CODEN: LUTFD2/(TFRT-5403)/1–123/(1989)

Grafiskt presentationsprogram för processimulator

Ralph Myrnäs

Institutionen for Reglerteknik
Lunds Tekniska Högskola
Juni 1989

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
June 1989

Document Number
CODEN: LUTFD2/(TFRT-5403)/1–123/(1989)

Author(s)
Ralph Myrnäs

Supervisor
Rolf Johansson

Sponsoring organisation

Title and subtitle
Grafiskt presentationsprogram för processimulator
(Graphical interface program for a process simulator)

Abstract

A process simulator has been in operation at the Barsebäck nuclear plant since 1984. The simulator is based on Simnon and a plant model developed by RISÖ. This report describes a graphical presentation program that makes the simulator easier to use.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
Swedish

Number of pages
123

Recipient's notes

Security classification

Innehåll

1. Inledning 2

2. Metoden 3
2.1 Ursprungligt system 3
2.2 Arbetet 4

3. Resultat 6
3.1 Inledning 6
3.2 Uppkoppling 6
3.3 Fluxpanelen 9
3.4 Vatten- och Ångpanelen 11
3.5 Styrstavspanelen 13
3.6 Diagram 14

4. Kommentarer 22
4.1 Hjälpmacron i Simnon 22

5. Referenser 24

A. Grafikprogrammering 25
A.1 Kodning 25
A.2 Grafiken 28
A.3 Inläsning av kommando 30
A.4 Segmentklasser 31
A.5 Uppdatering 33
A.6 Börvärden 34
A.7 AUTO och SIRM 34
A.8 Öppna och stänga 34

K. LITET SYNTAX REGISTER 36

L. Listningar 37
L.1 BSREAL.FOR 37
L.2 LPGRAPH.FOR 86
L.3 WATER.FOR 98
L.4 CROD.FOR 107
L.5 NEWUPP.T 113
L.6 NYLAND.T 116
L.7 LCONS.T 118

1. Inledning

Vid Barsebäcksverket installerades hösten 1984 en process simulator, som ursprungligen är utvecklad vid RISÖ. Denna simulator har haft en begränsad användning, eftersom körningen av programmet inneburit långa väntetider. Resultatet har i efterhand presenterats i diagramform, och man har inte kunnat påverka förloppet efter det att simuleringen startat.

Det har varit önskvärt att ta fram en modell med mer liv i ,en modell där resultaten presenteras kontinuerligt och där man kan gå in och påverka förloppet. Exempelvis ska pumpar kunna startas och stoppas och ventiler ska kunna öppnas och stängas.

Jag har eftersträvat att efterlikna verkligheten i så stor utsträckning som möjligt. Jag har sneglat på kontrollerna i manöverrummet och efterliknat vissa av dessa.

En viss kunskap om programmering i FORTRAN 77 är nödvändig för att kunna ta del av denna rapport.

2. Metoden

2.1 Ursprungligt system

Hårdvaran består av en VAX11-750 minidator med diskar och Tektronix T4107 terminaler se fig 2.1.

Mjukvaran är uppbyggd av ett antal block,se fig 2.2, skrivna i Fortran. Varje block är uppdelat i två bitar. En bit där själva beräkningarna utförs (t.ex. FWCON) och en bit som kopplar ihop SIMNON med fortranprogrammen (t.ex. FC).

Om vi tittar närmare på blocken, så är alla uppbyggda på samma sätt, se listning av t.ex FC i Appendix L. Man hittar bland annat följande bitar.

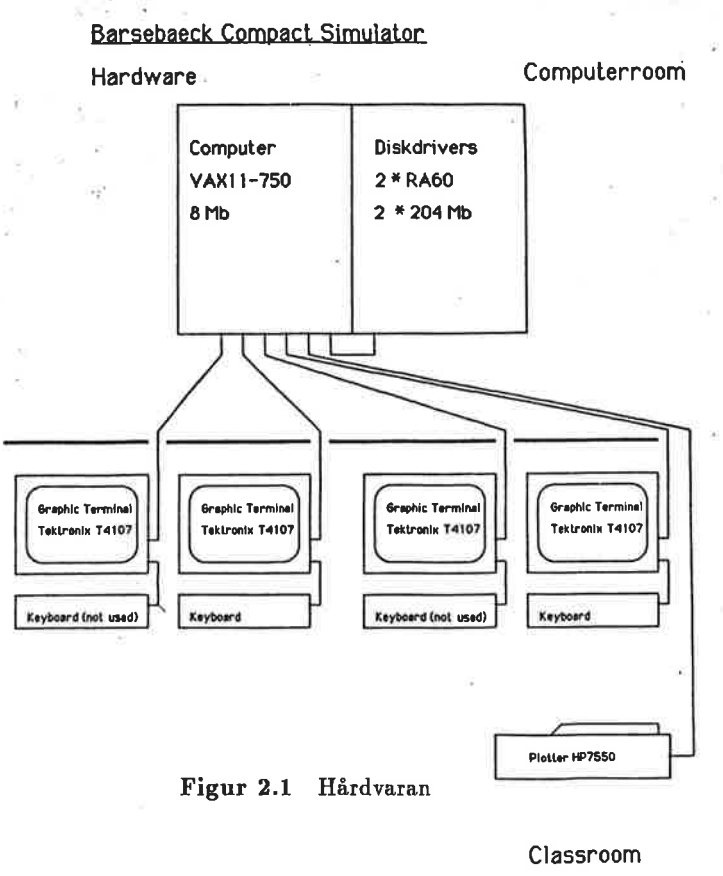
INITIAL SECTION

OUTPUT SECTION

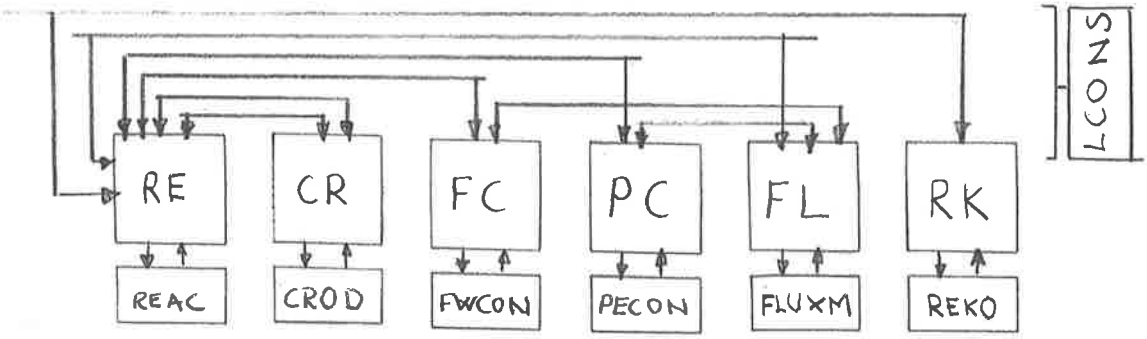
DYNAMIC SECTION

Fortranrutinerna knyts ihop med ett SIMNON program (*connection routine*) LCONS.

Modellen finns beskriven i "DESCRIPTION OF THE BARSEBÄCK LOW POWER PLANT MODEL FOR THE COMPACT SIMULATOR; PART 1& 2", P. la Cour Christensen.



Figur 2.1 Hårdvaran



Figur 2.2 Schematisk bild av simulatorn.

2.2 Arbetet

Uppgiften har gått ut på att plocka ut variabler ur programmet , kunna påverka dem och presentera dem på skärmen. Detaljer kring programmeringen finns beskrivna i appendix A.

Variablerna ligger i vektorer deklarerade i COMMON-block. I exempelvis FC finns COMMON-blocket.

COMMON / CFWCON / SFC(6),DFC(6),XFC(6),CFC(6)

I modulen FWCON har samma variabler andra mindre kryptiska namn typ P12=XFC(6) som är en indikator på om matarvattenpumparna är i drift.

Med användande av de olika grafiska operationerna som Tektronixterminalerna erbjuder ritades de tre panelerna, där presentation av processen sker. Panelerna och alla detaljer i dessa, definieras i form av *segment*. Dessa segment lagras i terminalens minne, och man kan sedan manipulera segmenten med kommandon från programmet. Panelerna och detaljerna i dessa ritas av subrutiner.

Rutinerna är uppdelade i tre block, ett för varje panel. Alla rutiner som har med fluxpanelen,(*MAIN - PANEL*), att göra ligger i LPGRAPH.FOR. Rutiner som hör ihop med vatten- och ångpanelen ligger i WATER.FOR och till styrstavspanelen ligger i CROD.FOR.

Själva panelerna definieras i :

- PANEL , flux-panelen.
- WATER , vatten- och ångpanelen.
- PANCROD , styrstavspanelen.

Segmenten för val av de olika panelerna och stopp*knappen* ritas i INGRAPH. Desseutom finns en mängd rutiner för att rita detaljer i de olika panelerna t.ex. pumpar(PUMP), ventiler(VENTIL,WENTIL) o.d.. Se listningarna i appendix L.

Flertalet variabler ska endast presenteras i en viss panel. Därför infördes olika grafiska moder. Det finns fem grafiska moder, GRMODE.

GRMODE = 0 ,simuleringen har inte startat.

GRMODE = 1 , fluxpanelen visas på skärmen.

GRMODE = 2 , vatten- och ångpanelen visas.

GRMODE = 3 , styrstavspanelen är visas.

GRMODE = 9 ,simuleringen har stoppats, den inställda tiden har löpt ut.

I programmet ligger villkor för att uppdatera de variabler som är presenterade på skärmen. Uppdateringen kan innebära att en ventil byter färg, om den har stängts eller öppnats. Ett flertal variabler är presenterade i form av siffror eller visare. Dessa variabler uppdateras om de faller utanför ett bestämt intervall, se appendix A för detaljer.

3. Resultat

3.1 Inledning

Detta kapitel är även avsett att kunna användas som manual för användare av Barsebäckverkets "realtids-simulator". Simulatorn är avsedd att användas i utbildningsändamål och för analyser av olika driftsfall.

Presentationen av simuleringen sker i tre menyer *(paneler)*.Du kan ändra parametrar och välja panel genom att peka med *cursorn* på ett fält på skärmen. Programmet läser av *cursorns* position en gång i sekunden.

Färgerna på segmenten har olika betydelse :

GRÖN	Öppen / i drift
RÖD	Stängd / ej i drift
GUL	Kontinuerlig varierbar

Längst upp till höger finns på skärmen ett fält som är gemensamt för alla paneler. Här visas:

- tiden (min:sek)
- effekten i reaktorhärden *(MW)*
- trycket i reaktorhärden*(bar)*
- nivån i reaktortanken *(m)*
- moderatortemperaturen *(^0C)*.

Här finns också indikator för snabbstopp *(S S)* och startförregling *(S)*. Dessa funktioner kan man blockera genom att trycka på respektive *(BLOCK)*-knapp. Man kan även lösa ut snabbstopp manuellt genom att påverka *(S S)*-knappen.

3.2 Uppkoppling

Logga in på SIM1 eller SIM2. Tryck sedan på funktionstangenten som anger start av "realtids"-simulatorn.

Simulatorn använder två skärmar, och det första datorn frågar efter är beteckningen för $SKÄRM_2$. Har du endast tillgång till en terminal, ange dess beteckning. Sedan frågar datorn efter terminaltyp, och eftersom grafiken är utformad för tektronix T4100 terminaler blir svaret T4100. För att starta simuleringen finns ett hjälpmacro, NEWUPP. Detta macro hjälper dig att koppla ihop simmon- och fortranprogrammen och starta simuleringen. NEWUPP innehåller också menyer för val av utgångslägen och styrstavsmanövrer.

Skriv :

>NEWUPP

Nu är det bara att välja mellan de alternativ som de olika menyerna erbjuder.

Huvudmenyn

```
**** BVT PROCESS-SIMULATOR ****

* HUVUDMENY , LAGEFFEKTMODELLEN BSO *

0.  - AVSLUTA

1.  - UTFORA SIMULERING MED NYTT UTGANGSLAGE

2.  - FORTSATTA SENASTE SIMULERINGEN

3.  - VISA SIMULERINGSRESULTAT

*****************************

Ange onskat alternativ: (0-3)
```

Välj utgångsläge enl. tabell 3.1

Styrstavar

Du kan nu välja vilken styrstavsmanöver du vill utföra. På skärm$_1$ får du upp menyn i fig 3.1, och på skärm$_2$ en tabell över aktuellt styrstavsläge. Efter det att du angett önskat alternativ frågar programmet efter nödvändiga parametrar, gruppnr, antal procent. Om du har valt alternativ 2, så ska även antal grupper anges.

Start

Välj i vilken fil du vill lagra resultaten, och under hur lång tid simuleringen ska fortgå. Välj inte för lång tid, det går att fortsätta en avbruten simulering. Om du fortsätter på föregående simulering, så tänk på att programmet pre-

Tabell 3.1 Tabell över möjliga utgångslägen

```
*** BVT PROCESS-SIMULATOR ,LAGEFFEKTMODELLEN BSO ***

      MOJLIGA UTGANGSLAGEN FOR START AV SIMULERING

Nr     Summa SS-monster  Reaktortryck  Reaktoreffekt  Moderatortemp
             %            Bar           MWt            Grad C
-------------------------------------------------------------------
1 :          0            1             5.0E-7         56
2 :          3300         15.6          2.0E-6         200
3 :          3520         15.6          5.0E-6         200
4 :          3735         15.6          2.0E-4         200
5 :          4237         40            1.7            250
6 :          4554         60            1.84           275
7 :          4673         40            17             249
8 :          5505         65            68             277
9 :          5820         70            102            286

Fran vilket utgangslage vill du borja ?
Knappa in onskat utgangslage (1-9)
```

```
0       INGA STYRSTAVSANDRINGAR ONSKAS

1       EN (1) STYRSTAVSGRUPP SKALL MANOVRERAS

2       ETT ANTAL STYRSTAVSGRUPPER SKALL MANOVRERAS

ANGE ONSKAT ALTERNATIV :
```

Figur 3.1 Meny för val av styrstavskörning

senterar tiden i minuter och sekunder, medan du här ska ange stopptid enbart i sekunder.

3.3 Fluxpanelen

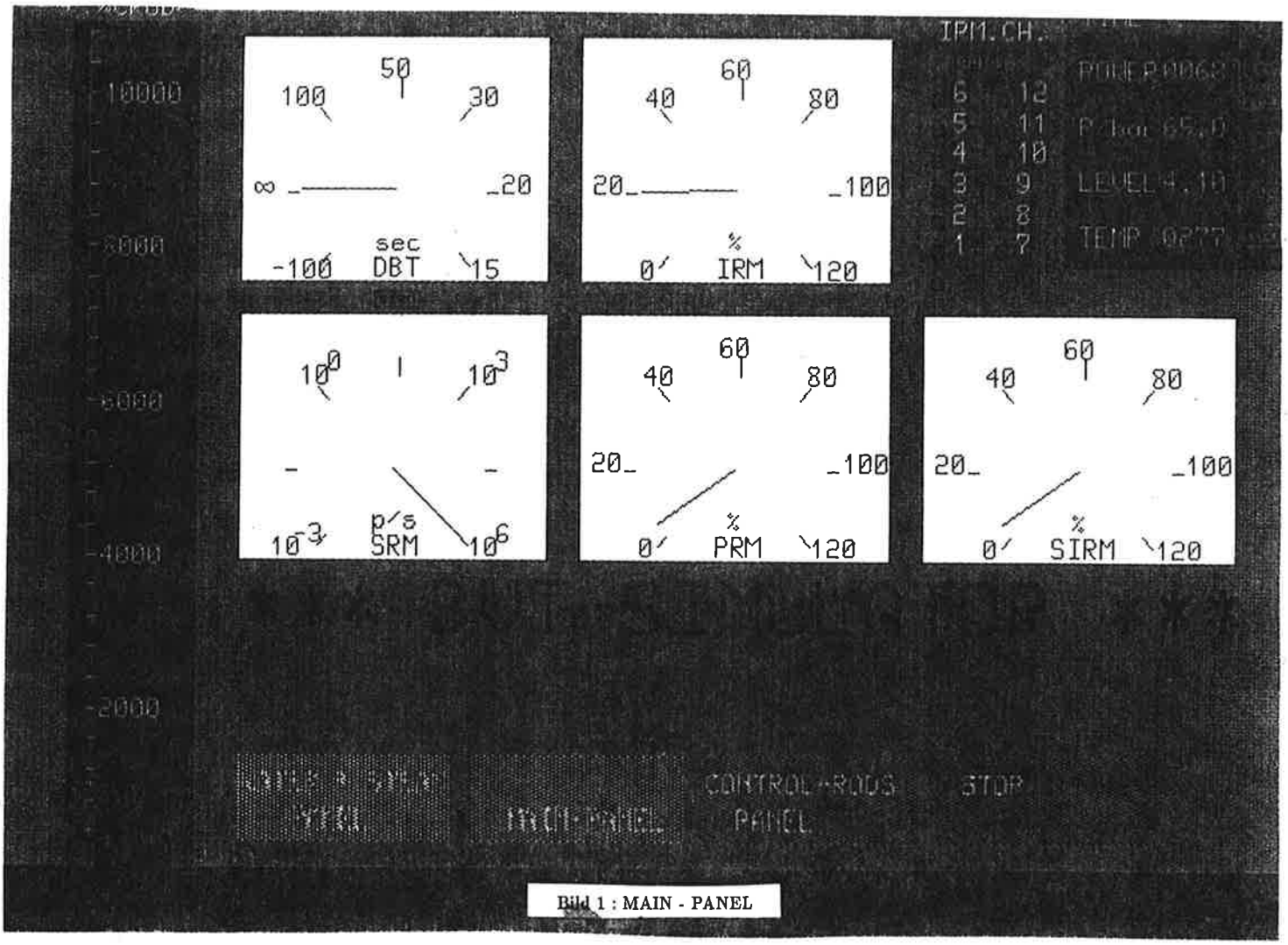


Bild 1 : MAIN - PANEL

Denna panel döpte jag till MAINPANEL, och jag har inte sett att det varit nödvändigt att ändra namnet.

Till vänster finns det totala styrstavsmönstret presenterat både i siffror och som en stapel.

Fördubblingstid (DBT)

Instrument som visar SRM-fördubblingstiden i sekunder.

Neutronflödesinstrument

Fyra instrument visar neutronflödet.

SRM : Visar källeffekten i pulser per sekund. Denna bottnar redan vid måttliga effekter.

IRM : Visar effekten i ett mellanområde (*IRM = intermediate range monitor)*. IRM har 12 olika mätområden *IRM.CH.*. Omkopplingen mellan dessa sker automatiskt, om vissa villkor är uppfyllda. Man kan spärra auto-omkopplingen genom att påverka autoknappen.

PRM : Visar neutroneffekten i procent av full effekt.

SIRM: Genom att påverka SIRM-knappen drar man ut detektorn ur härden. (Förr fanns det separata detektorer för de olika effekt områdena SRM och IRM. Nu har dessa båda en gemensam detektor och signalerna delas upp av elektronik).

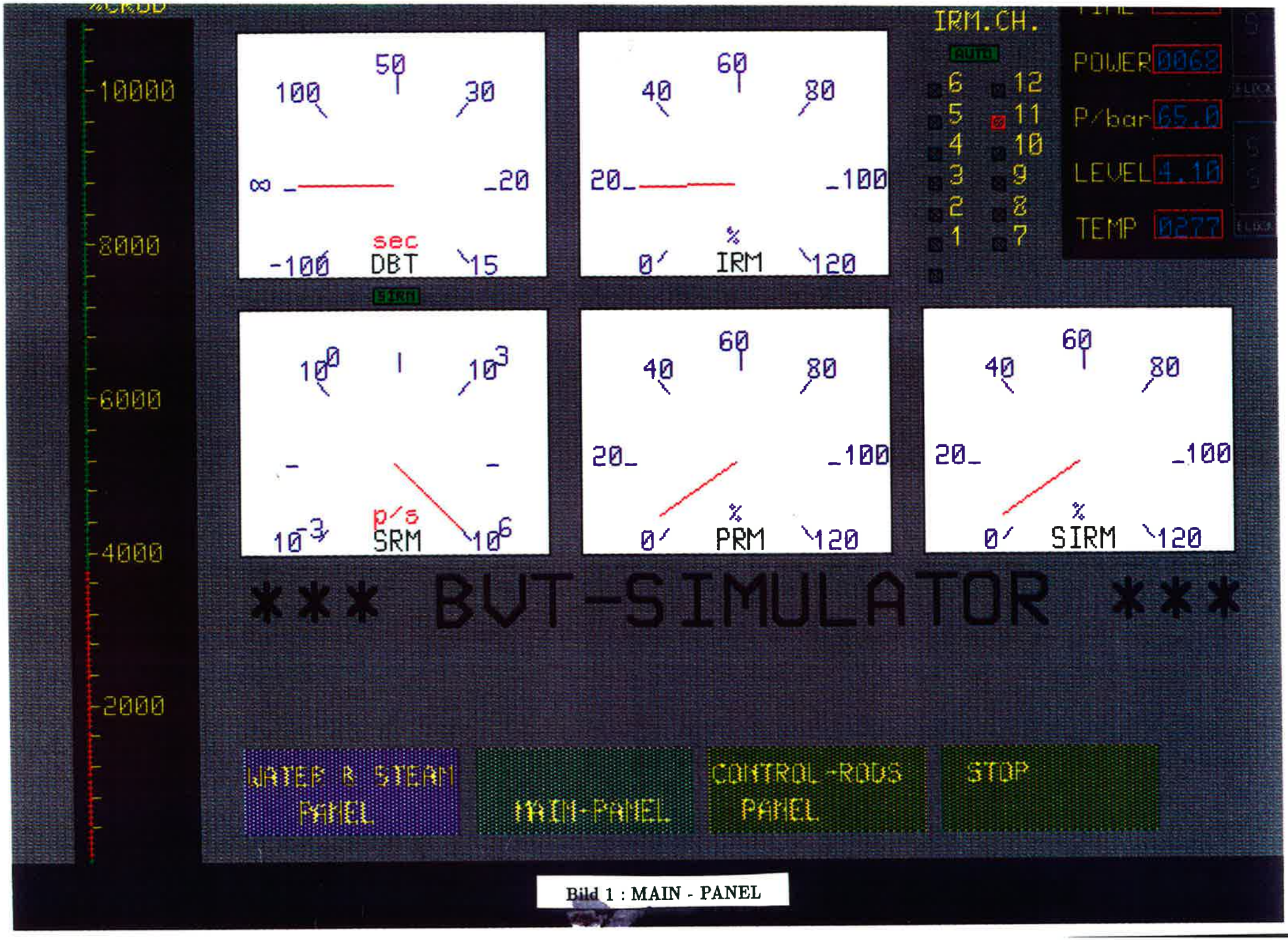
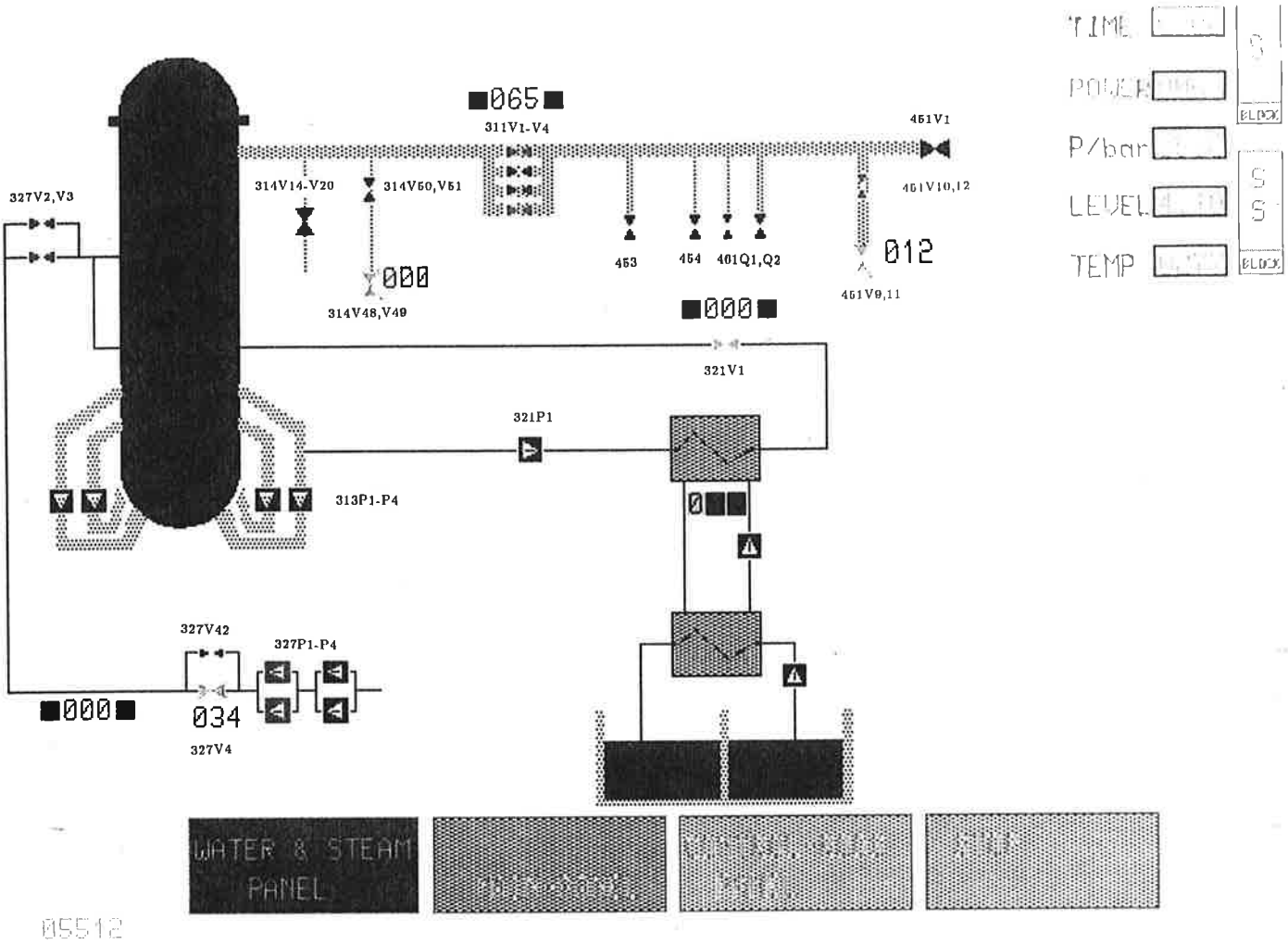


Bild 1 : MAIN - PANEL

3.4 Vatten- och Ångpanelen



Denna panel är en översiktsbild av modellen. Här kan du gå in och ändra på ett flertal parametrar.

SYSTEM 327 - Hjälpmatarvattnet

Pumparna 327P1,P2 stoppas och startas gemensamt.

Detta gäller likaså skalventilerna 327V2,V3.

Siffran under ventilen 327V4 visar ventilläget i procent. Ventilen 327V42 kan ej påverkas av operatören utan öppnas automatiskt, om nivån i tanken blir för låg.

Man kan också ställa in ett börvärde med hjälp av (+/-)-knapparna. Börvärdet ändras med 10 kg/s för varje tryck. Om börvärdet är 0, så sker regleringen automatiskt.

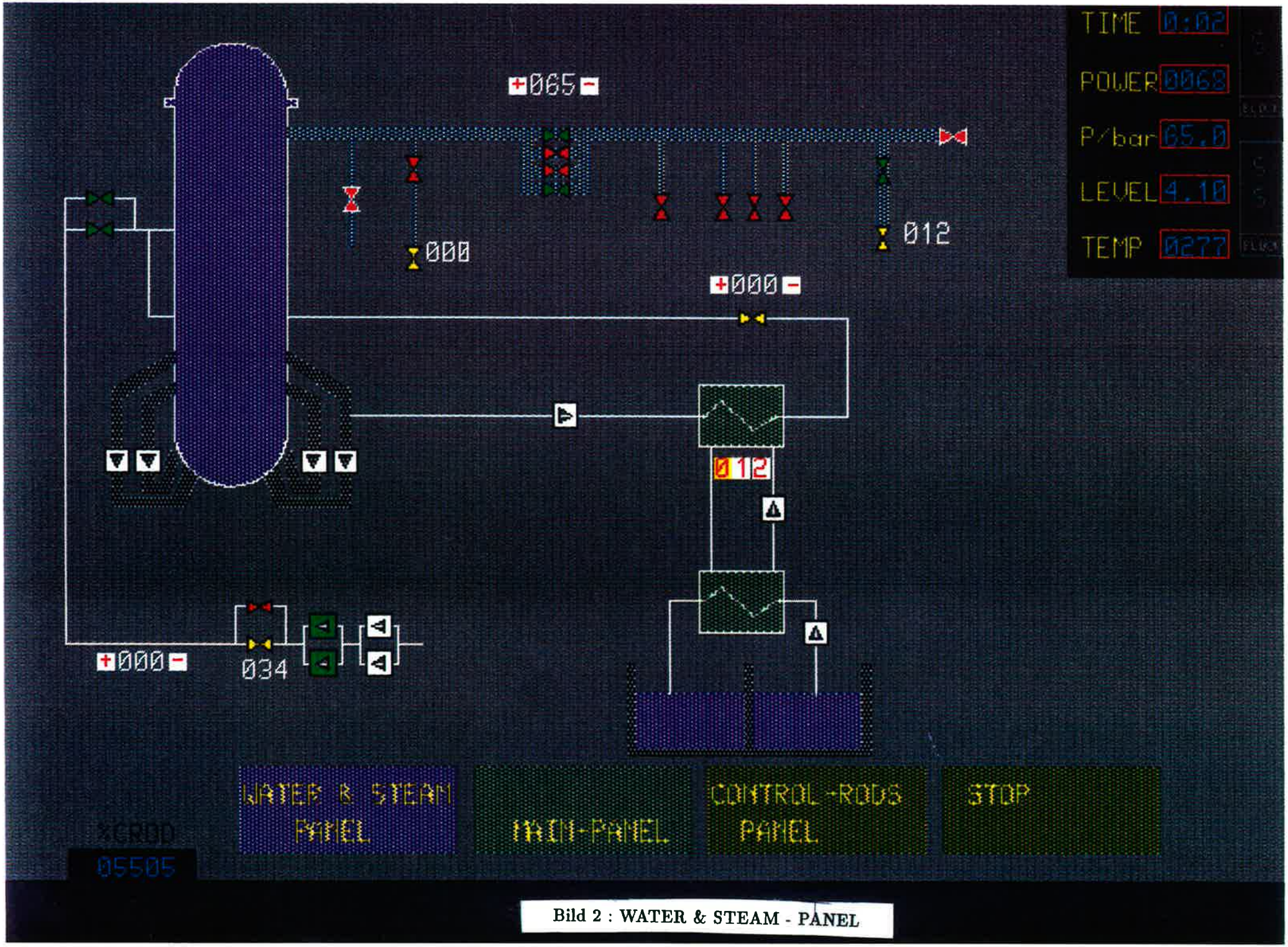


Bild 2 : WATER & STEAM - PANEL

SYSTEM 311 - Huvudångledningarnas ventiler

I den verkliga processen finns fyra ångledningar som i modellen har ersatts med en ledning med fyra ventiler.

De två mittersta är alltid stängda vid de låga effekter som modellen avser. De andra två kan man öppna och stänga.

SYSTEM 451 - Turbin- och dumpventiler

I denna modell är inte turbinen inkopplad och ventil 451V1, som representerar turbinreglerventilen, kan inte påverkas.

Ventil 451V10,V12, som representerar dumpblockerventilerna, är öppen om dumpning till turbinkondensatorn är önskvärd.

Ventil 451V9,V11 representerar dumpreglerventil.Dess läge i procent visas kontinuerligt.

SYSTEM 453,454,461

Systemet består av fyra ventiler. Dessa ventiler motsvarar uttag av ånga till olika processer :

Ventil	Uttag kg/s	Representerar
453	6	Reducerstation
454	1	Spärr- och läckageångsystem
461Q1	1	Driftejektorer, stråk 1
461Q2	1	Driftejektorer, stråk 2

SYSTEM 314, AVBLÅSNING

Detta system representerar avblåsning av ånga till en kondensationsbassäng.

Systemet består av tre ventiler :

314V14-V20. kan ej påverkas manuellt utan öppnas automatiskt om trycket blir för högt.

314V50,V51. öppnas och stängs manuellt.

314V48,V49. ,en reglerventil vars läge visas kontinuerligt.

TRYCKBÖRVÄRDE

Man kan ställa in ett tryckbörvärde med +/- -knapparna ovanför huvudångledningarnas ventiler 311V1-V4. Börvärdet ändras med 1 bar för varje tryck.

SYSTEM 321, RESTEFFEKTKYLNING

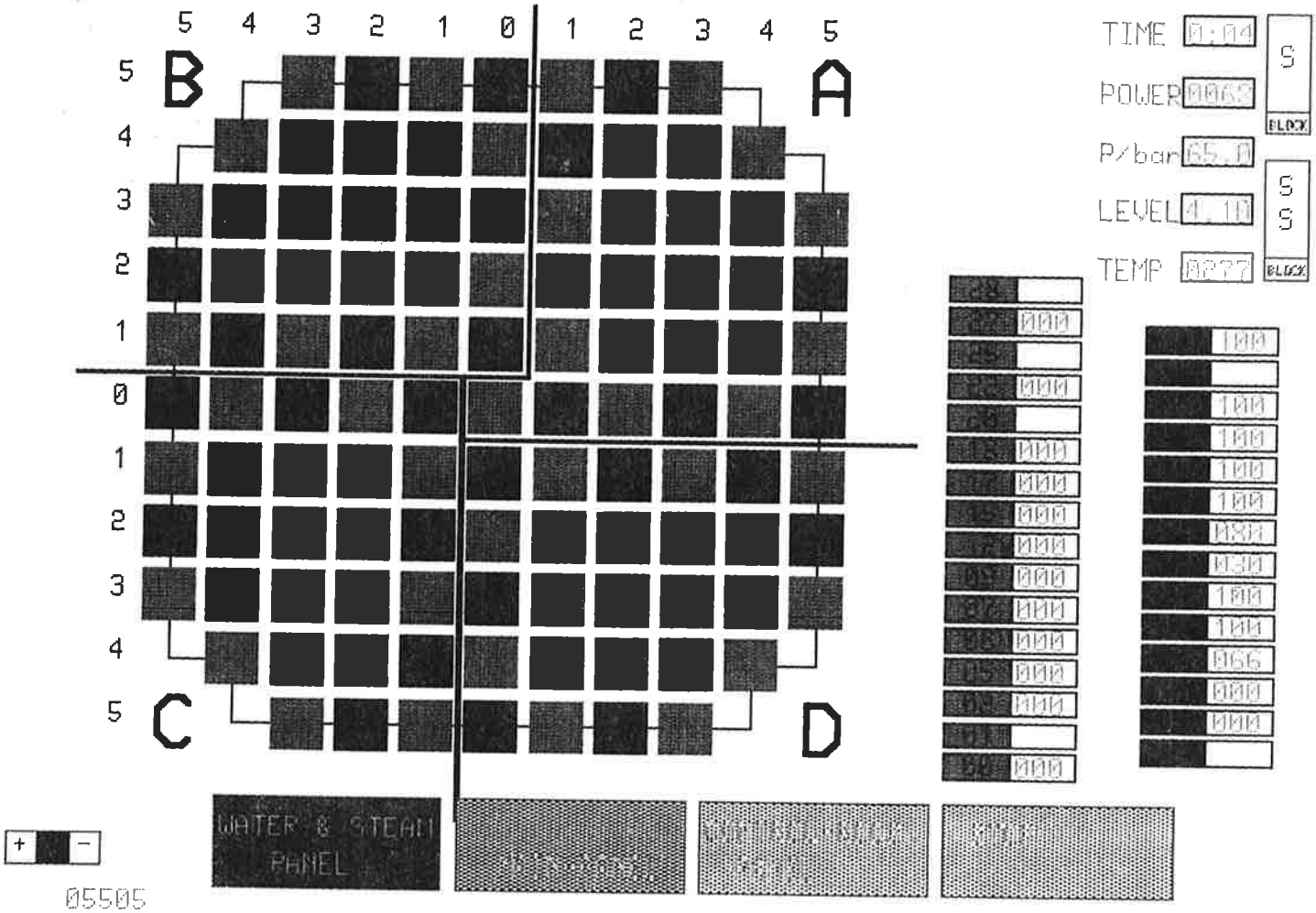
Även när styrstavarna är helt inskjutna återstår en viss effekt. Man använder då detta system för att kyla reaktorn. Systemet består av två kretsar. De innehåller vardera två pumpar, två värmeväxlare och en ventil.

Välj först antal kretsar (0,1 eller 2) och ställ sedan in önskat flöde med (+ / -)-knapparna. Flödet ändras med 10 kg/s för varje knapptryckning.

SYSTEM 313 - Huvudcirkulationspumparna

Dessa pumpar driver runt kylvattnet i reaktorn. De kan inte påverkas manuellt.

3.5 Styrstavspanelen



Panelen visar härden uppifrån. Till höger visas i procent hur mycket varje grupp är utdragen. I verkligheten är styrstavarna indelade i 29 grupper som i sin tur uppdelade i vita och svarta. Resultatet blir det schackbrädemönster, som syns på panelen. Hur grupperna är indelade framgår av fig.3.1.

Vid start av reaktorn skjuts först de vita grupperna in och därefter de svarta enligt ett föreskrivet schema. I modellen finns det endast 24 grupper med 8,4,2 eller 1 stav per grupp. Vissa av rutorna har därför lämnats tomma.

Om man vill se vilka stavar, som hör till en viss grupp, kan man gå in och peka antingen på en stav eller på boxen med gruppnummer. Då börjar alla stavar som hör till den utvalda gruppen att blinka.

Manövrering

* I skärmens nedre vänstra hörn finns tre segment för manövrering av styrstavar. välj först grupp genom att peka på det med *cursorn*. Välj därefter om du vill dra ut styrstavarna (+) eller skjuta in desamma (−). Du kan också stoppa

* Styrstavsmanövreringen fungerar f.n. inte på ett tillfredsställande sätt. Man kan däremot alltid stoppa en styrstavsmanöver, och även starta en grupp om ingen annan styrstavsmanöver pågår.

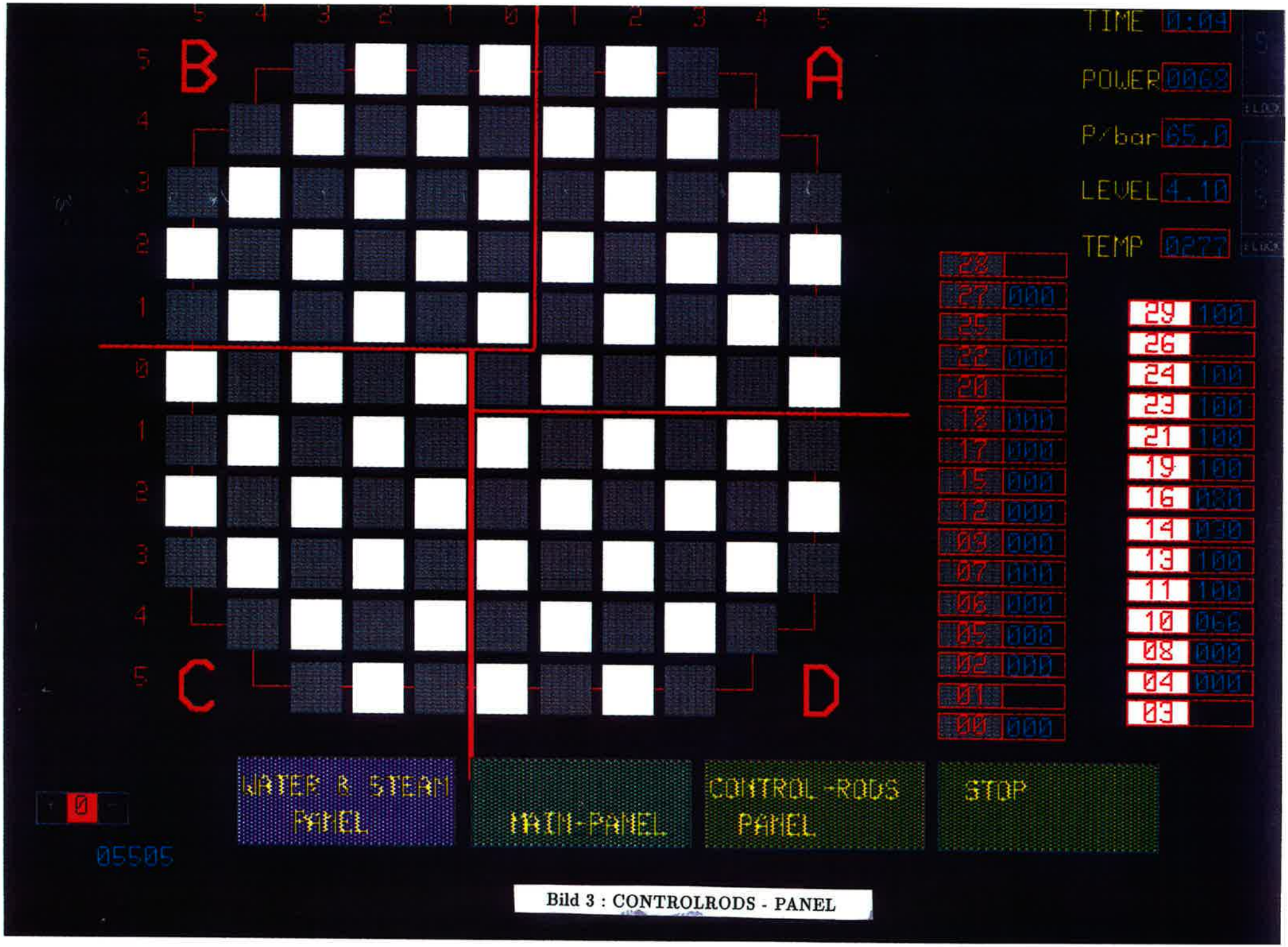


Bild 3 : CONTROLRODS - PANEL

stavarna med (0).

3.6 Diagram

I det ursprungliga programmet presenterades händelseförloppet genom uppritning av diagram efter simuleringen. Denna möjlighet finns kvar. När tiden har löpt ut får man åter upp huvudmenyn. Man kan då välja alternativ 3. På nästa sida finns ett register över de variabler, som man kan välja mellan. Diagrammen på de följande sidorna är ett exempel, där nedanstående operationer utförts:

t = 4	Ventilerna 327V2,V3 stängs.
t = 5,6	Snabbstoppsutlösningen blockeras.
t = 99	Dumpblockerventilerna 451V10,V12 stängs.
t = 168	Ventilerna 327V2,V3 öppnas.
t = 169	Avblåsningsventilerna 314V50,V51 öppnas.
t = 170	Ventil 453 öppnas.
t = 173	Ventil 454 öppnas.
t = 174	Ventil 461Q1 öppnas.
t = 175	Ventil 462Q2 öppnas.

