47－S 004

オンラインシミュレーション言語 SIMBOL の評価

―遠隔情報処理システムの研究開発―

昭和48年３月

JIPDEC

財団法人 日本情報処理開発センター

この報告書は、日本自転車振興会から競輪収益の一部である機械工業振興資金の補助を受けて昭和４７年度に実施した「遠隔情報処理システムの研究開発」の一環としてとりまとめたものであります。

序

当財団は情報処理に関する研究開発の一環として遠隔情報処理システムに関する各種のソフトウエアの開発をおこなっておりますが、本書はタイムシェアリングシステムのもとにおける「オンライン　シミュレーション言語」の開発成果を報告するものであります。

シミュレーションはコンピュータの代表的なアプリケーションの一つでありますが、特に人間と機械の会話型形式においてより効果があがる技法であると言われております。しかしその反面演算速度、記憶容量などの点で比較的大型のコンピュータシステムを必要とするため、オンラインシステムとしてはまだ広い範囲の実用化の段階には到っておりません。

当財団では今後のコンピュータのオンライン利用促進の一環として3ケ年計画により我が国で最初の汎用オンラインシミュレーションシステムを開発いたしました。本年度はその最終年度にあたり、システムを完成させるとともに、バッチ処理用のシミュレーションシステムとの比較評価を実施いたしました。

本報告書が、この方面に興味を持つ方々に広く利用され、情報処理技術の発展の一助として寄与できることを念願いたす次第であります。

昭和48年3月

財団法人　日本情報処理開発センター

会長　　難・波　捷　吾

ま　え　が　き

現在、離散系システムの動的過程を解析するためのシミュレーション言語として、GPSS、SIMSCRIPTあるいはSIMULAなど、数多くの言語が作成され、使用されている。しかしながら、これらは全てバッチ処理の形態をとるものである。

取り扱う問題領域が拡大され、問題解決を図るレベルが高度化されるに従って、様々な利用者が多様な角度から、多様な要求をシミュレーション言語に対し抱くようになる。汎用システムとしての体型を崩さない範囲で、豊富なメモリー・スペースと迅速な応答速度を提供する事は、設計における大きなトレード・オフとなる。

人間と計算機とが対話しながらシミュレーションが行なえるようなオンライン・インタラクティブ・システムとして開発されたSIMBOL（Simulation Model Builder for On−Line Usage）は、このような多様な要求の一端を解決する事を目的に設計されたものである。オンライン利用を主眼としているために、必然的に従来のバッチ処理下でのシミュレーション言語とは機能的にも異なるものが要求され、多くの試行錯誤と選択が成されて来た。

本報告書は、そのような多くの設計上のトライアルが果して妥当であったか否かを、既存のバッチ処理シミュレーション言語との比較に於て考察したものである。シミュレーション言語の一般論的比較評価に至らないまでも、本報告書が今後のシミュレーション言語のオンライン化に際し、何等かの資となれば幸いである。

なおこのプロジェクトは前年度からの継続であり、46年度はSIMBOLのインプリメンテーションを行ない、47年度は、既存のシミュレーション言語との比較に依るSIMBOLの評価を行なった。

1章では従来のバッチシミュレーションシステムに関する評価の経緯、オンラインシミュレーションシステムの発展過程、その一つとしてSIMBOL　の

開発成果がどうであるかと云った評価をどう行なうかと云った問題を論じている。

2章では、評価すべきSIMBOLの概要説明をネットワークフローモデルを例にとりあげて行ない、システムの全体的機能を捉えている。

3章では、SIMBOLの設計全般にかかわるような問題として、オブジェクトの形態・メインプログラム・ステージ・ファイル・ディスプレイ等を考察しマンーマシンシステムとしてのSIMBOLの有効性に関して述べている。

4章では、モデルビルディングと云う方向で、ソースプログラムインプット・ソースプログラムアップデイト・エラーチェック・プログラム構成法等に関する考察を行ない、インタラクティブなモデルビルディングの果す効果と云うものを検討している。

5章は同じくモデルビルディングの続きであるが、モデルを構成する際のロジックの記述に関する問題を取り上げている。

6章はモデルのテストとランニングに関する部分を扱っている。ここでは初期設定の問題であるとか、トレース処理とオンラインモニタリングの効果、ランタイムチェック・チェックポイントリスタート機能の有効性等シミュレーションイクスペリメントと云われる部分の検討を行なっている。

7章は、設計上の大きなトレードオフであるシミュレータ容量と実行速度に関する問題を、使用効率・処理効率と云う形で論じている。

最後に8章で今迄の総括を行ない、各種シミュレーションシステムの使用経験から得た感想等も織りまぜて検討項目を表形式にまとめている。

目　　次

第1章　総　論

第1章　総　論

1.1 従来の比較評価

1963，64年は、いろいろな意味で、離散系シミュレーション言語作成史上に於ける最盛期であったように見受けられる。現在の汎用シミュレーション言語の初版（オリジナル・バージョン）は、そのほとんどがこの時期に誕生していると云っても過言ではない。

因に、1963年には、GPSS－Ⅰ・Ⅱ，SIMSCRIPT，GASP等が、翌年にはSOL，ESP，OPS－Ⅰ・Ⅱ等が続けざまに開発されている。65年以降は、SIMULAの開発を除けば、機能及至、性能の充実に努力が払われ、定着期の様相を呈していると見る事ができるだろう。

この定着期は、又、開発した成果を、多様な角度から再検討する時期でもあり、様々なユーザに対するアンケート調査の結果や、将来のシミュレーション言語のあるべき姿を追求したレポート等がいくつか報告されている。問題にされた評価基準の概略は以下に記すような5つのクラスに大別される。

1. モデル作成の為の概念（アクティビィティ・エンティティ等）の普遍性・妥当性
2. シミュレーションのデータ・ベースに関する記述性・操作性
3. 時間軸制御に関する融通性
4. モデルのランニング及びデバッキングに関する機能、柔軟性および効率
5. 言語全体として捉えた場合の受容性・実用性

これらの5つのクラスが各々細分され、あるものは統合された形で展開され、論ぜられている。ユーザがシミュレーション言語を選択する場合に、何を基準にすべきかと云う形で論ぜられたり、或は設計者の立場から、シミュレーション言語はどういう機能を持つべきか、どの程度の性能を備え

るべきかと云う形で論ぜられたりする。

論ぜられる立場や価値観の相違はあっても総じて定性的な評価基準で把握されている。人間を積極的にシミュレーションシステムの構成要素として把握しようとする評価の方向性を望む声は強かったのであるが、そこまでには至らなかったのが実情のようである。MITのCTSSの下で動作したOPS－3（1965年）、MULTICSの下で動作するSIMPL（1972年）等数少ないオンライン・シミュレーション・システムの具体例は、この間の事情を物語るものである。

しかしながら、この期間に蓄積された経験はシミュレーションシステム設計上の問題点として現在に投影されており多くの示唆を与えるものとなっている。これらの問題点のいくつかを簡単に紹介しておく。

- シミュレーションの再現性の問題

  確率過程を表現するモデルであっても、同じ状態から出発すれば、いつでも同じ結果が得られる事の保証。

- 同時並行現象の処理

  同時現象が相互に依存し合う場合、処理の優先順位をどうするか。

- 多重割込現象に関するもの

  割込の深さをどこ迄にするか。

- 時間軸の非可逆性

  時間を過去に戻すために必要な機能は何か、何を保存しておけばよいか。（過去の結果を無駄にしないために）

- シミュレーション時間に関する問題

  人間の行動と計算機内部の動き等マクロな時間とミクロの時間が混在するモデルに於ける単位時間の選定をどうするか。

- デバックの為の機能

  空間的な位置関係として論理の流れが記述される一般のプログラムに加えて、位置関係が時間的にも変化するプログラムを対象とするため

に、強力なチェック機能を必要とする。従って、モデルの妥当性をチェックし得るためには、どういうデバック機能が必要か。

● データ・ベースの記述性に関するもの

物と物との複雑な関係を容易に表現し得るようなモデル化概念を提供するためには、どのようなデータ構造とアクセスメカニズムが必要か。

● 統計収集機能

システム・ライフ全体の特性を表現する時系列データの処理と、長期に渡り蓄積されたデータに関する総括的な統計量の処理をどこまで自動化するか。或は、ユーザ用の統計変数及びデータタイプとして何を用意すればよいか。

● シミュレーション期間に関するもの

開始時点、終了時点を固定化するか否か、固定化したときには、その変更の自由度をどこまでにするか。非固定の場合、どの時点で期間が明らかにされる必要があるのか。

● シミュレーションの再開

初期データの再設定、モデルの変更等をどこまで許すか。

● 処理速度と使用可能な容量

一般にシミュレーションは、非常に多くの実行時間と豊富なメモリー・スペースを必要とする代表的なものの1つであると云われている。従って必要な機能を充たす範囲でどこまでサービス機能を減らすか、或はオプションとするか。又、限られたメモリー・スペースをどこまで処理速度を低下させずに効率的に使用するか。

● レポート作成機能

書式が固定したシステムアウトプットにして使用者の負担を軽くするか、融通性を重んじ書式付アウトプットやレポートジェネレータに相当する機能を設けるか。

1.2 オンラインシミュレーション

前節でも多少触れたように、オンライン・シミュレーション・システムに対する要請が全くなかったわけではなく、むしろ多くの問題点は、バッチ・シミュレーション・システムの不備として認識されていたようである。計算速度とメモリー容量とのトレードオフは、根本的な問題として解決し得ないまま残るものではあるが、ハードウエア、ソフトウエア両面の進歩は、早晩そのような相剋を第一義の座から引きずり降してしまうであろう。

MULTICSの下で使用を予定されていたOPS－4, その思想を受けついだSIMPL等は、シミュレーション・システムにとってより重要な問題を解決しようとする積極的な試みとして高く評価されるものである。

シミュレーションに於ては、複雑な対象システムをモデル化する事に非常に高度な人間の論理判断を必要とし、モデルの信憑性を高める過程は、鋭い直観力と深い洞察力が要求される非常に困難な仕事となっている。このように本質的に人間のヒューリスティックな思考活動を、従来のバッチ・シミュレーション・言語では完全に人間の分担としていたが機械と人間の協業によるインタラクティブな手法がより効果を上げるという観点からオンライン・シミュレーション・システムが登上したわけである。従ってここでの主要な問題は、バッチシステムでの機能を失なわない範囲で、人間の思考活動を補うためにオンラインシステムとして何を成すべきかと云うことであろう。マンマシン・システムの形態が考えられ始めて、問題解決の為の道具としてのシミュレーションの意義もより大きいものと成り得る。

この段階では、もはや言語の域を脱して1つのシステムを形成するわけで、オンライン・シミュレーション言語は、システムの構成要素として把握され、機能として何を果たすべきかという方向で論ぜられる。前節で簡単に紹介しておいた問題点をも踏まえた上で、新たに発生する主要な問題点を以下に記す。

● 人間系と計算機系とが果たすべき機能は何か。

アルゴリズムが確定しているものは全て計算機側が代行する。

プログラム化し得る意志決定を人間から分離する。

経験、勘、認識等のプログラム化し得ない部分を人間が分担する。

● インターフェースの設計

人間系と機械系を結び合わせる接点の果たすべき機能は何か。

装置としてのインターフェースとして何が適切であるか。

● 人間の動作特性を考慮した機械系の性能応答時間は少なくともどの位の時間であれば良いか。

人間の恣意性をどこまで許すか。

上記の問題点に何らかの形で解を与えたものとしてSIMPLが開発された事は先に述べた通りである。

機能拡張という意味ではGPSSをオンライン化したGRAILやGPSS 360/NORDEN等も既に開発されているわけだが、バッチシステムとオンラインシステムでは機能のみならず思想をも異にするわけで、当初からオンラインシステムとして開発され実際に動作した実用システムと云う事になると数える程しかない。従って、既存のオンラインシミュレーションシステムを比較評価するには、対象が極めて限られており、対象と目されるSIMPLでさえ我々がマニュアルを入手し得たのは1972年の冬であった。

