling   :';
outtextXY( 72,120,outtxt);
Setcolor(2);
Line(56,130,584,130);
Line(56, 26, 56,130);                    {Lodret linje i venstre side}
Line(584,26,584,130);                    {Lodret linje i h°jre side}
Setcolor(10);
OuttextXY(  0,140,'BestrÕlingsstyrke');  {Tekst til Y-akse}
OuttextXY(  0,150,'pÕ solarimetre [W/m2]');
OuttextXY(508,470,'Solvogn. pos[mm]');   {Tekst til X-akse}
Setcolor(15);
line(40,165,40,465);                     {Opbygger Y-akse}
line(37,168,40,165);                     {Tegner akse-pil}
line(40,165,43,168);
line(40,465,630,465);                    {Opbygger X-akse}
line(627,462,630,465);                   {Tegner akse-pil}
line(630,465,627,468);
delta_y_ind:=0.25;                       {Antal pixels pr Watt solindstrÕling}
delta_x_ind:=1;                          {Antal pixels pr mm solvognsbevµgelse}
delestregnummer:=0;
repeat                                   {Delestreg pÕ y-akse}
  delestregnummer:=delestregnummer+1;
  y_stregkoor:=465-round(delta_y_ind*100*delestregnummer);
  line(37,y_stregkoor,43,y_stregkoor);   {Tegner delestreg}
  str(100*delestregnummer,outtxt);
  IF delestregnummer<10 THEN outtxt:=' '+outtxt;
                                         {Formatterer output}
  outtextXY(0,y_stregkoor-2,outtxt);     {Udskriver inddelingstal pÕ y-aksen}
  setcolor(7);
  line(43,y_stregkoor,490,y_stregkoor);  {Laver streger til baggrundsgitter}
  setcolor(15);
until y_stregkoor<214-delta_y_ind*100;   {Det er ikke plads til flere streger}

delestregnummer:=-1;
repeat                                   {Delestreg pÕ x-akse}
  delestregnummer:=delestregnummer+1;
  x_stregkoor:=40+(delestregnummer*50);
  line(x_stregkoor,462,x_stregkoor,468);
  str(50*delestregnummer-200,outtxt);
  IF abs(50*delestregnummer-200)<100 THEN outtxt:=' '+outtxt;
  IF (50*delestregnummer-200)>=0 THEN outtxt:=' '+outtxt;
  outtextXY(x_stregkoor-20,470,outtxt);
  setcolor(7);
  line(x_stregkoor,165,x_stregkoor,462); {Laver streger til baggrundsgitter}
  setcolor(15);
until (50*delestregnummer-200)>200;      {Der er ikke plads til flere streger}

Setcolor(9);
Line(200,144,240,144);                   {Signaturforklaring 1}
```

```
  OuttextXY(248,140,'Global strõling');
  Setcolor(13);                                {Signaturforklaring 2}
  Line(200,154,240,154);
  OuttextXY(248,150,'Diffus strõling (x10)');
  Setcolor(14);                                {Signaturforklaring 3}
  Line(424,144,464,144);
  OuttextXY(472,140,'Trans. strõling');
  Setcolor(15);
                                               {DER MANGLER UDSKRIVNING AF FORSÏGSANSV MM.}
end; {screen3}

procedure screen3_data(                        {Opbygger uddata-skµrm 3 med variable vµrdier}
            nuv_solvogn_position,              {Solvognens nuvµrende position}
            dvm_value_1,                       {MÕlevµrdi fra f"rste solarimeter}
            dvm_value_2,                       {MÕlevµrdi for andet  solarimeter}
            dvm_value_3          : Real;       {MÕlevµrdi for tredje solarimeter}
            oldtid               : Time_type);{Tidspunkt for sidste sovognstraversering}

var
  tim_str,min_str,                             {Strengvµrdi af time og minut}
  sek_str          : String[2];                {Strengvµrdi af sekund}
  x_plot, y_plot   : Integer;                  {Koordinater hvor der skal afsµttes punkter}
begin
  gettime(tim,min,sek,hun);                    {Aflµser tidspunkt fra PC'ers ur}
  SetFillStyle(0,0);                           {Sµtter udfyldning til baggrundsfarven}
