JAERI-Data/Code
97-042

核融合炉燃料循環系動特性コードの開発
— 単パルス運転の結果 —

1997年11月

青木　功・関　泰・佐々木誠*
新谷清憲*・Yeong-Chan Kim*

日本原子力研究所
Japan Atomic Energy Research Institute

本レポートは，日本原子力研究所が不定期に公刊している研究報告書です。
入手の問合わせは，日本原子力研究所研究情報部研究情報課（〒319-11 茨城県那珂郡東海村）あて，お申し越しください。なお，このほかに財団法人原子力弘済会資料センター（〒319-11 茨城県那珂郡東海村日本原子力研究所内）で複写による実費頒布をおこなっております。

This report is issued irregularly.
Inquiries about availability of the reports should be addressed to Research Information Division, Department of Intellectual Resources, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken, 319-11, Japan.

編集兼発行　日本原子力研究所
印　　刷　いばらき印刷㈱

JAERI-Data/Code 97-042

核融合炉燃料循環系動特性コードの開発
― 単パルス運転の結果 ―

日本原子力研究所那珂研究所核融合工学部
青木　功・関　泰・佐々木　誠*
新谷　清憲*・Yeong-Chan Kim*

(1997年10月2日受理)

核融合実験炉の燃料循環のシミュレーションコードを作成した。本コードはパルス運転時のプラズマチェンバ及び燃料循環系内に分布する燃料の時間変化を追跡する。プラズマチェンバ及び燃料循環系内における燃料の燃焼、排気、精製、供給の機能を時間当たりの処理量に着目しその時間変化を追跡した。プラズマチェンバ及び燃料循環系各サブシステム毎に状態方程式と出力方程式を定め、燃料の燃焼、排気、精製、供給の機能をモデル化し、時間に関し定常となるサブシステムの定数は、ＩＴＥＲの概念設計報告書に依拠した。本コードを用いて、燃焼状態と燃料循環系サブシステムの処理機能とに依存する供給量の時間変化と、滞留量の時間変化を示した。

那珂研究所：〒311－01　茨城県那珂郡那珂町向山801－1
＊（株）日本総合研究所

JAERI-Data/Code 97-042

Development of Dynamic Simulation Code for Fuel Cycle of Fusion Reactor
— I. Single Pulse Operation Simulation —

Isao AOKI, Yasushi SEKI, Makoto SASAKI*,
Kiyonori SHINTANI* and Yeong-Chan Kim*

Department of Fusion Engineering Research
Naka Fusion Research Establishment
Japan Atomic Energy Research Institute
Naka-machi, Naka-gun, Ibaraki-ken

(Received October 2, 1997)

A dynamic simulation code for the fuel cycle of a fusion experimental reactor has been developed. The code follows the fuel inventory change with time in the plasma chamber and the fuel cycle system during a single pulse operation. The time dependence of the fuel inventory distribution is evaluated considering the fuel burn and exhaust in the plasma chamber, purification and supply functions. For each subsystem of the plasma chamber and the fuel cycle system, the fuel inventory equation is written based on the equation of state considering the function of fuel burn, exhaust, purification, and supply. The processing constants of subsystem for the steady states were taken from the values in the ITER Conceptual Design Activity (CDA) report. Using the code, the time dependence of the fuel supply and inventory depending on the burn state and subsystem processing functions are shown.

Keywords: Simulation Code, Fuel Cycle, Fusion Experimental Reactor, Tritium Inventory, Single Pulse Operation, ITER CDA

* The Japan Research Institute, Limited

目　　次

Contents

1. はじめに

核融合実験炉の燃料循環系は、系内における燃料の滞留量がプラズマ燃焼に費やされる量の数十倍に達する。運転シナリオにおいて系内のサブシステムにおけるこれらの燃料の滞留量の時間変化を把握することが安全設計対応から求められる重要な課題の一つとなっている。各サブシステムの燃料の滞留量は、サブシステムの処理性能に依存する。燃料循環系が包含するトリチウムインベントリーを議論するときに、これらの処理量を計算モデル化し、滞留量に及ぼす影響の度合いを定量的に評価することが求められる。本コードの開発作業は、核融合実験炉の燃料循環系を含めた燃料処理体系の中でのトリチウムのインベントリ変化の評価に資するため、変化の挙動を与えるための簡易計算モデルの作成を目的としている。このために、次ぎの手順により全体計画を実施するものとした。

1) 単パルス運転シミュレーション
2) 2日連続パルス運転
3) トリチウムインベントリに対する各サブシステム定数の寄与度

本報告書は、全体計画の中の手順1)の作業経緯を纏めたものである。作業内容は、単パルス運転の燃焼状態を入力（負荷）とした時の燃料循環系各サブシステムの応答を見るために、各サブシステムのシステム変数を定義することと、このシステム変数に関する時間変化の簡易計算モデルを記述し、そのシミュレーションモデルを作成することである。計算モデル化されたシステム構成を第2章に示した。計算モデル化された各サブシステムの機器構成とそれらの相互関係を図で表し、模擬するシステム変数との位置関係を示した。また、第2章において、サブシステム間における入力と出力との相互関連を状態方程式の形で示し、時系列上の推移を明示してシミュレーションモデルの定め方を示した。

2. 動特性モデルの開発とシミュレーションモデルの構成

2.1 動特性モデル開発の方針と方法

2.1.1 動特性モデル開発の方針

プラズマチェンバ及び燃料循環系の各サブシステムの燃焼、排気、精製、供給に関する処理性能を、サブステム毎に時間依存の状態方程式で表した。これらの方程式を時間に伴って変化する燃焼量と連結させることにより、燃焼量の変化に伴い、各サブシステム間の処理性能の相違によって引き起こされるサブシステムに滞留する量（インベントリ）を評価した。また、各サブシステム間の相関量を各変数間に引数の形で引き渡し、サブシステム間のトリチウム、重水素、水素の分布の変化を求めた。燃料循環時における各サブシ

1. はじめに

核融合実験炉の燃料循環系は、系内における燃料の滞留量がプラズマ燃焼に費やされる量の数十倍に達する。運転シナリオにおいて系内のサブシステムにおけるこれらの燃料の滞留量の時間変化を把握することが安全設計対応から求められる重要な課題の一つとなっている。各サブシステムの燃料の滞留量は、サブシステムの処理性能に依存する。燃料循環系が包含するトリチウムインベントリーを議論するときに、これらの処理量を計算モデル化し、滞留量に及ぼす影響の度合いを定量的に評価することが求められる。本コードの開発作業は、核融合実験炉の燃料循環系を含めた燃料処理体系の中でのトリチウムのインベントリ変化の評価に資するため、変化の挙動を与えるための簡易計算モデルの作成を目的としている。このために、次ぎの手順により全体計画を実施するものとした。

1) 単パルス運転シミュレーション
2) 2日連続パルス運転
3) トリチウムインベントリに対する各サブシステム定数の寄与度

本報告書は、全体計画の中の手順1)の作業経緯を纏めたものである。作業内容は、単パルス運転の燃焼状態を入力（負荷）とした時の燃料循環系各サブシステムの応答を見るために、各サブシステムのシステム変数を定義することと、このシステム変数に関する時間変化の簡易計算モデルを記述し、そのシミュレーションモデルを作成することである。計算モデル化されたシステム構成を第2章に示した。計算モデル化された各サブシステムの機器構成とそれらの相互関係を図で表し、模擬するシステム変数との位置関係を示した。また、第2章において、サブシステム間における入力と出力との相互関連を状態方程式の形で示し、時系列上の推移を明示してシミュレーションモデルの定め方を示した。

2. 動特性モデルの開発とシミュレーションモデルの構成

2.1 動特性モデル開発の方針と方法

2.1.1 動特性モデル開発の方針

プラズマチェンバ及び燃料循環系の各サブシステムの燃焼、排気、精製、供給に関する処理性能を、サブステム毎に時間依存の状態方程式で表した。これらの方程式を時間に伴って変化する燃焼量と連結させることにより、燃焼量の変化に伴い、各サブシステム間の処理性能の相違によって引き起こされるサブシステムに滞留する量（インベントリ）を評価した。また、各サブシステム間の相関量を各変数間に引数の形で引き渡し、サブシステム間のトリチウム、重水素、水素の分布の変化を求めた。燃料循環時における各サブシ

ステムの処理機能は、その設備が保有する機器の設計条件、運転条件によって定まる。時間に関し定数となるこれらのシステム定数は、ＩＴＥＲの概念設計報告書に依拠した[1]。

2.1.2 動特性モデルの開発の方法

(1) 動特性モデルの作成

a. 動特性モデルの作成

体系の動特性をシミュレートするモデルを１階定常微分方程式にて記述する。燃料循環系各サブシステムの運転状態を

状態方程式

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{2.1}$$

及び出力方程式

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \tag{2.2}$$

により記述する。　ここで、ベクトル変数$\mathbf{x}(t) = (x1, x2, x3)$の各要素は、$x1$＝トリチウム、$x2$＝重水素、$x3$＝水素である。**A**，**B**，**C**，**D**はシステムの状態によって定まる行列であり、時間依存のときは非定常となるが、ここでは定常の場合のみ取り扱う。定義式(2.1)、(2.2)で表されるベクトル変数$\mathbf{x}(t)$と、システムの状態ベクトル**A**，**B**，**C**，**D**との関連をブロック線図でFig. 2.1に示す。

b. 動特性モデルの解法[3]、[4]

時間領域に対する解$\mathbf{x}(t)$は次式で与えられる。

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}_0 + \int_0^t e^{\mathbf{A}(t-\eta)}\mathbf{B}\mathbf{u}(\eta)d\eta \tag{2.3}$$

これより、$x1$、$x2$、$x3$の時間依存の振る舞いを求める。

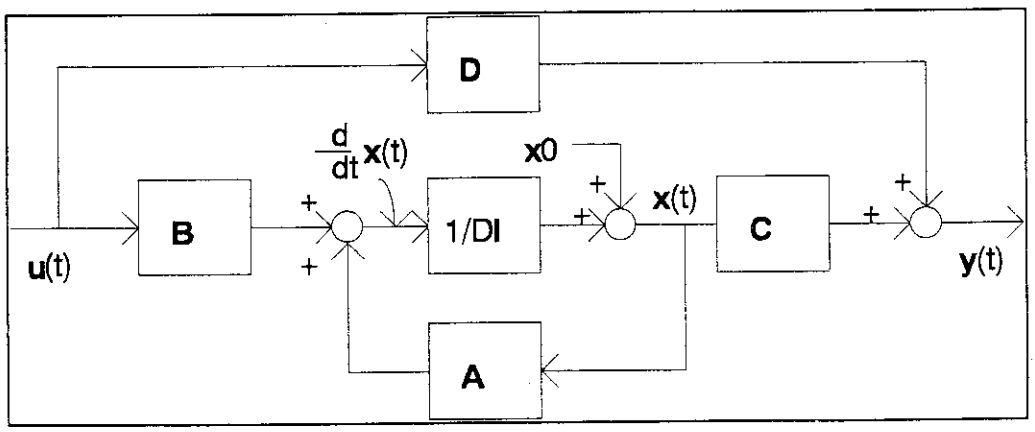


Fig. 2.1 Schematic diagram of the way to establish the dynamic equations

2.2 シミュレーションモデルの構成

燃料循環系を循環する燃料（トリチウム、重水素）の時間変化を模擬するために、燃料の処理機能を、燃焼、排気、精製、分離、供給に区分した。燃料循環系各サブシステムの構成を、プラズマチェンバ（コアプラズマ、プラズマエッジ、プラズマ対向機器）、真空排気、燃料精製、同位体分離、供給（ガスパフ、ペレット入射、中性粒子入射）、貯蔵・補給各設備から成るものとし、各々の処理機能を計算モデル化した。これにより、各サブシステムのインベントリの時間変化と処理量、並びに各サブシステム間に引き渡される燃料流量の時間変化が模擬される。計算モデル化された循環系各サブシステムの構成と燃料の流れをFig. 2.2に示す。

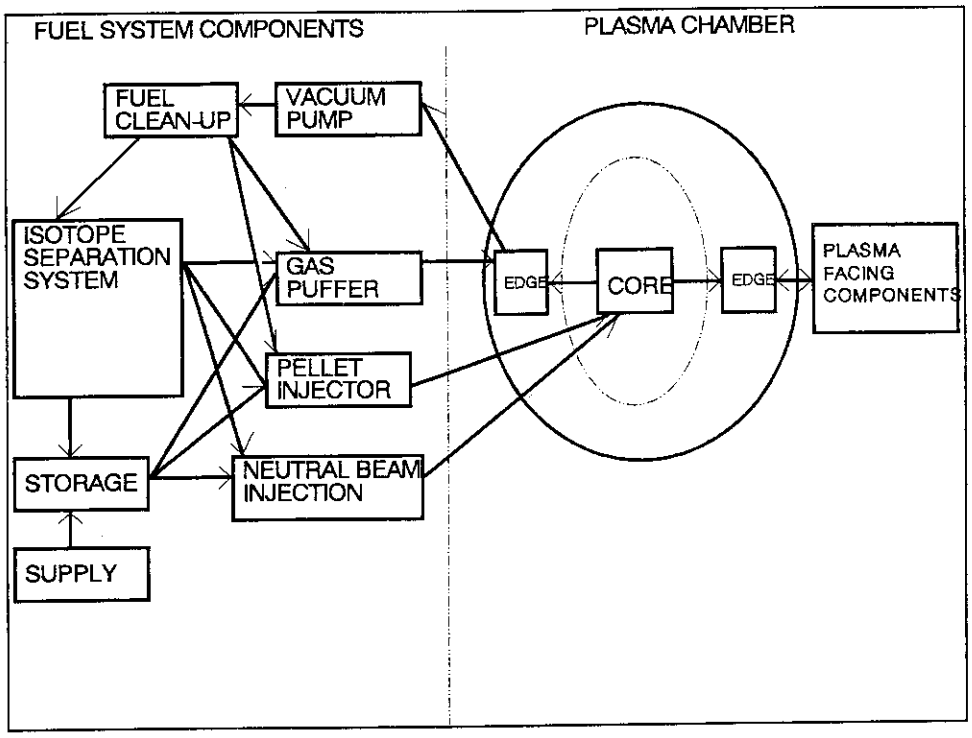


Fig. 2.2 Fuel flow cycle model

2.3 各サブシステムのシミュレーションモデル

2.3.1 システム変数の定め方とシミュレーション結果

(1) プラズマ燃焼と運転シナリオ

プラズマチェンバ内の燃料（トリチウム）インベントリは、高々グラムオーダーであるが、プラズマチェンバ内の燃焼速度の大きさが循環系各サブシステムの処理速度の決定要因となっている。また、各サブシステムの処理機能はそのシステム固有のシステム定数に

より定まるため、燃焼速度の大きさと連結してその時間変化を追跡することにより、サブシステム固有の処理機能の燃焼速度による影響の度合いを動的に評価できる。本モデルにおいては、プラズマチェンバの運転パターンをパルス運転とし、その時間変化を台形で模擬し、シミュレーションの入力条件とした。模擬されたパルス運転を、Fig. 2.3に示す。トリチウムの燃焼速度は、核融合出力により定まる。核融合出力はFig. 2.3により与えられるものとし、その速度$Burn_t(t)$を次式で定義した。

$$Burn\_t(t) = f_{power} \cdot \kappa_p P_{fusion} \frac{1}{\kappa_E E_f \cdot N_0} (\frac{mole}{h}) \qquad (2.4)$$

$$= 2.12 f_{power} (\frac{mole}{GW \cdot h})$$

ここで、

f_{power} : normalized factor of simulated pulse operation

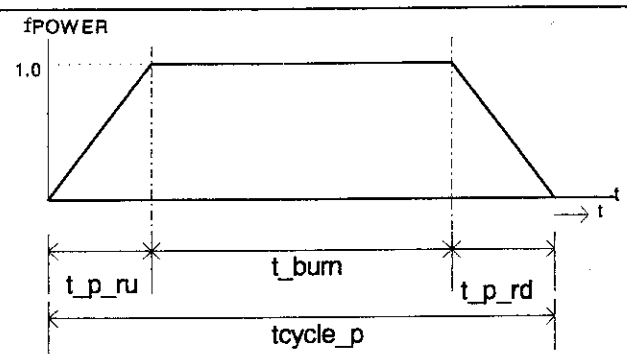
P_{fusion} : related power(1000MW)

E_f : D,T fusion energy（17.5MeV)

κ_p、κ_E : conversion factor

N_0 : Avogadro's Number

である。



t_burn: plasma burn time (0.17h)

tcycle_p: cycle time for plasma (0.28h)

t_p_ru: heating (0.056h)

t_p_rd: burn termination (0.056h)

Fig. 2.3 Simulated plasma burn cycle

(2) プラズマとプラズマ対向機器との相互作用

プラズマチェンバ内の燃料の消費を燃焼とプラズマ対向機器との相互作用に分けてモデル化した。プラズマ対向機器のインベントリ変化を、体積吸蔵 (bulk uptake)、コデポジット、トリチウムを含むダストの生成の三つのメカニズムに応じて計算モデル化した。三つのメカニズムはCauseyの評価モデルに基づく[2]。

1) 体積吸蔵

Causeyは、グラファイトの体積吸蔵を、時間、温度、トリチウムガス圧力、14MeV中性子照射による損傷率、グラファイト粒度(grains)の関数として評価している。ここでは、温度を1000℃、14MeV中性子照射による損傷率0.1%、グラファイト粒度1mmの結果を用いた。トリチウムガス圧力は、プラズマ対向機器、ダイバータとも10Paの結果を採用した。重水素、水素の各々の値については、エッジ部におけるこれらの組成比に比例するものとして、このトリチウムガス圧力の値から外挿した。エッジ部の組成比は、同部からの排気量の比で代表させた。シミュレーション時間1000秒の各計算ステップにおける値を、これらの数値からLagrangeの補間法により外挿した。ベリリウムの場合のトリチウムの体積吸蔵は、Smithの報告書に基づきグラファイトの場合の1/10とした[2]。Causeyの評価モデルから、トリチウムの吸蔵割合、$Bulk_{\_t}(t)$、は次式で定まる。

$$Bulk_{div\_t}(t) = \frac{\varepsilon_{Be} \cdot Bulk(T,t) \cdot V_{div}}{3.016} \cdot \chi_{edge\_t}(t - \Delta t) \tag{2.5}$$

$$Bulk_{fw\_t}(t) = \frac{\varepsilon_{Be} \cdot Bulk(T,t) \cdot V_{fw}}{3.016} \cdot \chi_{edge\_t}(t - \Delta t) \tag{2.6}$$

$$Bulk_{\_t}(t) = Bulk_{div\_t}(t) + Bulk_{fw\_t}(t) \tag{2.7}$$

$$\chi_{edge\_t}(t) = \frac{F_{ex\_t}(t)}{F_{ex\_t}(t) + F_{ex\_d}(t) + F_{ex\_p}(t)} \tag{2.8}$$

ここで、

$Bulk_{k\_t}(T,t)(k = div, fw)$: bulk intake of tritium (moles)

ε_{Be} : relative reduction of bulk inventory in Be compared to graphite

$Bulk(T,t)$: inventory concentration of tritium (g/m3)

$V_k(k = div, fw)$: total volume of divertor/first wall (m3)

である。具体的な数値は、付録1.に示す。燃焼中の吸蔵割合の時間変化を表わす状態方程式は、トリチウムの場合について次式で与えられる。

$$\frac{\Delta Bulk_{\_t}(t_i)}{\Delta t} = \frac{Bulk_{\_t}(t_i) - Bulk_{\_t}(t_{i\_1})}{\Delta t} - \lambda_t \cdot Bulk_{\_t}(t_i) \tag{2.9}$$

ここで、

λ_t : disintegration constant of tritium (1/h)

Δt : time step (h)

$t_k(k=i)$: time since beginning of simulation (h)

である。$Bulk_{div\_t}(t)$、$Bulk_{fw_t}(t)$のシミュレーション結果をFig.2.4、Fig.2.5に示す。

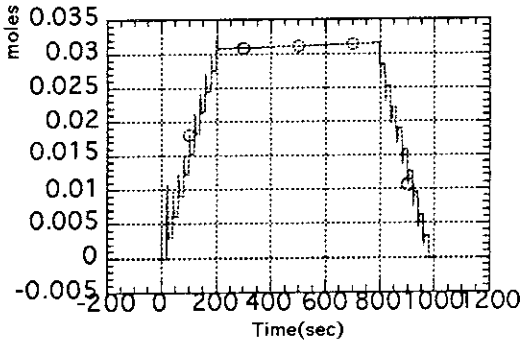


Fig. 2.4 Bulk intake of tritium from the edge into divertor (moles) : Bulkdiv_t(t)

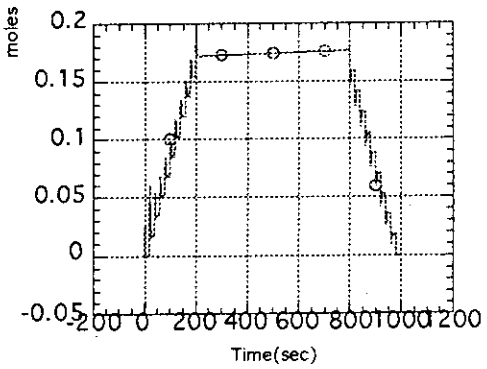


Fig. 2.5 Bulk intake of tritium from the edge into first wall (moles) : Bulkfw_t(t)

2) コデポジット

トリチウムのコデポジットする量の時間あたりの変化割合をＩＴＥＲ概念設計書から引用して次式で定めた。 重水素、水素の量は、Causey の評価モデルにしたがってエッジ部の組成比に比例するものとして定めた。

$$\frac{\Delta Co\_d_{\_t}(t)}{\Delta t} = \frac{1000}{8760} \cdot \frac{Co\_d(M,T)}{3.016} \cdot \chi_{edge\_t}(t-\Delta t) - \lambda_t \cdot Co\_d_{\_t}(t) \qquad (2.10)$$

ここで、

$Co\_d_{\_t}(t)$: tritium codeposited onto plasma facing components (moles)

$Co\_d(M,T)$: codeposition rate (kg/burn-year)

である。式(2.10)の解は、次式で与えられる。

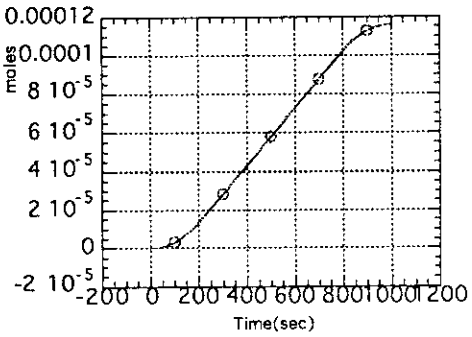


Fig. 2.6 Tritium codeposited onto plasma facing components (moles) : Co_d_t(t)

$$Co\_d_{\_t}(\Delta t) = \exp(-\lambda_t \Delta t) \cdot Co\_d(0) + \exp(-\lambda_t \Delta t) \cdot \int_0^{\Delta t} \exp(\lambda_t \eta) \frac{1000}{8760} \cdot \frac{Co\_d(M,T)}{3.016} \cdot \chi_{edge\_t}(0) d\eta \tag{2.11}$$

初期値から時間Δt毎に、計算ステップを繰り上げて各ステップ毎に上式の解を求めた[4]。数値解は、４次のルンゲ・クッタ法による。$Co_d(t)$のシミュレーション結果をFig. 2.6に示す。

3) トリチウムを含むダストの生成

ダストに含まれるトリチウムの量をITERの概念設計書の腐食割合から次のように定めた。重水素、水素の含まれる量は、エッジ部の組成比に比例するものとして定めた。

$$\frac{\Delta Dust_{\_t}(t)}{\Delta t} = \frac{1}{8760} \cdot \frac{Dust(M,T) \cdot A_d \cdot dc(M) \cdot \rho(M)}{3.016} \cdot \chi_{edge\_t}(t - \Delta t) - \lambda_t \cdot Dust_{\_t}(t) \tag{2.12}$$

ここで、

$Dust_{\_t}(t)$: tritium in dust (moles)

$Dust(M,T)$: erosion rate (m/burn-year)

A_d : divertor surface area (m2)

$dc(M)$: dust concentration ratio (kgtritium/kgdust)

である。解は次式で与えられる。

$$Dust_{\_t}(\Delta t) = \exp(-\lambda_t \Delta t) \cdot Dust_{\_t}(0) + \exp(-\lambda_t \Delta t) \cdot \int_0^{\Delta t} \exp(\lambda_t \eta) \cdot \frac{1}{8760} \cdot \frac{Dust(M,T) \cdot A_d \cdot dc(M) \cdot \rho(M)}{3.016} \cdot \chi_{edge\_t}(0) d\eta \tag{2.13}$$

$Dust_{_t}(t)$のシミュレーション結果をFig. 2.7に示す。

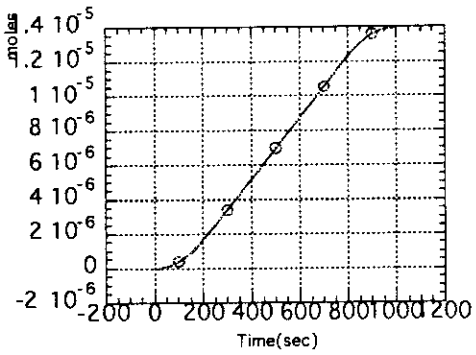


Fig .2.7 Tritium in dust (moles)

: Dust_t(t)

(3) 燃焼時のプラズマチェンバへの供給と排気との収支

燃焼時のプラズマチェンバへの供給と排気との収支を、燃焼時の消費量とプラズマ対

向機器との相互作用の大きさを表わす項とを取り入れて次のバランス式により定めた。

$$Fuel_{\_t}(t)(1-\varepsilon_{bu}) = F_{ex\_t}(t) + \frac{\Delta PFC_{\_t}(t)}{\Delta t} \tag{2.14}$$

$$Fuel_{\_d}(t) = F_{ex\_d}(t) + \frac{\Delta PFC_{\_d}(t)}{\Delta t} \tag{2.15}$$

$$Fuel_{\_p}(t) = F_{ex\_p}(t) - Gen_{\_p}(t) + \frac{\Delta PFC_{\_p}(t)}{\Delta t} \tag{2.16}$$

$$PFC_{\_k}(t) = Bulk_{\_k}(t) + Co\_d_{\_k}(t) + Dust_{\_k}(t) \quad (k = t, d, p) \tag{2.17}$$

ここで、$Fuel_{\_t}(t)$、$Fuel_{\_d}(t)$、$Fuel_{_p}(t)$は、それぞれトリチウム、重水素、水素の燃焼時に供給されている量である。$F_{ex\_t}(t)$、$F_{ex\_d}(t)$、$F_{ex_p}(t)$は、プラズマチェンバからの排気量である。$\varepsilon_{bu} \cdot Fuel_{_t}(t)$、$Burn_{\_d}(t)$、$Gen_{_p}(t)$は、燃焼時の消費または反応量であって次式で与えられる。

$$\begin{aligned} &\varepsilon_{bu} \cdot Fuel_{\_t}(t) = Burn_{\_t}(t) \\ &Burn_{\_d}(t) = Burn_{\_t}(t) \\ &Gen_{\_p}(t) = \varepsilon_{prot} \cdot \varepsilon_{bu} \cdot Burn_{\_t}(t) \end{aligned} \tag{2.18}$$

システム変数の位置関係をFig. 2.8に示す。そのうちの$F_{ex_t}(t)$のシミュレーション結果をFig. 2.9に示す。

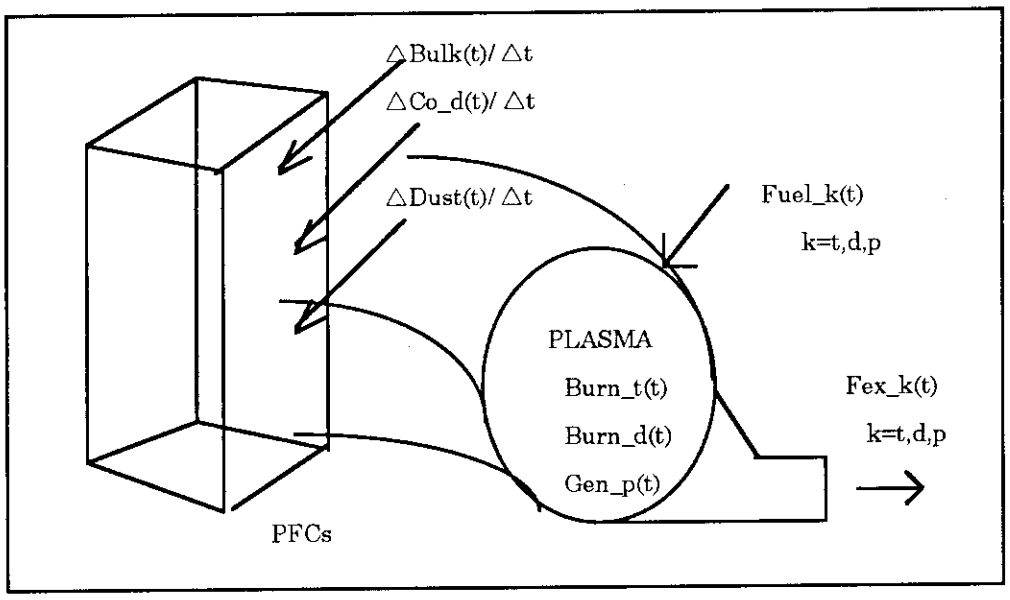


Fig. 2.8 A schematic of an idea of the mass balance in plasma interactions

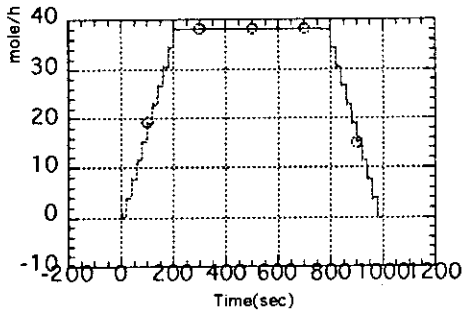


Fig. 2.9 Exhaust rate of tritium from plasma (mole/h) : Fex_t(t)

(4) 真空排気サブシステム

真空排気サブシステムは、燃焼灰を排気するに必要なコンダクタンスと排気能力を有するものとして、その排気能力をクライオポンプで受け持つものとする。クライオポンプの運転形態は、排気に要する台数の並列運転となる。本サブシステムを代表する処理機能は再生である。この再生処理機能により系のインベントリの大きさが定まる。このモデルでは、連続再生方式をモデル化した。

a. 状態方程式

連続再生方式において再生処理される、ポンプ１台あたりのトリチウム、重水素、水素の時間変化を表わすシステム変数、$R_{vacuum\_t}(t)$、$R_{vacuum\_d}(t)$、$R_{vacuum\_p}(t)$、$R_{vacuum\_he}(t)$、を導入した。サブシステム内に流入する量と、再生処理によりサブシステム外に流出する量とのバランスから、状態方程式を記述する。式(2.1)のベクトル変数$\mathbf{x}(t)=(x1,x2,x3)$の各要素に、サブシステム内のインベントリの時間変化を表わすシステム変数、$T_{vacuum}(t)$、$D_{vacuum}(t)$、$P_{vacuum}(t)$、$He_{vacuum(t)}$、を代入すると、状態方程式は次式で与えられる。

$$\frac{d}{dt}\begin{bmatrix} T_{vacuum}(t) \\ D_{vacuum}(t) \\ P_{vacuum}(t) \\ He_{vacuum(t)} \end{bmatrix} = \frac{1}{n_{pump}}\begin{bmatrix} F_{ex\_t}(t) \\ F_{ex\_d}(t) \\ F_{ex\_p}(t) \\ F_{ex\_he}(t) \end{bmatrix} - \begin{bmatrix} R_{vacuum\_t}(t) \\ R_{vacuum\_d}(t) \\ R_{vacuum\_p}(t) \\ R_{vacuum\_he}(t) \end{bmatrix} \tag{2.19}$$

$$R_{vacuum\_k}(t) = \frac{(M_{vacuum}(t)\cdot \varepsilon_{scrap}\cdot r_{scrap})}{a_{cryo}} \tag{2.20}$$

$$M = T,D,P,He \quad k = t,d,p,he$$

ここで、

ε_{scrap} : removal efficiency of cryodeposit

r_{scrap} : scraping rate on cryosurface (m2/h)

a_{cryo} : cryosurface per pump (m2)

n_{pump} : number of operating pumps

である。

b. 出力方程式

連続再生処理により再生されたトリチウム、重水素、水素は、燃料精製サブシステムに送られる。このときの出力方程式は、次式で与えられる。

$$F_{v\_out}(t) = (R_{vacuum\_t}(t) + R_{vacuum\_d}(t) + R_{vacuum\_p}(t) + R_{vacuum\_he}(t)) \cdot n_{pump} \quad (2.21)$$

システム変数の位置関係をFig. 2.10に示す。$R_{vacuum_t}(t)$と$F_{v_out}(t)$のシミュレーション結果を、それぞれFig. 2.11とFig. 2.12に示す。

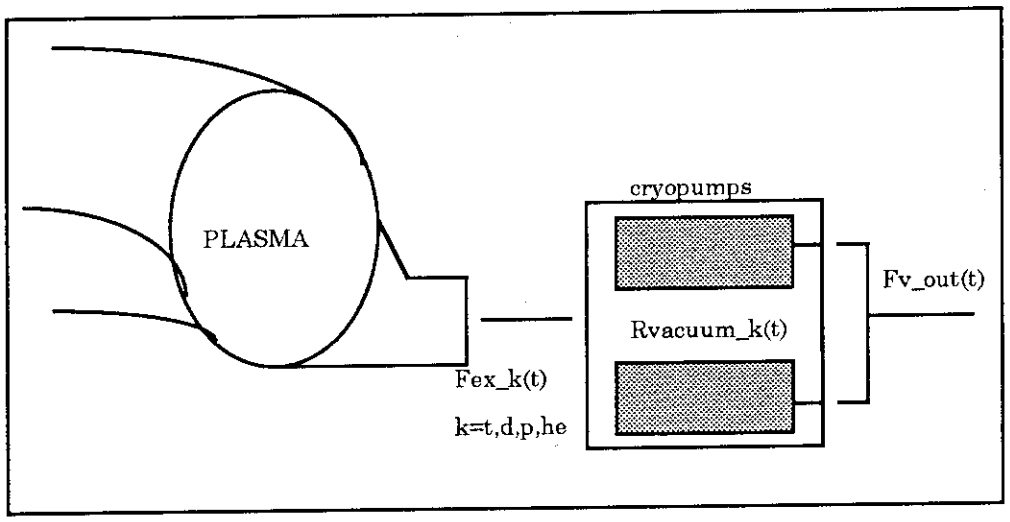


Fig. 2.10 Flow diagram of exhaust processing sub system

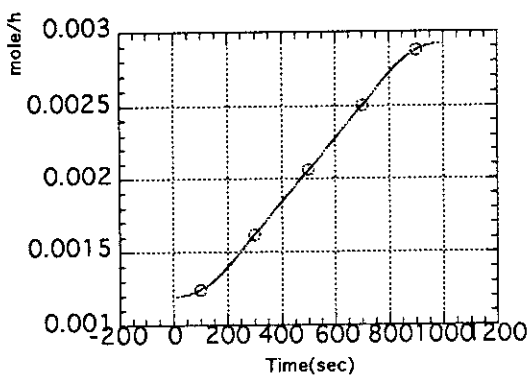


Fig. 2.11 Removal rate of tritium from continuous regeneration cryopump (mole/h)
: Rvacuum_t(t)

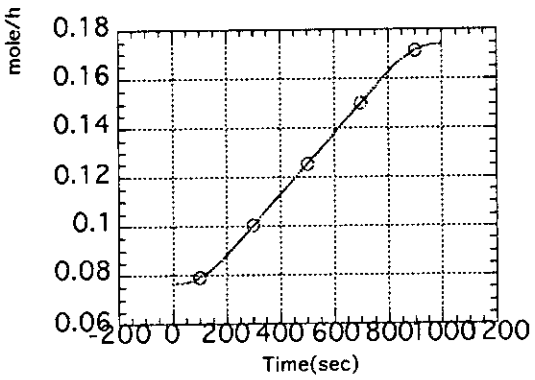


Fig. 2.12 Total flow from vacuum pumps to fuel clean-up sub system
: Fv_out(t)

(5) 燃料精製サブシステム

燃料精製サブシステムの処理機能は、プラズマチェンバ内のトリチウムを含む排気から不純物を除去し、回収することである。トリチウム、重水素、水素の各同位体は、モレキュラーシーブベッドにより Q2 (Q:generic Hydrogen)の水素化物として吸着される。このときの主要な不純物は、Q2O、NQ3、CQ4である。除去された不純物は、再生行程を経てパラジュウム透過膜装置へ送られる。ここで、Q2 が透過し、不純物が除かれる。機器構成を、精製、待機、再生行程からなるものとしてモデル化した。

a. 状態方程式

燃料精製サブシステムへの入口流量中の不純物の組成割合を次式で定義した。

$$F_{v\_out}(t) = F_{pure\_}(t) + F_{imp\_}(t) \tag{2.22}$$

$$F_{pure\_}(t) = F_{v\_out}(t) \cdot \chi_{exhau\_pure} \tag{2.23}$$

$$F_{imp\_}(t) = F_{v\_out}(t) \cdot \chi_{exhau\_imp} \tag{2.24}$$

$$\chi_{exhau\_pure} + \chi_{exhau\_imp} = 1 \tag{2.25}$$

$$F_{pure\_}(t) = F_{pure\_t}(t) + F_{pure\_d}(t) + F_{pure\_p}(t) + F_{\_he}(t) \tag{2.26}$$

$$F_{imp\_}(t) = F_{imp\_cq4}(t) + F_{imp\_nq3}(t) + F_{imp\_q2o}(t) + F_{imp\_nhydro}(t) \tag{2.27}$$

ここで、

$F_{pure\_}(t)$: pure hydrogen flow into fuel clean-up sub system (mole/h)

$F_{imp\_}(t)$: hydrogen flow containing impurities into fuel clean-up sub system (mole/h)

χ_{exhau_pure} : fraction of pure hydrogen in exhaust processing cryopumps

χ_{exhau_imp} : fraction of hydrogen containing impurities in exhaust processing cryopumps

である。再生される不純物流量のシミュレーション結果、$F_{imp\_cq4}(t)$、$F_{imp\_nq3}(t)$、$F_{imp\_q2o}(t)$、を Fig. 2.13、Fig. 2.14、Fig. 2.15 に示す。式(2.27)右辺の流量分配を定める排気中のモル組成比は文献[1]に基づく。Q2 のシミュレーション結果、$F_{pure\_t}(t)$、$F_{pure\_d}(t)$、$F_{pure\_p}(t)$、を Fig. 2.16、Fig. 2.17、Fig. 2.18 に示す。

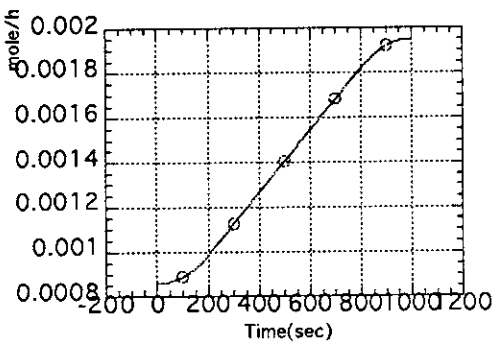


Fig. 2.13 Flow of methane to purification (mole/h) : Fimp_cq4(t)

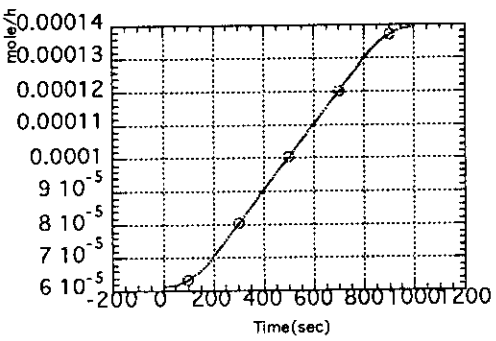


Fig. 2.14 Flow of ammonia to purification (mole/h) : Fimp_nq3(t)

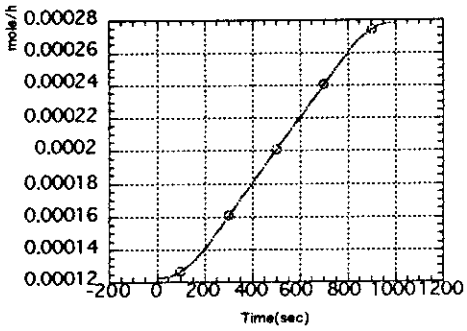


Fig. 2.15 Flow of water to purification (mole/h) : Fimp_q2o(t)

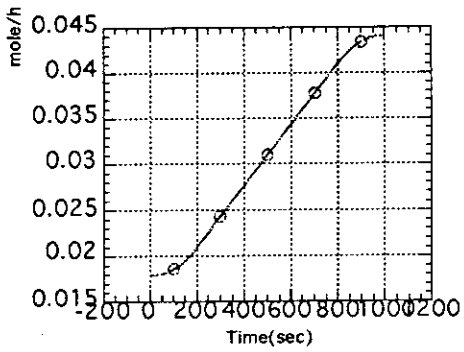


Fig. 2.16 Flow of pure tritium atoms to purification (mole/h) : Fpure_t(t)

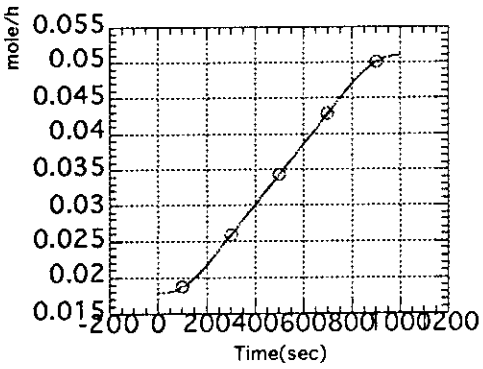


Fig. 2.17 Flow of pure deuterium atoms to purification (mole/h) : Fpure_d(t)

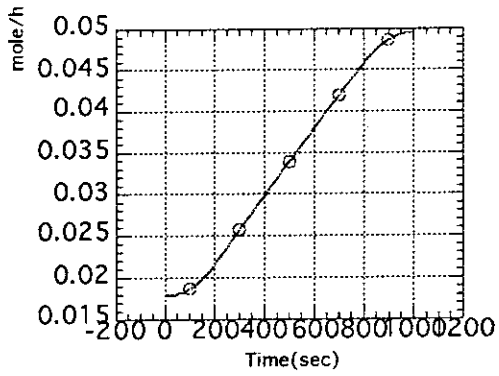


Fig. 2.18 Flow of protium atoms to purification (mole/h) : Fpure_p(t)

除去される不純物の量、$E_{cq4\_reg}(t)$、$E_{nq3\_reg}(t)$、$E_{q2o_reg}(t)$は、再生行程中再生割合が一定として次式で与えられる。

$$E_{cq4\_reg}(t) = \frac{N_{cq4\_reg}(t)}{t_{msieve\_reg}} \tag{2.28}$$

$$E_{nq3\_reg}(t) = \frac{N_{nq3\_reg}(t)}{t_{msieve\_reg}} \tag{2.29}$$

$$E_{q2o\_reg}(t) = \frac{N_{q2o\_reg}(t)}{t_{msieve\_reg}} \tag{2.30}$$

上式右辺の$N_{cq4_reg}(t)$、$N_{nq3\_reg}(t)$、$N_{q2o_reg}(t)$は、入口流量より次式で定められる。

$$\frac{dN_{cq4\_reg}(t)}{dt} = F_{imp\_cq4}(t) \tag{2.31}$$

$$\frac{dN_{nq3\_reg}(t)}{dt} = F_{nq3\_reg}(t) \tag{2.32}$$

$$\frac{dN_{q2o\_reg}(t)}{dt} = F_{q2o\_reg}(t) \tag{2.33}$$

ここで、

t_{msieve_reg} : regeneration time of molecular sieve

である。再生行程を経て、パラジュウム透過膜装置へ送られ、不純物からQ2が透過される割合をシステム変数、$T_{pmem}(t)$、$D_{pmem}(t)$、$P_{pmem}(t)$、を導入して、次式で定義した。

$$\frac{d}{dt}\begin{bmatrix} T_{pmem}(t) \\ D_{pmem}(t) \\ P_{pmem}(t) \end{bmatrix} = F_{pmem\_hydro}(t)\cdot\begin{bmatrix} \chi_{pmem\_t} \\ \chi_{pmem\_d} \\ \chi_{pmem\_p} \end{bmatrix} - \frac{1}{t_{pmem}}\begin{bmatrix} T_{pmem}(t) \\ D_{pmem}(t) \\ P_{pmem}(t) \end{bmatrix} \tag{2.34}$$

$$F_{pmem\_hydro}(t) = 4E_{cq4\_reg}(t) + 3E_{nq3\_reg}(t) + 2E_{q2o\_reg}(t) \tag{2.35}$$

ここで、

$K_{pmem}(t)(K = T, D, P)$: tritium, deuterium, protium in palladium membrane reactor (moles)

$F_{pmem\_hydro}(t)$: flow of hydrogen from regeneration unit to palladium membrane reactor (mole/h)

$\chi_{pmem\_k}(k = t, d, p)$: isotopic fraction on regeneration unit

t_{pmem} : time constant of palladium membrane reactor (h)

である。

b. 出力方程式

燃料精製サブシステムからの出口流量は、次式で与えられる。

$$\begin{bmatrix} F_{puri\_out\_t}(t) \\ F_{puri\_out\_d}(t) \\ F_{puri\_out\_p}(t) \end{bmatrix} = \begin{bmatrix} F_{sub\_out\_t}(t) \\ F_{sub\_out\_d}(t) \\ F_{sub\_out\_p}(t) \end{bmatrix} + \frac{1}{t_{pmem}}\begin{bmatrix} T_{pmem}(t) \\ D_{pmem}(t) \\ P_{pmem}(t) \end{bmatrix} \tag{2.36}$$

右辺第1項、待機行程の出口流量、を精製行程からの流入量と、待機行程のプリローデングの和として次式で与えた。

$$\begin{bmatrix} F_{sub\_out\_t}(t) \\ F_{sub\_out\_d}(t) \\ F_{sub\_out\_p}(t) \end{bmatrix} = \begin{bmatrix} F_{pure\_t}(t) \\ F_{pure\_d}(t) \\ F_{pure\_p}(t) \end{bmatrix} + F_{v\_out}(t)\cdot\begin{bmatrix} \chi_{sub\_t}(t) \\ \chi_{sub\_d}(t) \\ \chi_{sub\_p}(t) \end{bmatrix} \tag{2.37}$$

ここで、

$\chi_{sub\_k}(t)(k = t, d, p)$: mole fraction on standby bed after preloading

である。

出口流量は、式(2.36)左辺の和として、次式で与えられる。

$$F_{puri\_out}(t) = F_{puri\_out\_t}(t) + F_{puri\_out\_d}(t) + F_{puri\_out\_p}(t) \tag{2.38}$$

シミュレーション結果をFig.2.19に示す。再生行程における時定数、t_{msieve_reg}、は、

4時間である（式2.28から式2.30)。このため、1パルス運転のシミュレーション時間(1000秒)におけるパラジュウム透過膜装置への戻り量は零であり、出口流量は、式(2.37)の与えるものとなる。導入されたシステム変数の位置関係をFig.2.20に示す。

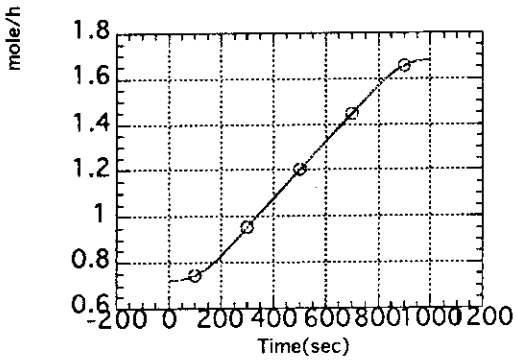


Fig. 2.19 Flow from fuel clean-up sub system to isotope separation sub system (mole/h) : Fpuri_out(t)

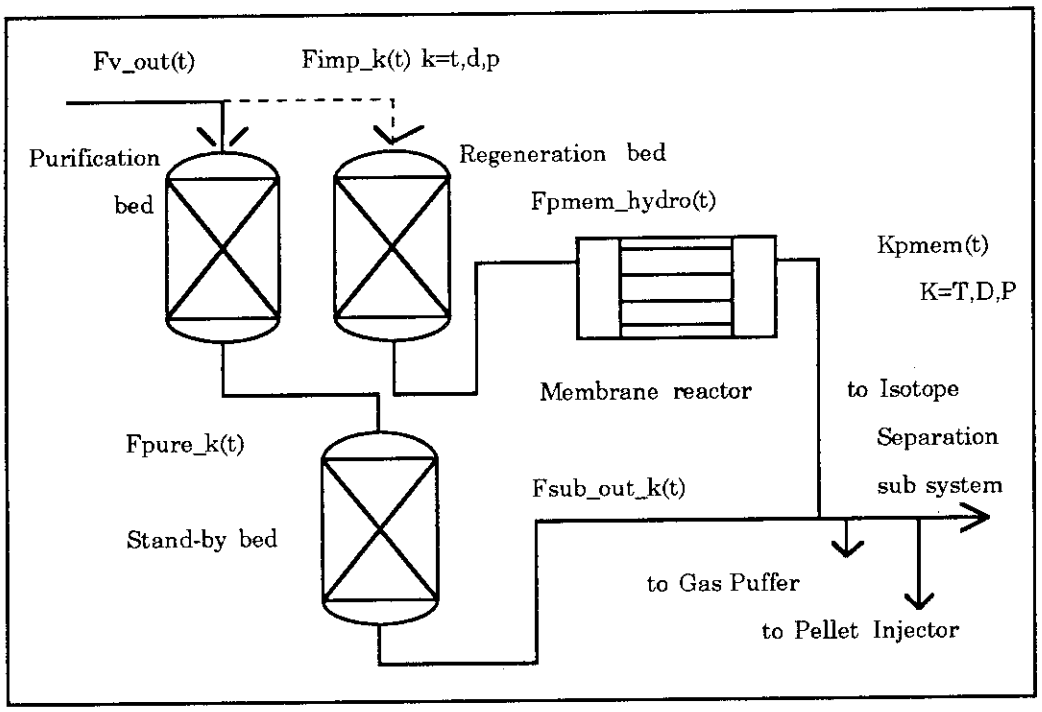


Fig. 2.20 Flow diagram of Fuel Clean-up sub system

(6) 同位体分離サブシステム

同位体分離サブシステムは、燃料精製サブシステムの精製処理に引き続いて水素同位体を分離する。分離されたもののうちトリチウムはペレット入射サブシステムとガスパ

フサブシステムに、重水素は中性粒子入射サブシステムとペレット入射サブシステム及びガスパフサブシステムに、水素はペレット入射サブシステムに供給される。深冷蒸留法による同位体分離サブシステムは多段構成であるが、計算モデルにおいては、各段毎の状態方程式は定義せず、サブシステムへの流入量と出口流量とのマスバランスから状態方程式を記述した。

a. 状態方程式

同位体分離サブシステムへの流入量、$F_{iss\_in}(t)$、は次式で定義される。

$$F_{iss\_in}(t) = (1 - h_1 - h_3) \cdot F_{puri\_out}(t) \tag{2.39}$$

$$\begin{bmatrix} F_{iss\_in\_t}(t) \\ F_{iss\_in\_d}(t) \end{bmatrix} = (1 - h_1 - h_3) \cdot F_{puri\_out}(t) \cdot \begin{bmatrix} \chi_{iss\_inj\_t}(t) \\ \chi_{iss\_inj\_d}(t) \end{bmatrix} \tag{2.40}$$

$$F_{iss\_in\_p}(t) = F_{iss\_in}(t) - F_{iss\_in\_t}(t) - F_{iss\_in\_d}(t) \tag{2.41}$$

ここで、

$F_{iss\_in\_t}(t)$: tritium flow in isotope separation sub system (mole/h)

$F_{iss\_in\_d}(t)$: deuterium flow in isotope separation sub system (mole/h)

$F_{iss\_in\_p}(t)$: protium flow in isotope separation sub system (mole/h)

$\chi_{iss\_in\_k}(t)(k = t, d)$

: mole fraction of isotope in stream from fuel clean-up sub system

$\chi_{nb\_k}(t)(k = t, d)$

: mole fraction of isotope in stream from neutral beam injection sub system

$\chi_{inj\_k}(t)(k = t, d)$

: mole fraction of isotope in stream from pellet injector sub system

h_1 : fraction of stream not processed and directed to gas puffer sub system

h_3 : fraction of stream not processed and directed to pellet injector sub system

シミュレーション結果を Fig. 2.21、Fig. 2.22、Fig. 2.23 に示す。

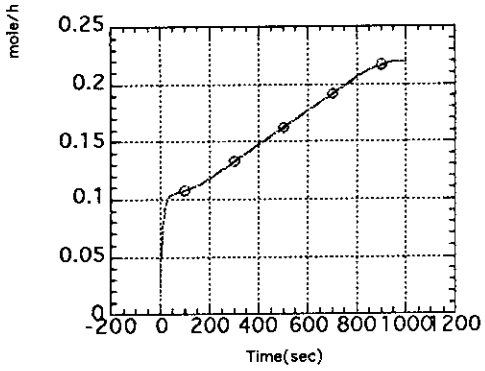


Fig. 2.21 Flow of tritium to isotope separation sub system (mole/h): Fiss_in_t(t)

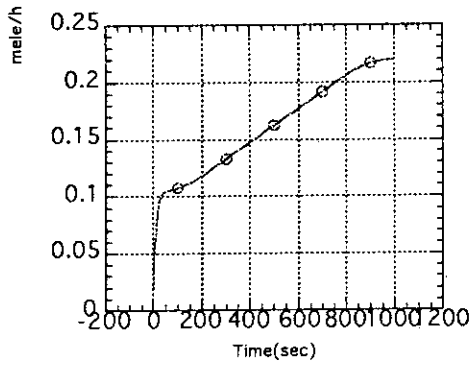


Fig. 2.22 Flow of deuterium to isotope separation sub system (mole/h) : Fiss_in_d(t)

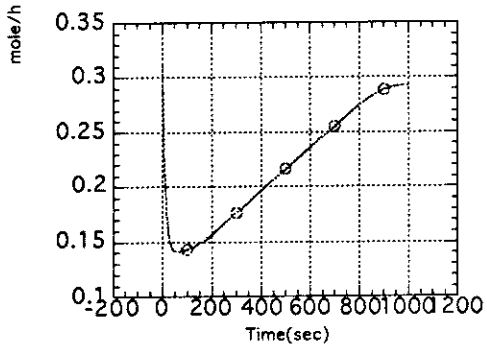


Fig. 2.23 Flow of protium to isotope separation sub system (mole/h): Fiss_in_p (t)

b. 出力方程式

同位体分離により濃度の増したトリチウム、重水素、水素各々の分流を表わす状態変数、$F_{iss\_top}(t)$、$F_{iss\_mid}(t)$、$F_{iss\_bot}(t)$、を導入して、これらにより、a. 状態方程式で定められた処理高との関連を以下のように記述した。

$$F_{iss\_top}(t)+F_{iss\_mid}(t)+F_{iss\_bot}(t)=\frac{N_{iss\_}(t)}{t_{iss}} \tag{2.42}$$

$$F_{iss\_top}(t)\cdot\chi_{iss\_top\_t}+F_{iss\_mid}(t)\cdot\chi_{iss\_mid\_t}+F_{iss\_bot}(t)\cdot\chi_{iss\_bot\_t}=\frac{T_{iss\_}(t)}{t_{iss}} \tag{2.43}$$

$$F_{iss\_top}(t)\cdot\chi_{iss\_top\_d}+F_{iss\_mid}(t)\cdot\chi_{iss\_mid\_d}+F_{iss\_bot}(t)\cdot\chi_{iss\_bot\_d}=\frac{D_{iss\_}(t)}{t_{iss}} \tag{2.44}$$

右辺に現われる変数、$N_{iss\_}(t)$、$T_{iss\_}(t)$、$D_{iss\_}(t)$、は、サブシステム内の処理量を表わし、a. 状態方程式で定められた流入量から次式で定義される。

$$\frac{dN_{iss\_}(t)}{dt} = F_{iss\_in}(t) - \frac{1}{t_{iss}} \cdot N_{iss\_}(t) \tag{2.45}$$

$$\frac{d}{dt}\begin{bmatrix} T_{iss\_}(t) \\ D_{iss\_}(t) \end{bmatrix} = \begin{bmatrix} F_{iss\_in\_t}(t) \\ F_{iss\_in\_d}(t) \end{bmatrix} - \frac{1}{t_{iss}} \cdot \begin{bmatrix} T_{iss\_}(t) \\ D_{iss\_}(t) \end{bmatrix} \tag{2.46}$$

$$P_{oiss\_}(t) = \chi_{iss\_p}(t) \cdot N_{iss\_}(t) \tag{2.47}$$

$$\chi_{iss\_t}(t) = \frac{T_{iss\_}(t)}{N_{iss\_}(t)} \tag{2.48}$$

$$\chi_{iss\_d}(t) = \frac{D_{iss\_}(t)}{N_{iss\_}(t)} \tag{2.49}$$

$$\chi_{iss\_p}(t) = 1 - \chi_{iss\_t}(t) - \chi_{iss\_d}(t) \tag{2.50}$$

同位体分離により濃度の増したトリチウム、重水素、水素各々の分流を表わす状態変数、$F_{iss\_top}(t)$、$F_{iss\_mid}(t)$、$F_{iss\_bot}(t)$、のシミュレーション結果を、Fig. 2.24、Fig. 2.25、Fig. 2.26に示す。導入されたシステム変数の位置関係をFig. 2.27に示す。