I diagrammen kan man bl.a. iaktta följande.

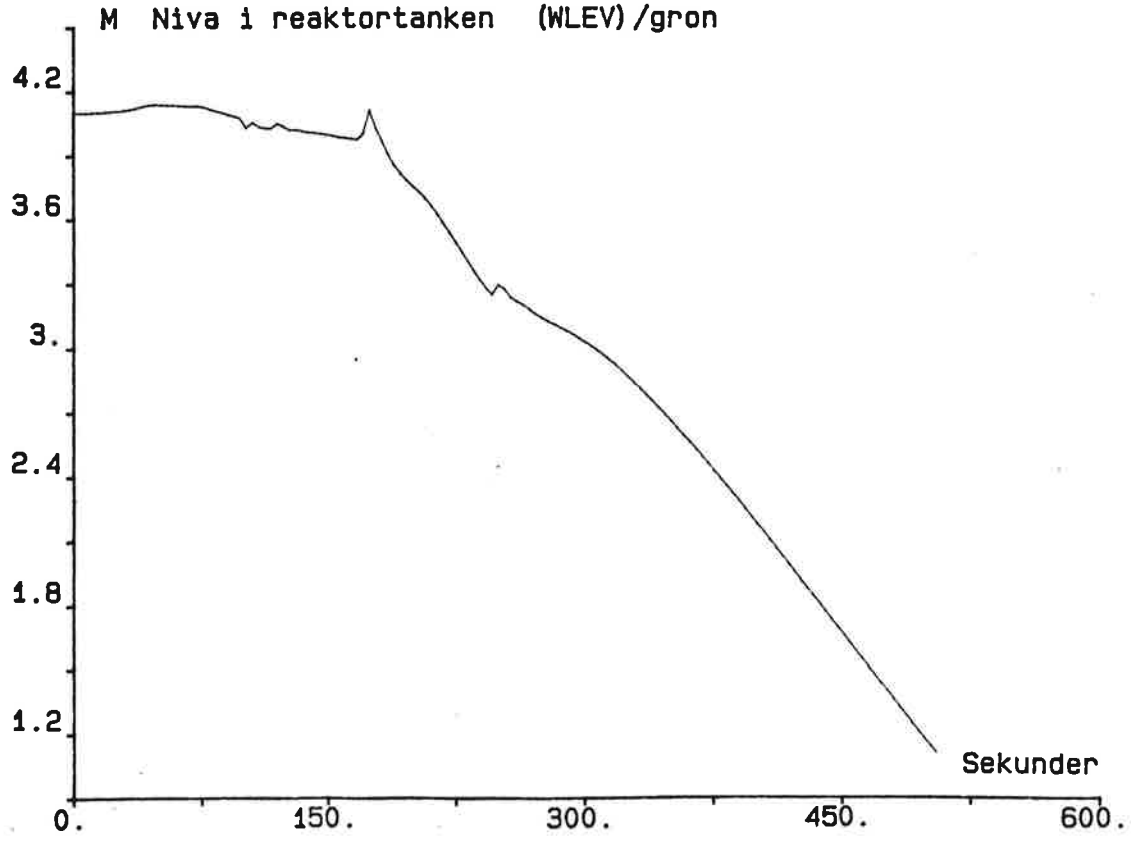
Eftersom matarvattnet stoppas helt sjunker nivån i tanken. Detta kompenseras till att börja med genom att voidhalten (förhållandet gas/vätska i reaktorvattnet) ökar. När dumpventilen stängs ökar trycket, eftersom ångan inte har någonstans att ta vägen. När man sedan släpper ut all ånga genom att öppna alla ventiler, börjar det koka kraftigt i reaktorn då trycket sjunker.

Bvt ~~Kompakt~~Process-simulator, Lågeffektmodellen, Variabellista

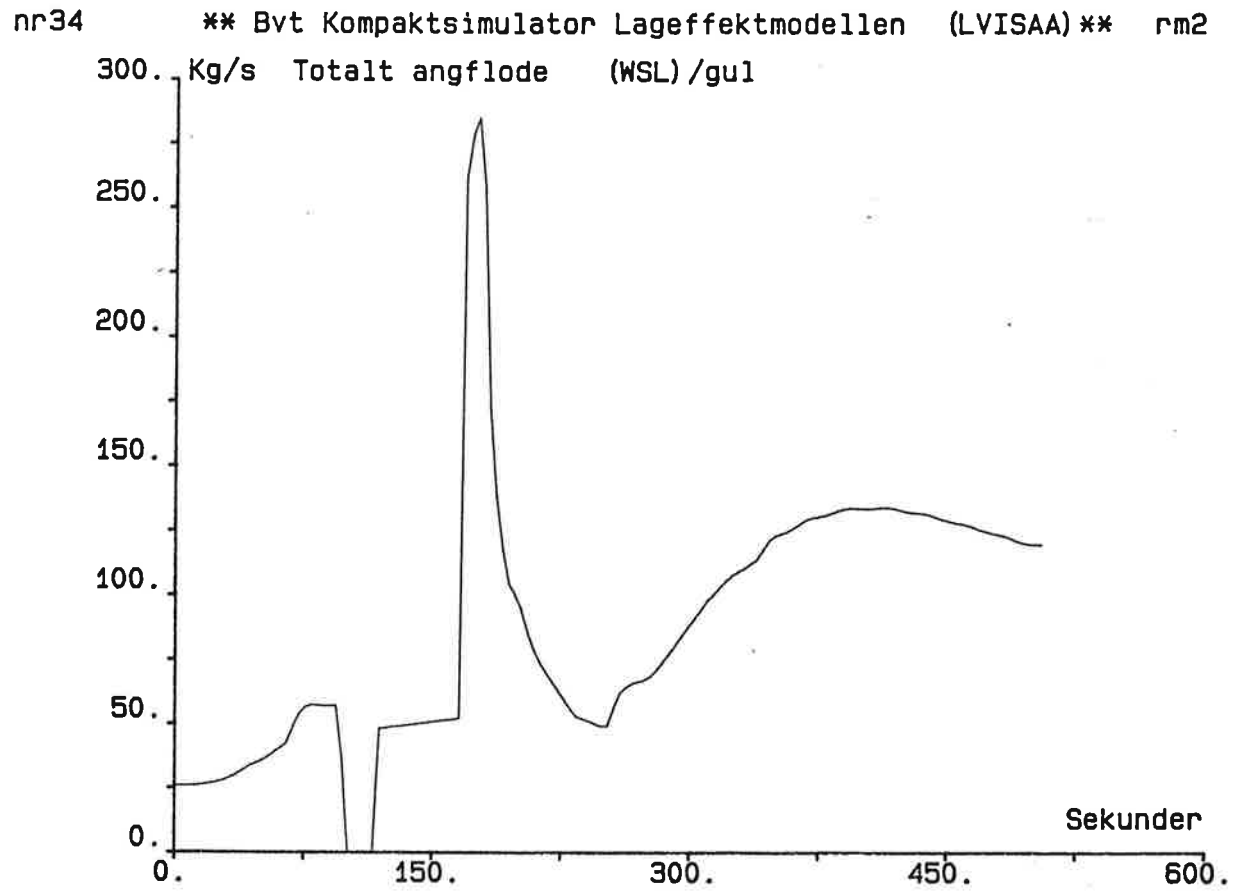
1. Temp. i "lower plenum 1"
2. Temp. i "lower plenum 2"
3. Temp. i "core bypass 1"
4. Temp. i "core bypass 2"
5. Temp. i ångseparator
6. Temp. i reaktorns övre vattenvolym
7. Temp. i "feedwater chamber"
8. Temp. fallspalten
9. Temp. i HC-kretsar
10. Tryck i ångseparatorer
11. Voidhalt i ångseparatorer
12. Totalt HC-flöde
13. Reaktivitet, filtrerat värde
14. Vattenflöde till "Top volume"
15. Reaktoreffekt
16. Medelvoidhalt
17. Ångflöde till reaktorns "ångdom"
18. Ångflöde i ångseparatorer
19. Vattenflöde i ångseparatorer
20. Massflöde i ångseparatorer
21. Medelneutronflöde, kalibr.
22. Reaktivitet utan neutronkallflöde
23. Moderatortemp. medel
24. Dränageflöde till system 352
25. Styrstavsläge Grupp 10
26. Styrstavsläge Grupp 11
27. Styrstavsläge Grupp 12
28. Summa styrstavsmönster
29. Summa ångflöde
30. Rel Ventilläge 327V4
31. Hjälpmatarvattenflöde
32. Ånguttag till processen
33. Reakt.eff.(utom resteff.)
34. Totalt ångflöde
35. Rel Ventilläge 314V48,V49
36. Rel Ventilläge 451V9, V11
37. SRM-effekt, logaritmisk
38. Valt IRM-område
39. % IRM-effekt
40. P/S SRM-effekt
41. Indikering för S-kedjan
42. Indikering för E-kedjan
43. Indikering för SS-kedjan
44. Bar Reaktortryck
45. Nivå i reaktortanken
46. Fördubbl.tid SRM,reciprok.
47. % PRM-effekt, filtrerad
48. Grad C 321-temp.eft.E1,E2
49. Kg/s 321-flöde genom E1/2
50. Grad C 321-temp.före E1/2

89.05.08 - 15: 14: 50 nr: 9
hcopy hp 1.2

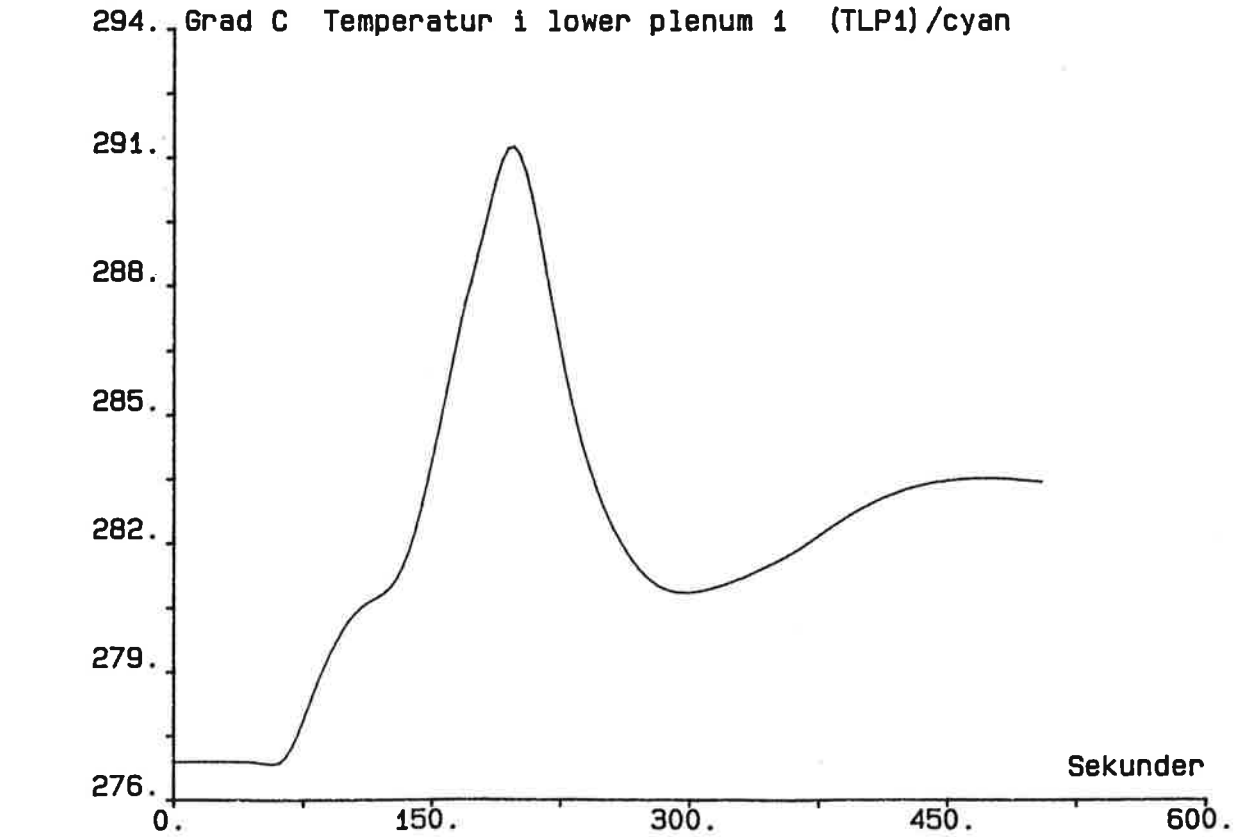
nr45 ** Bvt Kompaktsimulator Lageffektmodellen (LVISAA) ** rm2



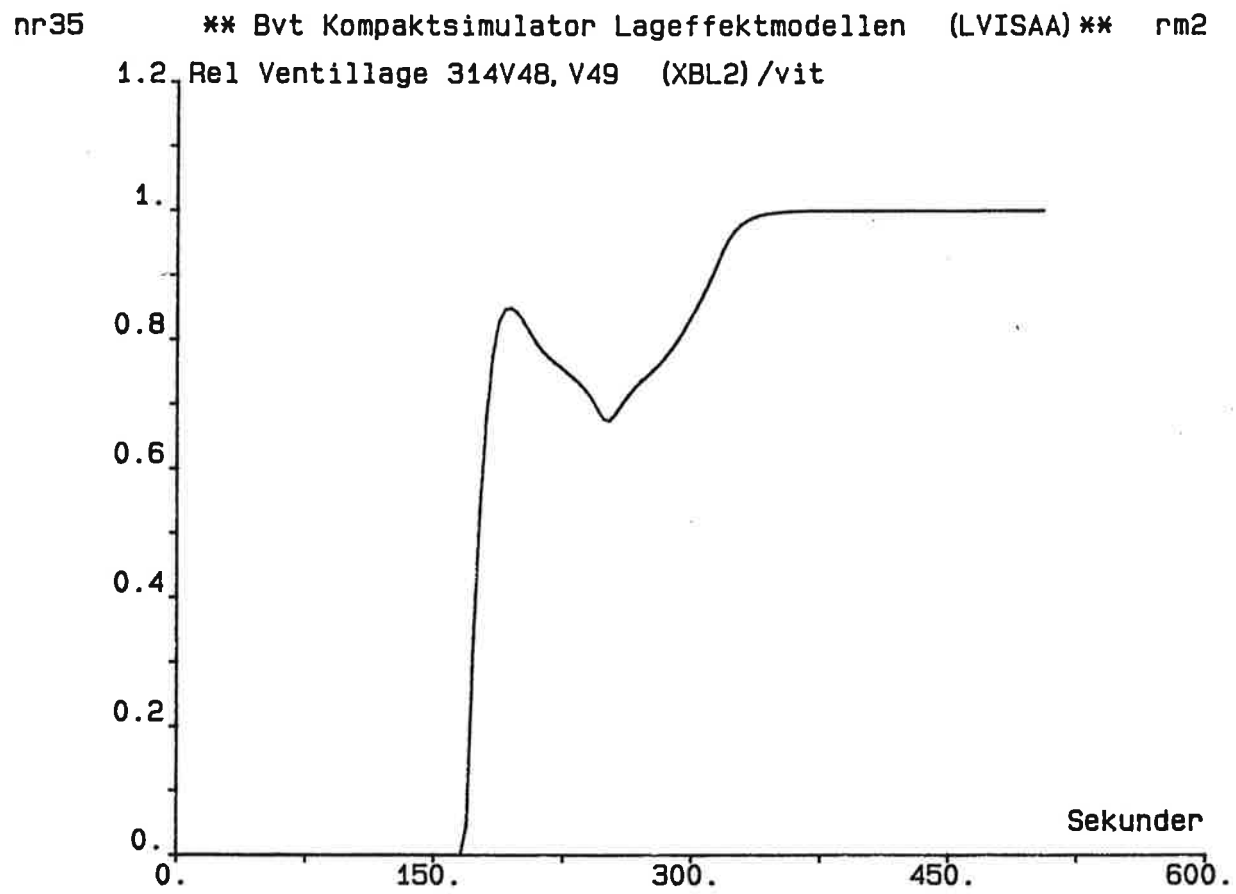
89.05.08 - 15:03:56 nr: 3
hcopy hp 1.2



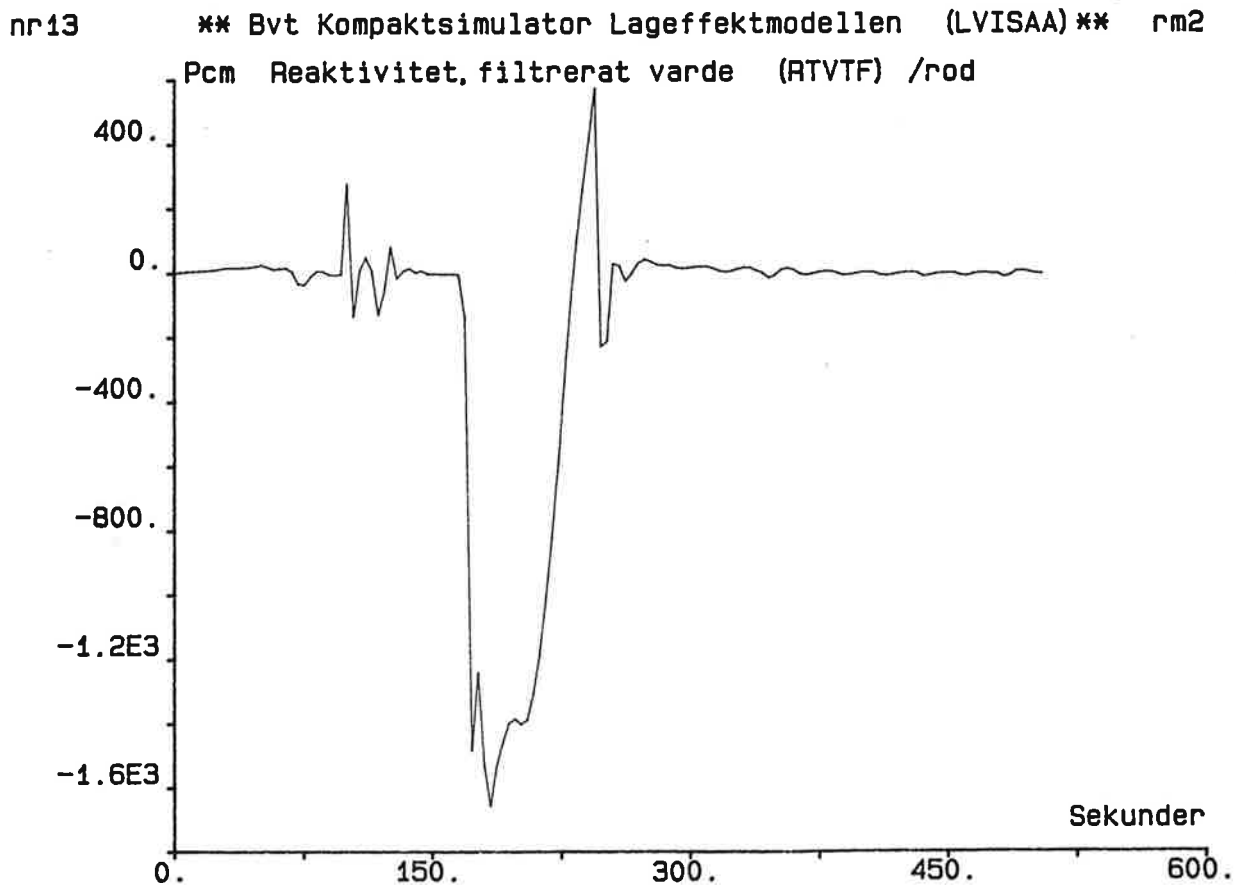
89.05.08 - 15:10:38 nr: 6
hcopy hp 1.2



89.05.08 - 15: 05: 38 nr: 4
hcopy hp 1.2

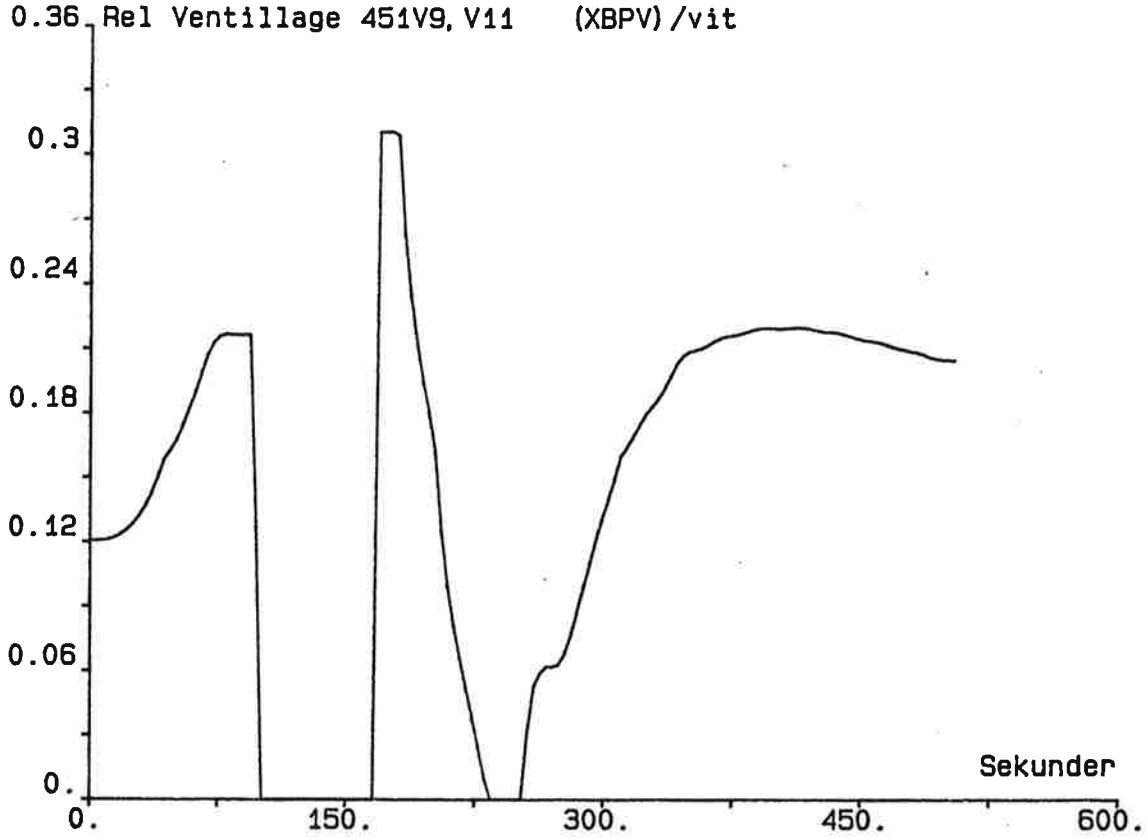


89.05.08 - 15: 12: 00 nr: 7
hcopy hp 1.2



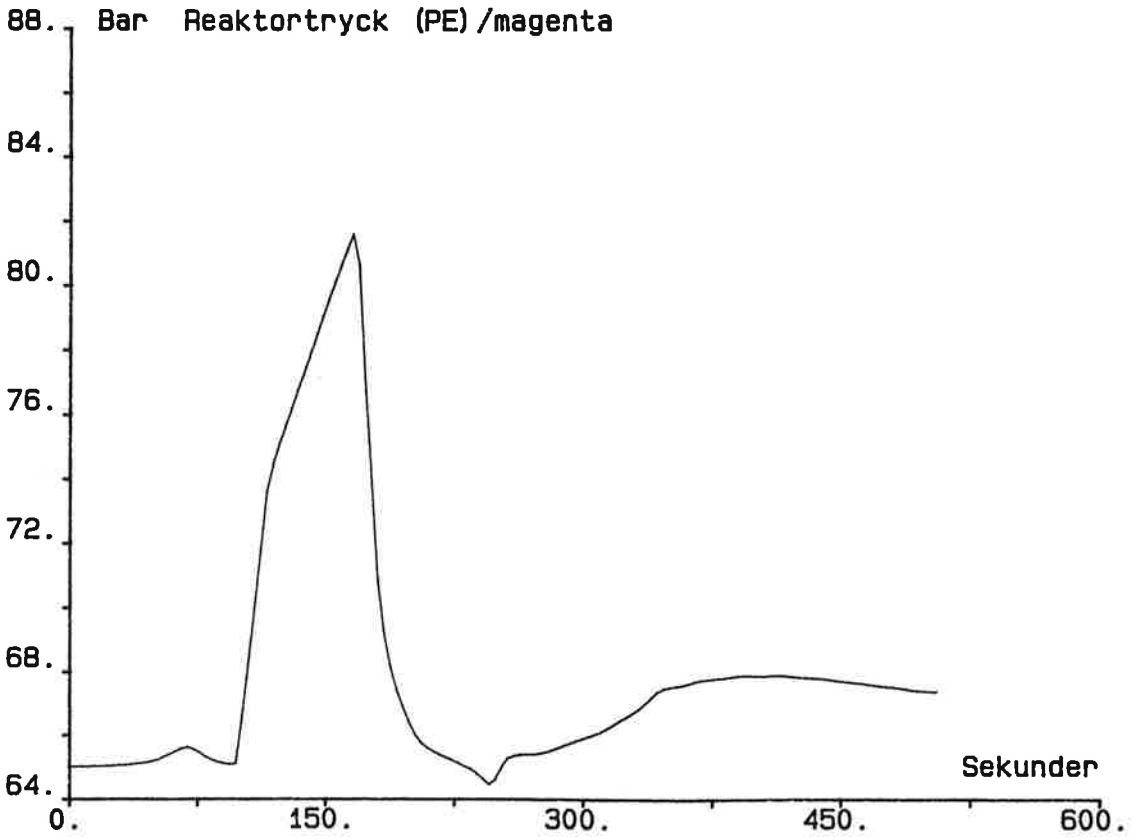
89.05.08 - 15:00:57 nr: 1
hcopy hp 1.2

nr36 ** Bvt Kompaktsimulator Lageffektmodellen (LVISAA) ** rm2



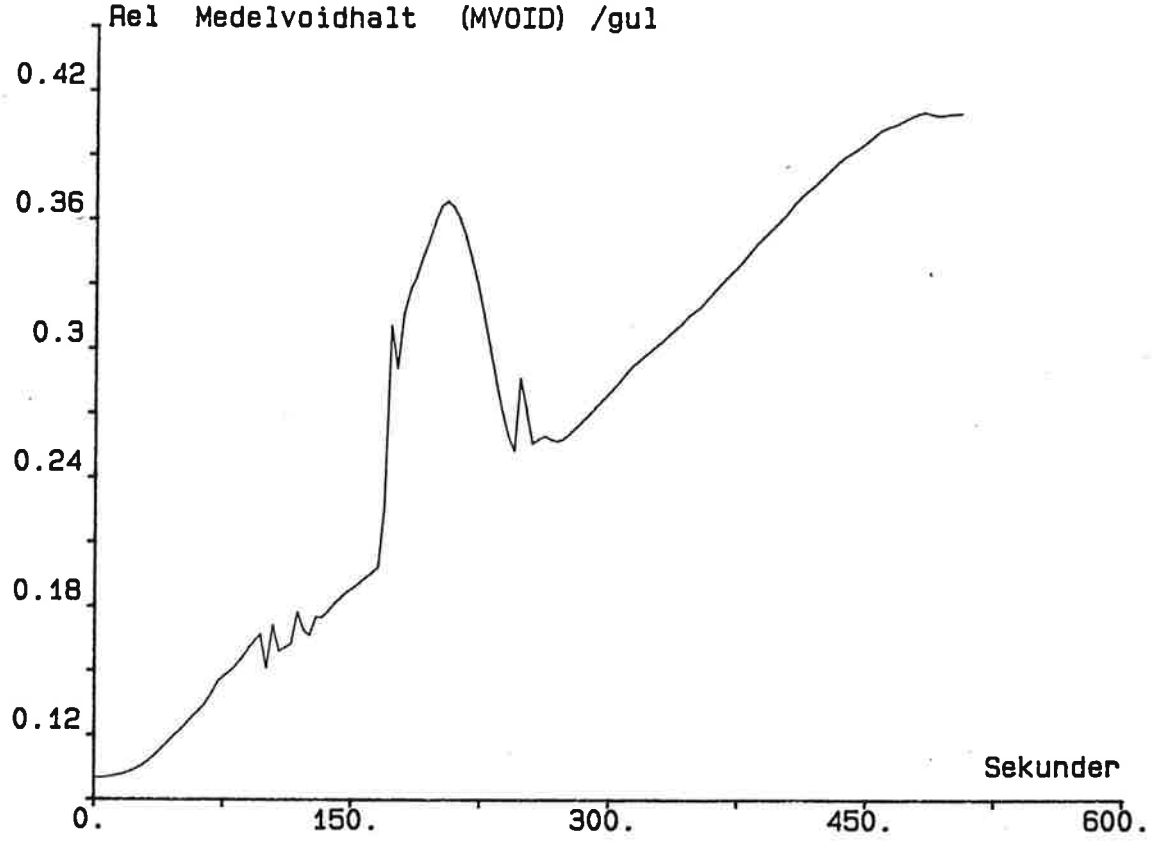
89.05.08 - 15:09:48 nr: 5
hcopy hp 1.2

nr44 ** Bvt Kompaktsimulator Lageffektmodellen (LVISAA) ** rm2



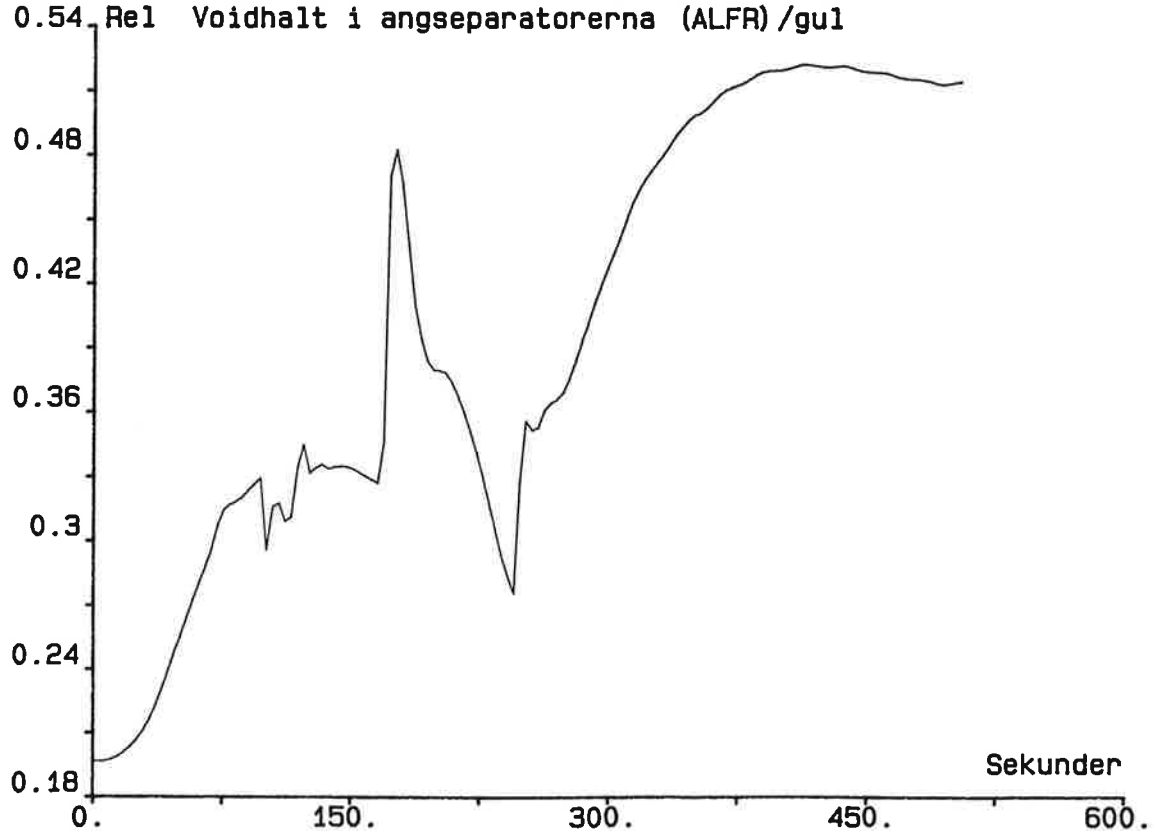
89.05.08 - 15: 12: 43 nr: 8
hcopy hp 1.2

nr16 ** Bvt Kompaktsimulator Lageffektmodellen (LVISAA) ** rm2



89.05.08 - 15: 03: 00 nr: 2
hcopy hp 1.2

nr11 ** Bvt Kompaktsimulator Lageffektmodellen (LVISAA) ** rm2



4. Kommentarer

Eftersom modellen ursprungligen inte är avsedd att fungera i realtid krävs en grundlig genomarbetning av modellen. Speciellt kan körningen av styrstavar förenklas betydligt. Med nuvarande modell fungerar inte styrstavsmanövreringen tillfredsställande.

Styrstavsgrupper

I verkligheten är styrstavarna indelade i 29 grupper de flesta med fyra stavar i varje grupp, se fig. 4.1. Jag har delat upp grupperna på ett identiskt sätt. Ett segment består av fyra rutor som motsvarar styrstavarna och en ruta med gruppnummer, där läget på styrstavarna i gruppen anges.

I modellen är dock styrstavarna indelade i 24 grupper med 1,2,4 eller 8 stavar. Jag anser att det är lämpligt att ändra modellen så att även denna består av 29 grupper, detta för att kunna simulera styrstavsmanövreringen på ett verklighetstroget sätt.

I modellen är alla styrstavar likvärdiga d.v.s. det finns inget geometriskt beroende på effektutvecklingen. Effektutvecklingen i de fyra patronerna, se fig. 4.1 , är i verkligheten en funktion av hur mycket staven är utdragen, stavens geometriska position och hur mycket de omgivande stavarna är utdragna. Man skulle kunna göra något slags viktfunktion för att simulera denna effekt.

STOPPA och FORTSÄTTA

Under pågående simulering måste man vänta tills den inställda tiden löpt ut om man vill fortsätta simuleringen. Om man påverkar stoppknappen, stoppar man både fortran och simnon, och det går inte att återuppta simuleringen. Det går ej heller att få ett gemensamt diagram för en simuleringen som gjorts i flera steg.

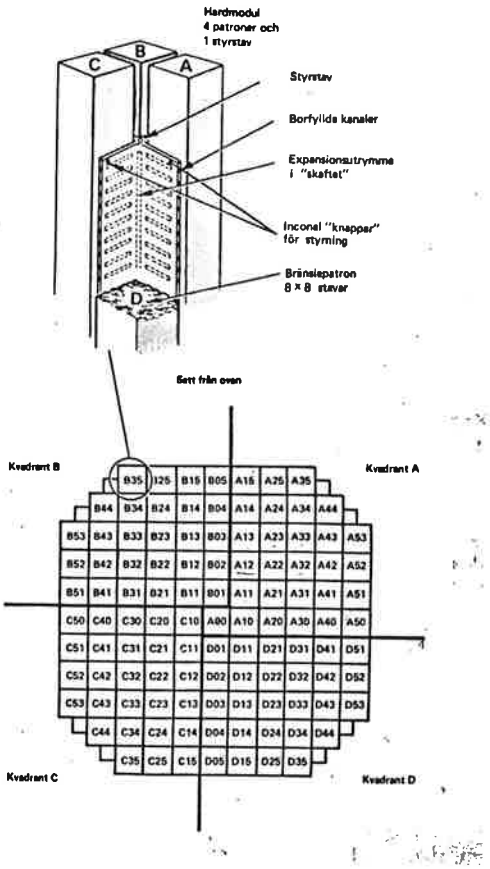
KOMMANDO I REALTID

Inläsning av cursorns position görs en gång i sekunden vilket kan vara frustrerande. Ibland får man vänta innan ett önskat kommando läses och ibland kan man av misstag dra cursorn över ett aktivt segment. Det hade varit önskvärt att kvittera ett kommando med en knapptryckning.

Denna möjlighet finns ej i SIMNON. Det kan finnas möjlighet att komma runt denna svaghet genom att skriva ett separat fortranprogram som sköter om inläsningen av kommandotecknen. Man startar sedan upp de båda programmen synkront och låter det nya programmet mata huvudprogrammet med kommandotecknen som läggs i CH och CH2. Detta är dock en komplicerad procedur, som författaren i skrivande stund ej har en aning om hur det skulle gå till.

4.1 Hjälpmacron i Simnon

Macrot LAND har stympats kraftigt. Här låg inläsning av de variabelinställningar som man nu kan ändra på under programmets gång. Kvar finns



Tabell 2 Manövergrupper

Svart grupp	Stavar					Vit grupp	Stavar			
E0	A00									
G1	A15	B51	C15	D51						
G3	A35	B53	C35	D53		G2	A25	B52	C25	D52
G5	A24	B42	C24	D42		G4	A14	B41	C14	D41
G7	A13	B31	C13	D31		G6	A34	B43	C34	B43
G9	A33	B33	C33	D33		G8	A23	B32	C23	D32
G11	A53	B35	C53	D35		G10	A43	B34	C43	D34
G13	A22	B22	C22	D22		G12	A12	B21	C12	D21
G15	A42	B24	C42	D24		G14	A32	B23	C32	D23
G17	A11		C11			G16	A52	B25	C52	D25
G19	A31	B13	C31	D13		G18	A21	B12	C21	D12
G21	A51	B15	C51	D15		G20	A41	B14	C41	D14
G23	A20	B02	C20	D02		G22	A10		C10	
G25	A40	B04	C40	D04		G24	A30	B03	C30	D03
G27		B11		D11		G26	A50	B05	C50	D05
G29	A44	B44	C44	D44		G28		B01		D01

Figur 4.1 Styrstavarnas utseende och indelning i grupper.

inläsning av styrstavskommando, macrot har fått nytt namn NEWLAND. När man får styrstavsmanövreringen att fungera kan detta macro skippas helt.

I Macrot UPP har endast små förändringar gjorts. Exempelvis sätts terminalen i grafisk mod innan simuleringen startar,SWITCH GRAPH ON. Man kunde i den gamla versionen inte välja, hur lång tid en återupptagen simulering skulle fortgå . Om man ställt in 100 sek i den ursprungliga simuleringen, så blev detta också tiden för den fortsatta simuleringen. Man kan nu välja en godtycklig stopptid. Macrot har nytt namn NEWUPP.

I connecting system LCONS har ett flertal rader med villkor på ventiler och pumpar fått ändras, se listningarna i appendix L.

5. Referenser

LA COUR CHRISTENSEN, P. och HVIDFELDT LARSEN, A. M. (1985): "Description of the Barsebäck low power plant model for the compact simulator ; Part 1 & 2," Risö.

(1983): "Tektronix 4107/4109 Display terminals, programmers reference," Part.no. 070-4893-00 Product group 18.

EKMAN och ERIKSSON (1981): "Programmering i Fortran 77," Studentlitteratur, Lund.

ÅSTRÖM, K. J. (1985): "A simnon tutorial," CODEN: LUTFD2/TFRT-3176, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

A. Grafikprogrammering

A.1 Kodning

Alla tal i de kommandon man skickar från ett fortranprogram måste vara kodade till ASCII-tecken. Detta gäller bl.a. skärmens x,y-koordinater. För detta ändamål skrevs ett antal subrutiner.

Heltal

I fig. A.1visas hur man kodar talet -2413. Jag ska gå igenom vad subrutinen INTKOD gör, se listning. A.1. Observera att rutinen inte fungerar för negativa tal då det har inte varit nödvändigt att använda sådana.

Vi tar ett tal ur luften, 2345. Skriv talet i binär form.

16384	4096	1024	512	256	128	64	32	16	8	4	2	1
0	0	0	0	1	0	0	1	0	0	1	0	1	0	0	1

Talet ska delas upp i tre delar. I den minst signifikanta delen får det plats fyra bitar plus information om tecknet medan de andra två tar sex bitar var.

Som framgår av figur A.1 tar man de sex mest signifikanta bitarna och sätter en etta framför dessa. De första tecknet ligger alltså mellan 1 0 0 0 0 0 0 och 1 1 1 1 1 1 1 eller mellan ASCII(64-127) se tabell A.1. Den första ettan anger att det är en Hi-I (High-Integer) bit. De sex mest signifikanta bitarna i 2345 är 0 0 0 0 1 0, vilket är 2. I rutinen utförs heltdivision med 1024 för att få fram Hi-I.

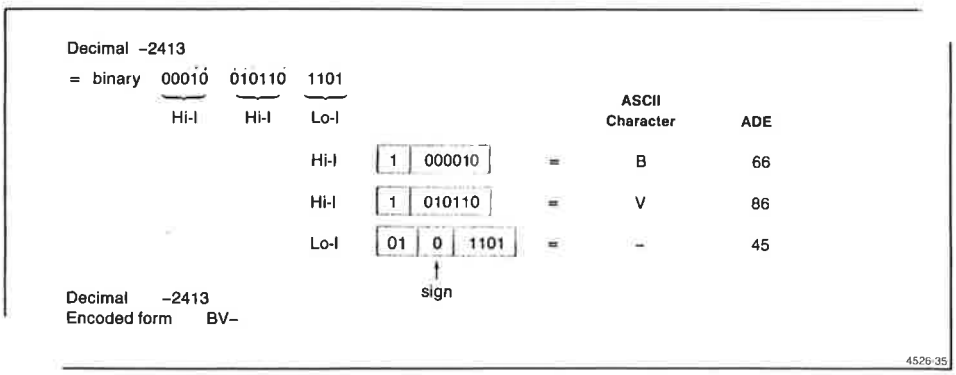
$$2345/1024 = 2$$

Första Hi-I blir därför.

$$1\ 0\ 0 \quad 0\ 0\ 1\ 0 = \text{ASCII}(64+2) = \text{ASCII}(66) = \text{'B'}$$

För att beräkna andra Hi-I dras nu den första Hi-I bort,dra ifrån $2 * 1024 = 2048, 2345 - 2048 = 297$. Därefter utförs heltalsdivision med 16.

$$297/16 = 18$$



Figur A.1 Kodning av heltal.