  Setcolor(15);                                {Skifter farve til hvid}
  IF tim<>oldtid.time THEN
  begin                                        {SÕ skal der µndres i timetallet pÕ skµrmen}
    str(tim,tim_str);IF tim<10 THEN tim_str:=' '+tim_str; {Formatterer output}
    Bar(472,0,488,8);                          {Udfylder feltet hvor timetallet skal stÕ}
    OuttextXY(472,0,tim_str);                  {Udskriver timetal}
  end; {IF time<>...}
  IF min<>oldtid.min THEN
  begin                                        {SÕ skal der µndres i minuttallet pÕ skµrmen}
    str(min,min_str);IF min<10 THEN min_str:='0'+min_str; {Formatterer output}
    Bar(496,0,512,8);                          {Udfylder feltet hvor minuttallet skal stÕ}
    OuttextXY(496,0,min_str);                  {Udskriver minuttal}
  end; {IF min<>...}
  IF sek<>oldtid.sek THEN
  begin                                        {SÕ skal der µndres i sekundtallet pÕ skµrmen}
    str(sek,sek_str);IF sek<10 THEN sek_str:='0'+sek_str; {Formatterer output}
    Bar(520,0,536,8);                          {Udfylder feltet hvor sekundtallet skal stÕ}
    OuttextXY(520,0,sek_str);                  {Udskriver sekundtal}
  end;
  x_plot:=round(240+nuv_solvogn_position);     {Udregner x-koordinat}
  y_plot:=round(465-0.25*dvm_value_1);         {Udregner y-koordinat}
  PutPixel(x_plot,y_plot,9);
  y_plot:=round(465-0.25*dvm_value_2);         {udregner y-koordinat}
  PutPixel(x_plot,y_plot,14);
  y_plot:=round(465-2.5* dvm_value_3);         {udregner y-koordinat}
  PutPixel(x_plot,y_plot,13);
end; {screen3_data}

function tjek_fil(out_fil:Str80) : Boolean; {Unders°ger gyldigheden af et brugerangivet filnavn}

var
  dir_navn,fil_navn: str80;                    {Navn pÕ directory og uddatafilen}
  taeller,
  B_Slashpos       : Shortint;
  DirInfo          : SearchRec;
  Doserror,
  Errorcode        : Byte;
```

```
begin
  Errorcode:=0;Doserror:=0;B_Slashpos:=0;   {Nulstiller variabel}
  tjek_fil:=TRUE;
  for taeller:=1 to length(out_fil) do
  begin
    IF copy(out_fil,taeller,1)='\' then
      B_Slashpos:=taeller;                  {Sætter B_slashpos til seneste position hvor der var
                                             et backslash '\'}
    write(B_Slashpos,' ');
  end; {for taeller:= ...}
  IF B_Slashpos=0 THEN                      {Der er ikke noget backslashes. Det gemmes i}
  begin                                     {aktuelle katalog}
    fil_navn:=out_fil;
  end
  else
  begin
    fil_navn:=copy(out_fil,B_Slashpos+1,length(out_fil)-B_Slashpos);
    writeln;
    dir_navn:=copy(out_fil,1,B_Slashpos-1); {Dir-navnet er fulde navn minus filnavn}
    writeln(fil_navn);
    writeln(dir_navn);
    {$I-}                                  {Automatisk I/O-fejltjek afbrydes. Fejltjek:IOResult}
    MKDIR(dir_navn);
    {$I+}                                   {Automatisk I/O-fejltjek tilsluttes}
    Errorcode:=IOResult;                    {Gemmer fejlkode fra FindFirst-kommando}
    Writeln('E ',Errorcode,' D ',Doserror);
    Errorcode:=IOResult;
    writeln('Errorcode : ',errorcode);
    writeln('DosError  : ',doserror);
  end; {IF B_Slashpos=0 ... ELSE}
                                           {Tjek af gyldighed af filnavn - Programmet tjekker}
  FOR taeller:=1 TO length(fil_navn) do     {ikke for flere punktmmer i filnavnet}
  begin
    CASE fil_navn[taeller] OF
      'a' .. 'z',                           {Minuskler}
      'A' .. 'Z',                           {Majuskler}
      '0' .. '9',                           {Cifre}
      '_','^','$','~','!','#','%','&','-',  {Specialtegn}