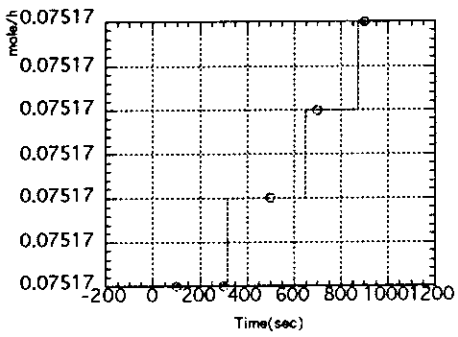


Fig. 2.24 Flow rate from the bottom of isotope separation sub system (mole/h)
: Fiss_bot (t)

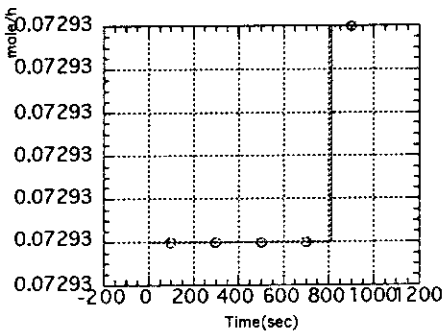


Fig. 2.25 Flow rate from the middle of isotope separation sub system (mole/h)
: Fiss_mid(t)

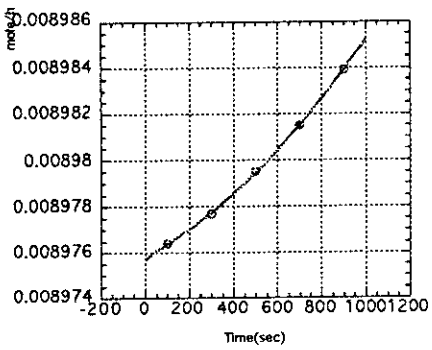


Fig. 2.26 Flow rate from the top of isotope separation sub system (mole/h) : Fiss_top (t)

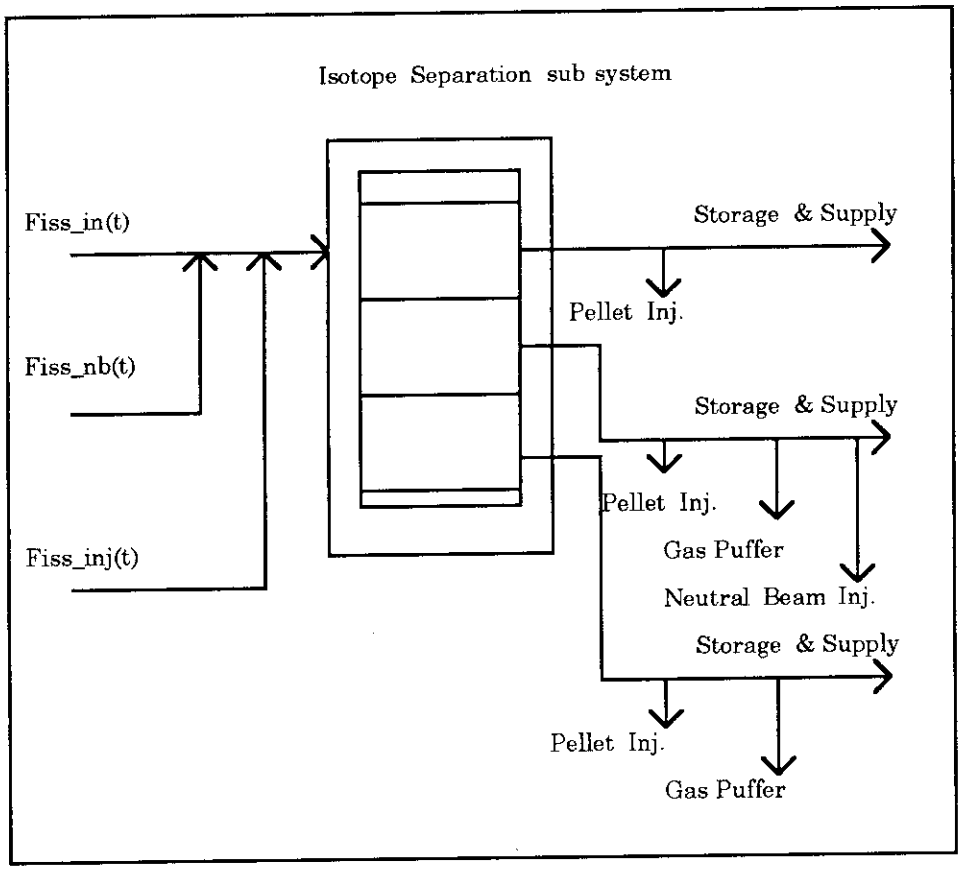


Fig. 2.27 Flow diagram for isotope separation sub system

(7) 中性粒子入射サブシステム

中性粒子入射サブシステムの受け持つ機能は、電流駆動、電流分布コントロールおよびプラズマ加熱である。入射量は、プラズマ燃焼の大きさにより定まる。その運転パタ

ーンは、Fig.2.3に示してある。運転パターンに従う入射流量、$F_{plas\_nb\_d}(t)$、を次式で定義した。

$$F_{plas\_nb\_d}(t)=\frac{3.6\cdot 10^3}{1.6}\cdot 10^9\cdot\frac{1}{N_0}\cdot\frac{f_{power}\cdot W_{\_nb}}{E_{\_nb}}(mole/h) \tag{2.51}$$

ここで、

$W_{\_nb}$: neutral beam power (75MW)

$E_{\_nb}$: neutral beam particle energy (1.3MeV)

N_0 : Avogadro's Number

である。

a. 状態方程式

運転パターンで定まる中性粒子入射流量に必要な量の重水素、$F_{plas\_nb\_d}(t)$、は、同位体分離サブシステムからの戻り量、$F_{iss\_mid}(t)$ 、と貯蔵・供給サブシステムからの供給量、$F_{in\_st\_d}(t)$ 、によりまかなわれる。入出力のバランス式は、以下となる。

$$\begin{aligned}\frac{F_{plas\_nb\_d}(t)}{\eta_{nb\_tot}} &= F_{in\_st\_d}(t)\cdot\chi_{sto\_deu\_d}(t)\\ &\quad +k_{iss\_mid}\cdot F_{iss\_mid}(t)\cdot\chi_{iss\_mid\_d}(t)\end{aligned} \tag{2.52}$$

ここで、

$\chi_{sto\_deu\_d}(t)$: mole fraction of deuterium in deuterium rich storage

$\chi_{iss\_mid\_d}(t)$: mole fraction of deuterium in the stream from the middle of isotope separation sub system

k_{iss_mid} : fraction of flow from middle of isotope separation sub system directed to the neutral beam injection

η_{nb_tot} : efficiency of neutral beam injection sub system

である。

中性粒子入射には、その入射ポートを高真空に維持するためクライオポンプが使用されている。クライオポンプにより排出される水素化物は再生行程を経て同位体分離サブシステムに回される。このときの再生方式は、真空排気サブシステムと同じく連続方式とした。再生量、$R_{nb\_reg\_t}(t)$、$R_{nb\_reg\_d}(t)$、$R_{nb\_reg\_p}(t)$、の緒元は、本章(4)節真空排気サブシステムの$R_{vacuum_k}(t)$に同じである。

$$\begin{aligned}\frac{d}{dt}\begin{bmatrix}N_{cryo\_}(t)\\ T_{nb\_cryo}(t)\\ D_{nb\_cryo}(t)\end{bmatrix} &= f_{cryo\_}(t)\cdot k_{iss\_mid}\cdot F_{iss\_mid}(t)\cdot\begin{bmatrix}1\\ \chi_{iss\_mid\_t}\\ \chi_{iss\_mid\_d}\end{bmatrix}\\ &\quad +f_{cryo\_}(t)\cdot F_{in\_sto\_d}(t)\cdot\begin{bmatrix}1\\ \chi_{sto\_deu\_t}\\ \chi_{sto\_deu\_d}\end{bmatrix}-\begin{bmatrix}R_{nb\_reg}(t)\\ R_{nb\_reg\_t}(t)\\ R_{nb\_reg\_d}(t)\end{bmatrix}\end{aligned} \tag{2.53}$$

$$P_{nb\_cryo}(t) = \chi_{nb\_cryo\_p}(t) \cdot N_{cryo\_}(t) \tag{2.54}$$

$$\chi_{nb\_cryo\_t}(t) = \frac{T_{nb\_cryo}(t)}{N_{cryo\_}(t)} \tag{2.55}$$

$$\chi_{nb\_cryo\_d}(t) = \frac{D_{nb\_cryo}(t)}{N_{cryo\_}(t)} \tag{2.56}$$

$$\chi_{nb\_cryo\_p}(t) = 1 - \chi_{nb\_cryo\_t}(t) - \chi_{nb\_cryo\_d}(t) \tag{2.57}$$

ここで、

$N_{cryo\_}(t)$: total amount on neutral beam cryopumps (moles)

$T_{nb\_cryo}(t)$: tritium on neutral beam cryopumps (moles)

$D_{nb\_cryo}(t)$: deuterium on neutral beam cryopumps (moles)

$\chi_{iss\_mid\_k}(k = t, d)$

: mole fraction on the stream from middle of isotope separation sub system

$\chi_{sto\_deu\_k}(k = t, d)$

: mole fraction in deuterium rich storage in storage and supply sub system

である。

b. 出力方程式

中性粒子入射サブシステムからプラズマチェンバへの入射流量、$F_{plas\_nb\_t}(t)$、$F_{plas\_nb\_d}(t)$、$F_{plas\_nb\_p}(t)$、を、入射効率、η_{nb_tot}、を導入して、次式で定めた。$F_{plas\_nb\_t}(t)$、$F_{plas_nb_p}(t)$は、中性粒子入射時に同時に入射される重水素以外の量を与える。

$$F_{plas\_nb\_}(t) = (k_{iss\_mid} \cdot F_{iss\_mid}(t) + F_{in\_st\_d}(t)) \cdot \eta_{nb\_tot} \tag{2.58}$$

$$\begin{aligned} F_{plas\_nb\_t}(t) = (&k_{iss\_mid} \cdot F_{iss\_mid}(t) \cdot \chi_{iss\_mid\_t} \\ &+ F_{in\_st\_d}(t) \cdot \chi_{sto\_deu\_t}) \cdot \eta_{nb\_tot} \end{aligned} \tag{2.59}$$

$$\begin{aligned} F_{plas\_nb\_d}(t) = (&k_{iss\_mid} \cdot F_{iss\_mid}(t) \cdot \chi_{iss\_mid\_d} \\ &+ F_{in\_st\_d}(t) \cdot \chi_{sto\_deu\_d}) \cdot \eta_{nb\_tot} \end{aligned} \tag{2.60}$$

$$F_{plas\_nb\_p}(t) = F_{plas\_nb\_}(t) - F_{plas\_nb\_t}(t) - F_{plas\_nb\_d}(t) \tag{2.61}$$

再生行程を経て、同位体分離サブシステムに回される流量は、次式で与えられる。

$$F_{nb\_cryo}(t) = R_{nb\_reg\_t}(t) + R_{nb\_reg\_d}(t) + R_{nb\_reg\_p}(t) \tag{2.62}$$

中性粒子入射流量、$F_{plas\_nb\_d}(t)$、と、同時に入射されることになるトリチウムと水素の流入量、$F_{plas\_nb\_t}(t)$、$F_{plas\_nb\_p}(t)$、のシミュレーション結果を、それぞれ Fig. 2.28、Fig. 2.29、Fig. 2.30 に示す。同位体分離サブシステムに回される流量、$F_{nb\_cryo}(t)$、のシミュレーション結果を Fig. 2.31 に示す。導入されたシステム変数の位置関係を Fig. 2.32 に示す。

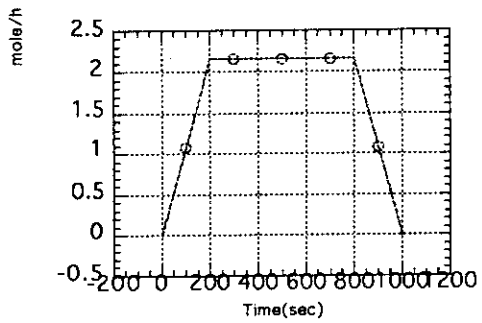


Fig. 2.28 Flow of deuterium to plasma passing neutral beam (mole/h) : Fplas_nb_d(t)

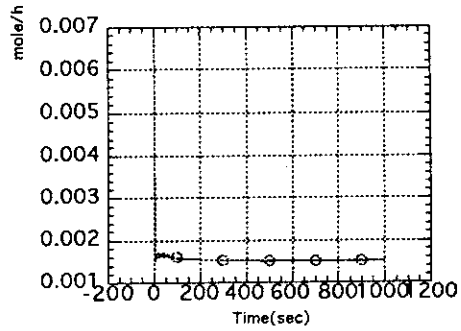


Fig. 2.29 Flow of tritium to plasma through neutral beam (mole/h) : Fplas_nb_t(t)

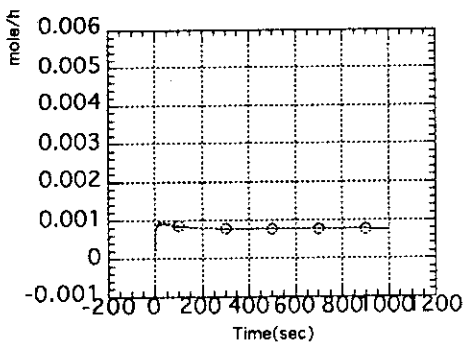


Fig. 2.30 Flow of protium to plasma through neutral beam (mole/h) : Fplas_nb_p(t)

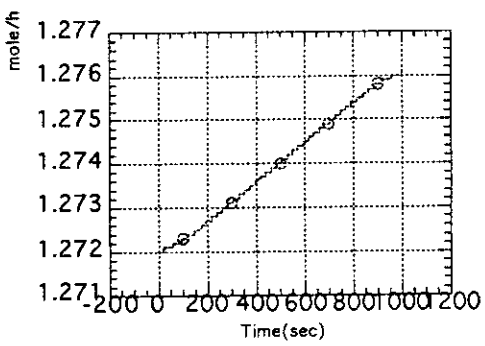


Fig. 2.31 Regeneration rate from neutral beam cryopumps (mole/h) : Fnb_cryo(t)

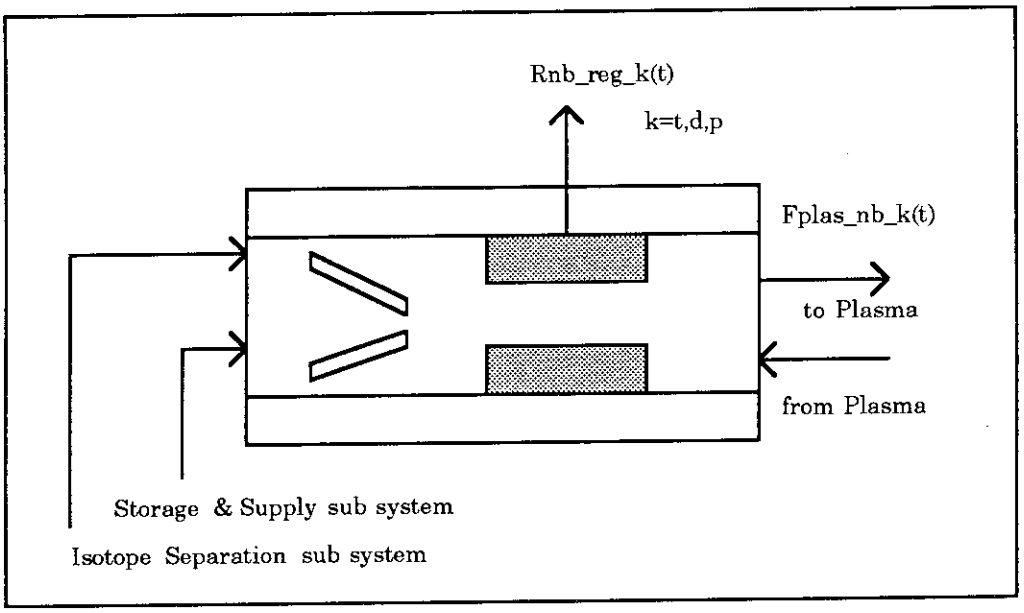


Fig. 2.32 Flow diagram for neutral beam injection sub system

(8) ペレット入射サブシステム

ペレット入射サブシステムの機器構成を、燃料ペレット精製機器と燃料ペレット押し出し機器より成るとした。ペレット入射サブシステムの行程は、交互に燃料ペレット精製と燃料ペレット押し出しが繰り返されるものとした。入射のためのプロペラントガスは、同位体分離サブシステムからの戻り量と貯蔵・供給サブシステムからの供給により用意される。

a. 状態方程式

ペレット入射量は、その時刻におけるプラズマ燃焼量に比例する大きさから定めた。注入されるペレットの大きさを定めることにより、1ペレット当りのトリチウム、重水素、水素の含有量を求め、次に、プラズマ燃焼から定められる量に見合う単位時間当りの注入回数を求めた。入射のためのプロペラントガスは水素とし、これらの量は、同位体分離サブシステムからの戻り量と貯蔵・供給サブシステムからの供給によりまかなわられるとした。プラズマチェンバへの入射量と供給量とのバランスを式(2.63)で定めた。ペレット入射サブシステムからの排気量を、排気効率、η_{cap_prop}、を導入して式(2.67)で定義した。ペレット入射流量のシミュレーション結果を、Fig. 2.33、Fig. 2.34、Fig. 2.35、に、排気流量のシミュレーション結果を Fig. 2.36 に示す。導入されたシステム変数の位置関係を Fig. 2.37 に示す。

$$\frac{1}{(1-\eta_{inj\_err})}\cdot\begin{bmatrix}F_{plas\_in\_t}(t)\\F_{plas\_in\_d}(t)\end{bmatrix}=F_{plas\_out\_t}(t)\cdot\begin{bmatrix}\chi_{sto\_tri\_t}(t)\\\chi_{sto\_tri\_d}(t)\end{bmatrix}$$
$$+F_{plas-out\_d}(t)\cdot\begin{bmatrix}\chi_{sto\_deu\_t}(t)\\\chi_{sto\_deu\_d}(t)\end{bmatrix}$$
$$+h_3\cdot F_{puri\_out}(t)\cdot\begin{bmatrix}\chi_{iss\_in\_t}(t)\\\chi_{iss\_in-d}(t)\end{bmatrix} \tag{2.63}$$
$$+l_{inj\_mid\_iss}\cdot F_{iss\_mid}(t)\cdot\begin{bmatrix}\chi_{iss\_mid\_t}(t)\\\chi_{iss\_mid\_d}(t)\end{bmatrix}$$
$$+l_{inj\_top\_iss}\cdot F_{iss\_top}(t)\cdot\begin{bmatrix}\chi_{iss\_top\_t}(t)\\\chi_{iss\_top\_d}(t)\end{bmatrix}$$

$$R_{inj\_}(t)=\frac{F_{plas\_in\_t}(t)}{n_{pellet\_t}(1-\eta_{inj\_err})} \tag{2.64}$$

$$F_{inj\_prop}(t)=n_{prop}\cdot R_{inj\_}(t) \tag{2.65}$$

$$F_{inj\_prop}(t)=F_{sto\_out\_p}(t)+l_{iss\_p}\cdot F_{iss\_top}(t) \tag{2.66}$$

$$F_{inj\_exh}(t)=(\frac{F_{plas\_in\_t}(t)+F_{plas\_in\_d}(t)}{(1-\eta_{inj\_err})}+F_{plas\_in\_p}(t))\cdot\eta_{inj\_err}$$
$$+F_{inj\_prop}(t)\cdot\eta_{cap\_prop} \tag{2.67}$$

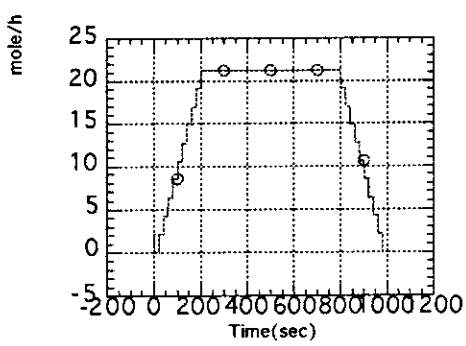


Fig.2.33 Flow of tritium to plasma through pellet injector sub system (mole/h) : Fplas_in_t(t)

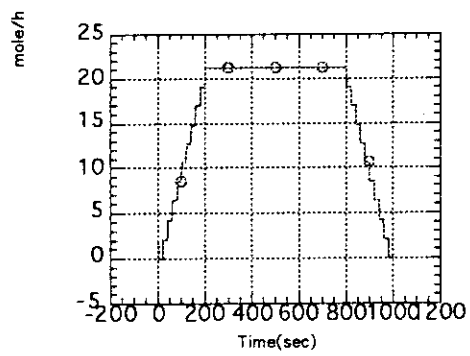


Fig.2.34 Flow of deuterium to plasma through pellet injector sub system (mole/h) : Fplas_in_d(t)

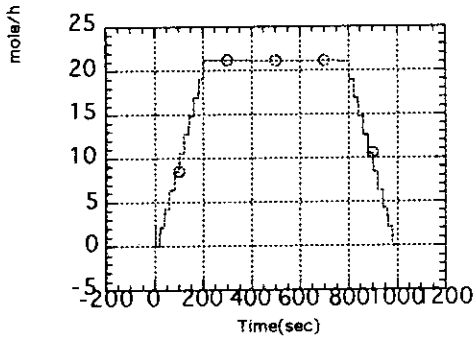


Fig.2.35 Flow of protium to plasma through pellet injector sub system (mole/h) : Fplas_in_p(t)

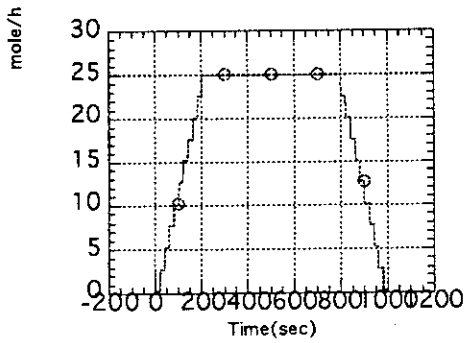


Fig.2.36 Exhaust flow rate from pellet injector sub system (mole/h): Finj_exh(t)

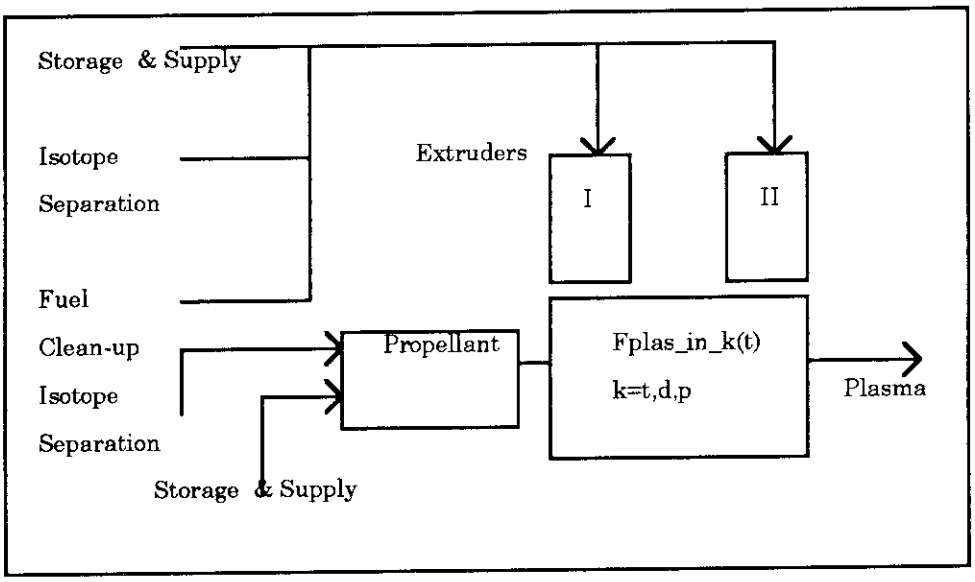


Fig.2.37 Flow diagram for pellet injector sub system

(9) ガスパフサブシステム

ガフパフサブシステムは、プラズマ燃焼の大きさに応じて定められるプラズマエッジ部の必要量を供給する。供給すべき量は、燃料精製サブシステムと同位体分離サブシステムからの戻り量と貯蔵・供給サブシステムからの供給量によりまかなわれる。これらの量は、リザーバタンクに確保されるものとした。

a. 状態方程式

プラズマエッジ部への供給量の時間変化は、燃料精製サブシステムと同位体分離サブシステムからの戻り量と貯蔵・供給サブシステムからの供給量との平衡式を満足するものとした。過不足分は、リザーバタンクに確保されるものとした。

$$\begin{bmatrix} F_{gas\_t}(t) \\ F_{gas\_d}(t) \end{bmatrix} = F_{gas\_in\_tri}(t) \cdot \begin{bmatrix} \chi_{sto\_tri\_t}(t) \\ \chi_{sto\_tri\_d}(t) \end{bmatrix} + F_{gas\_in\_deu}(t) \cdot \begin{bmatrix} \chi_{sto\_deu\_t}(t) \\ \chi_{sro\_deu\_d}(t) \end{bmatrix}$$
$$+h_1 \cdot F_{puri\_out}(t) \cdot \begin{bmatrix} \chi_{iss\_in\_t}(t) \\ \chi_{iss\_in\_d}(t) \end{bmatrix}$$
$$+m_{gas\_mid\_iss} \cdot F_{iss\_mid}(t) \cdot \begin{bmatrix} \chi_{iss\_mid\_t}(t) \\ \chi_{iss\_mid\_d}(t) \end{bmatrix} \tag{2.68}$$
$$+m_{gas\_top\_iss} \cdot F_{iss\_top}(t) \cdot \begin{bmatrix} \chi_{iss\_top\_t}(t) \\ \chi_{iss\_top\_d}(t) \end{bmatrix}$$

ここで、

$F_{gas\_t}(t)$: tritium flow to plasma from gas puffer sub system (mole/h)

$F_{gas\_d}(t)$: deuterium flow to plasma from gas puffer sub system (mole/h)

$F_{gas\_in\_tri}(t)$: feed from tritium rich storage (mole/h)

$F_{gas\_in\_deu}(t)$: feed from deuterium rich storage (mole/h)

$F_{gas\_t}(t)$ 、$F_{gas\_d}(t)$ 、のシミュレーション結果を Fig. 2.38、Fig. 2.39 に示す。

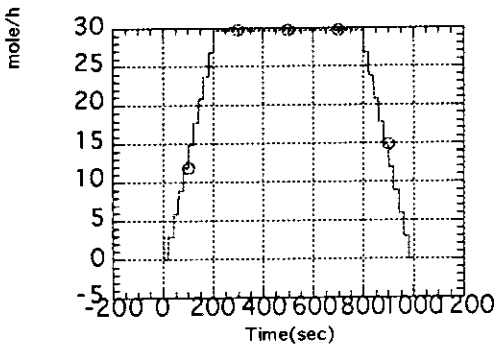


Fig. 2.38 Specified flow of tritium to plasma through gas puffer sub system (mole/h): Fgas_t(t)

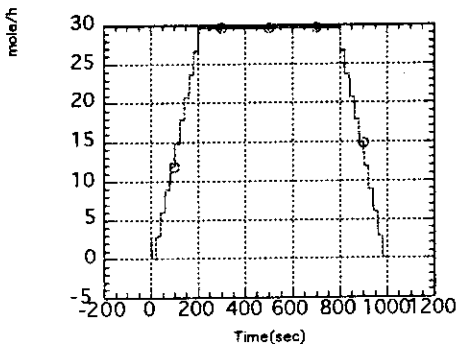


Fig. 2.39 Specified flow of deuterium to plasma through gas puffer sub system (mole/h): Fgas_d(t)

b. 出力方程式

トーラスへの流入量、$F_{gas\_tot}(t)$、をモル分率、$\chi_{gas\_t}(t)$、$\chi_{gas\_d}(t)$、を導入して次式で定めた。

$$F_{gas\_tot}(t) = \frac{F_{gas\_t}(t) + F_{gas\_d}(t)}{\chi_{gas\_t}(t) + \chi_{gas\_d}(t)} \tag{2.69}$$

$F_{gas_tot}(t)$のシミュレーション結果を Fig. 2.40 に示す。導入されたシステム変数の位置関係を Fig. 2.41 に示す。

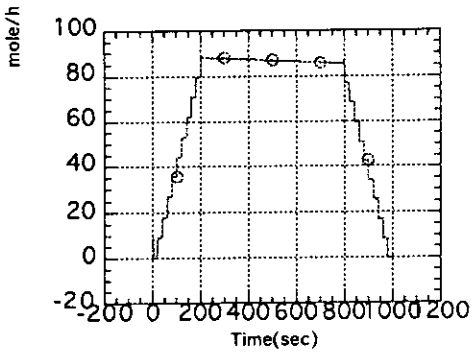


Fig.2.40 Total flow rate to plasma through gas puffer (mole/h) : Fgas_tot(t)

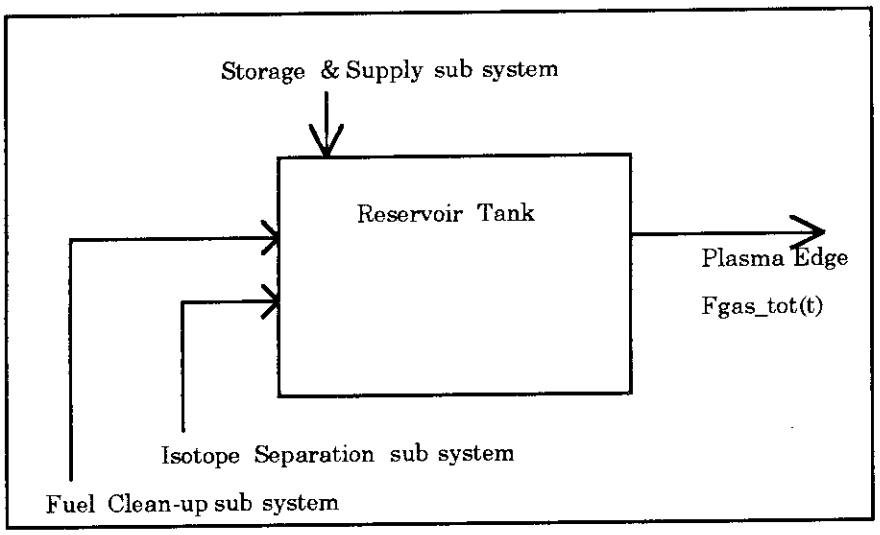


Fig.2.41 Flow diagram for gas puffer sub system

(10) 貯蔵・供給サブシステム

貯蔵供給サブシステムは、トリチウム、重水素、水素の初期装荷量を貯蔵するとともに、燃焼に要求される量を供給する。トリチウム回収を司るサブシステムからの戻り量は本サブシステムに戻される。供給量の不足分は、外部供給によりまかなわられるとした。

a. 状態方程式

トリチウムと重水素の貯蔵槽の増減を、戻り量と供給量からの差から求め、槽内に蓄積する量を次式で定義した。

$$\frac{d}{dt}\begin{bmatrix} N_{sto\_tri\_}(t) \\ N_{sto\_deu\_}(t) \\ N_{sto\_pro\_}(t) \end{bmatrix} = \begin{bmatrix} F_{sto\_tri\_in}(t) \\ F_{sto\_deu\_in}(t) \\ F_{sto\_pro\_in}(t) \end{bmatrix} - \begin{bmatrix} F_{sto\_tri\_out}(t) \\ F_{sto\_deu\_out}(t) \\ F_{sto\_pro\_out}(t) \end{bmatrix} \tag{2.70}$$

槽内におけるトリチウムの時間当りの変化量、$T_{sto\_tri}(t)$、を、式(2.70)で導入された状態量、$N_{sto\_tri\_}(t)$、から次式で定義した。

$$\frac{dT_{sto\_tri\_}(t)}{dt} = F_{sto\_tri\_in}(t)\cdot\chi_{iss\_bot\_t}(t) - F_{sto\_tri\_out}(t)\cdot\frac{T_{sto\_tri}(t)}{N_{sto\_tri\_}(t)} \tag{2.71}$$

式(2.71)で定められるトリチウムの変化量を積算して、槽内に蓄積するトリチウム量を次式より定めた。

$$K_{sto\_tri\_}(t) = \sum_{i=1}^{t}\left\{T_{sto\_tri\_}(i-1) - T_{sto\_tri\_}(i)\right\} \tag{2.72}$$

槽内におけるトリチウムの時間当りの変化量、$T_{sto\_tri\_}(t)$、と蓄積する量、$K_{sto\_tri\_}(t)$、のシミュレーション結果を Fig. 2.42、Fig. 2.43 に示す。導入されたシステム変数の位置関係を Fig. 2.44 に示す。

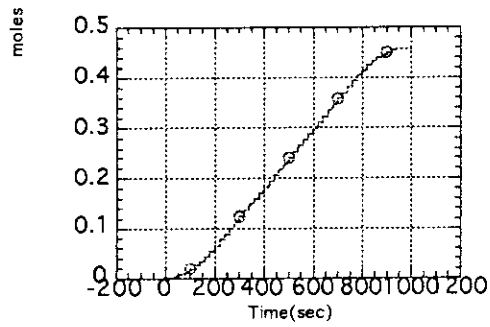


Fig. 2.42 Total amount of tritium in internal tritium rich storage (moles) : Tsto_tri_(t)

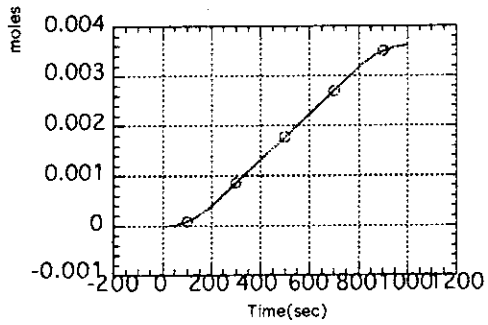


Fig. 2.43 Accumulated tritium inventory change in storage (moles) : Ksto_tri_(t)

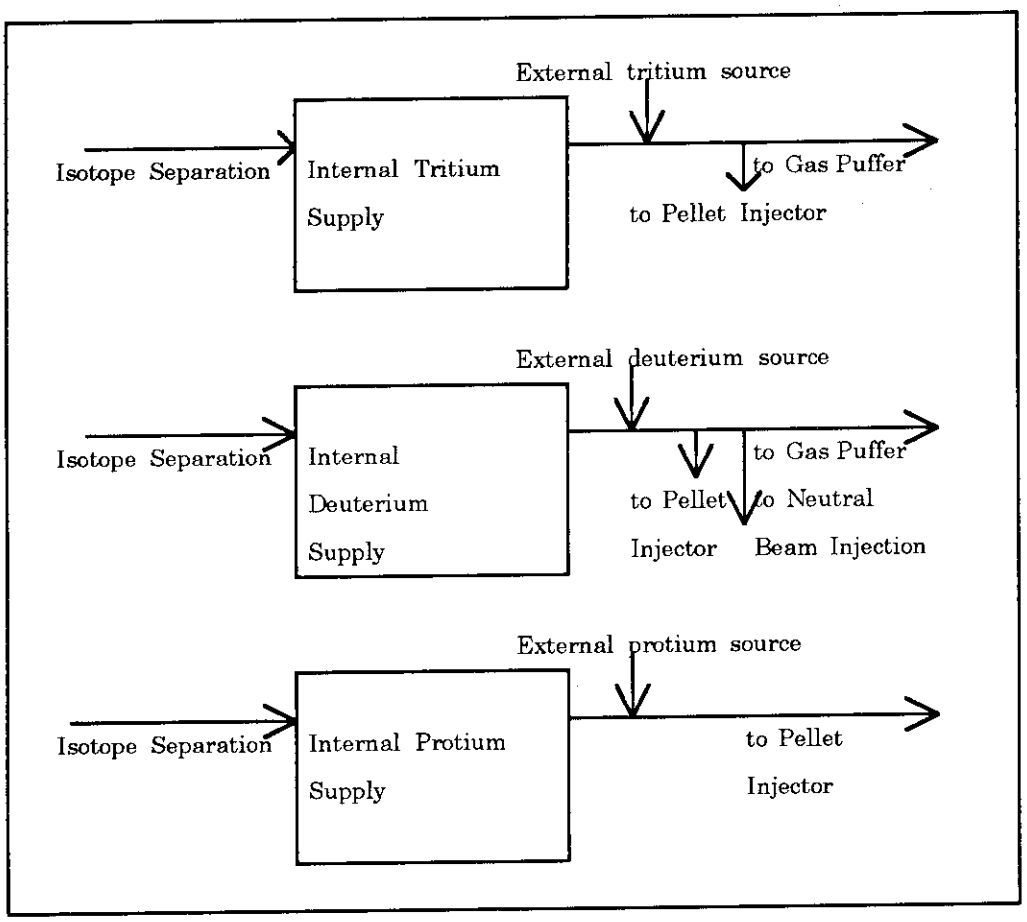


Fig. 2.44 Flow diagram for Storage and Supply sub system

3. まとめ

単パルス運転の燃焼状態を入力（負荷）としたときの各サブシステムのシステム変数の定義内容とシミュレーション結果を第2.3章に示した。

(1)システム変数

システム変数の定義の方法は、第2.1章に示したように

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{2.1}$$

とし、強制項、$\mathbf{u}(t)$、に1パルス運転の燃焼状態を表わす関数形を与えた。時刻tの時の解、$\mathbf{x}(t)$、は、自由応答の解と強制振動の解の和として与えられ、強制項は、時刻t-1から時刻tにわたって畳み込まれる。変化が急峻もしくは不連続の場合にこの効果が計算開始の数ステップの間に振動として現われることがある（Fig. 2.4とFig. 2.5）。

(2)インベントリ

各サブシステムのインベントリの時間変化は、状態方程式(2.1)において、強制項、$\mathbf{u}(t)$、の中に他のサブシステムに引き渡される量を時間の関数形で与えて$\mathbf{x}(t)$を解くこ

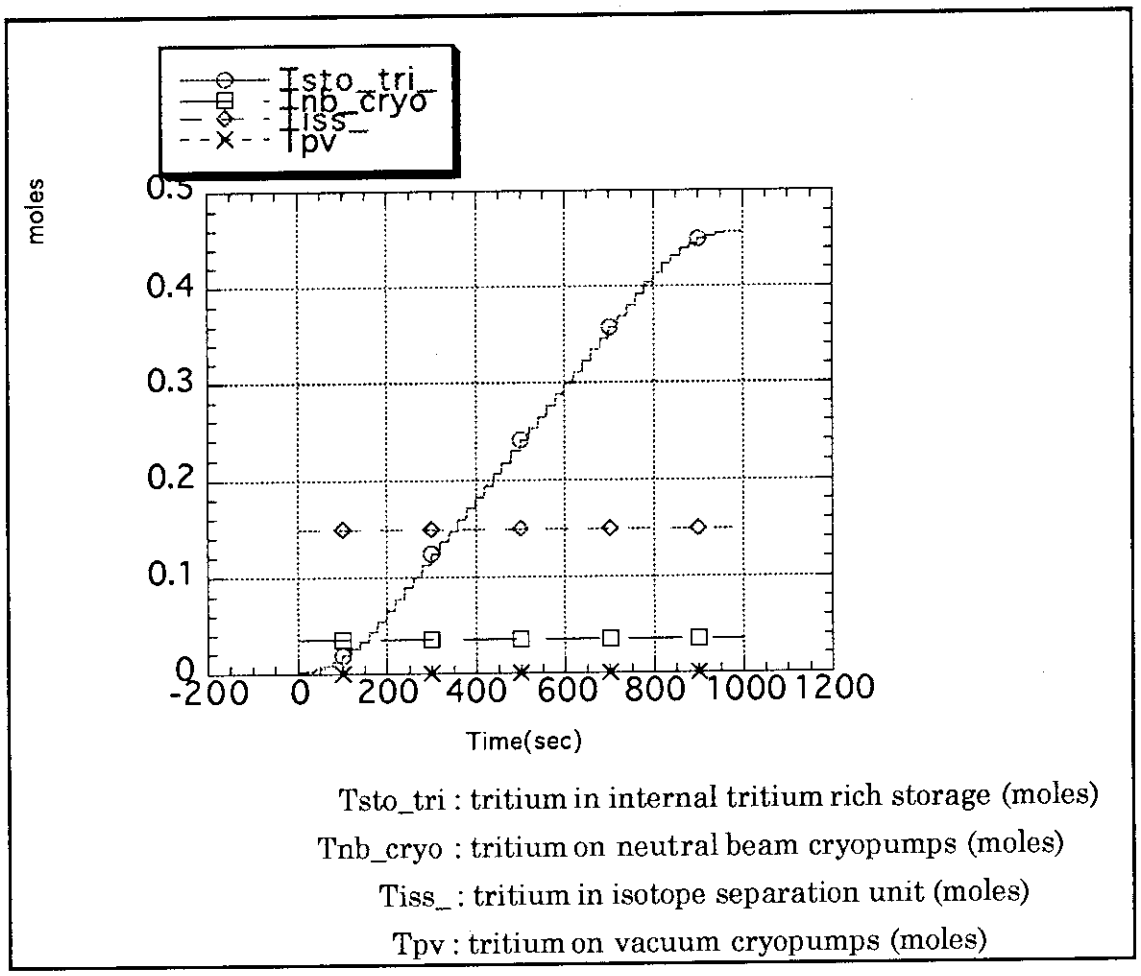


Fig. 3.1 Tritium inventory (moles)

とによって得られる。サブシステムのインベントリの時間変化をFig. 3.1に示す。

(3)サブシステムの運転キャンペーン及びシミュレーション結果

パルス運転の模擬として、燃焼の時間変化を台形とし、ヒーティングとバーニングターミネションを同時間とした。ITER概念設計書によるパルス運転キャンペーンは、ヒーティングとバーニングターミネションの所要時間が異なる。この時間の相違の応答のの結果としては、同位体分離サブシステム以外システム変数の時間応答に大きな相違はもたらさない。また、時間応答を支配するほとんどのシステム定数は時間のオーダーである。したがって、1000秒のシミュレーションの期間にその実質的な効果は現われない。ITER概念設計書による真空排気サブシステムの運転キャンペーンは、その運転サイクルをバッチサイクルと定めてある。本シミュレーションは、1パルス運転時におけるシステム変数の時間応答をみるために、連続運転を仮定したが、この仮定のもとでの応答として、以下の結果が観測された。

a)燃料精製サブシステムから同位体分離サブシステムへ流入するトリチウム、重水素、水素の流量変化（Fig. 2.21、Fig. 2.22、Fig. 2.23）が急峻である。この引き渡されるシステム変数の時間変化は、同サブシステムの時間応答の主要要因となる。

b)同位体分離サブシステムにたまるトリチウムインベントリ（Fig. 3.1のTiss_)の主な要因は、燃料精製サブシステムの待機行程のプリローディング運転からの受け入れによるものである（(2.37)式右辺第2項）。

これらの対策として、以下の対応が考えられる。

a) 同位体分離サブシステムのシステム設計から入力の時間変化の影響を除くため、入力に緩衝域（バッファタンク）を設ける。

b) 燃料精製サブシステムのプリローディング運転のシステム設計。

謝辞

本シミュレーションコードの作成は、その過程の段階において、週一回の頻度で開催されている室内会議における技術的な討議を経て成されたものである。ここに、核融合炉システム研究室各員の建設的な討議と助言に感謝いたします。第2章に示す動特性モデルの作定にあたっては、参考文献[3]の他、多くの教科書を参考にした。2.1.2節 b. 動特性モデルの解法に示される方法は、臨界安全研究室中島健副主任研究員のご教示によるものである。ここに、感謝いたします。

とによって得られる。サブシステムのインベントリの時間変化をFig. 3.1に示す。

(3)サブシステムの運転キャンペーン及びシミュレーション結果

パルス運転の模擬として、燃焼の時間変化を台形とし、ヒーティングとバーニングターミネションを同時間とした。ITER概念設計書によるパルス運転キャンペーンは、ヒーティングとバーニングターミネションの所要時間が異なる。この時間の相違の応答のの結果としては、同位体分離サブシステム以外システム変数の時間応答に大きな相違はもたらさない。また、時間応答を支配するほとんどのシステム定数は時間のオーダーである。したがって、1000秒のシミュレーションの期間にその実質的な効果は現われない。ITER概念設計書による真空排気サブシステムの運転キャンペーンは、その運転サイクルをバッチサイクルと定めてある。本シミュレーションは、1パルス運転時におけるシステム変数の時間応答をみるために、連続運転を仮定したが、この仮定のもとでの応答として、以下の結果が観測された。

a)燃料精製サブシステムから同位体分離サブシステムへ流入するトリチウム、重水素、水素の流量変化（Fig. 2.21、Fig. 2.22、Fig. 2.23）が急峻である。この引き渡されるシステム変数の時間変化は、同サブシステムの時間応答の主要要因となる。

b)同位体分離サブシステムにたまるトリチウムインベントリ（Fig. 3.1のTiss_)の主な要因は、燃料精製サブシステムの待機行程のプリローディング運転からの受け入れによるものである（(2.37)式右辺第2項）。

これらの対策として、以下の対応が考えられる。

a) 同位体分離サブシステムのシステム設計から入力の時間変化の影響を除くため、入力に緩衝域（バッファタンク）を設ける。

b) 燃料精製サブシステムのプリローディング運転のシステム設計。

謝辞

本シミュレーションコードの作成は、その過程の段階において、週一回の頻度で開催されている室内会議における技術的な討議を経て成されたものである。ここに、核融合炉システム研究室各員の建設的な討議と助言に感謝いたします。第2章に示す動特性モデルの作定にあたっては、参考文献[3]の他、多くの教科書を参考にした。2.1.2節 b. 動特性モデルの解法に示される方法は、臨界安全研究室中島健副主任研究員のご教示によるものである。ここに、感謝いたします。

参考文献

[1] ITER Documentation Series, No. 35, IAEA, Vienna,1991.

[2] R. T. McGrath et al., Design Consideration for ITER Plasma Facing Components, Sandia Report, SAND 89-0901, July 1989.

[3] 高橋安人, システムと制御, 東京,岩波書店, 1970.

[4] 中島健、大西信秋, 溶液燃料体系の動特性コード(AGNES)の開発, JAERI-M 85-212, Jan. 1986.

付録1. 動特性モデル開発の作業環境

1. 作業環境および計算の条件

(1) 計算実施時の機器構成

a. 使用したハードウエア

トリチウムインベントリシミュレーションプログラムの作成および実行環境としてアップル・コンピュータ社製パーソナルコンピュータMachintoshを使用した。

b. 使用言語とコンパイラ

シミュレーションプログラムに使用した言語はFORTRAN77であり、コンパイラにはMachitosh用FORTRANコンパイラ Mac Fortran II、もしくはAbsoft FORTRAN77（いずれもAbSoft社製）を使用した。このコンパイラではCPUに68000シリーズを使用するMachitoshとPowerPCを使用するMachitoshの両方で動作可能な実行形式（FATバイナリ）を生成することができる。

c. 計算結果の表示

計算結果はディスク上にプログラム変数名と同じ名前をもつファイルとして格納される。結果の表示にはAbelbeck Software社製グラフ表示ソフトKaleida Graphを使用した。多くの変数グラフ表示処理が容易に行なえるように計算結果のファイルをドラッグ・アンド・ドロップするだけで必要なグラフがKaleidaGpaphによって自動的に作成されるようなApple Scriptを作成した。

2. シミュレートされた状態変数のリストと計算条件

シミュレーションプログラムで使用された変数名とそれらの意味をTableA.1から Table A.8までに各構成機器ごとに示す。"変数名(t)"の様な変数は時間変化をともなう変数であり、"(t)"のついていない変数はシミュレーション中で一定値をとる定数であることを示している。

計算はシミュレーション上の時間で1000秒にわたって行なわれ、時間ステップ幅はすべて2秒とした。計算に必要な実時間はPower Machintosh 7200/90 (CPU: PowerPC602 90MHz, メモリ: 48Mbyte main, 256Kbyte secondary cache, OS: MacOS7.5)で５０秒程度であった。

Table A.1 Variables and constants used for simulation of Plasma Chamber

variable/constant	description	constant value
Pfusion(t)	fusion power (MW)	
Feedf_t(t)	amount of tritium deliberatelyfueledto the plasma from fuelers (mole/h)	
Fuel_t(t)	total tritium fueledto the plasma (mole/h)	
Fuel_d(t)	total deuteriumfueledto the plasma (mole/h)	
Fuel_p(t)	total protium sources to the plasma (mole/h)	
Burn_t(t)	tritium burnup rate (mole/h)	
Burn_d(t)	deuterium burnup rate (mole/h)	
Gen_p(t)	formation rate of protium from advanced fuel reactions (mole/h)	
Ef(t)	amount of fusion energy deliberately caused by one reaction (MeV)	1.75D1
Fex_t(t)	exhaust rate of tritium from plasma (mole/h)	
Fex_d(t)	exhaust rate of deuterium from plasma (mole/h)	
Fex_p(t)	exhaust rate of protium from plasma (mole/h)	
Fex_he(t)	exhaust rate of helium from plasma (mole/h)	
xedge_t(t)	isotopic fractionof exhaust tritium from plasma	
xedge_d(t)	isotopic fractionof exhaust deuterium from plasma	
xedge_p(t)	isotopic fractionof exhaust protium from plasma	
Bulkdiv_t(t)	adsorption of tritium from the edge into divertor (moles)	
Bulkfw_t(t)	adsorption of tritium from the edge into first wall (moles)	
d_Bulkdiv_t(t)	rate of adsorption of tritium from the edge into divertor (mole/h)	
d_Bulkfw_t(t)	rate ofadsorption of tritium from the edge into first wall (mole/h)	
Co_d_t(t)	tritium codeposited onto plasma facingcomponents (moles)	
d_Co_d_t(t)	rate of tritium codeposition onto plasma facingcomponents (mole/h)	

Dust_t(t)	tritium in tokamak dust (moles)	
d_Dust_t(t)	rate of change of tritium in tokamak dust (mole/h)	
Bulkdiv_d(t)	adsorption of deuterium from edge into divertor (moles)	
Bulkfw_d(t)	adsorption of deuterium from edge into first wall (moles)	
d_Bulkdiv_d(t)	rate of adsorption of deuterium from edge into divertor (mole/h)	
d_Bulkfw_d(t)	rate of adsorption of deuterium from edge into first wall (mole/h)	
Co_d_d(t)	deuterium codeposited onto plasma facing components (moles)	
d_Co_d_d(t)	rate of deuterium codeposition onto plasma facing components (mole/h)	
Dust_d(t)	deuterium in tokamak dust (moles)	
d_Dust_d(t)	rate of change of deuterium in tokamak dust (mole/h)	
Bulkdiv_p(t)	adsorption of protium from edge into divertor (moles)	
Bulkfw_p(t)	adsorption of protium from edge into first wall (moles)	
d_Bulkdiv_p(t)	rate of adsorption of protium from edge into divertor (mole/h)	
d_Bulkfw_p(t)	rate of adsorption of protium from edge into first wall (mole/h)	
Co_d_p(t)	protium codeposited onto plasma facing components (moles)	
d_Co_d_p(t)	rate of protium codeposition onto plasma facing components (mole/h)	
Dust_p(t)	protium in tokamak dust (moles)	
d_Dust_p(t)	rate of change of protium in tokamak dust (mole/h)	
εprot	relative combined reaction rates of proton branch of DD reaction and D3H reaction compared to DT	1.0D-2
εbu	fractional burn-up of tritium	5.0D-2
fin_pellet	fraction of tritium fueling through pellet injection	5.0D-1
fin_gas	fraction of tritium fieling through gas puffing	7.0D-1
PFC_t(t)	total tritium in plasma facing components (moles)	
Tpfc_m(t)	mobilizable tritium in plasma facing components (moles)	
PFC_d(t)	total deuterium in plasma facing components (moles)	
PFC_p(t)	total protium in plasma facing components (moles)	
Temperature	Diverter temperature (C degree)	1000
Tedge	Plasma Facing Material temperature (eV)	60
Medge	Plasma Facing Material (1/2/3=C/Be/W)	1
mu_A	mobility fraction of absorbed inventory	1.0D-1
mu_C	mobility fraction of codeposited inventory	1.0D0
mu_D	mobility fraction of dust inventory	1.0D0
dc	dust concentration ratio (kg_tritium/kg_dust)	1.0D-1
Vdiv	total volume of divertor (m3)	(2.0D-5)*(2.0D5)

Vfw	total volume of first wall (m3)	1.6D3*2.0*7.0
Ad	divertor wall surfacearea (m2)	2.0D2
lambda_T	Decay constant of Tritium (1/h)	6.380D-6

Table A.2 Variables and constants used for simulation of Exhaust Processing sub system

variable/constant	description	constant value
Tvacuum (t)	tritium inventory per pumps (moles)	
Tp_reg(t)	tritium inventory per pump (batch operation) at regeneration time (moles)	
Tp_tot(t)	total tritium inventory on all pumps	
Tp_tot_m(t)	total mobilizable tritium inventory on all pumps (moles)	
Rvacuum_t(t)	removal rate of tritium from continuos regeneration cryopump (mole/h)	
Dvacuum(t)	Deuterium inventory per pumps (moles).	
Rvacuum_d(t)	removal rate of deuterium from continuos regeneration cryopump (mole/h)	
Dp_reg(t)	tritium inventory per pump (batch operation) at regeneration time (moles)	
Dp_tot(t)	total deuterium inventory on all pumps (moles).	
Rvacuum_p(t)	removal rate of protium from continuos regeneration cryopump (mole/h).	
Pp_reg(t)	protium inventory per pump (batch operation) at regeneration time (moles).	
Pp_tot(t)	total protium inventory on all pumps (moles).	
Pvacuum(t)	protium inventory per pumps (moles).	
Rvacuum_he(t)	removal rate of helium from continuos regeneraton cryopump (mole/h)	
Hevacuum(t)	helium inventory per pumps (moles).	
Hep_reg(t)	helium inventory per pump (batch operation) at regeneration time (moles).	
Hep_tot(t)	total deuterium inventory on all pumps (moles)	
Fv_out(t)	total flow from vacuum pumps to fuel clean-up unit (mole/h).	
npump	number of operating pumps	16
treg	regeneration time for batch cryopumps in vacuum system (h)	2.0D0/3.0D0
tpump	pumping for batch cryopumps in vacuum system (h)	4.0D0/3.0D0
εscrap	efficiency of removal of cryodeposit for mechanical continuous regeneration cryopump	2.0D-1
rscrap	scraping rate for mechanical continuous regeneration scheme (m2/h)	1.92D0
acryo	pump surface area for mechanical continuous regeneration scheme (m2).	3.2D-2

mu_v	mobility fraction of inventory on cryopump.	1.0D-1

Table A.3 Variables and constants used for simulation of Fuel Clean-up sub system

variable/constant	description	constant value
Fpuri_in(t)	total flow from vacuum pumps to purification (mole/h)	
it_(t)	isotopic fraction of tritium on vacuum system cryopumps	
id_(t)	isotopic fraction of deuterium on vacuum system cryopumps	
ip_(t)	isotopic fraction of protium on vacuum system cryopumps	
xnhydro_(t)	fractin of non-hydrogen containing impurities in adjusted flow to purification	
xh_imp(t)	fraction of hydrogen containing impurities in adjusted flow to purification	
Fadj_puri(t)	adjusted flow rate to purification considering impurities (mole/h)	
Fhydro_pure(t)	flow of pure atoms to purification (mole/h)	
Fhydro_imp(t)	flow of hydrogen containing impurities to purification (mole/h)	
Fnhydro_imp(t)	flow of non-hydrogen containing impurities to purification(mole/h)	
Fpure_t(t)	flow of pure tritium atoms to purification (mole/h)	
Fpure_d(t)	flow of pure deuterium atoms to purification (mole/h)	
Fpure_p(t)	flow of pure protium atoms to purification (mole/h)	
F cq4_imp(t)	flow of methane to purification (mole/h)	
F nq3_imp(t)	flow of ammonia to purification (mole/h)	
Fq2o_imp(t)	flow of water to purification (mole/h)	
Fhydro_imp_w(t)	constituents of hydrogen flow containing impurities (mole/h)	
f_hydro_imp_w(t)	weighting fraction of constituents of hydrogen flow containing impurities	
Nimpt(t)	total impurities on purification unit (moles)	
Nhydro_imp(t)	total hydrogen in hydrogen-containing impurities on purification unit (moles)	
Nnhydro_imp(t)	total non-hydro impurities on purification unit (moles)	
Ncq4_imp(t)	methane on purification unit (moles)	
Nnq3_imp(t)	ammonia on purification unit (moles)	
Nq2o_imp(t)	water on purification unit (moles)	
Timp_(t)	total tritium on purification unit (moles)	
Dimp_(t)	total deuterium on purification unit (moles)	
Pimp_(t)	total protium on purification unit (moles)	
Fpuri_in_h2(t)	flow of pure hydrogen molecules to purification (mole/h)	
Fsub_in_t2(t)	flow of pure tritium molecules to standby unit (mole/h)	