このように前例が極めて限られているためにSIMBOLの設計に際しては、オンラインシミュレーションに対する未知の要素を解明する中で実際にシステムを開発するためには、何を成すべきかということと何を成し得るかということとの間の矛盾を解決しなければならなかった。SIMBOL開発の過程は、まさにこの種の幾多の難問に、不確実性を前提に意志決定するが如く、1つのユニークな解の集合を与える試行の連続であったとも云える。

価値体系を異にすれば全く違った判断も可能であった訳で、判断が正当であったか否か、或は判断した通りに実現されたか否かを検証する事が本報告書の目的である事は冒頭で述べた通りである。我々がどのような展開を試みたのかということは次節に譲ることとし、ここではオンラインシミュレーションシステムに対する要望、及至要請を簡単に紹介しまとめとしよう。

● モデル作成、変更の柔軟性

妥当なモデルを当初からユニークに決める事が至難である以上、試行錯誤的にモデルが構成できることが望ましい。そのためには部分的であれモデル構造の変更が任意に行なえる必要がある。

● 強力なデバック機能

モデルの構造を変更したときは、変更結果を即座にチェックし得るような機能が必要となる。変更部分だけを実行してみたり、途中結果を確認し得ることも必要な機能である。

● 実験過程に対する配慮

システムの行動を規定する特性変数の時間的経過を確認し乍ら安定状態を探索し得る機能や、特性変数そのものが明確でない時に試行結果を解析する手段や、最適化を計画するに必要な機能。

● モデルランに関する操作性

どの時点でも、どの部分に対してでも、操作したい時にモデルランの任意断面を参照でき、その結果にもとづき変更ができれば、高価なモデルランを無駄にしなくて済む。

● 試行錯誤過程に対する配慮

モデル作成・テストラン・デバックという単位は1回の流れで解決するものではなく何回となくループし、モデルがリファインされるものである。従ってこの仕事の単位は完全にフィードバックループを形成し、テストランからモデル作成へ次はデバックへというように任意に

切り替え可能である事が必要である。

● 視覚効果

インターフェースにグラフィック端末等を使用し得るなら、人間の図形認識能力に訴える多様な図形表示機能があることが望ましい。

● システムシェアリングの問題

計算機の時間の1部分を人間が使用する形態を取るため、同時に何人ものユーザーにシステムを提供することに依り計算機時間の有効使用と、使用料の低廉化を図る事。

● 人間の思考活動の円滑化

人間は思考を余儀なく中断された場合でも非常にパーフォーマンスの落ちる動物である。バッチシステムでの反応サイクルは人間にとって非常に負担となっている。従ってシミュレーションランの途中結果を保存しておく便宜や、ラン結果を人間が無理なく反応し得る時間内に提供する便宜が必要となるであろう。

1.3 評価の視点

前節迄で従来のシミュレーション言語の評価に関する大局的な経過を述べてきた訳であるが、そこで明らかにされたようにその大部分が、概念的な評価検討に終始している。従って大雑把な選択基準の方向は示唆し得ても、具体的なイメージとしての選択基準を提供する迄には至らない。これは使用者側の要求が多目的であるがために、一般的な選択基準を各種言語に対して論じ得ない事に起因している。従って使用者側の立場に立脚して各種言語比較を試みると、使われ方の多様性の故にどうしても評価基準を抽象的にせざるを得ない訳でそのような最大公約数的な表現ではもう1つ選択の決め手を提供する強力なものとは成り得ないであろう。

ここではシミュレーション言語の選択基準を与えることではなく、SIMBOLの開発に関する追跡評価を試みるという立場から評価検討を行

なっている。

即ち

- 設定されたデザイン・ゴールが達成されたか否か。
- 設定した機能は必要十分か否か。
- 性能は満足のいくものであるか否か。
- シミュレーション言語の問題点のいくつかは、オンライン化で解決がついたか。
- 新たに発生した問題点は何か。
- どこ迄が解決される範囲なのか、依然として残る問題点は何か。

という設計者側から見た検討を中心テーマとするものである。勿論、機能・性能の設定に関しては、使用者を最優先に考慮している訳で、具体的な検討項目は、少なからず利用者の選択基準としての役割をも果すであろう。

オンラインシミュレーションシステムが日本に於てはいまだ揺籃期であるという認識に基づき、検討対象としてのSIMBOLの全体的なイメージを紹介する意味をも含めて、より具体的な形で展開する。具体的な検討の段階では、各種バッチシミュレーションシステムを実際に使用して比較検討を行なった。

使用したパッケージは、GPSS-V, SIMSCRIPT-Ⅱ（以上 IBM 370-195）、SIMSCRIPT 1.5、SIMULA 67（CDC 6600）、SIMULA 65（UNIVAC 1108）である。GPSSはトランザクションの流れを基調とし、SIMSCRIPTはイベントを、SIMLAはプロセスタイプをという形でモデル化概念が相異なり、モデル構成が同一でないため、パーフォーマンスを一概に論じ得ない。従ってここでは、バッチシミュレーションシステム間でのパーフォーマンスの検討は省略している。

J.H.MIZE／J.G.COX 共著、「シミュレーションの基礎」に依れば、シミュレーション研究の体系的アプローチとして、次のようなものが紹介されている。

a. 問題の定式化

- 研究の目的
- システムの記述
- 仮定事項の認識

b. シミュレーション実験の設計

- 数学的モデルの定式化
- シミュレーション実験のデータ
- サンプリング上の考慮
- モデルの検証

c. 電子計算機モデルの作成

- 初期条件と均衡
- 時間進行機構
- 変換乱数発生ルーティン
- パラメータの変更と決定規則の選択
- 記録の保存と統計の作成
- 電子計算機モデルの作成
- 電子計算機モデルの検証

d. シミュレーション・データの分析

- 統計的検定
- 結果の解釈

これらの各ステップは順を追って逐次進められるものではなく、モデルの妥当性に不審な点があれば、モデルを改良したり、或る場合には再構成を試みたりするという具合に、本質的には多重の閉ループを成し、或る試行結果に基づき次の試行を純化する形の強力なフィードバックを背景としている。

従って各ステップ間に跨る形の問題も多く、どのステップの問題として論ずるべきかが又一つの問題となる。

ここでは、便宜上シミュレーション研究の各ステップを大きくモデルの作成とモデルのテストの２つの部分に分け、それを更に細分して取り上げている。

第2章　SIMBOLの概要

第2章　SIMBOL システムの概要

"SIMBOL システムは、モデル作成の段階とテスト段階との間の障壁を取り除き、モデルの構成が効率よく行なえる事を目的に開発された、インタラクティブな会話型シミュレーションシステムである"

1. モデルの作成がインタラクティブに出来、随時修正や追加が出来る。
2. モデル全体が完成していなくても部分的にテストしながらモデルを構成して行くことが出来る。
3. プロセスタイプの言語であり、モデル表現のし易さと記述性の豊さを備えている。
4. モデルの実行に関するオペレーション機能が豊富である。
5. トレース機能が豊かでそのオペレーションも自由である。
6. デバック済みのモデルの一部或は全部をコンパイルしてオブジェクトプログラムを作成し、モデルの実行スピードを上げる事が出来る。
7. モデル実行時に途中でステータスをセーブし、後にそこから再実行する事が出来る。
8. ランタイムに Quit をかけモデルのステータスを参照したり変更したりが自由に出来る。
9. キャラクターディスプレイを端末とし、モデルのステータス等をバーチャートで表示する事もできる。必要に応じてハードコピーの作成も可能である。

等、多くの特徴を持っており、実用的なシミュレーションシステムを構成するべく開発されている。

ここではその概要を、実際に稼動する様子を随所に織りまぜながら簡単に紹介しておく。

2.1　例題モデルの概略

まず実際に図 2.1.1 に示すようなネットワークのシミュレーションを

SIMBOL で行なう事を考えて見よう。

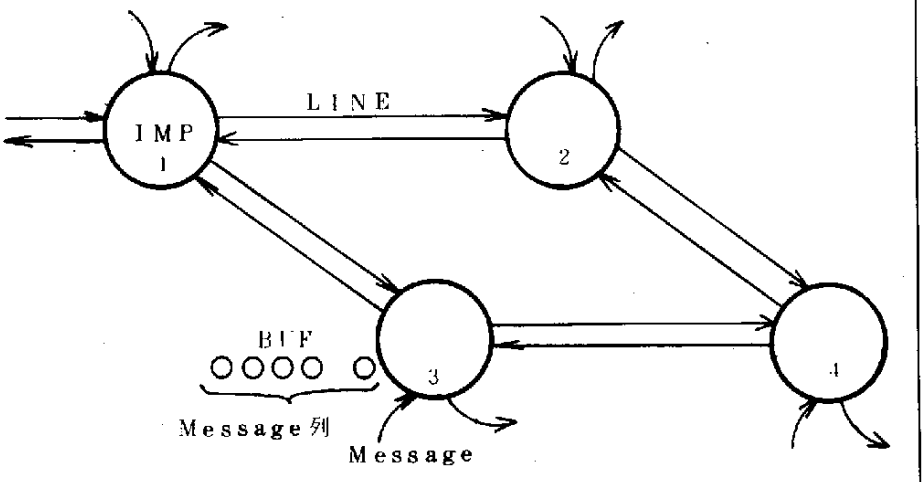


図2.1.1 Network Flow Model

SIMBOL を使用するに先立ち、まずシミュレーションの目的を明確にし、対象システムの論理的構造を明らかにしなければならない。

次にシステムを構成する要素の特性を調べ、システムの中を移動する"物"の特性、"物"と"要素"との関係、"物"相互の関係等を明らかにする必要がある。

評価すべきものを量的に表現する方法を考えたり、目的に合わせてモデルの単純化を行なう事も必要であろう。

SIMBOL を使用するか、GPSS で行なうか、或は FORTRAN で済ませるかという検討も又必要な事であろう。

SIMBOL システムを用いてシミュレーションを行なう事が決定された時点から、使用者と SIMBOL システムとの実際の対話が始まる。

対話の過程を追跡するに先だち、対象システムの概略の説明をしておく。

2.1.1 主要構成要素

- IMP (Interface Message Processor)
- BUF (IMP' Buffer)
- LINE (Communication Line)
- Message

ＩＭＰは、メッセージのスイッチングを行なうサービス窓口であり、メッセージを貯えておくＢＵＦ、即ち共同待合室を持っている。

ＬＩＮＥ は、ＩＭＰ相互を連結する通信回線であり、単一サービス窓口である。

メッセージ（Message）は、各ＩＭＰに連がれた端末から非同期に発生し、Ｎ番目のＩＭＰに連がれた Host コンピュータで PERT/TIME の処理を受けたいとか、或はＭ番目の IMP 下の Host で COBOL を使用して給与計算を行ないたいとか云う要求を持ったものである。

2.1.2 処理の概要

メッセージがどのＬＩＮＥ を通って目的とする IMP に到達すべきかと云う経路選択に関しては、一意に定まっており、選択の余地が無いものとする。

発生したメッセージは、自分の端末が所属する IMP の待合室に余裕が無い場合は無効となり、システムの場から立去る。

IMP に受けつけられたメッセージは、自分の行先に従って、LINE のサービスを受ける。

ＬＩＮＥ が使用中である場合は、共同待合室で待たされる。

ＬＩＮＥ のサービスが可能な場合は、その処理を受けた後、次のIMP に送られるが、そこでの IMP の待合室が満席であれば、再送しなけれ

ばならない。

次段のIMPに受けつけられた時、もしそのIMPがメッセージが要求するHostを従えたものであるなら、メッセージをシステムの場から消去する。

目的地に到着しない間は、次段のLINE, IMPを求めてメッセージが移動していく。

2.1.3 シミュレーションの目的

ネットワークへのメッセージ到着率が或る統計分布として与えられた時、効率的なIMPの待合室の大きさと、LINEの伝送速度とを求める事が目的であり、到着率が、サービス速度を上回る事はなく、少なからず最適解の存在は保証されているものとする。