      '{','}','(',')','@','.'             :;{SÅ skal der ikke ske noget}
      else
        tjek_fil:=FALSE;                    {Der er en ulovlig karakter i filnavnet}
    end {CASE filnavn[] ...};
  end; {FOR taeller:= ...}
end;

procedure vent_tid_diff(                    {Beregner forskel mellem stoptid og nytid og}
            nytid, stoptid : Longint;       {udskriver denne i (x,y)=(10,25)}
            antal_step : Longint);

var
  tid_str,                                  {Omformer antal sekunder til en tekststreng}
  old_tid_str           : String[6];        {"Gammel" værdi tid_str}
  mellem_count          : Shortint;         {Tæller hvor mange mellemrum der skal bruges}
  tid_diff              : Integer;

begin
  IF antal_step<>0 THEN                     {SÅ skrives der ikke noget i sidste tidsstep}
    gotoxy(1,25);Write('Venter i        sekunder');
  old_tid_str:='';                          {Nulstiller old_tid_str}
  while (nytid<stoptid) and (antal_step<>0) do{SÅ er der ikke gået lang tid nok. Ellers}
  begin                                    {er det sidste tidsstep hvorfor der ikke skal pauses}
    mellemrum:='';
```

```
    gettime(tim,min,sek,hun);                    {Aflæser tidspunkt fra computerens ur}
    nytid:=3600*tim+60*min+sek;
    tid_diff:=stoptid-nytid;                     {Beregner antal sekunder til næste tidsstep}
    Str(tid_diff,tid_str);                       {Omformer tal til streng}
    IF tid_str<>old_tid_str THEN                 {Så er der ændret i det der skal udskrives}
    begin
      FOR mellem_count:=1 TO 6-length(tid_str) DO
        mellemrum:=mellemrum+' ';
      gotoxy(10,25);Write(mellemrum,'',tid_str);
      old_tid_str:=tid_str;                      {Overskriver gamle værdi med "nye" værdi}
    end; {IF tid_str<>old_tid_str}
    tjek_esc;                                    {Tester om brugeren har trykket ESC}
  end; {WHILE ()}
end; {vent_tid_diff}

procedure slet_linje(y_koor:Byte);               {Sletter linje nummer y_koor på skærmen}
begin
  window(1,y_koor,80,y_koor);
  Clrscr;
  window(1,1,80,25);
end; {slet_linje}

begin
end.
```

Appendix C. Reserverede navne i styringsprogram

Ved kodningen af styringsprogrammet er der brugt en del forskellige variabelnavne. For at undgå en eventuel sammenblanding af variabelnavne og procedurenavne ved en senere udviddelse af programmet, er her angivet en oversigt over alle unit-, procedure, funktions-, type- og variabelnavne, der er brugt i styringsprogrammet.

C.1 Reserverede UNIT-navne

Nedenfor er anført de enkelte UNITs, der sammen med hovedprogrammet udgør såvel måledelen som måledelen af programmet.

Unitnavn
Adatool
Data
Errvar
Hpibtool
Inddata
Math
Uddata

C.2 Reserverede PROCEDURE-navne

Nedenfor er anført procedurerne, der bruges i programmet. Tillige er det beskrevet i hvilket unit proceduren findes.

PROCEDURE	Findes i UNIT
Beregn_flade_orient	Adatool
Bruger_inddata	Inddata
Esc_maaling	Adatool
Initier	Adatool
Init_solvogn	Adatool
I_vinkel	Adatool
Lubksb	Math
Ludcmp	Math
Mnewt	Math
Nulstil_flade	Adatool
Puls_taeller	Adatool
Roter_flade	Adatool
Screen1	Uddata
Screen2	Uddata
Screen3	Uddata

Screen3_data	Uddata
Set_start_pos	Adatool
Slet_linje	Uddata
Solpos	Adatool
Solvogn_outputfil	Adatool
Solvogn_travers_step	Adatool
Solvogn_travers_kont	Adatool
Stop_maaling	Math
Stop_tekst	Adatool
Tidsstep	Adatool
Tjek_esc	Inddata
Usrfun	Math
Vinkel_taeller	Adatool
Vent_tid_diff	Uddata

C.3 Reserverede FUNCTION-navne

Nedenfor er anført funktionerne, der bruges i programmet. Tillige er det beskrevet i hvilket unit funktionen findes.