Fsub_in_d2(t)	flow of pure deuterium molecules to standby unit (mole/h)	
Fsub_in_p2(t)	flow of pure protium molecules to standby unit (mole/h)	
Fsub_in(t)_h2	flow of pure fydrogen molecules to standby unit (mole/h)	
Mmsieve(t)	number of unfilled adsorption sites for hydrogen on standby uint (moles)	
xsub_t(t)	mole fraction of tritium entering standby unit	
xsub_d(t)	mole fraction of deuterium entering standby unit	
xsub_p(t)	mole fraction of protium entering standby unit	
T2_sub(t)	tritium molecules on standby unit (moles)	
D2_sub(t)	deuterium molecules on standby unit (moles)	
P2_sub(t)	protium molecules on standby unit (moles)	
Nhydro_sub(t)	total hydrogen on standby unit (moles)	
Fsub_out_t(t)	flow of pure tritium atoms out of standby unit (mole/h)	
Fsub_out_d(t)	flow of pure deuterium atoms out of standby unit (mole/h)	
Fsub_out_p(t)	flow of pure protium atoms out of standby unit (mole/h)	
Fsub_out_h(t)	flow of pure hydrogen atoms out of standby unit (mole/h)	
Treg_out(t)	tritium on regenerating molecular sieve beds (moles)	
Ecq4_reg(t)	evolution of methane from molecular sieve during regeneration (mole/h)	
Enq3_reg(t)	evolution of ammonia from molecular sieve during regeneration (mole/h)	
Eq2o_reg(t)	evolution of water from molecular sieve during regeneration (mole/h)	
Ncq4_reg(t)	amount of methane on purification bed at regeneration (moles)	
Nnq3_reg(t)	amount of ammonia on purification bed at regeneration (moles)	
Nq2o_reg(t)	amount of water on purification bed at regeneration (moles)	
Fpmem_hydro(t)	flow of hydrogen to palladium membrane reactor (mole/h)	
Tpmem(t)	tritium on palladium membrane reactor (moles)	
Dpmem(t)	deuterium on palladium membrane reactor (moles)	
Ppmem(t)	protium on palladium membrane reactor (moles)	
xpmem_t(t)	isotopic fraction of tritium on molecular sieve at end of purification	
xpmem_d(t)	isotopic fraction of deuterium on molecular sieve at end of purification	
xpmem_p(t)	isotopic fraction of protium on molecular sieve at end of purification	
Fpuri_out_t(t)	flow of tritium to isotope separarion system (mole/h)	
Fpuri_out_d(t)	flow of deuterium to isotope separation system (mole/h)	
Fpuri_out_p(t)	flow of protium to isotope separation system (mole/h)	

Fpuri_out(t)	total flow to isotope separation system (mole/h)	
Xiss_in_t(t)	mole fraction of tritium in stream from fuel clean-up unit to isotope separation system	
Xiss_in_d(t)	mole fraction of deuterium in stream from fuel clean-up unit to isotope separation system	
Xiss_in_p(t)	mole fraction of protium in stream from fuel clean-up unit to isotope separation system	
Fpure_(t)	total flow of pure tritium, deuterium, protium and helium atoms to purification (mole/h)	
Fimp_(t)	total flowa of impurities to purification (mole/h)	
Timp_m(t)	total mobilizable tritium inventory on purification unit (moles)	
Treg_out_m(t)	mobilizable tritium on regenerating molecular sieve beds (moles)	
mu_mob	mobility fraction of inventory on fuel clean-up unit molecular sieve bed	1.0D-2
Lhydro_	loading of mole sieves for hydrogen (scc/g)	1.2D2
Limp_	loading of mole sieves for impurities (scc/g)	1.0D2
Ns_max	maximum number of unfilled adsorption sites for hydrogen on fuel clean-up unit standby bed preloading with hydrogen (mole)	1.77D1
tmsieve_reg	regeneration time for molecular sieve (h)	4.0D0
mu_reg_m	mobility fraction of inventory on fuel clean-up unit molecular sieve beds during regeneration	1.0D0
tpmem	time constant for palladium membrane reactor (h)	2.0D0
w1	number of hydrogen atoms in a methane molecule	4.0D0
w2	number of hydrogen atoms in an ammonia molecule	3.0D0
w3	number of hydrogen atoms in a water molecule	2.0D0
xcq4_	specified mole fraction of methane in adjusted flow to purification	5.6D-1
xnq3_	specified mole fraction of ammonia in adjusted flow to purification	4.0D-2
xq2o_	specified mole fraction of water in adjusted flow to purification	8.0D-2

Table A.4 Variables and constants used for simulation of Isotope Separation sub system

variable/constant	description	constant value
Fiss_in(t)	total flow to isotope separation system (mole/h)	
Fiss_nb(t)	flow from neutral beam to isotope separation system (mole/h)	
Fiss_in_t(t)	flow of tritium to isotope separation system (mole/h)	
Xiss_t(t)	mole fraction of tritium in stream from fuel clean-up unit	
Xnb_t(t)	mole fraction of tritium in stream from neutral beam cryopump	
Xinj_t(t)	mole fraction of tritium in pellet injector exhaust stream	
Fiss_in_d(t)	flow of deuterium to isotope separation system (mole/h)	
Xiss_d(t)	mole fraction of deuterium in stream from fuel clean-up unit	
Xnb_d(t)	mole fraction of deuterium in stream from neutral beam cryopump fuel clean-up unit	
Xinj_d(t)	mole fraction of deuterium in pellet injector exhaust stream	
Fiss_in_p(t)	flow of protium to isotope separation system (mole/h)	
Niss_(t)	total amount in isotope separation system (moles)	
Tiss_(t)	amount of tritium in isotope separation system (moles)	
Tiss_m(t)	amount of mobilizable tritium in isotope saparation system (moles)	
Diss_(t)	amount of deuterium in isotope separation system (moles)	
Poiss_(t)	amount of protium in isotope separation system (moles)	
Xiss_p(t)	mole fraction of protium in isotope separation system	
Fiss_top_(t)	flow rate from top of isotope separation system (mole/h)	
Fiss_mid_(t)	flow rate from middle of isotope separation system (mole/h)	
Fiss_bot_(t)	flow rate from bottom of isotope separation system (mole/h)	
Fiss_inj(t)	flow directed to isotope separation system from pellet injector (mole/h)	
Fbl_cool	flow rate from coolant system (mole/h)	1.0D-2
Fbl_	flow rate from blanket system (mole/h)	5.0D-2
h1	fraction of stream from fuel clean-up system not processed and directed to gas puffer	3.0D-1
h3	fraction of stream from fuel clean-up system not processed and directed to pellet injector	3.0D-1
Fragnb	switch to include neutral beam in isotope separation system balance (included if equals one, not included if equals zero)	0.0D0
Fragiss_inj	switch to include pellet injector in isotope separation system balance (included if equals one, not included if equals zero)	0.0D0
Xc_t	mole fraction of tritium in coolant system	3.0D-1

Xbl_t	mole fraction of tritium in breeder system	3.0D-1
Xc_d	mole fraction of deuterium in coolant stream	3.0D-1
Xbl_d	mole fraction of deuterium in breeder stream	3.0D-1
t_iss	residence time in isotope separation system (h)	2.22D0
mu_iss_	mobility fraction of inventory in isotope separation system	1.0D0
Xiss_top_t	mole fraction of tritium from top of isotope separation system	6.0D0/9.0D0
Xiss_top_d	mole fraction of deuterium from top of isotope separation system	2.0D0/9.0D0
Xiss_top_p	mole fraction of tritium from top of isotope separation system	1.0D0/9.0D0
Xiss_mid_t	mole fraction of tritium from middle of isotope separation system	2.0D0/9.0D0
Xiss_mid_d	mole fraction of deuterium from middle of isotope separation system	6.0D0/9.0D0
Xiss_mid_p	mole fraction of tritium from middle of isotope separation system	1.0D0/9.0D0
Xiss_bot_t	mole fraction of tritium from bottom of isotope separation system	1.0D0/9.0D0
Xiss_bot_d	mole fraction of deuterium from bottom of isotope separation system	2.0D0/9.0D0
Xiss_bot_p	mole fraction of tritium from bottom of isotope separation system	6.0D0/9.0D0

Table A.5 Variables and constants used for simulation of Neutral Beam Injection sub system

variable/constant	description	constant value
Fplas_nb_d(t)	flow of deuterium to plasma passing neutral beam (mole/h)	
Fin_st_d(t)	flow from deuterium rich storage (mole/h)	
Fin_rec_(t)	recycle flow rate to neutral beam from neutral beam cryopump (mole/h)	
Fin_iss_mid(t)	flow rate from middle of isotope separation system (mole/h)	
Xcryo_d(t)	mole fraction of deuterium on neutral beam cryopump	
W_nb(t)	required neutral beam power to be delivered to the plasma (MW)	
Ncryo_(t)	total amount on neutral beam cryopump (moles)	
Fbk_nb_(t)	total backflow from plasma into neutral beam (mole/h)	
Dcryo_(t)	deuterium on neutral beam cryopump (moles)	
Dnb_reg_(t)	amount of deuterium on the neutral beam batch regeneration(moles)	
Tnb_cryo(t)	tritium on neutral beam cryopump (moles)	
Fbk_nb_t(t)	backflow of deuterium from plasma into neutral beam (mole/h)	
Rnb_reg_t(t)	regeneration rate of tritium from neutral beam cryopump (mole/h)	
tnb_	time during pumping cycle for neutral beam pumps (h)	
Tnb_reg_(t)	total amount of tritium on the neutral beam batch regeneration cryopump (moles)	
Tnb_cryo_tot(t)	total tritium on neutral beam cryopumps (moles)	
Tnb_cryo_mob(t)	total mobilizable tritium on neutral beam cryopump (moles)	
Dnb_cryo(t)	deuterium on neutral beam cryopump (moles)	
Fbk_nb_d(t)	backflow of deuterium from plasma into neutral beam (mole/h)	
Rnb_reg_d(t)	regeneration rate of deuterium from neutral beam cryopump (mole/h)	
Dnb_cryo_tot(t)	total deuterium on neutral beam cryopumps (moles)	
Xnb_cryo_t(t)	mole fraction of tritium on neutral beam cryopump	
Xnb_cryo_d(t)	mole fraction of deuterium on neutral beam cryopump	
Xnb_cryo_p(t)	mole fraction of protium on neutral beam cryopump	
Pnb_cryo(t)	protium on neutral beam cryopump (moles)	
Tnb_wall(t)	tritium on neutral beam walls (moles)	
Tnb_wall_mob(t)	mobilizable tritium on neutral beam walls (moles)	
Tnb_tot(t)	total tritium in neutral beam (moles)	
Tnb_tot_mob(t)	total mobilizable tritium in nueutral beam (moles)	
Fnb_cryo(t)	flow rate from neutral beam cryopump (mole/h)	
Crec(t)	fraction of gas from neutral beam cryopump directed to the isotope separation system	

Fplas_nb_(t)	total flow rate to plasma from neutral beam (mole/h)	
Fplas_nb_t(t)	flow of tritium to plasma through neutral beam (mole/h)	
Fplas_nb_p(t)	flow of protium to plasma through neutral beam (mole/h)	
ncycle_p(t)	number of plasma cycles	
Nnb_cryo(t)	total amount on neutral beam cryopump (moles)	
Rnb_reg_(t)	total regeneration rate from neutral beam cryopump (mole/h)	
Rnb_reg_p(t)	regeneration rate of protium from neutral beam cryopump (mole/h)	
eta_nb_tot	overall efficiency of transfer of neutral beam feed to the plasma	3.1D-1
kiss_mid_	faction of flow from middle of isotope separation system directed to the neutral beam	3.0D-1
fcryo_	fraction of feed to neutral beam that flows to cryopomp	3.7D-1
tnb_reg	regeneration time for batch neutral beam cryopumps (h)	2.0D0/3.0D0
tnb_pump	pumping time for neutral beam batch regeneration cyopump (h)	2.0D0
nnb_	pumping cycle	5.0D0
mu_nb_mob	mobility fraction of inventory on neutral beam cryopump	1.0D0
fnb_wall	fraction of feed to neutral beam that ends up on the walls	5.0D-2
mu_wall_	mobility fraction of inventory in neutral beam walls	1.0D0
tcycle_p	plasma burning time (h)	1000d0/3600d0
t_p_ru	ramp-up time for plasma (h)	200d0/3600d0
t_p_rd	ramp-down time for plasma (h)	200d0/3600d0
t_nb_ru	ramp-up time for neutral beam (h)	100d0/3600d0
t_nb_rd	ramp-down time for neutral beam (h)	100d0/3600d0
t_burn	plasma burn time (h)	600d0/3600d0
tcycle_nb	cycle time for neutral beam (h)	(100+600+100) /3600
E_nb	neutral beam particle energy (MeV)	1.3D0

Table A.6 Variables and constants used for simulation of Pellet Injector sub system

variable/constant	description	constant value
Fplas_in_t(t)	flow of tritium to plasma through pellet injector (mole/h)	
Fplas_in_d(t)	flow of deuterium to plasma through pellet injector (mole/h)	
Fplas_in_p(t)	flow of protium to plasma through pellet injector (mole/h)	
Fsto_out_t(t)	flow from tritium rich storage to pellet injector (mole/h)	
Fsto_out_d(t)	flow from deuterium rich storage to pellet injector (mole/h)	
Fsto_out_p(t)	flow from protium rich storage to pellet injector (mole/h)	
Next_12(t)	total moles (of atoms) in pellet injector extruder 1 or 2 (moles)	
Text_12(t)	tritium in pellet injector extruder 1 or 2 (moles)	
Dext_12(t)	deuterium in pellet injector extruder 1 or 2 (moles)	
Pext_12(t)	protium in pellet injector extruder 1 or 2 (moles)	
Xext_12_t(t)	mole fraction of tritium in pellet injector extruder 1 or 2	
Xext_12_d(t)	mole fraction of deuterium in pellet injector extruder 1 or 2	
Xext_12_p(t)	mole fraction of protium in pellet injector extruder 1 or 2	
Next_tot(t)	total in pellet injector extruders (moles)	
Text_tot(t)	total tritium in pellet injector extruders (moles)	
Dext_tot(t)	total deuterium in pellet injector extruders (moles)	
Pext_tot(t)	total protium in pellet injector extruders (moles)	
Text_m(t)	total mobilizable tritium in pellet injectors (moles)	
Vinj_t(t)	volume of tritium in pellet injector (m3)	
Vinj_d(t)	volume of deuterium in pellet injector (m3)	
Vinj_p(t)	volume of protium in pellet injector (m3)	
Xextru_t(t)	mole fraction of tritium in pellet injector extruder	
Xextru_d(t)	mole fraction of deuterium in pellet injector extruder	
Xextru_p(t)	mole fraction of protium in pellet injector extruder	
Nextru_(t)	total amount in pellet injector extruder	
Vinj_tot(t)	total volume in pellet injector extruder	
Dpellet(t)	diameter of pellet injector pellet (m)	
Hpellet(t)	height of pellet injector pellet (m)	
Vpellet	volume of pellet injector pellet (m3)	4.5D-7
npellet_t(t)	number of moles of tritium in pellet injector pellet (moles)	
npellet_d(t)	number of moles of deuterium in pellet injector pellet (moles)	
npellet_p(t)	number of moles of protium in pellet injector pellet (moles)	
Rinj_(t)	pellet repetition rate for pellet injector (Hz)	

Finj_in_tot(t)	total flow rate to plasma by pellet injector pellet (mole/h)	
Fbk_plas_(t)	backflow from plasma into pellet injector (mole/h)	
Fbk_plas_t(t)	tritium backflow from plasma into pellet injector (mole/h)	
Fbk_plas_d(t)	deuterium backflow from plasma into pellet injector (mole/h)	
Fbk_plas_p(t)	protium backflow from plasma into pellet injector (mole/h)	
Fplas_prop(t)	total flow rate to plasma from pellet injector propellant (mole/h)	
Fplas_prop_t(t)	tritium flow rate to plasma from pellet injector propellant (mole/h)	
Fplas_prop_d(t)	deuterium flow rate to plasma from pellet injector propellant (mole/h)	
Fplas_prop_p(t)	protium flow rate to plasma from pellet injector propellant (mole/h)	
Finj_exh(t)	exhaust flow rate from pellet injector (mole/h)	
Finj_exh_t(t)	tritium exhaust flow rate from pellet injector (mole/h)	
Finj_exh_d(t)	deuterium exhaust flow rate from pellet injector (mole/h)	
Xinj_p(t)	mole fraction of protium in pellet injector	
Finj_prop(t)	flow of propellant gas (mole/h)	
Xprop_t(t)	mole fraction of tritium in propellant gas flow (mole/h)	
Xprop_d(t)	mole fraction of deuterium in propellant gas flow (mole/h)	
Xprop_p(t)	mole fraction of protium in propellant gas flow (mole/h)	
Finj_rec(t)	recycle flow rate to sotope separation system from pellet injector (mole/h)	
finj_err	erosion fraction of deuterium and tritium for pellet injector	3.0D-1
linj_mid_iss	fraction of flow from middle of isotope separation system to pellet injector	3.0D-1
linj_bot_iss	fraction of flow from bottom of isotope separation system to pellet injector	3.0D-1
mu_ext	mobility fraction of inventory in extruders	1.0D0
kpellet_t	mole composition of tritium per unit volume of pellet injector pellet	2*5.294D4
kpellet_d	mole composition of deuterium per unit volume of pellet injector pellet	2*4.979D4
kpellet_p	mole composition of protium per unit volume of pellet injector pellet	2*4.287D4
fcap_prop	farction of propellant recaptured beforeentry into plasma	9.996D-1
fnon_iss	fraction of moles in pellet injector exhaust stream not processd bu the isotope separation system	0.7D0
liss_p	fraction of flow from top of isotope separation system directed to pellet injector	3.2D-2
nprop	amount of propellant gas required per pellet for pellet injector (moles)	4.45D-1

Table A.7 Variables and constants used for simulation of Gas Puffer sub system

variable/constant	description	constant value
Fgas_t(t)	flow of tritium to plasma through gas puffer(mole/h)	
Fgas_in_tri(t)	flow from tritium rich storage to gas puffer(mole/h)	
Fgas_in_deu(t)	flow from deuterium rich storage to gas puffer(mole/h)	
Fgas_d(t)	flow of deuterium to plasma (mole/h)	
Ngas_tot(t)	total amount in gas puffer(moles)	
Fgas_tot(t)	total flow rate to plasma from gas puffer(mole/h)	
Tgas_(t)	amount of tritium in gas puffer(moles)	
Dgas_(t)	amount of deuterium in gas puffer(moles)	
Xgas_t(t)	mole fraction of tritium in gas puffer exit stream	
Xgas_d(t)	mole fraction of deuterium in gas puffer exit stream	
Xgas_p(t)	mole fraction of protium in gas puffer exit stream	
Pgas_(t)	amount of protium in gas puffer(moles)	
Fpuff_plas(t)	total flow to plasma from gas puffer and pellet injector propellant (mole/h)	
Tgas_mob(t)	mobilizable tritium in gas puffer(moles)	
mgas_mid_iss	fraction of flow from middle of isotope separation system directed to gas puffer	3.0D-1
mgas_bot_iss	fraction of flow from bottom of isotope separation system directed to gas puffer	3.0D-1
mu_gas	mobility fraction for gas puffer	1.0D0

Table A.8 Variables and constants used for simulation of Storage and Supply sub system

variable/constant	description	constant value
Fsto_tri_in(t)	flow into tritium rich storage (mole/h)	
Fsto_tri_out(t)	flow out of tritium rich storage (mole/h)	
Nsto_tri_(t)	total amount in internal tritium rich storage (moles)	
Tsto_tri_(t)	total amount of tritium in internal tritium rich storage (moles)	
Tsto_tri_m(t)	"mobilizable tritium in internal deuterium rich storage (moles)	
Dsto_tri_(t)	total amount of deuterium in internal tritium rich storage (moles)	
Psto_tri_(t)	total amount of protium in internal tritium rich storage (moles)	
Xsto_tri_t(t)	mole fraction of tritium in tritium rich storage	
Xsto_tri_d(t)	mole fraction of deuterium in tritium rich storage	
Xsto_tri_p(t)	mole fraction of protium in tritium rich storage	
Fsto_deu_in(t)	flow into deuterium rich storage (mole/h)	
Fsto_deu_out(t)	flow out of deuterium rich storage (mole/h)	
Nsto_deu_(t)	total amount in internal deuterium rich storage (moles)	
Tsto_deu_(t)	total amount of tritium in internal deuterium rich storage (moles)	
Tsto_deu_m(t)	mobilizable tritium in internal deuterium rich storage (moles)	
Dsto_deu_(t)	total amount of deuterium in internal deuterium rich storage (moles)	
Psto_deu_(t)	total amount of protium in deuterium rich storage (moles)	
Ksto_tri_(t)	accumulated inventory change in storage (mole)	
Ksto_deu_(t)	accumulated inventory change in storage (mole)	
Ksto_pro_(t)	accumulated inventory change in storage (mole)	
Xsto_deu_t(t)	mole fraction of tritium in deuterium rich storage	
Xsto_deu_d(t)	mole fraction of deuterium in deuterium rich storage	
Xsto_deu_p(t)	mole fraction of protium in deuterium rich storage	
Fsto_pro_in(t)	flow into protium rich storage (moles)	
Fsto_pro_out(t)	flow out of protium rich storage (moles)	
Finj_sto_pro(t)	flow from protium rich storage to pellet injector (mole/h)	
Nsto_pro_(t)	total amount in internal protium rich storage (moles)	
Tsto_pro_(t)	total amount of tritium in internal protium rich storage (moles)	
Tsto_pro_m(t)	mobilizable tritium in internal protium rich storage (moles)	
Dsto_pro_(t)	total amount of deuterium in internal protium rich storage (moles)	
Psto_pro_(t)	total amount of protium in internal protium rich storage (moles)	
Xsto_pro_t(t)	mole fraction of tritium in protium rich storage	

Xsto_pro_d(t)	mole fraction of deuterium in protium rich storage	
Xsto_pro_p(t)	mole fraction of protium in protium rich storage	
Fsto_tri_out_t(t)	required flow of tritium out of tritium rich storage (mole/h)	
Fsto_deu_out_d(t)	required flow of deuterium out of deuterium rich storage (mole/h)	
Fsto_pro_out_p(t)	required flow of protium out of protium rich flow (mole/h)	
mu_sto	mobility fraction on storage inventories	1.0D-2
linj_p	fraction of flow from top of isotope separation system directed to pellet injector	3.2D-2

付録 2. ソースリスト

Program Source List

_main.f

```
 1:c
 2:c ****************************************************************
 3:c
 4:c     Fusion reactor tritium inventory evaluation system
 5:c
 6:c     By the Japan Atomic Energy Research Institute, 1997
 7:c
 8:c ****************************************************************
 9:c
10:c----------------------------------------------------------------
11:c $Id: _main.f,v 1.7 1997/05/07 07:28:18 $
12:c----------------------------------------------------------------
13:c
14:c    Main routine
15:c
16:c----------------------------------------------------------------
17:c
18:     program main
19:c
20:     implicit real*8 ( a-h, o-z )
21:c
22:     include '_sizes.h'
23:     include '_commons1.h'
24:c
25:     include '_storage.h'
26:c
27:     include 'varconst.h'
28:c
29:     integer errsw
30:     character*80 line
31:c
32:c----------------------------------------------------------------
33:c  System dependent initialization
34:c  this may includes ;
35:c    * Options (from command line, menus, or files)
36:c    * File name & I/O unit allocation. (if necessary)
37:c    * Menu control (if possible)
38:c    * Setting floating point hardware, exception handling etc.
39:c      (if necessary).
40:c
41:c    etc.
42:c
43:c    Returns system name such as 'Macintosh'
44:c----------------------------------------------------------------
45:c
46:     call INITPROG(sysname)
47:c
48:c----------------------------------------------------------------
49:c  Preprocess input and put lines on unit 50
50:c----------------------------------------------------------------
51:c
52:     open(unit=50,status='SCRATCH',form='FORMATTED')
53:  10 line = ' '
54:     read(5,FMT='(a)',end=20) line
55:c
56:     if( line(1:1).ne.'#' .and. line.ne.' ' ) then
57:        write(50,'(a)') line
58:     end if
59:     goto 10
60:c
61:  20 rewind(50)
62:c
63:c----------------------------------------------------------------
64:c  Input file name specification and options etc.:
65:c----------------------------------------------------------------
66:c
67:     call FILENAMEINPUT
68:c
69:c----------------------------------------------------------------
70:c  Program information echo
71:c----------------------------------------------------------------
72:c
73:     write(6,7000) sysname
74: 7000 format(
75:    & /1x,'================================================',
76:    & /1x,' Fusion reactor tritium inventory evaluation system',
77:    & /1x,'  Version 1.0 on ',a/
78:    & /1x,' 1997 The Japan Atomic Energy Research Institute',
79:    & /1x,'================================================'/)
80:c
81:c ==============================================================
82:c   Initialize variable name table and set their attributes,
83:c   And set default constant values
84:c ==============================================================
85:c
86:     lpr = 1
87:     call definevar( errsw, lpr )
88:c
89:c ==============================================================
90:c   Set default problem control parameters ......
91:c ==============================================================
92:c
93:c   ... maximum calculation time (hour)
94:c
95:     tmax  = 1000d0/3600d0
96:c
97:c   ... number of time steps
98:c
99:     ntimes  = MAXTIMES
100:c
101:c   ... default/current time step width
102:c
103:     delta_t = tmax / ntimes
104:c
105:c ==============================================================
```

```
c     Read problem control information file
c ------------------------------------------------------------
c
      call CONTROLINPUT
c
c ------------------------------------------------------------
c     Solver.
c ------------------------------------------------------------
c
      LIMIT   = MAXSTORAGE
c
      LX0     = 1
      LXJ     = LX0  + nvar
      LX      = LXJ  + nvar
      LXX     = LX  + nvar
      LX1     = LXX   + nvar
      LXW     = LX1   + nvar
      LF      = LXW   + nvar
      LF1     = LF   + ndfvar
      LAST    = LF1   + ndfvar
c
      if( LAST.gt.LIMIT+1 ) then
        write(6,*) '=== data arae requires ',LAST-1,' double words ',
     &       ' but program limit is ',limit
        stop 999
      end if
c
      call SOLVER( tmax, ntimes, times, nsteps,
     &  ntimezone, delta_t, method, sysname,
     &  jdebug,
     &  D(LF)
     & )
c
c ------------------------------------------------------------
c     output results
c ------------------------------------------------------------
c
      call OUTPUT
c
      stop
      end
```

_initprog.f

```
      subroutine initprog( sysname )
c------------------------------------------------------------------------
c      Fusion reactor tritium inventory evaluation system
c------------------------------------------------------------------------
c   system dependent program initialization
c------------------------------------------------------------------------
c $Id: _initprog.f,v 1.3 1997/05/07 02:18:32 $
c------------------------------------------------------------------------
      character*(*) sysname
c
c-----------------------------------------------------------------
c  For Macintosh (Absoft Fortran77/Mac Fortran II)
c-----------------------------------------------------------------
c
      sysname = 'Macintosh'
c
c-----------------------------------------------------------------
c  Open stdin as NULL File name pops up a file name input menu
c-----------------------------------------------------------------
      open(5, file="", status='OLD' )
c
      return
      end
```