対象とシステムの概要は以上の通りである。次にSIMBOLシステムと使用者との具体的な関わり合いを追ってみよう。

2.2 モデル化概念の具体化

まず SIMBOL が用意しているモデル化のための概念を良く吟味理解した上で、対象システムが SIMBOL のモデル化のための材料を用いてどう表現されるかを検討しなければならない。

メッセージの流れを中心にモデル化するか、或は LINE のサービスの開始・終了に着目してモデル化するか、IMP や LINE や BUF と云ったものを何で表現するか等々、着目のし方次第で、非常に簡単にモデル化出来たり、或は全くモデル化不能に陥ってしまったりする。

ここではメッセージの流れを中心にモデル化を進める事にしよう。

メッセージ処理の流れをメッセージ中心に（私は今 i 番目の IMP に居て、N番目のLINEのサービスを受けるために待合室で待っている。サービスが受けられたら j 番目の IMP に到着する。j 番目の IMP が目的地でなければ更に次の LINE のサービスを受けなければならない）と云う形で、その処理パターンをアクティビティとして記述する。

個々に発生するメッセージというものは、例えば "Kyoto" とか、"Kobe" と云った識別子を持って、アクティビティが表現している処理パターンを実行する事に依り、混同される事なく別個のものとして解釈される。

従って個々のメッセージは、アクティビティから識別子を伴って発生されたもの、即ちプロセスとして表現される事になる。

構成要素とモデル化材料との対応は次のようになる。

- メッセージ処理の流れ……アクティビティ
- 個々のメッセージ処理……プロセス
- IMP の BUF ……キュー
- LINE ……キュー
- 径路選択表……配列
- 入力分布 ……関数

以上の条件をもとに、ブロック図を描いて全体の関連を把え、漏がないか、重複している部分は無いか、単純化出来る部分は無いか等の検討も必要なステップであろう。

図2.2.1にそのブロック図の一部を示す。

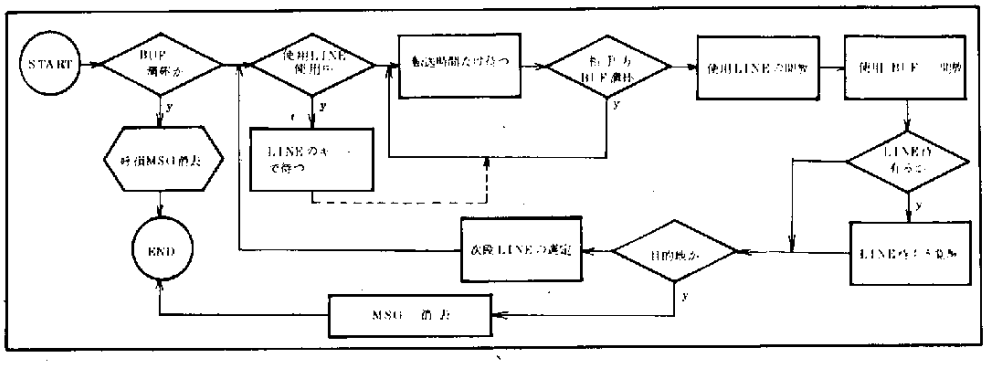


図2.2.1　Message 処理フロー

2.3 モデルの作成

2.3.1 SIMBOLシステムの呼出し

キャラクターディスプレイの前に座って実際に SIMBOL システムとの対話が開始される。

タイプライタ端末からシステムを起動させなければならない。

表2.3.1　SIMBOL　の起動

```
• ***D
• ? USER-ID･･･OZAWA
• ? YOUR-PSWD･･･01024348211
  /YOUR JOB-NO････DJIF021
  MACRO BUN NYUURYOKU.
• ** SIMBOL
  JOB KAISI.
```

LOG-INが完了するとキャラクターディスプレイが起動される。

この状態を SIMBOL では、モデリング・ステージと呼んでいる。

SIMBOLには、モデリング／シミュレーションと名付けられた2つのステージが用意されており、ステートメント或はコマンドのインプットが出来る状態を示すものである。

各ステージ間はコマンドに依り連結され、多重のフィードバックループを簡便に実現し、モデルのリファインを行ない易くしている。

他に SIMBOL がモデルランを実行している状態があり、便宜上エクゼキューション・ステージと名付けている。

1. モデリング・ステージ

   モデルの作成と部分的なテストをインタラクティブに行なう事を主目的とするステージであり、ファイルのメインテナンスも同時に行なう。

2. シミュレーション・ステージ

   実際にシミュレーション・クロックを進めながら、モデルの動作特性を解析するためのステージ

3. エクゼキューション・ステージ

   モデルラン実行中の状態であり、割込みをかける事によりシミュレーション・ステージへ復帰する。

各ステージ間の遷移は、コマンドをインプットする事により行なわれ図2.3.2に示すようになる。

この図で＜＞の中に記されたものがコマンドであり、ここでは主要なもののみを示してある。

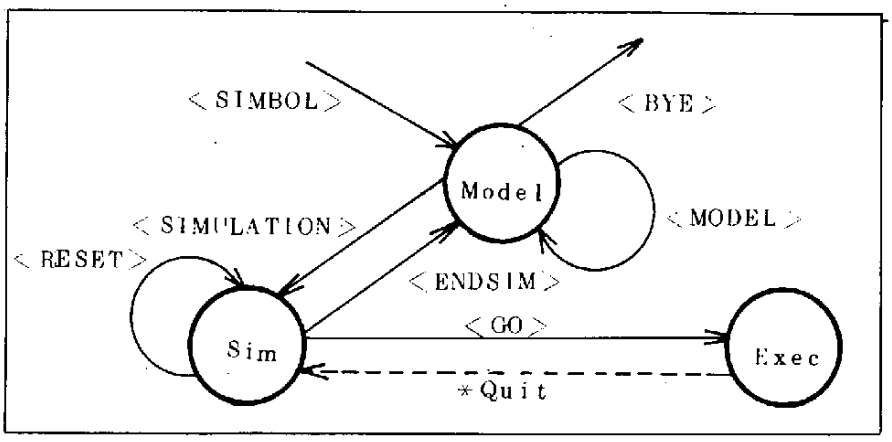


図2.3.2　ステージ間遷移

話をモデリング・ステージへ戻そう。

既に作成されたプログラムの消去等ファイルメィンテナンスの必要があれば DELETE コマンドを用いて行なう。

どのようなコマンド群が使用出来るのかを知りたければ＜TEACH＞コマンドをインプットする事に依り、どのステージでどのコマンドが使用出来るか又そのコマンドの機能の概要は何かと云った事がディスプレイ画面に表示される。

＜TEACH＞ コマンドは、 Execステージを除けばどこでもインプット可能である。

表2.3.3　コマンド一覧表

COMMAND	STAGE	FUNCTION
ACCEPT	M・S	Data Accept
BYE	M	Logout
CLEAR	S	Model Status clear
CHECK	M	Run Time check abolishment
COLLECT	S	Statistical Value Gathering
COMPILE	M	Source statement Compile
DISPLAY	M・S	Data display
EDIT	M	Program Editing
ENDSIM	S	Simulation Stage End
EXECUTE	M	Partial Execution
FILE	M	Source Program Filing
GO	S	Simulation Run Start
GRAPH	S	Histogram Output
INPM	M	Changing input mode
LET	M・S	assign
LOAD	M	program lood from File
MODEL	M	Change the Model
RESET	S	Statistical Value reset
RESTART	S	Simulation rerun
SAVE	S	Executing Program Saving
SIMBOL		Processor Call (Log in)
SIMULATION	M	Simulation Stage Transition
TEACH	M・S	Asking the Command
TRACE	S	Trace processing
USE	S	Using Program indication
		(M‥Modeling Stage；S‥Simulation Stage)

MODELコマンドは、モデル単位を変えるためのものであり新たにモデルを作成する場合には必ずモデリング・ステージの最初の段階でこの＜MODEL＞ コマンドをインプットしなければならない。

既に作成済みのモデルを呼び出して、再度追加・修正等をしたい場合には ＜LOAD＞コマンドが用意されており、先に SAVE コマンドに依り名前付けした名前で呼び出す事が出来る。

この場合は、＜MODEL＞ コマンドのインプットは不要である。

2.3.2 メインプログラムのインプット

SIMBOLではモデルを構成するためのプログラム単位としてメイン・アクティビティ・プロセデュア（含ファンクション）が用意されている。

プロセデュアは通常のサブルーチン乃至はファンクションに相当する。

アクティビィティは、擬似並行的に処理されるプロセスの処理パターンを記述するもので、シミュレーションプログラムの中核となるものである。

メインは通常 FORTRAN 等で扱われるものと同等であり、実験計画を記述してシミュレーションランを制御する事も可能である。

各プログラム単位はラインにより構成されている。

ラインは使用者がキャラクター・ディスプレイからインプットする最小の単位であり、ラインナンバーから始まるコレクトステートメントと、％記号から始まるダイレクトステートメント乃至コマンドとから構成されている。

＜％MODEL NETWORK＞

がインプットされた直後はメインのインプットが可能な状態に自動的にセットされる。

どのプログラム単位を処理中なのかを明確にするために SIMBOL では、プログラム単位のインプットに関連して、インプットモードと云

うものを設けている。

メインをインプットする時にはインプットモードはメインにセットされており、ここでアクティビティの記述は許されない。

モードの切替えは＜%INPM＞コマンドにパラメータとしてメインであるかアクティビティであるか等の識別子とそのプログラム単位の名前とを与えてインプットする事に依り行なわれる。

まずメインプログラムで、システムライフ中変化しないデータ構造、乃至は共通に使用されるデータと云ったものを定義する。

コレクトステートメントをインプットする場合には、ラインナンバの後に、レーベルが必要ならそれを付し、以下ステートメント本体をキーインしセミコロンでしめくくり、SENDキーを押せばよい。

図2.3.4にステートメントのキーインの様子を示してある。

```
MODELING STAGE                                    INPM=MAIN
  10 INTEGER  I,J,K,L,U1,U2,CIRCUIT(10,10);
  20 REAL  MATRIX(10,10), ORIGIN(10);
```

```
KEYIN PLEASE
  30 QUE CRCTQUE(28);
```

図2.3.4　プログラムのインプット

コメントはセミコロンの後につづけてインプットする。

シミュレーション期間中変化しないデータや初期設定データを与えてしまいたければ、ダイレクトステートメントが用意されている。

```
%ACCEPT U1,U2 ;  SEND
```

をキーインするとデータアクセプト状態になり、変数のタイプに従って

```
ACCEPT INTEGER DATA U1:
```

と表示されるのでカーソル以降にデータをキーインすればよい。

キーインは1データアイテム単位であり、配列名を＜ACCEPT＞のパラメータに記述した時は、配列要素が尽きる迄データのキーインを要求してくる。

大量のデータは紙テープから読み込ませればよい。

紙テープから読み込ませたデータが正しいものかどうか確認したければ、＜DISPLAY＞ステートメントが使用出来る。

シミュレーション開始を指示するステートメント（START）をキーインする事により、最初に実行すべきプロセスが明らかにされる。

```
10  INTEGER
 \
40  SET IMP(10) ;
50  READ(T)  CIRCUIT, MATRIX, ORIGIN;
60  ACCEPT U1, U2;
70  START MSGCNTL, 600;
```

図2.3.5　プログラムのインプット

START ステートメントにはシミュレーション開始・終了に関する種々の条件が記述出来るようになっており、ラインナンバー７０で示した MSGCNTL は最初に実行すべきプロセスの属するアクティビティを、その後の 600 は終了時間を示している。

ここでアクティビティが登上したので一応メインプログラムのキーインを終了して、アクティビティ単位のインプットに移ってみよう。

先に述べたインプットモード切替え用のコマンドを用いて、MSGCNTL と名付けたアクティビティへモードを切替えよう。

% INPM ACT MSGCNTL ; Ⓢ

2.3.3 アクティビティ単位のインプット

インプットモードが切替わると画面最上段にその旨の表示が成され、以後再度切替えが成される迄そのモードとプログラム単位名が表示されている。

メインはモデルに対して１つしか許されないので名前を宣言する必要が無かったが、アクティビティやプロセデュアは同時にいくつも存在出来るので識別名のキーインが必要となる。