B7 B6 B5 / BITS / B4 B3 B2 B1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	CONTROL		FIGURES		UPPERCASE		LOWERCASE	
0 0 0 0	NU 0	DL 16	SP 32	0 48	@ 64	P 80	\ 96	p 112
0 0 0 1	SH 1	D1 17	! 33	1 49	A 65	Q 81	a 97	q 113
0 0 1 0	SX 2	D2 18	" 34	2 50	B 66	R 82	b 98	r 114
0 0 1 1	EX 3	D3 19	# 35	3 51	C 67	S 83	c 99	s 115
0 1 0 0	ET 4	D4 20	$ 36	4 52	D 68	T 84	d 100	t 116
0 1 0 1	EQ 5	NK 21	% 37	5 53	E 69	U 85	e 101	u 117
0 1 1 0	AK 6	SY 22	& 38	6 54	F 70	V 86	f 102	v 118
0 1 1 1	BL 7	EB 23	' 39	7 55	G 71	W 87	g 103	w 119
1 0 0 0	BS 8	CN 24	(40	8 56	H 72	X 88	h 104	x 120
1 0 0 1	HT 9	EM 25	) 41	9 57	I 73	Y 89	i 105	y 121
1 0 1 0	LF 10	SB 26	* 42	: 58	J 74	Z 90	j 106	z 122
1 0 1 1	VT 11	EC 27	+ 43	; 59	K 75	[91	k 107	{ 123
1 1 0 0	FF 12	FS 28	, 44	< 60	L 76	\ 92	l 108	| 124
1 1 0 1	CR 13	GS 29	- 45	= 61	M 77	] 93	m 109	} 125
1 1 1 0	SO 14	RS 30	. 46	> 62	N 78	^ 94	n 110	~ 126
1 1 1 1	SI 15	US 31	/ 47	? 63	O 79	_ 95	o 111	DT 127

Tabell A.1 En tabell över ASCII-tecken

```
      SUBROUTINE INTKOD(INT,CHARARR)
C
C   ROUTINE FOR ENCODING INTEGERS TO THREE CHARACTERS
C
C   PARAMETERS :        INT     =   INTEGER
C                       CHARARR =   ARRAY WHERE THE CHARACTERS ARE SAVED
C
      CHARACTER CHARARR(1:5)
      INTEGER A,B,C,INT
      A=INT/1024
      B=(INT-A*1024)/16
      C=INT-A*1024-B*16
      CHARARR(1)=CHAR(64+A)
      CHARARR(2)=CHAR(64+B)
      CHARARR(3)=CHAR(48+C)
      RETURN
      END
```

Lista A.1 Rutin för kodning av integer.

Jämför med 2345 i binär form 18 = 0 1 0 0 1 0. Det andra tecknet blir.

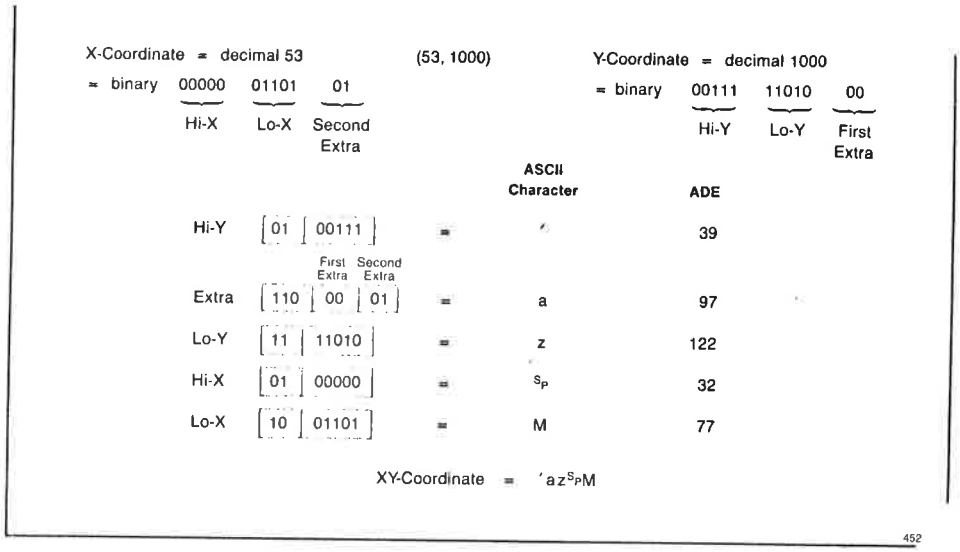
$$1\ 0\ 1 \quad 0\ 0\ 1\ 0 = \text{ASCII}(64+18)=\text{ASCII}(82)= \text{'R'}$$

Dra nu ifrån den andra Hi-I $18 * 16 = 288, 297 - 288 = 9$

Sätt 0 1 1 framför 9 = 1 0 0 1. 0 1 anger att det är en Lo-I (Low-Integer) bit och nästa etta att det är positivt tecken, se fig A.1.

$$0\ 1\ 1 \quad 1\ 0\ 0\ 1 = \text{ASCII}(48+9)=\text{ASCII}(57)= \text{'9'}$$

Vi får sluligen 2345 = 'B' 'R' '9'



Figur A.2 Hur xy-koordinater kodas.

```
*************************************************************************
C
      SUBROUTINE KOD(X,Y,KOORD)
C
C   ROUTINE FOR ENCODING X,Y-COORDINATES TO FIVE CHARACTERS
C
C   PARAMETERS :        X,Y    =   XY-COORDINATE
C                       KOORD  =   ARRAY WHERE THE CHARACTERS ARE SAVED
C
      INTEGER A,B,C,D,X,Y
      CHARACTER HIY,LOY,KOORD(1:5)
      A=X/128
      B=(X-128*A)/4
      D=X-128*A-4*B
      KOORD(4)=CHAR(32+A)
      KOORD(5)=CHAR(64+B)
      A=Y/128
      B=(Y-128*A)/4
      C=(Y-128*A-4*B)*4+D
      KOORD(2)=CHAR(96+C)
      KOORD(1)=CHAR(32+A)
      KOORD(3)=CHAR(96+B)
      RETURN
      END
*************************************************************************
```

Lista A.2 Rutin för kodning av x,y-koordinater.

X,Y-KOORDINATER

Skärmen är består av 4095 * 3071 pixel och man kan referera till dessa som x,y - koordinater. Hur kodningen går till bör framgå av föregående exempel och fig. A.2, där xy-koordinaten (53,1000) kodas till fem tecken. Rutinen heter KOD och visas i listning. A.2

Kodning av integer till siffror.

Man vill ibland skriva ut ett tal på skärmen t.ex. 5505 ska skrivas som 5505. 0 - 9 motsvarar ASCII-tecken 48-57 se tabell. A.1. Rutinen blir snarlik de föregående, listning. A.3.

```
*************************************************************************
      SUBROUTINE DECIKOD(TAL,C)
C
C   ROUTINE FOR ENCODING INTEGERS TO ITS CORRESPONDING  DECIMAL
C   CHARACTERS
C

C   PARAMETERS :        TAL     =   INTEGER
C                       C       =   ARRAY WHERE THE CHARACTERS ARE SAVED
C
      CHARACTER C(1:5)
      INTEGER DA,DB,DC,DD,DE,TAL
      DA=TAL/10000
      DB=(TAL-DA*10000)/1000
      DC=(TAL-DA*10000-DB*1000)/100
      DD=(TAL-DA*10000-DB*1000-DC*100)/10
      DE=TAL-DA*10000-DB*1000-DC*100-DD*10
      C(1)=CHAR(48+DA)
      C(2)=CHAR(48+DB)
      C(3)=CHAR(48+DC)
      C(4)=CHAR(48+DD)
      C(5)=CHAR(48+DE)
      RETURN
      END
```

Lista A.3 Rutin för kodning integer till decimala siffror.

A.2 Grafiken

Huvuddelen av arbetet har inneburit användning av Terminalens grafikdel. Se "Tektronix 4107 / 4109 COMPUTER DISPLAY TERMINALS , PROGRAMMERS REFERENCE", läs igenom kapitlet "GRAPHICS TERMINAL", beskrivning av kommandon finns i kapitlet "4100 COMMANDS". När man ger kommandon från ett Fortranprogram, måste man skicka ett kontrolltecken, (*ESC*), följt av ett antal styrtecken till terminalen, s.k. *host syntax*. I alla rutiner där grafiska kommandon används, finns därför följande rad.

ESC = CHAR(27)

Ett litet register över styrtecken finns i App K. Hur jag har använt grafiken ska jag visa med ett par enkla exempel.

Enkel grafik

Säg att vi vill dra en linje. Innan man drar linjen kan man välja hur linjen ska se ut och vilken färg den ska ha, se tabell. A.2. Med kommandot ML väljer man färg för alla följande linjer, panelgränser och märken (*eng. markers*). Om man exempelvis vill dra en röd linje från (1000,1000) till (2000,1000) skriver man följande kommandon :

```
WRITE(*,*) ESC,'ML2'                    SET LINE INDEX, välj röd färg
CALL KOD(1000,1000,KOORD)               kodar koordinaten till fem tecken.
WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)    MOVE, flyttar cursorn till 1000,100
CALL KOD(2000,1000,KOORD)
WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)    DRAW, drar en linje till 2000,1000.
```

En panel är ett område på skärmen som man kan fylla med ett av 156 förproggrammerade mönster,(*eng. fillpattern*). Här är ett exempel där man definierar ett område och fyller det med en grön färg. Mönster 0 - -15 motsvarar färgerna 0 - 15 i tabell. A.2

```
WRITE(*,*) ESC,'MP# '                   SELECT FILL PATTERN, # = -3 :
```

DEFAULT COLOR MIXTURES FOR SURFACES

Color Index	Color Mixture	Color Coordinates		
		H	L	S
0	Transparent	0	0	0
1	White	0	100	0
2	Red	120	50	100
3	Green	240	50	100
4	Blue	0	50	100
5	Cyan	300	50	100
6	Magenta	60	50	100
7	Yellow	180	50	100
8	Red-Yellow (Orange)	150	50	100
9	Green-Yellow	210	50	100
10	Green-Cyan	270	50	100
11	Blue-Cyan	330	50	100
12	Blue-Magenta	90	50	100
13	Red-Magenta	90	50	100
14	Dark-Gray	0	33	0
	Light-Gray	0	66	0

Tabell A.2 Tabell över färgindex.

```
CALL KOD(1000,1000,KOORD)
WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'1'     BEGIN PANEL,flyttar cursorn
                                             till 1000,1000.'1' anger att panelgrän
CALL KOD(2000,1000,KOORD)
WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)         DRAW, drar en linje till 2000,1000.
CALL KOD(2000,2000,KOORD)
WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)         DRAW, drar en linje till 2000,2000.
CALL KOD(1000,2000,KOORD)
WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)         DRAW, drar en linje till 1000,2000.
WRITE(*,*) ESC,'LE'                          END PANEL
```

Man har möjlighet att rita tio olika märken (*eng.* markers), detta används i programmet för att rita stapeln som visar aktuellt styrstavsmönster och "dioderna", som visar valt IRM-område. Om man vill rita ett märke av den typ som anger valt IRM-område i punkten (1500,1500) skriver man följande.

```
WRITE(*,*) ESC,'MM:'                         SET MARKER TYPE, : = 10
CALL KOD(1500,1500,KOORD)
WRITE(*,*) ESC,'LH',(KOORD(K),K=1,5)         DRAW MARKER
```

Man kan ändra på många parametrar som anger hur den grafiska texten ska se ut, höjd, bredd, färg o.d.. I listning. A.4visas program-avsnittet som skriver * * * BVT-SIMULATOR * * * med stora transparenta bokstäver. Observera att A5=21 efter LT anger hur många tecken som ska ritas.

Segment

Ett segment är ett grafiskt objekt, som efter det att det blivit definierat lagras i minnet. Det går inte att ändra på ett segments utseende i efterhand. Om man till exempel vill ändra ett segments färg måste man först ta bort det ur minnet och därefter definiera ett nytt segment. Ett segment byggs upp av linjer, märken, text och paneler.

Man kan sedan segmentet blivit definierat utföra ett antal operationer på detta. Man kan flytta, vrida, tända, släcka, förstora, förminska m.m.. Man kan i de olika kommandona istället för segmentnummer ange ett par specialtecken.

! = alla segment

```
      CALL KOD(750,770,KOORD)
      CALL INTKOD(106,CA1)
      CALL INTKOD(160,CA2)
      CALL INTKOD(52,CA3)
      WRITE(*,*) ESC,'MC',(CA1(I),I=1,3),(CA2(J),J=1,3),(CA3(K),K=1,3)
      WRITE(*,*) ESC,'MT0'
      WRITE(*,*) ESC,'MB!!'
      WRITE(*,*) ESC,'MQ1'
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LTA5*** BVT-SIMULATOR ***'
      WRITE(*,*) ESC,'MCB7C;<'
      WRITE(*,*) ESC,'MQ2'
```

Lista A.4 Utskrift av BVT-SIMULATOR i MAIN-PANEL.

```
      SUBROUTINE SEGP(X,Y,CH,CH2)
C
C    ROUTINE FOR DEFINING +/- SEGMENT
C
C    PARAMETERS :      X,Y  =  X,Y COORDINATE
C                      CH   =  NAME OF SEGMENT (FOLLOWS U)
C                      CH2  =  CHARACTER TO BE WRITTEN INSIDE THE SEGMENT
C
      CHARACTER ESC,KOORD(1:5),CH,CH2
      INTEGER X,Y
      ESC=CHAR(27)
      WRITE(*,*) ESC,'SOU',CH
      CALL KOD(X,Y,KOORD)
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'0'
      CALL KOD(X+50,Y,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(X+50,Y+50,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(X,Y+50,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
      CALL KOD(X+10,Y,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT1',CH2
      WRITE(*,*) ESC,'SC'
      WRITE(*,*) ESC,'SAU',CH,'1!12'
      WRITE(*,*) ESC,'SSU',CH,'8'
      RETURN
      END
```

Lista A.5 Rutinen SEGP

\# = segmenten i den aktuella klassen, se avsnitt 2.7.

" = alla segment som definieras i framtiden.

0 = cursorn.

Ett exempel hur ett segment definieras visas i listning A.5. Subrutinen SEGP ritar en liten ruta och skriver tecknet CH2 inuti rutan. Förväxla inte med kommandotecknet CH2 som används i huvudprogrammet.

Studera appendix K och titta i listningarna dessa kommandon använts.

A.3 Inläsning av kommando

Cursorn aktiveras med kommandot ENABLE GIN (ESC,'IE'). Normalt skickas en GIN REPORT, när operatören trycker ner en tangent på terminalen. I programmet avläses cursorns position en gång i sekunden, simuleringssekund. Detta görs med kommandot REPORT GIN POINT eller ESC,IP i host

syntax. Detta kommando medför att terminalen skickar uppgifter om cursorns position, vilket segment cursorn befinner sig i och vilken s.k. *PICK ID* segmentet har , utan att operatören trycker på någon tangent. Detta medför att programmet aldrig stannar upp och väntar på ett kommando. Informationen i en GIN REPORT kan ha följande utseende.

$$S_p(az\%NS_pV2S_pS_p0$$

12 tecken ska läsas in.

S_p	mellanslag, anger vilken tangent som blivit nedtryckt. När man använder REPORT GIN POINT sättes denna till mellanslag, eftersom ingen tangent blev nedtryckt.
(a z % N	är en kodad x,y-koordinat.
S_p V 2	segmentets namn. Det är endast dessa två programmet utnyttjar.
S_pS_p 0	PICK ID. Man kan ge delar av segment olika ID-nummer, används ej

I programmet skulle V 2 betytt, Ventil 2. I en GIN REPORT ligger alla tecken i rapporten mellan ASCII 32-63. Man får istället för V 2, 6 2. Dessa två tecken läggs i COMMON-deklarerade variabler, CH resp CH2, dessa kallar jag kommandotecken. Eftersom dessa tecken ska finnas tillgängliga i flera subrutiner bildades ett nytt COMMON-block GRAFI, se listningarna.

```
COMMON /GRAFI/ GRMODE,CH2,CH,STEPREK
```

Eftersom programmet hinner med många iterationer innan nya tecken läses in skulle t.ex. ett kommando typ stängning av en ventil innebära, att ventilen skulle börja hoppa mellan öppen och stängd. För att förhindra detta får programmet endast ett varv på sig att utföra det man begär. Därefter sätts CH till |.

A.4 Segmentklasser

Efter inläsning av GIN REPORT kommer en bit som behandlar byte av panel. Den aktuella panelen skall göras osynlig, medan den begärda panelen ska läggas ut på skärmen. Detta hade blivit en komplicerad procedur, om man skulle behandla varje segment för sig.

När segmenten definieras läggs de i olika klasser. Alla segment som ska vara med i fluxpanelen läggs i klass 1, segment tillhörande vatten- och ångpanelen i klass 2 och de som hör till styrstavspanelen i klass 3. I klass 4 ingår endast segmenten för val av panel och stoppsegmentet.

Ett kommando som tar bort segment P2 från alla segmentklasser och lägger det i klass 2 ser ut som följer.

```
WRITE(*,*) ESC,'SAP21!12'
```

ESC , Escape = CHAR(27), talar om för terminalen att de tecken som följer är styrtecken.

SA , SET SEGMENT CLASSES , kommandot.

P2 , segment.

1!, anger ur vilka klasser segmentet P2 skall tas bort.1 anger hur många tecken som följer, != -1 betyder alla klasser.

```
      CH2='|'
      IF (GRMODE.EQ.9) THEN
         WRITE(*,*) ESC,'SH!0'
         WRITE(*,*) ESC,'SV#0'
         WRITE(*,*) ESC,'SL14'
C         WRITE(*,*) ESC,'SV#1'
         WRITE(*,*) ESC,'SD#1'
         WRITE(*,*) ESC,'SL11'
         WRITE(*,*) ESC,'SV#1'
         WRITE(*,*) ESC,'SD10'
         GRMODE=1
      ENDIF
      IF((MODE.EQ.1).OR.(NR.NE.4))  GOTO 333
      NYTI=INT(T)
      IF(NYTI.LE.TI) GOTO 333
      TI=NYTI
C      WRITE(*,*) ESC,'KH2'
      CALL SHOWTIME
      WRITE(*,*) ESC,'IP1'
      WRITE(*,*) ESC,'IE11'
      READ(*,220,END=219,ERR=219) (H(K),K=1,7),CH2,CH,(H(K),K=1,6)
C      READ(*,220,END=219,ERR=219) (H(K),K=1,12)
C      WRITE(*,210) CH2,CH,CH2
C 210  FORMAT(3A)
      WRITE(*,*) ESC,'SH!0'
 219  IF ((CH.EQ.'1'.AND. CH2.EQ.' ').OR.GRMODE.EQ.9) THEN
         WRITE(*,*) ESC,'SH!0'
         WRITE(*,*) ESC,'SV#0'
         WRITE(*,*) ESC,'SL14'
C         WRITE(*,*) ESC,'SV#1'
         WRITE(*,*) ESC,'SD#1'
         WRITE(*,*) ESC,'SL11'
         WRITE(*,*) ESC,'SV#1'
         WRITE(*,*) ESC,'SD10'
         GRMODE=1
```

Lista A.6 Kommandot REPORT GIN POINT (ESC,'IP'), inläsning av tecknen och villkor för byte av panel

12 , anger vilka klasser segmentet ska tillhöra. 1 anger hur många tecken som följer.

Vissa segment ska ligga framför andra. Detta görs med kommandot SET SEGMENT DISPLAY PRIORITY. När en klass av segment görs synlig ser ett segment med högre PRIORITY ut att ligga framför ett med lägre.

WRITE(*,*) ESC,'SSP29'

SS , kommandot.

P2 , segment nummer.

9 , PRORITY.

Man kan nu referera till en mängd av segment på samma gång. Först bestämmer man vilken mängd av klasser, som man vill påverka. Detta görs med kommandot SET CURRENT MATCHING CLASS, ex ESC,'LL121!' detta skulle vara mängden av segment som tillhör klass 2 och inte någon annan. Man kan nu referera till den aktuella klassen genom att ange # istället för ett segmentnummer, när man skriver kommandon.

När man nu vill byta panel gör man först alla segment i den aktuella klassen osynliga,(SV# 0). Sedan ändrar man till en ny aktuell klass (SL11). Gör alla segment i den nya aktuella klassen synliga, (SV# 1). Slutligen sätts GRMODE till motsvarande tal. Se listning. A.6. Dessutom finns ett par rader som först gör alla panelernas segment detekterbara, (SL14 och SD!1), och sedan gör det aktuella panelens segment ej detekterbart ex. (SD20).

```
IF (SRE(56)*10.GE.LASTBAR+1 .OR. SRE(56)*10.LT.LASTBAR) THEN
  HELTAL=SRE(56)*10
  CALL DECIKOD(LASTBAR,DCOD)
  CALL KOD(3720,2390,KOORD)
  WRITE(*,*) ESC,'MT0'
  WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
  WRITE(*,*) ESC,'LT4',(DCOD(K),K=3,4),'.',DCOD(5)
  LASTBAR=HELTAL
  WRITE(*,*) ESC,'MT5'
  CALL DECIKOD(LASTBAR,DCOD)
  WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
  WRITE(*,*) ESC,'LT4',(DCOD(K),K=3,4),'.',DCOD(5)
ENDIF
```

Lista A.7 Villkor för uppdatering

A.5 Uppdatering

I huvudprogrammet ligger villkor för uppdatering. I det ursprungliga programmet visades simulerad tid en gång i sekunden. Detta gjordes från subrutinen RE. Jag har tagit bort den gamla funktionen och på dess plats lagt in anrop på SHOWTIME. Denna procedur räknar fram tiden och visar aktuellt styrstavsmönster.

Ytterligare fyra variabler visas oavsett vilken grafisk mod man ligger i, se bilderna i kap 3. De uppdateras i följande steg.

- Effekten i reaktorhärden (1 MW)
- Trycket i reaktorhärden (0.1 bar)
- Nivån i reaktortanken (1 cm)
- Temperatur i reaktortanken (1 ^0C)

Villkoren styrs av ett integer t.ex. LASTBAR för trycket. Om trycket understiger detta värde, räknas trycket ner och om trycket överstiger LASTBAR+0.1, så räknas trycket upp. Se listning A.7.

I WATER & STEAM - panelen finns tre ventiler, vars läge skall visas. Detta skall naturligtvis endast göras om GRMODE = 2, alltså om denna panel är displayad. Programmet måste också skriva ut aktuellt värde när man kommer från en annan panel. Om man begärt WATER & STEAM-panelen, så har man tryckt på motsvarande segment på skärmen, vilket innebär att man laddat de två kommandotecknen CH och CH2 med 2 respektive S_p. Observera att segmentet, som motsvarar den aktuella panelen, inte är aktivt. Detta har gjorts med kommandot SET SEGMENT DETECTABILITY, (ESC,'SD').

Flux- och fördubblingstidsinstrumenten ser annorlunda ut, men villkoren för uppdatering är uppbyggda på samma sätt.

Instrumenten ligger inritade i MAINPANEL segmentet. Det enda som händer när gränserna överskrids är att den gamla visaren suddas ut, ett vitt streck dras över den gamla visaren. Sedan ritas en ny röd visare.

Visaren är 300 pixels lång och dras från instrumentets mittpunkt i en vinkel mellan 225^0 och -45^0. Rutinen som utför detta heter IRMGR. Rutinen har tre inparametrar, X,Y är instrumentets mittpunkt och ett tal TAL som ligger mellan 0 och 1. Värdet 0 ger nollutslaget 225^0 och 1 ger maxutslag -45^0, se appendix L.

```
      IF (CH2.EQ.'5') THEN
        IF (CH.EQ.'7') CFC(3)=CFC(3)+10
        IF (CH.EQ.'8'.AND. CFC(3).GT.0) CFC(3)=CFC(3)-10
      ENDIF


     IF (CFC(3).GE.LASTWFER+1 .OR. CFC(3).LT. LASTWFER
#      .OR. (CH2.EQ.' '.AND. CH.EQ.'2')) THEN
       WRITE(*,*) ESC,'MT?'
       CALL KOD(370,680,KOORD)
       WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
       WRITE(*,*) ESC,'LT3',(CHRWF(K),K=3,5)
       LASTWFER=CFC(3)
       CALL DECIKOD(LASTWFER,CHRWF)
       WRITE(*,*) ESC,'MT1'
       WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
       WRITE(*,*) ESC,'LT3',(CHRWF(K),K=3,5)
     ENDIF
```

Lista A.8 Ändring av börvärden.

```
      IF (CH2.EQ.' ' .AND. CH.EQ.'6')  THEN
         IF (XFL(5).GT.0) THEN
           XFL(5)=0
           XFL(6)=0
           WRITE(*,*) ESC,'MP"'
         ELSE
           XFL(5)=1
           XFL(6)=1
           WRITE(*,*) ESC,'MP#'
         ENDIF
         WRITE(*,*) ESC,'SK6'
         CALL SIRM
       ENDIF
```

Lista A.9 Inläsning av SIRM och AUTO kommandon.

A.6 Börvärden

Tre börvärden kan ställas in, matarvattnet, trycket och flödet i resteffektkretsarna. Den sistnämda innebär ingen typ av reglering, utan flödet ställs genast in till det önskade. De andra två ändrar börvärdet till en PID-regulator som öppnar/stänger ventiler.

Värdet ställs in med +/- knappar, se bild 2 i kap 2. Segmenten heter Z1,Z2 o.s.v. se listning A.8och listningen av subrutinen SEGP i WATER-blocket.

A.7 AUTO och SIRM

I fluxpanelen finns två knappar att pilla på . AUTO-omkopplaren för val av IRM-område kan slås av och SIRM-detektorn kan dras ur härden. Se listning A.9och listning av AUTO och SIRM rutinerna i LPGRAPH-blocket.

A.8 Öppna och stänga

I modellen ligger variabler, som är indikatorer på om ventiler är öppna eller stängda och om pumpar är i drift eller inte. Exempelvis finns det en variabel P12 som anger om pumparna 321P1,P2 är i drift. Motsvarande segment definieras i modulen FC och heter P1 resp. P2.

```
IF (CH2.EQ.'0'.AND.XFC(6).EQ.1) THEN
  XFC(6)=0
ELSEIF(CH2.EQ.'0') THEN
  XFC(6)=1
ENDIF
```

```
IF (XFC(6).NE.LASTP12) THEN
  LASTP12=XFC(6)
  IF (XFC(6).EQ.0) THEN
    WRITE(*,*) ESC,'MP"'
  ELSE
    WRITE(*,*) ESC,'MP#'
  ENDIF
  WRITE(*,*) ESC,'SKP2'
  WRITE(*,*) ESC,'SKP1'
  CALL PUMP(1020,820,'1')
  CALL PUMP(1020,700,'2')
  WRITE(*,*) ESC,'SHP11'
  WRITE(*,*) ESC,'SHP21'
  WRITE(*,*) CHAR(7)
ENDIF
```

Lista A.10 Inläsning av P12 kommandon.

Som tidigare nämnts får man ASCII tecken mellan 32-95 istället för 64-127 för High-Integer biten i en GIN-REPORT. Således 01 och 02 istället för P1 respektive P2.

I listning A.10 ändras värdet på P12=XCR(6) och pumparna ritas om med en ny färg. Du hittar dessa rader i FC OUTPUT SECTION. Ändring av värdet och verkan av att variabeln har ändrats är separerade. Detta för att vissa variabler ändras automatiskt av modellen, om vissa gränser har över- eller underskridits. En sådan händelse måste naturligtvis också visas på skärmen.

K. LITET SYNTAX REGISTER

Enkel grafik

LF	Draw	ML	Set line index
LG	Move	MV	Set line style
LH	Draw marker	MM	Set marker type
LP	Begin panel	LE	End panel

Graphtext

LT	Graphic text	MT	Set text index
MC	Set graphtext size	MR	Set graphtext rotation
MQ	Set graphtext precision		

Segment

SP	Set pivot point	SO	Segment open
SC	Segment close	SI	Segment image transform
SA	Set segment class	SS	Set segment display priority
SH	Segment Highlightning	SX	Segment position
SH	Segment visibility	SX	Segment position
SL	Set current matching class		
LK	Include copy of segment		