defvar.f

```
 1:      subroutine definevar(ierr,lpr)
 2:C ***********************************************
 3:C   Automatically created.
 4:C   creation date : << Tue May 6 14:21:39 1997 >>
 5:C ***********************************************
 6:C=================================================
 7:C ... variables & constants definition in varconst.h
 8:C
 9:C
10:C  varname(*)  :  name of variables.
11:C  vardesc(*)  :  description of variables.
12:C  ivarflag(*)  :  type of variables.
13:C    =0 : does not appear in mode equations.
14:C    =1 : dX/dt is specified for this variable.
15:C    =2 : directly or indirectly refered in dX/dt =...
16:C    =3 : not related to dX/dt =... but refered
17:C         in definition of other variables.
18:C    =4 : not related to dX/dt =... and not refered
19:C         in definition of any variables.
20:C
21:C   Variables whose = 1, 2, 3 needs initial value.
22:C
23:C  constname(*)  :  name of constants.
24:C  constdesc(*)  :  description of constants.
25:C    ( Here "constant" means variables whose values are
26:C     over simulation time. Their values can be changed to
27:C     ppecified values during input phase of program.)
28:C=================================================
29:      implicit double precision (a-h,o-z)
30:      character*64 srcid
31:      integer ierr,lpr
32:C
33:      include '_sizes.h'
34:      include 'varconst.h'
35:
36:C ... statement function ...
37:      ABX(AA,BB,XX) = AA+ BB*XX
38:C ... ierr (>1:error occured ,=0 > no error)
39:      ierr = 1
40:      srcid = 'Tue May 6 14:21:39 1997'
41:      write(6,*) '=== model translation on:',
42:     &          'Tue May 6 14:21:39 1997'
43:C
44:C --- Variable names & descriptions
45:C
46:C
47:      varname(1) = 'Pfusion(t)'
48:      vardesc(1) = 'fusion power (MW)'
49:      ivarflag(1) = 2
50:C
51:      varname(2) = 'Fuelf_t(t)'
52:      vardesc(2) = 'amount of tritium deliberately fueled to the '
53:     &          //'plasma from fuelers (mole/h)'
54:      ivarflag(2) = 2
55:C
56:      varname(3) = 'Fuel_t(t)'
57:      vardesc(3) = 'total tritium fueled to the plasma (mole/h)'
58:      ivarflag(3) = 2
59:C
60:      varname(4) = 'Fuel_d(t)'
61:      vardesc(4) = 'total deuterium fueled to the plasma (mole/h)'
62:      ivarflag(4) = 2
63:C
64:      varname(5) = 'Fuel_p(t)'
65:      vardesc(5) = 'total protium sources to the plasma (mole/h)'
66:      ivarflag(5) = 2
67:C
68:      varname(6) = 'Burn_t(t)'
69:      vardesc(6) = 'tritium burnup rate (moles/h)'
70:      ivarflag(6) = 2
71:C
72:      varname(7) = 'Burn_d(t)'
73:      vardesc(7) = 'deuterium burnup rate (moles/h)'
74:      ivarflag(7) = 2
75:C
76:      varname(8) = 'Reac_p(t)'
77:      vardesc(8) = 'formation rate of protium from advanced fuel '
78:     &          //'reactions (mole/h)'
79:      ivarflag(8) = 2
80:C
81:      varname(9) = 'Efusion(t)'
82:      vardesc(9) = 'amount of fusion energy deliberately caused by '
83:     &          //'one reaction (MeV)'
84:      ivarflag(9) = -1
85:C
86:      varname(10) = 'Fex_t(t)'
87:      vardesc(10) = 'exhaust rate of tritium from plasma (mole/h)'
88:      ivarflag(10) = 2
89:C
90:      varname(11) = 'Fex_d(t)'
91:      vardesc(11) = 'exhaust rate of deuterium from plasma (mole/h)'
92:      ivarflag(11) = 2
93:C
94:      varname(12) = 'Fex_p(t)'
95:      vardesc(12) = 'exhaust rate of protium from plasma (mole/h)'
96:      ivarflag(12) = 2
97:C
98:      varname(13) = 'Fhe_ex(t)'
99:      vardesc(13) = 'exhaust rate of helium from plasma (mole/h)'
100:     ivarflag(13) = 2
101:C
102:     varname(14) = 'i_t(t)'
103:     vardesc(14) = 'isotopic fraction of exhaust tritium from plasma'
104:     ivarflag(14) = 2
105:C
106:     varname(15) = 'i_d(t)'
```

```
107:     vardesc(15) = 'isotopic fraction of exhaust deuterium from '
108:    &          //'plasma'
109:     ivarflag(15) = 2
110:C
111:     varname(16) = 'i_p(t)'
112:     vardesc(16) = 'isotopic fraction of exhaust protium from plasma'
113:     ivarflag(16) = 2
114:C
115:     varname(17) = 'Adiv_t(t)'
116:     vardesc(17) = 'adsorption of tritium from the edge into '
117:    &          //'divertor (moles)'
118:     ivarflag(17) = 2
119:C
120:     varname(18) = 'Afw_t(t)'
121:     vardesc(18) = 'adsorption of tritium from the edge into first '
122:    &          //'wall (moles)'
123:     ivarflag(18) = 2
124:C
125:     varname(19) = 'd_Adiv_t(t)'
126:     vardesc(19) = 'rate of adsorption of tritium from the edge '
127:    &          //'into divertor (mole/h)'
128:     ivarflag(19) = 2
129:C
130:     varname(20) = 'd_Afw_t(t)'
131:     vardesc(20) = 'rate ofadsorption of tritium from the edge into '
132:    &          //'first wall (mole/h)'
133:     ivarflag(20) = 2
134:C
135:     varname(21) = 'C_t(t)'
136:     vardesc(21) = 'tritium codeposited onto plasma facing '
137:    &          //'components (moles)'
138:     ivarflag(21) = 1
139:C
140:     varname(22) = 'd_C_t(t)'
141:     vardesc(22) = 'rate of tritium codeposition onto plasma facing '
142:    &          //'components (mole/h)'
143:     ivarflag(22) = 2
144:C
145:     varname(23) = 'D_t(t)'
146:     vardesc(23) = 'tritium in tokamak dust (moles)'
147:     ivarflag(23) = 1
148:C
149:     varname(24) = 'd_D_t(t)'
150:     vardesc(24) = 'rate of change of tritium in tokamak dust '
151:    &          //'(mole/h)'
152:     ivarflag(24) = 2
153:C
154:     varname(25) = 'Adiv_d(t)'
155:     vardesc(25) = 'adsorption of deuterium from edge into divertor '
156:    &          //'(moles)'
157:     ivarflag(25) = 2
158:C
159:     varname(26) = 'Afw_d(t)'
160:     vardesc(26) = 'adsorption of deuterium from edge into first '
161:    &          //'wall (moles)'
162:     ivarflag(26) = 2
163:C
164:     varname(27) = 'd_Adiv_d(t)'
165:     vardesc(27) = 'rate of adsorption of deuterium from edge into '
166:    &          //'divertor (mole/h)'
167:     ivarflag(27) = 2
168:C
169:     varname(28) = 'd_Afw_d(t)'
170:     vardesc(28) = 'rate of adsorption of deuterium from edge into '
171:    &          //'first wall (mole/h)'
172:     ivarflag(28) = 2
173:C
174:     varname(29) = 'C_d(t)'
175:     vardesc(29) = 'deuterium codeposited onto plasma facing '
176:    &          //'components (moles)'
177:     ivarflag(29) = 1
178:C
179:     varname(30) = 'd_C_d(t)'
180:     vardesc(30) = 'rate of deuterium codeposition onto plasma '
181:    &          //'facing components (mole/h)'
182:     ivarflag(30) = 2
183:C
184:     varname(31) = 'D_d(t)'
185:     vardesc(31) = 'deuterium in tokamak dust (moles)'
186:     ivarflag(31) = 1
187:C
188:     varname(32) = 'd_D_d(t)'
189:     vardesc(32) = 'rate of change of deuterium in tokamak dust '
190:    &          //'(mole/h)'
191:     ivarflag(32) = 2
192:C
193:     varname(33) = 'Adiv_p(t)'
194:     vardesc(33) = 'adsorption of protium from edge into divertor '
195:    &          //'(moles)'
196:     ivarflag(33) = 2
197:C
198:     varname(34) = 'Afw_p(t)'
199:     vardesc(34) = 'adsorption of protium from edge into first wall '
200:    &          //'(moles)'
201:     ivarflag(34) = 2
202:C
203:     varname(35) = 'd_Adiv_p(t)'
204:     vardesc(35) = 'rate of adsorption of protium from edge into '
205:    &          //'divertor (mole/h)'
206:     ivarflag(35) = 2
207:C
208:     varname(36) = 'd_Afw_p(t)'
209:     vardesc(36) = 'rate of adsorption of protium from edge into '
210:    &          //'first wall (mole/h)'
211:     ivarflag(36) = 2
212:C
213:     varname(37) = 'C_p(t)'
214:     vardesc(37) = 'protium codeposited onto plasma facing '
215:    &          //'components (moles)'
216:     ivarflag(37) = 1
217:C
218:     varname(38) = 'd_C_p(t)'
219:     vardesc(38) = 'rate of protium codeposition onto plasma facing '
220:    &          //'components (mole/h)'
```

```
      ivarflag(38) = 2
C
      varname(39) = 'D_p(t)'
      vardesc(39) = 'protium in tokamak dust (moles)'
      ivarflag(39) = 1
C
      varname(40) = 'd_D_p(t)'
      vardesc(40) = 'rate of change of protium in tokamak dust '
     &          //'(mole/h)'
      ivarflag(40) = 2
C
      varname(41) = 'Tpfc_(t)'
      vardesc(41) = 'total tritium in plasma facing componenta (moles)'
      ivarflag(41) = 4
C
      varname(42) = 'Tpfc_m(t)'
      vardesc(42) = 'mobilizable tritium in plasma facing components '
     &          //'(moles)'
      ivarflag(42) = 4
C
      varname(43) = 'Dpfc_(t)'
      vardesc(43) = 'total deuterium in plasma facing components '
     &          //'(moles)'
      ivarflag(43) = 4
C
      varname(44) = 'Ppfc_(t)'
      vardesc(44) = 'total protium in plasma facing components (moles)'
      ivarflag(44) = 4
C
      varname(45) = 'Tpv(t)'
      vardesc(45) = 'tritium inventory per pumps (moles)'
      ivarflag(45) = 1
C
      varname(46) = 'Tp_reg(t)'
      vardesc(46) = 'tritium inventory per pump (batch operation)  '
     &          //'at regeneration time (moles)'
      ivarflag(46) = -1
C
      varname(47) = 'Tp_tot(t)'
      vardesc(47) = 'total tritium inventory on all pumps'
      ivarflag(47) = 3
C
      varname(48) = 'Tp_tot_m(t)'
      vardesc(48) = 'total mobilizable tritium inventory on all '
     &          //'pumps (moles)'
      ivarflag(48) = 4
C
      varname(49) = 'Rt_rem(t)'
      vardesc(49) = 'removal rate of tritium from continuos '
     &          //'regeneration cryopump (mole/h)'
      ivarflag(49) = 2
C
      varname(50) = 'Dpv(t)'
      vardesc(50) = 'Deuterium inventory per pumps (moles).'
      ivarflag(50) = 1
C
      varname(51) = 'Rd_rem(t)'
      vardesc(51) = 'removal rate of deuterium from continuos '
     &          //'regeneration cryopump (mole/h)'
      ivarflag(51) = 2
C
      varname(52) = 'Dp_reg(t)'
      vardesc(52) = 'tritium inventory per pump (batch operation)  '
     &          //'at regeneration time (moles)'
      ivarflag(52) = -1
C
      varname(53) = 'Dp_tot(t)'
      vardesc(53) = 'total deuterium inventory on all pumps (moles).'
      ivarflag(53) = 4
C
      varname(54) = 'Rp_rem(t)'
      vardesc(54) = 'removal rate of protium from continuos '
     &          //'regeneration cryopump (mole/h).'
      ivarflag(54) = 2
C
      varname(55) = 'Pp_reg(t)'
      vardesc(55) = 'protium inventory per pump (batch operation)  '
     &          //'at regeneration time (moles).'
      ivarflag(55) = -1
C
      varname(56) = 'Pp_tot(t)'
      vardesc(56) = 'total protium inventory on all pumps (moles).'
      ivarflag(56) = 4
C
      varname(57) = 'Ppv(t)'
      vardesc(57) = 'protium inventory per pumps (moles).'
      ivarflag(57) = 1
C
      varname(58) = 'Rhe_rem(t)'
      vardesc(58) = 'removal rate of helium from continuos '
     &          //'regeneration cryopump (mole/h)'
      ivarflag(58) = 2
C
      varname(59) = 'Hepv(t)'
      vardesc(59) = 'helium inventory per pumps (moles).'
      ivarflag(59) = 1
C
      varname(60) = 'Hep_reg(t)'
      vardesc(60) = 'helium inventory per pump (batch operation) at '
     &          //'regeneration time (moles).'
      ivarflag(60) = -1
C
      varname(61) = 'Hep_tot(t)'
      vardesc(61) = 'total deuterium inventory on all pumps (moles)'
      ivarflag(61) = 4
C
      varname(62) = 'Fv_out(t)'
      vardesc(62) = 'total flow from vacuum pumps to fuel clean-up '
     &          //'unit (mole/h).'
      ivarflag(62) = 2
C
      varname(63) = 'Fpuri_in(t)'
      vardesc(63) = 'total flow from vacuum pumps to purification '
     &          //'(mole/h)'
```

```
      ivarflag(63) = 4
C
      varname(64) = 'it_(t)'
      vardesc(64) = 'isotopic fraction of tritium on vacuum system '
     &          //'cryopumps'
      ivarflag(64) = 2
C
      varname(65) = 'id_(t)'
      vardesc(65) = 'isotopic fraction of deuterium on vacuum system '
     &          //'cryopumps'
      ivarflag(65) = 2
C
      varname(66) = 'ip_(t)'
      vardesc(66) = 'isotopic fraction of protium on vacuum system '
     &          //'cryopumps'
      ivarflag(66) = 2
C
      varname(67) = 'xnhydro_(t)'
      vardesc(67) = 'fractin of non-hydrogen containing impurities '
     &          //'in adjusted flow to purification'
      ivarflag(67) = 2
C
      varname(68) = 'xh_imp(t)'
      vardesc(68) = 'fraction of hydrogen containing impurities in '
     &          //'adjusted flow to purification'
      ivarflag(68) = 2
C
      varname(69) = 'Fadj_puri(t)'
      vardesc(69) = 'adjusted flow rate to purification considering '
     &          //'impurities (mole/h)'
      ivarflag(69) = 2
C
      varname(70) = 'Fhydro_pure(t)'
      vardesc(70) = 'flow of pure atoms to purification (mole/h)'
      ivarflag(70) = -1
C
      varname(71) = 'Fhydro_imp(t)'
      vardesc(71) = 'flow of hydrogen containing impurities to '
     &          //'purification (mole/h)'
      ivarflag(71) = -1
C
      varname(72) = 'Fnhydro_imp(t)'
      vardesc(72) = 'flow of non-hydrogen containing impurities to '
     &          //'purification (mole/h)'
      ivarflag(72) = 4
C
      varname(73) = 'Ft_pure(t)'
      vardesc(73) = 'flow of pure tritium atoms to purification '
     &          //'(mole/h)'
      ivarflag(73) = 2
C
      varname(74) = 'Fd_pure(t)'
      vardesc(74) = 'flow of pure deuterium atoms to purification '
     &          //'(mole/h)'
      ivarflag(74) = 2
C
      varname(75) = 'Fp_pure(t)'
      vardesc(75) = 'flow of pure protium atoms to purification '
     &          //'(mole/h)'
      ivarflag(75) = 2
C
      varname(76) = 'Fcq4_imp(t)'
      vardesc(76) = 'flow of methane to purification (mole/h)'
      ivarflag(76) = 2
C
      varname(77) = 'Fnq3_imp(t)'
      vardesc(77) = 'flow of ammonia to purification (mole/h)'
      ivarflag(77) = 2
C
      varname(78) = 'Fq2o_imp(t)'
      vardesc(78) = 'flow of water to purification (mole/h)'
      ivarflag(78) = 2
C
      varname(79) = 'Fhydro_imp_w(t)'
      vardesc(79) = 'constituents of hydrogen flow containing '
     &          //'impurities (mole/h)'
      ivarflag(79) = 2
C
      varname(80) = 'f_hydro_imp_w(t)'
      vardesc(80) = 'weighting fraction of constituents of hydrogen '
     &          //'flow containing impurities'
      ivarflag(80) = 2
C
      varname(81) = 'Nimpt(t)'
      vardesc(81) = 'total impurities on purification unit (moles)'
      ivarflag(81) = 1
C
      varname(82) = 'Nhydro_imp(t)'
      vardesc(82) = 'total hydrogen in hydrogen-containing '
     &          //'impurities on purification unit (moles)'
      ivarflag(82) = 2
C
      varname(83) = 'Nnhydro_imp(t)'
      vardesc(83) = 'total non-hydro impurities on purification unit '
     &          //'(moles)'
      ivarflag(83) = 1
C
      varname(84) = 'Ncq4_imp(t)'
      vardesc(84) = 'methane on purification unit (moles)'
      ivarflag(84) = 1
C
      varname(85) = 'Nnq3_imp(t)'
      vardesc(85) = 'ammonia on purification unit (moles)'
      ivarflag(85) = 1
C
      varname(86) = 'Nq2o_imp(t)'
      vardesc(86) = 'water on purification unit (moles)'
      ivarflag(86) = 1
C
      varname(87) = 'Timp_(t)'
      vardesc(87) = 'total tritium on purification unit (moles)'
      ivarflag(87) = 1
C
      varname(88) = 'Dimp_(t)'
```

```
449:      vardesc(88) = 'total deuterium on purification unit (moles)'
450:      ivarflag(88) = 1
451:C
452:      varname(89) = 'Pimp_(t)'
453:      vardesc(89) = 'total protium on purification unit (moles)'
454:      ivarflag(89) = 1
455:C
456:      varname(90) = 'Fh2_puri_in(t)'
457:      vardesc(90) = 'flow of pure hydrogen molecules to purification '
458:     &          //'(mole/h)'
459:      ivarflag(90) = 4
460:C
461:      varname(91) = 'Ft2_sub_in(t)'
462:      vardesc(91) = 'flow of pure tritium molecules to standby unit '
463:     &          //'(mole/h)'
464:      ivarflag(91) = 2
465:C
466:      varname(92) = 'Fd2_sub_in(t)'
467:      vardesc(92) = 'flow of pure deuterium molecules to standby '
468:     &          //'unit (mole/h)'
469:      ivarflag(92) = 2
470:C
471:      varname(93) = 'Fp2_sub_in(t)'
472:      vardesc(93) = 'flow of pure protium molecules to standby unit '
473:     &          //'(mole/h)'
474:      ivarflag(93) = 2
475:C
476:      varname(94) = 'Fh2_sub_in(t)'
477:      vardesc(94) = 'flow of pure fydrogen molecules to standby unit '
478:     &          //'(mole/h)'
479:      ivarflag(94) = 2
480:C
481:      varname(95) = 'Ns(t)'
482:      vardesc(95) = 'number of unfilled adsorption sites for '
483:     &          //'hydrogen on standby uint (moles)'
484:      ivarflag(95) = 1
485:C
486:      varname(96) = 'xt_sub(t)'
487:      vardesc(96) = 'mole fraction of tritium entering standby unit'
488:      ivarflag(96) = 2
489:C
490:      varname(97) = 'xd_sub(t)'
491:      vardesc(97) = 'mole fraction of deuterium entering standby unit'
492:      ivarflag(97) = 2
493:C
494:      varname(98) = 'xp_sub(t)'
495:      vardesc(98) = 'mole fraction of protium entering standby unit'
496:      ivarflag(98) = 2
497:C
498:      varname(99) = 'T2_sub(t)'
499:      vardesc(99) = 'tritium molecules on standby unit (moles)'
500:      ivarflag(99) = 1
501:C
502:      varname(100) = 'D2_sub(t)'
503:      vardesc(100) = 'deuterium molecules on standby unit (moles)'
504:      ivarflag(100) = 1
505:C
506:      varname(101) = 'P2_sub(t)'
507:      vardesc(101) = 'protium molecules on standby unit (moles)'
508:      ivarflag(101) = 1
509:C
510:      varname(102) = 'Nhydro_sub(t)'
511:      vardesc(102) = 'total hydrogen on standby unit (moles)'
512:      ivarflag(102) = 4
513:C
514:      varname(103) = 'Ft_sub_out(t)'
515:      vardesc(103) = 'flow of pure tritium atoms out of standby unit '
516:     &          //'(mole/h)'
517:      ivarflag(103) = 2
518:C
519:      varname(104) = 'Fd_sub_out(t)'
520:      vardesc(104) = 'flow of pure deuterium atoms out of standby '
521:     &          //'unit (mole/h)'
522:      ivarflag(104) = 2
523:C
524:      varname(105) = 'Fp_sub_out(t)'
525:      vardesc(105) = 'flow of pure protium atoms out of standby unit '
526:     &          //'(mole/h)'
527:      ivarflag(105) = 2
528:C
529:      varname(106) = 'Fh_sub_out(t)'
530:      vardesc(106) = 'flow of pure hydrogen atoms out of standby unit '
531:     &          //'(mole/h)'
532:      ivarflag(106) = 4
533:C
534:      varname(107) = 'Treg_out(t)'
535:      vardesc(107) = 'tritium on regenerating molecular sieve beds '
536:     &          //'(moles)'
537:      ivarflag(107) = 3
538:C
539:      varname(108) = 'Ecq4_reg(t)'
540:      vardesc(108) = 'evolution of methane from molecular sieve '
541:     &          //'during regeneration (mole/h)'
542:      ivarflag(108) = 2
543:C
544:      varname(109) = 'Enq3_reg(t)'
545:      vardesc(109) = 'evolution of ammonia from molecular sieve '
546:     &          //'during regeneration (mole/h)'
547:      ivarflag(109) = 2
548:C
549:      varname(110) = 'Eq2o_reg(t)'
550:      vardesc(110) = 'evolution of water from molecular sieve during '
551:     &          //'regeneration (mole/h)'
552:      ivarflag(110) = 2
553:C
554:      varname(111) = 'Ncq4_reg(t)'
555:      vardesc(111) = 'amount of methane on purification bed at '
556:     &          //'regeneration (moles)'
557:      ivarflag(111) = 2
558:C
559:      varname(112) = 'Nnq3_reg(t)'
560:      vardesc(112) = 'amount of ammonia on purification bed at '
561:     &          //'regeneration (moles)'
562:      ivarflag(112) = 2
```

```
563:C
564:      varname(113) = 'Nq2o_reg(t)'
565:      vardesc(113) = 'amount of water on purification bed at '
566:     &          //'regeneration (moles)'
567:      ivarflag(113) = 2
568:C
569:      varname(114) = 'Fhydro_pmem(t)'
570:      vardesc(114) = 'flow of hydrogen to palladium membrane reactor '
571:     &          //'(mole/h)'
572:      ivarflag(114) = 2
573:C
574:      varname(115) = 'Tpmem(t)'
575:      vardesc(115) = 'tritium on palladium membrane reactor (moles)'
576:      ivarflag(115) = 1
577:C
578:      varname(116) = 'Dpmem(t)'
579:      vardesc(116) = 'deuterium on palladium membrane reactor (moles)'
580:      ivarflag(116) = 1
581:C
582:      varname(117) = 'Ppmem(t)'
583:      vardesc(117) = 'protium on palladium membrane reactor (moles)'
584:      ivarflag(117) = 1
585:C
586:      varname(118) = 'it_endj(t)'
587:      vardesc(118) = 'isotopic fraction of tritium on molecular sieve '
588:     &          //'at end of purification'
589:      ivarflag(118) = 2
590:C
591:      varname(119) = 'id_endj(t)'
592:      vardesc(119) = 'isotopic fraction of deuterium on molecular '
593:     &          //'sieve at end of purification'
594:      ivarflag(119) = 2
595:C
596:      varname(120) = 'ip_endj(t)'
597:      vardesc(120) = 'isotopic fraction of protium on molecular sieve '
598:     &          //'at end of purification'
599:      ivarflag(120) = 2
600:C
601:      varname(121) = 'Ft_puri_out(t)'
602:      vardesc(121) = 'flow of tritium to isotope separarion system '
603:     &          //'(mole/h)'
604:      ivarflag(121) = 2
605:C
606:      varname(122) = 'Fd_puri_out(t)'
607:      vardesc(122) = 'flow of deuterium to isotope separation system '
608:     &          //'(mole/h)'
609:      ivarflag(122) = 2
610:C
611:      varname(123) = 'Fp_puri_out(t)'
612:      vardesc(123) = 'flow of protium to isotope separation system '
613:     &          //'(mole/h)'
614:      ivarflag(123) = 2
615:C
616:      varname(124) = 'Fpuri_out(t)'
617:      vardesc(124) = 'total flow to isotope separation system (mole/h)'
618:      ivarflag(124) = 2
619:C
620:      varname(125) = 'Xiss_in_t(t)'
621:      vardesc(125) = 'mole fraction of tritium in stream from fuel '
622:     &          //'clean-up unit to isotope separation system'
623:      ivarflag(125) = 2
624:C
625:      varname(126) = 'Xiss_in_d(t)'
626:      vardesc(126) = 'mole fraction of deuterium in stream from fuel '
627:     &          //'clean-up unit to isotope separation system'
628:      ivarflag(126) = 2
629:C
630:      varname(127) = 'Xiss_in_p(t)'
631:      vardesc(127) = 'mole fraction of protium in stream from fuel '
632:     &          //'clean-up unit to isotope separation system'
633:      ivarflag(127) = 4
634:C
635:      varname(128) = 'Fpure_(t)'
636:      vardesc(128) = 'total flow of pure tritium, deuterium, protium '
637:     &          //'and helium atoms to purification (mole/h)'
638:      ivarflag(128) = 4
639:C
640:      varname(129) = 'Fimp_(t)'
641:      vardesc(129) = 'total flowa of impurities to purification '
642:     &          //'(mole/h)'
643:      ivarflag(129) = 2
644:C
645:      varname(130) = 'Timp_m(t)'
646:      vardesc(130) = 'total mobilizable tritium inventory on '
647:     &          //'purification unit (moles)'
648:      ivarflag(130) = 4
649:C
650:      varname(131) = 'Treg_out_m(t)'
651:      vardesc(131) = 'mobilizable tritium on regenerating molecular '
652:     &          //'sieve beds (moles)'
653:      ivarflag(131) = 4
654:C
655:      varname(132) = 'Fiss_in(t)'
656:      vardesc(132) = 'total flow to isotope separation system (mole/h)'
657:      ivarflag(132) = 2
658:C
659:      varname(133) = 'Fiss_nb(t)'
660:      vardesc(133) = 'flow from neutral beam to isotope separation '
661:     &          //'system (mole/h)'
662:      ivarflag(133) = 4
663:C
664:      varname(134) = 'Fiss_in_t(t)'
665:      vardesc(134) = 'flow of tritium to isotope separation system '
666:     &          //'(mole/h)'
667:      ivarflag(134) = 2
668:C
669:      varname(135) = 'Xiss_t(t)'
670:      vardesc(135) = 'mole fraction of tritium in stream from fuel '
671:     &          //'clean-up unit'
672:      ivarflag(135) = 3
673:C
674:      varname(136) = 'Xnb_t(t)'
675:      vardesc(136) = 'mole fraction of tritium in stream from neutral '
676:     &          //'beam cryopump'
```

```
677:      ivarflag(136) = 4
678:C
679:      varname(137) = 'Xinj_t(t)'
680:      vardesc(137) = 'mole fraction of tritium in pellet injector '
681:     &          //'exhaust stream'
682:      ivarflag(137) = 3
683:C
684:      varname(138) = 'Fiss_in_d(t)'
685:      vardesc(138) = 'flow of deuterium to isotope separation system '
686:     &          //'(mole/h)'
687:      ivarflag(138) = 2
688:C
689:      varname(139) = 'Xiss_d(t)'
690:      vardesc(139) = 'mole fraction of deuterium in stream from fuel '
691:     &          //'clean-up unit'
692:      ivarflag(139) = 3
693:C
694:      varname(140) = 'Xnb_d(t)'
695:      vardesc(140) = 'mole fraction of deuterium in stream from '
696:     &          //'neutral beam cryopump fuel clean-up unit'
697:      ivarflag(140) = 4
698:C
699:      varname(141) = 'Xinj_d(t)'
700:      vardesc(141) = 'mole fraction of deuterium in pellet injector '
701:     &          //'exhaust stream'
702:      ivarflag(141) = 3
703:C
704:      varname(142) = 'Fiss_in_p(t)'
705:      vardesc(142) = 'flow of protium to isotope separation system '
706:     &          //'(mole/h)'
707:      ivarflag(142) = 4
708:C
709:      varname(143) = 'Niss_(t)'
710:      vardesc(143) = 'total amount in isotope separation system (moles)'
711:      ivarflag(143) = 1
712:C
713:      varname(144) = 'Tiss_(t)'
714:      vardesc(144) = 'amount of tritium in isotope separation system '
715:     &          //'(moles)'
716:      ivarflag(144) = 1
717:C
718:      varname(145) = 'Tiss_m(t)'
719:      vardesc(145) = 'amount of mobilizable tritium in isotope '
720:     &          //'saparation system (moles)'
721:      ivarflag(145) = 4
722:C
723:      varname(146) = 'Diss_(t)'
724:      vardesc(146) = 'amount of deuterium in isotope separation '
725:     &          //'system (moles)'
726:      ivarflag(146) = 1
727:C
728:      varname(147) = 'Piss_(t)'
729:      vardesc(147) = 'amount of protium in isotope separation system '
730:     &          //'(moles)'
731:      ivarflag(147) = 4
732:C
733:      varname(148) = 'Xiss_p(t)'
734:      vardesc(148) = 'mole fraction of protium in isotope separation '
735:     &          //'system'
736:      ivarflag(148) = 3
737:C
738:      varname(149) = 'Fiss_top_(t)'
739:      vardesc(149) = 'flow rate from top of isotope separation system '
740:     &          //'(mole/h)'
741:      ivarflag(149) = 2
742:C
743:      varname(150) = 'Fiss_mid_(t)'
744:      vardesc(150) = 'flow rate from middle of isotope separation '
745:     &          //'system (mole/h)'
746:      ivarflag(150) = 2
747:C
748:      varname(151) = 'Fiss_bot_(t)'
749:      vardesc(151) = 'flow rate from bottom of isotope separation '
750:     &          //'system (mole/h)'
751:      ivarflag(151) = 2
752:C
753:      varname(152) = 'Fiss_inj(t)'
754:      vardesc(152) = 'flow directed to isotope separation system from '
755:     &          //'pellet injector (mole/h)'
756:      ivarflag(152) = 4
757:C
758:      varname(153) = 'Fplas_nb_d(t)'
759:      vardesc(153) = 'flow of deuterium to plasma passing neutral '
760:     &          //'beam (mole/h)'
761:      ivarflag(153) = 2
762:C
763:      varname(154) = 'Fin_st_d(t)'
764:      vardesc(154) = 'flow from deuterium rich storage (mole/h)'
765:      ivarflag(154) = 2
766:C
767:      varname(155) = 'Fin_rec_(t)'
768:      vardesc(155) = 'recycle flow rate to neutral beam from neutral '
769:     &          //'beam cryopump (mole/h)'
770:      ivarflag(155) = 4
771:C
772:      varname(156) = 'Fin_iss_mid(t)'
773:      vardesc(156) = 'flow rate from middle of isotope separation '
774:     &          //'system (mole/h)'
775:      ivarflag(156) = -1
776:C
777:      varname(157) = 'Xcryo_d(t)'
778:      vardesc(157) = 'mole fraction of deuterium on neutral beam '
779:     &          //'cryopump'
780:      ivarflag(157) = -1
781:C
782:      varname(158) = 'W_nb(t)'
783:      vardesc(158) = 'required neutral beam power to be delivered to '
784:     &          //'the plasma (MW)'
785:      ivarflag(158) = 2
786:C
787:      varname(159) = 'Ncryo_(t)'
788:      vardesc(159) = 'total amount on neutral beam cryopump (moles)'
789:      ivarflag(159) = 1
790:C
```

```
791:      varname(160) = 'Fbk_nb_(t)'
792:      vardesc(160) = 'total backflow from plasma into neutral beam '
793:     &          //'(mole/h)'
794:      ivarflag(160) = 2
795:C
796:      varname(161) = 'Dcryo_(t)'
797:      vardesc(161) = 'deuterium on neutral beam cryopump (moles)'
798:      ivarflag(161) = -1
799:C
800:      varname(162) = 'Dnb_reg_(t)'
801:      vardesc(162) = 'amount of deuterium on the neutral beam batch '
802:     &          //'regeneration (moles)'
803:      ivarflag(162) = -1
804:C
805:      varname(163) = 'Tnb_cryo(t)'
806:      vardesc(163) = 'tritium on neutral beam cryopump (moles)'
807:      ivarflag(163) = 1
808:C
809:      varname(164) = 'Fbk_nb_t(t)'
810:      vardesc(164) = 'backflow of deuterium from plasma into neutral '
811:     &          //'beam (mole/h)'
812:      ivarflag(164) = 2
813:C
814:      varname(165) = 'Rnb_reg_t(t)'
815:      vardesc(165) = 'regeneration rate of tritium from neutral beam '
816:     &          //'cryopump (mole/h)'
817:      ivarflag(165) = 3
818:C
819:      varname(166) = 'tnb_'
820:      vardesc(166) = 'time during pumping cycle for neutral beam '
821:     &          //'pumps (h)'
822:      ivarflag(166) = -1
823:C
824:      varname(167) = 'Tnb_reg_(t)'
825:      vardesc(167) = 'total amount of tritium on the neutral beam '
826:     &          //'batch regeneration cryopump (moles)'
827:      ivarflag(167) = -1
828:C
829:      varname(168) = 'Tnb_cryo_tot(t)'
830:      vardesc(168) = 'total tritium on neutral beam cryopumps (moles)'
831:      ivarflag(168) = -1
832:C
833:      varname(169) = 'Tnb_cryo_mob(t)'
834:      vardesc(169) = 'total mobilizable tritium on neutral beam '
835:     &          //'cryopump (moles)'
836:      ivarflag(169) = 4
837:C
838:      varname(170) = 'Dnb_cryo(t)'
839:      vardesc(170) = 'deuterium on neutral beam cryopump (moles)'
840:      ivarflag(170) = 1
841:C
842:      varname(171) = 'Fbk_nb_d(t)'
843:      vardesc(171) = 'backflow of deuterium from plasma into neutral '
844:     &          //'beam (mole/h)'
845:      ivarflag(171) = 2
846:C
847:      varname(172) = 'Rnb_reg_d(t)'
848:      vardesc(172) = 'regeneration rate of deuterium from neutral '
849:     &          //'beam cryopump (mole/h)'
850:      ivarflag(172) = 2
851:C
852:      varname(173) = 'Dnb_cryo_tot(t)'
853:      vardesc(173) = 'total deuterium on neutral beam cryopumps (moles)'
854:      ivarflag(173) = -1
855:C
856:      varname(174) = 'Xnb_cryo_t(t)'
857:      vardesc(174) = 'mole fraction of tritium on neutral beam cryopump'
858:      ivarflag(174) = 3
859:C
860:      varname(175) = 'Xnb_cryo_d(t)'
861:      vardesc(175) = 'mole fraction of deuterium on neutral beam '
862:     &          //'cryopump'
863:      ivarflag(175) = 3
864:C
865:      varname(176) = 'Xnb_cryo_p(t)'
866:      vardesc(176) = 'mole fraction of protium on neutral beam cryopump'
867:      ivarflag(176) = 3
868:C
869:      varname(177) = 'Pnb_cryo(t)'
870:      vardesc(177) = 'protium on neutral beam cryopump (moles)'
871:      ivarflag(177) = 3
872:C
873:      varname(178) = 'Tnb_wall(t)'
874:      vardesc(178) = 'tritium on neutral beam walls (moles)'
875:      ivarflag(178) = 1
876:C
877:      varname(179) = 'Tnb_wall_mob(t)'
878:      vardesc(179) = 'mobilizable tritium on neutral beam walls (moles)'
879:      ivarflag(179) = 4
880:C
881:      varname(180) = 'Tnb_tot(t)'
882:      vardesc(180) = 'total tritium in neutral beam (moles)'
883:      ivarflag(180) = -1
884:C
885:      varname(181) = 'Tnb_tot_mob(t)'
886:      vardesc(181) = 'total mobilizable tritium in nueutral beam '
887:     &          //'(moles)'
888:      ivarflag(181) = -1
889:C
890:      varname(182) = 'Fnb_cryo(t)'
891:      vardesc(182) = 'flow rate from neutral beam cryopump (mole/h)'
892:      ivarflag(182) = 4
893:C
894:      varname(183) = 'Crec(t)'
895:      vardesc(183) = 'fraction of gas from neutral beam cryopump '
896:     &          //'directed to the isotope separation system'
897:      ivarflag(183) = 4
898:C
899:      varname(184) = 'Fplas_nb_(t)'
900:      vardesc(184) = 'total flow rate to plasma from neutral beam '
901:     &          //'(mole/h)'
902:      ivarflag(184) = 3
903:C
904:      varname(185) = 'Fplas_nb_t(t)'
```

```
905:      vardesc(185) = 'flow of tritium to plasma through neutral beam '
906:     &          //'(mole/h)'
907:      ivarflag(185) = 2
908:C
909:      varname(186) = 'Fplas_nb_p(t)'
910:      vardesc(186) = 'flow of protium to plasma through neutral beam '
911:     &          //'(mole/h)'
912:      ivarflag(186) = 4
913:C
914:      varname(187) = 'ncycle_p(t)'
915:      vardesc(187) = 'number of plasma cycles'
916:      ivarflag(187) = -1
917:C
918:      varname(188) = 'Nnb_cryo(t)'
919:      vardesc(188) = 'total amount on neutral beam cryopump (moles)'
920:      ivarflag(188) = -1
921:C
922:      varname(189) = 'Rnb_reg_(t)'
923:      vardesc(189) = 'total regeneration rate from neutral beam '
924:     &          //'cryopump (mole/h)'
925:      ivarflag(189) = 4
926:C
927:      varname(190) = 'Rnb_reg_p(t)'
928:      vardesc(190) = 'regeneration rate of protium from neutral beam '
929:     &          //'cryopump (mole/h)'
930:      ivarflag(190) = 3
931:C
932:      varname(191) = 'Fgas_t(t)'
933:      vardesc(191) = 'flow of tritium to plasma through gas puffer '
934:     &          //'(mole/h)'
935:      ivarflag(191) = 2
936:C
937:      varname(192) = 'Fgas_in_tri(t)'
938:      vardesc(192) = 'flow from tritium rich storage to gas puffer '
939:     &          //'(mole/h)'
940:      ivarflag(192) = 2
941:C
942:      varname(193) = 'Fgas_in_deu(t)'
943:      vardesc(193) = 'flow from deuterium rich storage to gas puffer '
944:     &          //'(mole/h)'
945:      ivarflag(193) = 2
946:C
947:      varname(194) = 'Fgas_d(t)'
948:      vardesc(194) = 'flow of deuterium to plasma (mole/h)'
949:      ivarflag(194) = 2
950:C
951:      varname(195) = 'Ngas_tot(t)'
952:      vardesc(195) = 'total amount in gas puffer (moles)'
953:      ivarflag(195) = 1
954:C
955:      varname(196) = 'Fgas_tot(t)'
956:      vardesc(196) = 'total flow rate to plasma from gas puffer '
957:     &          //'(mole/h)'
958:      ivarflag(196) = 2
959:C
960:      varname(197) = 'Tgas_(t)'
961:      vardesc(197) = 'amount of tritium in gas puffer (moles)'
962:      ivarflag(197) = 1
963:C
964:      varname(198) = 'Dgas_(t)'
965:      vardesc(198) = 'amount of deuterium in gas puffer (moles)'
966:      ivarflag(198) = 1
967:C
968:      varname(199) = 'Xgas_t(t)'
969:      vardesc(199) = 'mole fraction of tritium in gas puffer exit '
970:     &          //'stream'
971:      ivarflag(199) = 2
972:C
973:      varname(200) = 'Xgas_d(t)'
974:      vardesc(200) = 'mole fraction of deuterium in gas puffer exit '
975:     &          //'stream'
976:      ivarflag(200) = 2
977:C
978:      varname(201) = 'Xgas_p(t)'
979:      vardesc(201) = 'mole fraction of protium in gas puffer exit '
980:     &          //'stream'
981:      ivarflag(201) = 3
982:C
983:      varname(202) = 'Pgas_(t)'
984:      vardesc(202) = 'amount of protium in gas puffer (moles)'
985:      ivarflag(202) = 4
986:C
987:      varname(203) = 'Fpuff_plas(t)'
988:      vardesc(203) = 'total flow to plasma from gas puffer and pellet '
989:     &          //'injector propellant (mole/h)'
990:      ivarflag(203) = 4
991:C
992:      varname(204) = 'Tgas_mob(t)'
993:      vardesc(204) = 'mobilizable tritium in gas puffer (moles)'
994:      ivarflag(204) = 4
995:C
996:      varname(205) = 'Fplas_in_t(t)'
997:      vardesc(205) = 'flow of tritium to plasma through pellet '
998:     &          //'injector (mole/h)'
999:      ivarflag(205) = 2
1000:C
1001:      varname(206) = 'Fplas_in_d(t)'
1002:      vardesc(206) = 'flow of deuterium to plasma through pellet '
1003:     &          //'injector (mole/h)'
1004:      ivarflag(206) = 2
1005:C
1006:      varname(207) = 'Fplas_in_p(t)'
1007:      vardesc(207) = 'flow of protium to plasma through pellet '
1008:     &          //'injector (mole/h)'
1009:      ivarflag(207) = 2
1010:C
1011:      varname(208) = 'Fsto_out_t(t)'
1012:      vardesc(208) = 'flow from tritium rich storage to pellet '
1013:     &          //'injector (mole/h)'
1014:      ivarflag(208) = 2
1015:C
1016:      varname(209) = 'Fsto_out_d(t)'
1017:      vardesc(209) = 'flow from deuterium rich storage to pellet '
1018:     &          //'injector (mole/h)'
```

```
1019:     ivarflag(209) = 2
1020:C
1021:     varname(210) = 'Fsto_out_p(t)'
1022:     vardesc(210) = 'flow from protium rich storage to pellet '
1023:    &          //'injector (mole/h)'
1024:     ivarflag(210) = 2
1025:C
1026:     varname(211) = 'Next_12(t)'
1027:     vardesc(211) = 'total moles (of atoms) in pellet injector '
1028:    &          //'extruder 1 or 2 (moles)'
1029:     ivarflag(211) = 1
1030:C
1031:     varname(212) = 'Text_12(t)'
1032:     vardesc(212) = 'tritium in pellet injector extruder 1 or 2 '
1033:    &          //'(moles)'
1034:     ivarflag(212) = 1
1035:C
1036:     varname(213) = 'Dext_12(t)'
1037:     vardesc(213) = 'deuterium in pellet injector extruder 1 or 2 '
1038:    &          //'(moles)'
1039:     ivarflag(213) = 1
1040:C
1041:     varname(214) = 'Pext_12(t)'
1042:     vardesc(214) = 'protium in pellet injector extruder 1 or 2 '
1043:    &          //'(moles)'
1044:     ivarflag(214) = 3
1045:C
1046:     varname(215) = 'Xext_12_t(t)'
1047:     vardesc(215) = 'mole fraction of tritium in pellet injector '
1048:    &          //'extruder 1 or 2'
1049:     ivarflag(215) = 3
1050:C
1051:     varname(216) = 'Xext_12_d(t)'
1052:     vardesc(216) = 'mole fraction of deuterium in pellet injector '
1053:    &          //'extruder 1 or 2'
1054:     ivarflag(216) = 3
1055:C
1056:     varname(217) = 'Xext_12_p(t)'
1057:     vardesc(217) = 'mole fraction of protium in pellet injector '
1058:    &          //'extruder 1 or 2'
1059:     ivarflag(217) = 3
1060:C
1061:     varname(218) = 'Next_tot(t)'
1062:     vardesc(218) = 'total in pellet injector extruders (moles)'
1063:     ivarflag(218) = 4
1064:C
1065:     varname(219) = 'Text_tot(t)'
1066:     vardesc(219) = 'total tritium in pellet injector extruders '
1067:    &          //'(moles)'
1068:     ivarflag(219) = 3
1069:C
1070:     varname(220) = 'Dext_tot(t)'
1071:     vardesc(220) = 'total deuterium in pellet injector extruders '
1072:    &          //'(moles)'
1073:     ivarflag(220) = 4
1074:C
1075:     varname(221) = 'Pext_tot(t)'
1076:     vardesc(221) = 'total protium in pellet injector extruders '
1077:    &          //'(moles)'
1078:     ivarflag(221) = 4
1079:C
1080:     varname(222) = 'Text_m(t)'
1081:     vardesc(222) = 'total mobilizable tritium in pellet injectors '
1082:    &          //'(moles)'
1083:     ivarflag(222) = 4
1084:C
1085:     varname(223) = 'Vinj_t(t)'
1086:     vardesc(223) = 'volume of tritium in pellet injector (m3)'
1087:     ivarflag(223) = 2
1088:C
1089:     varname(224) = 'Vinj_d(t)'
1090:     vardesc(224) = 'volume of deuterium in pellet injector (m3)'
1091:     ivarflag(224) = 2
1092:C
1093:     varname(225) = 'Vinj_p(t)'
1094:     vardesc(225) = 'volume of protium in pellet injector (m3)'
1095:     ivarflag(225) = 2
1096:C
1097:     varname(226) = 'Xextru_t(t)'
1098:     vardesc(226) = 'mole fraction of tritium in pellet injector '
1099:    &          //'extruder'
1100:     ivarflag(226) = 2
1101:C
1102:     varname(227) = 'Xextru_d(t)'
1103:     vardesc(227) = 'mole fraction of deuterium in pellet injector '
1104:    &          //'extruder'
1105:     ivarflag(227) = 2
1106:C
1107:     varname(228) = 'Xextru_p(t)'
1108:     vardesc(228) = 'mole fraction of protium in pellet injector '
1109:    &          //'extruder'
1110:     ivarflag(228) = 2
1111:C
1112:     varname(229) = 'Nextru_(t)'
1113:     vardesc(229) = 'total amount in pellet injector extruder'
1114:     ivarflag(229) = 2
1115:C
1116:     varname(230) = 'Vinj_tot(t)'
1117:     vardesc(230) = 'total volume in pellet injector extruder'
1118:     ivarflag(230) = 2
1119:C
1120:     varname(231) = 'Dpellet(t)'
1121:     vardesc(231) = 'diameter of pellet injector pellet (m)'
1122:     ivarflag(231) = -1
1123:C
1124:     varname(232) = 'Hpellet(t)'
1125:     vardesc(232) = 'height of pellet injector pellet (m)'
1126:     ivarflag(232) = -1
1127:C
1128:     varname(233) = 'npellet_t(t)'
1129:     vardesc(233) = 'number of moles of tritium in pellet injector '
1130:    &          //'pellet (moles)'
1131:     ivarflag(233) = 3
1132:C
```

```
1133:     varname(234) = 'npellet_d(t)'
1134:     vardesc(234) = 'number of moles of deuterium in pellet injector '
1135:    &          //'pellet (moles)'
1136:     ivarflag(234) = 4
1137:C
1138:     varname(235) = 'npellet_p(t)'
1139:     vardesc(235) = 'number of moles of protium in pellet injector '
1140:    &          //'pellet (moles)'
1141:     ivarflag(235) = 2
1142:C
1143:     varname(236) = 'Rinj_(t)'
1144:     vardesc(236) = 'pellet repetition rate for pellet injector (Hz)'
1145:     ivarflag(236) = 4
1146:C
1147:     varname(237) = 'Finj_in_tot(t)'
1148:     vardesc(237) = 'total flow rate to plasma by pellet injector '
1149:    &          //'pellet (mole/h)'
1150:     ivarflag(237) = 4
1151:C
1152:     varname(238) = 'Fbk_plas_(t)'
1153:     vardesc(238) = 'backflow from plasma into pellet injector '
1154:    &          //'(mole/h)'
1155:     ivarflag(238) = 3
1156:C
1157:     varname(239) = 'Fbk_plas_t(t)'
1158:     vardesc(239) = 'tritium backflow from plasma into pellet '
1159:    &          //'injector (mole/h)'
1160:     ivarflag(239) = 3
1161:C
1162:     varname(240) = 'Fbk_plas_d(t)'
1163:     vardesc(240) = 'deuterium backflow from plasma into pellet '
1164:    &          //'injector (mole/h)'
1165:     ivarflag(240) = 3
1166:C
1167:     varname(241) = 'Fbk_plas_p(t)'
1168:     vardesc(241) = 'protium backflow from plasma into pellet '
1169:    &          //'injector (mole/h)'
1170:     ivarflag(241) = 4
1171:C
1172:     varname(242) = 'Fplas_prop(t)'
1173:     vardesc(242) = 'total flow rate to plasma from pellet injector '
1174:    &          //'propellant (mole/h)'
1175:     ivarflag(242) = 2
1176:C
1177:     varname(243) = 'Fplas_prop_t(t)'
1178:     vardesc(243) = 'tritium flow rate to plasma from pellet '
1179:    &          //'injector propellant (mole/h)'
1180:     ivarflag(243) = 2
1181:C
1182:     varname(244) = 'Fplas_prop_d(t)'
1183:     vardesc(244) = 'deuterium flow rate to plasma from pellet '
1184:    &          //'injector propellant (mole/h)'
1185:     ivarflag(244) = 3
1186:C
1187:     varname(245) = 'Fplas_prop_p(t)'
1188:     vardesc(245) = 'protium flow rate to plasma from pellet '
1189:    &          //'injector propellant (mole/h)'
1190:     ivarflag(245) = 3
1191:C
1192:     varname(246) = 'Finj_exh(t)'
1193:     vardesc(246) = 'exhaust flow rate from pellet injector (mole/h)'
1194:     ivarflag(246) = 3
1195:C
1196:     varname(247) = 'Finj_exh_t(t)'
1197:     vardesc(247) = 'tritium exhaust flow rate from pellet injector '
1198:    &          //'(mole/h)'
1199:     ivarflag(247) = 3
1200:C
1201:     varname(248) = 'Finj_exh_d(t)'
1202:     vardesc(248) = 'deuterium exhaust flow rate from pellet '
1203:    &          //'injector (mole/h)'
1204:     ivarflag(248) = 3
1205:C
1206:     varname(249) = 'Xinj_p(t)'
1207:     vardesc(249) = 'mole fraction of protium in pellet injector'
1208:     ivarflag(249) = 4
1209:C
1210:     varname(250) = 'Finj_prop(t)'
1211:     vardesc(250) = 'flow of propellant gas (mole/h)'
1212:     ivarflag(250) = 2
1213:C
1214:     varname(251) = 'Xprop_t(t)'
1215:     vardesc(251) = 'mole fraction of tritium in propellant gas flow '
1216:    &          //'(mole/h)'
1217:     ivarflag(251) = 2
1218:C
1219:     varname(252) = 'Xprop_d(t)'
1220:     vardesc(252) = 'mole fraction of deuterium in propellant gas '
1221:    &          //'flow (mole/h)'
1222:     ivarflag(252) = 3
1223:C
1224:     varname(253) = 'Xprop_p(t)'
1225:     vardesc(253) = 'mole fraction of protium in propellant gas flow '
1226:    &          //'(mole/h)'
1227:     ivarflag(253) = 3
1228:C
1229:     varname(254) = 'Finj_rec(t)'
1230:     vardesc(254) = 'recycle flow rate to sotope separation system '
1231:    &          //'from pellet injector (mole/h)'
1232:     ivarflag(254) = 4
1233:C
1234:     varname(255) = 'Fsto_tri_in(t)'
1235:     vardesc(255) = 'flow into tritium rich storage (mole/h)'
1236:     ivarflag(255) = 2
1237:C
1238:     varname(256) = 'Fsto_tri_out(t)'
1239:     vardesc(256) = 'flow out of tritium rich storage (mole/h)'
1240:     ivarflag(256) = 2
1241:C
1242:     varname(257) = 'Nsto_tri_(t)'
1243:     vardesc(257) = 'total amount in internal tritium rich storage '
1244:    &          //'(moles)'
1245:     ivarflag(257) = 1
1246:C
```

```
      varname(258) = 'Tsto_tri_(t)'
      vardesc(258) = 'total amount of tritium in internal tritium '
     &          //'rich storage (moles)'
      ivarflag(258) = 1
C
      varname(259) = 'Tsto_tri_m(t)'
      vardesc(259) = '"mobilizable tritium in internal deuterium rich '
     &          //'storage (moles)'
      ivarflag(259) = 4
C
      varname(260) = 'Dsto_tri_(t)'
      vardesc(260) = 'total amount of deuterium in internal tritium '
     &          //'rich storage (moles)'
      ivarflag(260) = 1
C
      varname(261) = 'Psto_tri_(t)'
      vardesc(261) = 'total amount of protium in internal tritium '
     &          //'rich storage (moles)'
      ivarflag(261) = 4
C
      varname(262) = 'Xsto_tri_t(t)'
      vardesc(262) = 'mole fraction of tritium in tritium rich storage'
      ivarflag(262) = 2
C
      varname(263) = 'Xsto_tri_d(t)'
      vardesc(263) = 'mole fraction of deuterium in tritium rich '
     &          //'storage'
      ivarflag(263) = 2
C
      varname(264) = 'Xsto_tri_p(t)'
      vardesc(264) = 'mole fraction of protium in tritium rich storage'
      ivarflag(264) = 3
C
      varname(265) = 'Fsto_deu_in(t)'
      vardesc(265) = 'flow into deuterium rich storage (mole/h)'
      ivarflag(265) = 2
C
      varname(266) = 'Fsto_deu_out(t)'
      vardesc(266) = 'flow out of deuterium rich storage (mole/h)'
      ivarflag(266) = 2
C
      varname(267) = 'Nsto_deu_(t)'
      vardesc(267) = 'total amount in internal deuterium rich storage '
     &          //'(moles)'
      ivarflag(267) = 1
C
      varname(268) = 'Tsto_deu_(t)'
      vardesc(268) = 'total amount of tritium in internal deuterium '
     &          //'rich storage (moles)'
      ivarflag(268) = 1
C
      varname(269) = 'Tsto_deu_m(t)'
      vardesc(269) = 'mobilizable tritium in internal deuterium rich '
     &          //'storage (moles)'
      ivarflag(269) = 4
C
      varname(270) = 'Dsto_deu_(t)'
      vardesc(270) = 'total amount of deuterium in internal deuterium '
     &          //'rich storage (moles)'
      ivarflag(270) = 1
C
      varname(271) = 'Psto_deu_(t)'
      vardesc(271) = 'total amount of protium in deuterium rich '
     &          //'storage (moles)'
      ivarflag(271) = 4
C
      varname(272) = 'Ksto_tri_(t)'
      vardesc(272) = 'accumulated inventory change in storage (mole)'
      ivarflag(272) = 3
C
      varname(273) = 'Ksto_deu_(t)'
      vardesc(273) = 'accumulated inventory change in storage (mole)'
      ivarflag(273) = 3
C
      varname(274) = 'Ksto_pro_(t)'
      vardesc(274) = 'accumulated inventory change in storage (mole)'
      ivarflag(274) = 3
C
      varname(275) = 'Xsto_deu_t(t)'
      vardesc(275) = 'mole fraction of tritium in deuterium rich '
     &          //'storage'
      ivarflag(275) = 2
C
      varname(276) = 'Xsto_deu_d(t)'
      vardesc(276) = 'mole fraction of deuterium in deuterium rich '
     &          //'storage'
      ivarflag(276) = 2
C
      varname(277) = 'Xsto_deu_p(t)'
      vardesc(277) = 'mole fraction of protium in deuterium rich '
     &          //'storage'
      ivarflag(277) = 3
C
      varname(278) = 'Fsto_pro_in(t)'
      vardesc(278) = 'flow into protium rich storage (moles)'
      ivarflag(278) = 2
C
      varname(279) = 'Fsto_pro_out(t)'
      vardesc(279) = 'flow out of protium rich storage (moles)'
      ivarflag(279) = 2
C
      varname(280) = 'Finj_sto_pro(t)'
      vardesc(280) = 'flow from protium rich storage to pellet '
     &          //'injector (mole/h)'
      ivarflag(280) = -1
C
      varname(281) = 'Nsto_pro_(t)'
      vardesc(281) = 'total amount in internal protium rich storage '
     &          //'(moles)'
      ivarflag(281) = 1
C
      varname(282) = 'Tsto_pro_(t)'
      vardesc(282) = 'total amount of tritium in internal protium '
     &          //'rich storage (moles)'
```

```
      ivarflag(282) = 1
C
      varname(283) = 'Tsto_pro_m(t)'
      vardesc(283) = 'mobilizable tritium in internal protium rich '
     &          //'storage (moles)'
      ivarflag(283) = 4
C
      varname(284) = 'Dsto_pro_(t)'
      vardesc(284) = 'total amount of deuterium in internal protium '
     &          //'rich storage (moles)'
      ivarflag(284) = 1
C
      varname(285) = 'Psto_pro_(t)'
      vardesc(285) = 'total amount of protium in internal protium '
     &          //'rich storage (moles)'
      ivarflag(285) = 3
C
      varname(286) = 'Xsto_pro_t(t)'
      vardesc(286) = 'mole fraction of tritium in protium rich storage'
      ivarflag(286) = 3
C
      varname(287) = 'Xsto_pro_d(t)'
      vardesc(287) = 'mole fraction of deuterium in protium rich '
     &          //'storage'
      ivarflag(287) = 3
C
      varname(288) = 'Xsto_pro_p(t)'
      vardesc(288) = 'mole fraction of protium in protium rich storage'
      ivarflag(288) = 3
C
      varname(289) = 'Fsto_tri_out_t(t)'
      vardesc(289) = 'required flow of tritium out of tritium rich '
     &          //'storage (mole/h)'
      ivarflag(289) = 4
C
      varname(290) = 'Fsto_deu_out_d(t)'
      vardesc(290) = 'required flow of deuterium out of deuterium '
     &          //'rich storage (mole/h)'
      ivarflag(290) = 4
C
      varname(291) = 'Fsto_pro_out_p(t)'
      vardesc(291) = 'required flow of protium outof protium rich '
     &          //'flow (mole/h)'
      ivarflag(291) = 4
C
C --- constants & descriptions
C
C
      constname(1) = 'ap'
      ap = 1.0D-2
      constdesc(1) = 'relative combined reaction rates of proton '
     &        // 'branch of DD reaction and D3H reaction'
     &        // 'compared to DT'
C
      constname(2) = 'fbu'
      fbu = 5.0D-2
      constdesc(2) = 'fractional burn-up of tritium'
C
      constname(3) = 'fin_pellet'
      fin_pellet = 5.0D-1
      constdesc(3) = 'fraction of tritium fueling through pellet '
     &          //'injection'
C
      constname(4) = 'fin_gas'
      fin_gas = 7.0D-1
      constdesc(4) = 'fraction of tritium fieling through gas puffing'
C
      constname(5) = 'Temperature'
      Temperature = 1000
      constdesc(5) = 'Diverter temperature (C degree)'
C
      constname(6) = 'Tedge'
      Tedge = 60
      constdesc(6) = 'Plasma Facing Material temperature (eV)'
C
      constname(7) = 'Medge'
      Medge = 1
      constdesc(7) = 'Plasma Facing Material (1/2/3=C/Be/W)'
C
      constname(8) = 'mu_A'
      mu_A = 1.0D-1
      constdesc(8) = 'mobility fraction of absorbed inventory'
C
      constname(9) = 'mu_C'
      mu_C = 1.0D0
      constdesc(9) = 'mobility fraction of codeposited inventory'
C
      constname(10) = 'mu_D'
      mu_D = 1.0D0
      constdesc(10) = 'mobility fraction of dust inventory'
C
      constname(11) = 'dc'
      dc = 1.0D-1
      constdesc(11) = 'dust concentration ratio (kg_tritium/kg_dust)'
C
      constname(12) = 'Vdiv'
      Vdiv = (2.0D-5)*(2.0D5)
      constdesc(12) = 'total volume of divertor (m3)'
C
      constname(13) = 'Vfw'
      Vfw = (1.0D-3)*(1.6D3)*(2.0D-4)*(7.0D4)
      constdesc(13) = 'total volume of first wall (m3)'
C
      constname(14) = 'Ad'
      Ad = (1.0D-3)*(2.0D5)
      constdesc(14) = 'divertor wall surface area (m2)'
C
      constname(15) = 'lambda_T'
      lambda_T = 0.693/(12.4*365*24)
      constdesc(15) = 'Decay constant of Tritium (1/h)'
C
      constname(16) = 'npump'
      npump = 16
      constdesc(16) = 'number of operating pumps'
```

```
C
      constname(17) = 'treg'
      treg = 2.0D0/3.0D0
      constdesc(17) = 'regeneration time for batch cryopumps in vacuum '
     &          //'system (h)'
C
      constname(18) = 'tpump'
      tpump = 4.0D0/3.0D0
      constdesc(18) = 'pumping for batch cryopumps in vacuum system (h)'
C
      constname(19) = 'emr'
      emr = 2.0D-1
      constdesc(19) = 'efficiency of removal of cryodeposit for '
     &          //'mechanical continuous regeneration cryopump'
C
      constname(20) = 'Rs'
      Rs = 1.92D0
      constdesc(20) = 'scraping rate for mechanical continuous '
     &          //'regeneration scheme (m2/h)'
C
      constname(21) = 'Area_p'
      Area_p = 3.2D-2
      constdesc(21) = 'pump surface area for mechanical continuous '
     &          //'regeneration scheme (m2).'
C
      constname(22) = 'mu_v'
      mu_v = 1.0D-1
      constdesc(22) = 'mobility fraction of inventory on cryopump.'
C
      constname(23) = 'mu_mob'
      mu_mob = 1.0D-2
      constdesc(23) = 'mobility fraction of inventory on fuel clean-up '
     &          //'unit molecular sieve bed'
C
      constname(24) = 'Lhydro_'
      Lhydro_ = 1.2D2
      constdesc(24) = 'loading of mole sieves for hydrogen (scc/g)'
C
      constname(25) = 'Limp_'
      Limp_ = 1.0D2
      constdesc(25) = 'loading of mole sieves for impurities (scc/g)'
C
      constname(26) = 'Ns_max'
      Ns_max = 1.77D1
      constdesc(26) = 'maximum number of unfilled adsorption sites for '
     &        // 'hydrogen on fuel clean-up unit'
     &        // 'standby bed preloading with hydrogen (mole)'
C
      constname(27) = 'treg_msieve'
      treg_msieve = 4.0D0
      constdesc(27) = 'regeneration time for molecular sieve (h)'
C
      constname(28) = 'mu_reg_m'
      mu_reg_m = 1.0D0
      constdesc(28) = 'mobility fraction of inventory on fuel clean-up '
     &          //'unit molecular sieve beds during regeneration'
C
      constname(29) = 'tau_pmem'
      tau_pmem = 2.0D0
      constdesc(29) = 'time constant for palladium membrane reactor (h)'
C
      constname(30) = 'w1'
      w1 = 4.0D0
      constdesc(30) = 'number of hydrogen atoms in a methane molecule'
C
      constname(31) = 'w2'
      w2 = 3.0D0
      constdesc(31) = 'number of hydrogen atoms in an ammonia molecule'
C
      constname(32) = 'w3'
      w3 = 2.0D0
      constdesc(32) = 'number of hydrogen atoms in a water molecule'
C
      constname(33) = 'xcq4_'
      xcq4_ = 5.6D-1
      constdesc(33) = 'specified mole fraction of methane in adjusted '
     &          //'flow to purification'
C
      constname(34) = 'xnq3_'
      xnq3_ = 4.0D-2
      constdesc(34) = 'specified mole fraction of ammonia in adjusted '
     &          //'flow to purification'
C
      constname(35) = 'xq2o_'
      xq2o_ = 8.0D-2
      constdesc(35) = 'specified mole fraction of water in adjusted '
     &          //'flow to purification'
C
      constname(36) = 'Fbl_cool'
      Fbl_cool = 1.0D-2
      constdesc(36) = 'flow rate from coolant system (mole/h)'
C
      constname(37) = 'Fbl_'
      Fbl_ = 5.0D-2
      constdesc(37) = 'flow rate from blanket system (mole/h)'
C
      constname(38) = 'h1'
      h1 = 3.0D-1
      constdesc(38) = 'fraction of stream from fuel clean-up system '
     &          //'not processed and directed to gas puffer'
C
      constname(39) = 'h3'
      h3 = 3.0D-1
      constdesc(39) = 'fraction of stream from fuel clean-up system '
     &          //'not processed and directed to pellet injector'
C
      constname(40) = 'Fragnb'
      Fragnb = 0.0D0
      constdesc(40) = 'switch to include neutral beam in isotope '
     &        // 'separation system balance (included if'
     &        // 'equals one, not included if equals zero)'
C
      constname(41) = 'Fragiss_inj'
      Fragiss_inj = 0.0D0
```

```
1589:     constdesc(41) = 'switch to include pellet injector in isotope '
1590:    &        // 'separation system balance (included if'
1591:    &        // 'equals one, not included if equals zero)'
1592:C
1593:     constname(42) = 'Xc_t'
1594:     Xc_t = 3.0D-1
1595:     constdesc(42) = 'mole fraction of tritium in coolant system'
1596:C
1597:     constname(43) = 'Xbl_t'
1598:     Xbl_t = 3.0D-1
1599:     constdesc(43) = 'mole fraction of tritium in breeder system'
1600:C
1601:     constname(44) = 'Xc_d'
1602:     Xc_d = 3.0D-1
1603:     constdesc(44) = 'mole fraction of deuterium in coolant stream'
1604:C
1605:     constname(45) = 'Xbl_d'
1606:     Xbl_d = 3.0D-1
1607:     constdesc(45) = 'mole fraction of deuterium in breeder stream'
1608:C
1609:     constname(46) = 'tau_iss_'
1610:     tau_iss_ = 2.22D0
1611:     constdesc(46) = 'residence time in isotope separation system (h)'
1612:C
1613:     constname(47) = 'mu_iss_'
1614:     mu_iss_ = 1.0D0
1615:     constdesc(47) = 'mobility fraction of inventory in isotope '
1616:    &          //'separation system'
1617:C
1618:     constname(48) = 'Xiss_top_t'
1619:     Xiss_top_t = 6.0D0/9.0D0
1620:     constdesc(48) = 'mole fraction of tritium from top of isotope '
1621:    &          //'separation system'
1622:C
1623:     constname(49) = 'Xiss_top_d'
1624:     Xiss_top_d = 2.0D0/9.0D0
1625:     constdesc(49) = 'mole fraction of deuterium from top of isotope '
1626:    &          //'separation system'
1627:C
1628:     constname(50) = 'Xiss_top_p'
1629:     Xiss_top_p = 1.0D0/9.0D0
1630:     constdesc(50) = 'mole fraction of tritium from top of isotope '
1631:    &          //'separation system'
1632:C
1633:     constname(51) = 'Xiss_mid_t'
1634:     Xiss_mid_t = 2.0D0/9.0D0
1635:     constdesc(51) = 'mole fraction of tritium from middle of isotope '
1636:    &          //'separation system'
1637:C
1638:     constname(52) = 'Xiss_mid_d'
1639:     Xiss_mid_d = 6.0D0/9.0D0
1640:     constdesc(52) = 'mole fraction of deuterium from middle of '
1641:    &          //'isotope separation system'
1642:C
1643:     constname(53) = 'Xiss_mid_p'
1644:     Xiss_mid_p = 1.0D0/9.0D0
1645:     constdesc(53) = 'mole fraction of tritium from middle of isotope '
1646:    &          //'separation system'
1647:C
1648:     constname(54) = 'Xiss_bot_t'
1649:     Xiss_bot_t = 1.0D0/9.0D0
1650:     constdesc(54) = 'mole fraction of tritium from bottom of isotope '
1651:    &          //'separation system'
1652:C
1653:     constname(55) = 'Xiss_bot_d'
1654:     Xiss_bot_d = 2.0D0/9.0D0
1655:     constdesc(55) = 'mole fraction of deuterium from bottom of '
1656:    &          //'isotope separation system'
1657:C
1658:     constname(56) = 'Xiss_bot_p'
1659:     Xiss_bot_p = 6.0D0/9.0D0
1660:     constdesc(56) = 'mole fraction of tritium from bottom of isotope '
1661:    &          //'separation system'
1662:C
1663:     constname(57) = 'eta_nb_tot'
1664:     eta_nb_tot = 3.1D-1
1665:     constdesc(57) = 'overall efficiency of transfer of neutral beam '
1666:    &          //'feed to the plasma'
1667:C
1668:     constname(58) = 'kiss_mid_'
1669:     kiss_mid_ = 3.0D-1
1670:     constdesc(58) = 'faction of flow from middle of isotope '
1671:    &          //'separation system directed to the neutral beam'
1672:C
1673:     constname(59) = 'fcryo_'
1674:     fcryo_ = 3.7D-1
1675:     constdesc(59) = 'fraction of feed to neutral beam that flows to '
1676:    &          //'cryopomp'
1677:C
1678:     constname(60) = 'tnb_reg'
1679:     tnb_reg = 2.0D0/3.0D0
1680:     constdesc(60) = 'regeneration time for batch neutral beam '
1681:    &          //'cryopumps (h)'
1682:C
1683:     constname(61) = 'tnb_pump'
1684:     tnb_pump = 2.0D0
1685:     constdesc(61) = 'pumping time for neutral beam batch '
1686:    &          //'regeneration cyopump (h)'
1687:C
1688:     constname(62) = 'nnb_'
1689:     nnb_ = 5.0D0
1690:     constdesc(62) = 'pumping cycle'
1691:C
1692:     constname(63) = 'mu_nb_mob'
1693:     mu_nb_mob = 1.0D0
1694:     constdesc(63) = 'mobility fraction of inventory on neutral beam '
1695:    &          //'cryopump'
1696:C
1697:     constname(64) = 'fnb_wall'
1698:     fnb_wall = 5.0D-2
1699:     constdesc(64) = 'fraction of feed to neutral beam that ends up '
1700:    &          //'on the walls'
1701:C
1702:     constname(65) = 'mu_wall_'
```

```
1703:     mu_wall_ = 1.0D0
1704:     constdesc(65) = 'mobility fraction of inventory in neutral beam '
1705:    &          //'walls'
1706:C
1707:     constname(66) = 'tcycle_p'
1708:     tcycle_p = 1000d0/3600d0
1709:     constdesc(66) = 'plasma burning time (h)'
1710:C
1711:     constname(67) = 't_p_ru'
1712:     t_p_ru = 200d0/3600d0
1713:     constdesc(67) = 'ramp-up time for plasma (h)'
1714:C
1715:     constname(68) = 't_p_rd'
1716:     t_p_rd = 200d0/3600d0
1717:     constdesc(68) = 'ramp-down time for plasma (h)'
1718:C
1719:     constname(69) = 't_nb_ru'
1720:     t_nb_ru = 100d0/3600d0
1721:     constdesc(69) = 'ramp-up time for neutral beam (h)'
1722:C
1723:     constname(70) = 't_nb_rd'
1724:     t_nb_rd = 100d0/3600d0
1725:     constdesc(70) = 'ramp-down time for neutral beam (h)'
1726:C
1727:     constname(71) = 't_burn'
1728:     t_burn = 600d0/3600d0
1729:     constdesc(71) = 'plasma burn time (h)'
1730:C
1731:     constname(72) = 'tcycle_nb'
1732:     tcycle_nb = (100d0+600d0+100d0)/3600d0
1733:     constdesc(72) = 'cycle time for neutral beam (h)'
1734:C
1735:     constname(73) = 'E_nb'
1736:     E_nb = 1.3D0
1737:     constdesc(73) = 'neutral beam particle energy (MeV)'
1738:C
1739:     constname(74) = 'No'
1740:     No = 6.022D23
1741:     constdesc(74) = 'Avogadros number'
1742:C
1743:     constname(75) = 'nedge_tot'
1744:     nedge_tot = (1.0D20 + 1.0D20 + 1.0D20*5D-4)
1745:     constdesc(75) = ''
1746:C
1747:     constname(76) = 'nedge_t'
1748:     nedge_t = 1.0D20
1749:     constdesc(76) = ''
1750:C
1751:     constname(77) = 'nedge_d'
1752:     nedge_d = 1.0D20
1753:     constdesc(77) = ''
1754:C
1755:     constname(78) = 'nedge_p'
1756:     nedge_p = 1.0D20 * 5.0D-4
1757:     constdesc(78) = ''
1758:C
1759:     constname(79) = 'mgas_mid_iss'
1760:     mgas_mid_iss = 3.0D-1
1761:     constdesc(79) = 'fraction of flow from middle of isotope '
1762:    &          //'separation system directed to gas puffer'
1763:C
1764:     constname(80) = 'mgas_bot_iss'
1765:     mgas_bot_iss = 3.0D-1
1766:     constdesc(80) = 'fraction of flow from bottom of isotope '
1767:    &          //'separation system directed to gas puffer'
1768:C
1769:     constname(81) = 'mu_gas'
1770:     mu_gas = 1.0D0
1771:     constdesc(81) = 'mobility fraction for gas puffer'
1772:C
1773:     constname(82) = 'Vpellet'
1774:     Vpellet = 4.5D-7
1775:     constdesc(82) = 'volume of pellet injector pellet (m3)'
1776:C
1777:     constname(83) = 'finj_err'
1778:     finj_err = 3.0D-1
1779:     constdesc(83) = 'erosion fraction of deuterium and tritium for '
1780:    &          //'pellet injector'
1781:C
1782:     constname(84) = 'linj_mid_iss'
1783:     linj_mid_iss = 3.0D-1
1784:     constdesc(84) = 'fraction of flow from middle of isotope '
1785:    &          //'separation system to pellet injector'
1786:C
1787:     constname(85) = 'linj_bot_iss'
1788:     linj_bot_iss = 3.0D-1
1789:     constdesc(85) = 'fraction of flow from bottom of isotope '
1790:    &          //'separation system to pellet injector'
1791:C
1792:     constname(86) = 'mu_ext'
1793:     mu_ext = 1.0D0
1794:     constdesc(86) = 'mobility fraction of inventory in extruders'
1795:C
1796:     constname(87) = 'kpellet_t'
1797:     kpellet_t = 2*5.294D4
1798:     constdesc(87) = 'mole composition of tritium per unit volume of '
1799:    &          //'pellet injector pellet'
1800:C
1801:     constname(88) = 'kpellet_d'
1802:     kpellet_d = 2*4.979D4
1803:     constdesc(88) = 'mole composition of deuterium per unit volume '
1804:    &          //'of pellet injector pellet'
1805:C
1806:     constname(89) = 'kpellet_p'
1807:     kpellet_p = 2*4.287D4
1808:     constdesc(89) = 'mole composition of protium per unit volume of '
1809:    &          //'pellet injector pellet'
1810:C
1811:     constname(90) = 'fcap_prop'
1812:     fcap_prop = 9.996D-1
1813:     constdesc(90) = 'farction of propellant recaptured before entry '
1814:    &          //'into plasma'
1815:C
1816:     constname(91) = 'fnon_iss'
```

```
1817:      fnon_iss = 0.7D0
1818:      constdesc(91) = 'fraction of moles in pellet injector exhaust '
1819:     &        // 'stream not processd bu the isotope'
1820:     &        // 'separation system'
1821:C
1822:      constname(92) = 'liss_p'
1823:      liss_p = 3.2D-2
1824:      constdesc(92) = 'fraction of flow from top of isotope separation '
1825:     &          //'system directed to pellet injector'
1826:C
1827:      constname(93) = 'nprop'
1828:      nprop = 4.45D-1
1829:      constdesc(93) = 'amount of propellant gas required per pellet '
1830:     &          //'for pellet injector (moles)'
1831:C
1832:      constname(94) = 'mu_sto'
1833:      mu_sto = 1.0D-2
1834:      constdesc(94) = 'mobility fraction on storage inventories'
1835:C
1836:      constname(95) = 'linj_p'
1837:      linj_p = 3.2D-2
1838:      constdesc(95) = 'fraction of fliw from top of isotope separation '
1839:     &          //'system directed to pellet injector'
1840:
1841:C
1842:C --- idfvar(ndfvar) ...
1843:C
1844:
1845:C  ... C_t
1846:      idfvar(1) = 21
1847:C  ... C_d
1848:      idfvar(2) = 29
1849:C  ... C_p
1850:      idfvar(3) = 37
1851:C  ... D_t
1852:      idfvar(4) = 23
1853:C  ... D_d
1854:      idfvar(5) = 31
1855:C  ... D_p
1856:      idfvar(6) = 39
1857:C  ... Tpv
1858:      idfvar(7) = 45
1859:C  ... Dpv
1860:      idfvar(8) = 50
1861:C  ... Ppv
1862:      idfvar(9) = 57
1863:C  ... Hepv
1864:      idfvar(10) = 59
1865:C  ... Nimpt
1866:      idfvar(11) = 81
1867:C  ... Nnhydro_imp
1868:      idfvar(12) = 83
1869:C  ... Nco4_imp
1870:      idfvar(13) = 84
1871:C  ... Nnq3_imp
1872:      idfvar(14) = 85
1873:C  ... Nq2o_imp
1874:      idfvar(15) = 86
1875:C  ... Timp_
1876:      idfvar(16) = 87
1877:C  ... Dimp_
1878:      idfvar(17) = 88
1879:C  ... Pimp_
1880:      idfvar(18) = 89
1881:C  ... Ns
1882:      idfvar(19) = 95
1883:C  ... T2_sub
1884:      idfvar(20) = 99
1885:C  ... D2_sub
1886:      idfvar(21) = 100
1887:C  ... P2_sub
1888:      idfvar(22) = 101
1889:C  ... Tpmem
1890:      idfvar(23) = 115
1891:C  ... Dpmem
1892:      idfvar(24) = 116
1893:C  ... Ppmem
1894:      idfvar(25) = 117
1895:C  ... Niss_
1896:      idfvar(26) = 143
1897:C  ... Tiss_
1898:      idfvar(27) = 144
1899:C  ... Diss_
1900:      idfvar(28) = 146
1901:C  ... Ncryo_
1902:      idfvar(29) = 159
1903:C  ... Tnb_cryo
1904:      idfvar(30) = 163
1905:C  ... Dnb_cryo
1906:      idfvar(31) = 170
1907:C  ... Tnb_wall
1908:      idfvar(32) = 178
1909:C  ... Next_12
1910:      idfvar(33) = 211
1911:C  ... Text_12
1912:      idfvar(34) = 212
1913:C  ... Dext_12
1914:      idfvar(35) = 213
1915:C  ... Ngas_tot
1916:      idfvar(36) = 195
1917:C  ... Tgas_
1918:      idfvar(37) = 197
1919:C  ... Dgas_
1920:      idfvar(38) = 198
1921:C  ... Nsto_tri_
1922:      idfvar(39) = 257
1923:C  ... Tsto_tri_
1924:      idfvar(40) = 258
1925:C  ... Dsto_tri_
1926:      idfvar(41) = 260
1927:C  ... Nsto_deu_
1928:      idfvar(42) = 267
1929:C  ... Tsto_deu_
1930:      idfvar(43) = 268
```

```
1931:C  ... Dsto_deu_
1932:      idfvar(44) = 270
1933:C  ... Nsto_pro_
1934:      idfvar(45) = 281
1935:C  ... Tsto_pro_
1936:      idfvar(46) = 282
1937:C  ... Dsto_pro_
1938:      idfvar(47) = 284
1939:      ierr = 0
1940:
1941:      if(lpr.ge.1) then
1942:        write(6,*) '*** variables ***'
1943:        do i=1,nvar
1944:          write(6,*) 'varname(',i,') ',varname(i)
1945:        end do
1946:      end if
1947:      if(lpr.eq.2) then
1948:        write(6,*) '*** constants ***'
1949:        do i=1,nconst
1950:          write(6,*) constname(i),': ',C(i)
1951:        end do
1952:      end if
1953:
1954:      return
1955:      end
```

_fileninp.f

```
 1:      subroutine FILENAMEINPUT
 2:c=================================================================
 3:c      Fusion reactor tritium inventory evaluation system
 4:c-----------------------------------------------------------------
 5:c  Input file or directory/folder names and calculation options.
 6:c-----------------------------------------------------------------
 7:c $Id: _fileninp.f,v 1.6 1997/05/07 02:18:31 $
 8:c-----------------------------------------------------------------
 9:c=================================================================
10:      include '_sizes.h'
11:      include '_commons1.h'
12:c
13:*     character*128 print
14:*     character*128 result
15:*     common /FILES/ print, result
16:c*****************************************************************
17:c*   ... file names
18:c*
19:c*    PRINT : standard output ( I/O unit 6 )
20:c*    RESULT : top directory of result output files (I/O unit 10)
21:c*
22:c*     In Mac - Directory separator is ":" and put ":" on
23:c*            the head if the directory path is relative path
24:c*            ( as opposed to DOS/UNIX convention.)
25:c*
26:c*    OUTPUTALL = 1    (output calculation results of all variable
27:c*                     overriding specification in $OUTPUT)
28:c*    JDEBUG = 1       (debug output on if non zero)
29:c*
30:c*****************************************************************
31:c
32:      character*128 cwork
33:      namelist /FILES/ PRINT, RESULT, METHOD, JDEBUG, OUTPUTALL
34:c
35:      PRINT  = ' '
36:      RESULT = ' '
37:      METHOD = ' '
38:      JDEBUG = 0
39:      OUTPUTALL = 0
40:c
41:      write(6,*) '-----------------------------------------------'
42:      write(6,*) ' ... input file/folder name and options ...........'
43:      write(6,*) '-----------------------------------------------'
44:c
45:c----** input file names by namelist **-------------------------------
46:c
47:      read(unit=50,nml =FILES)
48:c
49:c
50:      if ( PRINT.ne.' ' ) then
51:c       LLP    = INDEX(PRINT,' ') - 1
52:c       if ( LLP.lt.0 ) LLP = LEN(PRINT)
53:        do LLP = len(PRINT),1,-1
54:          if (PRINT(LLP:LLP).ne.' ') goto 140
55:        end do
56:        LLP =1
57:  140   continue
58:
59:        open( 6, file =PRINT(:LLP), status ='UNKNOWN', err =100,
60:     &          iostat =IOS )
61:      end if
62:c
63:      if ( RESULT.ne.' ' ) then
64:c
65:c  ... remove ':' on the head if system is not Macintosh
66:c
67:        if( result(1:1).eq.':' .and. sysname.ne.'Macintosh' ) then
68:          cwork = result(2:)
69:          result = cwork
70:          write(6,*) '!!! ":" on the head of RESULT directory name',
71:     &                ' is removed.'
72:        end if
73:c
74:c       LLR    = INDEX(RESULT,' ') - 1
75:c       if ( LLR.lt.0 ) LLR = LEN(RESULT)
76:c
77:        do LLR = len(RESULT),1,-1
78:          if (RESULT(LLR:LLR).ne.' ') goto 150
79:        end do
```

```
80:        LLR = 1
81:  150    continue
82:c
83:c        open( 10, file =RESULT(:LLR), status ='UNKNOWN', form =
84:c    &          'FORMATTED', err =110, iostat =IOS )
85:c
86:      end if
87:c
88:      if ( PRINT.ne.' ' ) then
89:        write(6,*) ' ... printout filename : <', PRINT(:LLP), '>'
90:      end if
91:      if ( RESULT.ne.' ' ) then
92:        write(6,*) ' ... result output directory : <',RESULT(:LLR),'>'
93:      end if
94:c
95:c ...
96:c
97:      if(METHOD.eq.'EULER' ) then
98:        write(6,*) '=== Problem is solved by explicit Euler method.'
99:        METHOD = 'EULER'
100:     else if( METHOD.eq.' '.or.METHOD.eq.'RK' ) then
101:       write(6,*)
102:    &      '=== Problem is solved by 4th order Runge-Kutta method.'
103:       METHOD = 'RK'
104:     else
105:       write(6,*) 'XXX unsupported solution method. METHOD=<',
106:    &            METHOD,'>'
107:       stop 888
108:     end if
109:c
110:     if(JDEBUG.ne.0) then
111:       write(6,*) '!!! Specified debug option level ',jdebug
112:     end if
113:     if(OUTPUTALL.ne.0) then
114:       write(6,*) '!!! output all results.',
115:    &            ' Specifications in $OUTPUT are ignored.',
116:    &            outputall
117:     end if
118:c
119:     return
120:c
121:c-----** error message section **-----------------------------------------
122:c
123:  100 write(6,*) 'xxxx failed to open stdout as named file <',
124:    &       PRINT(:LLP), '> error code = ', IOS
125:     stop 888
126:c
127:c 110 write(6,*) 'xxxx failed to open result file <', RESULT(:LLR),
128:c   &       '> error code = ', IOS
129:c    stop 888
130:c
131:     end
```