ACTIVITY 宣言ステートメントによりこれを行なう。

アクティビティ "MSGCNTL" は平均到着間隔１０単位時間の負の指数分布に従ってメッセージを発生させるプログラムである。

このアクティビティは到着時間に呼び出されると新たにメッセージを１つ発生するプログラムとして記述される。

```
100  ACTIVITY MSGCNTL;
200  REAL     ARVTM;
300  CREATE.  NAME MSG:=NEW MESSAGE;

*ERROR……UNDEFINED VARIABLE NAME:MSG
 300  CREATE.  NAME MSG:=NEW  MESSAGE;
```

メッセージが固有に持つデータの構造を定義するブロックが先にキーインされていなかったので未定義変数と云う形でラインナンバー３００のラインが受けつけられなかった。

そのブロックをインプットするために、インプットモードをブロックへと切替え、ブロック宣言を行ない、所要データ構造をキーインしよう。

```
%INPM  BLOCK  MESSAGE;

100  BLOCK MESSAGE;
200  INTEGER ORG, DST, CRT, LINE;
300  REAL GENTM;
400  END
```

これによりメッセージはORG（発生局）、DST（目的局）、CRT（現存局）、LINE（使用回線）、GENTM（発生時間）をデータとして持つ事になる。

ブロックはアクティビティの変種であり、固有のデータを与える事が可能であるが、アクティビティと異なりオペレーション規則の記述が許されず、単にデータ構造をコンパクトに記述するためのものである。従ってアクティビティとして記述してもなんらさしつかえはない。

MESSAGEブロックのキーインが完了したので、元のアクティビティ

へ制御を戻そう。

この時には、既にアクティビティの宣言が成されているので（100 ACTIVITY MSGCNTL により）、単に名前だけをキーインすればよい。

即ち　%INPM MSGCNTL Ⓢ

これに依り切替えが行なわれ、既にキーインされたものが画面上に表示される。

このアクティビティでメッセージを発生させずに、メッセージ処理用の別のアクティビティ（MSGFLOW）で行ないたければ、先にエラーとなったライン（300）を例えば

300 SCHEDULE NEW MSGFLOW DELAY φ;

に変えればよい。

その場合にも先に MSGFLOW をアクティビティとして宣言しておかなければキーイン要求が起こる。

この時にもインプットモードの変更が必要で、その後必要な宣言を済ませればよい。

即ち情報が不足する場合は、その場で最低限必要なアクションは取ればよいようになっている。

```
%INPM   ACTIVITY   MSGFLOW

100  ACTIVITY   MSGFLOW;
200  INTEGER    OG, DT;
300  CALL       ODSELECT(OG, DT, ORIGIN, 10, U1, U2);
```

ここでプロセデュアODSELECTをキーインしなければならないと云う事はない。

プロセデュアはランタイムにリンクされるので、ランの直前迄に用意すれば良い訳だが、ここではプロセデュアのキーインを試みる事にする。

例によって

```
%INPM   PROC   ODSELECT Ⓢ
```

のキーインに依りプロセデュアのインプットモードとなる。

2.3.4 プロセデュア単位のインプット

```
  100 PROCEDURE ODSELECT(ORG,DST,ORIGIN,N,U1,U2
                                          U2, MATRIX)
  200 INTEGER    N,ORG,DST,ORIGIN(N),U1,U2;
  210 REAL MATRIX(N, N);
  300 LET        ORG=DISCRETE(ORIGIN,U1);/*
                 *ORIGINAL IMP SELECT*/
  400 LET        Y=UNIFORM(0, 1.0, U2);
*ERROR  UNDEFINED  VARIABLE  NAME:Y
  250 REAL       Y;
  400 LET        Y=UNIFORM(0, 1.0, U2);
  500 DO  LAB1, FOR  DST=1, 10;
  600 IF  Y < MATRIX(ORG,DST) THEN  BRANCH  EXIT;
  700 LAB1.  LOOP;
  800 EXIT.  END;
```

DISCRETEやUNIFORMは、システムが用意している乱数処理用のファンクショナル・プロセデュアで各々、乱数の値と配列要素の値とが比較され適当な配列要素の添字を値とする離散値関数、区間〔a，b〕の一様乱数を与える連続関数である。

プロセデュアにはいくつかの制限があるが、時間消費が出来ないとか、アクティビティやブロックの名前を直接参照する仕方が許されない等が大きな制約事項である。

これらの制約を使用者は十分理解した上で効果的な使い方を考える必要があるであろう。

最悪の場合でも（例えば制約事項とも知らずにキーインした場合等）、当然の事ではあるが、制限されている旨の告知が成されるので、混乱を招く事はない。

SIMBOL では処理効率を高めるために、プロセデュア単位をコンパイルしてしまう事が可能となっている。

コンパイルしてしまったものはライン単位の実行にシステムが関与しないので実行速度を向上させる上で効果的である。

使用者は原則としてデバッグ済みのプロセデュアに限ってコンパイルすべきであろう。

メインやアクティビティはコンパイル不能であり、ライン毎にシステムが介在した形でインタープリーティブに実行される。

従ってシミュレーション全体の処理効率を高めるためには、プロセデュアの性格と云うものを十分意識してモデル化を進める事が望ましい。

今作成したプロセデュア ODSELECT をコンパイルする事を考えよう。

コンパイルに先立ち、果して正しい論理的記述が成されたか否かをチェックする事が必要である。

モデリングステージに於ては、ダイレクトステートメント＜%CALL＞を用いた実行や、メインプログラムにしか使えないが部分的な実行を可能にするための＜%EXECUTE＞等が用意されている。

実行方法のいくつかを簡単に紹介しておく。

① ＜%CALL＞ に依るプロセデュアの実行

%CALL ODSELECT (I,J,ORIGIN,10,U1, U2) Ⓢ

U1, U2 は乱数の初期値、 ORIGIN は添字 10 の 1 元配列で累積確率の値が入っている。

このダイレクトな呼出しは、実効的にはメインプログラムの中からの呼出しと等価になる。

従って I,J,ORIGIN,U1,U2 等は全てメインで定義済みであり、必要ならば初期設定済みでなければならない。

実行結果はメインで定義されたグローバル変数 I,J に代入されているので ＜%DISPLAY＞ によりその値を確認出来る。

② ＜%EXECUTE＞に依るメインの延長としてのプロセデュアの実行。

先にキーインしたメインプログラム単位へインプットモードを切替え

```
100 INTEGER ARY (2,30)
200 DO L1 FOR I=1,30;
300 CALL OPSELECT (J,K,ORIGIN,10,U1,U2) ;
400 LET  ARY (1,I) =J ;
500 LET  ARY (2,I) =K ;
600 L1.  LOOP ;
700 DISPLAY ARY;
800 END
```

を追加する。

％EXECUTE 100,700 Ⓢ

のキーインによりプロセデュア ODSELECT は30回呼び出され、その度に実行結果を二元配列 ARY にしまい込み、30回の実行結果を一度に表示する。

なお当初から ODSELECT をメインプログラム単位として構成した場合も同じような状況になる。

この場合は当然の事乍らライン（300）の CALL ステートメントは不要である。

③ ＜％EDIT RENUM＞ によるメインとしての実行。

既にメインプログラム単位がキーインされていない場合は簡単であるが、今迄述べて来たように既にメインのキーインが部分的であれ成されている場合には、厄介な問題があるが原理的にはメインへと切替えて②の方法に帰着出来る。

その場合には

％EDIT RENUM Ⓢ

をキーインした後②と同じ様な作業をすればよい。

＜EDIT＞ は部分的な実行を可能にするためにプログラム単位をメイン・アクティビティ・プロセデュア相互間で切替えたり合成したりするためのものである。

④　その他のダイレクトステートメントに依る実行

```
%LET
%ACCEPT
%DISPLAY
%SCHEDULE
 :
```

上記①～④で述べて来たように、モデリングステージに於ける部分的な実行は極めて簡単なチェックをするためのものであり、更に複雑なチェックやダイナミックな動作解析は後述するシミュレーション・ステージで行なえるようになっている。

2.3.5　プロセデュア単位のコンパイル

チェックの完了したプロセデュアODSELECTをコンパイルしよう。

不要なラインは消去し、プログラム単位がメインならプロセデュアに切替えプロセデュア宣言もつけ必ずプログラム単位の終りを示すＥＮＤステートメントを付加して完全なプロセデュアの形に再編成する。

例えばチェックが前述の②の＜%EXECUTE＞で成されている場合は、プロセデュアの再編成の必要は無く即座にコンパイル出来る。

```
%COMPILE  ODSELECT  Ⓢ
```

のキーインにより対象とするプロセデュア名を指定するとコンパイルに必要な情報の問合せが成され、所要のキーインを済ませるとコンパイルが始まる。

2.3.6　ソースプログラムの編集

メインに追加したライン（100～800迄）は、もはや不要なので消去してしまっても良い。

ラインの追加・削除・修正も又、＜EDIT＞の機能の1つである。

EDIT は原則として現在画面に表示されているプログラム単位のラインを処理の対象とし、EDIT モード中に成される全ての修正は、このモードから脱出しなければリコンパイル処理の対象とは成らない。

ライン群の削除の場合は必ず消去しようとするラインはキャラクターディスプレイ上に表示されていなければならない。

今メインでプログラム単位の不要ラインを削除しようとする場合、メインを画面上に表示させる事が必要となる。

```
%EDIT   LIST   MAIN(,100);
```

のキーインに依りインプットモードはメインに切替わり、先にキーインしたラインが先頭から表示される。

途中から表示させたければ、プログラム単位名の後に ,n をキーインすればnライン目からの表示が成される。

1画面単位のラインの移動、1画面以内の表示ラインの移動は簡単に行なえるように成っている。

- 1画面単位のスクローリング
  - %*+;　　1画面先の表示
  - %*-;　　1画面前の表示
- 1画面内スクローリング
  - %n+;　　nライン先の表示
  - %n-;　　nライン前の表示