Funktionsnavn	Findes i unit
Arccos	Math
Arcsin	Math
Daynr	Math
Degreetorad	Math
Hund_diff	Math
Initadatool	Hpibtool
Getadcchannel	Hpibtool
Gethpcount	Hpibtool
Nulstilcouter	Hpibtool
Radtodegree	Math
Setdio	Hpibtool
Streng_til_real	Hpibtool
Tan	Math
Tjek_fil	Uddata

C.4 Reserverede TYPE-navne

I dette afsnit er de typenavne, der bruges i programmet, anført. Typenavnene beskriver forskellige datatyper, der er defineret. For hver typenavn er det anført hvor denne optræder med såvel procedurenavn og unitnavn. Typer, hvis beliggenhed er anført som "N/A - Global", er defineret globalt, hvilket vil sige at hele programmet (incl. hovedprogrammet) har mulighed for at bruge disse.

Typenavn	Findes i PROCEDURE/FUNCTION	Findes i UNIT
Integerarraynp	N/A - Global type	Math
Realarraynp	N/A - Global type	Math
Realarraynpbynp	N/A - Global type	Math
Real_filtype	N/A - Global type	Data
Solpointer	Solvogn_travers_kont	Adatool
Solpointer	Solvogn_travers_step	Adatool
Solpointertype	Solvogn_travers_kont	Adatool
Solpointertype	Solvogn_travers_step	Adatool
Str80	N/A - Global type	Data
Time_type	N/A - Global type	Data
Usr_dat_t	N/A - Global type	Data
Vinkelpointer	N/A - Global type	Data
Vinkelpointertype	N/A - Global type	Data

C.5 Reserverede VAR-navne

Dette afsnit indeholder alle navnene på de variable, der er defineret gennem programmet og dets units. For hver variabel er det anført i hvilken procedure og hvilket unit variablen er defineret. Hvis variablen er globalt defineret er proceduren angivet som "N/A - Global variabel".

Variabelnavn	Findes i procedure/funktion	Findes i unit
A	I_vinkel	Adatool
A	Lubksb	Math
A	Ludcmp	Math
Adc	N/A - Global variabel	Data
Alfa_0	Puls_taeller	Adatool
Alfa_0_rad	Puls_taeller	Adatool
Alfa_1	Puls_taeller	Adatool
Alfa_1	Vinkel_taeller	Adatool
Alfa_1_rad	Puls_taeller	Adatool
Alfa_1_rad	Vinkel_taeller	Adatool
Alfa_s	Beregn_flade_orient	Adatool
Alfa_s	Solpos	Adatool
Alfa_s_deg	Beregn_flade_orient	Adatool
Alfa_s_deg	Screen3	Uddata
Alfa_s_deg	Set_start_pos	Adatool
Alfa_s_deg	Solpos	Adatool
Alfa_s_deg	Solvogn_travers_step	Adatool
Alfa_s_deg	Tidsstep	Adatool
Alpha	Mnewt	Math
Alpha	Usrfun	Math
Al_s	Mnewt	Math
Al_s	Usrfun	Math
Antal_step	N/A - Global variabel	Data
Arccos_t	Arccos	Math
Az	Beregn_flade_orient	Adatool

Azimut_deg	Beregn_flade_orient	Adatool
Azimut_deg	Screen3	Uddata
Azimut_deg	Set_start_pos	Adatool
Azimut_deg	Solpos	Adatool
Azimut_deg	Solvogn_travers_step	Adatool
Azimut_deg	Tidsstep	Adatool
Azimut_deg	I_vinkel	Adatool
B	Lubksb	Math
B	Solpos	Adatool
B	Tjek_fil	Uddata
B_slashpos	Bruger_inddata	Inddata
Batchnummer	N/A - Global variabel	Data
Batchparam	N/A - Global variabel	Data
Bemaerkning	Mnewt	Math