_cntinp.f

```
1:     subroutine controlinput
2:C=================================================================
3:C      Fusion reactor tritium inventory evaluation system
4:C-----------------------------------------------------------------
5:C  input problem control data
6:C
7:C  first version Feb 1997:
8:C  update:
9:C-----------------------------------------------------------------
10:C $Id: _cntinp.f,v 1.8 1997/05/07 07:28:16 $
11:C=================================================================
12:     implicit real*8 (A-H,O-Z)
13:C
14:     include '_sizes.h'
15:C
16:C   .... including common areas here and not passing by arguments
17:C        is only to use NAMELIST input for variables in them.
18:C
19:     include '_commons1.h'
20:C
21:     include 'varconst.h'
22:C
23:C-----** local data **-----------------------------------------------
24:C
25:
26:C
27:C*******************************************************************
28:C  << control data specification >>
29:C
30:C   Input data are specified by NAMELIST input.
31:C   Input namelist blocks in the following order.
32:C
33:C   For convinience to users, current input allows comment lines
34:C   starting with '#'.
35:C
36:C  ----------------------------
37:C  *  namelist $FILES  ...  $END   (FILENAMEINPUT routine input this)
38:C  ----------------------------
39:C
40:C    Specify filenames one per one line;
41:C
42:C      PRINT = 'printout-file-name'
43:C      RESULT = 'calculation result output directory/folder'
44:C      JDEBUG = <printout debugging information if non zero>
45:C      OUTPUTALL = <output results of all variables ignoring $OUTPUT>
46:C
47:C  ----------------------------
48:C  *  namelist $TIME  ... $END    (this routine inputs)
49:C  ----------------------------
50:C
51:C    Simulation time & time step information.
52:C
```

```
53:C      TMAX = total-simulation-time  (in second)
54:C
55:C      NTIMES    = number of time intervals
56:C      TIMES()   = time interval boundaries (TIMES(1) must be 0)
57:C
58:C      NTIMEZONE = number of time zones
59:C      NSTEPS()  = number of time intervals in each time zone
60:C      TZBOUND() = boundaries of time zones.(NTIMEZONE+1 entries)
61:C
62:C    * Users should input in either NTIMES-TIMES combination or
63:C      NTIMEZONE-NSTES-TZBOUND combination.
64:C
65:C   Time values are convertd into hour for calculation
66:C
67:C ---------------------------------------------
68:C *  $OUTPUT ... $END   (this routine inputs)
69:C ---------------------------------------------
70:C
71:C    Specify variable names whose values of all time steps are output
72:C    as a comma separated list of names.
73:C
74:C      variable1, variable2, variable3, ..., variablen
75:C
76:C    Can be input any number of lines and a line having "$END" on
77:C    the head terminates this section..
78:C
79:C ---------------------------------------------
80:C *  $CONSTANT ... $END   (this routine inputs)
81:C ---------------------------------------------
82:C
83:C    Change constant value if necessary.
84:C    Repeat lines as follows;
85:C
86:C      constant-name = value
87:C
88:C    Can be input none or any number of lines and a line having "$END" on
89:C    the head terminates this section..
90:C
91:C ---------------------------------------------
92:C *  $INITIAL ... $END   (INITIALINP routine inputs)
93:C ---------------------------------------------
94:C
95:C    Change initial value if necessary.
96:C    Repeat lines as follows;
97:C
98:C      variable-name = value
99:C
100:C    Can be input none or any number of lines and a line having "$END" on
101:C    the head terminates this section..
102:C
103:C*********************************************************************
104:C
105:      namelist /TIME/ TMAX,NTIMES,TIMES,NTIMEZONE,NSTEPS,TZBOUND,
106:     &              NSUBSTEPS
107:CCC   namelist /OUTPUT/ NVAROUT, VAROUT
108:C
109:C  ... name & value pairs to modify constants or initial value
110:C
111:      common /TEMPNAME/ NAME
112:      common /TEMPVAL/ VAL
113:      parameter (MAXTEMPNM = 64)
114:      character*16 NAME(MAXTEMPNM)
115:      double precision VAL(MAXTEMPNM)
116:C
117:      namelist /CONSTANT/ NAME, VAL
118:C
119:      character*80 LINE
120:C
121:C--------** input time specifications **-------------------------------
122:C
123:      NTIMEZONE = 0
124:      TIMES(0)  = 0.0d0
125:      TZBOUND(0)  = 0.0d0
126:      do i=1,MAXTZONE
127:        NSUBSTEPS(i) = 1
128:      end do
129:C
130:      write(6,*) ' ---------------------------------'
131:      write(6,*) ' ... input TIME specififcations ...'
132:      write(6,*) ' ---------------------------------'
133:C
134:      NTIMES = 0
135:
136:      read(50,nml=time)
137:C
138:      nerr = 0
139:C
140:C  ... timezone is specified ................
141:C
142:      if( NTIMEZONE.ne.0 ) then
143:C
144:        if( ntimezone.gt.maxtzone ) then
145:          write(6,*) 'xxx number of time zones NTIMEZONE = ',
146:     &            ntimezone,
147:     &      'exceeds program limit (MAXTZONE=',MAXTZONE,')'
148:          stop 999
149:        end if
150:C
151:        write(6,*) ' NTIMEZONE = ',NTIMEZONE
152:C
153:        do i=1,ntimezone
154:C
155:          write(6,7100) i,NSTEPS(I),
156:     &        'tzbound(',i-1,') tzbound(',i,') : ',
157:     &                tzbound(i-1),tzbound(i)
158:          write(6,7102) NSUBSTEPS(I)
159:C
160: 7100      format(1x,'Timezone ',i3,' : ',i4,' time steps'
161:     &         /5x,a,i3,a,i3,a,1p,2d12.5)
162: 7102      format(1x,'   each step is divided into ',i5,' substeps.')
163:C
164:          if( nsteps(i).le.0 ) then
165:            write(6,*) 'xxx NSTEPS(',i,') must be positive but ',
166:     &          nsteps(i)
```

```
167:          nerr = nerr + 1
168:        endif
169:        if( nsubsteps(i).le.0 ) then
170:          write(6,*) 'xxx NSUBSTEPS(',i,') must be positive but ',
171:    &         nsubsteps(i)
172:          nerr = nerr + 1
173:        endif
174:C
175:        if( tzbound(i).lt.0.0d0 .or. tzbound(i).le.tzbound(i-1) )
176:    &         then
177:          write(6,*) 'xxx invalid time zone boundary (zone ',i,')'
178:          write(6,*) ' tzbound(',i,') tzbound(',i-1,') : ',
179:    &                tzbound(i),tzbound(i-1)
180:          nerr = nerr + 1
181:        end if
182:C
183:        if( nsteps(i).ne.0 ) then
184:          ddt = (tzbound(i)-tzbound(i-1))/nsteps(i)
185:          write(6,*) '     step width ',ddt,' (sec) ',ddt/3600.0,
186:    &              ' (hour)'
187:        end if
188:C
189:        k = ntimes
190:        if( k.lt.maxtimes ) then
191:          times(k) = tzbound(i-1)
192:          do j=1,nsteps(i)
193:            if( k+j.lt.maxtimes ) then
194:              times(k+j) = times(k)  + j*ddt
195:            end if
196:          end do
197:        end if
198:        ntimes = ntimes + nsteps(i)
199:      end do
200:C
201:      if( ntimes.gt.maxtimes ) then
202:        write(6,*) 'xxx number of time steps NTIMES = ', ntimes,
203:    &     'exceeds program limit (MAXTIMES=',MAXTIMES,')'
204:        stop 999
205:      end if
206:C
207:C ... no time zone specified .....
208:C
209:    else
210:
211:      if( NTIMES.le.0 ) then
212:        write(6,*) 'xxx number of time steps NTIMES = ', ntimes,
213:    &     'must be positive.'
214:        stop 999
215:      endif
216:
217:      if( ntimes.gt.maxtimes ) then
218:        write(6,*) 'xxx number of time steps NTIMES = ', ntimes,
219:    &     'exceeds program limit (MAXTIMES=',MAXTIMES,')'
220:        stop 999
221:      endif
222:
223:      do i=1,ntimes
224:        if( times(i).lt.0.0d0 .or. times(i).le.times(i-1) ) then
225:          write(6,*) 'xxx invalid time step boundary (step ',i,')'
226:          write(6,*) ' times(',i,') times(',i-1,') : ',
227:    &                times(i),times(i-1)
228:          nerr = nerr + 1
229:        end if
230:      end do
231:    end if
232:c
233:check %%%%%%
234:c     do i=0,ntimes
235:c     write(6,*) ' times(',i,') <',times(i),'>'
236:c     end do
237:c %%%%%%%%%%
238:C
239:    if( times(ntimes).ne.tmax ) then
240:      write(6,*) '!!! TMAX is not equal to the last time boundary'
241:      write(6,*) '    set TMAX to TIMES(NTIMES).'
242:      TMAX = TIMES(NTIMES)
243:    end if
244:c
245:c   ... convert times into hour.
246:c
247:    do i=0,ntimes
248:      times(i) = times(i)/3600.0d0
249:    end do
250:    tmax = tmax/3600.d0
251:C
252:    if( nerr.gt.0 ) then
253:      write(6,*) 'XXX time step information is not valid.'
254:    end if
255:C
256:C--------** output variable specifications **-------------------------
257:C
258:    NVAROUT = 0
259:    do i=1,maxvarout
260:      varout(i) = ' '
261:    end do
262:C
263:    write(6,*) ' ----------------------------------------'
264:    write(6,*) ' ... input variable names to be output ...'
265:    write(6,*) ' ----------------------------------------'
266:    write(6,*) ' (number of total defined variables = ',nvar,')'
267:
268:check %%%%%%
269:c     do i=1,nvar
270:c     write(6,*) ' varname(',i,') <',varname(i),'>'
271:c     end do
272:c %%%%%%%%%%
273:C
274:cccc  read(50,nml=output)
275:c
276:    line=' '
277:    read(50,'(a)') line
278:    if( line(1:7).ne.'&OUTPUT' .and. line(1:7).ne.'$OUTPUT') then
279:      write(6,*) 'xxx &OUTPUT or $OUTPUT is not found.'
280:      write(6,*) 'xxx line: ',line
```

```
      stop 888
    end if
c
    nvarout = 0
c
400 line = ' '
    read(50,'(a)',end=1900) line
    if( line(1:4).ne.'&END' .and. line(1:4).ne.'$END') then
      i1 = 1
      do i=1,len(line)

        k = index(line(i1:),',')
        nvarout = nvarout + 1

        if( nvarout.gt.maxvarout ) then
          write(6,*) 'xxx number of output variable NVAROUT = ',
     &      nVAROUT,
     &      'exceeds program limit (MAXVAROUT=',MAXVAROUT,')'
          stop 999
        end if

        if( k.eq.0 ) then
          i2 = len(line)
        else
          i2 = i1+k-2
        end if
        if(line(i1:i2).eq.' ' ) then
          nvarout = nvarout - 1
        else
          varout(nvarout) = line(i1:i2)
        end if
        i1 = i2 + 2
        if( i1.gt.len(line) ) goto 410
      end do
410   continue
      goto 400
    endif
C
C ... remove unnecessary blanks
C
    do i=1,nvarout
      call CCOMP( VAROUT(i), len(varout(i)), varout(i), IFL )
    end do
c
c ... match with VARNAME entry ...
c
    do 120 i=1,nvarout
c
      do j=1,i-1
        if( varout(j).eq.varout(i) ) then
          write(6,*) '!!! VAROUT ',varout(i),
     &        ' specified more than once.'
          ivarout(i) = 0
          goto 120
        end if
      end do
c
      do j=1,nvar
        kk = index(varname(j),'(t)')
        if( kk.gt.0 ) then
          kk = kk - 1
        else
          kk = len(varname(j))
        end if
        if( varname(j)(:kk).eq.varout(i) ) then
          ivarout(i) = j
          goto 130
        end if
      end do
c
      write(6,*) 'xxx ',varout(i),' is not in variable name list.'
      ivarout(i) = -1
      nerr = nerr + 1
      goto 120
c
130   write(6,'(1x,a,i3,a,a,a,i4,a)')
     &    ' VAROUT(',i,') : ',varout(i),' (X(',ivarout(i),'))'
c
120 continue
c
    kk = 0

    do i=1,nvarout
      if( ivarout(i).gt.0 ) then
        kk = kk + 1
        if( kk.ne.i ) varout(kk) = varout(i)
      end if
    end do

    nvarout = kk
    write(6,*) ' ... ',nvarout,' variables are output as result.'
C
    if(OUTPUTALL.ne.0) then
      write(6,*)
     &    '!!! all variables are output as calculation result.',
     &    ' Specifictions in $OUTPUT section will be igonored.'
    endif
C
C ... test that files can be opened under specified "result" directory.
C
    iunit = 10
    call fileopen( result(:llr), 'file000', iunit, ioerr )
    if( ioerr.ne.0 ) then
      write(6,*) 'XXX tried to open a file under specified output',
     &      ' directory, but failed.'
      write(6,*) ' Please check existence of the directory/folder ',
     &      'or access permission etc.'
c
      if( sysname.eq.'Macintosh') pause 'PAUSE'
c
      stop 888
    endif
c
    close (iunit, status='DELETE')
```

```
395:C
396:C---------** constant variable value re-definition **-----------------
397:C
398:      line=' '
399:      read(50,'(a)',end=1900) line
400:      if( line(1:9).ne.'&CONSTANT' .and. line(1:9).ne.'$CONSTANT') then
401:        write(6,*) 'xxx &CONSTANT or $CONSTANT is not found.'
402:        write(6,*) 'xxx line: ',line
403:        stop 888
404:      end if
405:c
406:  500 line = ' '
407:      read(50,'(a)',end=1900) line
408:      if( line(1:4).ne.'&END' .and. line(1:4).ne.'$END') then
409:
410:        k = index( line, '=' )
411:        if ( k.eq.0 ) then
412:          write(6,*) 'xxx constant change line must be "name = value"'
413:          write(6,*) 'xxx line: ',line
414:          stop 888
415:        end if
416:
417:        call changeconst( line(:k-1), line(k+1:), ierr )
418:
419:        goto 500
420:
421:      endif
422:C
423:C ... print out
424:C
425:      write(6,*) '--- constant values'
426:      do i=1,nconst
427:        write(6,711) 'C(',i,') <',constname(i),'> =',C(i)
428:      end do
429:  711 format(1x,a,i3,a,a,a,1p,d13.5)
430:C
431:C---------** initial value re-specification **-----------------
432:C
433:*     do i=1,MAXTEMPNM
434:*       NAME(i) = ' '
435:*       VAL(i) = 0.0d0
436:*     end do
437:*
438:*     read(50,nml=initial)
439:c
440:*     do i=1,MAXTEMPNM
441:*       if(NAME(i).ne.' ') then
442:*         do j=1,nvar
443:*           kk = index(varname(j),'(')
444:*           if( kk.eq.0 ) kk = len(varname(j))
445:*           if( varname(j)(:kk).eq.name(i) ) then
446:*             goto 300
447:*           end id
448:*         end do
449:*         write(*,*) 'XXX try to change initial value of <',name(i),
450:*    &        '> but no such a variable exists.'
451:*         nerr = nerr + 1
452:*         goto 320
453:C
454:* 300     write(*,*) '== initial value of <',name(i),
455:*    &        '> changed from ', X(j),' to ',val(i)
456:*         X(j) = val(i)
457:C
458:*       endif
459:*     enddo
460:C
461:* 320     continue
462:C
463:C-----** error stop **-----------------------------------------------
464:C
465:      if( nerr.gt.0 ) then
466:        write(6,*)
467:     & 'XXX Fatal error detected in $TIME,$OUTPUT & $CONSTANT input.'
468:        if( sysname.eq.'Macintosh' ) then
469:          write(*,*)
470:     & 'XXX Fatal error detected in $TIME,$OUTPUT & $CONSTANT input.'
471:          pause
472:        end if
473:        stop 888
474:      end if
475:C
476:      return
477:C
478: 1900 write(6,*) 'xxx unexpected end of file on input.'
479:      stop 888
480:      return
481:      end
482:C
483:C
484:      subroutine ccomp( outch, ln, inch, ifl )
485:C
486:C=================================================================
487:C PURPOSE:  'COMPRESS' CHARACTER STRING INCH BY REMOVING BLANK
488:C          INTO CHARACTER STRING 'OUTCH' UPTO LN CHARACTERS.
489:C          ('OUTCH' & 'INCH' CAN OVERLAP IN MEMORY BUT
490:C           THE STARTING ADDRESS OF 'OUTCH' MUST PRECEED OR BE SAME AS
491:C           THAT OF 'INCH'. )
492:C
493:C          IFL : FLAG
494:C            0 = NORMAL END
495:C            1 = NUMBER OF NON-BLANK CHARACTERS IN INCH EXEEDS LN.
496:C=================================================================
497:      character*(*)   outch, inch
498:      character*1     ch1
499:c
500:      ifl = 0
501:      ll  = len(inch)
502:      nn  = 0
503:      do 100 i=1,ll
504:        if( inch(i:i).ne.' ' ) then
505:          nn  = nn + 1
506:          if( nn.le.ln ) then
507:            ch1 = inch(i:i)
508:            outch(nn:nn) = ch1
```

```
509:          endif
510:        endif
511: 100 continue
512:      if( nn.lt.ln ) outch(nn+1:ln) = ' '
513:      if( nn.gt.ln ) ifl = 1
514:      return
515:      end
```

_initinp.f

```
 1:      subroutine initialinp
 2:C=================================================================
 3:C       Fusion reactor tritium inventory evaluation system
 4:C-----------------------------------------------------------------
 5:C  input problem control data (2) : iniitial value change if any.
 6:C
 7:C  first version Feb 1997:
 8:C  update:
 9:C-----------------------------------------------------------------
10:C $Id: _initinp.f,v 1.1 1997/05/07 05:21:33 $
11:C=================================================================
12:      implicit real*8 (A-H,O-Z)
13:C
14:      include '_sizes.h'
15:C
16:C    .... including common areas here and not passing by arguments
17:C        is only to use NAMELIST input for variables in them.
18:C
19:      include '_commonsl.h'
20:C
21:      include 'varconst.h'
22:C
23:C----** local data **-------------------------------------------
24:C
25:      character*80 LINE
26:C
27:C--------** constant variable value re-definition **----------------
28:C
29:      line=' '
30:      read(50,'(a)',end=1900) line
31:      if( line(1:8).ne.'&INITIAL' .and. line(1:8).ne.'$INITIAL') then
32:        write(6,*) 'xxx line is not &INITIAL or $INITIAL.'
33:        write(6,*) 'xxx line: ',line
34:        if( sysname.eq.'Macintosh' ) then
35:          write(*,*) 'xxx line is not &INITIAL or $INITIAL.'
36:          write(*,*) 'xxx line: ',line
37:          pause
38:        end if
39:        stop 888
40:      end if
41:c
42:      nerr = 0
43:c
44: 500 line = ' '
45:      read(50,'(a)',end=1900) line
46:      if( line(1:4).ne.'&END' .and. line(1:4).ne.'$END') then
47:
48:        k = index( line, '=' )
49:        if ( k.eq.0 ) then
50:          write(6,*) 'xxx line must be "name = value"'
51:          write(6,*) 'xxx line: ',line
52:          if( sysname.eq.'Macintosh' ) then
53:            write(*,*) 'xxx line must be "name = value"'
54:            write(*,*) 'xxx line: ',line
55:            pause
56:          end if
57:          stop 888
58:        end if
59:
60:        call changeinitial( line(:k-1), line(k+1:), ierr )
61:        if (ierr.ne.0) nerr = nerr + 1
62:
63:        goto 500
64:
65:      endif
66:c
67:      if( nerr.gt.0 ) then
68:        write(6,*) 'xxx Fatal error detected in $INITIAL input.'
69:        if( sysname.eq.'Macintosh' ) then
70:          write(*,*) 'xxx Fatal error detected in $INITIAL input.'
71:          pause
72:        end if
73:        stop 888
74:      end if
75:C
76:      return
77:C
78: 1900 write(6,*) 'No initial value specification by user.'
79:      return
80:c
81:      return
82:      end
```

_changes.f

```
 1:      subroutine changeconst( name, value, ierr )
 2:c=================================================================
 3:c       Fusion reactor tritium inventory evaluation system
 4:c-----------------------------------------------------------------
 5:c     change constant value
```

```
c-----------------------------------------------------------------------
c  $Id: _changes.f,v 1.2 1997/05/07 02:20:57 $
c=======================================================================
      character*(*) name, value
c
      include '_sizes.h'
      include 'varconst.h'
c
      character*32 cwork
      double precision C0
c ------------------------------------------------
c
      ierr = 0
c
c ... remove unnecessary blanks if any
c
      call ccomp( name, len(name), name, ifl )
      ln = index(name,' ')-1
      if( ln.lt.0 ) ln = len(name)
c
      call ccomp( value, len(value), value, ifl )
      lv = index(value,' ')-1
      if( lv.lt.0 ) lv = len(value)
c
c ... search constants name list
c
      do in=1,nconst
        if( constname(in).eq.name(:ln) ) then
          goto 120
        endif
      end do
      write(6,*) 'xxx cannot find constant name <',name(:ln),'>'
      ierr = 1
      return
c
  120 continue
c
      if( lv.gt.len(cwork) ) then
        write(6,*) 'xxx value character string is too long <',
     &           value(:lv),'>'
        ierr = 1
        return
      endif
c
c ... convert value string into numerical value
c
      C0 = C(in)
      cwork = value(:lv)
      read( cwork, '(d32.0)', err=190 ) C(in)
c
      write(6,*) '== value of constant <',name(:ln),'> is changed to ',
     & C(in),' from ',C0
      return
c
  190 write(6,*) 'xxx value form is invalid in <',name(:ln),' = ',
     &  value(:lv),'>'
      ierr = 1
      return
      end
c
c=======================================================================
c
      subroutine changeinitial( name , value, ierr )
c ------------------------------------------------
c  change initial value
c ------------------------------------------------
      character*(*) name, value
c
      include '_sizes.h'
      include 'varconst.h'
c
      character*32 cwork
      double precision C0
c ------------------------------------------------
c
      ierr = 0
c
c ... remove unnecessary blanks if any
c
      call ccomp( name, len(name), name, ifl )
      ln = index(name,' ')-1
      if( ln.lt.0 ) ln = len(name)
c
      call ccomp( value, len(value), value, ifl )
      lv = index(value,' ')-1
      if( lv.lt.0 ) lv = len(value)
c
c ... search variable name list
c
      do in=1,nvar
        kk = index(varname(in),'(t)')
        if( kk.gt.0 ) then
          kk = kk - 1
        else
          kk = len(varname(in))
        endif
        if( varname(in)(:kk).eq.name(:ln) ) then
          goto 120
        endif
      end do
      write(6,*) 'xxx cannot find variable name <',name(:ln),'>'
      ierr = 1
      return
c
  120 continue
c
      if( lv.gt.len(cwork) ) then
        write(6,*) 'xxx value character string is too long <',
     &           value(:lv),'>'
        ierr = 1
        return
      endif
c
c ... convert value string into numeric value
```

```
120:c
121:      C0 = XVARS(0,in)
122:      cwork = value(:lv)
123:      read( cwork, '(d32.0)', err=190 ) XVARS(0,in)
124:c
125:      XVARS(1,in) = XVARS(0,in)
126:c
127:      write(6,*) '== initial value of <',name(:ln),'> is changed to ',
128:     & XVARS(0,in),' from ',C0
129:      return
130:c
131:  190 write(6,*) 'xxx value form is invalid in <',name(:ln),' = ',
132:     & value(:lv),'>'
133:      ierr = 1
134:c
135:      return
136:      end
```

_solver.f

```
 1:      subroutine solver( tmax, ntimes, times, nsteps,
 2:     &  ntimezone, delta_t, method,
 3:     &  sysname, jdebug,
 4:     &  F
 5:     & )
 6:c=================================================================
 7:c      Fusion reactor tritium inventory evaluation system
 8:c-----------------------------------------------------------------
 9:c $Id: _solver.f,v 1.7 1997/05/07 07:28:19 $
10:c=================================================================
11:c
12:      implicit real*8 (a-h,o-z)
13:c
14:      include '_sizes.h'
15:      include 'varconst.h'
16:c
17:      real*8 times(0:ntimes)
18:      integer nsteps(ntimezone)
19:c
20:      character*(*) METHOD
21:      character*(*) SYSNAME
22:c
23:      real*8 F(ndfvar)
24:c
25:c-----------------------------------------------------------------
26:c ** set initial condition **
27:c-----------------------------------------------------------------
28:c
29:      write(6,'(/1x,a)') '--- set initial condition:'
30:c
31:      delta_t = times(1) - times(0)
32:c
33:      lpr = 0
34:      if(jdebug.ne.0) lpr = 2
35:      call VARINIT(delta_t, F, lpr, ierr )
36:c
37:      write(6,'(/1x,a)')'--- checking user specified initial condition:'
38:c
39:      call INITIALINP
40:c
41:      write(6,'(/1x,a/)') '--- initial condition set complete:'
42:c
43:c     ... Initial_step must be 1 just for the first call not to
44:c        overwrite initial value settings ...
45:c
46:      Initial_step = 1
47:c
48:      J = 0
49:      dt1 = delta_t
50:c
51:c-----------------------------------------------------------------
52:c  ** timestep loop **
53:c-----------------------------------------------------------------
54:c
55:      do it = 1, ntimes
56:c
57:         t = times(it-1)
58:         delta_t = times(it) - times(it-1)
59:
60:         write(6,7000) it,t,t*3600d0,delta_t*3600d0
61: 7000    format(1x,'--- step',i4,' t=',1p,d12.5,'(',d11.4,
62:     &          ' sec) dt = ',d10.3,' sec')
63:c
64:c ... write(*,...) is on MRWE window for AsSoft FORTRAN77 for MAC
65:c
66:         if(sysname.eq.'Macintosh') then
67:            write(*,7000) it,t,t*3600d0,delta_t*3600d0
68:         endif
69:
70:c=================================================================
71:         call fdiffvect( It, J, times(it), delta_t, dt1, F,
72:     &              Initial_step  )
73:c=================================================================
74:         Initial_step = 0
75:c
76:c        ... end of sub-timestep
77:         dt1 = delta_t
78:c
79:c  --- for debugging ---------------------------
80:c
81:         if( jdebug.ge.1 ) then
82:            do i=1,ndfvar
83:               ii = idfvar(i)
84:               if(F(I).ne.0.0d0) then
85:                  TT = abs(XVARS(it-1,ii)/F(i))
86:               else
87:                  TT = 1000.0
88:               endif
```

```
 89:c
 90:            write(6,7111)
 91:    &         it,'% ',varname(ii)(1:16),' X(',ii,') : ',XVARS(it,ii),
 92:    &           F(i), TT
 93: 7111 format(1x,i4,a,a,a,i3,a,d11.4,' F: ',d11.4,' TT: ',d11.4)
 94:c
 95:          end do
 96:          write(6,*) '%%%% refered variables from dX/dt %%%%'
 97:          do i=1,nvar
 98:            if( ivarflag(i).eq.2 )
 99:    &       write(6,*) it,'% ',varname(i)(1:16),
100:    &        ' X(',i,') : ',XVARS(it,i)
101:          end do
102:          write(6,*) '%%%% other variables %%%%'
103:          do i=1,nvar
104:            if( ivarflag(i).eq.3.or.ivarflag(i).eq.4 )
105:    &       write(6,*) it,'% ',varname(i)(1:16),
106:    &        ' X(',i,') : ',XVARS(it,i)
107:          end do
108:        end if
109:c
110:c      ... end of timestep
111:c
112:     end do
113:c
114:     write(6,*) '=== calculation finished ===='
115:c
116:     return
117:     end
```