但しこの機能はEDIT モード中では使用出来ない。

EDIT モード中には前述の LIST 機能を用いて、インプットモードの変更や表示画面のスクローリングを行なうようになっている。

EDIT モードになると¥のキーインでこのモードから脱出しない限り、必要なアクションをシステム側から尋ねて来る。

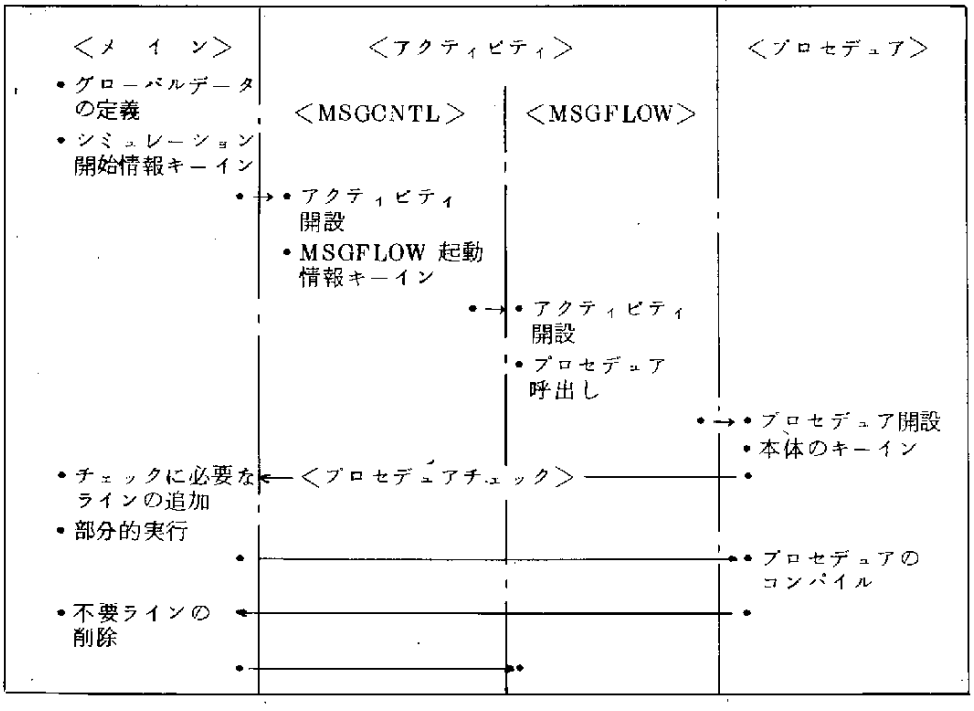
＞ DEL 100－800 ；

のキーインに依り指定したラインの頭に "＊" 印が表示され、削除してかまわないか否かを尋ねて来る。

"YES" 又は "NO" のキーインに依り、実際に削除又は、別のアクションが行なわれる。

2.4 モデルのチェックとラン

並列的にプログラム単位のキーインをし、試行錯誤的にモデルを構成して来た訳だが今迄のモデリングステージでの処理の流れを簡単に眺めてみよう。



この様に、モデリング・ステージでは並列的にプログラムのインプットが可能であり、部分的な実行やプロセデュアのコンパイル等も必要に応じて適時行なえる様になっている。

%SIMULATION

のキーインによりシミュレーションステージへ遷移出来る。

ここで TRACE コマンドを用いたり、ダイレクトステートメントを使用したりしてモデルの動作特性をチェックしたり出来る。

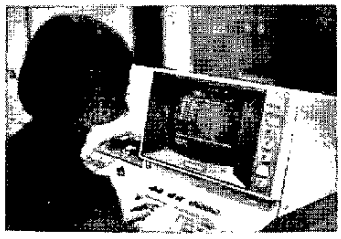
初期設定を何度も繰り返してランを試る事も出来る。

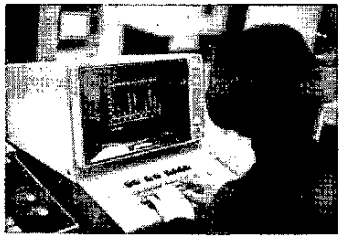
ここでGO コマンドをキーインすれば実際のシミュレーションが開始される。このステージをイクゼキューションステージと云う。

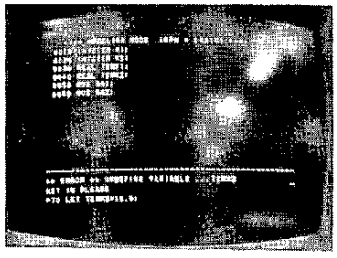
チェックが必要な時は、クイットをかけてイクゼキューションステージからシミュレーションステージへ戻ればよい。

シミュレーションの途中結果を見たければ DISPLAY コマンドや GRAPH コマンドが使用出来る。

写真は実際に SIMBOL を使用している様子を示したものである。







第3章　SIMBOLの全般的評価

3章　SIMBOL の全般的評価

細部の評価検討に先立ち、SIMBOL 全般にかかわる問題をいくつか論じておこう。

オブジェクトの形態、メインプログラムの扱い方、ステージ概念の導入、ファイルに関する処理、更には SIMBOL システムがインプリメントされている計算機システムの問題等、多くは SIMBOL の性格付けに何らかの形で影響を及ぼしているものである。

3.1 オブジェクトの形態

SIMBOL ではオブジェクトコードの形態として、C－形式とI－形式との2種類のものが用意されている。

C－形式（Compiling Object）と云うのは、プロセデュアの単位が完全にコンパイルされ、その実行に際して SIMBOL システムが介在しなくて済むオブジェクトコード群の総称である。

I－形式（Interpreting Object）は、通常の対話的なモデル作成に於てライン単位に作成されるものの総称であり、その実行には必ずシステムが介在するため若干余分な情報を持ったオブジェクトコード群である。

SIMBOL の様に対話形式のモデルの構成・実行が出来るシステムでは、一般に実行速度が極度に遅くなる事が懸念される。

ステートメントのインプットと同時にコンパイルを行なう、所謂インクリメンタルコンパイレーションに於ては、ランタイムにインタープリーティブに実行し得る様な中間コードを作成する方向と、出来る限り機械語に近い形のオブジェクトコードを作成してしまおうとする方向がある。

前者の考えに立てば、中間コードの形態を工夫する事で、ステートメントの修正を非常に簡単に行なう事が可能となる。

しかし乍ら、修正に強靱な中間コードを実行するには、システムがその都度意味を解読して然るべき処理を行なうために実行速度は非常に遅くなってしまう。

即ち繰り返しステートメントを実行する場合、再実行のたびに再解読の作業が必要となるからである。

SIMBOL では後者の考え方を押し進めて、直接機械語に翻訳している。

これは実行速度を極力向上させようとする配慮の現われの1つであり、作成したコードの先頭に制御を移せば即座に実行し得る様に成っている。

この様に曖昧性を許さない機械語に直接翻訳してしまうと、修正に関して非常に敏感になる。

インプット順・追加・削除・修正が任意に行なえる様に設計されているので変更に敏感すぎると、リコンパイルが頻繁に起こってしまい、実用的で無くなってしまう恐れがある。

SIMBOL では、出来る限り全体のリコンパイルの回数を少なく済ませて部分的な修正が達成出来るよう、配慮されている（詳細は後述する）。

I－形式ではステートメント単位に、オブジェクトコード本体と、ステートメントインフォメーションと呼ばれるブロックとが作成され相互にポインターで関連付けされている。

オブジェクト本体間の論理シーケンスは、ステートメントインフォメーション間のポインターのチェイニングで表現される。

ステートメントインフォメーションは、ステートメントの種類や、トレース処理に関するスイッチ類、使われている変数名、各種ポインター等、コンパイラー或はエクゼキュータの処理に必要な情報が格納されるブロックであり、オブジェクト本体はこのブロック内に作成される。

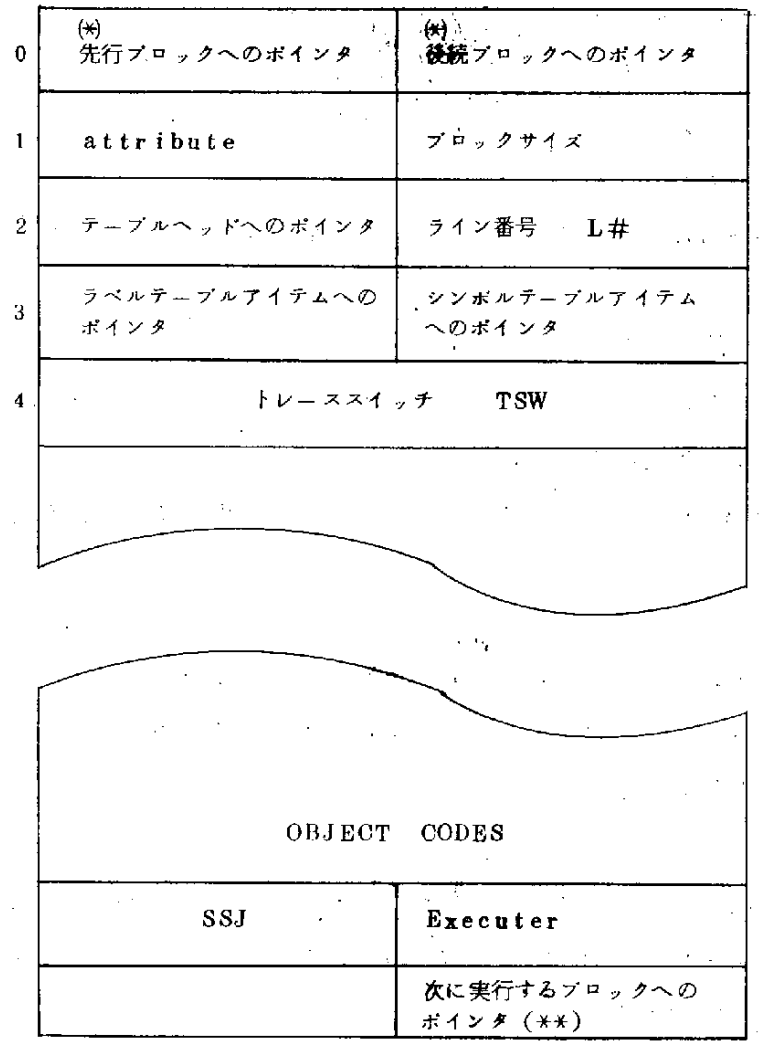


図3.1.1　ステートメントインフォメーション

1ステートメントの実行に際しエクゼキュータが介入し、トレースの指示があればそれを実行する。

ステップバイステップにエクゼキュータの介入により実行が進められるために、エクゼキュータのオーバーヘッドと云うものが問題になる。

そのためにエクゼキュータの介入なしで実行可能なオブジェクトブロッ

クとしてC－形式と云うものが考えられている。

オブジェクト本体としてはI－形式のものと大差は無いが、ステートメントインフォメーションは作成されず全体としてコンパクトに成り実行速度も向上する。

SIMBOL ではC－形式と云うものをプロセデュア単位に限って許している。

これで果して実行速度の向上にどの程度貢献し得るものかと云う問題は後述するとして、ここでは何故C－形式がプロセデュア単位に限定されたのかと云う問題に論点をしぼろう。

SIMBOL 開発段階に於けるC－形式の意味付けの遷移を概括してみよう、

"メイン・アクティビティ・プロセデュアの3種のプログラム単位個別にコンパイルする事に依り、全体として非常に効率的なオブジェクトプログラムを作成する。"

最終的には全プログラム単位がコンパイルされ、FACOM 230－60のモニターVの下で SIMBOL の介在無しで実行可能なオブジェクトプログラムを作成する。

所が3種のプログラム単位の1つアクティビティは、シミュレーションデータベースの参照が特殊であり、独立したコンパイルの単位としては考え得ないものであった。

即ち、メインプログラム単位で参照するデータベースはグローバルデータとローカルデータ、プロセデュアはローカルデータに限られるのに反し、アクティビティはグローバル・ローカルの両者が許される。

自分自身のプログラム単位で定義したデータを参照する仕方しか許されないプロセデュアを独立したコンパイルの単位とする事は何ら問題が無い

が、アクティビティの場合には、メインで定義されたグローバルデータや他のアクティビティで定義されたローカルデータ、自分自身で定義したローカルデータの参照があるために独立したコンパイル単位にならない。

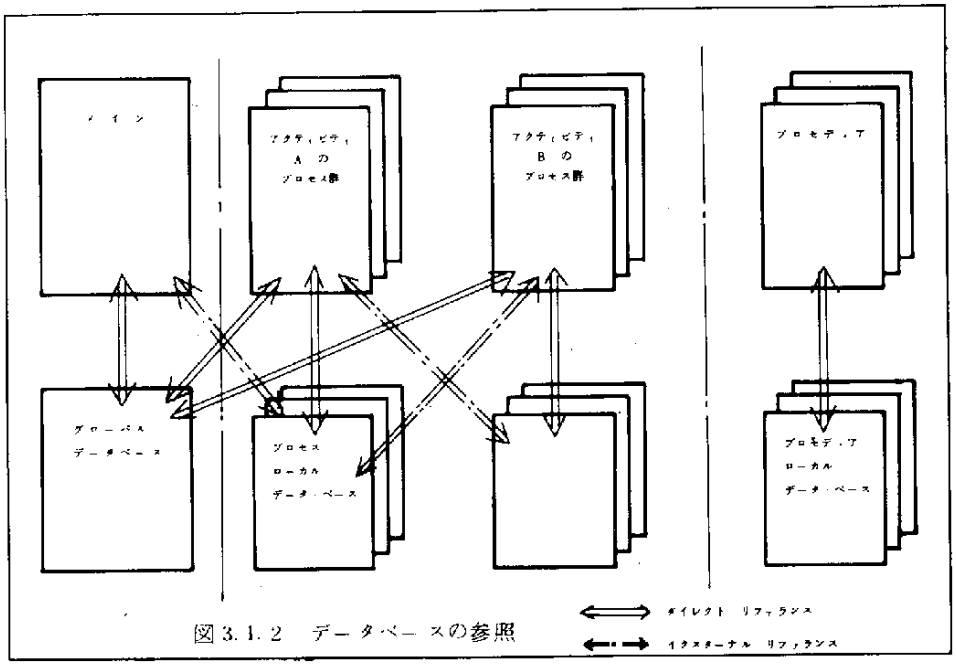


図3.1.2　データベースの参照

メインは自分が参照するアクティビティだけを知っていれば済む訳だが、アクティビティ相互、プロセス相互にデータ参照を行なうため、結局全てのアクティビティとメインプログラムと云うコンパイル単位を考えざるを得ない。