Beta	Usrfun	Math
Beta	Ludcmp	Math
Big	Solvogn_travers_kont	Adatool
Bund	Solvogn_travers_step	Adatool
Bund	N/A - Global variabel	Data
BundBP	Solpos	Adatool
C1	Solpos	Adatool
C2	Solpos	Adatool
C3	Esc_maaling	Adatool
Ch	Tjek_esc	Inddata
Ch	Getadcchannel	Hpibtool
Chan	Setdio	Hpibtool
Channel	Bruger_inddata	Inddata
Code	Getadcchannel	Hpibtool
Command	N/A - Global variabel	Hpibtool
Command	Setdio	Hpibtool
Command	Mnewt	Math
D	I_vinkel	Adatool
D	Ludcmp	Math
D	Bruger_inddata	Inddata
D_a	Beregn_flade_orient	Adatool
D_azimut_rad	Init_solvogn	Adatool
D_count	Roter_flade	Adatool
D_count	Solvogn_travers_kont	Adatool
D_count	Solvogn_travers_step	Adatool
D_count	Bruger_inddata	Inddata
D_h	Beregn_flade_orient	Adatool
D_haeld_rad	Daynr	Math
Dag	Screen3	Uddata
Day_str	Solvogn_travers_step	Adatool
Dd1	Streng_til_real	Inddata
Decimal	N/A - Global variabel	Data
Deklination	Solpos	Adatool
Deklination	N/A - Global variabel	Data
Deklination_deg	Solpos	Adatool
Deklination_deg	Screen3	Uddata
Delestregnummer		

Delta	Puls_taeller	Adatool
Delta	Vinkel_taeller	Adatool
Delta_azimut	N/A - Global variabel	Data
Delta_azimut	N/A - Global variabel	Data
Delta_haeld	N/A - Global type	Data
Delta_haeld	N/A - Global variabel	Data
Delta_x_ind	Screen3	Uddata
Delta_y_ind	Screen3	Uddata
Dir_navn	Tjek_fil	Uddata
Dirinfo	Tjek_fil	Uddata
Doserror	Tjek_fil	Uddata
Dow1	Solvogn_travers_step	Adatool
Dum	Ludcmp	Math
Dvm_taeller	N/A - Global variabel	Data
Dvm_value	Init_solvogn	Adatool
Dvm_value	Nulstil_flade	Adatool
Dvm_value	Roter_flade	Adatool
Dvm_value	Solvogn_travers_kont	Adatool
Dvm_value	Solvogn_travers_step	Adatool
Dvm_value_1	Screen3_data	Uddata
Dvm_value_1	Solvogn_travers_kont	Adatool
Dvm_value_1	Solvogn_travers_step	Adatool
Dvm_value_2	Screen3_data	Uddata
Dvm_value_2	Solvogn_travers_kont	Adatool
Dvm_value_2	Solvogn_travers_step	Adatool
Dvm_value_3	Screen3_data	Uddata
Dvm_value_3	Solvogn_travers_kont	Adatool
Dvm_value_3	Solvogn_travers_step	Adatool
Err	N/A - Global variabel	Hpibtool
Errf	Mnewt	Math
Errorcode	N/A - Global variabel	Errvar
Errorcode	Tjek_fil	Uddata
Errx	Mnewt	Math
Fejlkode	Screen3	Uddata
Fi	Solpos	Adatool
Fil_navn	Tjek_fil	Uddata
Flade_azimut	N/A - Global variabel	Data
Flade_azimut	Nulstil_flade	Adatoo
Flade_haeld	N/A - Global variabel	Data
Flade_haeld	Nulstil_flade	Adatool
Flade_orient	Beregn_flade_orient	Adatool
Gamma_s_deg	Solpos	Adatool
Ga_s	Mnewt	Math
Ga_s	Usrfun	Math
Get_char	Bruger_inddata	Inddata
Get_real	Bruger_inddata	Inddata
Get_streng	Bruger_inddata	Inddata
Grafikdriver	Screen3	Uddata
Grafiktilstand	Screen3	Uddata
Graphicstate	Beregn_flade_orient	Adatool

Graphicstate	Esc_maaling	Adatool
Hraphicstate	Mnewt	Math