varinit.f

```
  1:     subroutine varinit(delta_t,F,pr,jerr)
  2:C
  3:C ***********************************************
  4:C   Automatically created.
  5:C   creation date : << Tue May 6 14:21:39 1997 >>
  6:C ***********************************************
  7:     implicit double precision (a-h,o-z)
  8:C
  9:C ... initial condition ...
 10:C
 11:
 12:     include '_sizes.h'
 13:     include 'varconst.h'
 14:     integer pr,jerr
 15:     double precision F(ndfvar),delta_t
 16:     double precision fppower
 17:
 18:C ... statement function ...
 19:     ABX(AA,BB,XX) = AA+ BB*XX
 20:
 21:C ... jerr(if=0>noerror,if=1>error)
 22:C ... pr(if=0>not print,if=1>print)
 23:
 24:     jerr = 0
 25:
 26:C
 27:C  Initial_step = 0 is necessary for varinit.
 28:C
 29:     Initial_step = 0
 30:     t = 0.0D0
 31:     j = 0
 32:C ... initials of X(n) ...
 33:
 34:*
 35:* --- Storage ---------------------------------------------
 36:*
 37:
 38:     delt0 = 20.d0 / 3600d0
 39:
 40:     Burn_t(0) = 2.12D-3 * 1000 * delt0 / t_p_ru
 41:     if(pr.ge.2) write(6,*) 'Burn_t=',Burn_t(0)
 42:
 43:     Burn_d(0) = Burn_t(0)
 44:     if(pr.ge.2) write(6,*) 'Burn_d=',Burn_d(0)
 45:
 46:     Reac_p(0) = ( 0.01D0 * 0.05 ) * Burn_t(0)
 47:     if(pr.ge.2) write(6,*) 'Reac_p=',Reac_p(0)
 48:
 49:     ttstep = 20d0 / 3600
 50:
 51:     Tsto_tri_(0) = Burn_t(0) * ttstep
 52:     if(pr.ge.2) write(6,*) 'Tsto_tri_=',Tsto_tri_(0)
 53:
 54:     Dsto_tri_(0) = 1.0d-3 * Tsto_tri_(0)
 55:     if(pr.ge.2) write(6,*) 'Dsto_tri_=',Dsto_tri_(0)
 56:
 57:     Psto_tri_(0) = 1.0d-3 * Tsto_tri_(0)
 58:     if(pr.ge.2) write(6,*) 'Psto_tri_=',Psto_tri_(0)
 59:
 60:     Nsto_tri_(0) = Tsto_tri_(0) + Dsto_tri_(0) + Psto_tri_(0)
 61:     if(pr.ge.2) write(6,*) 'Nsto_tri_=',Nsto_tri_(0)
 62:
 63:     Tsto_tri_m(0) = mu_sto * Tsto_tri_(0)
 64:     if(pr.ge.2) write(6,*) 'Tsto_tri_m=',Tsto_tri_m(0)
 65:
 66:     Fsto_tri_out(0) = Burn_t(0)
 67:     if(pr.ge.2) write(6,*) 'Fsto_tri_out=',Fsto_tri_out(0)
 68:*Fsto_tri_in  = (1.0D0 - linj_bot_iss - mgas_bot_iss) * Fiss_bot_ ;
 69:
 70:     Ksto_tri_(0) = 0.0d0
 71:     if(pr.ge.2) write(6,*) 'Ksto_tri_=',Ksto_tri_(0)
 72:
 73:     Ksto_deu_(0) = 0.0d0
 74:     if(pr.ge.2) write(6,*) 'Ksto_deu_=',Ksto_deu_(0)
 75:
 76:     Ksto_pro_(0) = 0.0d0
```

```
77:     if(pr.ge.2) write(6,*) 'Ksto_pro_=',Ksto_pro_(0)
78:
79:     Xsto_tri_t(0) = Tsto_tri_(0) / Nsto_tri_(0)
80:     if(pr.ge.2) write(6,*) 'Xsto_tri_t=',Xsto_tri_t(0)
81:
82:     Xsto_tri_d(0) = Dsto_tri_(0) / Nsto_tri_(0)
83:     if(pr.ge.2) write(6,*) 'Xsto_tri_d=',Xsto_tri_d(0)
84:
85:     Xsto_tri_p(0) = 1.0D0 - Xsto_tri_t(0) - Xsto_tri_d(0)
86:     if(pr.ge.2) write(6,*) 'Xsto_tri_p=',Xsto_tri_p(0)
87:
88:     Dsto_deu_(0) = Burn_d(0) * ttstep
89:     if(pr.ge.2) write(6,*) 'Dsto_deu_=',Dsto_deu_(0)
90:
91:     Tsto_deu_(0) = 1.0d-3 * Dsto_deu_(0)
92:     if(pr.ge.2) write(6,*) 'Tsto_deu_=',Tsto_deu_(0)
93:
94:     Psto_deu_(0) = 1.0d-3 * Dsto_deu_(0)
95:     if(pr.ge.2) write(6,*) 'Psto_deu_=',Psto_deu_(0)
96:
97:     Nsto_deu_(0) = Tsto_deu_(0) + Dsto_deu_(0) + Psto_deu_(0)
98:     if(pr.ge.2) write(6,*) 'Nsto_deu_=',Nsto_deu_(0)
99:
100:    Tsto_deu_m(0) = mu_sto * Tsto_deu_(0)
101:    if(pr.ge.2) write(6,*) 'Tsto_deu_m=',Tsto_deu_m(0)
102:
103:    Fsto_deu_out(0) = Burn_d(0)
104:    if(pr.ge.2) write(6,*) 'Fsto_deu_out=',Fsto_deu_out(0)
105:
106:    Fgas_in_deu(0) = Fsto_deu_out(0) / 4
107:    if(pr.ge.2) write(6,*) 'Fgas_in_deu=',Fgas_in_deu(0)
108:*Fsto_deu_in  = (1.0D0 -0.3D0 -0.3D0 -0.3D0) * Fiss_mid_ ;
109:
110:    Xsto_deu_t(0) = Tsto_deu_(0) / Nsto_deu_(0)
111:    if(pr.ge.2) write(6,*) 'Xsto_deu_t=',Xsto_deu_t(0)
112:
113:    Xsto_deu_d(0) = Dsto_deu_(0) / Nsto_deu_(0)
114:    if(pr.ge.2) write(6,*) 'Xsto_deu_d=',Xsto_deu_d(0)
115:
116:    Xsto_deu_p(0) = 1.0D0 - Xsto_deu_t(0) - Xsto_deu_d(0)
117:    if(pr.ge.2) write(6,*) 'Xsto_deu_p=',Xsto_deu_p(0)
118:
119:    Psto_pro_(0) = Reac_p(0) * ttstep
120:    if(pr.ge.2) write(6,*) 'Psto_pro_=',Psto_pro_(0)
121:
122:    Tsto_pro_(0) = 1.0d-3 * Psto_pro_(0)
123:    if(pr.ge.2) write(6,*) 'Tsto_pro_=',Tsto_pro_(0)
124:
125:    Dsto_pro_(0) = 1.0d-3 * Psto_pro_(0)
126:    if(pr.ge.2) write(6,*) 'Dsto_pro_=',Dsto_pro_(0)
127:
128:    Nsto_pro_(0) = Tsto_pro_(0) + Dsto_pro_(0) + Psto_pro_(0)
129:    if(pr.ge.2) write(6,*) 'Nsto_pro_=',Nsto_pro_(0)
130:
131:    Tsto_pro_m(0) = mu_sto * Tsto_pro_(0)
132:    if(pr.ge.2) write(6,*) 'Tsto_pro_m=',Tsto_pro_m(0)
133:*Fsto_pro_in  = ( 1.0D0 - 0.032D0 ) * Fiss_top_ ;
134:
135:    Fsto_pro_out(0) = Reac_p(0)
136:    if(pr.ge.2) write(6,*) 'Fsto_pro_out=',Fsto_pro_out(0)
137:
138:    Fsto_out_p(0) = Fsto_pro_out(0)
139:    if(pr.ge.2) write(6,*) 'Fsto_out_p=',Fsto_out_p(0)
140:
141:    Xsto_pro_t(0) = Tsto_pro_(0) / Nsto_pro_(0)
142:    if(pr.ge.2) write(6,*) 'Xsto_pro_t=',Xsto_pro_t(0)
143:
144:    Xsto_pro_d(0) = Dsto_pro_(0) / Nsto_pro_(0)
145:    if(pr.ge.2) write(6,*) 'Xsto_pro_d=',Xsto_pro_d(0)
146:
147:    Xsto_pro_p(0) = Psto_pro_(0) / Nsto_pro_(0)
148:    if(pr.ge.2) write(6,*) 'Xsto_pro_p=',Xsto_pro_p(0)
149:
150:    Fsto_tri_out_t(0) = Fsto_tri_out(0) * Xsto_tri_t(0)
151:    if(pr.ge.2) write(6,*) 'Fsto_tri_out_t=',Fsto_tri_out_t(0)
152:
153:    Fsto_deu_out_d(0) = Fsto_deu_out(0) * Xsto_tri_d(0)
154:    if(pr.ge.2) write(6,*) 'Fsto_deu_out_d=',Fsto_deu_out_d(0)
155:*
156:* --- Fueling ----------------------------------------------
157:*
158:
159:    Fuelf_t(0) = Burn_t(0) / fbu
160:    if(pr.ge.2) write(6,*) 'Fuelf_t=',Fuelf_t(0)
161:*
162:* --- The Plasma -------------------------------------------
163:*
164:
165:    Pfusion(0) = Burn_t(0) / 2.12D-3
166:    if(pr.ge.2) write(6,*) 'Pfusion=',Pfusion(0)
167:
168:    Fuel_t(0) = Burn_t(0) / fbu
169:    if(pr.ge.2) write(6,*) 'Fuel_t=',Fuel_t(0)
170:
171:    Fex_t(0) = Fuel_t(0) * ( 1 - fbu )
172:    if(pr.ge.2) write(6,*) 'Fex_t=',Fex_t(0)
173:
174:    Fplas_in_t(0) = Fuelf_t(0) * fin_pellet
175:    if(pr.ge.2) write(6,*) 'Fplas_in_t=',Fplas_in_t(0)
176:
177:    Fgas_t(0) = Fuelf_t(0) * fin_gas
178:    if(pr.ge.2) write(6,*) 'Fgas_t=',Fgas_t(0)
179:
180:    Fplas_in_d(0) = Fuelf_t(0) * fin_pellet
181:    if(pr.ge.2) write(6,*) 'Fplas_in_d=',Fplas_in_d(0)
182:
183:    Fgas_t(0) = Fuelf_t(0) * fin_gas
184:    if(pr.ge.2) write(6,*) 'Fgas_t=',Fgas_t(0)
185:*** Fuel_d = Fplas_in_d + Fgas_d + Fplas_nb_d + Fplas_prop_d ;
186:*** Fex_d      = Fuel_d - Burn_d ;
187:
188:    Fhe_ex(0) = Burn_t(0)
189:    if(pr.ge.2) write(6,*) 'Fhe_ex=',Fhe_ex(0)
190:*
```

```
191:* --- Plasma Facing Components ------------------------------
192:*
193:
194:      i_t(0) = 0.4D0
195:      if(pr.ge.2) write(6,*) 'i_t=',i_t(0)
196:
197:      i_d(0) = 0.3D0
198:      if(pr.ge.2) write(6,*) 'i_d=',i_d(0)
199:
200:      i_p(0) = 0.3D0
201:      if(pr.ge.2) write(6,*) 'i_p=',i_p(0)
202:
203:      fmMVdiv = fm ( Medge ) * ai ( Temperature , 0d0 ) * Vdiv / ( 0.5
204:     &       * 3.016D0 )
205:
206:      fmMVfw = fm ( Medge ) * ai ( Temperature , 0d0 ) * Vfw / ( 0.5 *
207:     &       3.016D0 )
208:
209:      fppp = delt0 / t_p_ru
210:
211:      Adiv_t(0) = fmMVdiv * i_t(0) * fppp
212:      if(pr.ge.2) write(6,*) 'Adiv_t=',Adiv_t(0)
213:
214:      Afw_t(0) = fmMVfw * i_t(0) * fppp
215:      if(pr.ge.2) write(6,*) 'Afw_t=',Afw_t(0)
216:
217:      Adiv_d(0) = fmMVdiv * i_d(0) * fppp
218:      if(pr.ge.2) write(6,*) 'Adiv_d=',Adiv_d(0)
219:
220:      Afw_d(0) = fmMVfw * i_d(0) * fppp
221:      if(pr.ge.2) write(6,*) 'Afw_d=',Afw_d(0)
222:
223:      Adiv_p(0) = fmMVdiv * i_p(0) * fppp
224:      if(pr.ge.2) write(6,*) 'Adiv_p=',Adiv_p(0)
225:
226:      Afw_p(0) = fmMVfw * i_p(0) * fppp
227:      if(pr.ge.2) write(6,*) 'Afw_p=',Afw_p(0)
228:
229:      d_Adiv_t(0) = 0.0D0
230:      if(pr.ge.2) write(6,*) 'd_Adiv_t=',d_Adiv_t(0)
231:
232:      d_Adiv_d(0) = 0.0D0
233:      if(pr.ge.2) write(6,*) 'd_Adiv_d=',d_Adiv_d(0)
234:
235:      d_Adiv_p(0) = 0.0D0
236:      if(pr.ge.2) write(6,*) 'd_Adiv_p=',d_Adiv_p(0)
237:
238:      d_Afw_t(0) = 0.0D0
239:      if(pr.ge.2) write(6,*) 'd_Afw_t=',d_Afw_t(0)
240:
241:      d_Afw_d(0) = 0.0D0
242:      if(pr.ge.2) write(6,*) 'd_Afw_d=',d_Afw_d(0)
243:
244:      d_Afw_p(0) = 0.0D0
245:      if(pr.ge.2) write(6,*) 'd_Afw_p=',d_Afw_p(0)
246:
247:      C_t(0) = 0.0D0
248:      if(pr.ge.2) write(6,*) 'C_t=',C_t(0)
249:
250:      C_d(0) = 0.0D0
251:      if(pr.ge.2) write(6,*) 'C_d=',C_d(0)
252:
253:      C_p(0) = 0.0D0
254:      if(pr.ge.2) write(6,*) 'C_p=',C_p(0)
255:
256:      D_t(0) = 0.0D0
257:      if(pr.ge.2) write(6,*) 'D_t=',D_t(0)
258:
259:      D_d(0) = 0.0D0
260:      if(pr.ge.2) write(6,*) 'D_d=',D_d(0)
261:
262:      D_p(0) = 0.0D0
263:      if(pr.ge.2) write(6,*) 'D_p=',D_p(0)
264:
265:      d_C_t(0) = 0.0D0
266:      if(pr.ge.2) write(6,*) 'd_C_t=',d_C_t(0)
267:
268:      d_C_d(0) = 0.0D0
269:      if(pr.ge.2) write(6,*) 'd_C_d=',d_C_d(0)
270:
271:      d_C_p(0) = 0.0D0
272:      if(pr.ge.2) write(6,*) 'd_C_p=',d_C_p(0)
273:
274:      d_D_t(0) = 0.0D0
275:      if(pr.ge.2) write(6,*) 'd_D_t=',d_D_t(0)
276:
277:      d_D_d(0) = 0.0D0
278:      if(pr.ge.2) write(6,*) 'd_D_d=',d_D_d(0)
279:
280:      d_D_p(0) = 0.0D0
281:      if(pr.ge.2) write(6,*) 'd_D_p=',d_D_p(0)
282:
283:      Tpfc_(0) = Adiv_t(0) + Afw_t(0) + C_t(0) + D_t(0)
284:      if(pr.ge.2) write(6,*) 'Tpfc_=',Tpfc_(0)
285:
286:      Tpfc_m(0) = mu_A * ( Adiv_t(0) + Afw_t(0) ) + mu_C * C_t(0) +
287:     &       mu_D * D_t(0)
288:      if(pr.ge.2) write(6,*) 'Tpfc_m=',Tpfc_m(0)
289:
290:      Dpfc_(0) = Adiv_d(0) + Afw_d(0) + C_d(0) + D_d(0)
291:      if(pr.ge.2) write(6,*) 'Dpfc_=',Dpfc_(0)
292:
293:      Ppfc_(0) = Adiv_p(0) + Afw_p(0) + C_p(0) + D_p(0)
294:      if(pr.ge.2) write(6,*) 'Ppfc_=',Ppfc_(0)
295:*
296:* --- Vacuum ------------------------------
297:*
298:*
299:
300:      Tpv(0) = 1.0D-4
301:      if(pr.ge.2) write(6,*) 'Tpv=',Tpv(0)
302:
303:      Dpv(0) = 1.0D-4
304:      if(pr.ge.2) write(6,*) 'Dpv=',Dpv(0)
```

```
305:
306:     Ppv(0) = 1.0D-4
307:     if(pr.ge.2) write(6,*) 'Ppv=',Ppv(0)
308:
309:     Hepv(0) = 1.0D-4
310:     if(pr.ge.2) write(6,*) 'Hepv=',Hepv(0)
311:
312:     emrRsAp = emr * Rs / Area_p
313:
314:     Rt_rem(0) = Tpv(0) * emrRsAp
315:     if(pr.ge.2) write(6,*) 'Rt_rem=',Rt_rem(0)
316:
317:     Rd_rem(0) = Dpv(0) * emrRsAp
318:     if(pr.ge.2) write(6,*) 'Rd_rem=',Rd_rem(0)
319:
320:     Rp_rem(0) = Ppv(0) * emrRsAp
321:     if(pr.ge.2) write(6,*) 'Rp_rem=',Rp_rem(0)
322:
323:     Rhe_rem(0) = Hepv(0) * emrRsAp
324:     if(pr.ge.2) write(6,*) 'Rhe_rem=',Rhe_rem(0)
325:
326:     Tp_reg(0) = Rt_rem(0) * treg
327:     if(pr.ge.2) write(6,*) 'Tp_reg=',Tp_reg(0)
328:
329:     Dp_reg(0) = Rd_rem(0) * treg
330:     if(pr.ge.2) write(6,*) 'Dp_reg=',Dp_reg(0)
331:
332:     Pp_reg(0) = Rp_rem(0) * treg
333:     if(pr.ge.2) write(6,*) 'Pp_reg=',Pp_reg(0)
334:
335:     Hep_reg(0) = Rhe_rem(0) * treg
336:     if(pr.ge.2) write(6,*) 'Hep_reg=',Hep_reg(0)
337:
338:     Tp_tot(0) = Tpv(0) * npump
339:     if(pr.ge.2) write(6,*) 'Tp_tot=',Tp_tot(0)
340:
341:     Dp_tot(0) = Dpv(0) * npump
342:     if(pr.ge.2) write(6,*) 'Dp_tot=',Dp_tot(0)
343:
344:     Pp_tot(0) = Ppv(0) * npump
345:     if(pr.ge.2) write(6,*) 'Pp_tot=',Pp_tot(0)
346:
347:     Hep_tot(0) = Hepv(0) * npump
348:     if(pr.ge.2) write(6,*) 'Hep_tot=',Hep_tot(0)
349:
350:     Tp_tot_m(0) = mu_v * Tp_tot(0)
351:     if(pr.ge.2) write(6,*) 'Tp_tot_m=',Tp_tot_m(0)
352:
353:     Fv_out(0) = ( Rt_rem(0) + Rd_rem(0) + Rp_rem(0) + Rhe_rem(0) ) *
354:    &        npump
355:     if(pr.ge.2) write(6,*) 'Fv_out=',Fv_out(0)
356:
357:     Fpuri_in(0) = Fv_out(0)
358:     if(pr.ge.2) write(6,*) 'Fpuri_in=',Fpuri_in(0)
359:
360:     FF = Rt_rem(0) + Rd_rem(0) + Rp_rem(0)
361:
362:     if ( FF .gt. 0.0d0 ) then
363:
364:     it_(0) = Rt_rem(0) / FF
365:     if(pr.ge.2) write(6,*) 'it_=',it_(0)
366:
367:     id_(0) = Rd_rem(0) / FF
368:     if(pr.ge.2) write(6,*) 'id_=',id_(0)
369:
370:     ip_(0) = Rp_rem(0) / FF
371:     if(pr.ge.2) write(6,*) 'ip_=',ip_(0)
372:
373:     else
374:
375:     it_(0) = 0.0d0
376:     if(pr.ge.2) write(6,*) 'it_=',it_(0)
377:
378:     id_(0) = 0.0d0
379:     if(pr.ge.2) write(6,*) 'id_=',id_(0)
380:
381:     ip_(0) = 0.0d0
382:     if(pr.ge.2) write(6,*) 'ip_=',ip_(0)
383:
384:     endif
385:
386:     Fadj_puri(0) = Fv_out(0)
387:     if(pr.ge.2) write(6,*) 'Fadj_puri=',Fadj_puri(0)
388:
389:     Fpure_(0) = Fv_out(0) * 0.98
390:     if(pr.ge.2) write(6,*) 'Fpure_=',Fpure_(0)
391:
392:     Fimp_(0) = Fv_out(0) * 0.02
393:     if(pr.ge.2) write(6,*) 'Fimp_=',Fimp_(0)
394:
395:     Fcq4_imp(0) = xcq4_ * Fv_out(0) * 0.02
396:     if(pr.ge.2) write(6,*) 'Fcq4_imp=',Fcq4_imp(0)
397:
398:     Fnq3_imp(0) = xnq3_ * Fv_out(0) * 0.02
399:     if(pr.ge.2) write(6,*) 'Fnq3_imp=',Fnq3_imp(0)
400:
401:     Fq2o_imp(0) = xq2o_ * Fv_out(0) * 0.02
402:     if(pr.ge.2) write(6,*) 'Fq2o_imp=',Fq2o_imp(0)
403:
404:     xh_imp(0) = xcq4_ + xnq3_ + xq2o_
405:     if(pr.ge.2) write(6,*) 'xh_imp=',xh_imp(0)
406:
407:     xnhydro_(0) = 1 - 0.0136d0
408:     if(pr.ge.2) write(6,*) 'xnhydro_=',xnhydro_(0)
409:
410:     Fnhydro_imp(0) = ( 1 - xh_imp(0) ) * Fv_out(0) * 0.02
411:     if(pr.ge.2) write(6,*) 'Fnhydro_imp=',Fnhydro_imp(0)
412:
413:     Fhydro_imp_w(0) = w1 * Fcq4_imp(0) + w2 * Fnq3_imp(0) + w3 *
414:    &        Fq2o_imp(0)
415:     if(pr.ge.2) write(6,*) 'Fhydro_imp_w=',Fhydro_imp_w(0)
416:
417:     Ft_pure(0) = Rt_rem(0) * npump - Fhydro_imp_w(0) * it_(0)
418:     if(pr.ge.2) write(6,*) 'Ft_pure=',Ft_pure(0)
```

```
419:
420:      Fd_pure(0) = Rd_rem(0) * npump - Fhydro_imp_w(0) * id_(0)
421:      if(pr.ge.2) write(6,*) 'Fd_pure=',Fd_pure(0)
422:
423:      Fp_pure(0) = Rp_rem(0) * npump - Fhydro_imp_w(0) * ip_(0)
424:      if(pr.ge.2) write(6,*) 'Fp_pure=',Fp_pure(0)
425:*
426:* --- Flow into purification ----------------------------------
427:*
428:
429:      Nimpt(0) = Fv_out(0)
430:      if(pr.ge.2) write(6,*) 'Nimpt=',Nimpt(0)
431:
432:      Nnhydro_imp(0) = Fv_out(0) * 0.02 * 0.32
433:      if(pr.ge.2) write(6,*) 'Nnhydro_imp=',Nnhydro_imp(0)
434:
435:      Ncq4_imp(0) = Fv_out(0) * 0.02 * 0.56
436:      if(pr.ge.2) write(6,*) 'Ncq4_imp=',Ncq4_imp(0)
437:
438:      Nnq3_imp(0) = Fv_out(0) * 0.02 * 0.04
439:      if(pr.ge.2) write(6,*) 'Nnq3_imp=',Nnq3_imp(0)
440:
441:      Nq2o_imp(0) = Fv_out(0) * 0.02 * 0.08
442:      if(pr.ge.2) write(6,*) 'Nq2o_imp=',Nq2o_imp(0)
443:
444:      Timp_(0) = Fv_out(0) / 6
445:      if(pr.ge.2) write(6,*) 'Timp_=',Timp_(0)
446:
447:      Dimp_(0) = Fv_out(0) / 6
448:      if(pr.ge.2) write(6,*) 'Dimp_=',Dimp_(0)
449:
450:      Pimp_(0) = Fv_out(0) / 6
451:      if(pr.ge.2) write(6,*) 'Pimp_=',Pimp_(0)
452:
453:      f_hydro_imp_w(0) = w1 * xcq4_ + w2 * xnq3_ + w3 * xq2o_
454:      if(pr.ge.2) write(6,*) 'f_hydro_imp_w=',f_hydro_imp_w(0)
455:
456:      Timp_m(0) = mu_mob * Timp_(0)
457:      if(pr.ge.2) write(6,*) 'Timp_m=',Timp_m(0)
458:
459:      Nhydro_imp(0) = Timp_(0) + Dimp_(0) + Pimp_(0)
460:      if(pr.ge.2) write(6,*) 'Nhydro_imp=',Nhydro_imp(0)
461:
462:      Fh2_puri_in(0) = ( Ft_pure(0) + Fd_pure(0) + Fp_pure(0) ) / 2
463:      if(pr.ge.2) write(6,*) 'Fh2_puri_in=',Fh2_puri_in(0)
464:
465:      FF = Ft_pure(0) + Fd_pure(0) + Fp_pure(0)
466:
467:      if ( FF .gt. 0.0d0 ) then
468:
469:      Ft2_sub_in(0) = Ft_pure(0) * ( 0.5 + ( xnhydro_(0) + xh_imp(0) )
470:     &        / FF * Fadj_puri(0) * Lhydro_ / Limp_ )
471:      if(pr.ge.2) write(6,*) 'Ft2_sub_in=',Ft2_sub_in(0)
472:
473:      Fd2_sub_in(0) = Fd_pure(0) * ( 0.5 + ( xnhydro_(0) + xh_imp(0) )
474:     &        / FF * Fadj_puri(0) * Lhydro_ / Limp_ )
475:      if(pr.ge.2) write(6,*) 'Fd2_sub_in=',Fd2_sub_in(0)
476:
477:      Fp2_sub_in(0) = Fp_pure(0) * ( 0.5 + ( xnhydro_(0) + xh_imp(0) )
478:     &        / FF * Fadj_puri(0) * Lhydro_ / Limp_ )
479:      if(pr.ge.2) write(6,*) 'Fp2_sub_in=',Fp2_sub_in(0)
480:
481:      else
482:
483:      Ft2_sub_in(0) = 0.0
484:      if(pr.ge.2) write(6,*) 'Ft2_sub_in=',Ft2_sub_in(0)
485:
486:      Fd2_sub_in(0) = 0.0
487:      if(pr.ge.2) write(6,*) 'Fd2_sub_in=',Fd2_sub_in(0)
488:
489:      Fp2_sub_in(0) = 0.0
490:      if(pr.ge.2) write(6,*) 'Fp2_sub_in=',Fp2_sub_in(0)
491:
492:      endif
493:
494:      Fh2_sub_in(0) = Ft2_sub_in(0) + Fd2_sub_in(0) + Fp2_sub_in(0)
495:      if(pr.ge.2) write(6,*) 'Fh2_sub_in=',Fh2_sub_in(0)
496:
497:      if ( Fp2_sub_in(0) .gt. 1.0d-10 ) then
498:
499:      xt_sub(0) = Ft2_sub_in(0) / Fh2_sub_in(0)
500:      if(pr.ge.2) write(6,*) 'xt_sub=',xt_sub(0)
501:
502:      xd_sub(0) = Fd2_sub_in(0) / Fh2_sub_in(0)
503:      if(pr.ge.2) write(6,*) 'xd_sub=',xd_sub(0)
504:
505:      xp_sub(0) = Fp2_sub_in(0) / Fh2_sub_in(0)
506:      if(pr.ge.2) write(6,*) 'xp_sub=',xp_sub(0)
507:
508:      else
509:
510:      xt_sub(0) = 0.0d0
511:      if(pr.ge.2) write(6,*) 'xt_sub=',xt_sub(0)
512:
513:      xd_sub(0) = 0.0d0
514:      if(pr.ge.2) write(6,*) 'xd_sub=',xd_sub(0)
515:
516:      xp_sub(0) = 0.0d0
517:      if(pr.ge.2) write(6,*) 'xp_sub=',xp_sub(0)
518:
519:      endif
520:
521:      if ( abs ( Fh2_sub_in(0) ) .lt. 1.0d-10 ) then
522:
523:      tau_sb = 0.0d0
524:
525:      else
526:
527:      tau_sb = Ns_max / Fh2_sub_in(0)
528:
529:      endif
530:
531:      Ns(0) = Ns_max
532:      if(pr.ge.2) write(6,*) 'Ns=',Ns(0)
```

```
533:
534:     T2_sub(0) = Ns_max / 3
535:     if(pr.ge.2) write(6,*) 'T2_sub=',T2_sub(0)
536:
537:     D2_sub(0) = Ns_max / 3
538:     if(pr.ge.2) write(6,*) 'D2_sub=',D2_sub(0)
539:
540:     P2_sub(0) = Ns_max / 3
541:     if(pr.ge.2) write(6,*) 'P2_sub=',P2_sub(0)
542:
543:     Nhydro_sub(0) = T2_sub(0) + D2_sub(0) + P2_sub(0)
544:     if(pr.ge.2) write(6,*) 'Nhydro_sub=',Nhydro_sub(0)
545:
546:     if ( abs ( tau_sb ) .ge. 1.0d-10 ) then
547:
548:     Ft_sub_out(0) = 2.0 * ( Ft2_sub_in(0) + xt_sub(0) * Ns(0) /
549:    &       tau_sb )
550:     if(pr.ge.2) write(6,*) 'Ft_sub_out=',Ft_sub_out(0)
551:
552:     Fd_sub_out(0) = 2.0 * ( Fd2_sub_in(0) + xd_sub(0) * Ns(0) /
553:    &       tau_sb )
554:     if(pr.ge.2) write(6,*) 'Fd_sub_out=',Fd_sub_out(0)
555:
556:     Fp_sub_out(0) = 2.0 * ( Fp2_sub_in(0) + xp_sub(0) * Ns(0) /
557:    &       tau_sb )
558:     if(pr.ge.2) write(6,*) 'Fp_sub_out=',Fp_sub_out(0)
559:
560:     else
561:
562:     Ft_sub_out(0) = 0.0d0
563:     if(pr.ge.2) write(6,*) 'Ft_sub_out=',Ft_sub_out(0)
564:
565:     Fd_sub_out(0) = 0.0d0
566:     if(pr.ge.2) write(6,*) 'Fd_sub_out=',Fd_sub_out(0)
567:
568:     Fp_sub_out(0) = 0.0d0
569:     if(pr.ge.2) write(6,*) 'Fp_sub_out=',Fp_sub_out(0)
570:
571:     endif
572:
573:     Fh_sub_out(0) = Ft_sub_out(0) + Fd_sub_out(0) + Fp_sub_out(0)
574:     if(pr.ge.2) write(6,*) 'Fh_sub_out=',Fh_sub_out(0)
575:
576:     if ( tpoint_cycle .lt. treg_msieve ) then
577:
578:     Treg_out(0) = Timp_(j) * ( 1 - tpoint_cycle / treg_msieve )
579:     if(pr.ge.2) write(6,*) 'Treg_out=',Treg_out(0)
580:
581:     else
582:
583:     Treg_out(0) = 0D0
584:     if(pr.ge.2) write(6,*) 'Treg_out=',Treg_out(0)
585:
586:     endif
587:
588:     Treg_out_m(0) = mu_reg_m * Treg_out(0)
589:     if(pr.ge.2) write(6,*) 'Treg_out_m=',Treg_out_m(0)
590:* Evaluation of Hydrogen-containing Species:
591:
592:     Ncq4_reg(0) = Ncq4_imp(j)
593:     if(pr.ge.2) write(6,*) 'Ncq4_reg=',Ncq4_reg(0)
594:
595:     Nnq3_reg(0) = Nnq3_imp(j)
596:     if(pr.ge.2) write(6,*) 'Nnq3_reg=',Nnq3_reg(0)
597:
598:     Nq2o_reg(0) = Nq2o_imp(j)
599:     if(pr.ge.2) write(6,*) 'Nq2o_reg=',Nq2o_reg(0)
600:
601:     Ecq4_reg(0) = Ncq4_reg(0) / treg_msieve
602:     if(pr.ge.2) write(6,*) 'Ecq4_reg=',Ecq4_reg(0)
603:
604:     Enq3_reg(0) = Nnq3_reg(0) / treg_msieve
605:     if(pr.ge.2) write(6,*) 'Enq3_reg=',Enq3_reg(0)
606:
607:     Eq2o_reg(0) = Nq2o_reg(0) / treg_msieve
608:     if(pr.ge.2) write(6,*) 'Eq2o_reg=',Eq2o_reg(0)
609:
610:     Fhydro_pmem(0) = w1 * Ecq4_reg(0) + w2 * Enq3_reg(0) + w3 *
611:    &       Eq2o_reg(0)
612:     if(pr.ge.2) write(6,*) 'Fhydro_pmem=',Fhydro_pmem(0)
613:
614:     if ( Nhydro_imp(j) .ne. 0.0d0 ) then
615:
616:     it_endj(0) = Timp_(j) / Nhydro_imp(j)
617:     if(pr.ge.2) write(6,*) 'it_endj=',it_endj(0)
618:
619:     id_endj(0) = Pimp_(j) / Nhydro_imp(j)
620:     if(pr.ge.2) write(6,*) 'id_endj=',id_endj(0)
621:
622:     ip_endj(0) = Timp_(j) / Nhydro_imp(j)
623:     if(pr.ge.2) write(6,*) 'ip_endj=',ip_endj(0)
624:
625:     else
626:
627:     it_endj(0) = 0.0D0
628:     if(pr.ge.2) write(6,*) 'it_endj=',it_endj(0)
629:
630:     id_endj(0) = 0.0D0
631:     if(pr.ge.2) write(6,*) 'id_endj=',id_endj(0)
632:
633:     ip_endj(0) = 0.0D0
634:     if(pr.ge.2) write(6,*) 'ip_endj=',ip_endj(0)
635:
636:     endif
637:
638:     Tpmem(0) = 0.0D0
639:     if(pr.ge.2) write(6,*) 'Tpmem=',Tpmem(0)
640:
641:     Dpmem(0) = 0.0D0
642:     if(pr.ge.2) write(6,*) 'Dpmem=',Dpmem(0)
643:
644:     Ppmem(0) = 0.0D0
645:     if(pr.ge.2) write(6,*) 'Ppmem=',Ppmem(0)
646:
```

```
647:     Ft_puri_out(0) = Ft_sub_out(0) + Tpmem(0) / tau_pmem
648:     if(pr.ge.2) write(6,*) 'Ft_puri_out=',Ft_puri_out(0)
649:
650:     Fd_puri_out(0) = Fd_sub_out(0) + Dpmem(0) / tau_pmem
651:     if(pr.ge.2) write(6,*) 'Fd_puri_out=',Fd_puri_out(0)
652:
653:     Fp_puri_out(0) = Fp_sub_out(0) + Ppmem(0) / tau_pmem
654:     if(pr.ge.2) write(6,*) 'Fp_puri_out=',Fp_puri_out(0)
655:
656:     Fpuri_out(0) = Ft_puri_out(0) + Fd_puri_out(0) + Fp_puri_out(0)
657:     if(pr.ge.2) write(6,*) 'Fpuri_out=',Fpuri_out(0)
658:*
659:* ... Fiss_in: assuming Fragnb = Fragss_inj = 0
660:
661:     Fiss_in(0) = Fbl_cool + Fbl_ + ( 1 - h1 - h3 ) * Fpuri_out(0)
662:     if(pr.ge.2) write(6,*) 'Fiss_in=',Fiss_in(0)
663:
664:     Niss_(0) = Fiss_in(0)
665:     if(pr.ge.2) write(6,*) 'Niss_=',Niss_(0)
666:*Tiss_    = Niss_ / 3.0D0 ;
667:*Diss_    = Niss_ / 3.0D0 ;
668:
669:     Tiss_(0) = Niss_(0) * 3.0 / ( 3.0 + 3.0 + 1.0 )
670:     if(pr.ge.2) write(6,*) 'Tiss_=',Tiss_(0)
671:
672:     Diss_(0) = Niss_(0) * 3.0 / ( 3.0 + 3.0 + 1.0 )
673:     if(pr.ge.2) write(6,*) 'Diss_=',Diss_(0)
674:
675:     if ( Niss_(0) .gt. 0.0 ) then
676:
677:     Xiss_t(0) = Tiss_(0) / Niss_(0)
678:     if(pr.ge.2) write(6,*) 'Xiss_t=',Xiss_t(0)
679:
680:     Xiss_d(0) = Diss_(0) / Niss_(0)
681:     if(pr.ge.2) write(6,*) 'Xiss_d=',Xiss_d(0)
682:
683:     Xiss_p(0) = 1.0d0 - Xiss_t(0) - Xiss_d(0)
684:     if(pr.ge.2) write(6,*) 'Xiss_p=',Xiss_p(0)
685:
686:     else
687:
688:     Xiss_t(0) = 0.0d0
689:     if(pr.ge.2) write(6,*) 'Xiss_t=',Xiss_t(0)
690:
691:     Xiss_d(0) = 0.0d0
692:     if(pr.ge.2) write(6,*) 'Xiss_d=',Xiss_d(0)
693:
694:     Xiss_p(0) = 0.0d0
695:     if(pr.ge.2) write(6,*) 'Xiss_p=',Xiss_p(0)
696:
697:     endif
698:**Piss_    = Niss_ / 3.0D0 ;
699:
700:     Piss_(0) = Xiss_p(0) * Niss_(0)
701:     if(pr.ge.2) write(6,*) 'Piss_=',Piss_(0)
702:
703:     Tiss_m(0) = mu_iss_ * Tiss_(0)
704:     if(pr.ge.2) write(6,*) 'Tiss_m=',Tiss_m(0)
705:
706:     P1 = 1.0 / ( Xiss_mid_t * Xiss_bot_d + Xiss_bot_t * Xiss_top_d +
707:    &       Xiss_top_t * Xiss_mid_d - Xiss_mid_t * Xiss_top_d -
708:    &       Xiss_top_t * Xiss_bot_d - Xiss_bot_t * Xiss_mid_d )
709:
710:     Fiss_top_(0) = P1 * ( ( Xiss_mid_t * Xiss_bot_d - Xiss_bot_t *
711:    &       Xiss_mid_d ) * Niss_(0) / tau_iss_ + ( Xiss_mid_d -
712:    &       Xiss_bot_d ) * Tiss_(0) / tau_iss_ + ( Xiss_bot_t -
713:    &       Xiss_mid_t ) * Diss_(0) / tau_iss_ )
714:     if(pr.ge.2) write(6,*) 'Fiss_top_=',Fiss_top_(0)
715:
716:     Fiss_mid_(0) = P1 * ( ( Xiss_bot_t * Xiss_top_d - Xiss_top_t *
717:    &       Xiss_bot_d ) * Niss_(0) / tau_iss_ + ( Xiss_bot_d -
718:    &       Xiss_top_d ) * Tiss_(0) / tau_iss_ + ( Xiss_top_t -
719:    &       Xiss_bot_t ) * Diss_(0) / tau_iss_ )
720:     if(pr.ge.2) write(6,*) 'Fiss_mid_=',Fiss_mid_(0)
721:
722:     Fiss_bot_(0) = P1 * ( ( Xiss_top_t * Xiss_mid_d - Xiss_mid_t *
723:    &       Xiss_top_d ) * Niss_(0) / tau_iss_ + ( Xiss_top_d -
724:    &       Xiss_mid_d ) * Tiss_(0) / tau_iss_ + ( Xiss_mid_t -
725:    &       Xiss_top_t ) * Diss_(0) / tau_iss_ )
726:     if(pr.ge.2) write(6,*) 'Fiss_bot_=',Fiss_bot_(0)
727:*
728:* --- Pellet injector ----------------------------------------
729:*
730:
731:     Fsto_out_t(0) = Fsto_tri_out(0) / 2
732:     if(pr.ge.2) write(6,*) 'Fsto_out_t=',Fsto_out_t(0)
733:
734:     Fsto_out_d(0) = Fsto_deu_out(0) / 4
735:     if(pr.ge.2) write(6,*) 'Fsto_out_d=',Fsto_out_d(0)
736:
737:     Next_12(0) = Fplas_in_t(0) / 2
738:     if(pr.ge.2) write(6,*) 'Next_12=',Next_12(0)
739:
740:     Text_12(0) = Next_12(0) / 3
741:     if(pr.ge.2) write(6,*) 'Text_12=',Text_12(0)
742:
743:     Dext_12(0) = Next_12(0) / 3
744:     if(pr.ge.2) write(6,*) 'Dext_12=',Dext_12(0)
745:
746:     Pext_12(0) = Next_12(0) / 3
747:     if(pr.ge.2) write(6,*) 'Pext_12=',Pext_12(0)
748:
749:     Xext_12_t(0) = Text_12(0) / Next_12(0)
750:     if(pr.ge.2) write(6,*) 'Xext_12_t=',Xext_12_t(0)
751:
752:     Xext_12_d(0) = Dext_12(0) / Next_12(0)
753:     if(pr.ge.2) write(6,*) 'Xext_12_d=',Xext_12_d(0)
754:
755:     Xext_12_p(0) = 1 - Xext_12_t(0) - Xext_12_d(0)
756:     if(pr.ge.2) write(6,*) 'Xext_12_p=',Xext_12_p(0)
757:
758:     Next_tot(0) = Next_12(0) * 2.0
759:     if(pr.ge.2) write(6,*) 'Next_tot=',Next_tot(0)
760:
```

```
761:      Text_tot(0) = Text_12(0) * 2.0
762:      if(pr.ge.2) write(6,*) 'Text_tot=',Text_tot(0)
763:
764:      Dext_tot(0) = Dext_12(0) * 2.0
765:      if(pr.ge.2) write(6,*) 'Dext_tot=',Dext_tot(0)
766:
767:      Pext_tot(0) = Pext_12(0) * 2.0
768:      if(pr.ge.2) write(6,*) 'Pext_tot=',Pext_tot(0)
769:
770:      Text_m(0) = mu_ext * Text_tot(0)
771:      if(pr.ge.2) write(6,*) 'Text_m=',Text_m(0)
772:
773:      Nextru_(0) = Next_12(0)
774:      if(pr.ge.2) write(6,*) 'Nextru_=',Nextru_(0)
775:
776:      Xextru_t(0) = Text_12(0) / Next_12(0)
777:      if(pr.ge.2) write(6,*) 'Xextru_t=',Xextru_t(0)
778:
779:      Xextru_d(0) = Dext_12(0) / Next_12(0)
780:      if(pr.ge.2) write(6,*) 'Xextru_d=',Xextru_d(0)
781:
782:      Xextru_p(0) = 1.0d0 - Xextru_t(0) - Xextru_d(0)
783:      if(pr.ge.2) write(6,*) 'Xextru_p=',Xextru_p(0)
784:
785:      Vinj_t(0) = Xextru_t(0) * Nextru_(0) * 18.894D-6 / 2
786:      if(pr.ge.2) write(6,*) 'Vinj_t=',Vinj_t(0)
787:
788:      Vinj_d(0) = Xextru_d(0) * Nextru_(0) * 20.085D-6 / 2
789:      if(pr.ge.2) write(6,*) 'Vinj_d=',Vinj_d(0)
790:
791:      Vinj_p(0) = Xextru_p(0) * Nextru_(0) * 23.326D-6 / 2
792:      if(pr.ge.2) write(6,*) 'Vinj_p=',Vinj_p(0)
793:* total volume in extruder
794:
795:      Vinj_tot(0) = Vinj_t(0) + Vinj_d(0) + Vinj_p(0)
796:      if(pr.ge.2) write(6,*) 'Vinj_tot=',Vinj_tot(0)
797:
798:      npellet_t(0) = Vinj_t(0) / Vinj_tot(0) * Vpellet * 2 * kpellet_t
799:      if(pr.ge.2) write(6,*) 'npellet_t=',npellet_t(0)
800:
801:      npellet_d(0) = Vinj_d(0) / Vinj_tot(0) * Vpellet * 2 * kpellet_d
802:      if(pr.ge.2) write(6,*) 'npellet_d=',npellet_d(0)
803:
804:      npellet_p(0) = Vinj_p(0) / Vinj_tot(0) * Vpellet * 2 * kpellet_p
805:      if(pr.ge.2) write(6,*) 'npellet_p=',npellet_p(0)
806:
807:      Fbk_plas_(0) = 3 * 5.98D-22 * nedge_tot
808:      if(pr.ge.2) write(6,*) 'Fbk_plas_=',Fbk_plas_(0)
809:
810:      Fbk_plas_t(0) = 3 * 5.98D-22 * nedge_t
811:      if(pr.ge.2) write(6,*) 'Fbk_plas_t=',Fbk_plas_t(0)
812:
813:      Fbk_plas_d(0) = 3 * 5.98D-22 * nedge_d
814:      if(pr.ge.2) write(6,*) 'Fbk_plas_d=',Fbk_plas_d(0)
815:
816:      Fbk_plas_p(0) = 3 * 5.98D-22 * nedge_p
817:      if(pr.ge.2) write(6,*) 'Fbk_plas_p=',Fbk_plas_p(0)
818:
819:      Finj_prop(0) = Fsto_out_p(0) + liss_p * Fiss_top_(0)
820:      if(pr.ge.2) write(6,*) 'Finj_prop=',Finj_prop(0)
821:
822:      Rinj_(0) = Finj_prop(0) / nprop
823:      if(pr.ge.2) write(6,*) 'Rinj_=',Rinj_(0)
824:
825:      Fplas_in_p(0) = Rinj_(0) * npellet_p(0) * ( 1 - finj_err )
826:      if(pr.ge.2) write(6,*) 'Fplas_in_p=',Fplas_in_p(0)
827:
828:      Fsto_pro_out_p(0) = Fsto_out_p(0) * Xsto_pro_p(0)
829:      if(pr.ge.2) write(6,*) 'Fsto_pro_out_p=',Fsto_pro_out_p(0)
830:
831:      Fplas_prop(0) = ( 1 - fcap_prop ) * Finj_prop(0)
832:      if(pr.ge.2) write(6,*) 'Fplas_prop=',Fplas_prop(0)
833:*
834:*Fuel_p = Fplas_in_p + Xgas_p * Fgas_tot + Fplas_prop_p ;
835:*Fex_p = Reac_p + Fuel_p;
836:*
837:
838:      Xprop_t(0) = ( Fsto_out_p(0) * Xsto_pro_t(0) + liss_p *
839:     &       Fiss_top_(0) * Xiss_top_t ) / Finj_prop(0)
840:      if(pr.ge.2) write(6,*) 'Xprop_t=',Xprop_t(0)
841:
842:      Xprop_d(0) = ( Fsto_out_p(0) * Xsto_pro_d(0) + liss_p *
843:     &       Fiss_top_(0) * Xiss_top_d ) / Finj_prop(0)
844:      if(pr.ge.2) write(6,*) 'Xprop_d=',Xprop_d(0)
845:
846:      Xprop_p(0) = 1 - Xprop_t(0) - Xprop_d(0)
847:      if(pr.ge.2) write(6,*) 'Xprop_p=',Xprop_p(0)
848:
849:      Fplas_prop_t(0) = Xprop_t(0) * Fplas_prop(0)
850:      if(pr.ge.2) write(6,*) 'Fplas_prop_t=',Fplas_prop_t(0)
851:
852:      Fplas_prop_d(0) = Xprop_d(0) * Fplas_prop(0)
853:      if(pr.ge.2) write(6,*) 'Fplas_prop_d=',Fplas_prop_d(0)
854:
855:      Fplas_prop_p(0) = Xprop_p(0) * Fplas_prop(0)
856:      if(pr.ge.2) write(6,*) 'Fplas_prop_p=',Fplas_prop_p(0)
857:
858:      Finj_in_tot(0) = Fplas_in_t(0) + Fplas_in_d(0) + Fplas_in_p(0)
859:      if(pr.ge.2) write(6,*) 'Finj_in_tot=',Finj_in_tot(0)
860:
861:      Finj_exh(0) = ( ( Fplas_in_t(0) + Fplas_in_d(0) ) / ( 1 -
862:     &       finj_err ) + Fplas_in_p(0) ) * finj_err + Finj_prop(0) *
863:     &       fcap_prop + Fbk_plas_(0)
864:      if(pr.ge.2) write(6,*) 'Finj_exh=',Finj_exh(0)
865:
866:      Finj_exh_t(0) = Fplas_in_t(0) * finj_err / ( 1 - finj_err ) +
867:     &       Finj_prop(0) * Xprop_t(0) * fcap_prop + Fbk_plas_t(0)
868:      if(pr.ge.2) write(6,*) 'Finj_exh_t=',Finj_exh_t(0)
869:
870:      Finj_exh_d(0) = Fplas_in_d(0) * finj_err / ( 1 - finj_err ) +
871:     &       Finj_prop(0) * Xprop_d(0) * fcap_prop + Fbk_plas_d(0)
872:      if(pr.ge.2) write(6,*) 'Finj_exh_d=',Finj_exh_d(0)
873:
874:      Fiss_inj(0) = ( 1 - fnon_iss ) * Finj_exh(0)
```

```
875:      if(pr.ge.2) write(6,*) 'Fiss_inj=',Fiss_inj(0)
876:
877:      Finj_rec(0) = fnon_iss * Finj_exh(0)
878:      if(pr.ge.2) write(6,*) 'Finj_rec=',Finj_rec(0)
879:
880:      Xinj_t(0) = Finj_exh_t(0) / Finj_exh(0)
881:      if(pr.ge.2) write(6,*) 'Xinj_t=',Xinj_t(0)
882:
883:      Xinj_d(0) = Finj_exh_d(0) / Finj_exh(0)
884:      if(pr.ge.2) write(6,*) 'Xinj_d=',Xinj_d(0)
885:
886:      Xinj_p(0) = 1 - Xinj_t(0) - Xinj_d(0)
887:      if(pr.ge.2) write(6,*) 'Xinj_p=',Xinj_p(0)
888:
889:      Fgas_in_tri(0) = Fsto_tri_out(0) / 2
890:      if(pr.ge.2) write(6,*) 'Fgas_in_tri=',Fgas_in_tri(0)
891:*Fgas_d = Fgas_in_tri(t) * Xsto_tri_d + Fgas_in_deu(t) * Xsto_deu_d
892:*      + hl * Fpuri_out(t0) * Xiss_in_d
893:*      + mgas_mid_iss * Fiss_mid_(t0) * Xiss_mid_d
894:*      + mgas_bot_iss * Fiss_bot_(t0) * Xiss_bot_d ;
895:
896:      Ngas_tot(0) = Burn_t(0) / 6.0
897:      if(pr.ge.2) write(6,*) 'Ngas_tot=',Ngas_tot(0)
898:
899:      Tgas_(0) = Burn_t(0) / 18.0
900:      if(pr.ge.2) write(6,*) 'Tgas_=',Tgas_(0)
901:
902:      Tgas_mob(0) = mu_gas * Tgas_(0)
903:      if(pr.ge.2) write(6,*) 'Tgas_mob=',Tgas_mob(0)
904:
905:      Dgas_(0) = Burn_t(0) / 18.0
906:      if(pr.ge.2) write(6,*) 'Dgas_=',Dgas_(0)
907:
908:      Xgas_t(0) = Tgas_(0) / Ngas_tot(0)
909:      if(pr.ge.2) write(6,*) 'Xgas_t=',Xgas_t(0)
910:
911:      Xgas_d(0) = Dgas_(0) / Ngas_tot(0)
912:      if(pr.ge.2) write(6,*) 'Xgas_d=',Xgas_d(0)
913:
914:      Xgas_p(0) = 1 - Xgas_t(0) - Xgas_d(0)
915:      if(pr.ge.2) write(6,*) 'Xgas_p=',Xgas_p(0)
916:
917:      Pgas_(0) = Xgas_p(0) * Ngas_tot(0)
918:      if(pr.ge.2) write(6,*) 'Pgas_=',Pgas_(0)
919:*Fgas_tot = (Fgas_t(t)+Fgas_d(t)) / (Xgas_t(t)+Xgas_d(t)) ;
920:*Fpuff_plas = Fgas_tot + Fplas_prop_t + Fplas_prop_d + Fplas_prop_p ;
921:*Fgas_d = Fgas_in_tri(t) * Xsto_tri_d + Fgas_in_deu(t) * Xsto_deu_d
922:*      + hl * Fpuri_out(t0) * Xiss_in_d
923:*      + mgas_mid_iss * Fiss_mid_(t0) * Xiss_mid_d
924:*      + mgas_bot_iss * Fiss_bot_(t0) * Xiss_bot_d ;
925:*Fgas_tot = (Fgas_t(t)+Fgas_d(t)) / (Xgas_t(t)+Xgas_d(t)) ;
926:*Fpuff_plas = Fgas_tot + Fplas_prop_t + Fplas_prop_d + Fplas_prop_p ;
927:*
928:*  --- Isotope Separation System ------------------------------
929:*
930:*
931:*
932:
933:      Fiss_in_t(0) = Fbl_cool * Xc_t + Fbl_ * Xbl_t
934:      if(pr.ge.2) write(6,*) 'Fiss_in_t=',Fiss_in_t(0)
935:
936:      Fiss_in_d(0) = Fbl_cool * Xc_d + Fbl_ * Xbl_d
937:      if(pr.ge.2) write(6,*) 'Fiss_in_d=',Fiss_in_d(0)
938:
939:      Fiss_in_p(0) = Fiss_in(0) - Fiss_in_t(0) - Fiss_in_d(0)
940:      if(pr.ge.2) write(6,*) 'Fiss_in_p=',Fiss_in_p(0)
941:
942:      if ( Fiss_in(0) .ne. 0.0d0 ) then
943:
944:      Xiss_in_t(0) = Fiss_in_t(0) / Fiss_in(0)
945:      if(pr.ge.2) write(6,*) 'Xiss_in_t=',Xiss_in_t(0)
946:
947:      Xiss_in_d(0) = Fiss_in_d(0) / Fiss_in(0)
948:      if(pr.ge.2) write(6,*) 'Xiss_in_d=',Xiss_in_d(0)
949:
950:      Xiss_in_p(0) = 1 - Xiss_in_t(0) - Xiss_in_d(0)
951:      if(pr.ge.2) write(6,*) 'Xiss_in_p=',Xiss_in_p(0)
952:
953:      else
954:
955:      Xiss_in_t(0) = 0.0d0
956:      if(pr.ge.2) write(6,*) 'Xiss_in_t=',Xiss_in_t(0)
957:
958:      Xiss_in_d(0) = 0.0d0
959:      if(pr.ge.2) write(6,*) 'Xiss_in_d=',Xiss_in_d(0)
960:
961:      Xiss_in_p(0) = 0.0d0
962:      if(pr.ge.2) write(6,*) 'Xiss_in_p=',Xiss_in_p(0)
963:
964:      endif
965:* --- Fgas_d / Fgas_tot / Fpuff_plas are here bacause Fgas_d use
966:*     Xiss_in_d
967:
968:      Fgas_d(0) = Fgas_in_tri(0) * Xsto_tri_d(0) + Fgas_in_deu(0) *
969:     &       Xsto_deu_d(0) + hl * Fpuri_out(0) * Xiss_in_d(0) +
970:     &       mgas_mid_iss * Fiss_mid_(0) * Xiss_mid_d + mgas_bot_iss *
971:     &       Fiss_bot_(0) * Xiss_bot_d
972:      if(pr.ge.2) write(6,*) 'Fgas_d=',Fgas_d(0)
973:
974:      Fgas_tot(0) = ( Fgas_t(0) + Fgas_d(0) ) / ( Xgas_t(0) + Xgas_d(0)
975:     &       )
976:      if(pr.ge.2) write(6,*) 'Fgas_tot=',Fgas_tot(0)
977:
978:      Fpuff_plas(0) = Fgas_tot(0) + Fplas_prop_t(0) + Fplas_prop_d(0) +
979:     &       Fplas_prop_p(0)
980:      if(pr.ge.2) write(6,*) 'Fpuff_plas=',Fpuff_plas(0)
981:*
982:*  --- Neutral beam injector ------------------------------
983:*
984:
985:      W_nb(0) = 75D0 * fppower ( 0.0d0 , t_burn , t_p_ru , t_p_rd )
986:      if(pr.ge.2) write(6,*) 'W_nb=',W_nb(0)
987:
988:      Fplas_nb_d(0) = 3.6D3 / 1.6D0 * 1.0D19 * 1 / No * W_nb(0) / E_nb
```

```
989:      if(pr.ge.2) write(6,*) 'Fplas_nb_d=',Fplas_nb_d(0)
990:**Fin_st_d     = Fsto_deu_out / 2.0D0 ;
991:
992:      Fin_st_d(0) = 0.0D0
993:      if(pr.ge.2) write(6,*) 'Fin_st_d=',Fin_st_d(0)
994:
995:      Ncryo_(0) = Burn_d(0) / 2.0D0
996:      if(pr.ge.2) write(6,*) 'Ncryo_=',Ncryo_(0)
997:
998:      Tnb_cryo(0) = Ncryo_(0) / 3.0D0
999:      if(pr.ge.2) write(6,*) 'Tnb_cryo=',Tnb_cryo(0)
1000:
1001:     Dnb_cryo(0) = Ncryo_(0) / 3.0D0
1002:     if(pr.ge.2) write(6,*) 'Dnb_cryo=',Dnb_cryo(0)
1003:
1004:     Xnb_cryo_t(0) = Tnb_cryo(0) / Ncryo_(0)
1005:     if(pr.ge.2) write(6,*) 'Xnb_cryo_t=',Xnb_cryo_t(0)
1006:
1007:     Xnb_cryo_d(0) = Dnb_cryo(0) / Ncryo_(0)
1008:     if(pr.ge.2) write(6,*) 'Xnb_cryo_d=',Xnb_cryo_d(0)
1009:
1010:     Xnb_cryo_p(0) = 1.0 - Xnb_cryo_d(0) - Xnb_cryo_t(0)
1011:     if(pr.ge.2) write(6,*) 'Xnb_cryo_p=',Xnb_cryo_p(0)
1012:
1013:     Pnb_cryo(0) = Xnb_cryo_p(0) * Ncryo_(0)
1014:     if(pr.ge.2) write(6,*) 'Pnb_cryo=',Pnb_cryo(0)
1015:
1016:     Rnb_reg_p(0) = Pnb_cryo(0) * emrRsAp
1017:     if(pr.ge.2) write(6,*) 'Rnb_reg_p=',Rnb_reg_p(0)
1018:
1019:     tnb_(0) = t - nnb_ * tnb_pump
1020:     if(pr.ge.2) write(6,*) 'tnb_=',tnb_(0)
1021:
1022:     if ( tnb_(0) .gt. tnb_reg .and. tnb_(0) .lt. tnb_pump ) then
1023:
1024:     Rnb_reg_t(0) = 0.0d0
1025:     if(pr.ge.2) write(6,*) 'Rnb_reg_t=',Rnb_reg_t(0)
1026:
1027:     Rnb_reg_d(0) = 0.0d0
1028:     if(pr.ge.2) write(6,*) 'Rnb_reg_d=',Rnb_reg_d(0)
1029:
1030:     else
1031:
1032:     Rnb_reg_t(0) = Tnb_cryo(0) * emrRsAp
1033:     if(pr.ge.2) write(6,*) 'Rnb_reg_t=',Rnb_reg_t(0)
1034:
1035:     Rnb_reg_d(0) = Dnb_cryo(0) * emrRsAp
1036:     if(pr.ge.2) write(6,*) 'Rnb_reg_d=',Rnb_reg_d(0)
1037:
1038:     endif
1039:
1040:     Rnb_reg_(0) = Ncryo_(0) * emrRsAp
1041:     if(pr.ge.2) write(6,*) 'Rnb_reg_=',Rnb_reg_(0)
1042:
1043:     Fnb_cryo(0) = Rnb_reg_t(0) + Rnb_reg_d(0) + Rnb_reg_p(0)
1044:     if(pr.ge.2) write(6,*) 'Fnb_cryo=',Fnb_cryo(0)
1045:
1046:     Crec(0) = 0.0D0
1047:     if(pr.ge.2) write(6,*) 'Crec=',Crec(0)
1048:
1049:     Fin_rec_(0) = Crec(0) * Fnb_cryo(0)
1050:     if(pr.ge.2) write(6,*) 'Fin_rec_=',Fin_rec_(0)
1051:
1052:     Fiss_nb(0) = ( 1.0D0 - Crec(0) ) * Fnb_cryo(0)
1053:     if(pr.ge.2) write(6,*) 'Fiss_nb=',Fiss_nb(0)
1054:
1055:     if ( tnb_(0) .gt. 0.0d0 .and. tnb_(0) .lt. tnb_reg ) then
1056:
1057:     Tnb_reg_(0) = Rnb_reg_t(0) * tnb_reg
1058:     if(pr.ge.2) write(6,*) 'Tnb_reg_=',Tnb_reg_(0)
1059:
1060:     Dnb_reg_(0) = Rnb_reg_d(0) * tnb_reg
1061:     if(pr.ge.2) write(6,*) 'Dnb_reg_=',Dnb_reg_(0)
1062:
1063:     Dnb_cryo_tot(0) = Dnb_cryo(0) + Dnb_reg_(0) * ( 1.0 - tnb_(0) /
1064:    &       tnb_reg )
1065:     if(pr.ge.2) write(6,*) 'Dnb_cryo_tot=',Dnb_cryo_tot(0)
1066:
1067:     elseif ( tnb_(0) .gt. tnb_reg .and. tnb_(0) .lt. tnb_pump ) then
1068:
1069:     Tnb_reg_(0) = 0.0d0
1070:     if(pr.ge.2) write(6,*) 'Tnb_reg_=',Tnb_reg_(0)
1071:
1072:     Dnb_reg_(0) = 0.0d0
1073:     if(pr.ge.2) write(6,*) 'Dnb_reg_=',Dnb_reg_(0)
1074:
1075:     Dnb_cryo_tot(0) = Dnb_cryo(0)
1076:     if(pr.ge.2) write(6,*) 'Dnb_cryo_tot=',Dnb_cryo_tot(0)
1077:
1078:     endif
1079:
1080:     if ( tnb_(0) .lt. tnb_reg ) then
1081:
1082:     Tnb_cryo_tot(0) = Tnb_cryo(0) + Tnb_reg_(0) * ( 1.0 - tnb_(0) /
1083:    &       tnb_reg )
1084:     if(pr.ge.2) write(6,*) 'Tnb_cryo_tot=',Tnb_cryo_tot(0)
1085:
1086:     elseif ( tnb_(0) .gt. tnb_reg .and. tnb_(0) .lt. tnb_pump ) then
1087:
1088:     Tnb_cryo_tot(0) = Tnb_cryo(0)
1089:     if(pr.ge.2) write(6,*) 'Tnb_cryo_tot=',Tnb_cryo_tot(0)
1090:
1091:     endif
1092:
1093:     Tnb_cryo_mob(0) = mu_nb_mob * Tnb_cryo_tot(0)
1094:     if(pr.ge.2) write(6,*) 'Tnb_cryo_mob=',Tnb_cryo_mob(0)
1095:
1096:     Fbk_nb_(0) = 3 * 5.98D-21 * nedge_tot
1097:     if(pr.ge.2) write(6,*) 'Fbk_nb_=',Fbk_nb_(0)
1098:
1099:     Fbk_nb_t(0) = 3 * 5.98D-21 * nedge_t
1100:     if(pr.ge.2) write(6,*) 'Fbk_nb_t=',Fbk_nb_t(0)
1101:
1102:     Fbk_nb_d(0) = 3 * 5.98D-21 * nedge_d
```

```
1103:     if(pr.ge.2) write(6,*) 'Fbk_nb_d=',Fbk_nb_d(0)
1104:
1105:     Tnb_wall(0) = 1.0D-4
1106:     if(pr.ge.2) write(6,*) 'Tnb_wall=',Tnb_wall(0)
1107:
1108:     Tnb_wall_mob(0) = mu_wall_ * Tnb_wall(0)
1109:     if(pr.ge.2) write(6,*) 'Tnb_wall_mob=',Tnb_wall_mob(0)
1110:
1111:     Tnb_tot(0) = Tnb_wall(0) + Tnb_cryo_tot(0)
1112:     if(pr.ge.2) write(6,*) 'Tnb_tot=',Tnb_tot(0)
1113:
1114:     Tnb_tot_mob(0) = Tnb_wall_mob(0) + Tnb_cryo_mob(0)
1115:     if(pr.ge.2) write(6,*) 'Tnb_tot_mob=',Tnb_tot_mob(0)
1116:
1117:     Fplas_nb_(0) = ( kiss_mid_ * Fiss_mid_(0) + Fin_st_d(0) ) *
1118:    &       eta_nb_tot
1119:     if(pr.ge.2) write(6,*) 'Fplas_nb_=',Fplas_nb_(0)
1120:
1121:     Fplas_nb_t(0) = kiss_mid_ * Fiss_mid_(0) * eta_nb_tot
1122:     if(pr.ge.2) write(6,*) 'Fplas_nb_t=',Fplas_nb_t(0)
1123:
1124:     Fplas_nb_p(0) = Fplas_nb_(0) - Fplas_nb_t(0) - Fplas_nb_d(0)
1125:     if(pr.ge.2) write(6,*) 'Fplas_nb_p=',Fplas_nb_p(0)
1126:*
1127:* --- the following variables are set here ---
1128:*
1129:
1130:     Fuel_d(0) = Fplas_in_d(0) + Fgas_d(0) + Fplas_nb_d(0) +
1131:    &       Fplas_prop_d(0)
1132:     if(pr.ge.2) write(6,*) 'Fuel_d=',Fuel_d(0)
1133:
1134:     Fex_d(0) = Fuel_d(0) - Burn_d(0)
1135:     if(pr.ge.2) write(6,*) 'Fex_d=',Fex_d(0)
1136:
1137:     Fuel_p(0) = Fplas_in_p(0) + Xgas_p(0) * Fgas_tot(0) +
1138:    &       Fplas_prop_p(0)
1139:     if(pr.ge.2) write(6,*) 'Fuel_p=',Fuel_p(0)
1140:
1141:     Fex_p(0) = Reac_p(0) + Fuel_p(0)
1142:     if(pr.ge.2) write(6,*) 'Fex_p=',Fex_p(0)
1143:
1144:     Xnb_t(0) = Xnb_cryo_t(0)
1145:     if(pr.ge.2) write(6,*) 'Xnb_t=',Xnb_t(0)
1146:
1147:     Xnb_d(0) = Xnb_cryo_d(0)
1148:     if(pr.ge.2) write(6,*) 'Xnb_d=',Xnb_d(0)
1149:
1150:     Fsto_tri_in(0) = ( 1.0D0 - linj_bot_iss - mgas_bot_iss ) *
1151:    &       Fiss_bot_(0)
1152:     if(pr.ge.2) write(6,*) 'Fsto_tri_in=',Fsto_tri_in(0)
1153:
1154:     Fsto_deu_in(0) = ( 1.0D0 - kiss_mid_ - linj_mid_iss -
1155:    &       mgas_mid_iss ) * Fiss_mid_(0)
1156:     if(pr.ge.2) write(6,*) 'Fsto_deu_in=',Fsto_deu_in(0)
1157:
1158:     Fsto_pro_in(0) = ( 1.0D0 - linj_p ) * Fiss_top_(0)
1159:     if(pr.ge.2) write(6,*) 'Fsto_pro_in=',Fsto_pro_in(0)
1160:C ...
1161:C
1162:     jerr = 0
1163:
1164:C ... print out
1165:     write(6,*) '---initial values'
1166:     do i=1,nvar
1167:       write(6,711) 'X(0,',i,') <',varname(i),'> =',XVARS(0,i)
1168:     end do
1169:  711 format(1x,a,i3,a,a,a,a,1p,d13.5)
1170:     return
1171:     end
```