GIN-kommandon

IE	Enable GIN	ID	Disable GIN
IP	Report GIN-point		

L. Listningar

L.1 BSREAL.FOR

```
C
C
C     CHANGE HISTORY
C
C
C     JAN BLECKERT  851120-851129       CHANGES AT INSTALLATION
C
C     PLB 860117    CHANGES ACCORDING TO RISO SUGGESTION IN PECON AND
C                   FLUXM
C
C
C
C     RALPH MYRNAS  881220-890501     IMPLEMENTATION OF GRAPHIC INTERFACE
C                                     AND CHANGES DUE TO DETECTION OF ERRORS
C
C
C
C
C
C
C
      SUBROUTINE RE
C
C *   CONNECTION ROUTINE FOR REAC AND SIMNON. REACTOR.
C
      COMMON /DESTIN/ IDUM,IPART
      COMMON /TIME/ T
      COMMON /CREAC/ SRE(63),DRE(63),XRE(30),CRE(12),ARE(60)
      COMMON /CFWCON/ SFC(6),DFC(6),XFC(6),CFC(6)
      COMMON /INDI/ REIC(63),CRIC(3),FCIC(6),PCIC(9),FLIC(3),RKIC(4)
C RM  NEW COMMONBLOCK /GRAFI/
      COMMON /GRAFI/ GRMODE,CH2,CH,STEPREK
C
      REAL TCAIC(10),XENIC(10),QNDIC,TPUIIC,WGEIC,CWSFIC,FLUXIC,DBTIC   851122
      REAL PRINT,DUMMY(10)
      EQUIVALENCE(STI,CRE(1))
C
C-----------------------------------------------------------------------
C*RM  DECLARATONS FOR GRAPHIC FACILITIES
C
      INTEGER TI , GRMODE
      INTEGER STEPREK,HELTAL,LASTMW,LASTBAR,LASTLEV,LASTTEMP
      CHARACTER H(1:12),CH2,CH,CH1(1:3),KOORD(1:5),DCOD(1:5)
C-----------------------------------------------------------------------
C     DECLARATONS FOR SPECIAL FACILITIES                                851126
C                                                                         *
      CHARACTER BEL,ESC,CR,MUP
      LOGICAL SETMOD
C
C
C
      SAVE
C
      DATA SETMOD/.FALSE./
C
      BEL=CHAR(7)
      ESC=CHAR(27)
      CR=CHAR(13)
      MUP=CHAR(77)
```

```
C                                                                       *
C                                                                     851126
C-----------------------------------------------------------------------
C
C
      GOTO(1,2,3,4,5,6,7,8),IPART
C
 1    CALL IDENT(4HCONT,4HRE  )
C
C-----------------------------------------------------------------------
C     * INQUIRE IF RUNNING IN BATCH MODE                                851127
C
      IF(SETMOD) GOTO 99
      SETMOD=.TRUE.
      MODE=1
      OPEN(1,FILE='MODE.DAT',STATUS='OLD',FORM='FORMATTED',ERR=99)
      READ(1,211) MODE
211   FORMAT(I1)
      CLOSE(1,STATUS='DELETE')
99    CONTINUE
C                                                                     851127
C-----------------------------------------------------------------------
      RETURN
C
 2    CONTINUE
C *   STATE VARIABLES
      CALL STATEV(SRE(1),3,4HXQN )
      CALL STATEV(SRE(4),10,4HTC  )
      CALL STATEV(SRE(14),10,4HALF )
      CALL STATEV(SRE(24),10,4HTU  )
      CALL STATEV(SRE(34),10,4HCN1 )
      CALL STATE(SRE(44),4HTLP1)
      CALL STATE(SRE(45),4HTLP2)
      CALL STATE(SRE(46),4HTBP1)
      CALL STATE(SRE(47),4HTBP2)
      CALL STATE(SRE(48),4HTR  )
      CALL STATE(SRE(49),4HTT  )
      CALL STATE(SRE(50),4HTB  )
      CALL STATE(SRE(51),4HTDC )
      CALL STATE(SRE(52),4HTPU )
      CALL STATE(SRE(53),4HWLEV)
      CALL STATE(SRE(54),4HPP  )
      CALL STATE(SRE(55),4HPR  )
      CALL STATE(SRE(56),4HPE  )
      CALL STATE(SRE(57),4HALFR)
      CALL STATE(SRE(58),4HWRC )
      CALL STATE(SRE(59),4HYHC1)
      CALL STATE(SRE(60),4HYHC2)
      CALL STATE(SRE(61),4HTTW1)
      CALL STATE(SRE(62),4HTTW2)
      CALL STATE2(SRE(63),8HRTVTF   )
C
C *   DERIVATIVES
      CALL DERV(DRE(1),3,4HXQNP)
      CALL DERV(DRE(4),10,4HTCP )
      CALL DERV(DRE(14),10,4HALFP)
      CALL DERV(DRE(24),10,4HTUP )
      CALL DERV(DRE(34),10,4HCN1P)
      CALL DER2(DRE(44),8HTLP1P   )
      CALL DER2(DRE(45),8HTLP2P   )
      CALL DER2(DRE(46),8HTBP1P   )
      CALL DER2(DRE(47),8HTBP2P   )
      CALL DER(DRE(48),4HTRP )
      CALL DER(DRE(49),4HTTP )
      CALL DER(DRE(50),4HTBP )
      CALL DER(DRE(51),4HTDCP)
      CALL DER(DRE(52),4HTPUP)
      CALL DER2(DRE(53),8HWLEVP   )
      CALL DER(DRE(54),4HPPP )
      CALL DER(DRE(55),4HPRP )
      CALL DER(DRE(56),4HPEP )
```

```
      CALL DER2(DRE(57),8HALFRP   )
      CALL DER(DRE(58),4HWRCP)
      CALL DER2(DRE(59),8HYHC1P   )
      CALL DER2(DRE(60),8HYHC2P   )
      CALL DER2(DRE(61),8HTTW1P   )
      CALL DER2(DRE(62),8HTTW2P   )
      CALL DER2(DRE(63),8HRTVTFP  )
C
C *   INITIAL CONDITION VARIABLES
      CALL INITV(REIC,63,4HREIC)
C
C *   ALGEBRAIC VARIABLES
      CALL VARV(ARE(11),10,4HNPD )
      CALL VAR(ARE(31),4HTRCI)
      CALL VAR(ARE(32),4HTCS )
      CALL VAR(ARE(33),4HTES )
      CALL VAR2(ARE(35),8HMVOID   )
      CALL VAR(ARE(38),4HWGR )
      CALL VAR(ARE(39),4HWFR )
      CALL VAR(ARE(40),4HWR  )
      CALL VAR(ARE(41),4HWT  )
      CALL VAR(ARE(42),4HRTVT)
      CALL VAR2(ARE(46),8HREACTI  )
      CALL VAR(ARE(47),4HTCM )
      CALL VARV(ARE(49),12,4HPHI )
C
C *   OUTPUT VARIABLES
      CALL OUTPUV(ARE(1),10,4HTCA )
      CALL OUTPUV(ARE(21),10,4HXEN )
      CALL OUTPUT(ARE(34),4HQND )
      CALL OUTPUT(ARE(36),4HTPUI)
      CALL OUTPUT(ARE(37),4HWGE )
      CALL OUTPUT(ARE(43),4HCNSF)
      CALL OUTPUT(ARE(44),4HFLUX)
      CALL OUTPUT(ARE(45),4HDBT )
C
C *   INPUT VARIABLES
      CALL DINPUT(XRE(1),4HWLR )
      CALL DINPUT(XRE(2),4HWFE )
      CALL DINPUT(XRE(3),4HXSKD)
      CALL DINPUT(XRE(4),4HTREK)
      CALL DINPUT(XRE(5),4HWREK)
      CALL DINPUT(XRE(6),4HWLOS)
      CALL DINPUV(XRE(7),24,4HPCCR)
C
C *   PARAMETERS
      CALL PAR(CRE(1),4HSTI )
      CALL PAR(CRE(2),4HQSR )
      CALL PAR(CRE(7),4HQRES)
      CALL PAR(CRE(8),4HORP )
      CALL PAR(CRE(9),4HTFE )
      CALL PAR2(CRE(10),8HXENON   )
      CALL PARV2(TCAIC,10,8HTCAIC   )
      CALL PARV2(XENIC,10,8HXENIC   )
      CALL PAR2(QNDIC,8HQNDIC   )
      CALL PAR2(TPUIIC,8HTPUIIC  )
      CALL PAR2(WGEIC,8HWGEIC   )
      CALL PAR2(CNSFIC,8HCNSFIC  )
      CALL PAR2(FLUXIC,8HFLUXIC  )
      CALL PAR2(DBTIC,8HDBTIC   )
      CALL PAR2(PRINT,8HPRINT   )
      RETURN
C
 3    CONTINUE
C
C *   READ IC-VALUES AND PARAMETERS AT INSTALLATION. AFTERWARDS
C *   SKIP THIS SECTION AND USE THE SIMNON 'GET' COMMAND.
C      OPEN(15,FILE='BSOIC.DAT',STATUS='OLD',ACCESS='DIRECT',          851120
C     # FORM='FORMATTED',RECL=72)                                      851126
C      READ(15,35,REC=1) REIC                                         851126
C      READ(15,35,REC=16) CRE(1),CRE(2),(CRE(J),J=7,10)               851126
```

```
C      READ(15,35,REC=17) TCAIC,DUMMY,XENIC                              851126
C      READ(15,35,REC=22) QNDIC,TPUIIC,CNSFIC,FLUXIC,DBTIC,PRINT         851126
C 35   FORMAT(6G12.0)                                                    851126
C                                                                        851126
       RETURN
C
C *    INITIAL SECTION
 4     CONTINUE
C*RM
       WRITE(*,*) ESC,'%!0'
       NR=0
C *    INITIALIZE OUTPUT VARIABLES WITH IC-OUTPUT PARAMETERS
       DO 45 J=1,10
       ARE(J)=TCAIC(J)
 45    ARE(20+J)=XENIC(J)
       ARE(34)=QNDIC
       ARE(36)=TPUIIC
       ARE(37)=WGEIC
       ARE(43)=CNSFIC
       ARE(44)=FLUXIC
       ARE(45)=DBTIC
C *    INSERT FIXED INPUT DATA NOT DEFINED AS SIMNON PARAMETERS.
       CRE(3)=2.E-4
       CRE(4)=.485
       CRE(5)=45.
       CRE(6)=1.
C *    BYPASS TIME DELAY FUNCTION AT STEADY STATE AND CASE 104.
       IF(STI.GT.0..AND.STI.NE.104.) IERD=DELAY0(5)
       RETURN
C
C*RM   OUTPUT SECTION
C*RM-------------------------------------------------------------------------
C
 5     WRITE(*,*) ESC,'%!0'
       IF (ARE(34).GE.LASTMW+1 .OR. ARE(34).LT.LASTMW-1) THEN
         HELTAL=ARE(34)
         CALL DECIKOD(LASTMW,DCOD)
         CALL KOD(3720,2570,KOORD)
         WRITE(*,*) ESC,'MT0'
         WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
         WRITE(*,*) ESC,'LT4',(DCOD(K),K=2,5)
         LASTMW=HELTAL
         WRITE(*,*) ESC,'MT5'
         CALL DECIKOD(LASTMW,DCOD)
         WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
         WRITE(*,*) ESC,'LT4',(DCOD(K),K=2,5)
       ENDIF
       IF (SRE(56)*10.GE.LASTBAR+1 .OR. SRE(56)*10.LT.LASTBAR) THEN
         HELTAL=SRE(56)*10
         CALL DECIKOD(LASTBAR,DCOD)
         CALL KOD(3720,2390,KOORD)
         WRITE(*,*) ESC,'MT0'
         WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
         WRITE(*,*) ESC,'LT4',(DCOD(K),K=3,4),'.',DCOD(5)
         LASTBAR=HELTAL
         WRITE(*,*) ESC,'MT5'
         CALL DECIKOD(LASTBAR,DCOD)
         WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
         WRITE(*,*) ESC,'LT4',(DCOD(K),K=3,4),'.',DCOD(5)
       ENDIF
       IF (SRE(53)*100.GE.LASTLEV+1 .OR. SRE(53)*100.LT.LASTLEV) THEN
         HELTAL=SRE(53)*100
         CALL DECIKOD(LASTLEV,DCOD)
         CALL KOD(3720,2210,KOORD)
         WRITE(*,*) ESC,'MT0'
         WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
         WRITE(*,*) ESC,'LT4',DCOD(3),'.',(DCOD(K),K=4,5)
         LASTLEV=HELTAL
         WRITE(*,*) ESC,'MT5'
         CALL DECIKOD(LASTLEV,DCOD)
         WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
```

```
        WRITE(*,*) ESC,'LT4',DCOD(3),'.',(DCOD(K),K=4,5)
      ENDIF
      IF (SRE(4).GE.LASTTEMP+1 .OR. SRE(4).LT.LASTTEMP) THEN
        CALL DECIKOD(LASTTEMP,DCOD)
        CALL KOD(3720,2030,KOORD)
        WRITE(*,*) ESC,'MT0'
        WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
        WRITE(*,*) ESC,'LT4',(DCOD(K),K=2,5)
        LASTTEMP=SRE(4)
        WRITE(*,*) ESC,'MT5'
        CALL DECIKOD(LASTTEMP,DCOD)
        WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
        WRITE(*,*) ESC,'LT4',(DCOD(K),K=2,5)
      ENDIF
CRM-------------------------------------------------------------------
      RETURN
C
C *   DYNAMIC SECTION
 6    CONTINUE
      IF(NR.GT.0) GOTO 65
C-------------------------------------------------------------------
      IF(MODE.EQ.1) GOTO 64                                          851126
C      WRITE(*,*) CR                                                 880117
      TI=INT(T)                                                      *
      NYTI=INT(T)                                                    851126
C-------------------------------------------------------------------
 64   TID=0.
      CALL REAC(NR,TID)
      IF(STI.GT.0..AND.STI.NE.104.) IERD=DELAY0(1)
 65   TID=T
      NR=NR+1
      IF(NR.GT.4) NR=1
C
C-------------------------------------------------------------------
C     * PRINT SIMULATED TIME EVERY SECOND IF INTERACTIVE USER          851126
CRM
       CH2='|'
       IF (GRMODE.EQ.9) THEN
          WRITE(*,*) ESC,'SH!0'
          WRITE(*,*) ESC,'SV#0'
          WRITE(*,*) ESC,'SL14'
C          WRITE(*,*) ESC,'SV#1'
          WRITE(*,*) ESC,'SD#1'
          WRITE(*,*) ESC,'SL11'
          WRITE(*,*) ESC,'SV#1'
          WRITE(*,*) ESC,'SD10'
          GRMODE=1
       ENDIF
       IF((MODE.EQ.1).OR.(NR.NE.4))  GOTO 333
       NYTI=INT(T)
       IF(NYTI.LE.TI) GOTO 333
       TI=NYTI
C       WRITE(*,*) ESC,'KH2'
       CALL SHOWTIME
       WRITE(*,*) ESC,'IP1'
       WRITE(*,*) ESC,'IE11'
       READ(*,220,END=219,ERR=219) (H(K),K=1,7),CH2,CH,(H(K),K=1,6)
C       READ(*,220,END=219,ERR=219) (H(K),K=1,12)
C       WRITE(*,210) CH2,CH,CH2
C 210   FORMAT(3A)
       WRITE(*,*) ESC,'SH!0'
 219   IF ((CH.EQ.'1'.AND. CH2.EQ.' ').OR.GRMODE.EQ.9) THEN
          WRITE(*,*) ESC,'SH!0'
          WRITE(*,*) ESC,'SV#0'
          WRITE(*,*) ESC,'SL14'
C          WRITE(*,*) ESC,'SV#1'
          WRITE(*,*) ESC,'SD#1'
          WRITE(*,*) ESC,'SL11'
          WRITE(*,*) ESC,'SV#1'
          WRITE(*,*) ESC,'SD10'
          GRMODE=1
```

```
       ENDIF
       IF (CH.EQ.'2'.AND. CH2.EQ.' ') THEN
          WRITE(*,*) ESC,'SV#0'
          WRITE(*,*) ESC,'SL14'
          WRITE(*,*) ESC,'SD#1'
          WRITE(*,*) ESC,'SL12'
          WRITE(*,*) ESC,'SV#1'
C          WRITE(*,*) ESC,'SVP21'
          WRITE(*,*) ESC,'SD20'
          WRITE(*,*) ESC,'ML0'
          GRMODE=2
       ENDIF
       IF (CH.EQ.'3'.AND. CH2.EQ.' ') THEN
          WRITE(*,*) ESC,'SV#0'
          WRITE(*,*) ESC,'SL14'
          WRITE(*,*) ESC,'SD#1'
          WRITE(*,*) ESC,'SL13'
          WRITE(*,*) ESC,'SV#1'
C          WRITE(*,*) ESC,'SVP21'
          WRITE(*,*) ESC,'SD30'
          GRMODE=3
       ENDIF
       IF (CH.EQ.'4'.AND. CH2.EQ.' ') THEN
          WRITE(*,*) ESC,'SV!0'
          MODE=0
          NR=0
          GRMODE=0
          WRITE(*,*) ESC,'IP1'
          READ(*,220,END=336,ERR=336) (H(K),K=1,7),CH2,CH,(H(K),K=1,6)
          WRITE(*,*) ESC,'SK!'
          WRITE(*,*) ESC,'%!2'
          WRITE(*,*) 'FOR ATT STARTA SIMULATORN SKRIV RESIM !!!!!'
          STOP
       ENDIF
C
C      WRITE(*,*) CR,ESC,MUP,ESC,MUP
C      WRITE(*,220) TI,'R'
 220   FORMAT(12A)
C
 333   CONTINUE                                                          *
C                                                                       851126
C-----------------------------------------------------------------------
C
C
      CALL REAC(NR,TID)
      RETURN
C
C *   COMPUTATION ON ACCEPTED VALUES
 7    IF(STI.GT.0..AND.STI.NE.104.) IERD=DELAY0(2)
      RETURN
C
C *   FINAL COMPUTATIONS
 8    CONTINUE
C
C-----------------------------------------------------------------------
C     * ANNOUNCE THE COMPLETED SIMULATION IF INTERACTIVE USER           851126
CRM                                                                      *
      IF(MODE.EQ.1) GOTO 339
      BEL=CHAR(7)
      WRITE(*,*) ESC,'IP1'
      READ(*,220,END=336,ERR=336) (H(K),K=1,7),CH2,CH,(H(K),K=1,6)
336   WRITE(*,*) ESC,'SV#0'
      GRMODE=9
      WRITE(*,*) ESC,'%!2'
      DO 335 I=1,3
        WRITE(*,*) BEL
        DO 337 J=1,100000
337     CONTINUE
335   CONTINUE
      WRITE(*,*) CR,ESC,MUP,ESC,MUP,ESC,MUP
      WRITE(*,222) '--- READY ---'
```

```
222   FORMAT(A)
339   CONTINUE
C                                                                       *
C                                                                     851126
C-----------------------------------------------------------------------
C
C *   PRINT REACTOR PROFILES FOR PRINT > 0.
      IF(PRINT.GT.0.) CALL YOREAC
C *   STI <= 0 INDICATES A STEADY STATE CALCULATION. STORE OUTPUT VALUES
C *   AS PARAMETERS. AFTERWARDS USE THE SIMNON COMMAND 'SAVE' TO SAVE
C *   IC-VALUES AND PARAMETERS.
      IF(STI.LE.0.) THEN
        DO 85 J=1,10
        TCAIC(J)=ARE(J)
 85     XENIC(J)=ARE(20+J)
        QNDIC=ARE(34)
        TPUIIC=ARE(36)
        WGEIC=ARE(37)
        CNSFIC=ARE(43)
        FLUXIC=ARE(44)
        DBTIC=ARE(45)
      ENDIF
 999  RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE CR
C *   CONNECTION ROUTINE FOR CROD AND SIMNON. CONTROL RODS.
C
      COMMON /DESTIN/ IDUM,IPART
      COMMON /TIME/ T
      COMMON /CCROD/ SCR(3),DCR(3),XCR(12),CCR(12),ACR(30)
      COMMON /INDI/ REIC(63),CRIC(3),FCIC(6),PCIC(9),FLIC(3),RKIC(4)
      COMMON /GRAFI/ GRMODE,CH2,CH,STEPREK
C
C-----------------------------------------------------------------------
C*RM  DECLARATONS FOR GRAPHIC FACILITIES
C
      INTEGER LASTCR,LASTACR,FIRST,GRMODE,HELTAL,FLAGGA,FLAGGA1
      INTEGER HLT,NBR,GROUPNR
      CHARACTER ESC,CH2,KOORD(1:5),CRD(1:5),CH,KRD(1:5)
C-----------------------------------------------------------------------
      REAL PCRIC(24),PERIC,PRINT
      EQUIVALENCE(STI,CCR(1))
C
      SAVE
C
      GOTO(1,2,3,4,5,6,7,8),IPART
C
 1    CALL IDENT(4HCONT,4HCR  )
      RETURN
C
 2    CONTINUE
C *   STATE VARIABLES
      CALL STATE(SCR(1),4HXCR )
      CALL STATE(SCR(2),4HTREF)
      CALL STATE(SCR(3),4HXQ  )
C
C *   DERIVATIVES
      CALL DER(DCR(1),4HXCRP)
      CALL DER2(DCR(2),8HTREFP   )
      CALL DER(DCR(3),4HXQP )
C
C *   INITIAL CONDITIONS
      CALL INITV(CRIC(1),3,4HCRIC)
C
C *   ALGEBRAIC VARIABLES
      CALL VAR2(ACR(25),8HSUMCR   )
      CALL VAR(ACR(26),4HPREF)
C
```

```
C *   OUTPUT VARIABLES
      CALL OUTPUV(ACR(1),24,4HPCR )
      CALL OUTPUT(ACR(27),4HPER )
C
C *   INPUT VARIABLES
      CALL DINPUT(XCR(1),4HQN  )
      CALL DINPUT(XCR(2),4HPE  )
      CALL DINPUT(XCR(3),4HWGE )
      CALL DINPUT(XCR(4),4HSS  )
      CALL DINPUT(XCR(5),4HCRS )
      CALL DINPUT(XCR(6),4HDBT )
      CALL DINPUT(XCR(7),4HRT  )
C
C *   PARAMETERS
      CALL PAR(CCR(1),4HSTI )
      CALL PAR(CCR(2),4HQSR )
      CALL PAR(CCR(3),4HWLSR)
      CALL PAR(CCR(6),4HNGR )
      CALL PAR(CCR(7),4HGRP )
      CALL PAR(CCR(8),4HFCR )
      CALL PAR(CCR(9),4HVCR )
      CALL PAR(CCR(10),4HTG1 )
      CALL PAR(CCR(11),4HTG2 )
      CALL PAR(CCR(12),4HPTG )
      CALL PARV2(PCRIC(1),24,8HPCRIC   )
      CALL PAR2(PERIC,8HPERIC   )
      CALL PAR2(PRINT,8HPRINT   )
      RETURN
C
 3    CONTINUE
C
C *   READ IC-VALUES AND PARAMETERS AT INSTALLATION. AFTERWARDS          851126
C *   SKIP THIS SECTION AND USE THE SIMNON 'GET' COMMAND.                851126
C      OPEN(15,FILE='BSOIC.DAT',STATUS='OLD',ACCESS='DIRECT',           851120
C     # FORM='FORMATTED',RECL=72)                                       851126
C      READ(15,35,REC=11) D,D,D,CRIC                                    851126
C      READ(15,35,REC=23) (CCR(J),J=1,3),(CCR(J),J=6,12)                851126
C      READ(15,35,REC=25) PCRIC,PERIC,PRINT                             851126
C 35   FORMAT(6G12.0)                                                   851126
C                                                                       851126
      RETURN
C
C *   INITIAL SECTION
 4    CONTINUE
C*RM-----------------------------------------------------------------------
C
      GRMODE=0
      NR=0
C*RM-----------------------------------------------------------------------
C *   INITIALIZE OUTPUT VARIABLES WITH IC-OUTPUT PARAMETERS
      DO 45 J=1,24
 45   ACR(J)=PCRIC(J)
      ACR(27)=PERIC
C *   INSERT FIXED INPUT DATA NOT DEFINED AS SIMNON PARAMETERS.
      CCR(4)=1.
      CCR(5)=1.E-2
      RETURN
C
C *   GRAPHIC OUTPUT SECTION                                    RM 880119
C*RM-----------------------------------------------------------------------
C
 5    ESC=CHAR(27)
      IF (XCR(4).GT.0 .AND. FLAGGA.LT.1) THEN
          FLAGGA=2
          WRITE(*,*) ESC,'SKS2'
          WRITE(*,*) ESC,'MT?'
          WRITE(*,*) ESC,'MP"'
          CALL SINDI(3960,2400,'2')
          WRITE(*,*) ESC,'SHS2'
      ENDIF
      IF (XCR(5).GT.0 .AND. FLAGGA1.LT.1) THEN
```

```
            FLAGGA1=2
            WRITE(*,*) ESC,'SKS1'
            WRITE(*,*) ESC,'MT?'
            WRITE(*,*) ESC,'MP"'
            CALL SINDI(3960,2850,'1')
            WRITE(*,*) ESC,'SHS1'
         ENDIF
         IF (GRMODE.EQ.0) THEN
            IF (ACR(25).GT.0) THEN
               FIRST=1
               GRMODE=1
               CALL WATER
               WRITE(*,*) ESC,'SL12'
               WRITE(*,*) ESC,'SV#0'
               CALL PANCROD
               NBR=CCR(7)
               IF (CCR(7).GT.0 .AND. CCR(8).GT.0) THEN
                  CALL RODCOMSEG(100,150,'1','+','"')
               ELSE
                  CALL RODCOMSEG(100,150,'1','+','0')
               ENDIF
               IF (CCR(7).EQ.0 .OR. CCR(8).EQ.0) THEN
                  CALL RODCOMSEG(200,150,'2','0','"')
               ELSE
                  CALL RODCOMSEG(200,150,'2','0','0')
               ENDIF
               IF (CCR(7).GT.0 .AND. CCR(8).LT.0) THEN
                  CALL RODCOMSEG(300,150,'3','-','"')
               ELSE
                  CALL RODCOMSEG(300,150,'3','-','0')
               ENDIF
               WRITE(*,*) ESC,'SL13'
               WRITE(*,*) ESC,'SV#0'
               WRITE(*,*) ESC,'MM1'
               CALL PANEL
               WRITE(*,*) ESC,'SV50'
               WRITE(*,*) ESC,'SV51'
               WRITE(*,*) ESC,'SV60'
               WRITE(*,*) ESC,'SV61'
               CALL INGRAPH
               WRITE(*,*) ESC,'SDA10'
               WRITE(*,*) ESC,'SDA20'
               WRITE(*,*) ESC,'SL11'
C               WRITE(*,*) ESC,'SV#0'
               WRITE(*,*) ESC,'SV#1'
               WRITE(*,*) ESC,'SD10'
               WRITE(*,*) ESC,'IE11'
               GRMODE=1
            ENDIF
         ENDIF
         IF (CH2.EQ.' '.AND.CH.EQ.'1') LASTCR=0
         IF (GRMODE.EQ.1) THEN
            IF (ACR(25).LT.LASTCR-80) THEN
              WRITE(*,*) ESC,'ML3'
              CALL CRGRAPH(LASTCR)
              LASTCR=LASTCR-80
            ELSEIF (ACR(25).GT.LASTCR+80) THEN
              WRITE(*,*) ESC,'ML2'
              LASTCR=LASTCR+80
              CALL CRGRAPH(LASTCR)
            ENDIF
         ENDIF
         IF (CH2.EQ.' '.AND.CH.EQ.'3') THEN
           DO 555 HELTAL=1,24
555        CALL UPDATEROD(HELTAL,ACR(HELTAL),0)
         ENDIF
         IF (GRMODE.EQ.3) THEN
           IF(CH2.EQ.'#' .OR. CH2.EQ.'$' .OR. CH2.EQ.'%') THEN
             WRITE(*,*) ESC,'SH',CHAR(ICHAR(CH2)+32),CH,'1'
             NBR=GROUPNR(CH2,CH)
           ENDIF
```

```
          IF (CH2.EQ.':') THEN
            IF (CH.EQ.'1') THEN
              CCR(7)=NBR
              CCR(8)=100-ACR(NBR)
              SCR(1)=ACR(NBR)
              CCR(1)=101
              CALL KOD(100,500,KOORD)
              WRITE(*,*) ESC,'MTO'
              WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
              WRITE(*,*) ESC,'LT3',(KRD(K),K=3,5)
              WRITE(*,*) ESC,'MT4'
              CALL DECIKOD(NBR,KRD)
              WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
              WRITE(*,*) ESC,'LT3',(KRD(K),K=3,5)
              WRITE(*,*) ESC,'SKZ1'
              WRITE(*,*) ESC,'SKZ2'
              WRITE(*,*) ESC,'SKZ3'
              CALL RODCOMSEG(100,150,'1','+','"')
              CALL RODCOMSEG(200,150,'2','0','0')
              CALL RODCOMSEG(300,150,'3','-','0')
            ENDIF
            IF (CH.EQ.'2') THEN
              WRITE(*,*) ESC,'SKZ1'
              WRITE(*,*) ESC,'SKZ2'
              WRITE(*,*) ESC,'SKZ3'
              CALL RODCOMSEG(100,150,'1','+','0')
              CALL RODCOMSEG(200,150,'2','0','"')
              CALL RODCOMSEG(300,150,'3','-','0')
              CCR(7)=0
              CCR(8)=0
            ENDIF
            IF (CH.EQ.'3') THEN
              CALL KOD(100,500,KOORD)
              WRITE(*,*) ESC,'MTO'
              WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
              WRITE(*,*) ESC,'LT3',(KRD(K),K=3,5)
              WRITE(*,*) ESC,'MT4'
              CALL DECIKOD(NBR,KRD)
              WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
              WRITE(*,*) ESC,'LT3',(KRD(K),K=3,5)
              CCR(1)=101
              CCR(7)=NBR
              SCR(1)=ACR(NBR)
              CCR(8)=-ACR(NBR)
              CCR(9)=100
              WRITE(*,*) ESC,'SKZ1'
              WRITE(*,*) ESC,'SKZ2'
              WRITE(*,*) ESC,'SKZ3'
              CALL RODCOMSEG(100,150,'1','+','0')
              CALL RODCOMSEG(200,150,'2','0','0')
              CALL RODCOMSEG(300,150,'3','-','"')
            ENDIF
          ENDIF
          HELTAL=CCR(7)
          HLT=ACR(HELTAL)
          IF (LASTACR.GE.HLT+1 .OR. LASTACR.LT.HLT) THEN
            CALL UPDATEROD(HELTAL,ACR(HELTAL),1)
            LASTACR=HLT
          ENDIF
        ENDIF
CRM-----------------------------------------------------------------------
        RETURN
C
C *     DYNAMIC SECTION
 6      CONTINUE
        IF(NR.GT.0) GOTO 65
        TID=0.
        CALL CROD(NR,TID)
 65     TID=T
        NR=NR+1
        IF(NR.GT.4) NR=1
```

```
      CALL CROD(NR,TID)
      RETURN
C
C *   COMPUTATION ON ACCEPTED VALUES
 7    RETURN
C
C *   FINAL COMPUTATIONS
 8    CONTINUE
C *   PRINT CONTROL ROD GROUP TABLE IF PRINT > 0
      IF(PRINT.GT.0.) CALL YOCROD
C *   STI <= 0 INDICATES A STEADY STATE CALCULATION. STORE OUTPUT VALUES
C *   AS PARAMETERS. AFTERWARDS USE THE SIMNON COMMAND 'SAVE' TO SAVE
C *   IC-VALUES AND PARAMETERS.
      IF(STI.LE.0.) THEN
        DO 85 J=1,24
 85     PCRIC(J)=ACR(J)
        PERIC=ACR(27)
      ENDIF
      RETURN
      END
C
CRM---------------------------------------------------------------------
C
      SUBROUTINE FC
C *   CONNECTION ROUTINE FOR FWCON AND SIMNON. FEEDWATER CONTROL.
C
      COMMON /DESTIN/ IDUM,IPART
      COMMON /TIME/ T
      COMMON /CFWCON/ SFC(6),DFC(6),XFC(6),CFC(6)
      COMMON /INDI/ REIC(63),CRIC(3),FCIC(6),PCIC(9),FLIC(3),RKIC(4)
      COMMON /GRAFI/ GRMODE,CH2,CH,STEPREK
      COMMON /CSPEC/ SPECIAL
C
      SAVE
C---------------------------------------------------------------------
C*RM  DECLARATONS FOR GRAPHIC FACILITIES
C
      CHARACTER ESC,CH2,CH,CHRXH(1:5),KOORD(1:5),CHRWF(1:5)
      INTEGER GRMODE,LASTP12,LASTASKH,LASTXH,LASTV42,LASTWFER
      REAL SPECIAL
      ESC=CHAR(27)
CRM---------------------------------------------------------------------
C
      GOTO(1,2,3,4,5,6,7,8),IPART
C
 1    CALL IDENT(4HCONT,4HFC  )
      RETURN
C
 2    CONTINUE
C *   STATE VARIABLES
      CALL STATE(SFC(1),4HWFE )
      CALL STATEV(SFC(2),3,4HXH  )
      CALL STATE(SFC(5),4HV42 )
      CALL STATE(SFC(6),4HV23 )
C
C *   DERIVATIVES
      CALL DER(DFC(1),4HWFEP)
      CALL DERV(DFC(2),3,4HXHP )
      CALL DER(DFC(5),4HV42P)
      CALL DER(DFC(6),4HV23P)
C
C *   INITIAL CONDITIONS
      CALL INITV(FCIC(1),6,4HFCIC)
C
C *   INPUT VARIABLES
      CALL DINPUT(XFC(1),4HWLEV)
      CALL DINPUT(XFC(2),4HPE  )
      CALL DINPUT(XFC(3),4HWLR )
      CALL DINPUT(XFC(4),4HSS  )
      CALL DINPUT(XFC(5),4HASKH)
      CALL DINPUT(XFC(6),4HP12 )
```

```
C
C *   PARAMETERS
      CALL PAR(CFC(1),4HGL  )
      CALL PAR(CFC(2),4HGW  )
      CALL PAR(CFC(3),4HWFER)
      RETURN
C
 3    CONTINUE
C
C *   READ IC-VALUES AND PARAMETERS AT INSTALLATION. AFTERWARDS
C *   SKIP THIS SECTION AND USE THE SIMNON 'GET' COMMAND.
C      OPEN(15,FILE='BSOIC.DAT',STATUS='OLD',ACCESS='DIRECT',            851120
C     # FORM='FORMATTED',RECL=72)                                        851126
C      READ(15,35,REC=12) FCIC                                          851126
C      READ(15,35,REC=30) (CFC(J),J=1,3)                                851126
C 35   FORMAT(6G12.0)                                                   851126
C                                                                       851126
      RETURN
C
C *   INITIAL SECTION
 4    CONTINUE
C*RM-----------------------------------------------------------------------
C
      WRITE(*,*) ESC,'MP'''
      WRITE(*,*) ESC,'ML0'
      CALL VENTIL(820,760,'8')
      WRITE(*,*) ESC,'MP"'
      CALL VENTIL(820,880,'9')
      WRITE(*,*) ESC,'SAV21!12'
      WRITE(*,*) ESC,'SAV31!12'
      LASTV42=0
      LASTP12=2
      LASTASKH=2
      NR=0
C*RM-----------------------------------------------------------------------
      RETURN
C
C *   OUTPUT SECTION
 5    CONTINUE
C*RM-----------------------------------------------------------------------
C
      IF (GRMODE.EQ.2) THEN
        IF (CH2.EQ.'0'.AND.XFC(6).EQ.1) THEN
          XFC(6)=0
        ELSEIF(CH2.EQ.'0') THEN
          XFC(6)=1
        ENDIF
        IF (CH2.EQ.'5') THEN
          IF (CH.EQ.'7') CFC(3)=CFC(3)+10
          IF (CH.EQ.'8'.AND. CFC(3).GT.0) CFC(3)=CFC(3)-10
        ENDIF
        IF (CH2.EQ.'6') THEN
          IF (CH.EQ.'2'.OR. CH.EQ.'3') THEN
            IF (XFC(5).EQ.1) THEN
              XFC(5)=0
            ELSE
              XFC(5)=1
            ENDIF
          ENDIF
        ENDIF
        IF (XFC(5).NE.LASTASKH) THEN
          LASTASKH=XFC(5)
          IF (XFC(5).EQ.0) THEN
            WRITE(*,*) ESC,'MP"'
          ELSE
            WRITE(*,*) ESC,'MP#'
          ENDIF
          WRITE(*,*) CHAR(7)
          WRITE(*,*) ESC,'SKV2'
          WRITE(*,*) ESC,'SKV3'
          CALL VENTIL(310,2200,'2')
```

```
            CALL VENTIL(310,2100,'3')
            WRITE(*,*) ESC,'SHV31'
            WRITE(*,*) ESC,'SHV21'
          ENDIF
          IF (XFC(6).NE.LASTP12) THEN
            LASTP12=XFC(6)
            IF (XFC(6).EQ.0) THEN
              WRITE(*,*) ESC,'MP"'
            ELSE
              WRITE(*,*) ESC,'MP#'
            ENDIF
            WRITE(*,*) ESC,'SKP2'
            WRITE(*,*) ESC,'SKP1'
            CALL PUMP(1020,820,'1')
            CALL PUMP(1020,700,'2')
            WRITE(*,*) ESC,'SHP11'
            WRITE(*,*) ESC,'SHP21'
            WRITE(*,*) CHAR(7)
          ENDIF
          IF (SFC(5).LT.0.01 .AND. LASTV42.GT. 0) THEN
            LASTV42=0
            WRITE(*,*) ESC,'MP"'
            WRITE(*,*) ESC,'SKV9'
            CALL VENTIL(820,880,'9')
            WRITE(*,*) ESC,'SHV91'
            WRITE(*,*) CHAR(7)
          ENDIF
          IF (SFC(5).GT.0.99.AND. LASTV42.LT.1) THEN
            LASTV42=2
            WRITE(*,*) ESC,'MP#'
            WRITE(*,*) ESC,'SKV9'
            CALL VENTIL(820,880,'9')
            WRITE(*,*) ESC,'SHV91'
            WRITE(*,*) CHAR(7)
          ENDIF
          IF (SFC(4)*100.GE.LASTXH+1 .OR. SFC(4)*100.LT. LASTXH-.5
     #      .OR. (CH2.EQ.' '.AND. CH.EQ.'2')) THEN
            WRITE(*,*) ESC,'MT?'
            CALL KOD(770,650,KOORD)
            WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
            WRITE(*,*) ESC,'LT3',(CHRXH(K),K=3,5)
            LASTXH=SFC(4)*100
            CALL DECIKOD(LASTXH,CHRXH)
            WRITE(*,*) ESC,'MT1'
            WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
            WRITE(*,*) ESC,'LT3',(CHRXH(K),K=3,5)
          ENDIF
          IF (CFC(3).GE.LASTWFER+1 .OR. CFC(3).LT. LASTWFER
     #      .OR. (CH2.EQ.' '.AND. CH.EQ.'2')) THEN
            WRITE(*,*) ESC,'MT?'
            CALL KOD(370,680,KOORD)
            WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
            WRITE(*,*) ESC,'LT3',(CHRWF(K),K=3,5)
            LASTWFER=CFC(3)
            CALL DECIKOD(LASTWFER,CHRWF)
            WRITE(*,*) ESC,'MT1'
            WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
            WRITE(*,*) ESC,'LT3',(CHRWF(K),K=3,5)
          ENDIF
        ENDIF
C*RM-----------------------------------------------------------------------
        RETURN
C
C *     DYNAMIC SECTION
 6      CONTINUE
        IF(NR.GT.0) GOTO 65
        TID=0.
        CALL FWCON(NR,TID)
 65     TID=T
        NR=NR+1
        IF(NR.GT.4) NR=1
```

```
      CALL FWCON(NR,TID)
      RETURN
C
C *   COMPUTATION ON ACCEPTED VALUES
 7    RETURN
C
C *   FINAL COMPUTATIONS
 8    RETURN
      END
C
C---------------------------------------------------------------------
C
      SUBROUTINE PC
C *   CONNECTION ROUTINE FOR PECON AND SIMNON. PRESSURE CONTROL.
C
      COMMON /DESTIN/ IDUM,IPART
      COMMON /TIME/ T
      COMMON /CPECON/ SPC(9),DPC(9),XPC(12),CPC(6)                        851122
      COMMON /INDI/ REIC(63),CRIC(3),FCIC(6),PCIC(9),FLIC(3),RKIC(4)
      COMMON /GRAFI/ GRMODE,CH2,CH,STEPREK
C
      SAVE
C---------------------------------------------------------------------
C*RM  DECLARATONS FOR GRAPHIC FACILITIES
C
      CHARACTER ESC,CH2,CH,KOORD(1:5),CHR(1:5),CHRS7(1:5),CHRX4(1:5)
      CHARACTER CHRS5(1:5)
      INTEGER GRMODE,LASTI314,LASTA314,LXBL,LPE,H,LXPV
      INTEGER WLX1,WLX2,WLX3,WLX4
      REAL LASTXSKD
C---------------------------------------------------------------------
C
      GOTO(1,2,3,4,5,6,7,8),IPART
C
 1    CALL IDENT(4HCONT,4HPC  )
      RETURN
C
 2    CONTINUE
C *   STATE VARIABLES
      CALL STATEV(SPC(1),3,4HXG  )
      CALL STATE(SPC(4),4HXH  )
      CALL STATE(SPC(5),4HXBPV)
      CALL STATEV(SPC(6),2,4HXBL )
      CALL STATE(SPC(8),4HWSL )
      CALL STATE(SPC(9),4HPES )
C
C *   DERIVATIVES
      CALL DERV(DPC(1),3,4HXGP )
      CALL DER(DPC(4),4HXHP )
      CALL DER2(DPC(5),8HXBPVP  )
      CALL DERV(DPC(6),2,4HXBLP)
      CALL DER(DPC(8),4HWSLP)
      CALL DER(DPC(9),4HPESP)
C
C *   INITIAL CONDITIONS
      CALL INITV(PCIC(1),9,4HPCIC)
C
C *   INPUT VARIABLES
      CALL DINPUT(XPC(1),4HPE  )
      CALL DINPUT(XPC(2),4HQN  )
      CALL DINPUT(XPC(3),4HPER )
      CALL DINPUT(XPC(4),4HXSKD)
      CALL DINPUT(XPC(5),4HDUMP)
      CALL DINPUT(XPC(6),4HA314)
      CALL DINPUT(XPC(7),4HI314)
      CALL DINPUT(XPC(8),4HWLX )
C
C *   PARAMETERS
      CALL PAR2(CPC(1),8HPRATE  )
      RETURN
C
```

```
 3    CONTINUE
C
C *   READ IC-VALUES AND PARAMETERS AT INSTALLATION. AFTERWARDS
C *   SKIP THIS SECTION AND USE THE SIMNON 'GET' COMMAND.
C      OPEN(15,FILE='BSOIC.DAT',STATUS='OLD',ACCESS='DIRECT',           851120
C     # FORM='FORMATTED',RECL=72)                                       851126
C      READ(15,35,REC=13) PCIC                                         851126
C      READ(15,35,REC=31) CPC(1)                                       851126
C 35   FORMAT(6G12.0)                                                  851126
C                                                                      851126
      RETURN
C
C *   INITIAL SECTION
 4    CONTINUE
C*RM-----------------------------------------------------------------------
C
      ESC=CHAR(27)
      H=100
      WRITE(*,*) ESC,'MP'''
      CALL WENTIL(2815,2080,'6')
      CALL WENTIL(1315,2010,'7')
      WRITE(*,*) ESC,'MP"'
      CALL WENTIL(2108,2180,'1')
      CALL WENTIL(2305,2180,'2')
      CALL WENTIL(2405,2180,'3')
      CALL WENTIL(2505,2180,'4')
      CALL VENTIL(1770,2295,'<')
      CALL VENTIL(1770,2355,';')
      WRITE(*,*) ESC,'ML1'
      CALL VENTIL(3040,2415,'7')
      CALL WENTIL(1115,2200,'8')
      WRITE(*,*) ESC,'ML0'
      LASTDUMP=2
      LASTI314=0
      LASTA314=2
      LASTXSKD=2
      LXBL=2
      LPE=2
      NR=0
C*RM-----------------------------------------------------------------
      RETURN
C
C *   OUTPUT SECTION
 5    CONTINUE
C*RM-----------------------------------------------------------------------
C
      IF (GRMODE.EQ.2) THEN
        IF (CH2.EQ.'5') THEN
          IF (CH.EQ.'9') XPC(3)=XPC(3)+1
          IF (CH.EQ.':') XPC(3)=XPC(3)-1
        ENDIF
        IF (CH2.EQ.'7') THEN
C          IF (CH.EQ.'8') THEN
C            IF (XPC(7).EQ.1) THEN
C              XPC(7)=0
C            ELSE
C              XPC(7)=1
C            ENDIF
          IF (CH.EQ.'9') THEN
            IF (XPC(6).EQ.1) THEN
              XPC(6)=0
            ELSE
              XPC(6)=1
            ENDIF
          ELSEIF (CH.EQ.':') THEN
            IF (XPC(5).EQ.1) THEN
              XPC(5)=0
            ELSE
              XPC(5)=1
            ENDIF
          ENDIF
```

```
ENDIF
IF(CH2.EQ.'6') THEN
IF (CH.EQ.'>'.OR.CH.EQ.'=') THEN
  IF (XPC(4).GT.0.01) THEN
    WRITE(*,*) CHAR(7)
    XPC(4)=0
  ELSE
    WRITE(*,*) CHAR(7)
    XPC(4)=1
  ENDIF
ENDIF
ENDIF
IF (CH2.EQ.'7'.AND.CH.LT.'5') THEN
  WRITE(*,*) ESC,'SK',CHAR(ICHAR(CH2)+32),CH
  IF (CH.EQ.'1') THEN
    IF (WLX1.EQ.6) THEN
      WRITE(*,*) ESC,'MP"'
      WLX1=0
    ELSE
      WRITE(*,*) ESC,'MP#'
      WLX1=6
    ENDIF
    CALL WENTIL(2108,2180,'1')
  ELSEIF (CH.EQ.'2') THEN
    IF (WLX2.EQ.1) THEN
      WRITE(*,*) ESC,'MP"'
      WLX2=0
    ELSE
      WRITE(*,*) ESC,'MP#'
      WLX2=1
    ENDIF
    CALL WENTIL(2305,2180,'2')
  ELSEIF (CH.EQ.'3') THEN
    IF (WLX3.EQ.1) THEN
      WRITE(*,*) ESC,'MP"'
      WLX3=0
    ELSE
      WRITE(*,*) ESC,'MP#'
      WLX3=1
    ENDIF
    CALL WENTIL(2405,2180,'3')
  ELSEIF (CH.EQ.'4') THEN
    IF (WLX4.EQ.1) THEN
      WRITE(*,*) ESC,'MP"'
      WLX4=0
    ELSE
      WRITE(*,*) ESC,'MP#'
      WLX4=1
    ENDIF
    CALL WENTIL(2505,2180,'4')
  ENDIF
  WRITE(*,*) CHAR(7)
  WRITE(*,*) ESC,'SHW',CH,'1'
  XPC(8)=WLX1+WLX2+WLX3+WLX4
ENDIF
IF (XPC(7).NE.LASTI314) THEN
  LASTI314=XPC(7)
  IF (XPC(7).EQ.0) THEN
    WRITE(*,*) ESC,'MP"'
  ELSE
    WRITE(*,*) ESC,'MP#'
  ENDIF
  WRITE(*,*) ESC,'SKW8'
  WRITE(*,*) ESC,'ML1'
  CALL WENTIL(1115,2200,'8')
  WRITE(*,*) ESC,'ML0'
  WRITE(*,*) ESC,'SHW81'
ENDIF
IF (XPC(6).NE.LASTA314) THEN
  LASTA314=XPC(6)
  IF (XPC(6).EQ.0) THEN
```

```
        WRITE(*,*) ESC,'MP"'
      ELSE
        WRITE(*,*) ESC,'MP#'
      ENDIF
      WRITE(*,*) ESC,'SKW9'
      CALL WENTIL(1315,2300,'9')
      WRITE(*,*) ESC,'SHW91'
      WRITE(*,*) CHAR(7)
    ENDIF
    IF (XPC(5).NE.LASTDUMP) THEN
      LASTDUMP=XPC(5)
      IF (XPC(5).EQ.0) THEN
        WRITE(*,*) ESC,'MP"'
      ELSE
        WRITE(*,*) ESC,'MP#'
      ENDIF
      WRITE(*,*) ESC,'SKW:'
      CALL WENTIL(2815,2300,':')
      WRITE(*,*) ESC,'SHV:1'
      WRITE(*,*) CHAR(7)
    ENDIF
    IF (XPC(4).NE.LASTXSKD) THEN
      LASTXSKD=XPC(4)
      IF (XPC(4).LT.0.01) THEN
        WRITE(*,*) ESC,'MP"'
      ELSE
        WRITE(*,*) ESC,'MP#'
      ENDIF
      WRITE(*,*) ESC,'SKV='
      WRITE(*,*) ESC,'SKV>'
      CALL VENTIL(1770,2235,'=')
      CALL VENTIL(1770,2415,'>')
      WRITE(*,*) ESC,'SHV=1'
      WRITE(*,*) ESC,'SHV>1'
    ENDIF
    IF(SPC(7)*H.LT.LXBL.OR.SPC(7)*H.GT.LXBL+1
#     .OR.(CH2.EQ.' '.AND.CH.EQ.'2'))THEN
      WRITE(*,*) ESC,'MT?'
      CALL KOD(1355,2000,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT3',(CHRS7(K),K=3,5)
      LXBL=SPC(7)*100
      CALL DECIKOD(LXBL,CHRS7)
      WRITE(*,*) ESC,'MT1'
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT3',(CHRS7(K),K=3,5)
    ENDIF
    IF(SPC(5)*H.LT.LXPV.OR.SPC(5)*H.GT.LXPV+1
#     .OR.(CH2.EQ.' '.AND.CH.EQ.'2'))THEN
      WRITE(*,*) ESC,'MT?'
      CALL KOD(2890,2070,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT3',(CHRS5(K),K=3,5)
      LXPV=SPC(5)*100
      CALL DECIKOD(LXPV,CHRS5)
      WRITE(*,*) ESC,'MT1'
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT3',(CHRS5(K),K=3,5)
    ENDIF
    IF (XPC(3).LT.LPE .OR. XPC(3).GE.LPE+1
#     .OR.(CH2.EQ.' '.AND.CH.EQ.'2'))THEN
      WRITE(*,*) ESC,'MT?'
      CALL KOD(1690,2550,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT3',(CHR(K),K=3,5)
      LPE=XPC(3)
      CALL DECIKOD(LPE,CHR)
      WRITE(*,*) ESC,'MT1'
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT3',(CHR(K),K=3,5)
    ENDIF
```

```
      ENDIF
C*RM-----------------------------------------------------------------------
      RETURN
C
C *   DYNAMIC SECTION
 6    CONTINUE
      IF(NR.GT.0) GOTO 65
      TID=0.
      CALL PECON(NR,TID)
 65   TID=T
      NR=NR+1
      IF(NR.GT.4) NR=1
      CALL PECON(NR,TID)
      RETURN
C
C *   COMPUTATION ON ACCEPTED VALUES
 7    RETURN
C
C *   FINAL COMPUTATIONS
 8    RETURN
      END
C
C---------------------------------------------------------------------
C
      SUBROUTINE FL
C *   CONNECTION ROUTINE FOR FLUXM AND SIMNON. FLUX MONITORING.
C
      COMMON /DESTIN/ IDUM,IPART
      COMMON /TIME/ T
      COMMON /CFLUXM/ SFL(3),DFL(3),XFL(6),CFL(6),AFL(30)
      COMMON /INDI/ REIC(63),CRIC(3),FCIC(6),PCIC(9),FLIC(3),RKIC(4)
      COMMON /GRAFI/ GRMODE,CH2,CH,STEPREK
C
      REAL SS5IC,SS6IC,SS11IC,SSIC,EIC,SIC,PRMLIC,DBTFIC
      EQUIVALENCE(STI,CFL(1))
C
      SAVE
C---------------------------------------------------------------------
C*RM  DECLARATONS FOR GRAPHIC FACILITIES
C
      INTEGER LASTIRMC,FORST,Y,NUMB,GRMODE
      REAL LASTIRM,LPRM,LASTSRM,LDBT,XPRM,NDBT,LSIRM
      CHARACTER ESC,CH2,CH
      ESC=CHAR(27)
C---------------------------------------------------------------------
C
      GOTO(1,2,3,4,5,6,7,8),IPART
C
 1    CALL IDENT(4HCONT,4HFL  )
      RETURN
C
 2    CONTINUE
C *   STATE VARIABLES
      CALL STATE(SFL(1),4HX1  )
      CALL STATE(SFL(2),4HPRMF)
      CALL STATE(SFL(3),4HDBTR)
C
C *   DERIVATIVES
      CALL DER(DFL(1),4HX1P )
      CALL DER2(DFL(2),8HPRMFP   )
      CALL DER2(DFL(3),8HDBTRP   )
C
C *   INITIAL CONDITIONS
      CALL INITV(FLIC(1),3,4HFLIC)
C
C *   ALGEBRAIC VARIABLES
      CALL VAR(AFL(1),4HSS4 )
      CALL VAR(AFL(4),4HSS7 )
      CALL VAR(AFL(5),4HSS8 )
      CALL VAR(AFL(7),4HE2  )
      CALL VAR(AFL(8),4HE3  )
```

```
      CALL VAR(AFL(9),4HS4  )
      CALL VAR(AFL(10),4HS5  )
      CALL VAR(AFL(11),4HS6  )
      CALL VAR(AFL(12),4HS7  )
      CALL VAR(AFL(13),4HS8  )
      CALL VAR(AFL(14),4HS10 )
      CALL VAR(AFL(15),4HS11 )
      CALL VAR(AFL(20),4HSRM )
      CALL VAR(AFL(21),4HIRM )
      CALL VAR(AFL(22),4HPRM )
      CALL VAR(AFL(23),4HIRMC)
      CALL VAR2(AFL(25),8HSRMLG   )
C
C *   OUTPUT VARIABLES
      CALL OUTPUT(AFL(2),4HSS5 )
      CALL OUTPUT(AFL(3),4HSS6 )
      CALL OUTPUT(AFL(6),4HSS11)
      CALL OUTPUT(AFL(16),4HSS  )
      CALL OUTPUT(AFL(17),4HE   )
      CALL OUTPUT(AFL(18),4HS   )
      CALL OUTPUT(AFL(19),4HPRML)
      CALL OUTPUT(AFL(24),4HDBTF)
C
C *   INPUT VARIABLES
      CALL DINPUT(XFL(1),4HFLUX)
      CALL DINPUT(XFL(2),4HWLEV)
      CALL DINPUT(XFL(3),4HPE  )
      CALL DINPUT(XFL(4),4HDBT )
      CALL DINPUT(XFL(5),4HXXX )
      CALL DINPUT(XFL(6),4HYYY )
C
C *   PARAMETERS
      CALL PAR(CFL(1),4HSTI )
      CALL PAR(CFL(2),4HARS )
      CALL PAR(CFL(4),4HNOSS)
      CALL PAR(CFL(5),4HNOS )
      CALL PAR(CFL(6),4HSSD )
      CALL PAR2(SS5IC,8HSS5IC   )
      CALL PAR2(SS6IC,8HSS6IC   )
      CALL PAR2(SS11IC,8HSS11IC  )
      CALL PAR(SSIC,4HSSIC)
      CALL PAR(EIC,4HEIC )
      CALL PAR(SIC,4HSIC )
      CALL PAR2(PRMLIC,8HPRMLIC  )
      CALL PAR2(DBTFIC,8HDBTFIC  )
C
 3    CONTINUE
C
C *   READ IC-VALUES AND PARAMETERS AT INSTALLATION. AFTERWARDS
C *   SKIP THIS SECTION AND USE THE SIMNON 'GET' COMMAND.
C      OPEN(15,FILE='BSOIC.DAT',STATUS='OLD',ACCESS='DIRECT',          851120
C     # FORM='FORMATTED',RECL=72)                                      851126
C      READ(15,35,REC=14) D,D,D,FLIC                                   851126
C      READ(15,35,REC=32) CFL(1),CFL(2),(CFL(J),J=4,6)                 851120
C      READ(15,35,REC=33) SS5IC,SS6IC,SS11IC,SSIC,EIC,SIC,PRMLIC,DBTFIC 851126
C 35   FORMAT(6G12.0)                                                  851126
C                                                                      851126
      RETURN
C
C *   INITIAL SECTION
 4    CONTINUE
C*RM
      NR=0
C *   INITIALIZE OUTPUT VARIABLES WITH IC-OUTPUT PARAMETERS
      AFL(2)=SS5IC
      AFL(3)=SS6IC
      AFL(6)=SS11IC
      AFL(16)=SSIC
      AFL(17)=EIC
      AFL(18)=SIC
      AFL(19)=PRMLIC
```

```
      AFL(24)=DBTFIC
C *   INSERT FIXED INPUT DATA NOT DEFINED AS SIMNON PARAMETERS.
      CFL(3)=1.
      RETURN
C
C *   OUTPUT SECTION
C*RM-------------------------------------------------------------------------
C
 5    IF (GRMODE.EQ.0) THEN
       IF (CFL(2).EQ.0) THEN
        WRITE(*,*) ESC,'MP"'
       ELSE
        WRITE(*,*) ESC,'MP#'
       ENDIF
       WRITE(*,*) ESC,'SK5'
       CALL AUTO
       IF (XFL(5).EQ.0) THEN
        WRITE(*,*) ESC,'MP"'
       ELSE
        WRITE(*,*) ESC,'MP#'
       ENDIF
       WRITE(*,*) ESC,'SK6'
       CALL SIRM
      ENDIF
      IF (CH2.EQ.'3'.AND. CFL(6).LT.1) THEN
        WRITE(*,*) ESC,'MT?'
        WRITE(*,*) ESC,'ML?'
        IF (CH.EQ.'2') THEN
          CFL(6)=1
        ELSEIF (CH.EQ.'3') THEN
          IF (CFL(5).EQ.0) THEN
            WRITE(*,*) ESC,'MP"'
            CFL(5)=1
          ELSE
            WRITE(*,*) ESC,'MPO'
            CFL(5)=0
          ENDIF
          WRITE(*,*) ESC,'SKS3'
          CALL BLOCK(3960,2550,'3')
          WRITE(*,*) ESC,'SHS31'
        ELSEIF (CH.EQ.'4') THEN
          IF (CFL(4).EQ.0) THEN
            WRITE(*,*) ESC,'MP"'
            CFL(4)=1
          ELSE
            WRITE(*,*) ESC,'MPO'
            CFL(4)=0
          ENDIF
          WRITE(*,*) ESC,'SKS4'
          CALL BLOCK(3960,2100,'4')
          WRITE(*,*) ESC,'SHS41'
        ENDIF
      ENDIF
      IF (CH2.EQ.' '.AND.CH.EQ.'1') THEN
        LDBT=0
        LASTIRM=0
        LPRM=0
        LASTSRM=0
        LASTSRMLG=0
        LASTIRMC=LASTIRMC-1
        IF (LASTIRMC.EQ.0) LASTIRMC=3
      ENDIF
      IF (GRMODE.EQ.1) THEN
      IF (CH2.EQ.' ' .AND. CH.EQ.'5')  THEN
         IF (CFL(2).GT.0) THEN
           CFL(2)=0
           WRITE(*,*) ESC,'MP"'
         ELSE
           CFL(2)=1
           WRITE(*,*) ESC,'MP#'
         ENDIF
```

```
         WRITE(*,*) ESC,'SK5'
         CALL AUTO
      ENDIF
      IF (CH2.EQ.' ' .AND. CH.EQ.'6')  THEN
         IF (XFL(5).GT.0) THEN
           XFL(5)=0
           XFL(6)=0
           WRITE(*,*) ESC,'MP"'
         ELSE
           XFL(5)=1
           XFL(6)=1
           WRITE(*,*) ESC,'MP#'
         ENDIF
         WRITE(*,*) ESC,'SK6'
         CALL SIRM
      ENDIF
      IF (NUMB.EQ.8) THEN
      CALL IRMGR(LDBT+.167,10*(AFL(22)-XPRM)/AFL(22)+.167,1250,2200)
         LDBT=10*(AFL(22)-XPRM)/AFL(22)
         XPRM=AFL(22)
         NUMB=0
      ENDIF
      NUMB=NUMB+1
      IF (AFL(21).GT.(LASTIRM+1) .OR. AFL(21).LT.(LASTIRM-1)) THEN
         CALL IRMGR(LASTIRM/120,AFL(21)/120,2350,2200)
         LASTIRM=AFL(21)
      ENDIF
      IF (AFL(23).GE.1) THEN
         IF (AFL(23).GE.(LASTIRMC+1) .OR. AFL(23).LE.(LASTIRMC-1)) THEN
            WRITE(*,*) ESC,'ML0'
            CALL  FLGRAPH(LASTIRMC)
            WRITE(*,*) ESC,'ML2'
            LASTIRMC=AFL(23)
            CALL FLGRAPH(LASTIRMC)
         ENDIF
      ENDIF
      IF (AFL(25).GE.(LASTSRM+0.1) .OR. AFL(25).LT.(LASTSRM-0.1)) THEN
         CALL  IRMGR(LASTSRM/6,AFL(25)/6,1250,1300)
         LASTSRM=AFL(25)
      ENDIF
      IF (AFL(22).GT.(LPRM+.1) .OR. AFL(22).LT.(LPRM-.1)) THEN
         CALL IRMGR(LPRM/120,AFL(22)/120,2350,1300)
         CALL IRMGR(LSIRM/120,AFL(22)/120*XFL(5),3450,1300)
         LPRM=AFL(22)
         LSIRM=AFL(22)*XFL(5)
      ENDIF
      ENDIF
C*RM-----------------------------------------------------------------
      RETURN
C
C *   DYNAMIC SECTION
 6    CONTINUE
      IF(NR.GT.0) GOTO 65
      TID=0.
      CALL FLUXM(NR,TID)
 65   TID=T
      NR=NR+1
      IF(NR.GT.4) NR=1
      CALL FLUXM(NR,TID)
      RETURN
C
C *   COMPUTATION ON ACCEPTED VALUES
 7    RETURN
C
C *   FINAL COMPUTATIONS
 8    CONTINUE
C *   STI <= 0 INDICATES A STEADY STATE CALCULATION. STORE OUTPUT VALUES
C *   AS PARAMETERS. AFTERWARDS USE THE SIMNON COMMAND 'SAVE' TO SAVE
C *   IC-VALUES AND PARAMETERS.
      IF(STI.LE.0.) THEN
        SS5IC=AFL(2)
```

```
        SS6IC=AFL(3)
        SS11IC=AFL(6)
        SSIC=AFL(16)
        EIC=AFL(17)
        SIC=AFL(18)
        PRMLIC=AFL(19)
        DBTFIC=AFL(24)
      ENDIF
      RETURN
      END
C
C---------------------------------------------------------------------
C
      SUBROUTINE RK
C *   CONNECTION ROUTINE FOR REKO AND SIMNON. RESIDUAL HEAT REMOVAL.
C
      COMMON /DESTIN/ IDUM,IPART
      COMMON /TIME/ T
      COMMON /CREKO/ SRK(4),DRK(4),XRK(6),CRK(6)
      COMMON /INDI/ REIC(63),CRIC(3),FCIC(6),PCIC(9),FLIC(3),RKIC(4)
      COMMON /GRAFI/ GRMODE,CH2,CH,STEPREK
C
      SAVE
C
C---------------------------------------------------------------------
C*RM  DECLARATONS FOR GRAPHIC FACILITIES
C
      CHARACTER ESC,KOORD(1:5),CHR(1:5),CH2,CH
      INTEGER STEPREK,LASTNHEX,GRMODE
      ESC=CHAR(27)
C---------------------------------------------------------------------
      GOTO(1,2,3,4,5,6,7,8),IPART
C
 1    CALL IDENT(4HCONT,4HRK  )
      RETURN
C
 2    CONTINUE
C *   STATE VARIABLES
      CALL STATE(SRK(1),4HTRO )
      CALL STATE(SRK(2),4HTHO )
      CALL STATE(SRK(3),4HTMC )
      CALL STATE(SRK(4),4HTMH )
C
C *   DERIVATIVES
      CALL DER(DRK(1),4HTROP)
      CALL DER(DRK(2),4HTHOP)
      CALL DER(DRK(3),4HTMCP)
      CALL DER(DRK(4),4HTMHP)
C
C *   INITIAL CONDITIONS
      CALL INITV(RKIC(1),4,4HRKIC)                                    851125
C
C *   INPUT VARIABLES
      CALL DINPUT(XRK(1),4HTRI )
      CALL DINPUT(XRK(2),4HWREK)
C
C *   PARAMETERS
      CALL PAR(CRK(1),4HTHI )
      CALL PAR(CRK(2),4HNHEX)
C
C
 3    CONTINUE
C
C *   READ IC-VALUES AND PARAMETERS AT INSTALLATION. AFTERWARDS
C *   SKIP THIS SECTION AND USE THE SIMNON 'GET' COMMAND.
C      OPEN(15,FILE='BSOIC.DAT',STATUS='OLD',ACCESS='DIRECT',          851120
C     # FORM='FORMATTED',RECL=72)                                     851126
C      READ(15,35,REC=15) RKIC                                        851126
C      READ(15,35,REC=35) CRK(1),CRK(2)                               851126
C 35   FORMAT(6G12.0)                                                 851126
C                                                                     851126
```

```
      RETURN
C
C *   INITIAL SECTION
 4    CONTINUE
C*RM-----------------------------------------------------------------------------
C
      LASTNHEX=3
      STEPREK=1
      WRITE(*,*) ESC,'MP'''
      CALL VENTIL(2400,1820,'5')
      NR=0
C-------------------------------------------------------------------------------
      RETURN
C
C *   OUTPUT SECTION
 5    CONTINUE
C*RM-----------------------------------------------------------------------------
C
      IF (CH2.EQ.' '.AND.CH.EQ.'2') THEN
        STEPREK=XRK(2)
        CALL DECIKOD(STEPREK,CHR)
        WRITE(*,*) ESC,'MT1'
        CALL KOD(2340,1900,KOORD)
        WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
        WRITE(*,*) ESC,'LT3',(CHR(K),K=3,5)
      ENDIF
      IF (GRMODE.EQ.2) THEN
      IF (CH2.EQ.'&') CRK(2)=ICHAR(CH)-48
      IF (LASTNHEX.NE.CRK(2)) THEN
        IF (LASTNHEX.GT.2) GOTO 50
        WRITE(*,*) ESC,'MP!'
        WRITE(*,*) ESC,'SKF',CHAR(LASTNHEX+48)
        CALL NOF321(CHAR(LASTNHEX+48))
 50     LASTNHEX=CRK(2)
        WRITE(*,*) ESC,'MP'''
        WRITE(*,*) ESC,'SKF',CHAR(LASTNHEX+48)
        CALL NOF321(CHAR(LASTNHEX+48))
      ENDIF
      IF (CH2.EQ.'5'.AND.CH.EQ.'5') THEN
        WRITE(*,*) CHAR(7)
        WRITE(*,*) ESC,'MT?'
        CALL KOD(2340,1900,KOORD)
        WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
        WRITE(*,*) ESC,'LT3',(CHR(K),K=3,5)
        XRK(2)=XRK(2)+10
        STEPREK=XRK(2)
        CALL DECIKOD(STEPREK,CHR)
        WRITE(*,*) ESC,'MT1'
        CALL KOD(2340,1900,KOORD)
        WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
        WRITE(*,*) ESC,'LT3',(CHR(K),K=3,5)
        WRITE(*,*) ESC,'SHV51'
      ENDIF
      IF (CH2.EQ.'5'.AND.CH.EQ.'6') THEN
        WRITE(*,*) CHAR(7)
        WRITE(*,*) ESC,'MT?'
        CALL KOD(2340,1900,KOORD)
        WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
        WRITE(*,*) ESC,'LT3',(CHR(K),K=3,5)
        XRK(2)=XRK(2)-10
        STEPREK=XRK(2)
        CALL DECIKOD(STEPREK,CHR)
        WRITE(*,*) ESC,'MT1'
        CALL KOD(2340,1900,KOORD)
        WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
        WRITE(*,*) ESC,'LT3',(CHR(K),K=3,5)
        WRITE(*,*) ESC,'SHV51'
      ENDIF
      ENDIF
C*RM-----------------------------------------------------------------------------
 51   RETURN
```

```
C
C *   DYNAMIC SECTION
 6    CONTINUE
      IF(NR.GT.0) GOTO 65
      TID=0.
      CALL REKO(NR,TID)
 65   TID=T
      NR=NR+1
      IF(NR.GT.4) NR=1
      CALL REKO(NR,TID)
      RETURN
C
C *   COMPUTATION ON ACCEPTED VALUES
 7    RETURN
C
C *   FINAL COMPUTATIONS
 8    RETURN
      END
C
*****************************************************************
C
      SUBROUTINE REAC(NR,TID)
C *   ONE-DIMENSIONAL REACTOR MODEL WITH 10 CORE SECTIONS.
C *   SPECIAL LOW-POWER VERSION
C
C *   COMMON /CREAC/ CONTAINS:
C *   S: STATE VARIABELS    : 63
C *   D: DERIVATIVES        : 63
C *   X: INPUT VARIABELS    : 30
C *   C: INPUT CONSTANTS    : 12
C *   A: ALGEBRAIC VARIABELS: 48
C
      COMMON /CREAC/
     S XQN(3),TC(10),ALF(10),TU(10),CN1(10),
     S TLP1,TLP2,TBP1,TBP2,TR,TT,TB,TDC,TPU,WLEV,PP,PR,PE,ALFR,
     S WRC,YHC1,YHC2,TTW1,TTW2,RTVTF,
     D XQNP(3),TCP(10),ALFP(10),TUP(10),CN1P(10),
     D TLP1P,TLP2P,TBP1P,TBP2P,TRP,TTP,TBP,TDCP,TPUP,WLEVP,
     D PPP,PRP,PEP,ALFRP,WRCP,YHC1P,YHC2P,TTW1P,TTW2P,RTVTFP,
     X WLR,WFE,XSKD,TREK,WREK,WLOS,PCCR(24),
     C STI,QSR,CQN,FCAL,SOUR,TAUR,
     C QRES,ORP,TFE,XENON,CDU(2),
     A TCA(10),NPD(10),XEN(10),
     A TRCI,TCS,TES,QND,MVOID,TPUI,WGE,WGR,WFR,WR,WT,RTVT,
     A CNSF,FLUX,DBT,REACTI,TCM,ADU,PHI(12)
C
      DIMENSION CM(12,3),SLCN(12),SLCNR(12),NP(10),NYSF(10),
     # S(10),Q(10),Q2(10),CCR(10),XQRE(10),
     # AGA(3),ALA(3),
     # NCCR(24),ACCR(24)
      REAL LC,LS,KSF,LM1,LMI,LMX,NSF,NY
      REAL JOD,NP,NYSF,NPD,NSKD,KW1,KW2,MVOID
      CHARACTER LTAU1*4,LTAU2*4,TTAU1*4,TTAU2*4
C
      SAVE
C
C * NB: OCA,VF,VC,CU,CCA,ZUGCA ARE SECTION VALUES AND DEPENDS OF
C *     SECTION NUMBER NS.
      DATA OCA,DEC,AC,LC,VF,VU /399.06, .0139, 4.63, 3.71, 3.84, .890/
      DATA VC,VR,VES,VT,VB,VDC,VS1/ 1.718, 10.13, 130., 37.3,          851129
     #     15.5, 36.85, 13.2 /
      DATA VHC,VPU,VLP1,VLP2,VBP1,VBP2 / 6.31, 1.0, 24.24, 21.26,
     #    25.30, 21.62 /
      DATA AR,AS,LS,AT,ADC,TAUW /10.13, 5.30, 3.15, 15.9, 6.70, 19.5/
      DATA A,NY/.3E-10,2.43/
      DATA BETA1 / 6448.E-6 /
      DATA LM1 / .436 /
      DATA LMI,LMX,GMI,GMX,SGMX/2.9E-5,2.1E-5,.056,.003,3.5E-18/
      DATA AGA /.01124,.03485,.02727/
      DATA ALA /.8952,.03314,2.400E-4/
      DATA GA /.07336/
```

```
      DATA NCCR / 3*1, 3*2, 11*4, 7*8 /
      DATA XQRE /.063,.116,.141,.130,.114,.107,.104,.095,.078,.052/
      DATA KCCR,MCCR / 24, 109 /
      DATA ROU,CCA,CAT / 10000., .644, 4. /                              851122
      DATA G,EPS / 9.807, 1.E-20 /
      DATA KSF,DPDK,CDPL,FF /3.0, .0015, .00448, .15/
      DATA XWBP,YQBP,YQWC /.117, .012, .020/
      DATA CPFE,CPRE / .00418, .00418 /
      DATA KW1,HW1,CW1 / 44.8E-3, 1.11E-3, 66. /
      DATA KW2,HW2,CW2 / 67.2E-3, 1.66E-3, 99. /
      DATA NS /12/
      DATA ARCP,BRCP,CRCP /4.8402, 2.6211, -1.2385/
      DATA LTAU1,LTAU2 / 'TAU1','TAU2'/
      DATA TTAU1,TTAU2 / 'TTU1','TTU2'/
C
C *   FUNCTION ALIM LIMITS X TO XMIN--XMAX
      ALIM(X,XMAX,XMIN)=AMAX1(XMIN,AMIN1(X,XMAX))
C *   FUNCTIONS FOR NUCLEAR CROSS SECTIONS
      FSAMC(R,DTF)=(((.29644864E-13*R-.50045248E-10)*R+.26315027E-7)*R
     #               +.81740178E-5)*R+.99082229E-2
     #               +(.182189E-06+.413516E-10*R)*DTF
     #               -(.924333E-10+.214592E-13*R)*DTF**2
      FNSFMC(R,DTF)=(((.42698479E-13*R-.88926371E-10)*R+.64618061E-7)*R
     #               -.35478623E-5)*R+.13641720E-1
     #               +(-.268294E-6+.389378E-10*R)*DTF
     #               +( .136218E-9-.193537E-13*R)*DTF**2
      FDMC(R)=((.16083916E-9*R+.16433567E-6)*R-.13050175E-2)*R
     #         +.18524773E+1
       FSAPC(R,DTF)=(((.26175786E-13*R-.44690802E-10)*R+.22986393E-7)*R
     #               +.12768707E-4)*R+.12262982E-1
     #               +(.186066E-6+.507409E-10*R)*DTF
     #               -(.948321E-10+.260749E-13*R)*DTF**2
       FNSFPC(R,DTF)=(((.43644590E-13*R-.91785272E-10)*R+.70160323E-7)*R
     #                -.26432426E-5)*R+.99275605E-2
     #                -(.254245E-6+.209118E-10*R)*DTF
     #                +(.129606E-9+.108863E-13*R)*DTF**2