図3.1.2より明らかな様に、プロセデュアは自分の定義したローカルデータベースの参照しか行なわず、他のプログラム単位のデータベースはパラメータと云う形で自分の中のローカルデータベースに取り入れて参照するため完結したコンパイルの単位となり得る。

コンパイルの単位に制限を設ける事に依り、データベースに関する問題点を取り除く事が出来る。

技術的な問題はこの様な形で解消出来る訳だが、対話型シミュレーションシステムにとってより本質的な問題は、この種の技術的問題もさる事な

がら、より思想的・概念的な所にある様に思われる。

即ち、

対話型シミュレーションシステムにとって、コンパイルとは一体何を意味するものであるのか。

対話型システムの真の目的は何であるか。

コンパイルの目的と矛盾を来たさないか。

等、この種の思想的な問題に解を与え、システムとしての目的を明確に規定しない限り、C－形式と云うものの意味付けは成され得ない。

実行速度の向上を重要視する限りに於ては、いずれの対話型システムであっても早晩この種のシステム的課題に直面せざるを得ないであろう。

コンパイル機能を提供し、その意味付けをユーザに委ねるのも一つの解決策であるかも知れない。

その場合でも、コンパイルしたものを全く対話機能を持たないバッチシステムのような形態とするか否かは、当然の事ながら問題にされなければならないだろう。

以下若干のオンラインシミュレーションシステムに於けるコンパイルの考え方を参考に、SIMBOL を検討してみよう。

OPS－4の場合、モデルをコンパイルする事が可能な様に考えられている。

この場合、PL／I トランスレータがOPS－4 に用意されており、コンパイルの指示が成されると、OPS－4 プログラムを PL／I プログラムに再編集し、PL／I コンパイラに制御を渡す。

PL／I コンパイラは、それをコンパイルしオブジェクトを作成する。

OPS－4 自身の言葉に依れば、"プログラムが十分動作するものであると認められた時、実行速度を向上させるために、即座に実行可能な標準

形式の PL／I プログラムにコンパイル出来ます"とある。

コンパイルされたプログラムとインタープリーティブに実行するプログラムとの差異は実行速度のみであると考えられている。

実際インプリメントされた訳ではないので、確実な事は云えないが、OPS－4 のインクリメンタルコンパイルは、修正の容易な中間コードの作成を基本とし実行時に解読し乍ら行なう方針だったようである。

実行はステートメント1行単位に解読し乍ら行なわれる。

従ってコンパイルと云うのは、解読作業に際して、その作業が不要である事を示すものであって、ステートメント1行単位にシステムが介在する点では何らコンパイルされていないものと変わりはない。

OPS－4 で云われる実行速度の向上と SIMBOL のそれとではかなり様子が異なっている。

OPS－4 ではインクリメンタルコンパイレーションが成されたステートメントは即座に実行出来る形態にはなっていない。

従って実行が指示されると中間コードを解読し、オブジェクトコードを作成し、エクゼキューションセグメントと呼ばれるブロックにステートメントをコンパイルした旨表示し、作成オブジェクトに制御を移して実行する。

解読して実行すると云う表現は適切ではなく、実行時にオブジェクトコードを生成する訳である。

何故中間コードを作成し且オブジェクトコードまでも作成するのかと云うと中間コードは修正をリコンパイルなしで済ませモデル作成を効率的に進めるためのものであり、オブジェクトコードは、実行に際し、繰り返し何度もステートメントを実行する時に中間コードを解読し直す手間を省き最少限の労力で実行させるためのものであると云う認識に依っている。

従って OPS－4 が実行速度の向上として唱えたコンパイルと云うものは、中間コードの再解読を防止する事による効率化であって、ステートメ

ント単位のものである。

PL/I コンパイラへの入力となるプログラム単位は、実質的にはOPS－4 プログラムの中から呼出すようなプロセデュアと等価にされる。

従って最初に述べた標準形式の PL/I プログラムにコンパイルされると云う事は事実ではあるが、プロセデュアに変換されますと云う表現の方が適切でありこのプロセデュア呼出しを OPS－4 プログラムの中に記述すると、実行時にエクゼキューションセグメントがサーチされコンパイル済みと分かりそこへ制御を移すと云う事になる。

コンパイルされたプログラムが1ステートメントと等価になってしまうと云う事が特徴となっている。

コンパイルすべきプログラムにトレースに関するステートメントがあってもそれは無視される。

従ってコンパイルしてしまったプログラムとは、ユーザは自由に対話する機会は無い。

このように見てくると OPS－4 のコンパイルは、 SIMBOL のプロセデュア単位のコンパイルと非常に相似的である。

この事実はオンラインシステムに於ては、対話を除去し得ない事に由来するものであろう。

概念的に見るなら OPS－4 のコンパイルされたプログラムの実行が SIMBOL のインタープリーティブな実行と等価でありながらも、最終的な意味付けが同様な所に帰着されている事がこの間の事情を物語っている。

即ち、コンパイルとは、実行速度の向上を目的に行なうものであるが、それはユーザとの対話が不要な部分に限るべきであると云う基本的な考え方である。

OPS－4 のように一見全てコンパイル可能な印象を与えるにもかかわらず、プロセデュアと云う形態しか許されないならば、むしろ何故プロセ

デュアしか許さないのかと云う事を明確にすべきであろう。

SIMBOL ではシミュレーション研究と云うステップを対話型システムでサポートする場合のコンパイレーションと云う方向からC－形式の意味付けが成されている。

SIMBOL では、シミュレーション研究と云うものを図3.1 3に示すようなものとして把えている。

人間の判断・アクションとして示したボックスは人間のポテンシャルエネルギーを表現しており、閾値を超えた時にアクションが成される事を示している。

プログラムの進行として示した部分は、人間のアクションに基づき、所望の機能を果たすべくプログラムが作動される事を示している。

シミュレーション実験と云う軸を人間側と SIMBOL 側が動作する事に依り目的が達成される。

SIMBOL側のプログラムは人間が機能を与えるものである。

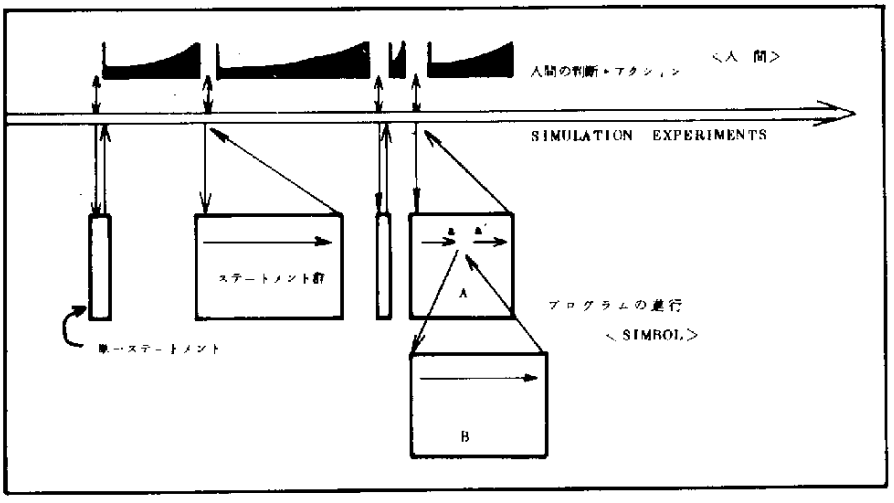


図3.1.3 対話型システムの特性

1ステートメントが最小の機能単位を表現し、ステートメントがいくつか集まってより次元の高い機能を果す事になる。

1つの機能を遂行中に人間が割込みを起こして SIMBOL と対話をする事は許されない。

即ち最小機能単位と表現した由縁であって、これ以上分割する事が無意味である機能単位に無理矢理割り込んで再分割する事は違法であり、まさに何の意味も持たないし混乱を招くのみであるからである。

又人間の側から云っても、然るべき機能を果したか否かをチェックする事は比較的容易であるが、混沌とした中から一体どんな機能を果しているのかをチェックする事は至難の業であろう。

従って機能遂行中の混沌とした状態をユーザが知り得なくとも問題は起らないし、いたずらにユーザを刺激しない上でも最小機能単位と云う考え方は必要であろう。

例えば図3.1.3に示したプログラムA, Bに於て、プログラムAのa点に来た時（プログラムBを呼出す前）とa′点に来た時（プログラムBの実行後）とで或るデータがどう変わるかを知ろうとしても、割り込みはきかない。

もしそのようなトレースが必要であるならプログラムAを2つの機能単位に分割すべきである。

コンパイルしてしまったものは、1つの機能単位でありその再分割は許されない。

対話が必要ならば、対話が取れるような機能単位にプログラムを構成すべきである。

機能が明確にされた時、処理プロセスのチェックは意味を持たない。

何故ならインプットが決まれば一意にアウトプットも決まるからであり両者をチェックすれば機能が正しく果されたか否か判断し得るからである。

逆に云えば、いくつかの機能単位の集合として1つの機能を表現する場

合、即ちコンパイルする場合、そのように表現された機能はインプットとアウトプットの関係で十分チェック可能な単位を表現していなければ意味がないと云える。

理想的には、人間が対話を必要としないと分かっている部分を1つの再分割不能な機能単位に構成する事が望ましい。

しかし人間の側からすれば試行錯誤を基調とするのであらかじめそのような機能単位を構成する事は不可能に近い。

従って実験を進めて行く過程で、機能が明確に把えられたものの処理効率を高めるために、プロセデュアのC－形式を可能にしている。

コンパイルされたものが分割不能な最小機能単位を表現すると云う思惟は、再分割不要なプログラム単位であるプロセデュアと云うものを暗に示唆している。

このようにSIMBOLでは対話を基本とし、対話を生ぜしめる機能単位と云う考え方を導入し、対話不要な単位を対話可能な処理形式で処理する無駄を省くと云う事からプロセデュア単位のコンパイル即ちC－形式と云うものを導いている。

これはSIMBOLが取った解決策であるが、対話型システムがSIMBOLと同じ様な方向に在るべきである事を意味するものではない。

しかし乍ら少なくともシミュレーション研究にとって対話が最終的に不要なものであるのか否かと云う問題に対しては、如何なる対話型システムであっても明確に答えなければならないであろう。

実行速度は早い程良いが、対話型システムの主眼である人間との共同に依る問題解決と云うものを阻害するなら目的なきシステムに堕する訳で、その意味に於ても、SIMBOLが取った解決の方向は、高く評価されて然るべきであろう

3.2 メインプログラムの位置付け

ＳＩＭＢＯＬでは、最初に実行されるべきプログラム単位としてメインプログラムを用意している。

メインプログラムは、シミュレーションの開始・終了・実験過程等を制御するための特別なプログラム単位であり、システムライフ中その特性を変えないデータ、或は共通に使用して差支えないデータ等、グローバルデータと呼んでいるものを定義するプログラム単位でもある。