fdiff.f

```
 1:      subroutine fdiffvect(i,j,t,delta_t,dt1,F,
 2:     &          Initial_step )
 3:C=================================================
 4:C
 5:C Calculate dX/dt and other variables at time t.
 6:C.....................................................
 7:C  Time "t" is time(i) and variable(i-1) is
 8:C  at time(i-1) = time t - delta_t.
 9:C.....................................................
10:C
11:C   i  : current time step no. (from 1).
12:C   j  : time step index of the last "tend(j)"
13:C   t  : current time.
14:C   delta_t  : time width of the current time step.
15:C   dt1  : time width of the last time step (if available).
16:C   F  : time differetiation coeeficients.
17:C   Initial_step  : set non-zero only for the first call
18:C
19:C*****************************************************
20:C  This file is created automatically.
21:C     Creation date : << Tue May 6 14:21:39 1997 >>
22:C*****************************************************
23:C
24:C=====================================================
25:
26:      implicit double precision (a-h,o-z)
27:C ... variables & constants definition ...
28:C
29:
30:      include '_sizes.h'
31:      include 'varconst.h'
32:
33:C ....... arguments ......
34:      double precision t,delta_t
35:      double precision F(ndfvar)
36:
37:C........ functions ......
38:      double precision ai,rho,Rc,Ed,fm,sigma_tend
39:
40:      double precision fnbpower
41:C-----------------------------------------------------
42:C
43:C   statement function
44:      ABX( aa,bb,xx ) = aa + bb * xx
45:C=====================================================
46:
47:
48:      tcycle = mod ( t , tcycle_p )
49:
50:      tpoint_cycle = t - Sigma_tend ( t , j )
51:*
52:*#######################################################
53:*##(The Plasma)
54:*#######################################################
55:*
56:
57:      fpp = fppower ( tcycle , t_burn , t_p_ru , t_p_rd )
58:
59:      Pfusion(i) = 1000D0 * fpp
60:
61:      Burn_t(i) = 2.12D-3 * Pfusion(i)
62:
63:      Fuel_t(i) = Burn_t(i) / fbu
64:
65:      RcM = Rc ( Medge , Tedge ) * fpp
66:
67:      ARcM = 1000D0 / 8760 * RcM / ( 0.5 * 3.016D0 )
68:
69:      F(1) = ABX ( d_C_t(i-1) , 0.0d0 , C_t(i-1) )
70:      call ABXdelta( d_C_t(i-1) , 0.0d0 , C_t, i, delta_t, 'C_t')
71:
72:      F(2) = ABX ( d_C_d(i-1) , 0.0d0 , C_d(i-1) )
73:      call ABXdelta( d_C_d(i-1) , 0.0d0 , C_d, i, delta_t, 'C_d')
74:
75:      F(3) = ABX ( d_C_p(i-1) , 0.0d0 , C_p(i-1) )
76:      call ABXdelta( d_C_p(i-1) , 0.0d0 , C_p, i, delta_t, 'C_p')
77:
78:      EdM = Ed ( Medge , Tedge ) * fpp
79:
80:      EdMAd = EdM * Ad * dc * rho ( Medge ) / ( 0.5 * 3.016D0 * 8760 )
81:
82:      F(4) = ABX ( d_D_t(i-1) , 0.0d0 , D_t(i-1) )
83:      call ABXdelta( d_D_t(i-1) , 0.0d0 , D_t, i, delta_t, 'D_t')
84:
85:      F(5) = ABX ( d_D_d(i-1) , 0.0d0 , D_d(i-1) )
86:      call ABXdelta( d_D_d(i-1) , 0.0d0 , D_d, i, delta_t, 'D_d')
87:
88:      F(6) = ABX ( d_D_p(i-1) , 0.0d0 , D_p(i-1) )
89:      call ABXdelta( d_D_p(i-1) , 0.0d0 , D_p, i, delta_t, 'D_p')
90:
91:      Fex_t(i) = Fuel_t(i) * ( 1 - fbu ) - ( d_Adiv_t(i-1) +
92:     &       d_Afw_t(i-1) ) - d_C_t(i-1) - d_D_t(i-1)
93:      if( Fex_t(i) .lt. 0.0d0 ) call negfix( Fex_t(i), 'Fex_t(i)',
94:     &       0.0d0 )
95:
96:      Fplas_in_t(i) = Fuelf_t(i-1) * fin_pellet
97:      if( Fplas_in_t(i) .lt. 0.0d0 ) call negfix( Fplas_in_t(i),
98:     &       'Fplas_in_t(i)', 0.0d0 )
99:
100:     Fgas_t(i) = Fuelf_t(i-1) * fin_gas
101:     if( Fgas_t(i) .lt. 0.0d0 ) call negfix( Fgas_t(i), 'Fgas_t(i)',
102:    &       0.0d0 )
103:
104:     Burn_d(i) = Burn_t(i)
105:
106:     Fplas_in_d(i) = Fuelf_t(i-1) * fin_pellet
```

```
107:     if( Fplas_in_d(i) .lt. 0.0d0 ) call negfix( Fplas_in_d(i),
108:    &        'Fplas_in_d(i)', 0.0d0 )
109:
110:     Fuel_d(i) = Fplas_in_d(i) + Fgas_d(i-1) + Fplas_nb_d(i-1) +
111:    &        Fplas_prop_d(i-1)
112:
113:     Fex_d(i) = Fuel_d(i) - Burn_d(i) - ( d_Adiv_d(i-1) + d_Afw_d(i-1)
114:    &        ) - d_C_d(i-1) - d_D_d(i-1)
115:     if( Fex_d(i) .lt. 0.0d0 ) call negfix( Fex_d(i), 'Fex_d(i)',
116:    &        0.0d0 )
117:
118:     Reac_p(i) = ap * fbu * Burn_t(i)
119:
120:     Fuel_p(i) = Fplas_in_p(i-1) + Xgas_p(i-1) * Fgas_tot(i-1) +
121:    &        Fplas_prop_p(i-1)
122:
123:     Fex_p(i) = Reac_p(i) + Fuel_p(i) - ( d_Adiv_p(i-1) + d_Afw_p(i-1)
124:    &        ) - d_C_p(i-1) - d_D_p(i-1)
125:     if( Fex_p(i) .lt. 0.0d0 ) call negfix( Fex_p(i), 'Fex_p(i)',
126:    &        0.0d0 )
127:
128:     Fhe_ex(i) = Burn_t(i)
129:
130:     fmMVdiv = fm ( Medge ) * ai ( Temperature , t ) * Vdiv / ( 0.5 *
131:    &        3.016D0 )
132:
133:     fmMVfw = fm ( Medge ) * ai ( Temperature , t ) * Vfw / ( 0.5 *
134:    &        3.016D0 )
135:
136:     FFF = Fex_t(i) + Fex_d(i) + Fex_p(i)
137:
138:     if ( FFF .ne. 0.0d0 ) then
139:
140:        i_t(i) = Fex_t(i) / FFF
141:
142:        i_d(i) = Fex_d(i) / FFF
143:
144:        i_p(i) = Fex_p(i) / FFF
145:
146:     else
147:
148:        i_t(i) = 0.0d0
149:
150:        i_d(i) = 0.0d0
151:
152:        i_p(i) = 0.0d0
153:
154:     endif
155:
156:     Adiv_t(i) = fmMVdiv * i_t(i) * fpp
157:
158:     Afw_t(i) = fmMVfw * i_t(i) * fpp
159:
160:     Adiv_d(i) = fmMVdiv * i_d(i) * fpp
161:
162:     Afw_d(i) = fmMVfw * i_d(i) * fpp
163:
164:     Adiv_p(i) = fmMVdiv * i_p(i) * fpp
165:
166:     Afw_p(i) = fmMVfw * i_p(i) * fpp
167:
168:     d_Adiv_t(i) = Adiv_t(i) - Adiv_t(i-1) - lambda_T * Adiv_t(i)
169:
170:     d_Adiv_d(i) = Adiv_d(i) - Adiv_d(i-1)
171:
172:     d_Adiv_p(i) = Adiv_p(i) - Adiv_p(i-1)
173:
174:     d_Afw_t(i) = Afw_t(i) - Afw_t(i-1) - lambda_T * Afw_t(i)
175:
176:     d_Afw_d(i) = Afw_d(i) - Afw_d(i-1)
177:
178:     d_Afw_p(i) = Afw_p(i) - Afw_p(i-1)
179:
180:     Tpfc_(i) = Adiv_t(i) + Afw_t(i) + C_t(i) + D_t(i)
181:
182:     Tpfc_m(i) = mu_A * ( Adiv_t(i) + Afw_t(i) ) + mu_C * C_t(i) +
183:    &        mu_D * D_t(i)
184:
185:     Dpfc_(i) = Adiv_d(i) + Afw_d(i) + C_d(i) + D_d(i)
186:
187:     Ppfc_(i) = Adiv_p(i) + Afw_p(i) + C_p(i) + D_p(i)
188:
189:     d_C_t(i) = ARcM * i_t(i) - lambda_T * C_t(i)
190:
191:     d_C_d(i) = ARcM * i_d(i)
192:
193:     d_C_p(i) = ARcM * i_p(i)
194:
195:     d_D_t(i) = EdMAd * i_t(i) - lambda_T * D_t(i)
196:
197:     d_D_d(i) = EdMAd * i_d(i)
198:
199:     d_D_p(i) = EdMAd * i_p(i)
200:*
201:* ... initial step is only for Plasma & fueling system
202:*
203:*if ( Initial_step.ne.0 ) then;
204:*   return ;
205:*endif ;
206:*
207:*****************************************************************
208:*## (Vacuum)
209:*****************************************************************
210:*
211:*
212:* Regeneration Cryopumps ------------------------------
213:*
214:*Tritium balance
215:
216:     emrRSAp = emr * Rs / Area_p
217:
218:     F(7) = ABX ( Fex_t(i-1) / npump , - emrRsAp , Tpv(i-1) )
219:     call ABXdelta( Fex_t(i-1) / npump , - emrRsAp  , Tpv, i, delta_t,
220:    &        'Tpv')
```

```
221:
222:      Rt_rem(i) = Tpv(i) * emrRsAp
223:
224:      Tp_tot(i) = Tpv(i) * npump
225:
226:      Tp_tot_m(i) = mu_v * Tp_tot(i)
227:* Deuterium Balance ------------------------------------
228:
229:      F(8) = ABX ( Fex_d(i-1) / npump , - emrRsAp , Dpv(i-1) )
230:      call ABXdelta( Fex_d(i-1) / npump , - emrRsAp  , Dpv, i, delta_t,
231:     &       'Dpv')
232:
233:      Rd_rem(i) = Dpv(i) * emrRsAp
234:
235:      Dp_tot(i) = Dpv(i) * npump
236:* Protium Balance ------------------------------------
237:
238:      F(9) = ABX ( Fex_p(i-1) / npump , - emrRsAp , Ppv(i-1) )
239:      call ABXdelta( Fex_p(i-1) / npump , - emrRsAp  , Ppv, i, delta_t,
240:     &       'Ppv')
241:
242:      Rp_rem(i) = Ppv(i) * emrRsAp
243:
244:      Pp_tot(i) = Ppv(i) * npump
245:* Helium Balance -------------------------------------
246:
247:      F(10) = ABX ( Fhe_ex(i-1) / npump , - emrRsAp , Hepv(i-1) )
248:      call ABXdelta( Fhe_ex(i-1) / npump , - emrRsAp  , Hepv, i,
249:     &       delta_t, 'Hepv')
250:
251:      Rhe_rem(i) = Hepv(i) * emrRsAp
252:
253:      Hep_tot(i) = Hepv(i) * npump
254:
255:      Fv_out(i) = ( Rt_rem(i) + Rd_rem(i) + Rp_rem(i) + Rhe_rem(i) ) *
256:     &       npump
257:      if( Fv_out(i) .lt. 0.0d0 ) call negfix( Fv_out(i), 'Fv_out(i)',
258:     &       0.0d0 )
259:
260:      Fpuri_in(i) = Fv_out(i)
261:* Isotopic Fractions ----------------------------------------
262:
263:      FF = Rt_rem(i) + Rd_rem(i) + Rp_rem(i)
264:
265:      if ( FF .gt. 0.0d0 ) then
266:
267:        it_(i) = Rt_rem(i) / FF
268:
269:        id_(i) = Rd_rem(i) / FF
270:
271:        ip_(i) = Rp_rem(i) / FF
272:
273:      else
274:
275:        it_(i) = 0.0d0
276:
277:        id_(i) = 0.0d0
278:
279:        ip_(i) = 0.0d0
280:
281:      endif
282:* To specify mole fractions of impurities in the exhaust stream
283:
284:      Fpure_(i) = Fv_out(i) * 0.98
285:
286:      Fimp_(i) = Fv_out(i) * 0.02
287:
288:      Fadj_puri(i) = Fv_out(i)
289:
290:      Fcq4_imp(i) = xcq4_ * Fimp_(i)
291:
292:      Fnq3_imp(i) = xnq3_ * Fimp_(i)
293:
294:      Fq2o_imp(i) = xq2o_ * Fimp_(i)
295:
296:      xh_imp(i) = xcq4_ + xnq3_ + xq2o_
297:
298:      xnhydro_(i) = 1 - 0.0136d0
299:
300:      Fnhydro_imp(i) = ( 1 - xh_imp(i) ) * Fimp_(i)
301:
302:      Fhydro_imp_w(i) = w1 * Fcq4_imp(i) + w2 * Fnq3_imp(i) + w3 *
303:     &       Fq2o_imp(i)
304:      if( Fhydro_imp_w(i) .lt. 0.0d0 ) call negfix( Fhydro_imp_w(i),
305:     &       'Fhydro_imp_w(i)', 0.0d0 )
306:
307:      Ft_pure(i) = Rt_rem(i) * npump - Fhydro_imp_w(i) * it_(i)
308:      if( Ft_pure(i) .lt. 0.0d0 ) call negfix( Ft_pure(i),
309:     &       'Ft_pure(i)', 0.0d0 )
310:
311:      Fd_pure(i) = Rd_rem(i) * npump - Fhydro_imp_w(i) * id_(i)
312:      if( Fd_pure(i) .lt. 0.0d0 ) call negfix( Fd_pure(i),
313:     &       'Fd_pure(i)', 0.0d0 )
314:
315:      Fp_pure(i) = Rp_rem(i) * npump - Fhydro_imp_w(i) * ip_(i)
316:      if( Fp_pure(i) .lt. 0.0d0 ) call negfix( Fp_pure(i),
317:     &       'Fp_pure(i)', 0.0d0 )
318:*
319:*#################################################################
320:*##(Fuel Clean-Up)
321:*#################################################################
322:*
323:*##.Flow into PURIFICATION
324:*
325:* .in purification mode
326:* Total Impurities:
327:
328:      F(11) = ABX ( ( xnhydro_(i) + xh_imp(i) ) * Fadj_puri(i-1) ,
329:     &       0.0d0 , Nimpt(i-1) )
330:      call ABXdelta( ( xnhydro_(i) + xh_imp(i) ) * Fadj_puri(i-1) ,
331:     &       0.0d0 , Nimpt, i, delta_t, 'Nimpt')
332:* Total Non-hydrogen-containing Impurities:
333:
334:      F(12) = ABX ( xnhydro_(i) * Fadj_puri(i-1) , 0.0d0 ,
```

```
335:     &       Nnhydro_imp(i-1) )
336:      call ABXdelta( xnhydro_(i) * Fadj_puri(i-1) , 0.0d0 ,
337:     &       Nnhydro_imp, i, delta_t, 'Nnhydro_imp')
338:* Hydrogen-containing Impurities:
339:
340:      F(13) = ABX ( xcq4_ * Fadj_puri(i-1) , 0.0d0 , Ncq4_imp(i-1) )
341:      call ABXdelta( xcq4_ * Fadj_puri(i-1) , 0.0d0 , Ncq4_imp, i,
342:     &       delta_t, 'Ncq4_imp')
343:
344:      F(14) = ABX ( xnq3_ * Fadj_puri(i-1) , 0.0d0 , Nnq3_imp(i-1) )
345:      call ABXdelta( xnq3_ * Fadj_puri(i-1) , 0.0d0 , Nnq3_imp, i,
346:     &       delta_t, 'Nnq3_imp')
347:
348:      F(15) = ABX ( xq2o_ * Fadj_puri(i-1) , 0.0d0 , Nq2o_imp(i-1) )
349:      call ABXdelta( xq2o_ * Fadj_puri(i-1) , 0.0d0 , Nq2o_imp, i,
350:     &       delta_t, 'Nq2o_imp')
351:* Hydrogen-containing Impurities:
352:
353:      f_hydro_imp_w(i) = w1 * xcq4_ + w2 * xnq3_ + w3 * xq2o_
354:*Tritium in Impurities:
355:
356:      F(16) = ABX ( f_hydro_imp_w(i) * it_(i-1) * Fadj_puri(i-1) ,
357:     &       0.0d0 , Timp_(i-1) )
358:      call ABXdelta( f_hydro_imp_w(i) * it_(i-1) * Fadj_puri(i-1) ,
359:     &       0.0d0 , Timp_, i, delta_t, 'Timp_')
360:*The mobilizable tritium is:
361:
362:      Timp_m(i) = mu_mob * Timp_(i)
363:*Deuterium in Impurities:
364:
365:      F(17) = ABX ( f_hydro_imp_w(i) * id_(i-1) * Fadj_puri(i-1) ,
366:     &       0.0d0 , Dimp_(i-1) )
367:      call ABXdelta( f_hydro_imp_w(i) * id_(i-1) * Fadj_puri(i-1) ,
368:     &       0.0d0 , Dimp_, i, delta_t, 'Dimp_')
369:*Protium in Impurities:
370:
371:      F(18) = ABX ( f_hydro_imp_w(i) * ip_(i-1) * Fadj_puri(i-1) ,
372:     &       0.0d0 , Pimp_(i-1) )
373:      call ABXdelta( f_hydro_imp_w(i) * ip_(i-1) * Fadj_puri(i-1) ,
374:     &       0.0d0 , Pimp_, i, delta_t, 'Pimp_')
375:*Total Hydrogen impurities:
376:
377:      Nhydro_imp(i) = Timp_(i) + Dimp_(i) + Pimp_(i)
378:*
379:* in Standby mode
380:*
381:
382:      Fh2_puri_in(i) = ( Ft_pure(i) + Fd_pure(i) + Fp_pure(i) ) / 2
383:
384:      FF = Ft_pure(i) + Fd_pure(i) + Fp_pure(i)
385:
386:      Ft2_sub_in(i) = Ft_pure(i) * ( 0.5 + ( xnhydro_(i-1) +
387:     &       xh_imp(i-1) ) / FF * Fadj_puri(i) * Lhydro_ / Limp_ )
388:      if( Ft2_sub_in(i) .lt. 0.0d0 ) call negfix( Ft2_sub_in(i),
389:     &       'Ft2_sub_in(i)', 0.0d0 )
390:
391:      Fd2_sub_in(i) = Fd_pure(i) * ( 0.5 + ( xnhydro_(i-1) +
392:     &       xh_imp(i-1) ) / FF * Fadj_puri(i) * Lhydro_ / Limp_ )
393:      if( Fd2_sub_in(i) .lt. 0.0d0 ) call negfix( Fd2_sub_in(i),
394:     &       'Fd2_sub_in(i)', 0.0d0 )
395:
396:      Fp2_sub_in(i) = Fp_pure(i) * ( 0.5 + ( xnhydro_(i-1) +
397:     &       xh_imp(i-1) ) / FF * Fadj_puri(i) * Lhydro_ / Limp_ )
398:      if( Fp2_sub_in(i) .lt. 0.0d0 ) call negfix( Fp2_sub_in(i),
399:     &       'Fp2_sub_in(i)', 0.0d0 )
400:
401:      Fh2_sub_in(i) = Ft2_sub_in(i) + Fd2_sub_in(i) + Fp2_sub_in(i)
402:      if( Fh2_sub_in(i) .lt. 0.0d0 ) call negfix( Fh2_sub_in(i),
403:     &       'Fh2_sub_in(i)', 0.0d0 )
404:* Preloading with Hydrogen:
405:* tau_sb = Ns_max / Fh2_sub_in(t) ;
406:* Mole fractions
407:
408:      if ( Fp2_sub_in(i) .gt. 1.0d-10 ) then
409:
410:        xt_sub(i) = Ft2_sub_in(i) / Fh2_sub_in(i)
411:
412:        xd_sub(i) = Fd2_sub_in(i) / Fh2_sub_in(i)
413:
414:        xp_sub(i) = Fp2_sub_in(i) / Fh2_sub_in(i)
415:
416:      else
417:
418:        xt_sub(i) = 0.0d0
419:
420:        xd_sub(i) = 0.0d0
421:
422:        xp_sub(i) = 0.0d0
423:
424:      endif
425:
426:      if ( abs ( Fh2_sub_in(i) ) .lt. 1.0d-10 ) then
427:
428:        tau_sb = 0.0d0
429:
430:      else
431:
432:        tau_sb = Ns_max / Fh2_sub_in(i)
433:
434:      endif
435:
436:      if ( abs ( tau_sb ) .ge. 1.0d-10 ) then
437:
438:        B = - 1.0d0 / tau_sb
439:
440:      else
441:
442:        B = 0.0d0
443:
444:      endif
445:
446:      F(19) = ABX ( 0.0d0 , B , Ns(i-1) )
447:      call ABXdelta( 0.0d0 , B , Ns, i, delta_t, 'Ns')
448:* Buildup of Hydrogen Molecules:
```

```
449:
450:     if ( abs ( tau_sb ) .ge. 1.0d-10 ) then
451:
452:       F(20) = ABX ( xt_sub(i-1) * Ns(i-1) / tau_sb , 0.0d0 ,
453:     &       T2_sub(i-1) )
454:       call ABXdelta( xt_sub(i-1) * Ns(i-1) / tau_sb , 0.0d0 ,
455:     &       T2_sub, i, delta_t, 'T2_sub')
456:
457:       F(21) = ABX ( xd_sub(i-1) * Ns(i-1) / tau_sb , 0.0d0 ,
458:     &       D2_sub(i-1) )
459:       call ABXdelta( xd_sub(i-1) * Ns(i-1) / tau_sb , 0.0d0 ,
460:     &       D2_sub, i, delta_t, 'D2_sub')
461:
462:       F(22) = ABX ( xp_sub(i-1) * Ns(i-1) / tau_sb , 0.0d0 ,
463:     &       P2_sub(i-1) )
464:       call ABXdelta( xp_sub(i-1) * Ns(i-1) / tau_sb , 0.0d0 ,
465:     &       P2_sub, i, delta_t, 'P2_sub')
466:
467:     else
468:
469:       F(20) = ABX ( 0.0d0 , 0.0d0 , T2_sub(i-1) )
470:       call ABXdelta( 0.0d0 , 0.0d0 , T2_sub, i, delta_t, 'T2_sub')
471:
472:       F(21) = ABX ( 0.0d0 , 0.0d0 , D2_sub(i-1) )
473:       call ABXdelta( 0.0d0 , 0.0d0 , D2_sub, i, delta_t, 'D2_sub')
474:
475:       F(22) = ABX ( 0.0d0 , 0.0d0 , P2_sub(i-1) )
476:       call ABXdelta( 0.0d0 , 0.0d0 , P2_sub, i, delta_t, 'P2_sub')
477:
478:     endif
479:
480:     Nhydro_sub(i) = T2_sub(i) + D2_sub(i) + P2_sub(i)
481:* Flow out from that is in Standby mode:
482:
483:     if ( abs ( tau_sb ) .ge. 1.0d-10 ) then
484:
485:       Ft_sub_out(i) = 2.0 * ( Ft2_sub_in(i) + xt_sub(i) * Ns(i) /
486:     &       tau_sb )
487:       if( Ft_sub_out(i) .lt. 0.0d0 ) call negfix( Ft_sub_out(i),
488:     &       'Ft_sub_out(i)', 0.0d0 )
489:
490:       Fd_sub_out(i) = 2.0 * ( Fd2_sub_in(i) + xd_sub(i) * Ns(i) /
491:     &       tau_sb )
492:       if( Fd_sub_out(i) .lt. 0.0d0 ) call negfix( Fd_sub_out(i),
493:     &       'Fd_sub_out(i)', 0.0d0 )
494:
495:       Fp_sub_out(i) = 2.0 * ( Fp2_sub_in(i) + xp_sub(i) * Ns(i) /
496:     &       tau_sb )
497:       if( Fp_sub_out(i) .lt. 0.0d0 ) call negfix( Fp_sub_out(i),
498:     &       'Fp_sub_out(i)', 0.0d0 )
499:
500:     else
501:
502:       Ft_sub_out(i) = 0.0d0
503:
504:       Fd_sub_out(i) = 0.0d0
505:
506:       Fp_sub_out(i) = 0.0d0
507:
508:     endif
509:
510:     Fh_sub_out(i) = Ft_sub_out(i) + Fd_sub_out(i) + Fp_sub_out(i)
511:     if( Fh_sub_out(i) .lt. 0.0d0 ) call negfix( Fh_sub_out(i),
512:     &       'Fh_sub_out(i)', 0.0d0 )
513:*
514:* .In Regeneration Mode
515:*
516:* Inventory during in Regeneration:
517:
518:     if ( tpoint_cycle .lt. treg_msieve ) then
519:
520:       Treg_out(i) = Timp_(j) * ( 1 - tpoint_cycle / treg_msieve )
521:
522:     else
523:
524:       Treg_out(i) = 0D0
525:
526:     endif
527:* The vulnerable tritium inventory on the regenerating beds is:
528:
529:     Treg_out_m(i) = mu_reg_m * Treg_out(i)
530:* Evaluation of Hydrogen-containing Species:
531:
532:     Ncq4_reg(i) = Ncq4_imp(j)
533:
534:     Nnq3_reg(i) = Nnq3_imp(j)
535:
536:     Nq2o_reg(i) = Nq2o_imp(j)
537:
538:     Ecq4_reg(i) = Ncq4_reg(i) / treg_msieve
539:
540:     Enq3_reg(i) = Nnq3_reg(i) / treg_msieve
541:
542:     Eq2o_reg(i) = Nq2o_reg(i) / treg_msieve
543:*
544:* Palladium Membrane Reactor:
545:*
546:
547:     Fhydro_pmem(i) = w1 * Ecq4_reg(i) + w2 * Enq3_reg(i) + w3 *
548:     &       Eq2o_reg(i)
549:* Inventories in the Palladium Membrane Reactor:
550:
551:     if ( j .lt. i .or. Nhydro_imp(j) .lt. 1.0d-5 ) then
552:
553:       it_endj(i) = 0.0D0
554:
555:       id_endj(i) = 0.0D0
556:
557:       ip_endj(i) = 0.0D0
558:
559:     else
560:
561:       it_endj(i) = Timp_(j) / Nhydro_imp(j)
562:
```

```
563:        id_endj(i) = Pimp_(j) / Nhydro_imp(j)
564:
565:        ip_endj(i) = Timp_(j) / Nhydro_imp(j)
566:
567:      endif
568:
569:      F(23) = ABX ( it_endj(i-1) * Fhydro_pmem(i-1) , - 1.0d0 /
570:     &       tau_pmem , Tpmem(i-1) )
571:      call ABXdelta( it_endj(i-1) * Fhydro_pmem(i-1) , - 1.0d0 /
572:     &       tau_pmem , Tpmem, i, delta_t, 'Tpmem')
573:
574:      F(24) = ABX ( id_endj(i-1) * Fhydro_pmem(i-1) , - 1.0d0 /
575:     &       tau_pmem , Dpmem(i-1) )
576:      call ABXdelta( id_endj(i-1) * Fhydro_pmem(i-1) , - 1.0d0 /
577:     &       tau_pmem , Dpmem, i, delta_t, 'Dpmem')
578:
579:      F(25) = ABX ( ip_endj(i-1) * Fhydro_pmem(i-1) , - 1.0d0 /
580:     &       tau_pmem , Ppmem(i-1) )
581:      call ABXdelta( ip_endj(i-1) * Fhydro_pmem(i-1) , - 1.0d0 /
582:     &       tau_pmem , Ppmem, i, delta_t, 'Ppmem')
583:*
584:* .Flow out from PURIFICATION
585:* .to Isotope Separation System
586:*
587:
588:      if ( Tpmem(i) .lt. 1.0d-10 ) then
589:
590:        Ft_puri_out(i) = Ft_sub_out(i)
591:
592:      else
593:
594:        Ft_puri_out(i) = Ft_sub_out(i) + Tpmem(i) / tau_pmem
595:
596:      endif
597:      if( Ft_puri_out(i) .lt. 0.0d0 ) call negfix( Ft_puri_out(i),
598:     &       'Ft_puri_out(i)', 0.0d0 )
599:
600:      if ( Dpmem(i) .lt. 1.0d-10 ) then
601:
602:        Fd_puri_out(i) = Fd_sub_out(i)
603:
604:      else
605:
606:        Fd_puri_out(i) = Fd_sub_out(i) + Dpmem(i) / tau_pmem
607:
608:      endif
609:      if( Fd_puri_out(i) .lt. 0.0d0 ) call negfix( Fd_puri_out(i),
610:     &       'Fd_puri_out(i)', 0.0d0 )
611:
612:      if ( Ppmem(i) .lt. 1.0d-10 ) then
613:
614:        Fp_puri_out(i) = Fp_sub_out(i)
615:
616:      else
617:
618:        Fp_puri_out(i) = Fp_sub_out(i) + Ppmem(i) / tau_pmem
619:
620:      endif
621:      if( Fp_puri_out(i) .lt. 0.0d0 ) call negfix( Fp_puri_out(i),
622:     &       'Fp_puri_out(i)', 0.0d0 )
623:
624:      Fpuri_out(i) = Ft_puri_out(i) + Fd_puri_out(i) + Fp_puri_out(i)
625:      if( Fpuri_out(i) .lt. 0.0d0 ) call negfix( Fpuri_out(i),
626:     &       'Fpuri_out(i)', 0.0d0 )
627:*
628:*#########################################################
629:*##(Isotope Separation System)
630:*#########################################################
631:*
632:* .Input stream to the ISS
633:*
634:
635:      Fiss_in(i) = Fbl_cool + Fbl_ + ( 1 - h1 - h3 ) * Fpuri_out(i) +
636:     &       Fragnb * Fiss_nb(i-1) + Fragiss_inj * Fiss_inj(i-1)
637:      if( Fiss_in(i) .lt. 0.0d0 ) call negfix( Fiss_in(i),
638:     &       'Fiss_in(i)', 0.0d0 )
639:* .tritium flow
640:
641:      Fiss_in_t(i) = Fbl_cool * Xc_t + Fbl_ * Xbl_t + ( 1 - h1 - h3 ) *
642:     &       Fpuri_out(i) * Xiss_in_t(i-1) + Fragnb * Fiss_nb(i-1) *
643:     &       Xnb_t(i-1) + Fragiss_inj * Fiss_inj(i-1) * Xinj_t(i-1)
644:      if( Fiss_in_t(i) .lt. 0.0d0 ) call negfix( Fiss_in_t(i),
645:     &       'Fiss_in_t(i)', 0.0d0 )
646:* .deuteriumflow
647:
648:      Fiss_in_d(i) = Fbl_cool * Xc_d + Fbl_ * Xbl_d + ( 1 - h1 - h3 ) *
649:     &       Fpuri_out(i) * Xiss_in_d(i-1) + Fragnb * Fiss_nb(i-1) *
650:     &       Xnb_d(i-1) + Fragiss_inj * Fiss_inj(i-1) * Xinj_d(i-1)
651:      if( Fiss_in_d(i) .lt. 0.0d0 ) call negfix( Fiss_in_d(i),
652:     &       'Fiss_in_d(i)', 0.0d0 )
653:
654:      Xnb_t(i) = Xnb_cryo_t(i-1)
655:
656:      Xnb_d(i) = Xnb_cryo_d(i-1)
657:* Protium flow
658:
659:      Fiss_in_p(i) = Fiss_in(i) - Fiss_in_t(i) - Fiss_in_d(i)
660:      if( Fiss_in_p(i) .lt. 0.0d0 ) call negfix( Fiss_in_p(i),
661:     &       'Fiss_in_p(i)', 0.0d0 )
662:* mole fraction
663:
664:      if ( Fiss_in(i) .ne. 0.0d0 ) then
665:
666:        Xiss_in_t(i) = Fiss_in_t(i) / Fiss_in(i)
667:
668:        Xiss_in_d(i) = Fiss_in_d(i) / Fiss_in(i)
669:
670:        Xiss_in_p(i) = 1 - Xiss_in_t(i) - Xiss_in_d(i)
671:
672:      else
673:
674:        Xiss_in_t(i) = 0.0d0
675:
676:        Xiss_in_d(i) = 0.0d0
```

```
677:
678:        Xiss_in_p(i) = 0.0d0
679:
680:      endif
681:*
682:* .Isotope Separation System
683:*
684:*  overall balance
685:
686:      F(26) = ABX ( Fiss_in(i-1) , - 1.0d0 / tau_iss_ , Niss_(i-1) )
687:      call ABXdelta( Fiss_in(i-1) , - 1.0d0 / tau_iss_ , Niss_, i,
688:     &       delta_t, 'Niss_')
689:* tritium balance
690:
691:      F(27) = ABX ( Fiss_in_t(i-1) , - 1.0d0 / tau_iss_ , Tiss_(i-1) )
692:      call ABXdelta( Fiss_in_t(i-1) , - 1.0d0 / tau_iss_ , Tiss_, i,
693:     &       delta_t, 'Tiss_')
694:* mobilizable tritium
695:
696:      Tiss_m(i) = mu_iss_ * Tiss_(i)
697:* .deuterium balance
698:
699:      F(28) = ABX ( Fiss_in_d(i-1) , - 1.0d0 / tau_iss_ , Diss_(i-1) )
700:      call ABXdelta( Fiss_in_d(i-1) , - 1.0d0 / tau_iss_ , Diss_, i,
701:     &       delta_t, 'Diss_')
702:*
703:
704:      if ( Niss_(i) .gt. 0.0 ) then
705:
706:        Xiss_t(i) = Tiss_(i) / Niss_(i)
707:
708:        Xiss_d(i) = Diss_(i) / Niss_(i)
709:
710:        Xiss_p(i) = 1.0d0 - Xiss_t(i) - Xiss_d(i)
711:
712:      else
713:
714:        Xiss_t(i) = 0.0d0
715:
716:        Xiss_d(i) = 0.0d0
717:
718:        Xiss_p(i) = 0.0d0
719:
720:      endif
721:* .protium balance
722:
723:      Piss_(i) = Xiss_p(i) * Niss_(i)
724:*
725:* III. exit flow from the ISS
726:*
727:*-----------------------------------------------------------------
728:*  Solve the follwing equations:
729:*  Fiss_top_ + Fiss_mid_ + Fiss_bot_ = Niss_/tau_iss_
730:*  Fiss_top_*Xiss_top_t + Fiss_mid_*Xiss_mid_t + Fiss_bot_*Xiss_bot_t
731:*        = Tiss_/tau_iss_
732:*  Fiss_top_*Xiss_top_d + Fiss_mid_*Xiss_mid_d + Fiss_bot_*Xiss_bot_d
733:*        = Diss_/tau_iss_
734:*-----------------------------------------------------------------
735:*
736:
737:      P1 = 1.0 / ( Xiss_mid_t * Xiss_bot_d + Xiss_bot_t * Xiss_top_d +
738:     &       Xiss_top_t * Xiss_mid_d - Xiss_mid_t * Xiss_top_d -
739:     &       Xiss_top_t * Xiss_bot_d - Xiss_bot_t * Xiss_mid_d )
740:
741:      Fiss_top_(i) = P1 * ( ( Xiss_mid_t * Xiss_bot_d - Xiss_bot_t *
742:     &       Xiss_mid_d ) * Niss_(i) / tau_iss_ + ( Xiss_mid_d -
743:     &       Xiss_bot_d ) * Tiss_(i) / tau_iss_ + ( Xiss_bot_t -
744:     &       Xiss_mid_t ) * Diss_(i) / tau_iss_ )
745:      if( Fiss_top_(i) .lt. 0.0d0 ) call negfix( Fiss_top_(i),
746:     &       'Fiss_top_(i)', 0.0d0 )
747:
748:      Fiss_mid_(i) = P1 * ( ( Xiss_bot_t * Xiss_top_d - Xiss_top_t *
749:     &       Xiss_bot_d ) * Niss_(i) / tau_iss_ + ( Xiss_bot_d -
750:     &       Xiss_top_d ) * Tiss_(i) / tau_iss_ + ( Xiss_top_t -
751:     &       Xiss_bot_t ) * Diss_(i) / tau_iss_ )
752:      if( Fiss_mid_(i) .lt. 0.0d0 ) call negfix( Fiss_mid_(i),
753:     &       'Fiss_mid_(i)', 0.0d0 )
754:
755:      Fiss_bot_(i) = P1 * ( ( Xiss_top_t * Xiss_mid_d - Xiss_mid_t *
756:     &       Xiss_top_d ) * Niss_(i) / tau_iss_ + ( Xiss_top_d -
757:     &       Xiss_mid_d ) * Tiss_(i) / tau_iss_ + ( Xiss_mid_t -
758:     &       Xiss_top_t ) * Diss_(i) / tau_iss_ )
759:      if( Fiss_bot_(i) .lt. 0.0d0 ) call negfix( Fiss_bot_(i),
760:     &       'Fiss_bot_(i)', 0.0d0 )
761:*
762:*#######################################################
763:*##(Deuterium Neutral Beams)
764:*#######################################################
765:*
766:* .flow to the Neutral beams
767:*
768:
769:      W_nb(i) = 75D0 * fppower0 ( tcycle , t_burn , t_p_ru , t_p_rd )
770:
771:      Fplas_nb_d(i) = 3.6D3 / 1.6D0 * 1.0D19 * 1 / No * W_nb(i) / E_nb
772:
773:      if ( ( tcycle .lt. t_p_ru ) .or. ( tcycle .gt. tcycle_nb ) ) then
774:
775:        Crec(i) = 0.0D0
776:
777:      else
778:
779:        Crec(i) = 1.0D0
780:
781:      endif
782:
783:      Fin_rec_(i) = Crec(i-1) * Fnb_cryo(i-1)
784:
785:      Fiss_nb(i) = ( 1.0D0 - Crec(i-1) ) * Fnb_cryo(i-1)
786:*
787:* required flow from storage
788:*
789:
790:      Fin_st_d(i) = ( Fplas_nb_d(i) / eta_nb_tot - kiss_mid_ *
```

```
791:     &       Fiss_mid_(i) * Xiss_mid_d ) / Xsto_deu_d(i-1)
792:      if( Fin_st_d(i) .lt. 0.0d0 ) call negfix( Fin_st_d(i),
793:     &       'Fin_st_d(i)', 0.0d0 )
794:* III.Continuous regeneration cryopump
795:*
796:
797:      F(29) = ABX ( fcryo_ * ( kiss_mid_ * Fiss_mid_(i-1) + Fin_st_d(i)
798:     &       ) + Fbk_nb_(i-1) , - emrRsAp , Ncryo_(i-1) )
799:      call ABXdelta( fcryo_ * ( kiss_mid_ * Fiss_mid_(i-1) +
800:     &       Fin_st_d(i) ) + Fbk_nb_(i-1) , - emrRsAp , Ncryo_, i,
801:     &       delta_t, 'Ncryo_')
802:
803:      Rnb_reg_(i) = Ncryo_(i) * emr * Rs / Area_p
804:
805:      F(30) = ABX ( fcryo_ * ( kiss_mid_ * Fiss_mid_(i-1) * Xiss_mid_t
806:     &       + Fin_st_d(i) * Xsto_deu_t(i-1) ) + Fbk_nb_t(i-1) , -
807:     &       emrRsAp , Tnb_cryo(i-1) )
808:      call ABXdelta( fcryo_ * ( kiss_mid_ * Fiss_mid_(i-1) * Xiss_mid_t
809:     &       + Fin_st_d(i) * Xsto_deu_t(i-1) ) + Fbk_nb_t(i-1) , -
810:     &       emrRsAp , Tnb_cryo, i, delta_t, 'Tnb_cryo')
811:
812:      Tnb_cryo_mob(i) = mu_nb_mob * Tnb_cryo(i)
813:
814:      Rnb_reg_t(i) = Tnb_cryo(i) * emrRsAp
815:
816:      F(31) = ABX ( fcryo_ * ( kiss_mid_ * Fiss_mid_(i-1) * Xiss_mid_d
817:     &       + Fin_st_d(i) * Xsto_deu_d(i-1) ) + Fbk_nb_d(i-1) -
818:     &       Rnb_reg_d(i-1) , - emrRsAp , Dnb_cryo(i-1) )
819:      call ABXdelta( fcryo_ * ( kiss_mid_ * Fiss_mid_(i-1) * Xiss_mid_d
820:     &       + Fin_st_d(i) * Xsto_deu_d(i-1) ) + Fbk_nb_d(i-1) -
821:     &       Rnb_reg_d(i-1) , - emrRsAp , Dnb_cryo, i, delta_t,
822:     &       'Dnb_cryo')
823:
824:      Rnb_reg_d(i) = Dnb_cryo(i) * emrRsAp
825:* mole fraction on the neutral beam cryopump:
826:
827:      if ( Ncryo_(i) .gt. 0.0d0 ) then
828:
829:        Xnb_cryo_t(i) = Tnb_cryo(i) / Ncryo_(i)
830:
831:        Xnb_cryo_d(i) = Dnb_cryo(i) / Ncryo_(i)
832:
833:        Xnb_cryo_p(i) = 1.0 - Xnb_cryo_d(i) - Xnb_cryo_t(i)
834:
835:        if ( Xnb_cryo_p(i) .lt. 0.0d0 ) then
836:
837:          Xnb_cryo_p(i) = 0.0d0
838:
839:        endif
840:
841:      else
842:
843:        Xnb_cryo_t(i) = 0.0d0
844:
845:        Xnb_cryo_d(i) = 0.0d0
846:
847:        Xnb_cryo_p(i) = 0.0d0
848:
849:      endif
850:* protium balance
851:
852:      Pnb_cryo(i) = Xnb_cryo_p(i) * Ncryo_(i)
853:
854:      Rnb_reg_p(i) = Pnb_cryo(i) * emr * Rs / Area_p
855:*
856:* .Tritium in dump
857:*
858:* .Tritium in the walls
859:* total inventory:
860:
861:      F(32) = ABX ( fnb_wall * ( kiss_mid_ * Fiss_mid_(i-1) *
862:     &       Xiss_mid_t + Fin_st_d(i-1) * Xsto_deu_t(i-1) ) , 0.0d0 ,
863:     &       Tnb_wall(i-1) )
864:      call ABXdelta( fnb_wall * ( kiss_mid_ * Fiss_mid_(i-1) *
865:     &       Xiss_mid_t + Fin_st_d(i-1) * Xsto_deu_t(i-1) ) , 0.0d0 ,
866:     &       Tnb_wall, i, delta_t, 'Tnb_wall')
867:
868:      Tnb_wall_mob(i) = mu_wall_ * Tnb_wall(i)
869:*
870:* Backflow from plasma:
871:*
872:
873:      Fbk_nb_(i) = 3 * 5.98D-21 * nedge_tot
874:
875:      Fbk_nb_t(i) = 3 * 5.98D-21 * nedge_t
876:
877:      Fbk_nb_d(i) = 3 * 5.98D-21 * nedge_d
878:*
879:* Flow to plasma
880:*
881:*total flow to plasma
882:*tritium flow to the plasma
883:*deuterium flow to the plasma
884:
885:      Fplas_nb_(i) = ( kiss_mid_ * Fiss_mid_(i) + Fin_st_d(i) ) *
886:     &       eta_nb_tot
887:      if( Fplas_nb_(i) .lt. 0.0d0 ) call negfix( Fplas_nb_(i),
888:     &       'Fplas_nb_(i)', 0.0d0 )
889:
890:      Fplas_nb_t(i) = ( kiss_mid_ * Fiss_mid_(i) * Xiss_mid_t +
891:     &       Fin_st_d(i) * Xsto_deu_t(i-1) ) * eta_nb_tot
892:      if( Fplas_nb_t(i) .lt. 0.0d0 ) call negfix( Fplas_nb_t(i),
893:     &       'Fplas_nb_t(i)', 0.0d0 )
894:* protium flow to the plasma
895:
896:      Fplas_nb_p(i) = Fplas_nb_(i) - Fplas_nb_t(i) - Fplas_nb_d(i)
897:      if( Fplas_nb_p(i) .lt. 0.0d0 ) call negfix( Fplas_nb_p(i),
898:     &       'Fplas_nb_p(i)', 0.0d0 )
899:* .Exhaust flow from neutral beam cryopump
900:
901:      Fnb_cryo(i) = Rnb_reg_t(i) + Rnb_reg_d(i) + Rnb_reg_p(i)
902:      if( Fnb_cryo(i) .lt. 0.0d0 ) call negfix( Fnb_cryo(i),
903:     &       'Fnb_cryo(i)', 0.0d0 )
904:*
```

```
905:*************************************************************
906:*## (Pellet injector)
907:*************************************************************
908:*
909:*  II. The required flows
910:
911:      P8 = Fplas_in_t(i) / ( 1.0 - finj_err ) - Fsto_out_p(i-1) *
912:     &       Xsto_pro_t(i-1) - h3 * Fpuri_out(i-1) * Xiss_in_t(i-1) -
913:     &       linj_mid_iss * Fiss_mid_(i-1) * Xiss_mid_t - linj_bot_iss
914:     &       * Fiss_bot_(i-1) * Xiss_bot_t
915:
916:      P9 = Fplas_in_d(i) / ( 1.0 - finj_err ) - Fsto_out_p(i-1) *
917:     &       Xsto_pro_d(i-1) - h3 * Fpuri_out(i-1) * Xiss_in_d(i-1) -
918:     &       linj_mid_iss * Fiss_mid_(i-1) * Xiss_mid_d - linj_bot_iss
919:     &       * Fiss_bot_(i-1) * Xiss_bot_d
920:
921:      PP = Xsto_tri_t(i-1) * Xsto_deu_d(i-1) - Xsto_deu_t(i-1) *
922:     &       Xsto_tri_d(i-1)
923:
924:      Fsto_out_t(i) = ( Xsto_deu_d(i-1) * P8 - Xsto_deu_t(i-1) * P9 ) /
925:     &       PP
926:      if( Fsto_out_t(i) .lt. 0.0d0 ) call negfix( Fsto_out_t(i),
927:     &       'Fsto_out_t(i)', 0.0d0 )
928:
929:      Fsto_out_d(i) = ( - Xsto_tri_d(i-1) * P8 + Xsto_tri_t(i-1) * P9 )
930:     &       / PP
931:      if( Fsto_out_d(i) .lt. 0.0d0 ) call negfix( Fsto_out_d(i),
932:     &       'Fsto_out_d(i)', 0.0d0 )
933:*
934:* Filling extruder:
935:*
936:
937:      dNext_12 = h3 * Fpuri_out(i-1) + linj_bot_iss * Fiss_bot_(i-1) +
938:     &       linj_mid_iss * Fiss_mid_(i-1) + Fsto_out_t(i-1) +
939:     &       Fsto_out_d(i-1) + Fsto_out_p(i-1)
940:
941:      dText_12 = h3 * Fpuri_out(i-1) * Xiss_in_t(i-1) + linj_bot_iss *
942:     &       Fiss_bot_(i-1) * Xiss_bot_t + linj_mid_iss *
943:     &       Fiss_mid_(i-1) * Xiss_mid_t + Fsto_out_t(i-1) *
944:     &       Xsto_tri_t(i-1) + Fsto_out_d(i-1) * Xsto_deu_t(i-1) +
945:     &       Fsto_out_p(i-1) * Xsto_pro_t(i-1)
946:
947:      dDext_12 = h3 * Fpuri_out(i-1) * Xiss_in_d(i-1) + linj_bot_iss *
948:     &       Fiss_bot_(i-1) * Xiss_bot_d + linj_mid_iss *
949:     &       Fiss_mid_(i-1) * Xiss_mid_d + Fsto_out_t(i-1) *
950:     &       Xsto_tri_d(i-1) + Fsto_out_d(i-1) * Xsto_deu_d(i-1) +
951:     &       Fsto_out_p(i-1) * Xsto_pro_d(i-1)
952:
953:      F(33) = ABX ( dNext_12 - ( Fplas_in_t(i-1) + Fplas_in_d(i-1) +
954:     &       Fplas_in_p(i-1) ) / ( 1 - finj_err ) , 0.0d0 ,
955:     &       Next_12(i-1) )
956:      call ABXdelta( dNext_12 - ( Fplas_in_t(i-1) + Fplas_in_d(i-1) +
957:     &       Fplas_in_p(i-1) ) / ( 1 - finj_err ) , 0.0d0 , Next_12,
958:     &       i, delta_t, 'Next_12')
959:
960:      F(34) = ABX ( dText_12 - Fplas_in_t(i-1) / ( 1 - finj_err ) ,
961:     &       0.0d0 , Text_12(i-1) )
962:      call ABXdelta( dText_12 - Fplas_in_t(i-1) / ( 1 - finj_err ) ,
963:     &       0.0d0 , Text_12, i, delta_t, 'Text_12')
964:
965:      F(35) = ABX ( dDext_12 - Fplas_in_d(i-1) / ( 1 - finj_err ) ,
966:     &       0.0d0 , Dext_12(i-1) )
967:      call ABXdelta( dDext_12 - Fplas_in_d(i-1) / ( 1 - finj_err ) ,
968:     &       0.0d0 , Dext_12, i, delta_t, 'Dext_12')
969:
970:      if ( Next_12(i) .ne. 0.0d0 ) then
971:
972:        Xext_12_t(i) = Text_12(i) / Next_12(i)
973:
974:        Xext_12_d(i) = Dext_12(i) / Next_12(i)
975:
976:        Xext_12_p(i) = 1.0 - Xext_12_t(i) - Xext_12_d(i)
977:
978:      else
979:
980:        Xext_12_t(i) = 0.0d0
981:
982:        Xext_12_d(i) = 0.0d0
983:
984:        Xext_12_p(i) = 0.0d0
985:
986:      endif
987:
988:      Pext_12(i) = Xext_12_p(i) * Next_12(i)
989:*
990:* Total inventory in extruders:
991:*
992:
993:      Next_tot(i) = Next_12(i) * 2.0
994:
995:      Text_tot(i) = Text_12(i) * 2.0
996:
997:      Dext_tot(i) = Dext_12(i) * 2.0
998:
999:      Pext_tot(i) = Pext_12(i) * 2.0
1000:*
1001:* The mobilizable tritium inventory is:
1002:*
1003:
1004:      Text_m(i) = mu_ext * Text_tot(i)
1005:
1006:      Nextru_(i) = Next_12(i)
1007:
1008:      Xextru_t(i) = Text_12(i) / Next_12(i)
1009:
1010:      Xextru_d(i) = Dext_12(i) / Next_12(i)
1011:
1012:      Xextru_p(i) = 1.0d0 - Xextru_t(i) - Xextru_d(i)
1013:*
1014:* Pellet Characteristics
1015:*
1016:
1017:      Vinj_t(i) = Xextru_t(i) * Nextru_(i) * 18.894D-6 / 2
1018:
```

```
1019:     Vinj_d(i) = Xextru_d(i) * Nextru_(i) * 20.085D-6 / 2
1020:
1021:     Vinj_p(i) = Xextru_p(i) * Nextru_(i) * 23.326D-6 / 2
1022:* total volume in extruder
1023:
1024:     Vinj_tot(i) = Vinj_t(i) + Vinj_d(i) + Vinj_p(i)
1025:* pellet volume
1026:* molar content of pellet
1027:
1028:     npellet_t(i) = Vinj_t(i) / Vinj_tot(i) * Vpellet * 2 * kpellet_t
1029:
1030:     npellet_d(i) = Vinj_d(i) / Vinj_tot(i) * Vpellet * 2 * kpellet_d
1031:
1032:     npellet_p(i) = Vinj_p(i) / Vinj_tot(i) * Vpellet * 2 * kpellet_p
1033:* repetition rate of injector
1034:
1035:     hrinj_ = Fplas_in_t(i) / ( npellet_t(i) * ( 1 - finj_err ) )
1036:*
1037:*  ... Rinj_ in Hz ...
1038:
1039:     Rinj_(i) = hrinj_ / 3600d0
1040:*
1041:* protium to plasma from pellet:
1042:
1043:     Fplas_in_p(i) = hrinj_ * npellet_p(i) * ( 1 - finj_err )
1044:     if( Fplas_in_p(i) .lt. 0.0d0 ) call negfix( Fplas_in_p(i),
1045:    &       'Fplas_in_p(i)', 0.0d0 )
1046:* total flow to plasma from pellet:
1047:
1048:     Finj_in_tot(i) = Fplas_in_t(i) + Fplas_in_d(i) + Fplas_in_p(i)
1049:     if( Finj_in_tot(i) .lt. 0.0d0 ) call negfix( Finj_in_tot(i),
1050:    &       'Finj_in_tot(i)', 0.0d0 )
1051:*
1052:* .Backflow from plasma
1053:*
1054:
1055:     Fbk_plas_(i) = 3 * 5.98D-22 * nedge_tot
1056:
1057:     Fbk_plas_t(i) = 3 * 5.98D-22 * nedge_t
1058:
1059:     Fbk_plas_d(i) = 3 * 5.98D-22 * nedge_d
1060:
1061:     Fbk_plas_p(i) = 3 * 5.98D-22 * nedge_p
1062:*
1063:* Propellant requirements
1064:*
1065:
1066:     Finj_prop(i) = nprop * hrinj_ / 3600d0
1067:     if( Finj_prop(i) .lt. 0.0d0 ) call negfix( Finj_prop(i),
1068:    &       'Finj_prop(i)', 0.0d0 )
1069:
1070:     Fsto_out_p(i) = Finj_prop(i) - liss_p * Fiss_top_(i-1)
1071:
1072:     if ( Fsto_out_p(i) .gt. 0 ) then
1073:
1074:        Xprop_t(i) = ( Fsto_out_p(i-1) * Xsto_pro_t(i-1) + liss_p *
1075:    &       Fiss_top_(i-1) * Xiss_top_t ) / Finj_prop(i)
1076:
1077:        Xprop_d(i) = ( Fsto_out_p(i-1) * Xsto_pro_d(i-1) + liss_p *
1078:    &       Fiss_top_(i-1) * Xiss_top_d ) / Finj_prop(i)
1079:
1080:     else
1081:
1082:        Fsto_out_p(i) = 0.0d0
1083:
1084:        Xprop_t(i) = Xiss_top_t
1085:
1086:        Xprop_d(i) = Xiss_top_d
1087:
1088:     endif
1089:
1090:     Xprop_p(i) = 1 - Xprop_t(i) - Xprop_d(i)
1091:*
1092:* propellant flow to plasma
1093:*
1094:
1095:     Fplas_prop(i) = ( 1 - fcap_prop ) * Finj_prop(i)
1096:     if( Fplas_prop(i) .lt. 0.0d0 ) call negfix( Fplas_prop(i),
1097:    &       'Fplas_prop(i)', 0.0d0 )
1098:
1099:     Fplas_prop_t(i) = Xprop_t(i) * Fplas_prop(i)
1100:     if( Fplas_prop_t(i) .lt. 0.0d0 ) call negfix( Fplas_prop_t(i),
1101:    &       'Fplas_prop_t(i)', 0.0d0 )
1102:
1103:     Fplas_prop_d(i) = Xprop_d(i) * Fplas_prop(i)
1104:     if( Fplas_prop_d(i) .lt. 0.0d0 ) call negfix( Fplas_prop_d(i),
1105:    &       'Fplas_prop_d(i)', 0.0d0 )
1106:
1107:     Fplas_prop_p(i) = Xprop_p(i) * Fplas_prop(i)
1108:     if( Fplas_prop_p(i) .lt. 0.0d0 ) call negfix( Fplas_prop_p(i),
1109:    &       'Fplas_prop_p(i)', 0.0d0 )
1110:*
1111:* Exhaust flow from pellet injector
1112:*
1113:
1114:     Finj_exh(i) = ( ( Fplas_in_t(i) + Fplas_in_d(i) ) / ( 1 -
1115:    &       finj_err ) + Fplas_in_p(i) ) * finj_err + Finj_prop(i) *
1116:    &       fcap_prop + Fbk_plas_(i)
1117:     if( Finj_exh(i) .lt. 0.0d0 ) call negfix( Finj_exh(i),
1118:    &       'Finj_exh(i)', 0.0d0 )
1119:
1120:     Finj_exh_t(i) = Fplas_in_t(i) * finj_err / ( 1 - finj_err ) +
1121:    &       Finj_prop(i) * Xprop_t(i) * fcap_prop + Fbk_plas_t(i)
1122:     if( Finj_exh_t(i) .lt. 0.0d0 ) call negfix( Finj_exh_t(i),
1123:    &       'Finj_exh_t(i)', 0.0d0 )
1124:
1125:     Finj_exh_d(i) = Fplas_in_d(i) * finj_err / ( 1 - finj_err ) +
1126:    &       Finj_prop(i) * Xprop_d(i) * fcap_prop + Fbk_plas_d(i)
1127:     if( Finj_exh_d(i) .lt. 0.0d0 ) call negfix( Finj_exh_d(i),
1128:    &       'Finj_exh_d(i)', 0.0d0 )
1129:*
1130:* The flow directed to the ISS is given by:
1131:*
1132:
```

```
1133:     Fiss_inj(i) = ( 1 - fnon_iss ) * Finj_exh(i)
1134:     if( Fiss_inj(i) .lt. 0.0d0 ) call negfix( Fiss_inj(i),
1135:    &       'Fiss_inj(i)', 0.0d0 )
1136:*
1137:* recycle flow:
1138:*
1139:
1140:     Finj_rec(i) = fnon_iss * Finj_exh(i)
1141:     if( Finj_rec(i) .lt. 0.0d0 ) call negfix( Finj_rec(i),
1142:    &       'Finj_rec(i)', 0.0d0 )
1143:
1144:     if ( Finj_exh(i) .ne. 0.0d0 ) then
1145:
1146:       Xinj_t(i) = Finj_exh_t(i) / Finj_exh(i)
1147:
1148:       Xinj_d(i) = Finj_exh_d(i) / Finj_exh(i)
1149:
1150:       Xinj_p(i) = 1 - Xinj_t(i) - Xinj_d(i)
1151:
1152:     else
1153:
1154:       Xinj_t(i) = 0.0d0
1155:
1156:       Xinj_d(i) = 0.0d0
1157:
1158:       Xinj_p(i) = 0.0d0
1159:
1160:     endif
1161:*
1162:*#########################################################
1163:*## (Gas Puffer)
1164:*#########################################################
1165:*
1166:
1167:     Fgas_d(i) = Fgas_t(i)
1168:*-------------------------------------------------------------
1169:*
1170:*  Fgas_in_tri(t) & Fgas_in_deu(t) are determined by solving
1171:*  the following equation.
1172:*
1173:* Fgas_t = Fgas_in_tri(t) * Xsto_tri_t + Fgas_in_deu(t) * Xsto_deu_t
1174:*      + h1 * Fpuri_out(t0) * Xiss_in_t
1175:*      + mgas_mid_iss * Fiss_mid_(t0) * Xiss_mid_t
1176:*      + mgas_bot_iss * Fiss_bot_(t0) * Xiss_bot_t ;
1177:*
1178:* Fgas_d = Fgas_in_tri(t) * Xsto_tri_d + Fgas_in_deu(t) * Xsto_deu_d
1179:*      + h1 * Fpuri_out(t0) * Xiss_in_d
1180:*      + mgas_mid_iss * Fiss_mid_(t0) * Xiss_mid_d
1181:*      + mgas_bot_iss * Fiss_bot_(t0) * Xiss_bot_d ;
1182:*
1183:*-------------------------------------------------------------
1184:*  ... solution ...
1185:*
1186:
1187:     B1 = Fgas_t(i) - ( h1 * Fpuri_out(i-1) * Xiss_in_t(i-1) +
1188:    &       mgas_mid_iss * Fiss_mid_(i-1) * Xiss_mid_t + mgas_bot_iss
1189:    &       * Fiss_bot_(i-1) * Xiss_bot_t )
1190:
1191:     B2 = Fgas_d(i) - ( h1 * Fpuri_out(i-1) * Xiss_in_d(i-1) +
1192:    &       mgas_mid_iss * Fiss_mid_(i-1) * Xiss_mid_d + mgas_bot_iss
1193:    &       * Fiss_bot_(i-1) * Xiss_bot_d )
1194:
1195:     DD = Xsto_tri_t(i-1) * Xsto_deu_d(i-1) - Xsto_deu_t(i-1) *
1196:    &       Xsto_tri_d(i-1)
1197:
1198:     if ( abs ( DD ) .gt. 1.0d-10 ) then
1199:
1200:       Fgas_in_tri(i) = ( B1 * Xsto_deu_d(i-1) - B2 * Xsto_deu_t(i-1)
1201:    &       ) / DD
1202:       if( Fgas_in_tri(i) .lt. 0.0d0 ) call negfix( Fgas_in_tri(i),
1203:    &       'Fgas_in_tri(i)', 0.0d0 )
1204:
1205:       Fgas_in_deu(i) = ( B2 * Xsto_tri_t(i-1) - B1 * Xsto_tri_d(i-1)
1206:    &       ) / DD
1207:       if( Fgas_in_deu(i) .lt. 0.0d0 ) call negfix( Fgas_in_deu(i),
1208:    &       'Fgas_in_deu(i)', 0.0d0 )
1209:
1210:     else
1211:
1212:       Fgas_in_tri(i) = ( Fgas_t(i) - ( Fgas_in_deu(i-1) *
1213:    &       Xsto_deu_t(i-1) + h1 * Fpuri_out(i-1) * Xiss_in_t(i-1) +
1214:    &       mgas_mid_iss * Fiss_mid_(i-1) * Xiss_mid_t + mgas_bot_iss
1215:    &       * Fiss_bot_(i-1) * Xiss_bot_t ) ) / Xsto_tri_t(i-1)
1216:
1217:       Fgas_in_deu(i) = ( Fgas_d(i) - ( Fgas_in_tri(i-1) *
1218:    &       Xsto_tri_d(i-1) + h1 * Fpuri_out(i-1) * Xiss_in_d(i-1) +
1219:    &       mgas_mid_iss * Fiss_mid_(i-1) * Xiss_mid_d + mgas_bot_iss
1220:    &       * Fiss_bot_(i-1) * Xiss_bot_d ) ) / Xsto_deu_d(i-1)
1221:
1222:     endif
1223:*
1224:* .Overall balance
1225:*
1226:
1227:     F(36) = ABX ( - Fgas_tot(i-1) + h1 * Fpuri_out(i-1) +
1228:    &       mgas_mid_iss * Fiss_mid_(i-1) + mgas_bot_iss *
1229:    &       Fiss_bot_(i-1) + Fgas_in_tri(i-1) + Fgas_in_deu(i-1) ,
1230:    &       0.0d0 , Ngas_tot(i-1) )
1231:     call ABXdelta( - Fgas_tot(i-1) + h1 * Fpuri_out(i-1) +
1232:    &       mgas_mid_iss * Fiss_mid_(i-1) + mgas_bot_iss *
1233:    &       Fiss_bot_(i-1) + Fgas_in_tri(i-1) + Fgas_in_deu(i-1) ,
1234:    &       0.0d0 , Ngas_tot, i, delta_t, 'Ngas_tot')
1235:* .tritium balance
1236:
1237:     F(37) = ABX ( - Fgas_t(i-1) + h1 * Fpuri_out(i-1) *
1238:    &       Xiss_in_t(i-1) + mgas_mid_iss * Fiss_mid_(i-1) *
1239:    &       Xiss_mid_t + mgas_bot_iss * Fiss_bot_(i-1) * Xiss_bot_t +
1240:    &       Fgas_in_tri(i-1) * Xsto_tri_t(i-1) + Fgas_in_deu(i-1) *
1241:    &       Xsto_deu_t(i-1) , 0.0d0 , Tgas_(i-1) )
1242:     call ABXdelta( - Fgas_t(i-1) + h1 * Fpuri_out(i-1) *
1243:    &       Xiss_in_t(i-1) + mgas_mid_iss * Fiss_mid_(i-1) *
1244:    &       Xiss_mid_t + mgas_bot_iss * Fiss_bot_(i-1) * Xiss_bot_t +
1245:    &       Fgas_in_tri(i-1) * Xsto_tri_t(i-1) + Fgas_in_deu(i-1) *
1246:    &       Xsto_deu_t(i-1) , 0.0d0 , Tgas_, i, delta_t, 'Tgas_')
```

```
1247:* mobilizable tritium:
1248:
1249:     Tgas_mob(i) = mu_gas * Tgas_(i)
1250:* .deuterium balance
1251:
1252:     F(38) = ABX ( - Fgas_d(i-1) + h1 * Fpuri_out(i-1) *
1253:    &       Xiss_in_d(i-1) + mgas_mid_iss * Fiss_mid_(i-1) *
1254:    &       Xiss_mid_d + mgas_bot_iss * Fiss_bot_(i-1) * Xiss_bot_d +
1255:    &       Fgas_in_tri(i-1) * Xsto_tri_d(i-1) + Fgas_in_deu(i-1) *
1256:    &       Xsto_deu_d(i-1) , 0.0d0 , Dgas_(i-1) )
1257:     call ABXdelta( - Fgas_d(i-1) + h1 * Fpuri_out(i-1) *
1258:    &       Xiss_in_d(i-1) + mgas_mid_iss * Fiss_mid_(i-1) *
1259:    &       Xiss_mid_d + mgas_bot_iss * Fiss_bot_(i-1) * Xiss_bot_d +
1260:    &       Fgas_in_tri(i-1) * Xsto_tri_d(i-1) + Fgas_in_deu(i-1) *
1261:    &       Xsto_deu_d(i-1) , 0.0d0 , Dgas_, i, delta_t, 'Dgas_')
1262:* mole fraction:
1263:
1264:     Xgas_t(i) = Tgas_(i) / Ngas_tot(i)
1265:
1266:     Xgas_d(i) = Dgas_(i) / Ngas_tot(i)
1267:
1268:     Xgas_p(i) = 1 - Xgas_t(i) - Xgas_d(i)
1269:* .protium balance
1270:
1271:     Pgas_(i) = Xgas_p(i) * Ngas_tot(i)
1272:* total flow to plasma
1273:
1274:     Fgas_tot(i) = ( Fgas_t(i) + Fgas_d(i) ) / ( Xgas_t(i) + Xgas_d(i)
1275:    &       )
1276:     if( Fgas_tot(i) .lt. 0.0d0 ) call negfix( Fgas_tot(i),
1277:    &       'Fgas_tot(i)', 0.0d0 )
1278:*
1279:* The total flow to be puffered into the plasma includes
1280:* an amount of pellet injector propellant:
1281:*
1282:
1283:     Fpuff_plas(i) = Fgas_tot(i) + Fplas_prop_t(i) + Fplas_prop_d(i) +
1284:    &       Fplas_prop_p(i)
1285:     if( Fpuff_plas(i) .lt. 0.0d0 ) call negfix( Fpuff_plas(i),
1286:    &       'Fpuff_plas(i)', 0.0d0 )
1287:*
1288:*#############################################################
1289:*###( Storage and external supply)
1290:*#############################################################
1291:*
1292:*  ---------tritium rich storage
1293:*
1294:
1295:     Fsto_tri_in(i) = ( 1 - linj_bot_iss - mgas_bot_iss ) *
1296:    &       Fiss_bot_(i-1)
1297:     if( Fsto_tri_in(i) .lt. 0.0d0 ) call negfix( Fsto_tri_in(i),
1298:    &       'Fsto_tri_in(i)', 0.0d0 )
1299:
1300:     Fsto_tri_out(i) = Fsto_out_t(i-1) + Fgas_in_tri(i-1)
1301:     if( Fsto_tri_out(i) .lt. 0.0d0 ) call negfix( Fsto_tri_out(i),
1302:    &       'Fsto_tri_out(i)', 0.0d0 )
1303:* overall balance:
1304:
1305:     F(39) = ABX ( Fsto_tri_in(i-1) - Fsto_tri_out(i-1) , 0.0d0 ,
1306:    &       Nsto_tri_(i-1) )
1307:     call ABXdelta( Fsto_tri_in(i-1) - Fsto_tri_out(i-1) , 0.0d0 ,
1308:    &       Nsto_tri_, i, delta_t, 'Nsto_tri_')
1309:* total tritium:
1310:
1311:     F(40) = ABX ( Fsto_tri_in(i-1) * Xiss_bot_t , - Fsto_tri_out(i-1)
1312:    &       / Nsto_tri_(i-1) , Tsto_tri_(i-1) )
1313:     call ABXdelta( Fsto_tri_in(i-1) * Xiss_bot_t , -
1314:    &       Fsto_tri_out(i-1) / Nsto_tri_(i-1)  , Tsto_tri_, i,
1315:    &       delta_t, 'Tsto_tri_')
1316:* deuterium balance:
1317:
1318:     F(41) = ABX ( Fsto_tri_in(i-1) * Xiss_bot_d , - Fsto_tri_out(i-1)
1319:    &       / Nsto_tri_(i-1) , Dsto_tri_(i-1) )
1320:     call ABXdelta( Fsto_tri_in(i-1) * Xiss_bot_d , -
1321:    &       Fsto_tri_out(i-1) / Nsto_tri_(i-1)  , Dsto_tri_, i,
1322:    &       delta_t, 'Dsto_tri_')
1323:*
1324:*  --------- Deuterium rich storage
1325:*
1326:
1327:     Fsto_deu_in(i) = ( 1 - kiss_mid_ - linj_mid_iss - mgas_mid_iss )
1328:    &       * Fiss_mid_(i)
1329:     if( Fsto_deu_in(i) .lt. 0.0d0 ) call negfix( Fsto_deu_in(i),
1330:    &       'Fsto_deu_in(i)', 0.0d0 )
1331:
1332:     Fsto_deu_out(i) = Fsto_out_d(i-1) + Fgas_in_deu(i-1) +
1333:    &       Fin_st_d(i-1)
1334:     if( Fsto_deu_out(i) .lt. 0.0d0 ) call negfix( Fsto_deu_out(i),
1335:    &       'Fsto_deu_out(i)', 0.0d0 )
1336:* overall balance:
1337:
1338:     F(42) = ABX ( Fsto_deu_in(i-1) - Fsto_deu_out(i-1) , 0.0d0 ,
1339:    &       Nsto_deu_(i-1) )
1340:     call ABXdelta( Fsto_deu_in(i-1) - Fsto_deu_out(i-1) , 0.0d0 ,
1341:    &       Nsto_deu_, i, delta_t, 'Nsto_deu_')
1342:* total tritium:
1343:
1344:     F(43) = ABX ( Fsto_deu_in(i-1) * Xiss_mid_t , - Fsto_deu_out(i-1)
1345:    &       / Nsto_deu_(i-1) , Tsto_deu_(i-1) )
1346:     call ABXdelta( Fsto_deu_in(i-1) * Xiss_mid_t , -
1347:    &       Fsto_deu_out(i-1) / Nsto_deu_(i-1)  , Tsto_deu_, i,
1348:    &       delta_t, 'Tsto_deu_')
1349:* deuterium balance:
1350:
1351:     F(44) = ABX ( Fsto_deu_in(i-1) * Xiss_bot_d , - Fsto_deu_out(i-1)
1352:    &       / Nsto_deu_(i-1) , Dsto_deu_(i-1) )
1353:     call ABXdelta( Fsto_deu_in(i-1) * Xiss_bot_d , -
1354:    &       Fsto_deu_out(i-1) / Nsto_deu_(i-1)  , Dsto_deu_, i,
1355:    &       delta_t, 'Dsto_deu_')
1356:*
1357:*  --------- Protium rich storage
1358:*
1359:
1360:     Fsto_pro_in(i) = ( 1.0 - linj_p ) * Fiss_top_(i)
```

```
1361:     if( Fsto_pro_in(i) .lt. 0.0d0 ) call negfix( Fsto_pro_in(i),
1362:    &       'Fsto_pro_in(i)', 0.0d0 )
1363:
1364:     Fsto_pro_out(i) = Fsto_out_p(i-1)
1365:     if( Fsto_pro_out(i) .lt. 0.0d0 ) call negfix( Fsto_pro_out(i),
1366:    &       'Fsto_pro_out(i)', 0.0d0 )
1367:* overall balance:
1368:
1369:     F(45) = ABX ( Fsto_pro_in(i-1) - Fsto_pro_out(i-1) , 0.0d0 ,
1370:    &       Nsto_pro_(i-1) )
1371:     call ABXdelta( Fsto_pro_in(i-1) - Fsto_pro_out(i-1) , 0.0d0 ,
1372:    &       Nsto_pro_, i, delta_t, 'Nsto_pro_')
1373:* total tritium:
1374:
1375:     F(46) = ABX ( Fsto_pro_in(i-1) * Xiss_top_t , - Fsto_pro_out(i-1)
1376:    &       / Nsto_pro_(i-1) , Tsto_pro_(i-1) )
1377:     call ABXdelta( Fsto_pro_in(i-1) * Xiss_top_t , -
1378:    &       Fsto_pro_out(i-1) / Nsto_pro_(i-1) , Tsto_pro_, i,
1379:    &       delta_t, 'Tsto_pro_')
1380:* deuterium balance:
1381:
1382:     F(47) = ABX ( Fsto_pro_in(i-1) * Xiss_top_d , - Fsto_pro_out(i-1)
1383:    &       / Nsto_pro_(i-1) , Dsto_pro_(i-1) )
1384:     call ABXdelta( Fsto_pro_in(i-1) * Xiss_top_d , -
1385:    &       Fsto_pro_out(i-1) / Nsto_pro_(i-1) , Dsto_pro_, i,
1386:    &       delta_t, 'Dsto_pro_')
1387:*  ... Ksto_tri_ : accumulated inventory change in storage
1388:
1389:     Ksto_tri_(i) = Ksto_tri_(i-1) + ( Tsto_tri_(i-1) - Tsto_tri_(i) )
1390:
1391:     Ksto_deu_(i) = Ksto_deu_(i-1) + ( Tsto_deu_(i-1) - Tsto_deu_(i) )
1392:
1393:     Ksto_pro_(i) = Ksto_pro_(i-1) + ( Tsto_pro_(i-1) - Tsto_pro_(i) )
1394:*
1395:*-----------------------------------------------------------------
1396:* Reset inventories in storage when power level changes
1397:*-----------------------------------------------------------------
1398:*
1399:
1400:     ttstep = 20d0 / 3600d0
1401:*
1402:*  ... add T/D/P on every 20 seconds ...
1403:*
1404:
1405:     if ( Initial_step .eq. 0 .and. abs ( tcycle / ttstep - nint (
1406:    &       tcycle / ttstep ) ) .lt. 0.00001 ) then
1407:
1408:       Tsto_tri_(i) = Tsto_tri_(i) + Burn_t(i) * ttstep
1409:
1410:       Nsto_tri_(i) = Nsto_tri_(i) + Burn_t(i) * ttstep
1411:
1412:       Dsto_deu_(i) = Dsto_deu_(i) + Burn_d(i) * ttstep
1413:
1414:       Nsto_deu_(i) = Nsto_deu_(i) + Burn_d(i) * ttstep
1415:
1416:       Psto_pro_(i) = Psto_pro_(i) + Reac_p(i) * ttstep
1417:
1418:       Nsto_pro_(i) = Nsto_pro_(i) + Reac_p(i) * ttstep
1419:
1420:     endif
1421:*
1422:* Mole fraction in tritium rich storage
1423:*
1424:
1425:     if ( Nsto_tri_(i) .ne. 0.0d0 ) then
1426:
1427:       Xsto_tri_t(i) = Tsto_tri_(i) / Nsto_tri_(i)
1428:
1429:       Xsto_tri_d(i) = Dsto_tri_(i) / Nsto_tri_(i)
1430:
1431:       Xsto_tri_p(i) = 1.0 - Xsto_tri_t(i) - Xsto_tri_d(i)
1432:
1433:     else
1434:
1435:       Xsto_tri_t(i) = 0.0d0
1436:
1437:       Xsto_tri_d(i) = 0.0d0
1438:
1439:       Xsto_tri_p(i) = 0.0d0
1440:
1441:     endif
1442:
1443:     Psto_tri_(i) = Xsto_tri_p(i) * Nsto_tri_(i)
1444:* mobilizable tritium:
1445:
1446:     Tsto_tri_m(i) = mu_sto * Tsto_tri_(i)
1447:*
1448:* Mole fraction in deuterium rich storage
1449:*
1450:
1451:     if ( Nsto_deu_(i) .ne. 0.0d0 ) then
1452:
1453:       Xsto_deu_t(i) = Tsto_deu_(i) / Nsto_deu_(i)
1454:
1455:       Xsto_deu_d(i) = Dsto_deu_(i) / Nsto_deu_(i)
1456:
1457:       Xsto_deu_p(i) = 1.0 - Xsto_deu_t(i) - Xsto_deu_d(i)
1458:
1459:     else
1460:
1461:       Xsto_deu_t(i) = 0.0d0
1462:
1463:       Xsto_deu_d(i) = 0.0d0
1464:
1465:       Xsto_deu_p(i) = 0.0d0
1466:
1467:     endif
1468:
1469:     Psto_deu_(i) = Xsto_deu_p(i) * Nsto_deu_(i)
1470:* mobilizable tritium:
1471:
1472:     Tsto_deu_m(i) = mu_sto * Tsto_deu_(i)
1473:* Mole fraction in deuterium rich storage
1474:
```

```
1475:     if ( Nsto_pro_(i) .ne. 0.0d0 ) then
1476:
1477:       Xsto_pro_t(i) = Tsto_pro_(i) / Nsto_pro_(i)
1478:
1479:       Xsto_pro_d(i) = Dsto_pro_(i) / Nsto_pro_(i)
1480:
1481:       Xsto_pro_p(i) = 1.0 - Xsto_pro_t(i) - Xsto_pro_d(i)
1482:
1483:     else
1484:
1485:       Xsto_pro_t(i) = 0.0d0
1486:
1487:       Xsto_pro_d(i) = 0.0d0
1488:
1489:       Xsto_pro_p(i) = 0.0d0
1490:
1491:     endif
1492:
1493:     Psto_pro_(i) = Xsto_pro_p(i) * Nsto_pro_(i)
1494:* mobilizable tritium:
1495:
1496:     Tsto_pro_m(i) = mu_sto * Tsto_pro_(i)
1497:*
1498:*  --- Tritium/Deuterium/Protium requirements
1499:*
1500:
1501:     Fsto_tri_out_t(i) = Fsto_tri_out(i) * Xsto_tri_t(i)
1502:
1503:     Fsto_deu_out_d(i) = Fsto_deu_out(i) * Xsto_deu_d(i)
1504:
1505:     Fsto_pro_out_p(i) = Fsto_out_p(i) * Xsto_pro_p(i)
1506:*
1507:*  ... Fueling ..........................
1508:*
1509:
1510:     Fuelf_t(i) = Fuel_t(i) - Fplas_nb_t(i-1) - Fplas_prop_t(i-1)
1511:     if( Fuelf_t(i) .lt. 0.0d0 ) call negfix( Fuelf_t(i),
1512:    &       'Fuelf_t(i)', 0.0d0 )
1513:
1514:     return
1515:     end
1516:C=================================================
1517:C=== Funtions ==================================
1518:C=================================================
1519:     subroutine negfix( X , name, fix )
1520:C -------------------------------------------------------
1521:C  negative value fixup with a message
1522:C -------------------------------------------------------
1523:     double precision X,fix
1524:     character*(*) name
1525:     if( X.lt.0.0d0 ) then
1526:       write(6,*) '!!! negative value of ',name,' (=',X,') is fixed',
1527:    &     ' to ',fix
1528:       X = fix
1529:     end if
1530:     return
1531:     end
```

```
1532:c
1533:     subroutine ABXdelta( A0,B0,X,i,delta_t, xname )
1534:cccc  subroutine ABXdelta( A,B,X,i,delta_t, xname )
1535:C -------------------------------------------------------
1536:C  time promotion from X(i-1) to X(i) when
1537:C    dX/dt = A + B*X
1538:C -------------------------------------------------------
1539:     implicit double precision(a-h,o-z)
1540:c
1541:     double precision a,b,delta_t
1542:     integer i
1543:     double precision X(0:*)
1544:     character*(*) xname
1545:C
1546:     factor = 3600.d0
1547:     A= A0/FACTOR
1548:     B= B0/FACTOR
1549:c
1550:C
1551:C -------------------------------------------------------
1552:C    analytic form
1553:C -------------------------------------------------------
1554:C
1555:*       if( abs(B) .gt. 1.0d-10*abs(A) ) then
1556:*         AB = A/B
1557:*         X(i) =  ( X(i-1) + AB ) * exp( B*delta_t ) - AB
1558:*       else
1559:*         X(i) = X(i-1) + (A + B*X(i-1))*delta_t
1560:*       endif
1561:C
1562:C -------------------------------------------------------
1563:C    explicit Euler
1564:C -------------------------------------------------------
1565:C
1566:*       X(i) =  X(i-1) + ( A + B*X(i-1) ) * delta_t
1567:C
1568:C -------------------------------------------------------
1569:C    implicit Euler
1570:C -------------------------------------------------------
1571:Ccccc  X(i) = X(i-1) + ( A + B*X(i) ) * delta_t
1572:C
1573:*       BB = 1.0d0 - B*delta_t
1574:*       if( abs(BB) .gt. 1.0d-30 ) then
1575:*         X(i) =  (X(i-1) + A* delta_t) / BB
1576:*       else
1577:*         X(i) =  X(i-1) + ( A + B*X(i-1) ) * delta_t
1578:*       endif
1579:C
1580:C -------------------------------------------------------
1581:C    like 4'th order Runge-Kutta
1582:C -------------------------------------------------------
1583:C
1584:       R1  =  delta_t * ( A + B*X(i-1) )
1585:       R2  =  delta_t * ( A + B * ( X(i-1) + 0.5d0*R1 ) )
1586:       R3  =  delta_t * ( A + B * ( X(i-1) + 0.5d0*R2 ) )
1587:       R4  =  delta_t * ( A + B * ( X(i-1) + R3 ) )
1588:       X(i) =  X(i-1) + (1d0/6d0) * ( R1 + 2*R2 + 2*R3 + R4 )
```

```
1589:C
1590:      return
1591:      end
1592:*  $Id: function.txt,v 1.6 1997/04/30 04:11:33 $
1593:*********************************************************************
1594:* function FPPOWER : Fusion power profile ( 0 < fppower < 1 )
1595:*
1596:*  Rump up & rumpdown are given in steps of 20 sec width.
1597:*********************************************************************
1598:      double precision function FPPOWER(TCYCLE,T_BURN,T_P_RU,T_P_RD)
1599:
1600:      double precision TCYCLE, T_BURN, T_P_RU, T_P_RD
1601:      double precision TSTEP,DNU,DND
1602:
1603:      parameter ( TSTEP = 20d0/3600d0 )
1604:      DNU = nint( T_P_RU / TSTEP )
1605:      DND = nint( T_P_RD / TSTEP )
1606:cc    write(6,*) '(fppower) tcycle t_burn t_p_ru t_p_rd tstep',
1607:cc   & tcycle*3600, t_burn*3600, t_p_ru*3600, t_p_rd*3600, tstep*3600
1608:
1609:      if ( TCYCLE.lt.T_P_RU*0.999 ) then
1610:        FPPOWER = int((TCYCLE+1.0d-9)/TSTEP) / DNU
1611:      else if ( TCYCLE.lt.(T_P_RU+T_BURN)*0.999 ) then
1612:        FPPOWER = 1.0D0
1613:      else
1614:        FPPOWER = int((T_P_RU+T_BURN+T_P_RD-TCYCLE-1.0d-9)/TSTEP)/DND
1615:      end if
1616:      return
1617:      end
1618:*********************************************************************
1619:* function FPWIDTH
1620:*  Step width when rump up & rumpdown are given in steps
1621:*********************************************************************
1622:      double precision function FPWIDTH(TCYCLE,T_BURN,T_P_RU,T_P_RD)
1623:
1624:      double precision TCYCLE, T_BURN, T_P_RU, T_P_RD
1625:      double precision TSTEP,DNU,DND
1626:      parameter ( TSTEP = 20d0/3600d0 )
1627:
1628:cc    write(6,*) '(fppower) tcycle t_burn t_p_ru t_p_rd tstep',
1629:cc   & tcycle*3600, t_burn*3600, t_p_ru*3600, t_p_rd*3600, tstep*3600
1630:
1631:      if ( TCYCLE.lt.T_P_RU*0.999 ) then
1632:        FPWIDTH = TSTEP
1633:      else if ( TCYCLE.lt.(T_P_RU+T_BURN)*0.999 ) then
1634:        FPWIDTH = T_BURN
1635:      else
1636:        FPWIDTH = TSTEP
1637:      end if
1638:      return
1639:      end
1640:*********************************************************************
1641:* function FPPOWER0
1642:*  rump up & rumdown are continuous.
1643:*********************************************************************
1644:      double precision function FPPOWER0(TCYCLE,T_BURN,T_P_RU,T_P_RD)
1645:
1646:      double precision TCYCLE, T_BURN, T_P_RU, T_P_RD
1647:
1648:      if ( TCYCLE.lt.T_P_RU ) then
1649:        FPPOWER0 = TCYCLE/T_P_RU
1650:      else if ( TCYCLE.lt.(T_P_RU+T_BURN) ) then
1651:        FPPOWER0 = 1.0D0
1652:      else
1653:        FPPOWER0 = (1.0D0 - (TCYCLE-(T_P_RU+T_BURN)) /T_P_RD)
1654:      end if
1655:      return
1656:      end
1657:*********************************************************************
1658:* function fnbpower : neutral beam injection profile.
1659:*********************************************************************
1660:      double precision function FNBPOWER(TCYCLE,T_BURN,T_P_RU,T_P_RD,
1661:     &       T_NB_RU,T_NB_RD)
1662:
1663:      double precision TCYCLE, T_BURN, T_P_RU, T_P_RD, T_NB_RU, T_NB_RD
1664:
1665:      if ( TCYCLE.lt.(T_P_RU-T_NB_RU) ) then
1666:        FNBPOWER    = 0.0D0
1667:      else if ( TCYCLE.lt.T_P_RU ) then
1668:        FNBPOWER    = (TCYCLE-(T_P_RU-T_NB_RU)) /(T_P_RU-T_NB_RU)
1669:      else if ( TCYCLE.lt.(T_P_RU+T_BURN) ) then
1670:        FNBPOWER    = 1.0D0
1671:      else if ( TCYCLE.lt.(T_P_RU+T_BURN+T_NB_RD) ) then
1672:        FNBPOWER    = 1.0D0 - (TCYCLE-(T_P_RU+T_BURN))/(T_P_RD-T_NB_RD)
1673:      else
1674:        FNBPOWER    = 0.0D0
1675:      end if
1676:      return
1677:      end
1678:*********************************************************************
1679:* function Sigma_tend : return sum of tend(j)
1680:*   ( currently returns zero )
1681:*********************************************************************
1682:      double precision function SIGMA_TEND(T,J)
1683:
1684:      double precision T
1685:      integer J
1686:C*
1687:C*  ... Sigma_tend(j) is always zero for 1997 case ( t <1000 sec)
1688:C*
1689:      SIGMA_TEND  = 0.0D0
1690:      return
1691:      end
1692:*********************************************************************
1693:*    function f(M) : 0.1 if Be is used, otherwise 1.0
1694:*********************************************************************
1695:      double precision function FM(M)
1696:
1697:      double precision M
1698:      integer NINF
1699:
1700:ccccc if ( M.eq.'Be' ) then
1701:      if ( M.eq.2 ) then
1702:        FM     = 0.1D0
```

```
1703:      else
1704:        FM     = 1.0d0
1705:      end if
1706:      return
1707:      end
1708:*************************************************************************
1709:*     function rho(M) : material density (kg/m3)
1710:*************************************************************************
1711:      double precision function RHO(M)
1712:
1713:      double precision M
1714:      integer NINRHO
1715:
1716:      NINRHO  = 1
1717:ccccc if ( M.eq.'C' ) then
1718:      if ( M.eq.1 ) then
1719:        RHO    = 2.3D3
1720:ccccc else if ( M.eq.'Be' ) then
1721:      else if ( M.eq.2 ) then
1722:        RHO    = 1.8D3
1723:ccccc else if ( M.eq.'W' ) then
1724:      else if ( M.eq.3 ) then
1725:        RHO    = 19.3D3
1726:      else
1727:        NINRHO  = 0
1728:      end if
1729:
1730:      if ( NINRHO.eq.0 ) then
1731:        write(6,*) 'xxx(function rho) no matching meterial ', ' <', M,
1732:     &          '>'
1733:        stop 888
1734:      end if
1735:      return
1736:      end
1737:*************************************************************************
1738:*     function Rc(M,T) : codeposition rate (kg tritium/burn*year)
1739:*************************************************************************
1740:      double precision function RC(MEDGE,TEDGE)
1741:
1742:      double precision MEDGE, TEDGE
1743:
1744:      NINRC   = 1
1745:ccccc if(Medge.eq.'C') then
1746:      if ( MEDGE.eq.1 ) then
1747:        if ( TEDGE.eq.150 ) then
1748:          RC     = 155
1749:        else if ( TEDGE.eq.100 ) then
1750:          RC     = 135
1751:        else if ( TEDGE.eq.60 ) then
1752:          RC     = 88
1753:        else if ( TEDGE.eq.130 ) then
1754:          RC     = 130
1755:        else
1756:          NINRC   = 0
1757:        end if
1758:cccccc  else if(Medge.eq.'Be') then
1759:      else if ( MEDGE.eq.2 ) then
1760:        if ( TEDGE.eq.150 ) then
1761:          RC     = 40
1762:        else if ( TEDGE.eq.60 ) then
1763:          RC     = 33
1764:        else if ( TEDGE.eq.30 ) then
1765:          RC     = 18
1766:        else
1767:          NINRC   = 0
1768:        end if
1769:cccc    else if(Medge.eq.'W') then
1770:      else if ( MEDGE.eq.3 ) then
1771:        if ( TEDGE.eq.40 ) then
1772:          RC     = 0.005
1773:        else
1774:          NINRC   = 0
1775:        end if
1776:      else
1777:        NINRC   = 0
1778:      end if
1779:
1780:      if ( NINRC.eq.0 ) then
1781:        write(6,*) 'xxx(function Rc) no matching meterial or',
1782:     &          ' temperature <', MEDGE, '> T= ', TEDGE
1783:        stop 888
1784:      end if
1785:      return
1786:      end
1787:*************************************************************************
1788:*     function Ed(M,T)
1789:*************************************************************************
1790:      double precision function ED(MEDGE,TEDGE)
1791:
1792:      double precision MEDGE, TEDGE
1793:      NINED   = 1
1794:
1795:ccccc if(Medge.eq.'C') then
1796:      if ( MEDGE.eq.1 ) then
1797:        if ( TEDGE.eq.150D0 ) then
1798:          ED     = 0.27D0
1799:        else if ( TEDGE.eq.100D0 ) then
1800:          ED     = 0.28D0
1801:        else if ( TEDGE.eq.60D0 ) then
1802:          ED     = 0.23D0
1803:        else if ( TEDGE.eq.30D0 ) then
1804:          ED     = 0.17D0
1805:        else
1806:          NINED   = 0
1807:        end if
1808:ccccc else if(Medge.eq.'Be') then
1809:      else if ( MEDGE.eq.2 ) then
1810:        if ( TEDGE.eq.150D0 ) then
1811:          ED     = 0.46D0
1812:        else if ( TEDGE.eq.60D0 ) then
1813:          ED     = 0.73D0
1814:        else if ( TEDGE.eq.30D0 ) then
1815:          ED     = 0.49D0
1816:        else
```

```
1817:          NINED   = 0
1818:        end if
1819:ccccc else if(Medge.eq.'W') then
1820:      else if ( MEDGE.eq.3 ) then
1821:        if ( TEDGE.eq.40D0 ) then
1822:          ED      = 0.007D0
1823:        else
1824:          NINED   = 0
1825:        end if
1826:      else
1827:        NINED   = 0
1828:      end if
1829:
1830:      if ( NINED.eq.0 ) then
1831:        write(6,*) 'xxx(function Ed) no matching meterial or',
1832:     &          ' temperature: M = <', MEDGE, '> T= ', TEDGE
1833:        stop 888
1834:      end if
1835:      return
1836:      end
1837:
1838:*************************************************************************
1839:*     function I(T,t)
1840:*
1841:* Tritium inventry from diverter by log-linear interpolation.
1842:*************************************************************************
1843:      double precision function AI(TEMP,T)
1844:
1845:      double precision TEMP, T
1846:      parameter( NUMT = 6 )
1847:      parameter( NTEMP   = 3 )
1848:      double precision TI(NUMT)
1849:      double precision IV(NUMT,NTEMP)
1850:      double precision TEMPS(NTEMP)
1851:c ------------------------------------------------------------------
1852:      data TI /278D0, 417D0, 694D0, 1.39D3, 2.78D3, 4.86D3/
1853:
1854:      data TEMPS(1) /1000.D0/
1855:      data(IV(I,1),I=1,NUMT) /14D0, 260D0, 400D0, 420D0, 440D0, 460D0/
1856:      data TEMPS(2) /1400.D0/
1857:      data(IV(I,2),I=1,NUMT) /14D0, 260D0, 360D0, 380D0, 400D0, 410D0/
1858:      data TEMPS(3) /1800.D0/
1859:      data(IV(I,3),I=1,NUMT) /14D0, 260D0, 360D0, 380D0, 400D0, 410D0/
1860:c ------------------------------------------------------------------
1861:
1862:      if ( TEMP.eq.TEMPS(1) ) then
1863:        K      = 1
1864:      else if ( TEMP.eq.TEMPS(2) ) then
1865:        K      = 2
1866:      else if ( TEMP.eq.TEMPS(3) ) then
1867:        K      = 3
1868:      else
1869:        write(6,*) 'xxx(function ai) no matching temperature ', TEMP
1870:        stop 888
1871:      end if
1872:
1873:      do II = 1, NUMT
1874:        if ( TI(II).ge.T ) go to 100
1875:      end do
1876:      II      = NUMT - 1
1877:  100 continue
1878:C
1879:      AI      = EXP((LOG(IV(II+1,K))-LOG(IV(II,K)))/(TI(II+1)-TI(II))*
1880:     &        (T-TI(II))+LOG(IV(II,K)))
1881:C
1882:      return
1883:      end
```