       FDPC(R)=((.15151515E-9*R+.19830170E-6)*R-.13991600E-2)*R
     #          +.19701976E+1
C
C
      IF(NR.NE.0) GO TO 10
C *   INITIAL CALCULATIONS AND RESETTINGS
      DX=LC/(NS-2)
      DZ=DX*100.
C *   DZ IN CM FOR NEUTRONIC
      AFP=A*VF/NY
      BETA=BETA1
      HQ1=.023E-3*DCA/(DEC**.2*AC**.8)
      DPCK=.092*FF*DX/DEC**1.2
      VE=VES
      IF(STI.GT.0.) GOTO 10
C *   ACCELERATION FOR STEADY STATE CALCULATION
      ALA(2)=ALA(1)
      ALA(3)=ALA(1)
      CW1=66./10000.
      CW2=99./10000.
C
 10   CONTINUE
      IF(NR.GT.1) GOTO 200
C *   PARAMETERS FOR WATER AND STEAM
      TCS=TSAF(PR)
      TES=TSAF(PE)
      RFS=RFSF(TR)
      RGS=RGSF(TCS)
      CP=CPFSF(TR)/1000.
      HFG=HFGF(TCS)/1000.
      DRFS=DRFSDP(TCS)
      DRGS=DRGSDP(TCS)
      DHGS=DHGSDP(TCS)/1000.
      DRFT=DRFSDT(TR)
      CO=RGS/RFS
```

```
      C1=(RFS-RGS)/RGS
      C2=RFS*DRGS/RGS
      HC2=1.973E-3*OCA*EXP(PR/43.4)
      CCC=((.746987E-16*TR-.209935E-13)*TR+.326887E-11)*TR+.373412E-9
      CCCC=CCC*VC/DEC**2
C *   INCLUDE STEAM LINE VOLUME IN STEAM DOME (2*VS1+4*XSKD*VS1)
      VE=VES+2.*VS1+4.*XSKD*VS1
C
C
C
C *   REACTOR CORE DYNAMICS
C *   ---------------------
 200  CONTINUE
C *   CALCULATE NEUTRON FLUX
      GOTO 1000
 201  CONTINUE
C *   THERMAL POWER WITH DELAYED HEAT: QND
      QND=(1.-GA)*QN
      DO 205 J=1,3
      XQNP(J)=(QN-XQN(J))*ALA(J)
 205  QND=QND+AGA(J)*XQN(J)
C *   CORE BYPASS FLOW AND CORE FLOW: WBP,WC
      WBP=XWBP*WRC
      WC=WRC-WBP
C *   HEAT TRANSFER PARAMETER: HQ1W
      HQ1T=(((.244899E-9*TR-.135426E-6)*TR+.204433E-4)*TR
     # +.208339E-2)*TR+.290801
      HC1=HQ1T*HQ1*WC**.8
C
C *   POWER AND FUEL TEMP.PROFILE:
      DO 210 J=1,NS-2
C *   SPACIAL POWER WITH DELAYED HEAT: NPD
      NPD(J)=NP(J)*QND/QN
C *   ADDITION OF CONSTANT DECAY HEAT TERM
      NPD(J)=NPD(J)+QRES*XQRE(J)
      T1=TU(J)
      T2=TCA(J)
      T3=TC(J)
C *   HEAT TRANSFER PARAMETER: HC1
C *   HEAT RESISTANCE IN FUEL + CANNING
      ZUGCA=(((-.316239E-7*T1+.137226E-4)*T1-.165183E-2)*T1
     # +4.85748)/1.4
C *   HEAT TRANSFER AT SURFACE BY DITTUS-BOELTER AND THOM
C *   COMBINED IN ONE COEFFICIENT
      H2=(HC2*(AMAX1(0.,T2-TCS))**2)/(T2-T3)                          851122
      H=H2+HC1
C *   TOTAL HEAT RESISTANCE FROM FUEL TO COOLANT
      Z=ZUGCA+1./H
C *   HEAT FLOW FROM FUEL TO COOLANT
      QCA=(T1-T3)/Z
C *   DIVIDED IN TWO TERMS: FOR HEATING AND FOR SURFACE BOILING
      QC1=HC1*(T2-T3)
      QC2=AMAX1(0.,QCA-QC1)
C *   FUEL TEMP.WITH TEMP.DEPENDENT HEAT CAPACITY
      CPU=((-.294195E-3*T1+.287807)*T1+231.398)*1.E-6
      CUCA=CCA+CPU*VU*ROU
      TUP(J)=(NPD(J)*(1.-YQBP-YQWC)-QCA)/CUCA
C *   CANNING TEMP. CALCULATED ALGEBRAIC USING A DIGITAL FILTER
      TCAN=T3+QCA/H
      TCA(J)=(TCAN+(CAT-1.)*TCA(J))/CAT
C *   TOTAL HEAT FLOW TO WATER (INCL.DIRECT HEAT): QC
      QC=QCA+NPD(J)*YQWC
C *   STORE POWER TERMS FOR LATER USE
      Q(J)=QC
      Q2(J)=QC2
C *   DELAYED NEUTRON DENSITY: CN1
      X=NYSF(J)*PHI(J+1)
      CN1P(J)=BETA1*X*1.E-10-LM1*CN1(J)
C *   STEADY STATE XENON CALCULATION.
C *   IF XENON > 0 IT GIVES THE PERCENTAGE POWER LEVEL FOR XENON.
C *   THE FULL POWER AXIAL DISTRIBUTION IS USED.
```

```
      IF(STI.GT.0.) GOTO 210
      IF(XENON.GT.0.) THEN
         FI=4.17E12/FCAL*XENON/5.*XQRE(J)*10.
         X=NYSF(J)*FI
        ELSE
         FI=PHI(J+1)
      ENDIF
      XEN(J)=((GMI+GMX)*X/NY/(LMX+SGMX*FI)+23.*XEN(J))/24.
210   CONTINUE
C
C *   HYDRAULICS:
      IF(WFE.LT.0.) WFE=0.
      SALF=0.
      DP=0.
      TCM=0.
C *   VOID AND TEMP. PROFILE IN CORE:
      WG1=0.
      WF1=WC
      TCI=TLP2
      DO 250 J=1,NS-2
      A2=AMAX1(ALF(J),EPS)
      T3=TC(J)
C *   SLIP FACTOR FOR NR<=1: S
      IF(NR.LE.1) S(J)=BANK(PR,A2)
      B2=S(J)*A2/(1.-A2)
C *   SURFACE BOILING
      WSF=Q2(J)/(HFG+CP*(TCS-T3)/C0)
      CK=4.*A2*(1.-A2)
      IF(CK.GT..25) CK=SQRT(CK/4.)
      CC=CCCC*CK
C *   BULK BOILING
      IF(T3.GT.TCS) WBB=200.E5*CC*(T3-TCS)**2
C *   STEAM CONDENSATION
      IF(T3.LE.TCS) THEN
        IF(T3.GT.110.) WBB=-1.E7*CC*C0*(TCS-T3)**2
        IF(T3.LE.110.) WBB=-1.E7*CC*C0*AMIN1(TCS-T3,20.)
      ENDIF
C *   TOTAL EVAPORATION: WE
      WE=WSF+WBB
C *   WATER FLOW OUT FROM SECTION: WF2
      WF2=(WF1+WG1/C0+WE*C1-PP*VC*(A2*C2+(1.-A2)*DRFS))/(1.+B2)
C *   STEAM FLOW OUT FROM SECTION: WG2
      WG2=WF2*C0*B2
C *   COOLANT TEMP.: TC
      TCP(J)=(Q(J)-(WF1+WG1)*CP*(T3-TCI)-WE*HFG)/
     # (VC*(RFS*(1.-A2)+RGS*A2)*CP)
C *   STEAM VOID: ALF
      ALFP(J)=(WF2-WF1+WE)/(VC*RFS)+PP*(1.-A2)*DRFS/RFS
      IF(ALF(J).LE.EPS.AND.ALFP(J).LT.0.) ALFP(J)=EPS-ALF(J)
C *   STEAM QUALITY: SQ
      SQ=WG2/(WG2+WF2)
C *   FRICTION TERM: DP
      DP=DP+(1.+2400.*SQ/PR)*((WG2+WF2)/(RFS*AC))**2
      SALF=SALF+A2
      TCM=TCM+T3
      WF1=WF2
      WG1=WG2
      TCI=T3
250   CONTINUE
C *   STEAM AND WATER FLOW OUT FROM CORE: WG0,WF0
      WG0=WG2
      WF0=WF2
C *   MEAN CORE VOID AND TEMP.
      MVOID=SALF/(NS-2)
      TCM=TCM/(NS-2)
C
C *   RISER:
      ALR=AMAX1(ALFR,EPS)
      CK=4.*ALR*(1.-ALR)
      IF(CK.GT..25) CK=SQRT(CK/4.)
      CC=2.E11*C0*CCC*VR*CK
```

```
C *   STEAM FLASHING BY DECREASING PRESSURE: -DWR
      IF(TR.GT.TCS) DWR=-CC*(TR-TCS)**2
      IF(TR.LE.TCS) THEN
C *     STEAM CONDENSATION DUE TO COOLING BY THE BYPASS FLOW: DWR
        IF(TR.GT.110.) DWR=CC*(TCS-TR)**2
        IF(TR.LE.110.) DWR=CC*AMIN1(TCS-TR,20.)
      ENDIF
C *   WATER TEMP.: TR
      TRP=(WBP*(TBP2-TR)+WFO*(TC(NS-2)-TR)+WGO*(TC(NS-2)-TCS)
     # +(DWR*HFG-PP*VR*ALR*RGS*DHGS)/CP)/(VR*(1.-ALR)*RFS)
C *   STEAM AND WATER FLOW CORRECTIONS: DWR
      WG=WGO-DWR
      WF=WFO+DWR+WBP
C *   SLIP FACTOR: SR
      IF(NR.LE.1) SR=BANK(PR,ALR)
      BR=SR*ALR/(1.-ALR)
C *   WATER AND STEAM FLOW OUT FROM RISER: WFR,WGR
      WFR=(WG/CO+WF-PP*VR*(ALR*C2+(1.-ALR)*DRFS))/(1.+BR)
      WGR=WFR*CO*BR
      WR=WGR+WFR
C *   STEAM VOID: ALFR
      ALFRP=(WG-WGR)/(VR*RGS)-PP*ALR*DRGS/RGS
      IF(ALFR.LE.EPS.AND.ALFRP.LT.0.) ALFRP=EPS-ALFR
C *   STEAM QUALITY: SQR
      SQR=WGR/WR
C
C *   SEPERATOR AND STEAM VOLUME:
C *   FRICTION PRESSURE DROP: DPSP
      DPSP=KSF*(1.+2400.*SQR/PR)*(WR/(AS*RFS))**2*RFS
C *   PRESSURE IN RISER FOLLOWS STEAM PRESSURE BY 0.1 SEC.TIME CONST.
      PRP=(PE-PR+(DPSP+G*LS*(RFS-ALR*(RFS-RGS)))*1.E-5)/.2
C *   PRESSURE RATE FOR HYDRAULIC FEEDBACK: PP
      PPP=(PRP-PP)/.2
C *   STEAM-WATER MASS EXCHANGE
      DWSP=WGR*(PR-PE)*DHGS/HFG
      IF(TR.GT.TES) DWSP=DWSP+WFR*(TR-TES)*CP/HFG
C *   STEAM RELEASE FROM TOP WATER VOLUME BY DECREASING PRESSURE: DWGT
      DWGT=0.
      IF(TT.GT.TES) DWGT=VT*1.*(TT-TES)**2
C *   STEAM CONDENSATION AT THE WATER SURFASE: DWSC
      PFU=ALIM(2.*(PE-1.5),1.,0.)
      DWSC=.002*AT*AMAX1(TES-TT,0.)*PFU
C *   STEAM CONDENSATION AT THE TANK WALL SURFASE: DWWC
      QW1=KW1*AMAX1(TES-TTW1,0.)*PFU
      DWWC=QW1/HFG
C *   TOTAL STEAM PRODUCTION: WGE
      WGE=WGR+DWSP+DWGT-DWSC-DWWC
C *   STEAM PRESSURE: PE
      PEP=(WGE-WLR)/(VE*DRGS)
C *   PRESSURE IS LIMITED DOWNWARDS TO 1.5 BAR
      IF(PE.LE.1.5.AND.PEP.LT.0.) PEP=1.5-PE
C
C *   TOP WATER VOLUME AND MIXING VOLUME
C *   WATER FLOW FROM SEPERATOR: WT
      WT=WFR-DWSP
C *   TEMP.IN TOP VOLUME: TT
      TTP=(WT*(AMIN1(TES,TR)-TT)-(DWGT-DWSC)*HFG/CP+(DWSC+DWWC)         851129
     #*(TES-TT))/(VT*RFS)                                               851129
C *   FLOW IN DOWNCOMER = FLOW OUT OF MIXING VOLUME : WD
      WD=WRC+WREK+WLOS
C *   TEMP.IN MIXING VOLUME: TB
      TBP=((WD-WFE-WREK)*TT*CP+WFE*TFE*CPFE+WREK*TREK*CPRE
     # -WD*TB*CP)/(VB*RFS*CP)
C
C *   DOWNCOMER TEMP.: TDC
C *   HEAT FLOW TO TANK WALL: QW2
      QW2=KW2*(TDC-TTW2)
      TDCP=(WD*(TB-TDC)-QW2/CP)/(VDC*RFS)
C
C *   RECIRCULATION WITH 1 HC-PUMP LOOP
C *   DRIVING FORCE FROM VOID AND WATER HEADS AND FRICTION IN TANK
```

```
      DC=G*(RFS-RGS)*DX*SALF
      DPC=DP*DPCK*RFS
      DPD=DPDK*WD**2
C *   STATIC PRESSURE DROP OVER TANK INLET-OUTLET: DPRE
      DPRE=(PE-PR)*1.E5+DC+G*WLEV*RFS-DPC-DPD
      VRD=WD/(RFS*4)
      VRC=WRC/(RFS*4)
C *   PUMP PRESSURE AND FLOW RATE : DPU,WRC
      DPU=ARCP*ORP**2+BRCP*ORP*VRC+CRCP*VRC**2
      DPL=CDPL*(.5*WRC**2+.5*WD**2)
      WRCP=(DPU*1.E5+DPRE-DPL)/TAUW
C *   INLET TEMP.TO PUMPS : TPUI
C *   INLET TEMP.TO REACTOR : TRCI
C *   BYPASS TIME DELAYS AT CASE 104 AND BY STEADY STATE CALCULATIONS
C *   AND INSTEAD ADD TUBE VOLUME TO PUMP VOLUME.
        IF(STI.EQ.104..OR.STI.LE.0.) THEN
          TPUI=TDC
          TRCI=TPU
          VPU=26.24
        ELSE
          YHC1P=VRD
          TAU1=TRNSTM(LTAU1,YHC1,YHC1-.5*VHC)
          TPUI=DEADTM(TTAU1,TDC,TAU1)
          YHC2P=VRC
          TAU2=TRNSTM(LTAU2,YHC2,YHC2-.5*VHC)
          TRCI=DEADTM(TTAU2,TPU,TAU2)
      ENDIF
C *   PUMP HEAT: QPU
      QPU=.55+6.4E-4*(TPU-20.)
C *   TEMP IN PUMPS : TPU
      TPUP=(WRC*(TPUI-TPU)+QPU/CP)/(VPU*RFS)
C
C *   TEMP IN LOWER PLENUM 1: TLP1
      TLP1P=(TRCI-TLP1)*WRC/(VLP1*RFS)
C *   TEMP.IN LOWER PLENUM 2: TLP2
      TLP2P=WC*(TLP1-TLP2)/(VLP2*RFS)
C *   TEMP.IN CORE BYPAS 1 AND 2: TBP1,TBP2
      TBP1P=WBP*(TLP1-TBP1)/(VBP1*RFS)
      TBP2P=(QN*YQBP/CP-WBP*(TBP2-TBP1))/(VBP2*RFS)
C
C *   TEMP.OF TWO TANK WALL SECTIONS
C *   HEAT FLOWS FROM INSIDE: QW1,QW2
C *   -    -     TO OUTSIDE : QA1,QA2
      QA1=HW1*(TTW1-20.)
      QA2=HW2*(TTW2-20.)
      TTW1P=(QW1-QA1)/CW1
      TTW2P=(QW2-QA2)/CW2
C
C *   WATER LEVEL: WLEV
      TPVDR=DRFT*(TTP*VT+TDCP*VDC+TLP1P*VLP1+TLP2P*VLP2+TBP1P*VBP1
     # +TBP2P*VBP2)
C *   THE THERMAL EXPANSION TERMS ARE NEGLECTED AT STEADY STATE
      IF(STI.LE.0.) TPVDR=0.
      WLEVP=(WT+WFE+WREK-WD-DWGT+DWSC+DWWC-TPVDR)/(AT*RFS)
      IF(STI.GT.0.) RETURN
C
C *   STEADY STATE CALCULATIONS
      YHCP1=0.
      YHC2P=0.
      PPP=0.
      IF(WLEV.LT.4.0) WLEVP=4.1-WLEV
      IF(WLEV.GT.4.1.AND.WFE.LT.1.E-3) WLEVP=(4.1-WLEV)/30.
C *   STEADY STATE ACCELERATIONS
      TLP1P=10.*TLP1P
      TLP2P=10.*TLP2P
      TBP1P=10.*TBP1P
      TBP2P=10.*TBP2P
      TRP=10.*TRP
      TTP=10.*TTP
      TBP=10.*TBP
      TDCP=10.*TDCP
```

```
      RETURN
CCCCCC      ======
CCCCCC      ======
C *   CALCULATION OF FLUX WITH PROMT JUMP APPROXIMATION.
 1000 CONTINUE
C *   CALCULATE CONTROL ROD DENSITY IN ARRAY CCR
      DO 1020 K=1,KCCR
      ACCR(K)=(1.-PCCR(K)/100.)*(NS-2)
 1020 IF(ACCR(K).GT.(NS-2)) ACCR(K)=NS-2
      DO 1050 J=1,NS-2
      CCR(J)=0.
      DO 1040 K=1,KCCR
      IF(ACCR(K).GE.1.) CCR(J)=CCR(J)+NCCR(K)
      IF(ACCR(K).GT.0..AND.ACCR(K).LT.1.) THEN
        X=ACCR(K)-.5
        AX=PHI(J+1)                                                    851122
        BX=(PHI(J+2)-PHI(J))/2.                                        851122
        CX=(PHI(J+2)+PHI(J)-2.*PHI(J+1))/2.                            851122
        IF(ACCR(K).GT.1.E-10) THEN
          BCR=(((CX/3.*X+BX/2.)*X+AX)*X+AX/2.-BX/8.+CX/24.)/(X+.5)     851122
          ELSE
          BCR=AX-BX/2.+CX/4.                                           851122
        ENDIF
C                                                                      851122
C     * SET WEIGHT FACTOR=1 ON START                                    851122
C                                                                      851122
        IF(NR.EQ.0) THEN                                               851122
          BCR=1                                                        851122
        ELSE                                                           851122
          BCR=BCR/(AX+CX/12.)                                          851122
        ENDIF                                                          851122
        CCR(J)=CCR(J)+ACCR(K)*BCR*NCCR(K)
      ENDIF
 1040 ACCR(K)=ACCR(K)-1.
      CCR(J)=CCR(J)/MCCR
 1050 CONTINUE
 1060 CONTINUE
C
C *   D,SIGMA A,NY*SIGMA F,CM-MATRICE,SLCN
C *   FIRST SECTION BELOW ACTIVE FUEL
      R=RFS+DRFT*(TLP2-TR)
      C=CCR(1)
      D=FDMC(R)*(1.-C)+FDPC(R)*C
      SA=FSAMC(R,0.)*(1.-C)+FSAPC(R,0.)*C
      X=3.*DZ+16.*D
      CM(1,2)=-SA-(1./DZ+9./X)*D/DZ
      CM(1,3)=(1./DZ+1./X)*D/DZ
      CM(1,1)=0.
      SLCN(1)=0.
C *   ACTIVE FUEL SECTIONS
      DO 1070 J=2,NS-1
      RF=RFS+DRFT*(TC(J-1)-TR)
      R=(RF-ALF(J-1)*(RF-RGS))
      C=CCR(J-1)
      U=TU(J-1)+273.-500.
      D=FDMC(R)*(1.-C)+FDPC(R)*C
      SA=FSAMC(R,U)*(1.-C)+FSAPC(R,U)*C+XEN(J-1)*SGMX
      NSF=FNSFMC(R,U)*(1.-C)+FNSFPC(R,U)*C
      NSF=CNSF*NSF
      NYSF(J-1)=NSF
      X=D/(DZ*DZ)
      CM(J,1)=X
      CM(J,3)=X
      CM(J,2)=(1.-BETA)*NSF-SA-2.*X
      X=-LM1*CN1(J-1)*1.E10-SOUR
      SLCN(J)=X
      SLCNR(J)=X+SOUR
1070  CONTINUE
C *   LAST SECTION ABOVE ACTIVE FUEL
      R=(RFS-ALF(NS-2)*(RFS-RGS))
      D=FDMC(R)
```

```
      SA=FSAMC(R,0.)
      X=3.*DZ+16.*D
      CM(NS,2)=-SA-(1./DZ+9./X)*D/DZ
      CM(NS,1)=(1./DZ+1./X)*D/DZ
      CM(NS,3)=0.
      SLCN(NS)=0.
C
C *   SOLVE THE DIFFUSION EQUATION.
C *   NEUTRON FLUX IN ARRAY PHI
      DO 1080 J=1,NS-1
      X=CM(J+1,1)/CM(J,2)
      CM(J+1,2)=CM(J+1,2)-X*CM(J,3)
1080  SLCN(J+1)=SLCN(J+1)-X*SLCN(J)
      DO 1090 J=1,NS-1
      X=SLCN(NS+1-J)/CM(NS+1-J,2)
      PHI(NS+1-J)=X
1090  SLCN(NS-J)=SLCN(NS-J)-X*CM(NS-J,3)
      PHI(1)=SLCN(1)/CM(1,2)
C
C *   NUCLEAR POWER IN MW IN ARRAY NP
C *   IQN=-1 INDICATES A NEGATIVE FLUX VALUE
      QN=0.
      FLUX=0.
      IQN=1
      DO 1100 J=1,NS-2
      NP(J)=PHI(J+1)*NYSF(J)*AFP
      IF(NP(J).LT.0.) IQN=-1
      QN=QN+NP(J)
      FLUX=FLUX+PHI(J+1)
 1100 CONTINUE
      FLUX=FLUX/(NS-2)*FCAL
      IF(STI.GE.0.) GOTO 1200
C
C *   STEADY STATE ITERATIONS
C *   IMERGENCY RECOVERY FOR NEG.FLUX VALUES WILL CHANGE CNSF
      IF(IQN.GT.0) GOTO 1120
      CNSF=CNSF-.001
      GOTO 1060
 1120 CONTINUE
      IF(STI.LT.-1.) GOTO 1200
C *   STI=-1, ITERATION ON POWER WITH CNSF
      IF(NR.LE.1) CNSF=CNSF-CQN*(QN-QSR)/QSR
      IF(NR.EQ.0.AND.ABS(QN-QSR).GT.QSR/100.) GOTO 1060
 1200 CONTINUE
C
C *   REACTIVITY IN PCM  (RTVT)
      ALFA=0.
      ALFA0=0.
      ALFA1=0.
      DO 1210 J=2,NS-1
      FI=PHI(J)
      ALFA=ALFA+FI*SLCNR(J)
      ALFA1=ALFA1+FI*(SLCNR(J)-SOUR)
 1210 ALFA0=ALFA0+FI*NYSF(J-1)*FI
      RTVT=(BETA+ALFA/ALFA0)*1.E+5
      REACTI=(BETA+ALFA1/ALFA0)*1.E+5
C *   REACTIVITY FILTER:  1/(1+TAUR*S)
      RTVTFP=(RTVT-RTVTF)/TAUR
C
C *   DOUBLING TIME: DBT
      RVT=1.E-5*AMAX1(RTVTF,.01)
      DBT=.69315*(BETA1-RVT)/(LM1*RVT)
      GOTO 201
CCCCCC      ======
CCCCCC      ======
C *   PROFILE OUTPUT TABLE:
      ENTRY YOREAC
      WRITE (6,3100) PHI,XEN,NPD,CCR,TU,TCA,TC,ALF,
     # PR,PE,TCS,TES,TR,TT,TB,TRCI,TBP2,
     # QND,WRC,WR,WC,WGO,WGR,WGE,WLR,WFE,ALFR,SQR,TR,WLEV,DPU,FLUX,
     # CNSF,DC,DPC,DPD,DPL
```

```
3100  FORMAT (1H1,T48,'R E A C T O R',
     # /X,        T48,'=============',
     # //X,'FLUX:',/X,1P,12E11.3,
     # //X,'XENON:',/X,11X,10E11.3,0P,
     # //X,'THERMAL CORE POWER:',/X,11X,10G11.4,
     # //X,'CONTROL ROD DENSITY:',/X,11X,10F11.4,
     # //X,'FUEL TEMP:',/X,11X,10F11.2,
     # //X,'CANNING TEMP.:',/X,11X,10F11.2,
     # //X,'COOLANT TEMP.:',/X,11X,10F11.2,
     # //X,'VOID FRACTION:',/X,11X,10F11.4,
     # //X,'PRESSURE IN BAR: PR,PE:',T48,2F8.2,
     # //X,'TEMPERATURE: TCS,TES,TR,TT,TB,TRCI,TBP2:',T48,7F8.2,
     # //X,'NUCLEAR POWER WITHOUT RESIDUAL HEAT:',T48,G11.4,
     # //X,'FLOWS: WRC,WR,WC,WGO,WGR,WGE,WLR,WFE:',T48,3F8.1,5F8.2,
     # //X,'RISER: VOID,STEAM QUALITY,TEMP.:',T48,2F8.4,F8.2,
     # //X,'WATER LEVEL:',T48,F8.4,
     # //X,'PUMP PRESSURE IN BAR:',T48,F8.3,
     # //X,'MEAN NEUTRON FLUX:',T48,G11.4,
     # //X,'CNSF:',E14.7,
     # //X,'DC,DPC,DPD,DPL:',4E12.5)
      RETURN
      END
C
*********************************************************************
C
      SUBROUTINE CROD(NR,TID)
C *   ROUTINE FOR CONTROL ROD POSITIONING WITH STEADY STATE ITERATIONS
C *   AND SIMULATION OF OPERATORS ROD CONTROL.
C
C *   COMMON /CCROD/ CONTAINS:
C *   S: STATE VARIABLES    : 3
C *   D: DERIVATIVES        : 3
C *   X: INPUT VARIABLES    : 12
C *   C: INPUT CONSTANTS    : 12
C *   A: ALGEBRAIC VARIABLES: 30
C
      COMMON /CCROD/
     S XCR,TREF,XQ,
     D XCRP,TREFP,XQP,
     X QN,PE,WGE,SS,CRS,DBT,
     X RT,XDU(5),
     C STI,QSR,WLSR,CQCR,CPCR,NGR,
     C GRP,FCR,VCR,TG1,TG2,PTG,
     A PCR(24),SUMCR,PREF,PER
C
      COMMON /GRAFI/ GRMODE,CH2,CH,STEPREK
      CHARACTER CH2
C
      DIMENSION NCR(24),PCRS(24),NTRI(12)
      REAL NGR
      DATA NCR / 3*1, 3*2, 11*4, 7*8 /
      DATA VMAX / .41666667 /
      DATA TSCRAM,VSCRAM,TAUS / -1., 66.67, 1.5 /
      DATA NTRI / 0, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110,
     #  111 /
C
      SAVE
C
      ALIM(X,XMAX,XMIN)=AMAX1(XMIN,AMIN1(X,XMAX))
C
C *   INITIAL RESETTINGS
      IF(NR.EQ.0) TSCRAM=-1.
C
      IF(STI.GT.0.) GOTO 1000
C
C *   STEADY STATE ITERATIONS
C *   -----------------------
      IF(STI.EQ.0.) GOTO 30
      IF(NR.EQ.0) THEN
C *     FIND FIRST GROUP WITH 4 RODS NOT FULLY INSERTED
        MCR=0
```

```
        DO 5 J=7,17
 5      IF(PCR(J).GT.0.) GOTO 6
        GOTO 8
 6      MCR=J
      ENDIF
C
 8    CONTINUE
      IF(NR.GT.1) RETURN
      IF(MCR.EQ.0) RETURN
      GOTO (1,10,100),NINT(-STI)
C
C *   STI=-1, ITERATION IN REAC
 1    GOTO 30
C
C *   STI=-2, ITERATION ON QN.
 10   CONTINUE
C *   ADJUST THE CR-POSITIONS ACCORDING TO THE POWER ERROR
C *   SHIFT GROUP NUMBER WHEN A CR-LIMIT IS REACHED
      DCR=CQCR*(QSR-QN)/QSR*109/NCR(MCR)
 20   CONTINUE
      DCR=ALIM(DCR,1.,-1.)
      E=PCR(MCR)
      PCR(MCR)=ALIM(E+DCR,100.,0.)
C *   CHECK IF GROUP NO.MCR HAS REACHED A LIMIT , THEN ADD +-1
      IF(E+DCR.GE.100.) MCR=MCR-1
      IF(E+DCR.LE.0.)   MCR=MCR+1
C
 30   CONTINUE
C *   STATE VARIABLES XCR=0 , TREF=RT AND XQ=QN
      XCRP=-XCR
      TREFP=RT-TREF
      XQP=QN-XQ
      GOTO 3010
C
C *   STI=-3, ITERATION ON WGE.
 100  CONTINUE
      IF(NR.EQ.0) QNR=QSR
C *   CORRECT QNR FOR DEVIATIONS IN WGE
      QNR=QNR+CPCR*(WLSR-WGE)
      DCR=CQCR*(QNR-QN)*109/NCR(MCR)
      GOTO 20
C
C
C
C *   TRANSIENTS
C *   ----------
 1000 CONTINUE
      IF(SS.EQ.0.) GOTO 1050
C *   CALCULATE CR-POSITIONS AT SCRAM
      IF(TSCRAM.LT.0.) THEN
        TSCRAM=TID
        DO 1010 J=1,24
 1010   PCRS(J)=PCR(J)
      ENDIF
      DPCR=AMAX1(VSCRAM*(TID-TSCRAM-TAUS),0.)
      DO 1020 J=1,24
 1020 PCR(J)=AMAX1(PCRS(J)-DPCR,0.)
      GOTO 3010
C
 1050 CONTINUE
C *   RESET PRESSURE SETPOINT ONLY USED IN CASE 104
      PER=0.
C *   TRANSIENT TASK DISTRIBUTION
      DO 1060 J=1,12
      IF(NINT(STI).EQ.NTRI(J)) GOTO 1070
 1060 CONTINUE
      GOTO 3000
 1070 GOTO(3010,1100,1200,1300,1400,1500,1600,1700,1800,1900,2000,
     # 2100),J
C
 1100 CONTINUE
```

```
C *   TRANSIENT CASE 101
      GOTO 1200
C
 1200 CONTINUE
C *   TRANSIENT CASE 102
C *   WITHDRAW GROUP NO. GRP FCR % WITH THE RELATIVE SPEED VCR
      IF(GRP.EQ.0.) GOTO 3000
      IF(NR.EQ.0) PCR0=PCR(NINT(GRP))
      IF (CH2.EQ.':') THEN
        YCR=0
        PCR0=PCR(NINT(GRP))
      ENDIF
      XCRP=VMAX*VCR/100.
      YCR=SIGN(AMIN1(XCR,ABS(FCR)),FCR)
      PCR(NINT(GRP))=ALIM(PCR0+YCR,100.,0.)
      GOTO 3000
C
 1300 CONTINUE
C *   TRANSIENT CASE 103
      GOTO 3000
C
 1400 CONTINUE
C *   TRANSIENT CASE 104
C *   WITHDRAW RODS WITH MAX.DOUBLING TIME EQUAL TO 30 S.
C *   SIMULATE AN OPERATOR KEEPING  RT CLOSE TO TREF WITH A SMOOTH
C *   VARIATION OF POWER. A PI-CONTROLLER GIVES A POWER SETPOINT
C *   WITH TREF-RT AS INPUT. THE POWER ERROR CONTROLS THE ROD SPEED.
C *   THE CR-GROUPS ARE WITHDRAWN WITH DECREASING GROUP NUMBERS
C *   STARTING WITH GROUP NO.GRP.
C *   START TEMP.PROGRAM AT Q=5 MW.
C *   START PRESSURE PROGRAM AT PE=5 BAR.
C *   REVERSE THE PROGRAMMER FOR PREF > PE+1 UNTIL PREF <= PE.
C *   P-SETPOINT FOR PRESSURE CONTROL SYSTEM = MAX(PREF,8).
      IF(GRP.EQ.0.) GOTO 3000
      IF(NR.EQ.0) THEN
        MCR=NINT(GRP)
        JQN=0
        TSA70=TSAF(70.)
      ENDIF
      XCRP=0.
      IF(QN.LT.5.AND.JQN.EQ.0) THEN
        TREFP=RT-TREF
        XQP=QN-XQ
        IF(DBT.GT.30.) XCRP=VMAX
      ELSE
        JQN=1
        ETT=TREF-RT
        XQP=ETT/60.
        QREF=3.*ETT+XQ
        XCRP=ALIM(4.*(QREF-QN)/QREF,1.,-1.)*VMAX
        IF(PE.LT.5.) THEN
          SIG=1.
        ELSE
          PREF=PSATF(TREF)
          IF(PREF.GE.PE+1.) SIG=-1.
          IF(PREF.LE.PE)    SIG= 1.
          IF(TREF.GE.TSA70)   SIG= 0.
        ENDIF
        IF(PE.LT.PTG) TREFP=SIG*TG1/3600.
        IF(PE.GE.PTG) TREFP=SIG*TG2/3600.
      ENDIF
      PER=AMAX1(PREF,8.)
C *   MOVE CONTROL RODS BEGINNING WITH NO.GRP AS THE HIGHEST GROUP
      IF(NR.GT.1) GOTO 3000
 1450 DCR=XCR-(GRP-MCR)*100.
      PCR(MCR)=ALIM(DCR,100.,0.)
      IF(DCR.GE.100.) THEN                                          851129
        MCR=MCR-1                                                   851129
        GOTO 1450                                                   851129
      ENDIF                                                         851129
      GOTO 3000
```

```
C
 1500 CONTINUE
C *   TRANSIENT CASE 105
      GOTO 1200
C
 1600 CONTINUE
C *   TRANSIENT CASE 106
      GOTO 3000
C
 1700 CONTINUE
C *   TRANSIENT CASE 107
      GOTO 1200
C
 1800 CONTINUE
C *   TRANSIENT CASE 108
      GOTO 1200
C
 1900 CONTINUE
C *   TRANSIENT CASE 109
      GOTO 1200
C
 2000 CONTINUE
C *   TRANSIENT CASE 110 ..... .... .... ...
C *   WITHDRAW NGR CR-GROUPS WITH 4 RODS IN EACH GROUP.
C *   BEGIN WITH GROUP NO.GRP WITH INCREASING GROUP NUMBER.
      IF(GRP.EQ.0.) GOTO 3000
      IF(NR.EQ.0) MCR=NINT(GRP)
      IF(MCR.GE.NINT(GRP+NGR)) GOTO 3000
      XCRP=VMAX*VCR/100.
      DCR=XCR-(MCR-GRP)*100.                                          851122
      PCR(MCR)=AMIN1(DCR,100.)
      IF(DCR.GE.100.) MCR=MCR+1
      GOTO 3000
C
 2100 CONTINUE
C *   TRANSIENT CASE 111
      GOTO 1200
C
 3000 CONTINUE
C *   STOP ROD WITHDRAWING IF CRS > 0.
C *   XCRP > 0 MOVES RODS OUT OF CORE
      IF(CRS.GT.0..AND.XCRP.GT.0.) XCRP=0.
C
 3010 CONTINUE
C *   CALCULATE SUMMA PROCENT RODS OUT
      SUMCR=0.
      DO 3020 J=1,24
 3020 SUMCR=SUMCR+PCR(J)*NCR(J)
      RETURN
C
C
      ENTRY YOCROD
      WRITE(6,4000) (NCR(J),J=1,12) ,(PCR(J),J=1,12),
     #              (NCR(J),J=13,24),(PCR(J),J=13,24)
 4000 FORMAT(//X,T48,'C O N T R O L   R O D S',
     #        /X,T48,'======================',
     # //X,'NUMBER OF RODS IN GROUP    1-12:',12I8,
     # /X,'POSITION OF RODS IN GROUP  1-12:',12F8.2,
     # //X,'NUMBER OF RODS IN GROUP   13-24:',12I8,
     # /X,'POSITION OF RODS IN GROUP 13-24:',12F8.2)
      RETURN
      END
C
*********************************************************************
C
      SUBROUTINE FWCON(NR,TID)
C *   FEEDWATER CONTROL BY THE AUXILIARY SYSTEM
C
C     COMMON /CFWCON/ CONTAINS:
C     S: STATE VARIABELS    : 6
C     D: DERIVATIVES        : 6
```

```
C     X: INPUT VARIABELS    : 6
C     C: INPUT CONSTANTS    : 6
C     A: ALGEBRAIC VARIABELS: 0
C
      COMMON /CFWCON/
     S WFE,XH(3),V42,V23,
     D WFEP,XHP(3),V42P,V23P,
     X WLEV,PE,WLR,SS,ASKH,P12,
     C GL,GW,WFER

      COMMON /CSPEC/ SPEC
C
      REAL YH(6)
      REAL KV4,KV42,KV23,KHF,SPEC
C
      DATA WLERO,TC1,TV42,TV23 /4.1, 10., 30., 45./
      DATA KV4,KV42,KV23,KHF /5.63E-3, 5.63E-5, 7.E-4, 8.E-4/
      DATA A34,B34,C34 /-.101853E-2,.325180E-2,25.7480/
      DATA A12,B12,C12 /-.394141E-2,.647404E-1,76.3186/
      DATA WLEVS,SST / 0., -1. /
C
      SAVE
C
      ALIM(X,XMAX,XMIN)=AMAX1(XMIN,AMIN1(X,XMAX))
C
      IF(NR.EQ.0) THEN
C *     INITIALIZATIONS AT START
        SST=-1.
        WLEVS=0.
      ENDIF
C
      IF(SS.GT.0.) THEN
C *   CHANGE OF WATER LEVEL SETPOINT BY SCRAM
C *     SST: EVENT TIME
        IF(SST.LT.0.) SST=TID
        WLEVS=-.4
        IF(TID-SST.GE.600.) WLEVS=-.4*AMAX1(0.,1.-(TID-SST-600.)/120.)
      ENDIF
C
C *   AUXILIARY FEEDWATER FLOW CALCULATION
C *   YH1: LEVEL SIGNAL
      YH(1)=ALIM((WLEV-2.5)/3.5,1.,0.)
C *   XH1: VARIABLE IN COMPENSATION CIRCUIT
      XHP(1)=(YH(1)-XH(1))/TC1
C *   YH2: COMPENSATED LEVEL SIGNAL
      YH(2)=ALIM(YH(1)-XH(1),.05,-.05)+YH(1)
C *   YH3: LEVEL ERROR SIGNAL
*!NB! YH3 IS MODIFIED WITH AN EXTRA FEEDBACK SIGNAL (WFE-WLR)
*     REPRESENTING THE FLOW ERROR
*     GL IS THE LEVEL ERROR GAIN BEFORE THE CONTROLLER
*     GW - -   FLOW  -     -    -      -   -
      YH(3)=GL*((WLERO+WLEVS-2.5)/3.5-YH(2))-GW*(WFE-WLR)/100.
C *   PI-CONTROLLER. INPUT YH3, OUTPUT YH4
      XHP(2)=20.*YH(3)/60.
      IF(XH(2).LE.0..AND.XHP(2).LT.0.) XHP(2)=0.
      IF(XH(2).GE..72.AND.XHP(2).GT.0.) XHP(2)=0.
      XH(2)=ALIM(XH(2),.72,.00)
      YH(4)=ALIM(XH(2)+20.*YH(3),.72,.00)
      IF(WFER.GT.0.) YH(4)=AMIN1(WFER/100.,.72)
C *   CONTROL VALVE SERVO (V4).  XH3:VALVE POSITION
C *   : AUXILIARY FEEDWATER FLOW
      WFES=WFE/100.
      YH(5)=ALIM(20.*(YH(4)-WFES),1.,-1.)
      XH(3)=ALIM(XH(3),1.,1.E-10)
      XHP(3)=YH(5)/30.
      IF(XH(3).LE.1.E-10.AND.XHP(3).LT.0.) XHP(3)=0.
      IF(XH(3).GT.1..AND.XHP(3).GT.0.) XHP(3)=0.
C *   BYPASS VALVE OPENING (V42), ON-OFF CONTROL BY WATER LEVEL
      V42=ALIM(V42,1.,1.E-10)
      V42P=0.
      IF(WLEV.LT.2.9.AND.V42.LT.1.) V42P=1./TV42
```

```
      IF(WLEV.GT.4.2.AND.V42.GT.1.E-10) V42P=-1./TV42
C *   CONTAINMENT VALVE (V2,V3) CONTROLLED BY INPUT ASKH
      V23=ALIM(V23,1.,1.E-10)
      V23P=0.
      IF(ASKH.GT.0..AND.V23.LT.1.) V23P=1./TV23
      IF(ASKH.EQ.0..AND.V23.GT.1.E-10) V23P=-1./TV23
C *   FRICTION COEFF.FOR V4 AND V42 INDIVIDUALLY
      HV4=KV4/XH(3)**2
      HV42=KV42/V42**2
C *   FRICTION COEFF.FOR V4 PARALLEL WITH V42
      HV442=HV4*HV42/(SQRT(HV4)+SQRT(HV42))**2
C *   FRICTION COEFF.FOR V2/V3
      HV23=KV23/V23**2
C *   FLOW CALCULATION WITH PUMPS P3/P4 + P1/P2 PUMPING
C *   AGAINST PE+3 WITH FRICTION COEFF. HV442,HV23 AND KHF
      A=A34+P12*A12-(HV442+HV23+KHF)
      B=B34+P12*B12
      C=C34+P12*C12-PE-3.
CRM   BACK SHOULD NOT BE LESS THEN ZERO , ERROR WHEN SQRT
      BACK=B*B-4.*A*C
      IF (BACK.LT.0.) BACK=0
      W=(-B-SQRT(BACK))/(2.*A)+SPEC
C *   CALCULATED FLOW IS FILTERED WITH .2 S TIME LAG
      WFEP=(W-WFE)/.20
      RETURN
      END
C
*********************************************************************
C
      SUBROUTINE PECON(NR,TID)
C *   MODEL OF STEAM PRESSURE CONTROL SYSTEM.
C
C *   COMMON /CPECON/ CONTAINS:
C *   S: STATE VARIABLES     : 9
C *   D: DERIVATIVES         : 9
C *   X: INPUT VARIABLES     : 12
C *   C: INPUT CONSTANTS     : 6
C *   A: ALGEBRAIC VARIABLES : 0
C
      COMMON /CPECON/
     S XG(3),XH,XBPV,XBL(2),WSL,PES,
     D XGP(3),XHP,XBPVP,XBLP(2),WSLP,PESP,
     X PE,QN,PER,XSKD,DUMP,A314,
     X I314,WLX,XDU(4),
     C PRATE
C
      REAL YG(3),BV1(15),BV2(15),AVX(14),AVY(14)
      REAL I314
C
C *   BYPASS VALVE FEEDBACK FUNCTION
      DATA BV1 /.00,.10,.16,.18,.22,.30,.38,.42,.45,.48,.52,.56,.60,
     # .70, 1.0/
      DATA BV2 /.00,.051,.085,.121,.223,.467,.718,.805,.856,.897,
     # .933,.959,.974,.985, 1.0/
C *   BYPASS VALVE CHARACTERISTIC
      DATA AVX /.000,.200,.250,.300,.350,.500,.550,.600,
     # .650,.700,.750,.800,.900,1.00/
      DATA AVY /.000,.055,.110,.195,.295,.635,.740,.825,
     # .883,.927,.955,.975,.990, 1.00/
C     DATA QN0 / 1700. /                                              860117
      DATA TVA,NVA / -1., 0 /
C
      SAVE
C
C *   FUNCTION ALIM LIMITS X TO XMIN--XMAX
      ALIM(X,XMAX,XMIN)=AMAX1(XMIN,AMIN1(X,XMAX))
C
      IF(NR.EQ.0) THEN
        TVA=-1.
        NVA=0
      ENDIF
```

```
C
C *   PRESSURE SETPOINT CALCULATION. PES FOLLOWS PER WITH PRATE
      PESP=ALIM(PER-PES,PRATE,-PRATE)
C
C *   COARSE CONTROLLER AND DUMP VALVE
C *   XG: STATE VARIABLES FOR COARSE CONTROLLER
C *   YG: ALGEBRAIC  -     -    -        -
      YG(1)=ALIM((PE-PES)/90.,(90.-PES)/90.,-PES/90.)
      XG(1)=ALIM(XG(1),.5,.0)
      XGP(1)=7.*YG(1)/10.
      IF(XG(1).LE.0..AND.XGP(1).LT.0.) XGP(1)=0.
      IF(XG(1).GE..5.AND.XGP(1).GT.0.) XGP(1)=0.
      IF(DUMP.EQ.0.) XGP(1)=-XGP(1)
      YG(2)=ALIM(7.*YG(1)+XG(1),.5,.0)
      XGP(2)=2.25*(YG(2)-.93*XG(2)-XG(3))
      XGP(3)=XG(2)
      YG(3)=ALIM(XG(3)+.33*XG(2),.5,.0)
C
C *   CONTROLLER OUTPUT
C *   IF DUMP=0 THEN CLOSE DUMPVALVE
      Z4=YG(3)
      IF(DUMP.EQ.0.) Z4=0.
C *   HYDRAULIC DYNAMICS
      XHP=(Z4-XH)/.2                                                    860117
C
C *   DUMP VALVE SERVO
      IF(XBPV.LT.0.) XBPV=0.
C *   CALCULATE FEEDBACK
      DO 10 J=2,15
 10   IF(BV1(J).GE.XBPV) GOTO 20
 20   BFB=BV2(J-1)+(XBPV-BV1(J-1))*(BV2(J)-BV2(J-1))/(BV1(J)-BV1(J-1))
      XBPVP=(XH-BFB)/.20
      IF(XBPV.LE.0..AND.XBBVP.LT.0.) XBPVP=0.
C *   STEAM VALVE FLOW AREA
      DO 30 J=2,14
 30   IF(AVX(J).GE.XBPV) GOTO 40
 40   ABPV=AVY(J-1)+(XBPV-AVX(J-1))*(AVY(J)-AVY(J-1))/(AVX(J)-AVX(J-1))
C *   STEAM FLOW THROUGH DUMP VALVE
      AV=AMIN1(ABPV,XSKD)
      WD=12.*AV*PE
C
C *   CONTROL SYSTEM FOR BLOWDOWN VALVE
C *   PI-CONTROLLER PLUS OBJECT SERVO
      IF(A314.LT.1.) THEN
C *     NO BLOWDOWN FOR I314=0
        XBLP(1)=-XBL(1)/3.
CRM   WRONG SIGN |  CAUSED VALVE TO OPEN WHEN IT SHOULD CLOSE
        XBLP(2)=(XBL(1)-XBL(2))/10.
      IF(XBL(1).LE.0..AND.XBLP(1).LT.0.) XBLP(1)=0.
      IF(XBL(1).GE.1..AND.XBLP(1).GT.0.) XBLP(1)=0.
      ELSE
C *     BLOWDOWN ACTIVATED FOR A314>0
        EBL=(PE-PES)/90.
        XBLP(1)=EBL/3.
      IF(XBL(1).LE.0..AND.XBLP(1).LT.0.) XBLP(1)=0.
      IF(XBL(1).GE.1..AND.XBLP(1).GT.0.) XBLP(1)=0.
        XBL(1)=ALIM(XBL(1),1.,0.)
        YBL=ALIM(XBL(1)+10.*EBL,1.,0.)
        XBLP(2)=(YBL-XBL(2))/10.
      ENDIF
C
      IF(I314.EQ.0.) GOTO 50
C *   OPENING OF VALVE IN GROUP A
        NVA=2
        IF(TVA.LT.0.) TVA=TID
        GOTO 60
 50   CONTINUE
C *   CLOSING OF VALVE IN GROUP A
      IF(PE.GT.71) GOTO 60                                              860117
      IF(NVA.GT.0.AND.TID.GT.TVA+6.) THEN
        NVA=0
```

```
        TVA=-1.
      ENDIF
 60   CONTINUE
C *   BLOWDOWN STEAM FLOW
      WB=2.*.46*XBL(2)*PE+NVA*(23.+.27*(PE-70.))
C
C *   TOTAL STEAM FLOW INCLUDING INPUT TERM WLX
C *   PASSED THROUGH A FILTER WITH .2 S TIME LAG
      WSLP=(WD+WB+WLX-WSL)/.2
      RETURN
      END
C
*********************************************************************
C
      SUBROUTINE FLUXM(NR,TID)
C
C *   SIMULATION OF SRM,IRM AND PRM FLUX MONITORS
C *   AND OF SS,S,E CHAINS
C
C *   COMMON /CFLUXM/ CONTAINS:
C *   S: STATE VARIABELS    : 3
C *   D: DERIVATIVES        : 3
C *   X: INPUT VARIABELS    : 6
C *   C: INPUT CONSTANTS    : 6
C *   A: ALGEBRAIC VARIABELS: 30
C
      COMMON /CFLUXM/
     S X1,PRMF,DBTR,
     D X1P,PRMFP,DBTRP,
     X FLUX,WLEV,PE,DBT,XXX,YYY,
     C STI,ARS,TAUD,NOSS,NOS,SSD,
     A SS4,SS5,SS6,SS7,SS8,SS11,
     A E2,E3,S4,S5,S6,S7,
     A S8,S10,S11,SS,E,S,
     A PRML,SRM,IRM,PRM,IRMC,DBTF,
     A SRMLG
C
      REAL IRM,IRMC,NOSS,NOS
      LOGICAL SRMH2,SRMH1,SRML1,IRMH2,IRMH1,IRML1,IRML2,PRML1
      LOGICAL SRMON,IRMON
      PARAMETER (SQ10=3.162278)                                        851120
C
      SAVE
C
      IF(NR.GT.0) GOTO 10
C     INITIAL DETERMINATION OF IRM RANGE
      IRMC=2.
 2    CONTINUE
      PM=125.
      IF(IRMC/2.-INT(IRMC/2.).GT..25) PM=40.
      IRM=FLUX/2.16E+13*PM*SQ10**(12-INT(IRMC))
      IF(PM.EQ.125..AND.IRM.GT.109.5.OR.PM.EQ.40..AND.IRM.GT.34.6) THEN
        IRMC=IRMC+1.
        GOTO 2
      ENDIF
      SRMON=.TRUE.
      IRMON=.TRUE.
 10   CONTINUE
C
C *   SRM-SIGNAL
C *   ----------
C *   SRM-SIGNAL DE-ACTIVATION BY INPUT XXX=0 OR
C *   BY LOCAL VARIABLE SRMON=.FALSE.
C *   CALIBRATED SO PULSE RATE = FLUX/1000
      SRM=0.
C     IF(SRMON.AND.XXX.GT.0.) SRM=AMIN1(FLUX/1000.,1.E6)               860117
C     SRMLG=ALOG10(SRM)                                                860117
      SRMLG=0.                                                         860117
      IF(SRMON.AND.XXX.GT.0.) THEN                                     860117
        SRM=AMIN1(FLUX/1000.,1.E6)                                     860117
        SRMLG=ALOG10(SRM)                                              860117
```

```
      ENDIF                                                              860117
C
C *   IRM-SIGNAL
C *   ----------
C *   IRM-SIGNAL DE-ACTIVATION BY INPUT YYY=0 OR
C *   BY LOCAL VARIABLE IRMON=.FALSE.
C *   AUTOMATIC RANGE SELECTION FOR ARS > 0
      IRM=0.
      IF(.NOT.IRMON.OR.YYY.EQ.0.) GOTO 50
 20   CONTINUE
      PM=125.
      IF(IRMC/2.-INT(IRMC/2.).GT..25) PM=40.
      IRM=FLUX/2.16E+13*PM*SQ10**(12-INT(IRMC))
      IF(PM.EQ.125..AND.IRM.GT.109.5.OR.PM.EQ.40..AND.IRM.GT.34.5) THEN
C *     SHIFT TO NEXT HIGHER CHANNEL IF ARS > 0
        IF(ARS.EQ.0.) GOTO 30
        IF(IRMC.EQ.12.) GOTO 30
        IRMC=IRMC+1.
        GOTO 20
      ENDIF
 30   IRM=AMIN1(IRM,125.)
      IF(PM.EQ.125..AND.IRM.LT.31.5.OR.PM.EQ.40..AND.IRM.LT.10.5) THEN
C *     SHIFT TO CHANNEL BELOW IF ARS > 0
        IF(ARS.EQ.0.) GOTO 40
        IF(IRMC.EQ.2.) GOTO 40
        IRMC=IRMC-1.
        GOTO 20
 40     CONTINUE
      ENDIF
 50   CONTINUE
C
C *   PRM-SIGNAL
C *   ----------
C *   PRM-SIGNAL CALIBRATED TO GIVE 5 % AT FLUX=4.17E+12
      PRM=FLUX/.834E+12
C
C *   FILTER FOR PRM-SIGNAL: (1+2S)/(1+5.5S) * 1/(1+1.1S)
C *   INPUT/OUTPUT:  PRM/PRMF
      X1P=(PRM-X1)/5.5
      PRMFP=(2.*(PRM-X1)/5.5+X1-PRMF)/1.1
C
C *   FILTER FOR RECIPROCAL VALUE OF DOUBLING TIME: 1/(1+TAUD*S)
C *   INPUT/OUTPUT:  DBT/DBTR
      DBTRP=(1./DBT-DBTR)/TAUD
      DBTF=1./DBTR
C
      IF(STI.LE.0.) THEN
C *     RESETTING OF ALL LIMIT INDICATORS FOR STEADY STATE
        SS4=0.
        SS5=0.
        SS6=0.
        SS7=0.
        SS8=0.
        SS11=0.
        SS=0.
        E2=0.
        E3=0.
        IF(PRM.LT.5.) E3=1.
        E=0.
        S4=0.
        S5=0.
        S6=0.
        S7=0.
        S8=0.
        S10=0.
        S11=0.
        IF(PRM.GE.5.) S11=1.
        S=0.
        PRML=0.
        IF(PRM.GE.5.) PRML=1.
        RETURN
```

```
      ENDIF
C
C *   CHECK LIMITS FOR SRM
C *   --------------------
      SRMH2=.FALSE.
      SRMH1=.FALSE.
      SRML1=.FALSE.
      IF(XXX.GT.0.) THEN
        IF(SRM.GT.6.E+5) SRMH2=.TRUE.
        IF(SRM.GT.2.E+5) SRMH1=.TRUE.
        IF(SRM.LT.3.E+0) SRML1=.TRUE.
      ENDIF
C
C *   CHECK LIMITS FOR IRM
C *   --------------------
      IRMH2=.FALSE.
      IRMH1=.FALSE.
      IRML1=.FALSE.
      IRML2=.FALSE.
      IF(YYY.GT.0.) THEN
        IF(IRM.GT.120..OR.(IRM.GT.38..AND.PM.EQ.40.)) IRMH2=.TRUE.
        IF(IRM.GT.110..OR.(IRM.GT.35..AND.PM.EQ.40.)) IRMH1=.TRUE.
        IF(IRM.LT.5.0.OR.(IRM.LT.16..AND.PM.EQ.125.)) IRML2=.TRUE.
        IF(IRM.LT.10..OR.(IRM.LT.31..AND.PM.EQ.125.)) IRML1=.TRUE.
      ENDIF
C
C *   CHECK LIMITS FOR PRM
C *   --------------------
      PRML1=.TRUE.
      IF(PRM.GT.5..AND.PRMF.GT.5.) PRML1=.FALSE.
      PRML=0.
      IF(.NOT.PRML1) PRML=1.
C
C *   SELECTION OF E OR S-CHAIN
      IF(PRM.GT.5.5.OR.PRM.GE.5..AND.NR.EQ.0) THEN
        S11=1.
        E3=0.
      ENDIF
      IF(PRM.LT.5.) THEN
        E3=1.
        S11=0.
      ENDIF
C
C *   CHECK OF SS-CHAIN SET CONDITIONS
C *   --------------------------------
      SS4=0.
      SS5=0.
      SS6=0.
      SS7=0.
      SS8=0.
      SS11=0.
      IF(WLEV.LE.2.9) SS4=1.
      IF(WLEV.GE.4.8) SS5=1.
      IF(PE.GE.74.) SS6=1.
      IF(PRML1.AND.SRMH2.AND.IRML2) SS7=1.
      IF(IRMH2.AND.(PRML1.OR.PE.LT.60.)) SS8=1.
      IF(.NOT.PRML1.AND.PE.LT.60.) SS11=1.
      IF(SS4+SS5+SS6+SS7+SS8+SS11.GT.0.) SS=1.
C *   MANUAL SCRAM IF SSD>0
      IF(SSD.GT.0.) SS=1.
C *   SCRAM BYPASS IF NOSS>0
      IF(NOSS.GT.0.) SS=0.
C
C *   CHECK OF E-CHAIN SET CONDITIONS
C *   -------------------------------
C *   E2=0.
      E=0.
      IF(S11.GT.0.) THEN
        IF(PE.LT.60.) E2=1.
      ENDIF
      IF(E2.GT.0.) E=1.
```

```
C
C *   CHECK OF S-CHAIN SET CONDITIONS
C *   -------------------------------
C *   S-CHAIN ONLY ACTIVE FOR PRM < 5 %
      S4=0.
      S5=0.
      S6=0.
      S7=0.
      S8=0.
      S10=0.
      S=0.
      IF(E3.GT.0.) THEN
        IF(SRML1.AND.IRML2) S4=1.
        IF(DBTF.LT.25..AND.IRML2) S5=1.
        IF(YYY.EQ.0.) S6=1.
        IF(IRML2.AND.SRMH2) S7=1.
        IF(IRML1.AND.IRMC.GE.3.) S8=1.
        IF(IRMH1) S10=1.
      ENDIF
      IF(S4+S5+S6+S7+S8+S10.GT.0.) S=1.
C *   S-SIGNAL SUPPRESSION FOR NOS>0
      IF(NOS.GT.0.) S=0.
      RETURN
      END
C
*********************************************************************
C
      SUBROUTINE REKO(NR,TID)
C *   RESIDUAL HEAT REMOVAL SYSTEM SIMULATION.
C *   TWO HEAT EXCHANGERS ARE SIMULATED AS DRAIN COOLERS IN THE
C *   FULL POWER MODEL BUT WITH TUBE HEAT CAPACITY INCLUDED IN
C *   THE WATER CAPACITY.
C *   THE TUBE CONNECTIONS TO THE REACTOR TANK ARE NOT SIMULATED.
C
C *   COMMON /CREKO/ CONTAINS:
C *   S: STATE VARIABLES    : 4
C *   D: DERIVATIVES        : 4
C *   X: INPUT VARIABLES    : 6
C *   C: INPUT CONSTANTS    : 6
C *   A: ALGEBRAIC VARIABLES: 0
C
      COMMON /CREKO/
     S TRO,THO,TMC,TMH,
     D TROP,THOP,TMCP,TMHP,
     X TRI,WREK,XDU(4),
     C THI,NHEX
C
      REAL NHEX,KH,KR
C
      DATA CH,CR,CMH,CMR / 10., 5., 5., 5. /
      DATA CP /.00418/
      DATA KH,KR,BH / .628, .185, .562 /
      DATA WH,WM / 200., 100. /
C
      SAVE
C
C *   HEAT EXCHANGER COOLED BY SEAWATER
C *   SEAWATER FLOW = 200 KG/S
C *   -        TEMP. INLET:  THI
C *   -        -     OUTLET: THO
C *   HOT WATER FLOW = 100 KG/S
C *   -   -     TEMP. INLET:  TMH
C *   -   -     -     OUTLET: TMC
C *   IF NOT ACTIVE (NHEX=0) TEMPERATURES ARE KEPT AT 20 C
C
      IF(NHEX.EQ.0.) THEN
        THOP=20.-THO
        TMCP=20.-TMC
      ELSE
        THM=BH*THI+(1.-BH)*THO
        TMM=(1.-BH)*TMH+BH*TMC
```

```
        QH=KH*(TMM-THM)
        THOP=(QH-WH*CP*(THO-THI))/CH
        TMCP=(WM*CP*(TMH-TMC)-QH)/CMH
      ENDIF
C
C
C *   HEAT EXCHANGER FOR REACTOR COOLANT.
C *   REACTOR COOLANT FLOW: WREK/NHEX  (NHEX=1 OR 2)
C *   -       -       TEMP. INLET:  TRI
C *   -       -       -     OUTLET: TRO
C *   COLD WATER FLOW = 100 KG/S
C *   -    -     TEMP. INLET:  TMC
C *   -    -     -     OUTLET: TMH
C *   IF NOT ACTIVE (NHEX=0) TEMPERATURES ARE KEPT AT 20 C.
C
      IF(NHEX.EQ.0.) THEN
        TROP=20.-TRO
        TMHP=20.-TMH
      ELSE
        WR=WREK/NHEX
        DW=WM-WR
        IF(ABS(DW).LT.1.E-12) DW=SIGN(1.E-12,DW)
        W=WM*WR/DW
        BR=TMA(KR/CP,W)
        TRM=BR*TRI+(1.-BR)*TRO
        TMM=(1.-BR)*TMC+BR*TMH
        QR=KR*(TRM-TMM)
        TROP=(WR*CP*(TRI-TRO)-QR)/CR
        TMHP=(QR-WM*CP*(TMH-TMC))/CMR
      ENDIF
      RETURN
      END
C
*****************************************************************
C
      REAL FUNCTION RFSF(T)
*****************************************************************
*                                                               *
*  temperature interval [20, 349]                               *
*  unit of specific density RFSF:  kg/m3                        *
*  maximal relative deviation:  0.3 %                           *
*                                                               *
*****************************************************************
      DATA A0 / 0.1005493812E+4/,    A1 /-0.3018835215E+0/,
     -     A2 /-0.2206957248E-3/,    A3 /-0.2353660831E-4/,
     -     A4 / 0.1061131468E-6/,    A5 /-0.1673865756E-9/
      RFSF = 0.0
      IF (T .GE. 20  .AND.  T .LE. 349) THEN
        RFSF =  A0 + (A1 + (A2 + (A3 + (A4 + A5*T)*T )*T )*T )*T
      ENDIF
      RETURN
      END



      REAL FUNCTION DRFSDT(T)
*****************************************************************
*                                                               *
*  temperature intervals: [20, 228.0] + ]228.0, 349]            *
*  unit of specific density-derivative DRFSDT:  (kg/m3)/C       *
*  maximal relative deviation:  0.8 %                           *
*                                                               *
*****************************************************************
      DATA A0 / 0.3437717836E-1/,     A1 / -0.1416289273E-1/,
     &     A2 / 0.1156086386E-3/,     A3 / -0.6728852701E-6/,
     &     A4 / 0.2055000985E-8/,     A5 / -0.2787133466E-11/
      DATA B0 / 0.5334329637E+3/,     B1 / -0.9895819938E+1/,
     &     B2 / 0.7308652833E-1/,     B3 / -0.2691927202E-3/,
     &     B4 / 0.4945037848E-6/,     B5 / -0.3628908295E-9/
      DRFSDT = 0.0
```

```
      IF (T .GE. 20  .AND.  T .LE. 228.0) THEN
        DRFSDT =  A0 + (A1 + (A2 + (A3 + (A4 + A5*T)*T )*T )*T )*T
      ELSE IF (T .GT. 228.0 .AND.  T .LE. 349) THEN
        DRFSDT =  B0 + (B1 + (B2 + (B3 + (B4 + B5*T)*T )*T )*T )*T
      ENDIF
      RETURN
      END



      REAL FUNCTION RGSF(T)
*****************************************************************
*                                                               *
*  temperature intervals: [100, 227.3] + ]227.3, 349] C         *
*  unit of specific density RGSF:  kg/m3                        *
*  maximal relative deviation:  0.6 %                           *
*                                                               *
*****************************************************************
      DATA A0 / 0.1735749083E+1/,       A1 /-0.5269083859E-1/,
     &     A2 / 0.6304656846E-3/,       A3 /-0.3273796912E-5/,
     &     A4 / 0.1102414334E-7/
      DATA B0 /-0.6140992612E+4/,       B1 / 0.1151251961E+3/,
     &     B2 /-0.8616023519E+0/,       B3 / 0.3220027249E-2/,
     &     B4 /-0.6011150718E-5/,       B5 / 0.4503478865E-8/
      RGSF = 0.0
      IF (T .GE. 100  .AND.  T .LE. 227.3) THEN
        RGSF = A0 + (A1+(A2+(A3+A4*T)*T)*T)*T
      ELSE IF (T .GT. 227.3  .AND.  T .LE. 349) THEN
        RGSF = B0 + (B1+(B2+(B3+(B4+B5*T)*T)*T)*T)*T
      ENDIF
      RETURN
      END



      REAL FUNCTION HFGF(T)
*****************************************************************
*                                                               *
*  temperature interval [100, 349]                              *
*  unit of heat of evaporation HFGF:  kJ/kg                     *
*  maximal relative deviation:  0.5 %                           *
*                                                               *
*****************************************************************
      DATA  A0 / 0.2935870339E+4/,      A1 /-0.1482271455E+2/,
     &      A2 / 0.1384884479E+0/,      A3 /-0.7533015053E-3/,
     &      A4 / 0.1936536231E-5/,      A5 /-0.2013297912E-8/
      HFGF = 0.0
      IF ( T .GE. 100  .AND.  T .LE. 349) THEN
         HFGF =  A0 + (A1 + (A2 + (A3 + (A4 + A5*T)*T )*T )*T )*T
      ENDIF
      RETURN
      END



      REAL FUNCTION DRFSDP(T)
*****************************************************************
*                                                               *
*  temperature range [100, 349] C                               *
*  unit of specific density-derivative:  (kg/m3)/bar            *
*  maximal relative deviation:  0.7 %                           *
*                                                               *
*****************************************************************
      DATA   A0 /-0.2031828028E+3/,     A1 / 0.4379027525E+1/,
     -       A2 /-0.4178752191E-1/,     A3 / 0.2187696452E-3/,
     -       A4 /-0.6524295808E-6/,     A5 / 0.1041696166E-8/,
```

```
     -       A6 /-0.692441105OE-12/
      DRFSDP = 0.0
      IF (T .GE. 100  .AND.  T .LE. 349) THEN
         DRFSDP = A0 + (A1+(A2+(A3+(A4+(A5+A6*T)*T)*T)*T)*T)*T
      ENDIF
      RETURN
      END



      REAL FUNCTION DRGSDP(T)
*****************************************************************
*                                                               *
*   temperature range [100, 349] C                              *
*   unit of specific density-derivative:  (kg/m3)/bar           *
*   maximal relative deviation:  0.5 %                          *
*                                                               *
*****************************************************************
      DATA A0 / 0.3971411148E+1/,          A1 /-0.1121203675E+0/,
     &     A2 / 0.1507915548E-2/,          A3 /-0.1064847323E-4/,
     &     A4 / 0.4130545450E-7/,          A5 /-0.8335217048E-10/,
     &     A6 / 0.6866921550E-13/
      DRGSDP = 0
      IF ( T .GE. 100  .AND.  T .LE. 349) THEN
           DRGSDP = A0 + (A1+(A2+(A3+(A4+(A5+A6*T)*T)*T)*T)*T)*T
      ENDIF
      RETURN
      END



      REAL FUNCTION DHGSDP(T)
*****************************************************************
*                                                               *
*  temperature intervals [100, 225.0] + ]225.0, 349] C          *
*  unit of enthalphy-derivative DHGSDP:  (kJ/kg)/bar            *
*                                                               *
*****************************************************************
      DATA  A0 / 0.1132427261E+4/,        A1 /-0.3265259167E+2/,
     &      A2 / 0.4120282008E+0/,        A3 /-0.2860619323E-2/,
     &      A4 / 0.1138739072E-4/,        A5 /-0.2445464618E-7/,
     &      A6 / 0.2202219548E-10/
      DATA  B0 / 0.3447744759E+4/,        B1 /-0.7302853373E+2/,
     &      B2 / 0.6427913056E+0/,        B3 /-0.3005052777E-2/,
     &      B4 / 0.7860723513E-5/,        B5 /-0.1089980236E-7/,
     &      B6 / 0.6253970320E-11/
      DHGSDP = 0.0
      IF (T .GE. 100  .AND.  T .LE. 225.0) THEN
         DHGSDP = A0 + (A1+(A2+(A3+(A4+(A5+A6*T)*T)*T)*T)*T)*T
      ELSE IF (T .GT. 225.0  .AND.  T .LE. 349) THEN
        DHGSDP = B0 + (B1+(B2+(B3+(B4+(B5+B6*T)*T)*T)*T)*T)*T
      ENDIF
      END



      REAL FUNCTION TSAF(P)
*****************************************************************
*                                                               *
*  pressure intervals:  [.015, .198] + ].198, .583] +           *
*  ].583, 1.079] + ]1.079, 5.82] + ]5.82, 20.04] +              *
*  ]20.04, 95.2] + ]95.2, 215]   BAR                            *
*  unit of TSAF:  C                                             *
*  maximal absolute deviation:  0.4 C                           *
*                                                               *
```

```
*****************************************************************
      DATA  A0 /-0.1986486572E+1/,       A1 / 0.1235282918E+4/,
     &      A2 /-0.1565427598E+5/,       A3 / 0.1212484442E+6/,
     &      A4 /-0.4800529212E+6/,       A5 / 0.7482279674E+6/
      DATA  B0 / 0.3306170561E+2/,       B1 / 0.1732713654E+3/,
     &      B2 /-0.2124834206E+3/,       B3 / 0.1179362746E+3/
      DATA  C0 / 0.4791308323E+2/,       C1 / 0.8938057175E+2/,
     &      C2 /-0.5173245680E+2/,       C3 / 0.1407354000E+2/
      DATA  D0 / 0.6694839024E+2/,       D1 / 0.4133920828E+2/,
     &      D2 /-0.9735335820E+1/,       D3 / 0.1342144934E+1/,
     &      D4 /-0.7383067289E-1/
      DATA  E0 / 0.1123259404E+3/,       E1 / 0.9704304144E+1/,
     &      E2 /-0.3552547938E+0/,       E3 / 0.6025789290E-2/
      DATA  F0 / 0.1498312295E+3/,       F1 / 0.4017940797E+1/,
     &      F2 /-0.5174186096E-1/,       F3 / 0.4054302082E-3/,
     &      F4 /-0.1287743113E-5/
      DATA  G0 / 0.2112637557E+3/,       G1 / 0.1320227019E+1/,
     &      G2 /-0.3714497389E-2/,       G3 / 0.4868745363E-5/
      IF (P .LT. 0.015  .OR.  P .GT. 215) TSAF = 0.0
      IF (P .LE. 0.198) THEN
        TSAF = A0 + (A1+(A2+(A3+(A4+A5*P)*P)*P)*P)*P
      ELSE IF (P .LE. 0.583) THEN
        TSAF = B0 + (B1+(B2+B3*P)*P)*P
      ELSE IF (P .LE. 1.079) THEN
        TSAF = C0 + (C1+(C2+C3*P)*P)*P
      ELSE IF (P .LE. 5.82) THEN
        TSAF = D0 + (D1+(D2+(D3+D4*P)*P)*P)*P
      ELSE IF (P .LE. 20.04) THEN
        TSAF = E0 + (E1+(E2+E3*P)*P)*P
      ELSE IF (P .LE. 95.2) THEN
        TSAF = F0 + (F1+(F2+(F3+F4*P)*P)*P)*P
      ELSE IF (P .LE. 215) THEN
        TSAF = G0 + (G1+(G2+G3*P)*P)*P
      ENDIF
      RETURN
      END


      REAL FUNCTION PSATF(T)
*****************************************************************
*                                                               *
*  temperature intervals:  [20, 230.5] + ]230.5, 370] C         *
*  unit of pressure PSATF:  bar                                 *
*  maximal relative deviation:  1 %                             *
*                                                               *
*****************************************************************
      DATA  A0 / 0.1106940804E-1/,       A1 / 0.6870879720E-5/,
     &      A2 / 0.2682724849E-4/,       A3 / 0.1435082492E-6/,
     &      A4 / 0.2500322168E-8/,       A5 / 0.3879453164E-10/,
     &      A6 /-0.4843103403E-13/
      DATA  B0 / 0.2468783974E+3/,       B1 /-0.3624533621E+1/,
     &      B2 / 0.1995290371E-1/,       B3 /-0.4946177715E-4/,
     &      B4 / 0.5754542600E-7/
      PSATF = 0.0
      IF (T .GE. 20  .AND.   T .LE. 230.5) THEN
        PSATF = A0 + (A1+(A2+(A3+(A4+(A5+A6*T)*T)*T)*T)*T)*T
      ELSE IF (T .GT. 230.5   .AND.  T .LE. 370) THEN
        PSATF = B0 + (B1+(B2+(B3+B4*T)*T)*T)*T
      ENDIF
      RETURN
      END


      REAL FUNCTION CPFSF(T)
*****************************************************************
*                                                               *
*  temperature interval [20, 349]                               *
*  unit of isobaric heatcapacity CPFSF:  kJ/kg/C                *
*  maximal relative deviation:  0.8 %                           *
```

```
*                                                                      *
************************************************************************
      DATA  A0 / 0.4717461212E+1/,      A1 /-0.5224223375E-1/,
     &      A2 / 0.1879277329E-2/,      A3 /-0.3362514716E-4/,
     &      A4 / 0.3361953344E-6/,      A5 /-0.1955025331E-8/,
     &      A6 / 0.6569898778E-11/,     A7 /-0.1182075243E-13/,
     &      A8 / 0.8820659651E-17/
      CPFSF = 0.0
      IF (T .GE. 20  .AND.  T .LE. 349) THEN
        CPFSF = A0 +
     &  (A1+(A2+(A3+(A4+(A5+(A6+(A7+A8*T)*T)*T)*T)*T)*T)*T)*T
      ENDIF
      RETURN
      END

************************************************************************
*
C     SLIP CORRELATION BY BRYCE
      FUNCTION BRYCE(PB,RS,RW,GA,A)
      DATA PC /22.106/
C
      P=PB/10.
      SQP=SQRT(P)
      G1=2667.*P+6192.*SQP-1229.
      G2=2060.*P+203.*SQP-23.4+32470*RS/RW
      X=A*RS
      X=X/(X+(1.-A)*RW)
      GHS=1./SQRT(X/G2**2+(1.-X)/G1**2)
      F=EXP(-3.*GA/GHS)
      B2=SQRT(RW/RS)
      B1=(.00856*P+.0231)*P+2.94
      B0=B1*B2/(1.+F*(B2-1.))
      A0=1.-.328*(1.-P/PC)*F**2
      IF(A.LE..9999) BRYCE=(1.-A)/(A0-A+(1.-A0)*A**B0)
      IF(A.GT..9999) BRYCE=1./(1.-B0*(1.-A0))
      RETURN
      END
*
************************************************************************
*
      FUNCTION TMA(X,W)
*
      IF(ABS(W).GT.1.E-20*X) GOTO 10
      IF(W.GE.0) B=0.
      IF(W.LT.0) B=1.
      GOTO 20
 10   Z=X/W
      B=.5
      IF(ABS(Z).LE.1.E-3) GOTO 20
      IF(ABS(Z).LE.47.) E=EXP(-Z)                                    851122
      IF(Z.LT.-47.) E=2.581E20                                       851122
      IF(Z.GT.47.) E=0.                                              851122
      B=1./Z-E/(1.-E)
 20   TMA=B
      RETURN
      END
C
************************************************************************
C
C *   SLIP CORRELATION BY BANKOFF-JONES
      FUNCTION BANK(PB,A)
C
      P=PB*14.50/1000.
      C=.09*P+.712
      R=(.46*P+.18)*P+3.33
      BANK=(1.-A)/(C-A+(1.-C)*A**R)
      RETURN
      END
C
************************************************************************
C
```

```
C
C     SIMULATION OF VARIABEL TIME DELAYS.
C     NUMBER OF DELAY FUNCTIONS AND ASSOTIATED DIMENSIONS MUST
C     BE UPDATED TO SAVE CORE SPACE OR GET ROOM FOR MORE FUNCTIONS.
C     PRESENT NUMBER OF FUNCTIONS IS KMAX=4.
C     THE BUFFER FOR EACH FUNCTION IS NBUF=5000 WORDS.
C     TOTAL BUFFER DB IS KMAX*NBUF
C
      FUNCTION DELAY0(INDI)
C
      COMMON /TIME/ T
      DIMENSION NUD(4),VAL(4),NIN(4),VALO(4)
      DIMENSION DB(20000)
      LOGICAL STORE
      CHARACTER*6 NAME(4),IDF
      DATA KMAX,NBUF /4,5000/
      DATA NUD,NIN /4*1,4*1/
      DATA KD,JS,IERD /0,0,0/
      DATA DTID,STID /.4,.4/
      DATA STORE /.FALSE./
      DATA NAME /4*' '/
C
      SAVE
C
      GOTO (500,600,700,800,900),INDI
CCCCCC      ======
      ENTRY TRNSTM(IDF,VNY,VG)
      NTYP=1
      GOTO 10
CCCCCC      ======
      ENTRY DEADTM(IDF,VNY,TAU)
      NTYP=2
C
 10   IF(T.LT.0) GOTO (30,40),NTYP
C     INSERT VARIABEL NAME IN NAME BUFFER IF NOT PRESENT
      DO 20 J=1,KD
 20   IF(IDF.EQ.NAME(J)) GOTO 100
      KD=KD+1
      IF(KD.GT.KMAX) GOTO 70
      NAME(KD)=IDF
      VALO(KD)=VNY
      STORE=.TRUE.
 25   GOTO (30,40),NTYP
 30   TRNSTM=0.
      RETURN
 40   DEADTM=VNY
      RETURN
C     ERROR REPORTS
 70   WRITE(6,80) IDF
 80   FORMAT(/X,'TOO MANY DELAY VARIABELS AT:',2X,A6)
 90   IERD=1
      GOTO 25
C
 100  CONTINUE
      JB=J
      VAL(JB)=VNY
      GOTO (200,300),NTYP
C
C     FIND DELAY AS NUMBER OF POSITIONS IN DELAY BUFFER DB
 200  IF(VG.LE.0) GOTO 240
 210  K=NUD(JB)+(JB-1)*NBUF
 220  IF(DB(K).GT.VG) GOTO 230
      IF(NUD(JB).GE.NIN(JB)) GOTO 240
      K=K+1
      NUD(JB)=NUD(JB)+1
      GOTO 220
 230  TRNSTM=NIN(JB)-NUD(JB)+(DB(K)-VG)/(DB(K)-DB(K-1))
      RETURN
 240  TRNSTM=NIN(JB)-NUD(JB)
      RETURN
C
```

```
C     FIND VALUE DELAYED TAU POSITIONS
 300  IF(TAU.EQ.0) GOTO 320
      NTAU=IFIX(TAU)
      REST=TAU-NTAU
      NUD(JB)=NIN(JB)-NTAU
      IF(NUD(JB).LE.1) GOTO 330
      K=NUD(JB)+(JB-1)*NBUF
      DEADTM=DB(K)-REST*(DB(K)-DB(K-1))
      RETURN
 320  DEADTM=VNY
      RETURN
 330  DEADTM=DB(1+(JB-1)*NBUF)
      RETURN
CCCCCC      ======
C     INSERTION OF VNY IN DB AFTER FIRST CALL OF DELAYO
 500  IF(.NOT.STORE) RETURN
      DO 510 J=JS+1,KD
      I=(J-1)*NBUF+NIN(J)
      DB(I)=VALO(J)
 510  NIN(J)=NIN(J)+1
      JS=KD
      STORE=.FALSE.
      RETURN
C
C     INSERTION OF VNY IN DB AFTER ACCEPTED TIME STEP
 600  IF(STORE) GOTO 500
      IF(T.LT.STID) RETURN
      STID=T+DTID
      DO 610 J=1,KD
      I=(J-1)*NBUF+NIN(J)
      DB(I)=VAL(J)
      NIN(J)=NIN(J)+1
      IF(NIN(J).GT.NBUF) GOTO 620
 610  CONTINUE
      DELAY=IERD
      RETURN
 620  WRITE(6,630)
 630  FORMAT(//X,'DELAY BUFFERS TOO SMALL',/)
      IERD=1
      DELAY=IERD
      RETURN
C
 700  RETURN
C
 800  RETURN
C
C     RESET FOR A NEW START
 900  KD=0
      JS=0
      IERD=0
      STID=DTID
      STORE=.FALSE.
      DO 910 J=1,KMAX
      NAME(J)=' '
      NIN(J)=1
      NUD(J)=1
 910  CONTINUE
      RETURN
      END
```