メインの中には少なくとも１つの ＜ＳＴＡＲＴ＞ ステートメントを記述し、最初に実行すべきアクティビティを指示しなければならない。

勿論シミュレーション以外の目的で ＳＩＭＢＯＬ を使用する場合は必要ない。

この様にメインを特別なプログラム単位として扱う事に依り、非常に簡単にシミュレーションランの制御が可能となり、ラン間の解析も行ない易くなっている。

反面プロセスタイプの言語形式を取っている事から、この様な特別なプログラム単位の導入は概念の不統一を招いている。

ＳＩＭＢＯＬ設計当初は、メインプログラムを特別なものとして扱わずにプロセスとして扱おうとする考え方があった。

いわばメインプロセスとでも呼ぶべき性格のもので、最初の実行は ＜ＳＴＡＲＴ＞ コマンドで指示する形態を取るものであった。

メインプロセスとして扱うようにすれば全体の概念が統一され、言語仕様上好ましいと考えられる。

他方プロセスと同様の扱いが適用される事に依り、種々の技術的制約と対峙しなければならなくなる。

その１つとして、プロセスに許されているステータス・セーブ機能がある。

何をセーブしておけば再実行可能であるかを検討し、セーブは必要最小

限のもので済ませなければならない。

又、リスタートの制御方式に関する問題も解決しなければならない。

スタート・ストップ等をメインプロセスで使える様にするのが技術的にむずかしく、コマンドの形態を取らざるを得ない。

又その事によって、実験計画がモデル内に組み込みずらくなる等、より大きな犠牲を強要される事になる。

この様に概念の統一を求める方向は、解決すべき問題をより多く作り出してしまうベクトルとして働き、技術的制約を超えて、達成し得たとしても効率のよいものは望めないであろうと云う結論に帰着された。

多少の概念の不統一があったとしてもユーザが使い易く、且開発工程も短縮される方向の方がより好ましいとの判断に基づいて現在のメインプログラムが位置づけされている。

では我々が比較に用いたパッケージではどの様な扱いが成されているかを次に検討しよう。

SIMBOLのメインプログラムと同じ様な性格を持つSIMSCRIPT-Ⅰから順次眺めてみよう。

表3.2.1 SIMBOLのMAIN PROGRAM

```
0010 INTEGER RMATRIX(10,10),LMATRIX(10,10);
0020 INTEGER LSTATE(28),BUFC(10);
0030 INTEGER U1,U2,U3,I,J,K,MS,BSIZE;
0040 REAL MDIST(10,10),CRGDIST(20);
0050 REAL X,Y,ENDIMP,LSPEED,TRANSTM,ARRIVT;
0060 QUEUE LQUE(28);
0070 START GENERATE;
0080 END;
```

3.2.1 SIMSCRIPT－Ⅱの場合

S－Ⅱは、メインプログラムと云う考え方を導入しており、これが最初に実行される特別なプログラム単位である点 SIMBOL と同様である。

データ構造をスタティックに決定するため、データ構造をPREAMBLE宣言ブロックで定義する。

S－Ⅱに於ても、メインプログラム・イベントプログラム・サブプログラムの3種類のプログラム単位があり、イベントプログラムはスケジュールされて実行されるものであるし、サブプログラムも呼び出しが無ければ実行されないため、両者とも直接実行出来るプログラム単位ではない。

従って全プログラムデックに対し少なくとも1つの非サブプログラムが必要である。

これをメインプログラムと名付けており、プログラムデックがコンパイルされ実行のために主記憶にロードされた段階で、メインプログラムの最初の命令から実行されるようになっている。

プログラム単位の定義は、次のものから成っている。

① プログラム単位定義ステートメント

② プログラム単位に与える名前

③ データ伝送メカニズムの記述

その後にプログラム本体が記述され、プログラムの完全な終了を知らせる END ステートメントでしめくくられて始めて1つのプログラム単位が完了する。

メインプログラムの場合 MAIN ステートメントは必ずしも必要ではない。

これは他のプログラム単位に必ずどのプログラム単位であるかを示すヘッダーがあり、且つモデルに対して1つのメインプログラムしか許し

ていないためである。（SIMBOL も同様である）。

実例を表 3.2.1 に示す。

表3.2.1 SIMSCRIPT－Ⅱに於ける Main Program 例

```
**
**      /*  INITIALIZATION  &  START  SIMULATION  */
**
        MAIN
            "INITIALIZE"          PERFORM  INITIALIZATION
                LET  BETWEEN.V="TRACE"
            START  SIMULATION
**
**      /*  PERFORM  NEXT  SIMULATION  EXPERIMENTS  */
**
            START NEW PAGE
          WRITE  HOUR.F(TIME.V) AS 8 3C,
           "*=*=* N E T W O R K    F L O W    S I M U L A T I O N *=*=*",/,/,/,
            8 7C," START TIME=",I 2,"HOUR"
STOP          END
```

この様にS－Ⅱでは、あらかじめメイン乃至はメインに付随するサブプログラムで必要なイベントプログラムをスケジュールしておけば、ライン7で示した

START SIMULATION

ステートメントが実行されると、タイミングルーチンに制御が渡りイベントリストの先頭にスケジュールされているイベントプログラムの実行を開始する。

このステートメントはシミュレーションの開始の機能しか持っていないが、適当な終了条件が満たされて再びメインプログラムに制御が戻った時点で、ラン結果を解析し、パラメータ等の修正を行なって再度シミュレーションを再開出来る。

実験計画をメインで組める点 S－1.5 よりも融通性があり、使い易くなっている。

3.2.2　SIMSCRIPT－1.5の場合

S－1.5の場合にもメインプログラムと云うものが存在する。

プログラムの起動は外生事象（EXOGENOUS EVENT）制御カードの読み込みにより行なわれる。

従って最初に実行されるプログラム単位は常に外生事象プログラムである。

外生事象プログラムの中で内生事象プログラムのスケジュールを行なう事によりシミュレーションが開始される。

従って少なくとも1つの外生事象が必ず必要で、これがメインプログラム的な役割を果す事になる。

MAIN〔t〕

t　　任意の文章

非シミュレーションの場合は、event list が不要となり、timing routine が作られない。その代りに、main ルーチンを書かなければならない。MAIN ステートメントは、main ルーチンの最初のステートメントとして用いられる。main ルーチンの最後のステートメントは END である。

```
EVENTS
   1        EXOGENOUS
            CNTL(1)
   5        ENDOGENOUS
                MSGCL
                ENDSV
            SMOP1
            SMOP2
            SMOP3
END
```

図3.2.2　イベントリスト

イベントリストに外生事象の個数・名前・識別番号を登録する。

識別番号は初期設定終了後に外生事象制御カードが読み込まれる時に、対応関係を取るためのものである。

exogenous event card（またはテープ）とは、exogenous event が、何日の何時何分に起こるかを指示するためのデータ・カードである。

event card の形式は、つぎのとおりである。

column 1～3	exogenous event の identification number
column 4～7	日
column 8～10	時　event を起こす時刻
column 11～12	分
column 13～72	もしあれば任意のデータ

表3.2.3 SIMSCRIPT 1.5 の MAIN PROGRAM 例

```
      EXOGENOUS  EVENT  CNTL
C
C     /*  SIMULATION  EXPERIMENTS  CONTROL  PROCEDURE  */
C
          SAVE
          READ  NO
          FORMAT(I6)
          IF (NO) EQ (0) , GO TO L5
          CALL  PRINT
          IF  (NO) EQ (1),GO TO L7
          LET  NO=NO-1
          DO TO L1, FOR I=(1)(28)
               LET XPRM(NO,I)=BUF(I)
               LET YPRM(NO,I)=TRNST(I)
      LET  APRM(NO,I) = UTIL(I)
               LET LPRM(NO,I)=LOST(I)
L1    LOOP
          IF (NO) EQ (3) , GO TO L3
          IF  (NO) EQ (1),GO TO L7
          DO TO L2, FOR I=(1)(28)
               LET  BUF ( I ) = 8
               LET  TRNST(I)=200
L2    LOOP
          GO TO L7
L3        DO TO L4,FOR I=(1)(28)
               LET  BUF ( I ) = 6
               LET  TRNST(I)=1200
L4    LOOP
          GO TO L7
L5        DO TO L6,FOR I=(1)(28)
               LET  BUF ( I ) = 8
               LET  TRNST(I)=1200
L6    LOOP
L11       CALL RESET
          CREATE  MSGCL
          CAUSE  MSGCL  AT TIME
          GO TO L10
L7        IF  (NO) NE (3),GO TO L11
          CREATE  SMOP1
          CAUSE  SMOP1  AT TIME+0.3
L10    CALL  BCON
      RETURN
      END
```

イニシャリゼーションカードに続けて、外生事象制御カードを以下の様に続ければ指定時刻が来る度に起動される事になる。

表3.2.4 Exogenous Event Cards

イベント番号	生起時刻	データ
1	0	0
1	60	1
1	360	2
1	660	3
1	960	4

S－1.5の場合シミュレーションを外部から一応制御出来る訳だが、前述の通り時間軸上での制御のみに限られるため、状態に依る制御を時間に依る制御に変換し直さなければならない。

この様な変換が常に可能である訳ではないので、非常に限定された形の実験計画しか組み得ない。

これはメインプログラムと云うものがシミュレーションランを制御する性格のものとして提供されていない事に起因するものであろう。

3.2.3 GPSS—V

G－Vではブロック命令の開始を＜START＞コントロールステートメントで制御する方式を取っている。

＜START＞ コントロールステートメントは次の2つの機能を有する。

1. モデルの初期設定が終了しシミュレーションの実行に入る事を指示する。
2. シミュレーション終了条件（消去すべきトランザクションの個数）の指示。

表3.2.5　GPSS－V の Main Program 例

```
          ********** TIME CONTROL **********
34            GENERATE    ,,600000,1,,4PH,4PL   *** 600 5 ***
35            ASSIGN      1,10,,PH
36            ASSIGN      2,41,,PH
37            ASSIGN      3,QA*PH1,,PL
38            ASSIGN      4,X*PH1,,PL
39            SAVEVALUE   PH2,V2,XL
40            ASSIGN      2+,1,,PH
41            LOOP        1PH,*-4
42            TERMINATE   1
              START       1
```

G－Vのプログラム単位はGENERATEブロックで始まり TERMINATE ブロックで完結したものを1つの単位として考えており、この様なプログラム単位が複数個用意される事に依りモデルが構成される。

その様なプログラム単位の内の少なくとも1つのプログラム単位にはTERMINATE ブロックにターミネーションカウントが記述されていなければならない。

GENERATE ブロックは、インプットの段階で擬似トランザクションが作成され、次に実行すべきブロックが対応するGENERATEブロックとして指定されて FEC（Future Event Chain）に連がれる。

発生時刻は勿論GENERATEブロックのフィールドオペランドから指定されたものが取られる。

TERMINATE ブロックではトランザクションの消去を行なうと同時にもしオペランドにターミネーションカウントが指定されていれば、START ステートメントのオペランドの数値から指定された数値を差し引いて行く。

表3.2.5 に示した例では42番のブロックが実行され、START ステートメントからターミネーションカウント1が引かれた時1回のシミュレーションが終了する。

＜START＞ ステートメントが実行されると GPSS-SCAN（時

問制御ルーチン）に制御が渡り、CEC（Current Event Chain）のスキャンを開始し、トランザクションのブロックへの移動が行なわれる。

＜START＞ ステートメントが実行される度にシミュレーションランが開始されるので、初期設定用のステートメントと組み合わせる事に依り多数回のランを計画出来る。

初期設定用のステートメントの機能が豊富な為、通常のシミュレーションランに於ては余り不便さを感じない。

むしろ使い易いとさえいえるかも知れない。

しかし乍らラン結果を解析する手段に欠けるため、ラン結果を生かして次のランを計画する事が困難である。

これもメインプログラムという考え方が無い事に由来する。

3.2.4 SIMULA—65，67

SIMULA－65 は ALGOL－60 の拡張であり、ALGOL－60 をサブセットとして含む言語である。

BEGIN と END ではさまれたステートメントが1つのプログラム単位を構成している。

これをブロックと呼んでいるが、SIMULA ブロックと云う特別なブロックがあり、これがシミュレーションに際してメインプロセスとしての役割を演ずる仕組になっている。

SIMULA ブロックは ALGOL ブロックの中の1つのステートメントであると考えられ、ALGOL プログラムは1つ以上の SIMULA ブロックを含む事が出来るようになっている。

SIMULA ブロックが SIMULA ブロックを含む事は許されない。

SQS（Sequencing Set）と呼ばれるスケジュールテーブルは、その SIMULA ブロック単位に固有であり、時刻 ϕ にはメインプロセスの最初の実行ステートメントに制御が渡るように初期設定される。