H	I_vinkel	Adatool
H_flade_bred	N/A - Global variabel	Data
H1	I_vinkel	Adatool
H2	I_vinkel	Adatool
H3	I_vinkel	Adatool
H4	I_vinkel	Adatool
H5	I_vinkel	Adatool
Hpcount	N/A - Global variabel	Data
Hpoldcount	N/A - Global variabel	Data
Hun	N/A - Global variabel	Data
Hun1	Solvogn_travers_step	Adatool
I	Initadatool	Hpibtool
I	Ludcmp	Math
I	Mnewt	Math
I	Lubksb	Math
I	Lubksb	Math
Ii	Setdio	Hpibtool
Io	Ludcmp	Math
Imax	I_vinkel	Adatool
Indfaldsvinkel	N/A - Global variabel	Data
Indfaldsvinkel	Bruger_inddata	Inddata
Indlaes_tast	Bruger_inddata	Inddata
Indsvingtid	Lubksb	Math
Indx	Ludcmp	Math
Indx	Mnewt	Math
Indx	Lubksb	Math
Ip	Lubksb	Math
J	Ludcmp	Math
J	Ludcmp	Math
K	Mnewt	Math
K	Solpos	Adatool
K	Mnewt	Math
K1	Usrfun	Math
K1	Mnewt	Math
K2	Usrfun	Math
K2	Bruger_inddata	Inddata
L_a	Bruger_inddata	Inddata
L_h	Puls_taeller	Adatool
Length_bc	Vinkel_taeller	Adatool
Length_bc	Puls_taeller	Adatool
Length_cd_0	Roter_flade	Adatool
Length_cd_0	Vinkel_taeller	Adatool
Length_cd_0	Puls_taeller	Adatool
Length_cd_01	Vinkel_taeller	Adatool
Length_cd_01	Puls_taeller	Adatool
Length_cd_1	Vinkel_taeller	Adatool
Length_cd_1	N/A - Global variabel	Data
Locked_azimut	N/A - Global variabel	Data
Locked_azimut		

Locked_haeld	N/A - Global variabel	Data
Locked_haeld	N/A - Global variabel	Data
M_nr	Daynr	Math
M_nummer1	Esc_maaling	Adatool
M_nummer2	Esc_maaling	Adatool
Mainpointerstatus	N/A - Global variabel	Hovedprogram
Materiale	N/A - Global variabel	Data
Max_graense	Streng_til_real	Inddata
Measure_type	N/A - Global variabel	Data
Mellem_count	Vent_tid_diff	Uddata
Mellemrum	N/A - Global variabel	Data
Mereinput	Bruger_inddata	Inddata
Min	N/A - Global variabel	Data
Min_graense	Streng_til_real	Inddata
Min_solvogn_step	Bruger_inddata	Inddata
Min_str	Screen3_data	Uddata
Min1	Solvogn_travers_step	Adatool
Mm1	Solvogn_travers_step	Adatool
Mon_str	Screen3	Uddata
Motorstart	Roter_flade	Adatool
N	Lubksb	Math
N	Ludcmp	Math
N	Mnewt	Math
N	Usrfun	Math
Naeste	N/A - Global variabel	Data
Naeste	Solvogn_travers_kont	Adatool
Naeste	Solvogn_travers_step	Adatool
Navn	N/A - Global variabel	Data
New_hpcount	Init_solvogn	Adatool
New_hpcount	Roter_flade	Adatool
New_hpcount	Solvogn_travers_kont	Adatool
New_hpcount	Solvogn_travers_step	Adatool
New_hpcount	Vinkel_taeller	Adatool
Nr	Getadcchannel	Hpibtool
Nr	Setdio	Hpibtool
Ntrial	Mnewt	Math
Nuv_solvogn_position	Screen3_data	Uddata
Nuv_solvogn_position	Solvogn_travers_kont	Adatool
Nuv_solvogn_position	Solvogn_travers_step	Adatool
Nytid	Vent_tid_diff	Uddata
Ny_flade_azimut	N/A - Global variabel	Data
Ny_flade_haeld	N/A - Global variabel	Data
Ok_svar	Bruger_inddata	Inddata
Old_hpcount	Init_solvogn	Adatool