L.2 LPGRAPH.FOR

```
C      GRAPHIC & DECODING ROUTINES FOR THE BVT LOW-POWER
C
C      GRAPHICS USED IN MAIN-PANEL
C
C
C      RALPH MYRNAS      881220-890420
C
C
C
C**********************************************************************
       SUBROUTINE CRGRAPH(SUMCR)
C
C   ROUTINE FOR WRITING THE DIODES THAT SHOWS CURRENT TOTAL CONTROL-ROD
C
C   DISPLACEMENT
C
C   PARAMETERS :        SUMCR   =  SUMMA CONTROL-RODS OUT (0-10900 %)
C
       CHARACTER ESC,KOORD(1:5)
       INTEGER A,X,Y,SUMCR
       X=256
       IF(SUMCR.EQ.0) GOTO 999
       ESC=CHAR(27)
       Y=5+SUMCR/4
       CALL KOD(X,Y,KOORD)
       ESC=CHAR(27)
       WRITE(*,*) ESC,'L','H',(KOORD(K),K=1,5)
 999   RETURN
       END
**********************************************************************
C
       SUBROUTINE IRMGR(LASTX,X,IX,Y)
C
C   ROUTINE FOR DRAWING THE RED LINE IN THE FLUX & DBT - INDICATOR
C
C   PARAMETERS :        IX,Y    =  XY-COORDINATE  ( THE CENTER OF THE INDICATOR )
C                       LASTX   =  THE LAST VALUE ( FOR RUBBING OUT THE OLD LINE)
C                       X       =  NEW VALUE
C
       REAL LASTX,X,PI
       INTEGER IRMX,IRMY,Y,IX
       CHARACTER ESC,KOORD(1:5)
       ESC=CHAR(27)
       PI=3.14
       CALL KOD(IX,Y,KOORD)
       WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
       WRITE(*,*) ESC,'ML1'
       IRMX=300*COS(LASTX*PI*3/2-PI/4)
       IRMY=300*SIN(LASTX*PI*3/2-PI/4)
       CALL KOD(IX-IRMX,Y+IRMY,KOORD)
       WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
       IRMX=300*COS(X*PI*3/2-PI/4)
       IRMY=300*SIN(X*PI*3/2-PI/4)
       CALL KOD(IX,Y,KOORD)
       WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
       WRITE(*,*) ESC,'ML2'
       CALL KOD(IX-IRMX,Y+IRMY,KOORD)
       WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
       RETURN
       END
**********************************************************************
C
       SUBROUTINE INGRAPH
C
C   ROUTINE FOR DRAWING THE COMMAND SEGMENTS AND THE WINDOWS IN THE
C
C   UPPER RIGHT CORNER
C
```

```
      CHARACTER ESC,KOORD(1:5)
      ESC=CHAR(27)
      WRITE(*,*) ESC,'%!0'
      WRITE(*,*) ESC,'MM1'
      WRITE(*,*) ESC,'ML2'
C *-------------------------------------------------------
C  TIME-WINDOW
      CALL KOD(3450,2755,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT4TIME'
      CALL KOD(3700,2750,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(3700,2842,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3920,2842,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3920,2750,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3700,2750,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
C *-------------------------------------------------------
C  POWER-WINDOW
      CALL KOD(3450,2665,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT5POWER'
      CALL KOD(3700,2660,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(3700,2652,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3920,2652,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3920,2660,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3700,2660,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
C *-------------------------------------------------------
C  PRESSURE-WINDOW
      CALL KOD(3450,2385,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT5P/bar'
      CALL KOD(3700,2380,KOORD)
      WRITE(*,*) ESC,'MP0'
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(3700,2472,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3920,2472,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3920,2380,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3700,2380,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
C *-------------------------------------------------------
C  WATERLEVEL-WINDOW
      CALL KOD(3450,2205,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT5LEVEL'
      CALL KOD(3700,2200,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(3700,2292,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3920,2292,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3920,2200,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3700,2200,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
C *-------------------------------------------------------
C  TEMP-WINDOW
      CALL KOD(3450,2025,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT4TEMP'
```

```
      CALL KOD(3700,2020,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(3700,2112,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3920,2112,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3920,2020,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3700,2020,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
C *-------------------------------------------------------------
      WRITE(*,*) ESC,'MP0'
      WRITE(*,*) ESC,'MT?'
      WRITE(*,*) ESC,'ML?'
      CALL SINDI(3960,2850,'1')
      CALL BLOCK(3960,2550,'3')
      CALL SINDI(3960,2400,'2')
      CALL BLOCK(3960,2100,'4')
C *-------------------------------------------------------------
C DEFINE COMMAND SEGMENTS
C WATER
      WRITE(*,*) ESC,'MT7'
      CALL KOD(750,80,KOORD)
      WRITE(*,*) ESC,'SP',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'S02'
      WRITE(*,*) ESC,'MPE4'
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'1'
      CALL KOD(750,370,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1450,370,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1450,80,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
      CALL KOD(770,250,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT=WATER & STEAM'
      CALL KOD(932,130,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT5PANEL'
      WRITE(*,*) ESC,'SC'
C MAIN
      WRITE(*,*) ESC,'MT7'
      CALL KOD(1500,80,KOORD)
      WRITE(*,*) ESC,'SP',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'S01'
      WRITE(*,*) ESC,'MPE6'
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'1'
      CALL KOD(1500,370,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2200,370,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2200,80,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
C      CALL KOD(770,270,KOORD)
C      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
C      WRITE(*,*) ESC,'LT=WATER & STEAM'
      CALL KOD(1630,130,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT:MAIN-PANEL'
      WRITE(*,*) ESC,'SC'
C CRODS
      WRITE(*,*) ESC,'MT7'
      CALL KOD(2250,80,KOORD)
      WRITE(*,*) ESC,'SP',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'S03'
      WRITE(*,*) ESC,'MPE5'
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'1'
      CALL KOD(2250,370,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
```

```
      CALL KOD(2960,370,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2960,80,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
      CALL KOD(2270,260,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT<CONTROL-RODS'
      CALL KOD(2362,130,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT5PANEL'
      WRITE(*,*) ESC,'SC'
C STOP
      WRITE(*,*) ESC,'MPE:'
      CALL KOD(3000,80,KOORD)
      WRITE(*,*) ESC,'SP',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'S04'
      WRITE(*,*) ESC,'MPE5'
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'1'
      CALL KOD(3000,370,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3700,370,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3700,80,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
      CALL KOD(3090,260,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT4STOP'
C      CALL KOD(2362,130,KOORD)
C      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
C      WRITE(*,*) ESC,'LT5PANEL'
      WRITE(*,*) ESC,'SC'
      WRITE(*,*) ESC,'SS19'
      WRITE(*,*) ESC,'SS29'
      WRITE(*,*) ESC,'SS39'
      WRITE(*,*) ESC,'SS49'
      WRITE(*,*) ESC,'SA11!41234'
      WRITE(*,*) ESC,'SA21!41234'
      WRITE(*,*) ESC,'SA31!41234'
      WRITE(*,*) ESC,'SA41!41234'
      RETURN
      END
*************************************************************************
      SUBROUTINE SINDI(X,Y,CH)
C
C   ROUTINE FOR DRAWING THE REACTOR SCRAM SEGMENT
C
C   PARAMETERS :        X,Y    =   XY-COORDINATE
C                       CH     =   NAME OF SEGMENT (CH FOLLOWS S ex. S1)
C
      INTEGER X,Y
      CHARACTER ESC,KOORD(1:5),CH
      ESC=CHAR(27)
      WRITE(*,*) ESC,'S0S',CH
      CALL KOD(X,Y,KOORD)
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'1'
      CALL KOD(X+135,Y,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(X+135,Y-300,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(X,Y-300,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
      IF (CH.EQ.'1') THEN
        CALL KOD(4005,2690,KOORD)
        WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
        WRITE(*,*) ESC,'LT1S'
      ELSE
        CALL KOD(4005,2290,KOORD)
        WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
```

```
        WRITE(*,*) ESC,'LT1S'
        CALL KOD(4005,2190,KOORD)
        WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
        WRITE(*,*) ESC,'LT1S'
      ENDIF
      WRITE(*,*) ESC,'SC'
      RETURN
      END
*************************************************************************
      SUBROUTINE BLOCK(X,Y,CH)
C
C   ROUTINE FOR DRAWING THE  SCRAM-BYPASS SEGMENT
C
C   PARAMETERS :        X,Y    =   XY-COORDINATE
C                       CH     =   NAME OF SEGMENT (CH FOLLOWS S ex. S1)
C
      INTEGER X,Y
      CHARACTER ESC,KOORD(1:5),CH
      ESC=CHAR(27)
      WRITE(*,*) ESC,'SOS',CH
      CALL KOD(X,Y,KOORD)
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'1'
      CALL KOD(X+135,Y,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(X+135,Y-70,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(X,Y-70,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
      WRITE(*,*) ESC,'MCA1A:7'
      CALL KOD(X+12,Y-52,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT5BLOCK'
      WRITE(*,*) ESC,'MCB7C;<'
      WRITE(*,*) ESC,'SC'
      RETURN
      END
*************************************************************************
      SUBROUTINE AUTO
C
C   ROUTINE FOR DRAWING THE AUTO SEGMENT
C
      CHARACTER ESC,KOORD(1:5)
      ESC=CHAR(27)
      WRITE(*,*) ESC,'MT0'
      WRITE(*,*) ESC,'S05'
      CALL KOD(3050,2650,KOORD)
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'1'
      CALL KOD(3200,2650,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3200,2600,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3050,2600,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
      WRITE(*,*) ESC,'MCA4A<:'
      CALL KOD(3070,2610,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT4AUTO'
      WRITE(*,*) ESC,'MCB7C;<'
      WRITE(*,*) ESC,'SC'
      WRITE(*,*) ESC,'SA51!11'
      WRITE(*,*) ESC,'SS58'
      RETURN
      END
*************************************************************************
      SUBROUTINE SIRM
C
C   ROUTINE FOR DRAWING THE SIRM SEGMENT
C
      CHARACTER ESC,KOORD(1:5)
```

```
      ESC=CHAR(27)
      WRITE(*,*) ESC,'MT0'
      WRITE(*,*) ESC,'S06'
      CALL KOD(1184,1820,KOORD)
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'1'
      CALL KOD(1334,1820,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1334,1870,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1184,1870,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
      WRITE(*,*) ESC,'MCA4A<:'
      CALL KOD(1204,1830,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT4SIRM'
      WRITE(*,*) ESC,'MCB7C;<'
      WRITE(*,*) ESC,'SC'
      WRITE(*,*) ESC,'SA61!11'
      WRITE(*,*) ESC,'SS68'
      RETURN
      END
*************************************************************************
C
      SUBROUTINE FLGRAPH(Y)
C
C   ROUTINE FOR DRAWING THE DIODES THAT INDICATES CHOSEN IRM-CHANNEL
C
C   PARAMETERS :        Y    =   Y-COORDINATE
C
      INTEGER X,Y
      CHARACTER ESC,KOORD(1:5)
      ESC=CHAR(27)
      WRITE(*,*) ESC,'MM:'
      IF (Y.LT.7) THEN
         X=3000
         CALL KOD(X,1900+Y*100,KOORD)
         WRITE(*,*) ESC,'L','H',(KOORD(K),K=1,5)
      ELSE
         X=3200
         CALL KOD(X,1900+(Y-6)*100,KOORD)
         WRITE(*,*) ESC,'L','H',(KOORD(K),K=1,5)
      ENDIF
      WRITE(*,*) ESC,'MM1'
      RETURN
      END
*************************************************************************
C
      SUBROUTINE KOD(X,Y,KOORD)
C
C   ROUTINE FOR ENCODING X,Y-COORDINATES TO FIVE CHARACTERS
C
C   PARAMETERS :        X,Y    =   XY-COORDINATE
C                       KOORD  =   ARRAY WHERE THE CHARACTERS ARE SAVED
C
      INTEGER A,B,C,D,X,Y
      CHARACTER HIY,LOY,KOORD(1:5)
      A=X/128
      B=(X-128*A)/4
      D=X-128*A-4*B
      KOORD(4)=CHAR(32+A)
      KOORD(5)=CHAR(64+B)
      A=Y/128
      B=(Y-128*A)/4
      C=(Y-128*A-4*B)*4+D
      KOORD(2)=CHAR(96+C)
      KOORD(1)=CHAR(32+A)
      KOORD(3)=CHAR(96+B)
      RETURN
      END
*************************************************************************
```

```
C
      SUBROUTINE INTKOD(INT,CHARARR)
C
C   ROUTINE FOR ENCODING INTEGERS TO THREE CHARACTERS
C
C   PARAMETERS :        INT     =   INTEGER
C                       CHARARR =   ARRAY WHERE THE CHARACTERS ARE SAVED
C
      CHARACTER CHARARR(1:5)
      INTEGER A,B,C,INT
      A=INT/1024
      B=(INT-A*1024)/16
      C=INT-A*1024-B*16
      CHARARR(1)=CHAR(64+A)
      CHARARR(2)=CHAR(64+B)
      CHARARR(3)=CHAR(48+C)
      RETURN
      END
*************************************************************************
      SUBROUTINE PANEL
C
C   ROUTINE FOR DRAWING THE MAIN-PANEL SEGMENT
C
      COMMON /CCROD/ SCR(3),DCR(3),XCR(12),CCR(12),ACR(30)
      CHARACTER CH
      CHARACTER ESC,KRD(1:5),KOORD(1:5),CA1(1:3),CA2(1:3),CA3(1:3)
      INTEGER I,X,Y,Z,SUMCR,START,J,K
      X=256
      ESC=CHAR(27)
      SUMCR=ACR(25)/80*80
C*-------------------------------------------------------------
C START SEGMENT
      WRITE(*,*) ESC,'SOA1'
C *-------------------------------------------------------------------
C   GREY BACKGROUND
      CALL KOD(0,0,KOORD)
      WRITE(*,*) ESC,'MLO'
      WRITE(*,*) ESC,'MP/'
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'0'
      CALL KOD(0,3000,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3400,3000,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3400,1950,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(4095,1950,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(4095,0,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(612,0,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(612,2750,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(200,2750,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(200,0,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
C      CALL KOD(639,511,KRD)
C      WRITE(*,*) ESC,'RR',(KOORD(J),J=1,5),(KRD(K),K=1,5),'?'
C* ------------------------------------------------------
C* DRAW INSTRUMENT LINES
      WRITE(*,*) ESC,'MT4'
      Y = 2300
      DO 45 I=1,2
         DO 35 X=1250,3450,1100
            IF (I.EQ.1 .AND. X.EQ.3450) GOTO 35
            WRITE(*,*) ESC,'MP!'
            CALL KOD(X-500,Y-400,KOORD)
            WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'1'
            CALL KOD(X-500,Y+400,KOORD)
```

```
                WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
                CALL KOD(X+500,Y+400,KOORD)
                WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
                CALL KOD(X+500,Y-400,KOORD)
                WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
                WRITE(*,*) ESC,'LE'
                CALL KOD(X-242,Y-362,KOORD)
                WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                WRITE(*,*) ESC,'LT1/'
                CALL KOD(X-350,Y-95,KOORD)
                WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                WRITE(*,*) ESC,'LT1_'
                CALL KOD(X-242,Y+110,KOORD)
                WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                WRITE(*,*) ESC,'LT1\'
                CALL KOD(X,Y+200,KOORD)
                WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                WRITE(*,*) ESC,'LT1|'
                CALL KOD(X+210,Y+110,KOORD)
                WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                WRITE(*,*) ESC,'LT1/'
                CALL KOD(X+300,Y-95,KOORD)
                WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                WRITE(*,*) ESC,'LT1_'
                CALL KOD(X+210,Y-362,KOORD)
                WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                WRITE(*,*) ESC,'LT1\'
C * MARK %
                IF (X.GT.1300) THEN
                   CALL KOD(X-16,Y-300,KOORD)
                   WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                   WRITE(*,*) ESC,'LT1%'
                   CALL KOD(X-300,Y-385,KOORD)
                   WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                   WRITE(*,*) ESC,'LT10'
                   CALL KOD(X-455,Y-110,KOORD)
                   WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                   WRITE(*,*) ESC,'LT220'
                   CALL KOD(X-290,Y+175,KOORD)
                   WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                   WRITE(*,*) ESC,'LT240'
                   CALL KOD(X-44,Y+265,KOORD)
                   WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                   WRITE(*,*) ESC,'LT260'
                   CALL KOD(X+240,Y+175,KOORD)
                   WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                   WRITE(*,*) ESC,'LT280'
                   CALL KOD(X+350,Y-110,KOORD)
                   WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                   WRITE(*,*) ESC,'LT3100'
                   CALL KOD(X+250,Y-385,KOORD)
                   WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                   WRITE(*,*) ESC,'LT3120'
                ENDIF
C * MARK SRM
                IF (X.EQ.1250 .AND. Y.EQ.1400) THEN
                   CALL KOD(X-390,Y-385,KOORD)
                   WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                   WRITE(*,*) ESC,'LT210'
                   CALL KOD(X-320,Y-340,KOORD)
                   WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                   WRITE(*,*) ESC,'LT2-3'
                   CALL KOD(X-300,Y+175,KOORD)
                   WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                   WRITE(*,*) ESC,'LT210'
                   CALL KOD(X-210,Y+220,KOORD)
                   WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                   WRITE(*,*) ESC,'LT10'
                   CALL KOD(X+240,Y+175,KOORD)
                   WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                   WRITE(*,*) ESC,'LT210'
```

```
                CALL KOD(X+330,Y+220,KOORD)
                WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                WRITE(*,*) ESC,'LT13'
                CALL KOD(X+250,Y-385,KOORD)
                WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                WRITE(*,*) ESC,'LT210'
                CALL KOD(X+340,Y-340,KOORD)
                WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                WRITE(*,*) ESC,'LT16'
             ENDIF
C * MARK DBT
             IF (X.EQ.1250 .AND. Y.EQ.2300) THEN
                CALL KOD(X-400,Y-385,KOORD)
                WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                WRITE(*,*) ESC,'LT4-100'
                CALL KOD(X-400,Y-110,KOORD)
                WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                WRITE(*,*) ESC,'MRE:0'
                WRITE(*,*) ESC,'LT18'
                WRITE(*,*) ESC,'MR00'
                CALL KOD(X-370,Y+175,KOORD)
                WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                WRITE(*,*) ESC,'LT3100'
                CALL KOD(X-44,Y+265,KOORD)
                WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                WRITE(*,*) ESC,'LT250'
                CALL KOD(X+240,Y+175,KOORD)
                WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                WRITE(*,*) ESC,'LT230'
                CALL KOD(X+350,Y-110,KOORD)
                WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                WRITE(*,*) ESC,'LT220'
                CALL KOD(X+250,Y-385,KOORD)
                WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
                WRITE(*,*) ESC,'LT215'
             ENDIF
 35       CONTINUE
          Y=1400
 45    CONTINUE
C-----------------------------------------------------------------
C* DRAW FLUX INDICATOR LINES, LIGHT GREY
C       DO 10 I=800,3300,500
C          CALL KOD(I,1000,KOORD)
C          WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
C          CALL KOD(I,2750,KOORD)
C          WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
C          CALL KOD(I+400,2750,KOORD)
C          WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
C          CALL KOD(I+400,1000,KOORD)
C          WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
C          CALL KOD(I,1000,KOORD)
C          WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
C 10    CONTINUE
C *-------------------------------------------------------------
C  WRITE PANEL TEXT
C  MAGENTA
       WRITE(*,*) ESC,'MT7'
       CALL KOD(280,2770,KOORD)
       WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
       WRITE(*,*) ESC,'LT5%CROD'
       WRITE(*,*) ESC,'MT0'
       CALL KOD(1184,1020,KOORD)
       WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
       WRITE(*,*) ESC,'LT3SRM'
       CALL KOD(1184,1920,KOORD)
       WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
       WRITE(*,*) ESC,'LT3DBT'
       CALL KOD(2290,1920,KOORD)
       WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
       WRITE(*,*) ESC,'LT3IRM'
       CALL KOD(2290,1020,KOORD)
```

```
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT3PRM'
      CALL KOD(3368,1020,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT4SIRM'
      WRITE(*,*) ESC,'MT2'
      CALL KOD(1190,2000,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT3sec'
      CALL KOD(1184,1110,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT3p/s'
      CALL KOD(750,770,KOORD)
      CALL INTKOD(106,CA1)
      CALL INTKOD(160,CA2)
      CALL INTKOD(52,CA3)
      WRITE(*,*) ESC,'MC',(CA1(I),I=1,3),(CA2(J),J=1,3),(CA3(K),K=1,3)
      WRITE(*,*) ESC,'MT0'
      WRITE(*,*) ESC,'MB!!'
      WRITE(*,*) ESC,'MQ1'
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LTA5*** BVT-SIMULATOR ***'
      WRITE(*,*) ESC,'MCB7C;<'
      WRITE(*,*) ESC,'MQ2'
C      CALL KOD(972,700,KOORD)
C      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
C      WRITE(*,*) ESC,'LT?PROCESSIMULATOR'
C *-------------------------------------------------------------
C  SET CROD INDICATOR
      WRITE(*,*) ESC,'MT7'
      WRITE(*,*) ESC,'ML3'
      Z=0
      X=256
 5    CALL KOD(X,5+Z/4,KOORD)
      WRITE(*,*) ESC,'LH',(KOORD(K),K=1,5)
      Z=Z+80
      IF (Z/400*400-Z.EQ.0) THEN
         WRITE(*,*) ESC,'LT1-'
         IF (Z.EQ.2000) WRITE(*,*) ESC,'LT42000'
         IF (Z.EQ.4000) WRITE(*,*) ESC,'LT44000'
         IF (Z.EQ.6000) WRITE(*,*) ESC,'LT46000'
         IF (Z.EQ.8000) WRITE(*,*) ESC,'LT48000'
         IF (Z.EQ.10000) WRITE(*,*) ESC,'LT510000'
      ENDIF
      IF (Z.LT.10900) GOTO 5
C      Z=0
C      WRITE(*,*) ESC,'ML2'
      WRITE(*,*) CHAR(7)
C 6    CALL KOD(X,5+Z/4,KOORD)
C      WRITE(*,*) ESC,'LH',(KOORD(K),K=1,5)
C      Z=Z+80
C      IF (Z.LE.SUMCR) GOTO 6
      WRITE(*,*) ESC,'MT0'
C *-------------------------------------------------------------------
C  SET IRM-CHANNEL INDICATOR
      WRITE(*,*) ESC,'MT7'
      CALL KOD(3000,2700,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT7IRM.CH.'
      WRITE(*,*) ESC,'MM:'
      WRITE(*,*) ESC,'ML0'
      I=1
      DO 30 X=3000,3200,200
         Z=100
 7       CALL KOD(X,1900+Z,KOORD)
         WRITE(*,*) ESC,'LH',(KOORD(K),K=1,5)
         IF (I.LT.10)   WRITE(*,*) ESC,'LT2 ',CHAR(48+I)
         IF (I.EQ.10)   WRITE(*,*) ESC,'LT3 10'
         IF (I.EQ.11)   WRITE(*,*) ESC,'LT3 11'
         IF (I.EQ.12)   WRITE(*,*) ESC,'LT3 12'
         I=I+1
```

```
         Z=Z+100
         IF (Z.LE.600) GOTO 7
 30    CONTINUE
       WRITE(*,*) ESC,'SC'
       WRITE(*,*) ESC,'SAA11!11'
       WRITE(*,*) ESC,'MM1'
C *-------------------------------------------------------------------
       RETURN
       END
*************************************************************************
C
       SUBROUTINE SHOWTIME
C
C   ROUTINE FOR DISPLAYING TIME AND CONTROL-RODS OUT
C
       COMMON /CCROD/ SCR(3),DCR(3),XCR(12),CCR(12),ACR(30)
       CHARACTER ESC,CHR1,CHR2,CHR3
       CHARACTER C1,C2,C3,C4,C5,KOORD(1:5),KRD(1:5)
       INTEGER  SEK1,SEK2,MIN,SUMCR,SA,SB,SC,SD,SE
       ESC=CHAR(27)
       WRITE(*,*) ESC,'MB1!'
       SUMCR=ACR(25)
       C1=CHAR(48+SA)
       C2=CHAR(48+SB)
       C3=CHAR(48+SC)
       C4=CHAR(48+SD)
       C5=CHAR(48+SE)
       SA=SUMCR/10000
       SB=(SUMCR-SA*10000)/1000
       SC=(SUMCR-SA*10000-SB*1000)/100
       SD=(SUMCR-SA*10000-SB*1000-SC*100)/10
       SE=SUMCR-SA*10000-SB*1000-SC*100-SD*10
       WRITE(*,*) ESC,'MT0'
       CALL KOD(300,10,KOORD)
       WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
       WRITE(*,*) ESC,'LT5',C1,C2,C3,C4,C5
       C1=CHAR(48+SA)
       C2=CHAR(48+SB)
       C3=CHAR(48+SC)
       C4=CHAR(48+SD)
       C5=CHAR(48+SE)
       WRITE(*,*) ESC,'MT5'
       WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
       WRITE(*,*) ESC,'LT5',C1,C2,C3,C4,C5
       CHR1=CHAR(48+MIN)
       CHR2=CHAR(48+SEK2)
       CHR3=CHAR(48+SEK1)
       WRITE(*,*) ESC,'MT0'
       CALL KOD(3720,2760,KOORD)
       WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
       WRITE(*,*) ESC,'LT4',CHR1,':',CHR2,CHR3
       SEK1=SEK1+1
       IF (SEK1.GT.9) THEN
          SEK1=0
          SEK2=SEK2+1
       ENDIF
       IF (SEK2.GT.5) THEN
          SEK2=0
          MIN=MIN+1
       ENDIF
       WRITE(*,*) ESC,'MT5'
       WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
       WRITE(*,*) ESC,'LT4',CHAR(48+MIN),':',CHAR(48+SEK2),CHAR(48+SEK1)
       RETURN
       END
*************************************************************************
       SUBROUTINE DECIKOD(TAL,C)
C
C   ROUTINE FOR ENCODING INTEGERS TO ITS CORRESPONDING  DECIMAL
C   CHARACTERS
C
```

```
C   PARAMETERS :        TAL     =   INTEGER
C                       C       =   ARRAY WHERE THE CHARACTERS ARE SAVED
C
      CHARACTER C(1:5)
      INTEGER DA,DB,DC,DD,DE,TAL
      DA=TAL/10000
      DB=(TAL-DA*10000)/1000
      DC=(TAL-DA*10000-DB*1000)/100
      DD=(TAL-DA*10000-DB*1000-DC*100)/10
      DE=TAL-DA*10000-DB*1000-DC*100-DD*10
      C(1)=CHAR(48+DA)
      C(2)=CHAR(48+DB)
      C(3)=CHAR(48+DC)
      C(4)=CHAR(48+DD)
      C(5)=CHAR(48+DE)
      RETURN
      END
```