_output.f

```
 1:      subroutine output
 2:c=================================================================
 3:c       Fusion reactor tritium inventory evaluation system
 4:c-----------------------------------------------------------------
 5:c   output calculated results
 6:c-----------------------------------------------------------------
 7:c $Id: _output.f,v 1.9 1997/05/07 02:18:35 $
 8:c=================================================================
 9:c
10:cccc  implicit real*8 (a-h,o-z)
11:c
12:      include '_sizes.h'
13:      include '_commons1.h'
14:      include 'varconst.h'
15:c
16:      logical opd
17:c -----------------------------------------------------------
18:c
19:c  .... one file per variable  output
20:c
21:      write(6,*) '------- Output results of simulation on files ----'
22:      write(6,*) ' nvarout= ', nvarout
23:c
24:      iunit =10
25:      inquire( unit=iunit, opened=opd )
26:      if( opd ) close (iunit)
27:c
28:      if(OUTPUTALL.eq.0) then
29:c
30:        do i=1,nvarout
31:          call fileoutput( varout(i), ivarout(i), iunit, ierr )
32:        end do
33:c
34:      else
35:c
36:        do i=1,nvar
37:          kk = index(varname(i), '(t)')
38:          if( kk.ne.0 ) then
39:            kk =kk - 1
40:          else
41:            kk = len(varname(i))
42:          endif
43:          call fileoutput( varname(i)(:kk), i, iunit, ierr )
44:        end do
45:c
46:      end if
47:c
48:      return
49:      end
50:c
51:c
52:c
53:      subroutine fileoutput( outvar, ivaro, iunit, ierr )
54:c=================================================================
55:c   output results of variable named outvar and sequential # ivaro
56:c   on I/O unit iunit.
57:c=================================================================
58:      character*(*) outvar
59:      integer      ivaro
60:c
61:      include '_sizes.h'
62:      include '_commons1.h'
63:      include 'varconst.h'
64:c
65:      character*32 unit
66:c
67:      do i=len(outvar),1,-1
68:        if( outvar(i:i).ne.' ' ) then
69:          lv = i
70:          goto 110
71:        end if
72:      end do
73:      lv = len(outvar)
74:  110 continue
75:c
76:      call fileopen( result(:llr), outvar(:lv), iunit, ierr )
77:      if(ierr.ne.0) then
78:        write(6,*) 'XXX failed to open output file ',outvar(:lv)
79:        if( sysname.eq.'Macintosh' ) pause 'PAUSE'
80:        stop 888
81:      end if
82:c
83:      unit = ' '
84:      if( outvar(1:1).eq.'F' .or.
85:     &   outvar(1:1).eq.'R' ) then
86:        unit = 'Mole/hour'
87:      else if( outvar(1:1).eq.'T' .or.
88:     &        outvar(1:1).eq.'D' .or.
89:     &        outvar(1:1).eq.'P' .or.
90:     &        outvar(1:1).eq.'N' .or.
91:     &        outvar(1:1).eq.'C' .or.
92:     &        outvar(1:1).eq.'H' .or.
93:     &        outvar(1:1).eq.'A' ) then
94:        unit = 'Moles'
95:      else if( outvar(1:4).eq.'Burn' ) then
96:        unit = 'Mega Watts'
97:      else
98:        unit = 'arbitrary unit'
99:      end if
100:c
101:      do i=len(unit),1,-1
102:        if( unit(i:i).ne.' ' ) then
103:          lu = i
104:          goto 130
105:        end if
106:      end do
107:      lu = len(unit)
108:  130 continue
```

```
c
c  ... char(9) is ASC II Tab character ...
c
      if(sysname.eq.'UNIX') then
        write(iunit,'(5a)') '#Time(sec)',char(9),outvar(:lv),char(9),
     &          unit(:lu)
      else
cc      write(iunit,'(5a)') 'Time(sec)',char(9),outvar(:lv),char(9),
c     &          unit(:lu)
        write(iunit,'(5a)') 'Time(sec)',char(9),outvar(:lv)
      endif

      do it=0,ntimes
        write(iunit,7100) times(it)*3600,char(9),XVARS(it,ivaro)
      end do
 7100 format(e12.5,a,1x,e12.5)
c
      close(iunit)
      return
      end
c ------------------------------------------------------------------------
c fileopen routine opens file under specified directory/folder
c ------------------------------------------------------------------------
      subroutine fileopen( path, fname, iunit, ierr )
c-----------------------------------------------------------------------
c Open a file named as "PATH + <derectory separator> + FNAME" on
c I/O unit IUNIT.
c
c  Return open error code on IERR
c-----------------------------------------------------------------------
      character*(*) fname, path
c
      include '_sizes.h'
      include '_commons1.h'
c
      character*256 ff
c
      ierr = 0
c
      ff = ' '
c
      if( path .eq. ' ' ) then
        ff = fname
      else
        lp = index(path,' ') - 1
        if(lp.lt.0) lp = len(path)
c
c  ... add directory separator ...
c
c  ":" for Macintosh, "/" for UNIX, "Y" or char(92) for MS-DOS/Windows
c
        if( sysname.eq.'Macintosh'.or.index(sysname,'Mac').ne.0
     &      .or. index(sysname,'MAC').ne.0 ) then

          ff = path(:len(path))//':'//fname

        else if( sysname.eq.'UNIX' .or. index(sysname,'UNIX').ne.0)
     &      then

          ff = path(:len(path))//'/'//fname

        else if( sysname.eq.'Windows'
     &      .or. index(sysname,'Win').ne.0
     &      .or. index(sysname,'WIN').ne.0
     &      .or. index(sysname,'DOS').ne.0
     &      ) then

          ff = path(:len(path))//char(92)//fname

        else

          write(6,*) 'XXX cannot compose file pathname because of',
     &      ' unknown system <',sysname,'> is specified.'
          stop 888

        end if
      end if
c
      lf = index(ff,' ') - 1
      if(lf.lt.0) lf = len(ff)
      do lf=len(ff),1,-1
        if( ff(lf:lf).ne.' ' ) goto 200
      end do
      lf = len(ff)
  200 continue
c
      write(6,*) '--- opening a file <',ff(:lf),'> on I/O unit ',
     &          iunit
c
      if(sysname.eq.'Macintosh') then
        write(*,*) '--- opening a file <',ff(:lf),'> on I/O unit ',
     &          iunit
      end if
c
c
      open( iunit, file=ff(:lf),  status='UNKNOWN', form='FORMATTED',
     &      iostat = ios )
c
c
      if( ios.ne.0 ) then
        write(6,*) '--- failed to open a file <',ff(:lf),'> code=',ios
        ierr = ios

        if(sysname.eq.'Macintosh') then
          write(*,*) '--- failed to open a file <',ff(:lf),'> code=',
     &              ios
        end if
      end if
c
      return
      end
```

varconst.h

```
C ---- varconst.h
C
C --- variables and constants ---
C
C    Creation date : << Tue May 6 14:21:39 1997 >>
C
C ... variables ...
C
      double precision XVARS(0:MAXTIMES,0:0)
C  ... XVARS can be used to refer variables with variable sequential no.
C  ... if necessary.
      common /VARIABLES/ XVARS
C
C ... variable No. 1 to 20 ...
      double precision
     &      Pfusion(0:MAXTIMES), Fuelf_t(0:MAXTIMES),
     &      Fuel_t(0:MAXTIMES), Fuel_d(0:MAXTIMES),
     &      Fuel_p(0:MAXTIMES), Burn_t(0:MAXTIMES),
     &      Burn_d(0:MAXTIMES), Reac_p(0:MAXTIMES),
     &      Efusion(0:MAXTIMES), Fex_t(0:MAXTIMES),
     &      Fex_d(0:MAXTIMES), Fex_p(0:MAXTIMES), Fhe_ex(0:MAXTIMES),
     &      i_t(0:MAXTIMES), i_d(0:MAXTIMES), i_p(0:MAXTIMES),
     &      Adiv_t(0:MAXTIMES), Afw_t(0:MAXTIMES),
     &      d_Adiv_t(0:MAXTIMES), d_Afw_t(0:MAXTIMES)
      common /VARIABLES/
     &      Pfusion, Fuelf_t, Fuel_t, Fuel_d, Fuel_p, Burn_t, Burn_d,
     &      Reac_p, Efusion, Fex_t, Fex_d, Fex_p, Fhe_ex, i_t, i_d,
     &      i_p, Adiv_t, Afw_t, d_Adiv_t, d_Afw_t
C
C ... variable No. 21 to 40 ...
      double precision
     &      C_t(0:MAXTIMES), d_C_t(0:MAXTIMES), D_t(0:MAXTIMES),
     &      d_D_t(0:MAXTIMES), Adiv_d(0:MAXTIMES), Afw_d(0:MAXTIMES),
     &      d_Adiv_d(0:MAXTIMES), d_Afw_d(0:MAXTIMES),
     &      C_d(0:MAXTIMES), d_C_d(0:MAXTIMES), D_d(0:MAXTIMES),
     &      d_D_d(0:MAXTIMES), Adiv_p(0:MAXTIMES), Afw_p(0:MAXTIMES),
     &      d_Adiv_p(0:MAXTIMES), d_Afw_p(0:MAXTIMES),
     &      C_p(0:MAXTIMES), d_C_p(0:MAXTIMES), D_p(0:MAXTIMES),
     &      d_D_p(0:MAXTIMES)
      common /VARIABLES/
     &      C_t, d_C_t, D_t, d_D_t, Adiv_d, Afw_d, d_Adiv_d, d_Afw_d,
     &      C_d, d_C_d, D_d, d_D_d, Adiv_p, Afw_p, d_Adiv_p, d_Afw_p,
     &      C_p, d_C_p, D_p, d_D_p
C
C ... variable No. 41 to 60 ...
      double precision
     &      Tpfc_(0:MAXTIMES), Tpfc_m(0:MAXTIMES), Dpfc_(0:MAXTIMES),
     &      Ppfc_(0:MAXTIMES), Tpv(0:MAXTIMES), Tp_reg(0:MAXTIMES),
     &      Tp_tot(0:MAXTIMES), Tp_tot_m(0:MAXTIMES),
     &      Rt_rem(0:MAXTIMES), Dpv(0:MAXTIMES), Rd_rem(0:MAXTIMES),
     &      Dp_reg(0:MAXTIMES), Dp_tot(0:MAXTIMES),
     &      Rp_rem(0:MAXTIMES), Pp_reg(0:MAXTIMES),
     &      Pp_tot(0:MAXTIMES), Ppv(0:MAXTIMES), Rhe_rem(0:MAXTIMES),
     &      Hepv(0:MAXTIMES), Hep_reg(0:MAXTIMES)
      common /VARIABLES/
     &      Tpfc_, Tpfc_m, Dpfc_, Ppfc_, Tpv, Tp_reg, Tp_tot,
     &      Tp_tot_m, Rt_rem, Dpv, Rd_rem, Dp_reg, Dp_tot, Rp_rem,
     &      Pp_reg, Pp_tot, Ppv, Rhe_rem, Hepv, Hep_reg
C
C ... variable No. 61 to 80 ...
      double precision
     &      Hep_tot(0:MAXTIMES), Fv_out(0:MAXTIMES),
     &      Fpuri_in(0:MAXTIMES), it_(0:MAXTIMES), id_(0:MAXTIMES),
     &      ip_(0:MAXTIMES), xnhydro_(0:MAXTIMES),
     &      xh_imp(0:MAXTIMES), Fadj_puri(0:MAXTIMES),
     &      Fhydro_pure(0:MAXTIMES), Fhydro_imp(0:MAXTIMES),
     &      Fnhydro_imp(0:MAXTIMES), Ft_pure(0:MAXTIMES),
     &      Fd_pure(0:MAXTIMES), Fp_pure(0:MAXTIMES),
     &      Fcq4_imp(0:MAXTIMES), Fnq3_imp(0:MAXTIMES),
     &      Fq2o_imp(0:MAXTIMES), Fhydro_imp_w(0:MAXTIMES),
     &      f_hydro_imp_w(0:MAXTIMES)
      common /VARIABLES/
     &      Hep_tot, Fv_out, Fpuri_in, it_, id_, ip_, xnhydro_,
     &      xh_imp, Fadj_puri, Fhydro_pure, Fhydro_imp, Fnhydro_imp,
     &      Ft_pure, Fd_pure, Fp_pure, Fcq4_imp, Fnq3_imp, Fq2o_imp,
     &      Fhydro_imp_w, f_hydro_imp_w
C
C ... variable No. 81 to 100 ...
      double precision
     &      Nimpt(0:MAXTIMES), Nhydro_imp(0:MAXTIMES),
     &      Nnhydro_imp(0:MAXTIMES), Ncq4_imp(0:MAXTIMES),
     &      Nnq3_imp(0:MAXTIMES), Nq2o_imp(0:MAXTIMES),
     &      Timp_(0:MAXTIMES), Dimp_(0:MAXTIMES), Pimp_(0:MAXTIMES),
     &      Fh2_puri_in(0:MAXTIMES), Ft2_sub_in(0:MAXTIMES),
     &      Fd2_sub_in(0:MAXTIMES), Fp2_sub_in(0:MAXTIMES),
     &      Fh2_sub_in(0:MAXTIMES), Ns(0:MAXTIMES),
     &      xt_sub(0:MAXTIMES), xd_sub(0:MAXTIMES),
     &      xp_sub(0:MAXTIMES), T2_sub(0:MAXTIMES),
     &      D2_sub(0:MAXTIMES)
      common /VARIABLES/
     &      Nimpt, Nhydro_imp, Nnhydro_imp, Ncq4_imp, Nnq3_imp,
     &      Nq2o_imp, Timp_, Dimp_, Pimp_, Fh2_puri_in, Ft2_sub_in,
     &      Fd2_sub_in, Fp2_sub_in, Fh2_sub_in, Ns, xt_sub, xd_sub,
     &      xp_sub, T2_sub, D2_sub
C
C ... variable No. 101 to 120 ...
      double precision
     &      P2_sub(0:MAXTIMES), Nhydro_sub(0:MAXTIMES),
     &      Ft_sub_out(0:MAXTIMES), Fd_sub_out(0:MAXTIMES),
     &      Fp_sub_out(0:MAXTIMES), Fh_sub_out(0:MAXTIMES),
     &      Treg_out(0:MAXTIMES), Ecq4_reg(0:MAXTIMES),
     &      Enq3_reg(0:MAXTIMES), Eq2o_reg(0:MAXTIMES),
     &      Ncq4_reg(0:MAXTIMES), Nnq3_reg(0:MAXTIMES),
     &      Nq2o_reg(0:MAXTIMES), Fhydro_pmem(0:MAXTIMES),
     &      Tpmem(0:MAXTIMES), Dpmem(0:MAXTIMES), Ppmem(0:MAXTIMES),
     &      it_endj(0:MAXTIMES), id_endj(0:MAXTIMES),
     &      ip_endj(0:MAXTIMES)
      common /VARIABLES/
```

```
109:     &       P2_sub, Nhydro_sub, Ft_sub_out, Fd_sub_out, Fp_sub_out,
110:     &       Fh_sub_out, Treq_out, Ecq4_req, Enq3_reg, Eq2o_reg,
111:     &       Ncq4_reg, Nnq3_reg, Nq2o_reg, Fhydro_pmem, Tpmem, Dpmem,
112:     &       Ppmem, it_endj, id_endj, ip_endj
113:
114:C ... variable No. 121 to 140 ...
115:      double precision
116:     &       Ft_puri_out(0:MAXTIMES), Fd_puri_out(0:MAXTIMES),
117:     &       Fp_puri_out(0:MAXTIMES), Fpuri_out(0:MAXTIMES),
118:     &       Xiss_in_t(0:MAXTIMES), Xiss_in_d(0:MAXTIMES),
119:     &       Xiss_in_p(0:MAXTIMES), Fpure_(0:MAXTIMES),
120:     &       Fimp_(0:MAXTIMES), Timp_m(0:MAXTIMES),
121:     &       Treq_out_m(0:MAXTIMES), Fiss_in(0:MAXTIMES),
122:     &       Fiss_nb(0:MAXTIMES), Fiss_in_t(0:MAXTIMES),
123:     &       Xiss_t(0:MAXTIMES), Xnb_t(0:MAXTIMES),
124:     &       Xinj_t(0:MAXTIMES), Fiss_in_d(0:MAXTIMES),
125:     &       Xiss_d(0:MAXTIMES), Xnb_d(0:MAXTIMES)
126:      common /VARIABLES/
127:     &       Ft_puri_out, Fd_puri_out, Fp_puri_out, Fpuri_out,
128:     &       Xiss_in_t, Xiss_in_d, Xiss_in_p, Fpure_, Fimp_, Timp_m,
129:     &       Treq_out_m, Fiss_in, Fiss_nb, Fiss_in_t, Xiss_t, Xnb_t,
130:     &       Xinj_t, Fiss_in_d, Xiss_d, Xnb_d
131:
132:C ... variable No. 141 to 160 ...
133:      double precision
134:     &       Xinj_d(0:MAXTIMES), Fiss_in_p(0:MAXTIMES),
135:     &       Niss_(0:MAXTIMES), Tiss_(0:MAXTIMES), Tiss_m(0:MAXTIMES),
136:     &       Diss_(0:MAXTIMES), Piss_(0:MAXTIMES), Xiss_p(0:MAXTIMES),
137:     &       Fiss_top_(0:MAXTIMES), Fiss_mid_(0:MAXTIMES),
138:     &       Fiss_bot_(0:MAXTIMES), Fiss_inj(0:MAXTIMES),
139:     &       Fplas_nb_d(0:MAXTIMES), Fin_st_d(0:MAXTIMES),
140:     &       Fin_rec_(0:MAXTIMES), Fin_iss_mid(0:MAXTIMES),
141:     &       Xcryo_d(0:MAXTIMES), W_nb(0:MAXTIMES),
142:     &       Ncryo_(0:MAXTIMES), Fbk_nb_(0:MAXTIMES)
143:      common /VARIABLES/
144:     &       Xinj_d, Fiss_in_p, Niss_, Tiss_, Tiss_m, Diss_, Piss_,
145:     &       Xiss_p, Fiss_top_, Fiss_mid_, Fiss_bot_, Fiss_inj,
146:     &       Fplas_nb_d, Fin_st_d, Fin_rec_, Fin_iss_mid, Xcryo_d,
147:     &       W_nb, Ncryo_, Fbk_nb_
148:
149:C ... variable No. 161 to 180 ...
150:      double precision
151:     &       Dcryo_(0:MAXTIMES), Dnb_reg_(0:MAXTIMES),
152:     &       Tnb_cryo(0:MAXTIMES), Fbk_nb_t(0:MAXTIMES),
153:     &       Rnb_reg_t(0:MAXTIMES), tnb_(0:MAXTIMES),
154:     &       Tnb_reg_(0:MAXTIMES), Tnb_cryo_tot(0:MAXTIMES),
155:     &       Tnb_cryo_mob(0:MAXTIMES), Dnb_cryo(0:MAXTIMES),
156:     &       Fbk_nb_d(0:MAXTIMES), Rnb_reg_d(0:MAXTIMES),
157:     &       Dnb_cryo_tot(0:MAXTIMES), Xnb_cryo_t(0:MAXTIMES),
158:     &       Xnb_cryo_d(0:MAXTIMES), Xnb_cryo_p(0:MAXTIMES),
159:     &       Pnb_cryo(0:MAXTIMES), Tnb_wall(0:MAXTIMES),
160:     &       Tnb_wall_mob(0:MAXTIMES), Tnb_tot(0:MAXTIMES)
161:      common /VARIABLES/
162:     &       Dcryo_, Dnb_reg_, Tnb_cryo, Fbk_nb_t, Rnb_reg_t, tnb_,
163:     &       Tnb_reg_, Tnb_cryo_tot, Tnb_cryo_mob, Dnb_cryo, Fbk_nb_d,
164:     &       Rnb_reg_d, Dnb_cryo_tot, Xnb_cryo_t, Xnb_cryo_d,
165:     &       Xnb_cryo_p, Pnb_cryo, Tnb_wall, Tnb_wall_mob, Tnb_tot
166:
167:C ... variable No. 181 to 200 ...
168:      double precision
169:     &       Tnb_tot_mob(0:MAXTIMES), Fnb_cryo(0:MAXTIMES),
170:     &       Crec(0:MAXTIMES), Fplas_nb_(0:MAXTIMES),
171:     &       Fplas_nb_t(0:MAXTIMES), Fplas_nb_p(0:MAXTIMES),
172:     &       ncycle_p(0:MAXTIMES), Nnb_cryo(0:MAXTIMES),
173:     &       Rnb_reg_(0:MAXTIMES), Rnb_reg_p(0:MAXTIMES),
174:     &       Fgas_t(0:MAXTIMES), Fgas_in_tri(0:MAXTIMES),
175:     &       Fgas_in_deu(0:MAXTIMES), Fgas_d(0:MAXTIMES),
176:     &       Ngas_tot(0:MAXTIMES), Fgas_tot(0:MAXTIMES),
177:     &       Tgas_(0:MAXTIMES), Dgas_(0:MAXTIMES), Xgas_t(0:MAXTIMES),
178:     &       Xgas_d(0:MAXTIMES)
179:      common /VARIABLES/
180:     &       Tnb_tot_mob, Fnb_cryo, Crec, Fplas_nb_, Fplas_nb_t,
181:     &       Fplas_nb_p, ncycle_p, Nnb_cryo, Rnb_reg_, Rnb_reg_p,
182:     &       Fgas_t, Fgas_in_tri, Fgas_in_deu, Fgas_d, Ngas_tot,
183:     &       Fgas_tot, Tgas_, Dgas_, Xgas_t, Xgas_d
184:
185:C ... variable No. 201 to 220 ...
186:      double precision
187:     &       Xgas_p(0:MAXTIMES), Pgas_(0:MAXTIMES),
188:     &       Fpuff_plas(0:MAXTIMES), Tgas_mob(0:MAXTIMES),
189:     &       Fplas_in_t(0:MAXTIMES), Fplas_in_d(0:MAXTIMES),
190:     &       Fplas_in_p(0:MAXTIMES), Fsto_out_t(0:MAXTIMES),
191:     &       Fsto_out_d(0:MAXTIMES), Fsto_out_p(0:MAXTIMES),
192:     &       Next_12(0:MAXTIMES), Text_12(0:MAXTIMES),
193:     &       Dext_12(0:MAXTIMES), Pext_12(0:MAXTIMES),
194:     &       Xext_12_t(0:MAXTIMES), Xext_12_d(0:MAXTIMES),
195:     &       Xext_12_p(0:MAXTIMES), Next_tot(0:MAXTIMES),
196:     &       Text_tot(0:MAXTIMES), Dext_tot(0:MAXTIMES)
197:      common /VARIABLES/
198:     &       Xgas_p, Pgas_, Fpuff_plas, Tgas_mob, Fplas_in_t,
199:     &       Fplas_in_d, Fplas_in_p, Fsto_out_t, Fsto_out_d,
200:     &       Fsto_out_p, Next_12, Text_12, Dext_12, Pext_12,
201:     &       Xext_12_t, Xext_12_d, Xext_12_p, Next_tot, Text_tot,
202:     &       Dext_tot
203:
204:C ... variable No. 221 to 240 ...
205:      double precision
206:     &       Pext_tot(0:MAXTIMES), Text_m(0:MAXTIMES),
207:     &       Vinj_t(0:MAXTIMES), Vinj_d(0:MAXTIMES),
208:     &       Vinj_p(0:MAXTIMES), Xextru_t(0:MAXTIMES),
209:     &       Xextru_d(0:MAXTIMES), Xextru_p(0:MAXTIMES),
210:     &       Nextru_(0:MAXTIMES), Vinj_tot(0:MAXTIMES),
211:     &       Dpellet(0:MAXTIMES), Hpellet(0:MAXTIMES),
212:     &       npellet_t(0:MAXTIMES), npellet_d(0:MAXTIMES),
213:     &       npellet_p(0:MAXTIMES), Rinj_(0:MAXTIMES),
214:     &       Finj_in_tot(0:MAXTIMES), Fbk_plas_(0:MAXTIMES),
215:     &       Fbk_plas_t(0:MAXTIMES), Fbk_plas_d(0:MAXTIMES)
216:      common /VARIABLES/
217:     &       Pext_tot, Text_m, Vinj_t, Vinj_d, Vinj_p, Xextru_t,
218:     &       Xextru_d, Xextru_p, Nextru_, Vinj_tot, Dpellet, Hpellet,
219:     &       npellet_t, npellet_d, npellet_p, Rinj_, Finj_in_tot,
220:     &       Fbk_plas_, Fbk_plas_t, Fbk_plas_d
221:
222:C ... variable No. 241 to 260 ...
```

```
223:      double precision
224:     &        Fbk_plas_p(0:MAXTIMES), Fplas_prop(0:MAXTIMES),
225:     &        Fplas_prop_t(0:MAXTIMES), Fplas_prop_d(0:MAXTIMES),
226:     &        Fplas_prop_p(0:MAXTIMES), Finj_exh(0:MAXTIMES),
227:     &        Finj_exh_t(0:MAXTIMES), Finj_exh_d(0:MAXTIMES),
228:     &        Xinj_p(0:MAXTIMES), Finj_prop(0:MAXTIMES),
229:     &        Xprop_t(0:MAXTIMES), Xprop_d(0:MAXTIMES),
230:     &        Xprop_p(0:MAXTIMES), Finj_rec(0:MAXTIMES),
231:     &        Fsto_tri_in(0:MAXTIMES), Fsto_tri_out(0:MAXTIMES),
232:     &        Nsto_tri_(0:MAXTIMES), Tsto_tri_(0:MAXTIMES),
233:     &        Tsto_tri_m(0:MAXTIMES), Dsto_tri_(0:MAXTIMES)
234:      common /VARIABLES/
235:     &        Fbk_plas_p, Fplas_prop, Fplas_prop_t, Fplas_prop_d,
236:     &        Fplas_prop_p, Finj_exh, Finj_exh_t, Finj_exh_d, Xinj_p,
237:     &        Finj_prop, Xprop_t, Xprop_d, Xprop_p, Finj_rec,
238:     &        Fsto_tri_in, Fsto_tri_out, Nsto_tri_, Tsto_tri_,
239:     &        Tsto_tri_m, Dsto_tri_
240:
241:C ... variable No. 261 to 280 ...
242:      double precision
243:     &        Psto_tri_(0:MAXTIMES), Xsto_tri_t(0:MAXTIMES),
244:     &        Xsto_tri_d(0:MAXTIMES), Xsto_tri_p(0:MAXTIMES),
245:     &        Fsto_deu_in(0:MAXTIMES), Fsto_deu_out(0:MAXTIMES),
246:     &        Nsto_deu_(0:MAXTIMES), Tsto_deu_(0:MAXTIMES),
247:     &        Tsto_deu_m(0:MAXTIMES), Dsto_deu_(0:MAXTIMES),
248:     &        Psto_deu_(0:MAXTIMES), Ksto_tri_(0:MAXTIMES),
249:     &        Ksto_deu_(0:MAXTIMES), Ksto_pro_(0:MAXTIMES),
250:     &        Xsto_deu_t(0:MAXTIMES), Xsto_deu_d(0:MAXTIMES),
251:     &        Xsto_deu_p(0:MAXTIMES), Fsto_pro_in(0:MAXTIMES),
252:     &        Fsto_pro_out(0:MAXTIMES), Finj_sto_pro(0:MAXTIMES)
253:      common /VARIABLES/
254:     &        Psto_tri_, Xsto_tri_t, Xsto_tri_d, Xsto_tri_p,
255:     &        Fsto_deu_in, Fsto_deu_out, Nsto_deu_, Tsto_deu_,
256:     &        Tsto_deu_m, Dsto_deu_, Psto_deu_, Ksto_tri_, Ksto_deu_,
257:     &        Ksto_pro_, Xsto_deu_t, Xsto_deu_d, Xsto_deu_p,
258:     &        Fsto_pro_in, Fsto_pro_out, Finj_sto_pro
259:
260:C ... variable No. 281 to 291 ...
261:      double precision
262:     &        Nsto_pro_(0:MAXTIMES), Tsto_pro_(0:MAXTIMES),
263:     &        Tsto_pro_m(0:MAXTIMES), Dsto_pro_(0:MAXTIMES),
264:     &        Psto_pro_(0:MAXTIMES), Xsto_pro_t(0:MAXTIMES),
265:     &        Xsto_pro_d(0:MAXTIMES), Xsto_pro_p(0:MAXTIMES),
266:     &        Fsto_tri_out_t(0:MAXTIMES), Fsto_deu_out_d(0:MAXTIMES),
267:     &        Fsto_pro_out_p(0:MAXTIMES)
268:      common /VARIABLES/
269:     &        Nsto_pro_, Tsto_pro_, Tsto_pro_m, Dsto_pro_, Psto_pro_,
270:     &        Xsto_pro_t, Xsto_pro_d, Xsto_pro_p, Fsto_tri_out_t,
271:     &        Fsto_deu_out_d, Fsto_pro_out_p
272:      parameter ( NVAR = 291 )
273:
274:      common /VARIABLES/ VARNAME(NVAR),VARDESC(NVAR),IVARFLAG(NVAR)
275:
276:      character*32 VARNAME
277:      character*64 VARDESC
278:      integer IVARFLAG
279:
280:C
281:      double precision C(0:0)
282:C ... C can be used to refer constants with sequential no.
283:C ... if necessary.
284:      common /CONSTANTS/ C
285:
286:C ... constants ...
287:
288:      double precision ap
289:      common /CONSTANTS/ ap
290:      double precision fbu
291:      common /CONSTANTS/ fbu
292:      double precision fin_pellet
293:      common /CONSTANTS/ fin_pellet
294:      double precision fin_gas
295:      common /CONSTANTS/ fin_gas
296:      double precision Temperature
297:      common /CONSTANTS/ Temperature
298:      double precision Tedge
299:      common /CONSTANTS/ Tedge
300:      double precision Medge
301:      common /CONSTANTS/ Medge
302:      double precision mu_A
303:      common /CONSTANTS/ mu_A
304:      double precision mu_C
305:      common /CONSTANTS/ mu_C
306:      double precision mu_D
307:      common /CONSTANTS/ mu_D
308:      double precision dc
309:      common /CONSTANTS/ dc
310:      double precision Vdiv
311:      common /CONSTANTS/ Vdiv
312:      double precision Vfw
313:      common /CONSTANTS/ Vfw
314:      double precision Ad
315:      common /CONSTANTS/ Ad
316:      double precision lambda_T
317:      common /CONSTANTS/ lambda_T
318:      double precision npump
319:      common /CONSTANTS/ npump
320:      double precision treg
321:      common /CONSTANTS/ treg
322:      double precision tpump
323:      common /CONSTANTS/ tpump
324:      double precision emr
325:      common /CONSTANTS/ emr
326:      double precision Rs
327:      common /CONSTANTS/ Rs
328:      double precision Area_p
329:      common /CONSTANTS/ Area_p
330:      double precision mu_v
331:      common /CONSTANTS/ mu_v
332:      double precision mu_mob
333:      common /CONSTANTS/ mu_mob
334:      double precision Lhydro_
335:      common /CONSTANTS/ Lhydro_
336:      double precision Limp_
```

```
      common /CONSTANTS/ Limp_
      double precision Ns_max
      common /CONSTANTS/ Ns_max
      double precision treg_msieve
      common /CONSTANTS/ treg_msieve
      double precision mu_req_m
      common /CONSTANTS/ mu_req_m
      double precision tau_pmem
      common /CONSTANTS/ tau_pmem
      double precision w1
      common /CONSTANTS/ w1
      double precision w2
      common /CONSTANTS/ w2
      double precision w3
      common /CONSTANTS/ w3
      double precision xcq4_
      common /CONSTANTS/ xcq4_
      double precision xnq3_
      common /CONSTANTS/ xnq3_
      double precision xq2o_
      common /CONSTANTS/ xq2o_
      double precision Fbl_cool
      common /CONSTANTS/ Fbl_cool
      double precision Fbl_
      common /CONSTANTS/ Fbl_
      double precision h1
      common /CONSTANTS/ h1
      double precision h3
      common /CONSTANTS/ h3
      double precision Fragnb
      common /CONSTANTS/ Fragnb
      double precision Fragiss_inj
      common /CONSTANTS/ Fragiss_inj
      double precision Xc_t
      common /CONSTANTS/ Xc_t
      double precision Xbl_t
      common /CONSTANTS/ Xbl_t
      double precision Xc_d
      common /CONSTANTS/ Xc_d
      double precision Xbl_d
      common /CONSTANTS/ Xbl_d
      double precision tau_iss_
      common /CONSTANTS/ tau_iss_
      double precision mu_iss_
      common /CONSTANTS/ mu_iss_
      double precision Xiss_top_t
      common /CONSTANTS/ Xiss_top_t
      double precision Xiss_top_d
      common /CONSTANTS/ Xiss_top_d
      double precision Xiss_top_p
      common /CONSTANTS/ Xiss_top_p
      double precision Xiss_mid_t
      common /CONSTANTS/ Xiss_mid_t
      double precision Xiss_mid_d
      common /CONSTANTS/ Xiss_mid_d
      double precision Xiss_mid_p
      common /CONSTANTS/ Xiss_mid_p
      double precision Xiss_bot_t
      common /CONSTANTS/ Xiss_bot_t
      double precision Xiss_bot_d
      common /CONSTANTS/ Xiss_bot_d
      double precision Xiss_bot_p
      common /CONSTANTS/ Xiss_bot_p
      double precision eta_nb_tot
      common /CONSTANTS/ eta_nb_tot
      double precision kiss_mid_
      common /CONSTANTS/ kiss_mid_
      double precision fcryo_
      common /CONSTANTS/ fcryo_
      double precision tnb_reg
      common /CONSTANTS/ tnb_reg
      double precision tnb_pump
      common /CONSTANTS/ tnb_pump
      double precision nnb_
      common /CONSTANTS/ nnb_
      double precision mu_nb_mob
      common /CONSTANTS/ mu_nb_mob
      double precision fnb_wall
      common /CONSTANTS/ fnb_wall
      double precision mu_wall_
      common /CONSTANTS/ mu_wall_
      double precision tcycle_p
      common /CONSTANTS/ tcycle_p
      double precision t_p_ru
      common /CONSTANTS/ t_p_ru
      double precision t_p_rd
      common /CONSTANTS/ t_p_rd
      double precision t_nb_ru
      common /CONSTANTS/ t_nb_ru
      double precision t_nb_rd
      common /CONSTANTS/ t_nb_rd
      double precision t_burn
      common /CONSTANTS/ t_burn
      double precision tcycle_nb
      common /CONSTANTS/ tcycle_nb
      double precision E_nb
      common /CONSTANTS/ E_nb
      double precision No
      common /CONSTANTS/ No
      double precision nedge_tot
      common /CONSTANTS/ nedge_tot
      double precision nedge_t
      common /CONSTANTS/ nedge_t
      double precision nedge_d
      common /CONSTANTS/ nedge_d
      double precision nedge_p
      common /CONSTANTS/ nedge_p
      double precision mgas_mid_iss
      common /CONSTANTS/ mgas_mid_iss
      double precision mgas_bot_iss
      common /CONSTANTS/ mgas_bot_iss
      double precision mu_gas
      common /CONSTANTS/ mu_gas
      double precision Vpellet
```

```
      common /CONSTANTS/ Vpellet
      double precision finj_err
      common /CONSTANTS/ finj_err
      double precision linj_mid_iss
      common /CONSTANTS/ linj_mid_iss
      double precision linj_bot_iss
      common /CONSTANTS/ linj_bot_iss
      double precision mu_ext
      common /CONSTANTS/ mu_ext
      double precision kpellet_t
      common /CONSTANTS/ kpellet_t
      double precision kpellet_d
      common /CONSTANTS/ kpellet_d
      double precision kpellet_p
      common /CONSTANTS/ kpellet_p
      double precision fcap_prop
      common /CONSTANTS/ fcap_prop
      double precision fnon_iss
      common /CONSTANTS/ fnon_iss
      double precision liss_p
      common /CONSTANTS/ liss_p
      double precision nprop
      common /CONSTANTS/ nprop
      double precision mu_sto
      common /CONSTANTS/ mu_sto
      double precision linj_p
      common /CONSTANTS/ linj_p
      parameter ( NCONST = 95)

      common /CONSTANTS/ CONSTNAME(NCONST),CONSTDESC(NCONST)

      character*32 CONSTNAME
      character*64 CONSTDESC

C ... dX/dt

      parameter ( NDFVAR = 47)

      common /BLOCK3/ IDFVAR(NDFVAR)

      integer IDFVAR

C ---- end of varconst.h
```

_commons1.h

```
C
C ------------------------------------------------------------------
C  Problem control variables
C ------------------------------------------------------------------
C------------------------------------------------------------------
C $Id: _commons1.h,v 1.6 1997/04/21 08:53:12 $
C------------------------------------------------------------------
C
      character*128 print
      character*128 result
      character*32 sysname
      common /FILES/ print, result, sysname
      common /FILESI/ llp, llr
C*****************************************************************
C*  ... file names
C*
C*   PRINT : standard output ( I/O unit 6 )
C*   RESULT : directory for result output  (I/O unit 10)
C*   SYSNAME : system type on which this program is running
C*            (Machintosh/UNIX/Windows/....)
C*
C*   LLP: length of effective part as name in PRINT
C*   LLR: length of effective part as name in RESULT
C*
C*****************************************************************
      character*32 METHOD
      integer JDEBUG, OUTPUTALL
      common /OPTION/ METHOD, JDEBUG, OUTPUTALL
C*****************************************************************
C*  ... options
C*
C*   METHOD : method of solution
C*        'RK' or ' ' : 4th order Runge-Kutta method.
C*        'EULER' : Explicit euler method.
C*   JDEBUG : debug output if nonzero.
C*
C*****************************************************************
CCC   parameter ( MAXTIMES = 1500, MAXTZONE = 32 )
C
      integer ntimes, ntimezone, nsteps,nsubsteps
      common /CONTROL/  ntimes, ntimezone, nsteps(MAXTZONE),
     &                  nsubsteps(MAXTZONE)
C
C*****************************************************************
C*  ... number of time steps etc.
C*
C*  NTIMES  : total number of calculation time steps.
C*
C*  NTIMEZONE  : total number of time zones.
C*           "Time zone" is a time range during which time steps
C*           have the same witdh.
```

```
52:C*          When ntimezone=0, all time meshs have the same width
53:C*          from t=0 to TMAX,
54:C*
55:C*   NSTEPS(NTIMEZONE)  : number of time steps of each time zone.
56:C*   NSUBSTEPS(NTIMEZONE) : number of sub-time steps of each time zone.
57:C*
58:C*****************************************************************
59:C
60:      double precision t, tmax,delta_t,times,tzbound
61:      common /DCONTROL/ t, tmax,delta_t,times(0:MAXTIMES),
62:     &                 tzbound(0:MAXTZONE)
63:C
64:C*****************************************************************
65:C*   ... floating point data
66:C*
67:C*   T     : current time
68:C*   TMAX  : max simulation time
69:C*   DELTA_T  : current time step width
70:C*   TIMES(0:NTIMES)  : time step boundaries
71:C*   TZNOUND(0:NTIMES)  : time zone boundaries
72:C*
73:C*****************************************************************
74:C
75:      parameter (maxvarout = 300)
76:      character*32 varout
77:      common /NMONITOR/ varout(maxvarout)
78:      integer nvarout, ivarout
79:      common /VMONITOR/ nvarout, ivarout(maxvarout)
80:C*****************************************************************
81:C*   ... Variables name of monitored variables etc.
82:C*
83:C*   NVAROUT  : number of variables whose value on each time step
84:C*              are output as results.
85:C*   VAROUT(NVAROUT)  : name of variable whose value is output.
86:C*   IVAROUT(NVAROUT)  : variable position no. in varname(nvar)
87:C*****************************************************************
```

_sizes.h

```
1:C
2:C   ... Problem size parameters
3:C
4:C  $Id: _sizes.h,v 1.3 1997/04/07 04:35:29 $
5:C
6:      parameter ( MAXTIMES = 2000 )
7:      parameter ( MAXTZONE = 64 )
```

_storage.h

```
1:      parameter (MAXSTORAGE= 50000)
2:      double precision D
3:      common /STORAGE/ D(MAXSTORAGE), LIMIT
4:C-----------------------------------------------------------------
5:C $Id: _storage.h,v 1.3 1997/04/06 05:51:58 $
6:C*****************************************************************
7:C*   ... data area to store calculated variable values etc.
8:C*
9:C*   D(*) : double precision memory area
10:C*  LIMIT : size of arry D(*)
11:C*
12:C*****************************************************************
13:C
```