Old_hpcount	Roter_flade	Adatool
Old_hpcount	Solvogn_travers_kont	Adatool
Old_hpcount	Solvogn_travers_step	Adatool
Old_hpcount	Vinkel_taeller	Adatool
Old_tid_str	Vent_tid_diff	Uddata
Oldtid	Screen3_data	Uddata

Oldtid	Solvogn_travers_kont	Adatool
Oldtid	Solvogn_travers_step	Adatool
Out_fil	N/A - Global variabel	Data
Out_navn	Solvogn_outputfil	Adatool
Outputkolonne	Streng_til_real	Inddata
Outtxt	Screen3	Uddata
Outtxt	Solvogn_travers_step	Adatool
Pause	Solvogn_travers_kont	Adatool
Pause	Solvogn_travers_step	Adatool
Phi	Puls_taeller	Adatool
Phi	Vinkel_taeller	Adatool
Placering	Solvogn_travers_kont	Adatool
Placering	Solvogn_travers_step	Adatool
Pointerstatus	Solvogn_travers_kont	Adatool
Pointerstatus	Solvogn_travers_step	Adatool
Relae_count	Esc_maaling	Adatool
Relae_taeller	Mnewt	Math
Relae_taeller	Stop_maaling	Math
Retning	Vinkel_taeller	Adatool
Scan_retning	N/A - Global variabel	Data
Scanning	N/A - Global variabel	Data
Sek	N/A - Global variabel	Data
Sek_str	Screen3_data	Uddata
Sek1	Solvogn_travers_step	Adatool
Sol_1	Solvogn_travers_kont	Adatool
Sol_1	Solvogn_travers_step	Adatool
Sol_2	Solvogn_travers_kont	Adatool
Sol_2	Solvogn_travers_step	Adatool
Sol_3	Solvogn_travers_kont	Adatool
Sol_3	Solvogn_travers_step	Adatool
Solarimeter_indsving	N/A - Global variabel	Data
Solvogn_maal_type	N/A - Global variabel	Data
Solvogn_position	N/A - Global variabel	Data
Solvogn_step	N/A - Global variabel	Data
Solvognstid	Bruger_inddata	Inddata
Solvstep	Solvogn_travers_kont	Adatool
Solvstep	Solvogn_travers_step	Adatool
Sp	Solvogn_travers_step	Adatool
Starttid	N/A - Global variabel	Hovedprogram
Stoptid	N/A - Global variabel	Hovedprogram
Stoptid	Vent_tid_diff	Uddata
Sum	Lubksb	Math
Sum	Ludcmp	Math
T	I_vinkel	Adatool
T_j	Solpos	Adatool
T_s	Solpos	Adatool
T_z	Solpos	Adatool
Taeller	Tjek_fil	Uddata
Teta_0	Puls_taeller	Adatool
Teta_1	Puls_taeller	Adatool

Teta_1	Vinkel_taeller	Adatool
Teta_z	Solpos	Adatool
Tid_diff	Vent_tid_diff	Uddata
Tid_str	Vent_tid_diff	Uddata
Tim_str	Screen3_data	Uddata
Tim1	Solvogn_travers_step	Adatool
Time	N/A - Global variabel	Data
Timevinkel	N/A - Global variabel	Data
Timevinkel	Solpos	Adatool
Tolx	Mnewt	Math
Toly	Mnewt	Math
Top	Solvogn_travers_kont	Adatool
Top	Solvogn_travers_step	Adatool
TopBP	N/A - Global variabel	Data
Tracking_type	N/A - Global variabel	Data
Tracktid	N/A - Global variabel	Data
Tracktid_min	Bruger_inddata	Inddata
U	Solpos	Adatool
U_deg	Solpos	Adatool
Used_delay	Solvogn_travers_step	Adatool
Usr_data	N/A - Global variabel	Data
Vandret_puls	Puls_taeller	Adatool
V_flade_bred	N/A - Global variabel	Data
V1_txt	Screen3	Uddata
V2_txt	Screen3	Uddata
Vv	Ludcmp	Math
W	Solpos	Adatool
W_deg	Solpos	Adatool