L.3 WATER.FOR

```
C    GRAPHIC ROUTINES FOR THE BVT LOW-POWER PLANT MODEL
C
C    CONNECTED WITH FWCON,PECON & REAC
C
C    GRAPHICS USED IN WATER & STEAM CONTROL
C
C    RALPH MYRNAS 881220-890420
C
C*****************************************************************
C
       SUBROUTINE WATER
C
C   ROUTINE FOR DEFINING THE WATER & STEAM SEGMENT
C
       CHARACTER ESC
       CHARACTER KOORD(1:5)
       REAL ANGLE
       INTEGER X,Y
       ESC=CHAR(27)
C *-------------------------------------------------------
C
       WRITE(*,*) ESC,'ML0'
       WRITE(*,*) ESC,'MT2'
       WRITE(*,*) ESC,'MP!'
C *-------------------------------------------------------
C   DEFINE +/- SEGMENT
       CALL SEGP(2270,1900,'5','+')
       CALL SEGP(2500,1900,'6','-')
       CALL SEGP(300,680,'7','+')
       CALL SEGP(530,680,'8','-')
       CALL SEGP(1620,2550,'9','+')
       CALL SEGP(1850,2550,':','-')
C *-------------------------------------------------------
C    DEFINE SEGMENTS FOR CHOOSING NUMBER OF 321
       WRITE(*,*) ESC,'MP!'
       CALL NOF321('0')
       CALL NOF321('1')
       CALL NOF321('2')
C *-------------------------------------------------------
C    DEFINE A PUMP
       CALL PUMP(1200,820,'0')
C BEGIN SEGMENT
       WRITE(*,*) ESC,'SOA2'
C GREY BACKGROUND
       WRITE(*,*) ESC,'ML1'
       WRITE(*,*) ESC,'MP/'
       CALL KOD(0,0,KOORD)
       WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'0'
       CALL KOD(0,3000,KOORD)
       WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
       CALL KOD(3400,3000,KOORD)
       WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
       CALL KOD(3400,1950,KOORD)
       WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
       CALL KOD(4095,1950,KOORD)
       WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
       CALL KOD(4095,0,KOORD)
       WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
       CALL KOD(612,0,KOORD)
       WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
       CALL KOD(612,100,KOORD)
       WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
       CALL KOD(200,100,KOORD)
       WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
       CALL KOD(200,0,KOORD)
       WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
       WRITE(*,*) ESC,'LE'
C TEXT
```

```
      WRITE(*,*) ESC,'MT0'
      CALL KOD(300,120,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT5%CROD'
C COOLERS
      WRITE(*,*) ESC,'MPE6'
      DO 5 Y=800,1400,600
         CALL KOD(2230,Y,KOORD)
         WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'1'
         CALL KOD(2230,Y+200,KOORD)
         WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
         CALL KOD(2500,Y+200,KOORD)
         WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
         CALL KOD(2500,Y,KOORD)
         WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
         WRITE(*,*) ESC,'LE'
 5    CONTINUE
C REACTOR
      WRITE(*,*) ESC,'MPE4'
      CALL KOD(550,1450,KOORD)
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'1'
      CALL KOD(550,2500,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(520,2500,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(520,2530,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(550,2530,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
C TOP
      DO 3 ANGLE=0.02,3.14,.08
         X=730-180*COS(ANGLE)
         Y=2530+180*SIN(ANGLE)
         CALL KOD(X,Y,KOORD)
         WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
 3    CONTINUE
      CALL KOD(940,2530,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(940,2500,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(910,2500,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(910,1450,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
C BOTTOM
      DO 4 ANGLE=0,3.14,.02
         X=730+180*COS(ANGLE)
         Y=1450-180*SIN(ANGLE)
         CALL KOD(X,Y,KOORD)
         WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
 4    CONTINUE
      WRITE(*,*) ESC,'LE'
C SEA-WATER
      WRITE(*,*) ESC,'MPE4'
      CALL KOD(2020,600,KOORD)
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'0'
      CALL KOD(2750,600,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2750,420,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2020,420,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
C IN/OUT-CHANNEL
      WRITE(*,*) ESC,'MP9'
      CALL KOD(2000,700,KOORD)
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'0'
      CALL KOD(2000,400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2770,400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
```

```
      CALL KOD(2770,700,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2750,700,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2750,420,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2395,420,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2395,700,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2375,700,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2375,420,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2020,420,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2020,700,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
C
C MAIN CIRKULATION 1
      CALL KOD(550,1700,KOORD)
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'0'
      CALL KOD(350,1580,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(350,1200,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(580,1200,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(620,1300,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(580,1330,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(550,1230,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(380,1230,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(380,1550,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(550,1670,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
C NEXT
      CALL KOD(550,1600,KOORD)
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'0'
      CALL KOD(450,1550,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(450,1250,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(530,1250,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(560,1350,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(540,1380,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(510,1280,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(480,1280,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(480,1520,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(550,1570,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
C MAIN CIRKULATION 2
      CALL KOD(910,1700,KOORD)
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'0'
      CALL KOD(1110,1580,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1110,1200,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
```

```
      CALL KOD(880,1200,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(840,1300,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(880,1330,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(910,1230,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1080,1230,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1080,1550,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(910,1670,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
C NEXT
      CALL KOD(910,1600,KOORD)
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'0'
      CALL KOD(1010,1550,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1010,1250,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(930,1250,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(900,1350,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(920,1380,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(950,1280,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(980,1280,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(980,1520,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(910,1570,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
C STEAMLINES
      WRITE(*,*) ESC,'MPE7'
      CALL KOD(910,2400,KOORD)
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'0'
      CALL KOD(1110,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1110,2050,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1120,2050,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1120,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1310,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1310,2050,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1320,2050,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1320,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1660,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1660,2220,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1875,2220,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1875,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2100,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2100,2220,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2115,2220,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
```

```
      CALL KOD(2115,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2300,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2300,2220,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2310,2220,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2310,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2400,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2400,2220,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2410,2220,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2410,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2500,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2500,2220,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2510,2220,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2510,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2800,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2800,2120,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2830,2120,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2830,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3000,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(3000,2430,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
C      CALL KOD(2600,2420,KOORD)
C      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
C      CALL KOD(2600,2430,KOORD)
C      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(910,2430,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      DO 10 Y=2400,2280,-60
         CALL KOD(1845,Y,KOORD)
         WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'O'
         CALL KOD(1690,Y,KOORD)
         WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
         CALL KOD(1690,Y-30,KOORD)
         WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
         CALL KOD(1845,Y-30,KOORD)
         WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
         CALL KOD(1845,Y,KOORD)
         WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
 10   CONTINUE
      WRITE(*,*) ESC,'LE'
C
C LEFT WATERPIPES
      CALL KOD(550,2100,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(200,2100,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(200,760,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(960,760,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1080,760,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(1140,760,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
```

```
      CALL KOD(1260,760,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(1340,760,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1260,700,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(1260,820,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1140,820,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1140,700,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1260,700,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1080,700,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(1080,820,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(960,820,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(960,700,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1080,700,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(900,760,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(900,880,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(740,880,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(740,760,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(200,2100,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(200,2200,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(420,2200,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(420,2100,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(470,2100,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(470,1820,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(550,1820,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
C
C RIGHT WATERPIPES
      CALL KOD(910,1820,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(2700,1820,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2700,1500,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2475,1500,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2400,1450,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2300,1550,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2250,1500,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1110,1500,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
C* LOWER WATERPIPES
      CALL KOD(2265,1400,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(2265,1000,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2465,1400,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
```

```
      CALL KOD(2465,1000,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
C
      CALL KOD(2600,600,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(2600,900,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2475,900,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2400,850,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2300,950,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2250,900,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2130,900,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2130,600,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
C
C INCLUDE COPIES OF PUMP
      WRITE(*,*) ESC,'MP'''
      WRITE(*,*) ESC,'LKP0'
      CALL KOD(1200,700,KOORD)
      WRITE(*,*) ESC,'SXP0',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LKP0'
      CALL KOD(1095,1350,KOORD)
      WRITE(*,*) ESC,'SIP01010B=1',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LKP0'
      CALL KOD(995,1350,KOORD)
      WRITE(*,*) ESC,'SXP0',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LKP0'
      CALL KOD(465,1350,KOORD)
      WRITE(*,*) ESC,'SXP0',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LKP0'
      CALL KOD(365,1350,KOORD)
      WRITE(*,*) ESC,'SXP0',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LKP0'
      CALL KOD(1800,1500,KOORD)
      WRITE(*,*) ESC,'SIP01010B=2',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LKP0'
      CALL KOD(2465,1200,KOORD)
      WRITE(*,*) ESC,'SIP01010H71',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LKP0'
      CALL KOD(2600,800,KOORD)
      WRITE(*,*) ESC,'SXP0',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LKP0'
      WRITE(*,*) ESC,'SC'
C   CLOSE SEGMENT
C *-------------------------------------------------------------------
      WRITE(*,*) ESC,'SAA21!12'
      RETURN
      END
*************************************************************************
      SUBROUTINE SEGP(X,Y,CH,CH2)
C
C    ROUTINE FOR DEFINING +/- SEGMENT
C
C    PARAMETERS :     X,Y  =  X,Y COORDINATE
C                     CH   =  NAME OF SEGMENT (FOLLOWS U)
C                     CH2  =  CHARACTER TO BE WRITTEN INSIDE THE SEGMENT
C
      CHARACTER ESC,KOORD(1:5),CH,CH2
      INTEGER X,Y
      ESC=CHAR(27)
      WRITE(*,*) ESC,'SOU',CH
      CALL KOD(X,Y,KOORD)
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'O'
      CALL KOD(X+50,Y,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(X+50,Y+50,KOORD)
```

```
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(X,Y+50,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
      CALL KOD(X+10,Y,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT1',CH2
      WRITE(*,*) ESC,'SC'
      WRITE(*,*) ESC,'SAU',CH,'1!12'
      WRITE(*,*) ESC,'SSU',CH,'8'
      RETURN
      END
*********************************************************************
      SUBROUTINE NOF321(CH)
C
C    ROUTINE FOR DEFINING NUMBER OF 321 SEGMENT
C
C    PARAMETERS :     CH   =  NAME OF SEGMENT (FOLLOWS F)  &
C                             CHARACTER TO BE WRITTEN INSIDE THE SEGMENT
C
      CHARACTER ESC,KOORD(1:5),CH
      INTEGER X
      ESC=CHAR(27)
      IF (CH.EQ.'0') X=2280
      IF (CH.EQ.'1') X=2340
      IF (CH.EQ.'2') X=2400
      WRITE(*,*) ESC,'MT2'
      WRITE(*,*) ESC,'SOF',CH
      CALL KOD(X,1300,KOORD)
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'0'
      CALL KOD(X+50,1300,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(X+50,1365,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(X,1365,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
      CALL KOD(X+8,1305,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT1',CH
      WRITE(*,*) ESC,'SC'
      WRITE(*,*) ESC,'SAF',CH,'1!12'
      WRITE(*,*) ESC,'SSF',CH,'8'
      RETURN
      END
*********************************************************************
      SUBROUTINE PUMP(A,B,CH)
C
C    ROUTINE FOR DEFINING PUMP SEGMENT
C
C    PARAMETERS :     A,B  =  X,Y COORDINATE
C                     CH   =  NAME OF SEGMENT (FOLLOWS P)
C
C PUMP
      CHARACTER ESC,KOORD(1:5),CH
      INTEGER X,Y,A,B
      REAL ANGLE
      ESC=CHAR(27)
      CALL KOD(A,B,KOORD)
      WRITE(*,*) ESC,'SP',(KOORD(K),K=1,5)
      CALL KOD(A-40,B-40,KOORD)
      WRITE(*,*) ESC,'SOP',CH
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'1'
      CALL KOD(A-40,B+40,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(A+40,B+40,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(A+40,B-40,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(A+20,B+20,KOORD)
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'1'
```

```
      CALL KOD(A-20,B,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(A+20,B-20,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
      WRITE(*,*) ESC,'SC'
      WRITE(*,*) ESC,'SAP',CH,'1!12'
      WRITE(*,*) ESC,'SSP',CH,'8'
      RETURN
      END
*************************************************************************
      SUBROUTINE VENTIL(A,B,CH)
C
C    ROUTINE FOR DEFINING VENTIL SEGMENT
C
C    PARAMETERS :     A,B  =  X,Y COORDINATE
C                     CH   =  NAME OF SEGMENT (FOLLOWS V)
C
C VENTIL
      CHARACTER ESC,KOORD(1:5),CH
      INTEGER X,Y,A,B
      REAL ANGLE
      ESC=CHAR(27)
      CALL KOD(A,B,KOORD)
      WRITE(*,*) ESC,'SP',(KOORD(K),K=1,5)
      CALL KOD(A-40,B-25,KOORD)
      WRITE(*,*) ESC,'SOV',CH
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'1'
      CALL KOD(A-40,B+25,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(A+40,B-25,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(A+40,B+25,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
      WRITE(*,*) ESC,'SC'
      WRITE(*,*) ESC,'SAV',CH,'1!12'
      WRITE(*,*) ESC,'SSV',CH,'8'
      RETURN
      END
*************************************************************************
      SUBROUTINE WENTIL(A,B,CH)
C
C VERTIKAL VENTIL
C
C    PARAMETERS :     A,B  =  X,Y COORDINATE
C                     CH   =  NAME OF SEGMENT (FOLLOWS W)
C
      CHARACTER ESC,KOORD(1:5),CH
      INTEGER X,Y,A,B
      REAL ANGLE
      ESC=CHAR(27)
      CALL KOD(A,B,KOORD)
      WRITE(*,*) ESC,'SP',(KOORD(K),K=1,5)
      CALL KOD(A-25,B-40,KOORD)
      WRITE(*,*) ESC,'SOW',CH
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'1'
      CALL KOD(A+25,B-40,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(A-25,B+40,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(A+25,B+40,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
      WRITE(*,*) ESC,'SC'
      WRITE(*,*) ESC,'SAW',CH,'1!12'
      WRITE(*,*) ESC,'SSW',CH,'8'
      RETURN
      END
```