W_ew	Solpos	Adatool
W_u	Solpos	Adatool
W_u_deg	Solpos	Adatool
X	Arccos	Math
X	Arcsin	Math
X	Degreetorad	Math
X	Mnewt	Math
X	Radtodegree	Math
X	Tan	Math
X	Usrfun	Math
X_plot	Screen3_data	Uddata
X_stregkoor	Screen3	Uddata
X_taeller	Streng_til_real	Inddata
Y_koor	Slet_linje	Uddata
Y_plot	Screen3_data	Uddata
Y_stregkoor	Screen3	Uddata
Yy_str	Screen3	Uddata
Yy1	Solvogn_travers_step	Adatool

C.6 Reserverede CONST-navne

I dette afsnit er alle navne på konstanter gennem programmet defineret. Konstnternes beliggenhed i programmet kan ses af procedure- eller funktionsnavn og navnet på det unit konstanten er defineret i. Hvis "N/A - Global konstant"er anført ved procedure/funktionsnavn er konstanten defineret globalt, hvorved alle peocedurer/funktioner har adgang til denne variabel.

Const navn	Findes i procedure/funktion	Findes i unit
Best_sol_1	N/A - Global konstant	Data
Best_sol_2	N/A - Global konstant	Data
Best_sol_3	N/A - Global konstant	Data
Cntadd	N/A - Global konstant	Data
Dacadd	N/A - Global konstant	Data
Dioadd	N/A - Global konstant	Data
Dvmadd	N/A - Global konstant	Data
Graddiff_azimut	N/A - Global konstant	Data
Graddiff_haeld	N/A - Global konstant	Data
H_arm_bred	N/A - Global konstant	Data
Isc	Solpos	Adatool
L_meridian	N/A - Global konstant	Data
Latt	Puls_taeller	Adatool
Length_ab	Vinkel_taeller	Adatool
Length_ab	Puls_taeller	Adatool
Length_bc	Vinkel_taeller	Adatool
Length_bc	Puls_taeller	Adatool
Length_bd	Vinkel_taeller	Adatool
Length_bd	N/A - Global konstant	Data
Long	N/A - Global konstant	Data
M1_puls_length	N/A - Global konstant	Data
M2_puls_length	N/A - Global konstant	Data
M3_puls_length	N/A - Global konstant	Data
Max_azimut	N/A - Global konstant	Data
Max_dvm_taeller	N/A - Global konstant	Data
Max_haeldning	N/A - Global konstant	Data
Maxsolstraal	N/A - Global konstant	Data
Min_azimut	N/A - Global konstant	Data
Min_haeldning	N/A - Global konstant	Data
Motor_pause	N/A - Global konstant	Data
Motortid	N/A - Global konstant	Data
Noerr	N/A - Global konstant	Math
Np	N/A - Global konstant	Data
R1_M1	N/A - Global konstant	Data
R2_M1	N/A - Global konstant	Data
R3_M2	N/A - Global konstant	Data
R4_M2	N/A - Global konstant	Data
R5_M3	N/A - Global konstant	Data
R6_M3	N/A - Global konstant	Data
Sol_1	N/A - Global konstant	Data
Sol_2	N/A - Global konstant	Data
Sol_3		

Solvogn_hast	N/A - Global konstant	Data
Stop_neg_1	N/A - Global konstant	Data
Stop_neg_2	N/A - Global konstant	Data
Stop_neg_3	N/A - Global konstant	Data
Stop_pos_1	N/A - Global konstant	Data
Stop_pos_2	N/A - Global konstant	Data
Stop_pos_3	N/A - Global konstant	Data
Tiny	Ludcmp	Math
Ugedag	N/A - Global konstant	Data
V_arm_bred	N/A - Global konstant	Data