L.4 CROD.FOR

```
C    GRAPHIC ROUTINES FOR THE BVT LOW-POWER PLANT MODEL
C
C    GRAPHICS USED IN CONTROL ROD-PANEL
C
C    RALPH MYRNAS    881220-890420
C
C*****************************************************************
      SUBROUTINE PANCROD
C
C   ROUTINE FOR DRAWING THE SEGMENTS IN THE CONTROL ROD PANEL
C
      CHARACTER ESC,KOORD(1:5),CH
      INTEGER Y
      ESC=CHAR(27)
      CALL KOD(1600,1600,KOORD)
      WRITE(*,*) ESC,'SP',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'MPO'
      WRITE(*,*) ESC,'MT2'
      WRITE(*,*) ESC,'ML2'
      WRITE(*,*) ESC,'SOA3'
C *-------------------------------------------------------
C LETTERS
      WRITE(*,*) ESC,'MQ1'
      WRITE(*,*) ESC,'MCF:J0C4'
      CALL KOD(2547,2520,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT1A'
      CALL KOD(547,2520,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT1B'
      CALL KOD(547,520,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT1C'
      CALL KOD(2547,520,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT1D'
      WRITE(*,*) ESC,'MCB7C;<'
      WRITE(*,*) ESC,'MQ2'
C *-------------------------------------------------------
C NUMBERS
      DO 10 Y=600,2600,200
       IF (Y.EQ.600 .OR. Y.EQ.2600) CH='5'
       IF (Y.EQ.800 .OR. Y.EQ.2400) CH='4'
       IF (Y.EQ.1000 .OR. Y.EQ.2200) CH='3'
       IF (Y.EQ.1200 .OR. Y.EQ.2000) CH='2'
       IF (Y.EQ.1400 .OR. Y.EQ.1800) CH='1'
       IF (Y.EQ.1600) CH='0'
       CALL KOD(Y,2750,KOORD)
       WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
       WRITE(*,*) ESC,'LT1',CH
       CALL KOD(420,Y,KOORD)
       WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
       WRITE(*,*) ESC,'LT1',CH
 10    CONTINUE
C *-------------------------------------------------------
C RED LINE
      CALL KOD(2400,2600,KOORD)
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'1'
      CALL KOD(800,2600,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(800,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(600,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(600,800,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(800,800,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
```

```
      CALL KOD(800,600,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2400,600,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2400,800,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2600,800,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2600,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2400,2400,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
C *-------------------------------------------------------
C BORDER 1
      CALL KOD(1700,2900,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(1700,1700,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(300,1700,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1500,1700,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(1500,300,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2900,1500,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(1500,1500,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
C *-------------------------------------------------------
C BORDER 2
      CALL KOD(1705,2900,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(1705,1695,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(300,1695,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1505,1700,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(1505,300,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2900,1495,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(1500,1495,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
C *-------------------------------------------------------
C BORDER 3
      CALL KOD(1695,2900,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(1695,1705,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(300,1705,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(1495,1700,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(1495,300,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(2900,1505,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(1500,1505,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'SC'
      WRITE(*,*) ESC,'SAA31!13'
      WRITE(*,*) ESC,'MP/'
      WRITE(*,*) ESC,'SOE0'
      CALL ROD(1600,1600,'E','0',0)
      CALL RUTA(3000,450,0)
      WRITE(*,*) ESC,'SC'
      WRITE(*,*) ESC,'SAE01!13'
      CALL GROUP(200,200,'C','1','/',1)
      CALL GROUP(200,0,'D','1','!',3)
```

```
      CALL GROUP(400,0,'C','3','/',5)
      CALL GROUP(600,200,'C','4','/',6)
      CALL GROUP(400,400,'C','5','/',7)
      CALL GROUP(400,200,'D','3','!',8)
      CALL GROUP(200,600,'C','6','/',9)
      CALL GROUP(600,0,'D','4','!',10)
      CALL GROUP(200,400,'D','5','!',11)
      CALL GROUP(800,0,'C','7','/',12)
      CALL GROUP(800,200,'D','6','!',13)
      CALL GROUP(600,400,'D','7','!',14)
      CALL GROUP(600,600,'C','8','/',15)
      CALL GROUP(400,600,'D','8','!',16)
      CALL GROUP(800,400,'C','9','/',17)
      CALL GROUP(400,800,'C',':','/',18)
      CALL GROUP(200,800,'D','9','!',19)
      CALL GROUP(200,1000,'C',';','/',20)
      CALL GROUP(400,1000,'D',':','!',21)
      CALL GROUP(600,1000,'C','<','/',22)
      CALL GROUP(600,800,'D',';','!',23)
      CALL GROUP(800,600,'D','<','!',24)
      CALL GROUP(800,800,'C','=','/',25)
      CALL GROUP(1000,400,'D','=','!',26)
      CALL GROUP(1000,600,'C','>','/',27)
      CALL GROUP(1000,200,'C','?','/',28)
      CALL GROUP(1000,0,'D','>','!',29)
      RETURN
      END
*****************************************************************
      SUBROUTINE GROUP(A,B,CH1,CH2,FARG,NB)
C
C   ROUTINE FOR DRAWING CONTROL ROD GROUPS
C
C   PARAMETERS   :    A,B      =  X,Y-COORDINATE ( FROM CENTER OF THE CORE )
C                     CH1,CH2  =  NAME OF SEGMENT
C                     FARG     =  COLOR INDEX
C                     NR       =  GROUPNUMBER
C
      CHARACTER ESC,CH1,CH2,FARG,CH
      INTEGER A,B,NB,NUMB
      ESC=CHAR(27)
      CH=CH2
      NUMB=NB
      WRITE(*,*) ESC,'SO',CH1,CH2
      WRITE(*,*) ESC,'MP',FARG
      CALL ROD(1600+A,1600+B,CH1,CH2)
      CALL ROD(1600-A,1600-B,CH1,CH2)
      IF (CH1.EQ.'C'.AND.CH2.EQ.'1')THEN
        CALL RUTA(3000,450+(ICHAR(CH)-48)*100,NB)
        CH='2'
        WRITE(*,*) ESC,'SC'
        WRITE(*,*) ESC,'SA',CH1,CH2,'1!13'
        WRITE(*,*) ESC,'MP',FARG
        WRITE(*,*) ESC,'SO',CH1,CH
        NUMB=NB+1
      ENDIF
      IF (CH1.EQ.'D'.AND.CH2.EQ.'1') THEN
        CALL RUTA(3600,400+(ICHAR(CH)-48)*100,NB)
        WRITE(*,*) ESC,'SC'
        WRITE(*,*) ESC,'SA',CH1,CH2,'1!13'
        CH='2'
        WRITE(*,*) ESC,'MP',FARG
        WRITE(*,*) ESC,'SO',CH1,CH
        NUMB=NB+1
      ENDIF
      CALL ROD(1600-B,1600+A,CH1,CH)
      CALL ROD(1600+B,1600-A,CH1,CH)
      IF (CH1.EQ.'D') THEN
        CALL RUTA(3600,400+(ICHAR(CH)-48)*100,NUMB)
      ELSE
        CALL RUTA(3000,450+(ICHAR(CH)-48)*100,NUMB)
      ENDIF
```

```
      WRITE(*,*) ESC,'SC'
      WRITE(*,*) ESC,'SA',CH1,CH,'1!13'
      RETURN
      END
*****************************************************************
      SUBROUTINE ROD(X,Y,CH1,CH2)
C
C   ROUTINE FOR DRAWING ONE ROD
C
C   PARAMETERS   :    X,Y      =  X,Y-COORDINATE
C                     CH1,CH2  =  NOT USED
C
      CHARACTER ESC,CH1,CH2,KOORD(1:5)
      INTEGER X,Y
      ESC=CHAR(27)
      CALL KOD(X-80,Y-80,KOORD)
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'0'
      CALL KOD(X+80,Y-80,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(X+80,Y+80,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(X-80,Y+80,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
C      CALL KOD(X-40,Y-26,KOORD)
C      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
C      WRITE(*,*) ESC,'LT2',CH1,CH2
      RETURN
      END
*****************************************************************
      SUBROUTINE RUTA(X,Y,NB)
C
C   ROUTINE FOR DRAWING THE SQUARES WERE THE GROUPNUMBER AND CURRENT
C   DISPLACEMENT ARE DISPLAYED
C
C   PARAMETERS   :    X,Y      =  X,Y-COORDINATE ( FROM CENTER OF THE CORE )
C                     NB       =  GROUPNUMBER
C
      CHARACTER ESC,KOORD(1:5),CRD(1:5)
      INTEGER X,Y,NB
      ESC=CHAR(27)
      CALL KOD(X,Y,KOORD)
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'1'
      CALL KOD(X+200,Y,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(X+200,Y+80,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(X,Y+80,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
      CALL KOD(X+200,Y,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      CALL KOD(X+400,Y,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(X+400,Y+80,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(X+200,Y+80,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
C      CALL KOD(X+200,Y,KOORD)
C      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL DECIKOD(NB,CRD)
      CALL KOD(X+60,Y+14,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT2',(CRD(K),K=4,5)
      RETURN
      END
*****************************************************************
      SUBROUTINE UPDATEROD(NR,PROC,I)
C
C   ROUTINE FOR WRITING CURRENT DISPLACEMENT FOR A GIVEN GROUP
C
```

```
C   PARAMETERS  :    NR    =  GROUPNUMBER
C                    PROC  =  DISPLACEMENT
C                    I     =  INDICATES RUBBING OF THE WINDOW
C
      CHARACTER ESC,CHRPROC(1:5),KOORD(1:5)
      INTEGER X,Y,NR,V(1:24,1:2),HLT,I
      REAL PROC
      DATA V(1,1),V(2,1),V(3,1),V(4,1)/ 3220,3220,3220,3220 /
      DATA V(5,1),V(6,1),V(7,1),V(8,1)/ 3220,3220,3220,3220 /
      DATA V(9,1),V(10,1),V(11,1),V(12,1)/ 3220,3220,3820,3820 /
      DATA V(13,1),V(14,1),V(15,1),V(16,1)/ 3820,3820,3820,3820 /
      DATA V(17,1),V(18,1),V(19,1),V(20,1)/ 3820,3820,3820,3820 /
      DATA V(21,1),V(22,1),V(23,1),V(24,1)/ 3820,3820,3220,3220 /
      DATA V(1,2),V(2,2),V(3,2),V(4,2)/ 460,660,760,860 /
      DATA V(5,2),V(6,2),V(7,2),V(8,2)/ 960,1060,1160,1260 /
      DATA V(9,2),V(10,2),V(11,2),V(12,2)/ 1360,1460,610,710 /
      DATA V(13,2),V(14,2),V(15,2),V(16,2)/ 810,910,1010,1110 /
      DATA V(17,2),V(18,2),V(19,2),V(20,2)/ 1210,1310,1410,1510 /
      DATA V(21,2),V(22,2),V(23,2),V(24,2)/ 1610,1810,1660,1860 /
      ESC=CHAR(27)
      X=V(NR,1)
      Y=V(NR,2)
      IF (I.EQ.1) THEN
        WRITE(*,*) ESC,'MP0'
        CALL KOD(X,Y,KOORD)
        WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'0'
        CALL KOD(X+150,Y,KOORD)
        WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
        CALL KOD(X+150,Y+60,KOORD)
        WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
        CALL KOD(X,Y+60,KOORD)
        WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
        WRITE(*,*) ESC,'LE'
      ENDIF
      HLT=PROC
      CALL DECIKOD(HLT,CHRPROC)
      CALL KOD(X+2,Y+2,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'MT5'
      WRITE(*,*) ESC,'LT3',(CHRPROC(K),K=3,5)
      RETURN
      END
*****************************************************************
      SUBROUTINE RODCOMSEG(X,Y,CH1,CH2,FARG)
C
C   ROUTINE FOR DRAWING CONTROL RODS COMMAND SEGMENTS
C
C   PARAMETERS  :    X,Y     =  XY-COORDINATE
C                    CH1     =  NAME OF SEGMENT
C                    CH2     =  CHARACTER TO BE WRITTEN INSIDE SEGMENT
C                    FARG    =  COLORINDEX
C
      CHARACTER ESC,KOORD(1:5),CH1,CH2,FARG
      INTEGER X,Y
      ESC=CHAR(27)
      WRITE(*,*) ESC,'MT?'
      WRITE(*,*) ESC,'ML?'
      WRITE(*,*) ESC,'MP',FARG
      WRITE(*,*) ESC,'SOZ',CH1
      CALL KOD(X,Y,KOORD)
      WRITE(*,*) ESC,'LP',(KOORD(K),K=1,5),'1'
      CALL KOD(X+100,Y,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(X+100,Y+100,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      CALL KOD(X,Y+100,KOORD)
      WRITE(*,*) ESC,'LG',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LE'
      CALL KOD(X+30,Y+30,KOORD)
      WRITE(*,*) ESC,'LF',(KOORD(K),K=1,5)
      WRITE(*,*) ESC,'LT1',CH2
```

```
      WRITE(*,*) ESC,'SC'
      WRITE(*,*) ESC,'SAZ',CH1,'1!13'
      RETURN
      END
*****************************************************************
      INTEGER FUNCTION GROUPNR(CH2,CH)
C
C   GIVES THE CORRESPONDING GROUPNUMBER TO A SEGMENT CH2,CH
C
C
      CHARACTER CH,CH2
      IF (CH2.EQ.'#'.AND.CH.EQ.'1') GROUPNR=1
      IF (CH2.EQ.'#'.AND.CH.EQ.'2') GROUPNR=2
      IF (CH2.EQ.'#'.AND.CH.EQ.'3') GROUPNR=3
      IF (CH2.EQ.'#'.AND.CH.EQ.'4') GROUPNR=4
      IF (CH2.EQ.'#'.AND.CH.EQ.'5') GROUPNR=5
      IF (CH2.EQ.'#'.AND.CH.EQ.'6') GROUPNR=6
      IF (CH2.EQ.'#'.AND.CH.EQ.'7') GROUPNR=7
      IF (CH2.EQ.'#'.AND.CH.EQ.'8') GROUPNR=8
      IF (CH2.EQ.'#'.AND.CH.EQ.'9') GROUPNR=9
      IF (CH2.EQ.'#'.AND.CH.EQ.':') GROUPNR=10
      IF (CH2.EQ.'#'.AND.CH.EQ.';') GROUPNR=23
      IF (CH2.EQ.'#'.AND.CH.EQ.'<') GROUPNR=23
      IF (CH2.EQ.'#'.AND.CH.EQ.'=') GROUPNR=24
      IF (CH2.EQ.'#'.AND.CH.EQ.'?') GROUPNR=24
      IF (CH2.EQ.'$'.AND.CH.EQ.'1') GROUPNR=11
      IF (CH2.EQ.'$'.AND.CH.EQ.'2') GROUPNR=11
      IF (CH2.EQ.'$'.AND.CH.EQ.'3') GROUPNR=12
      IF (CH2.EQ.'$'.AND.CH.EQ.'4') GROUPNR=13
      IF (CH2.EQ.'$'.AND.CH.EQ.'5') GROUPNR=14
      IF (CH2.EQ.'$'.AND.CH.EQ.'6') GROUPNR=15
      IF (CH2.EQ.'$'.AND.CH.EQ.'7') GROUPNR=16
      IF (CH2.EQ.'$'.AND.CH.EQ.'8') GROUPNR=17
      IF (CH2.EQ.'$'.AND.CH.EQ.'9') GROUPNR=18
      IF (CH2.EQ.'$'.AND.CH.EQ.':') GROUPNR=19
      IF (CH2.EQ.'$'.AND.CH.EQ.';') GROUPNR=20
      IF (CH2.EQ.'$'.AND.CH.EQ.'<') GROUPNR=21
      IF (CH2.EQ.'$'.AND.CH.EQ.'=') GROUPNR=22
      IF (CH2.EQ.'$'.AND.CH.EQ.'?') GROUPNR=22
      IF (CH2.EQ.'%'.AND.CH.EQ.'0') GROUPNR=1
      RETURN
      END
```

L.5 NEWUPP.T

```
MACRO NEWUPP "HJALPPROGRAM FOR ATT UNDERLATTA ANVANDNING AV KOMPAKTSIMULATORN
             "870731/LSE

        DEFAULT TK.=0

        IF TK. EQ 1 GOTO BEGIN      "OM TERMINAL 2 REDAN AR DEFINIERAD !
        SWITCH DIS 2 T4100          "ANGER ATT TERMINAL 2 AR EN TEKTRONIXTERMINAL
        LET TK.=1                   "SA MAN SLIPPER KOMPILERING I ONODAN !

LABEL KOMPIL
        BLANK                       "RENSAR BILDSKARMEN

        WRITE'VANTA CIRKA 40 SEKUNDER, KOMPILERING PAGAR !!'

        SYST RE CR FC PC FL RK LCONS    "KOMPILERAR LAGEFFEKTMODELLEN
        LVISAA SIM                      "LAGRING AV VARIABLER FRAN MACRO LVISAA

LABEL BEGIN
       LET SIMSTAT=0

LABEL MENY              "VAL AV ARBETSSATT
       SWITCH GRAPH OFF "ALFANUMERISK MOD
       BLANK            "RADERA SKARMEN
       WRITE' '
       WRITE' '
       WRITE'           **** BVT PROCESS-SIMULATOR ****'
       WRITE' '
       WRITE' '
       WRITE'       * HUVUDMENY , LAGEFFEKTMODELLEN BSO * '
       WRITE' '
       WRITE'           0.  - AVSLUTA '
       WRITE' '
       WRITE'           1.  - UTFORA SIMULERING MED NYTT UTGANGSLAGE '
       WRITE' '
       WRITE'           2.  - FORTSATTA SENASTE SIMULERINGEN '
       WRITE' '
       WRITE'           3.  - VISA SIMULERINGSRESULTAT'
       WRITE' '
       WRITE'            ******************************'
       WRITE' '
       WRITE'           Ange onskat alternativ: (0-3)'
       WRITE' '
       READ MVAL INT

       IF MVAL EQ 0 GOTO SLUT
       , , , 1 , ULVAL1
       , , , 2 , PARSET
       , , , 3 , VISA

       WRITE 'Felslagning ! Svara 0, 1, 2 eller 3 '
       GOTO MENY

LABEL ULVAL1              "VAL AV UTGANGSLAGE FOR SIMULERING
       BLANK
   WRITE ' '
   , '    *** BVT PROCESS-SIMULATOR ,LAGEFFEKTMODELLEN BSO ***'
   , ' '
   , '         MOJLIGA UTGANGSLAGEN FOR START AV SIMULERING '
   , ' '
   , '  Nr     Summa SS-monster  Reaktortryck  Reaktoreffekt  Moderatortemp'
   , '               %              Bar          MWt            Grad C'
   , ' ------------------------------------------------------------------'
   , '  1 :          0              1           5.0E-7         56'
   , '  2 :          3300           15.6        2.0E-6         200'
   , '  3 :          3520           15.6        5.0E-6         200'
   , '  4 :          3735           15.6        2.0E-4         200'
   , '  5 :          4237           40          1.7            250'
   , '  6 :          4554           60          1.84           275'
```

```
   , ' 7 :          4673          40          17          249'
   , ' 8 :          5505          65          68          277'
   , ' 9 :          5820          70          102         286'
   ,' '
   ,'   Fran vilket utgangslage vill du borja ?'
   ,'   Knappa in onskat utgangslage (1-9)     '

   READ UX INT

   IF UX LT 1 GOTO MENY
   , , GT 9 , ,

   WRITE' '
   ,'INLASNING AV INITIALVARDEN FOR UTGANGSLAGE NUMMER ' UX ' PAGAR !'
   ,' '
   LET INITX=INITO+UX
   GET INITX
   PAR PRINT[RE]:0                    "INGEN UTSKRIFT I RE
   PAR PRINT[CR]:0                    "  -       -    I CR
   LET SIMSTAT=1

LABEL PARSET
   IF SIMSTAT EQ 0 GOTO ULVAL1
   NYLAND UX                          "GOR PARAMETERINSTALLNINGAR
   GOTO NASTA

LABEL SIMU1
   WRITE' '
   ,' ANGE NAMN PA DATALAGRINGSFIL !'
   ,' EN BOKSTAV FOLJT AV UPP TILL SJU SIFFROR ELLER BOKSTAVER  '
   ,' '
   READ FILX NAME
   IF SIMSTAT EQ 2 GOTO SIMUN
   WRITE' '
   ,'ANGE ONSKAD SIMULERINGSTID I SEKUNDER !'
   ,'SIMULERINGEN SKER MED UNGEFAR VERKLIG TID !'
   ,' '
   READ T1 INT
   SWITCH GRAPH ON
   SIMU 0 T1 0.25/FILX
   LET SIMSTAT=2
   GOTO MENY

LABEL SIMUN                       "FORTSATT STOPPAD SIMULERING
   WRITE' '
   ,'ANGE ONSKAD STOPP-TID I SEKUNDER !'
   ,' '
   READ T2 INT
   BLANK
   SIMU T1 T2 -CONT/FILX
   LET T1=T2
   GOTO MENY

LABEL VISA
   LVISAA VIS
   GOTO MENY

LABEL SLUT
   BLANK
   WRITE'FOR ATT STOPPA SIMNON SKRIV  :  STOP  '
   ,' '
   ,'GLOM INTE ATT LOGGA UR DA DU AR KLAR FOR DAGEN !'
   ,'SKRIV  :  STOP '
   END
```

MACRON SOM ERFORDRAS TILL UPP.T

LAND.T FOR ATT KUNNA ANDRA PARAMETRAR

LCONS.T CONNECTING SYSTEM

INIT01-INIT09.T	INITIALVARDESFILER
LVISAA.T	FOR ATT LAGRA OCH PRESENTERA VARIABLER
BLANK.T	

L.6 NYLAND.T

```
MACRO NEWLAND UL              "861113/TYN
                              "MODIFIERAD 861229/LSE
                              "MODIFIERAD 890321/RMY

LABEL START
      goto ssul0
LABEL SSVALX
SWITCH DIS 1
       write' '
,' 0     INGA STYRSTAVSANDRINGAR ONSKAS'
, ' '
       ,' 1      EN (1) STYRSTAVSGRUPP SKALL MANOVRERAS '
       ,' '
       ,' 2      ETT ANTAL STYRSTAVSGRUPPER SKALL MANOVRERAS '
       ,' '
       ,' ANGE ONSKAT ALTERNATIV : '
       ,' '
       read ssval int

       if ssval lt 0 goto SSVALX
       ,, gt 2 ,,
       if ssval eq 1 goto sslab1
       ,,, 2 , sslab2
goto slut

label sslab1
par sti[re]:101
       par sti[cr]:101
       par sti[fl]:101


      write'Vilken styrstavsgrupp vill du manovrera ?'
          ,'Ange onskat gruppnummer : 1-24'
          ,' '
      read ssgr1 int

      if ssgr1 lt 1 goto sslab1
      ,, gt 24 ,,

      let summass=pcric+ssgr1

           par grp:ssgr1

      write' '
          ,'Hur manga % skall grupp'ssgr1' manovreras ?     '
          ,'(Negativt varde = inkorning !'
          ,' '
      read ssforfl int

      par fcr:ssforfl
      par vcr:100

      write' '
          ,'Styrstavsgrupp nr 'ssgr1' kommer att forflyttas 'ssforfl' % !'
          ,' '
TF 5
      goto slut

label sslab2
      write'Fran vilken styrstavsgrupp skall Du borja manovrera ?'
          ,'Ange onskat gruppnummer : 1-24'
          ,' '
      read ssgr2 int

      if ssgr2 lt 1 goto sslab2
      ,, gt 24 ,,

     par grp:ssgr2
```

```
      par sti[re]:110
      par sti[cr]:110
      par sti[fl]:110

      write'Ange det antal styrstavsgrupper som skall manovreras :   '
          ,' '
      read antgr int
      par ngr:antgr

      write'Ange den totala styrstavsinsatsen i summa % :     '
          ,' '
      read sstot int

      par fcr:sstot

      write'Antal grupper som skall manovreras = 'antgr''
          ,'Manovern skall starta med grupp = 'ssgr2''
          ,'Den totala styrstavsinsatsen blir ='sstot''
          ,' '
      TF 5
goto slut

LABEL SSULO
       switch dis 2
       switch graph on      "for att fa bort storande bakgrundstext m.m.
       switch graph off

write' '
    ,' '
    ,' '
    ,'  Styrstavsgrupp  Antal stavar  Stavlage % ute '
    ,' ------------------------------------------------'
    ,'         1              1              0     '
    ,'         2              1              0    '
    ,'         3              1              'sg3' '
    ,'         4              2              0    '
    ,'         5              2              0    '
    ,'         6              2              0    '
    ,'         7              4              0    '
    ,'         8              4              0    '
    ,'         9              4              0    '
    ,'         10             4              0    '
    ,'         11             4              0    '
    ,'         12             4              'sg12''
    ,'         13             4              'sg13''
    ,'         14             4              'sg14''
    ,'         15             4              'sg15''
    ,'         16             4              'sg16''
    ,'         17             4              'sg17''
    ,'         18             8              'sg18''
    ,'         19             8              'sg19''
    ,'         20             8              'sg20''
    ,'         21             8              'sg21''
    ,'         22             8              'sg22''
    ,'         23             8              0    '
    ,'         24             8              0    '
goto SSVALX

label slut

end
```

L.7 LCONS.T

```
CONNECTING SYSTEM LCONS

TIME T

" UNIVERSAL CONNECTING SYSTEM FOR BARSEBAECK LOW POWER SIMULATOR.
" 860617/LSE
" MODIFIERAD RM

"** REAKTORMODULEN  (ROUTINE 'RE') **

 WLR[RE]=WSL[PC]
 WFE[RE]=WFE[FC]
 XSKD[RE]= IF SS5[FL]>0 THEN 0 ELSE XSKD
"RM XSKD[RE]= XSKD
 TREK[RE]=TRO[RK]
 WREK[RE]=WREK
 WLOS[RE]= IF WLEV[RE] > 4.2 AND DRAIN > 0 THEN 5.0 ELSE 0
 PCCR1[RE]=PCR1[CR]
 PCCR2[RE]=PCR2[CR]
 PCCR3[RE]=PCR3[CR]
 PCCR4[RE]=PCR4[CR]
 PCCR5[RE]=PCR5[CR]
 PCCR6[RE]=PCR6[CR]
 PCCR7[RE]=PCR7[CR]
 PCCR8[RE]=PCR8[CR]
 PCCR9[RE]=PCR9[CR]
 PCCR10[RE]=PCR10[CR]
 PCCR11[RE]=PCR11[CR]
 PCCR12[RE]=PCR12[CR]
 PCCR13[RE]=PCR13[CR]
 PCCR14[RE]=PCR14[CR]
 PCCR15[RE]=PCR15[CR]
 PCCR16[RE]=PCR16[CR]
 PCCR17[RE]=PCR17[CR]
 PCCR18[RE]=PCR18[CR]
 PCCR19[RE]=PCR19[CR]
 PCCR20[RE]=PCR20[CR]
 PCCR21[RE]=PCR21[CR]
 PCCR22[RE]=PCR22[CR]
 PCCR23[RE]=PCR23[CR]
 PCCR24[RE]=PCR24[CR]


"** STYRSTAVSMANOVRERING  (ROUTINE 'CR') **

 QN[CR]=QND[RE]
 PE[CR]=PE[RE]
 WGE[CR]=WGE[RE]
 SS[CR]=SS[FL]
"RM CRS[CR]= IF E[FL] + S[FL] > 0 THEN 1 ELSE 0
 CRS[CR]= 0
 DBT[CR]=DBTF[FL]
 RT[CR]=TT[RE]


"** HJALPMATARVATTENSYSTEMET  (ROUTINE 'FC') **

 WLEV[FC]=WLEV[RE]
 PE[FC]=PE[RE]
 WLR[FC]=WSL[PC]
 SS[FC]=SS[FL]
"RM ASKH[FC]= IF SS5[FL]>0 THEN 0 ELSE ASKH
 ASKH[FC]= ASKH
 P12[FC]=P12
"RM: KONTROLLEN SKOTES MANUELLT

"** TRYCKREGLERSYSTEMET  (ROUTINE 'PC') **
```

```
 PE[PC]=PE[RE]
 QN[PC]=QND[RE]
"RM PER[PC]=IF PER[CR] > 8 THEN PER[CR] ELSE PER
 PER[PC]=  PER
"RM XSKD[PC]= IF SS5[FL] > 0 THEN 0 ELSE XSKD
 XSKD[PC]= XSKD
"RM DUMP[PC]= IF SS11[FL] > 0 THEN 0 ELSE DUMP
 DUMP[PC]= DUMP
"RM A314[PC]=IF SS5[FL]*PRML[FL]+SS6[FL]+SS11[FL] >0 THEN 1 ELSE A314
 A314[PC]= A314
 I314[PC]=IF SS5[FL]*PRML[FL]+SS6[FL]+SS11[FL] >0 THEN 1 ELSE I314
"RM WLX[PC]= MIN(WLX * T/WLXT, WLX)
 WLX[PC]= WLX


"** NEUTRONFLODESMATSYSTEMET  (ROUTINE 'FL') **

 FLUX[FL]=FLUX[RE]
 WLEV[FL]=WLEV[RE]
 PE[FL]=PE[RE]
 DBT[FL]=DBT[RE]
 XXX[FL]=SIRMD
 YYY[FL]=SIRMD

"** RESTEFFEKTKYLSYSTEMET  (ROUTINE 'RK') **

 TRI[RK]=TPUI[RE]
 WREK[RK]=WREK

ASKH:1      "INDIKATOR FOR 327-SKALVENTILLAGE              (0,1)
XSKD:0      "LAGE PA 311-SKALVENTILERNA                    (0-1)
PER:40      "TRYCKBORVARDE                                 (BAR)
WREK:0      "FLODE GENOM 321E1,E2                          (KG/S)
DUMP:0      "INDIKATOR FOR TRYCKREGLERING VIA 451 V9,V11   (0,1)
A314:1      "    -       -        -          -  314 V48,V49 (0,1)
I314:0      "INDIKATOR FOR UTLOST 'START 314'              (0,1)
WLX:0       "ANGUTTAG TILL PROCESSEN                       (KG/S)
WLXT:1      "   -       -       -      RAMPTID (S)  (WLXT>0)
P12:1       "INDIKATOR FOR DRIFT AV 327-PUMPARNA           (0,1)
DRAIN:0     "INDIKATOR FOR DRANERING AV REAKTORN OM NIVAN > 4.2 M (0,1)
SIRMD:1.    "DETEKTORLAGE SIRM 1=INNE  ** SIRM = SRM+IRM  /RM

 END
```