SIMULA に導入された概念は、SIMULA ブロックの中でのみ有効である。

モデルとして最初に実行されるのは、ALGOL プログラムの先頭のステートメントである。

シミュレーションの終了は SIMULA ブロックの END ステートメントの実行か、SIMULA ブロックから GOTO ステートメントに依り飛び出した場合に起こる。

SIMULA ブロックをいくつでも ALGOL プログラムの中に組み込めるので、何回でもラン可能であるし、ラン間の解析をして最適化を図って行く事も可能である。

且概念もプロセスとして統一されており非常にスマートな言語となっている。

SIMULA の場合にはステータスセーブ機能を持たないし、インプット順序に制限があり、且最初に実行すべきステートメントが容易に識別出来るようになっているので、SIMBOL が抱えた様な問題が起こらなかったのであろう。

表3.2.6　SIMULA－65　Main　Proceso の例

```
QQ.ALG.ISCD     SIMS.SIMRB
CYCLE  000  COMPILED BY  1204  0005  ON 02/02/73 AT 16:09:49
      B1
    1             BEGIN
     B2    S1     L1
    2             SIMULA  BEGIN
    3             INTEGER ARRAY  RMATRIX(1:10,1:10);
    4             INTEGER ARRAY  LMATRIX(1:10,1:10);


   17             FORMAT    F6(3(10,D10.3,A1));
   18             FORMAT    F7(X10,'LINE MATRIX',A1);


   53             READ(U1,U2,U3,ENDTIME);
   54             READ(RMATRIX);
   55             READ(LMATRIX);
   56             READ(ORGDIST);
   57             READ(MDIST);
   58             WRITE(F5);


   84        CREATE:  ACTIVATE NEW MESSAGE(K,J) DELAY 0.0;
   85             HOLD(NEGEXP(1.0,U3));
   86             IF TIME LSS ENDTIME THEN GO TO NEWMESG;
   87             WRITE('SIMULATION END');
     E2    F1
   88             END;
     E1
   89             END;
COMPILATION COMPLETE
```

SIMULA ブロック

最初の実行ステートメント

3.3　ステージの概念の有効性

SIMBOL に於ては、シミュレーション実験を大きく２つのステージに分けて考えている。

１つはモデルを構成するステージであり、対象システムをモデルプログラムとして構成し、キャラクターディスプレイからキーインする作業単位である。

他の１つは、構成したモデルの妥当性をチェックするためのステージで

あり、論理シーケンスのチェックや動作解析を行なう作業単位である。

最後にモデルの動作解析に依りシミュレーション目的を果そうとするステージのためのシミュレーション実行中と云う作業単位がある。

これらをそれぞれモデリング・シミュレーション・エクゼキューションステージと名付けており、特有の機能を持たせている。

最後のエクゼキューションステージは、シミュレーションステージのサブステージである事から、主要ステージから除外している。

我々が比較対象に用いた5つのバッチシミュレーション言語に於ては、ステージと云う考え方は無い。

勿論 GPSS で述べられるようなインプットフェイズであるとか、アセンブルフェイズ、エクゼキューションフェイズと云った概念は、SIMBOLにも適用されるものである。

然し乍ら SIMBOL で云う所のステージは、GPSS 等で云われるフェイズとは意味的に異なったものである。

即ち GPSS 等の作業単位と云う概念は、シミュレーションシステム側から眺めた処理単位であり使用者のアクションが含まれないものであるのに反して、SIMBOL で規定する作業単位と云うものは、使用者側から眺めたものであると云う事である。

この事は単に視点を変えた事のみにとどまらず、作業単位の実質的な意味内容をも変貌させる事になる。

ではその実質的な差異がどこにあるのかを探って見よう。

シミュレーションの基本的なステップは次図（3.3.1）に示すようなものであろう。

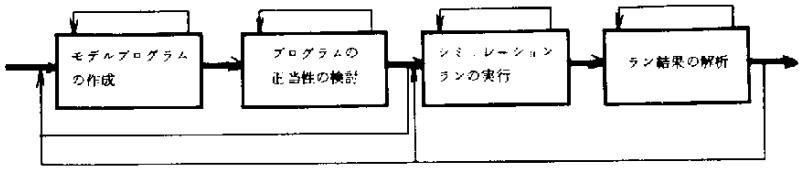


図3.3.1　シミュレーションのステップ

太線で示した流れに沿うボックスが基本的なステップであり、太線の下側に示した矢線が前段のステップの修正を　上側で示した矢線で自分自身のステップの修正を表現している。

図式的に述べるなら、各ボックスを結ぶ太線はパイプに、実際の行為はパイプを流れるボールに各パイプを連ぐボックスはタンクに相当し、パイプの太さが最適解を表現していると見る事が出来よう。

ボールの大きさが行為の質を示すと見て良い。

行為の質の向上と共にボールの径は膨張し、最適行為の選択が成された時ボールの径はパイプの径と一致する。

この時に最適なシミュレーションが成され目的が達成される。

一般に最初からパイプの径と同じ径を持つボールを選択しパイプの中を転す事は不可能であり、何回ものフィードバックの蓄積効果に依りボールの径を成長させるしかない。

この時フィードバック用のパイプが詰っていたのでは、ボールを前段に戻す事も出来ない。

従ってフィードバック用のパイプは、ボールをスムーズに転がし得る様設計されていなければならないだろう。

又基幹となるパイプの流れの方向は、ボールを転がしたい方向と一致していなければ無意味であろう。

更には基幹パイプを結び合わせるタンク（ボックス）は、ボールの成長にとって本質的に有意義でなければならないはずである。

SIMBOLではフィードバック用のパイプとして各種コマンド類が、基幹パイプとして主要コマンドが、タンクとしてステージが対応する様設計されている訳である。

そして図式的に述べたタンクを更にリファインした形で、"作成"と"検討"を主要タンクとし、"検討"の補助タンクと云う形で"実行"と云うものを把えている。

これはまさに"PLAN", "DO", "SEE"の関係と対応するものである。

この様な把え方をした時、従来のバッチシミュレーション言語は総じて"DO" 即ち、"実行"タンクしか持たなかったと云えるであろう。

では他のタンクは不必要であったのかと云うと決してそうではなく、まさに試行錯誤のボール転がしゲームの最大の関門である"SEE" 即ち"検討"タンクを設定すべき場所を見い出し得なかったが為に、その部分を使用者の頭脳に設定せざるを得なかったと云うのが真相であろう。

何を検討すべきか、どの時点で検討すべきか、と云った問題が使用者固有の問題として有る時、ボールがパイプの途中を転っている様子をつぶさに観察して転がり具合を制御し得る様にするには、パイプの途中から手を入れてボールの回転を制御する以外に方法が無いであろう。

この為には、ボールの回転の様子を見たい時に見えるようにしなければならないし、静止断面や回転方向の解析が出来、制御が加え得る様にパイプやタンクの構造を改造しなければならない。

この様なボール転がしのシステムは、対話形式のシステムに於て始めて可能である。

SIMBOLの様な対話型システムであれば、使用者は自分の頭脳の延長として SIMBOL を使用し、共同作業として問題解決にあたる事により効率的に実験を行なう事が出来るであろう。

この様に人間の思考活動に相似のタンクとパイプを用意している事に依

り、従来のバッチシステムに比べ遙かに人間の負担が軽減される。

逐次確認しながらモデルを構成し、モデルの振舞を逐一観察しリファイン出来る事により、人間の思考の中断も抑えられ全体のパーフォーマンスも向上する。

動作過程を飛び越して最終結果を目の前につき出すバッチシュミレーションシステムでは到底得られない非常に強力な機能であると云えるであろう。

実際の様子は第1節の SIMBOL の概要紹介から推察される通りである。

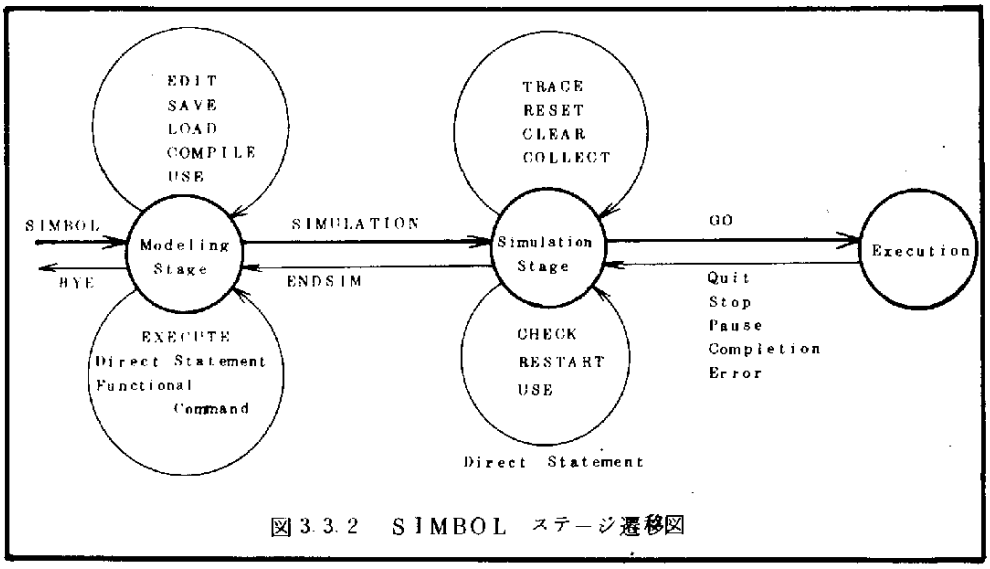


図3.3.2 SIMBOL ステージ遷移図

3.4 ファイルの取扱い

3.4.1 ファイルの取扱い

SIMBOLではユーザ固有のシミュレーション用ファイルが必要である。

固有に割当てられたファイルをいくつかの区画に再分割して使用出来

るような分割型順編成ファイルの形態を取っている。

ファイルには、必要に応じてユーザが構成したモデルのソースプログラムやステータスセーブに依る実行形式プログラム、或はコンパイルされたプロセデュアなどが格納される。

ユーザが始めて SIMBOL を使用する時は、使用に先立ち、SIMBOL が提供するファイル開設用のユティリティに依り、ユーザ固有のファイルを創成しておかなければならない。

二度目以降はユーザ ID に依り開設済みのファイルがいつでも使用可能となる。

ファイルは3種類のものを用意しなければならない。

ソースプログラムをセーブするためのソースファイル、実行形式プログラムセーブ用の R・B（リロケイタブル・バイナリ）ファイル、ステータスセーブ用の Save ファイルである。

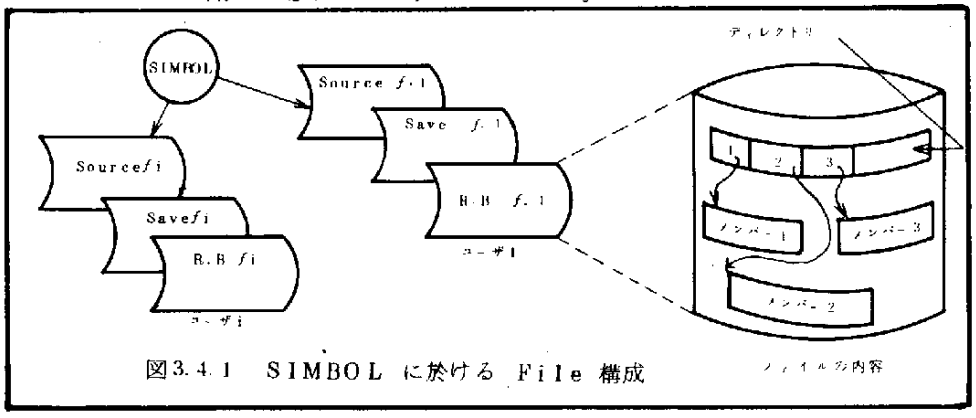


図3.4.1 SIMBOL に於ける File 構成

ユーザ自身にファイルの開設を SIMBOL の使用に先立って行なわせる事には問題があり、出来ればファイルの開設は要求が生じた時点で行なえる方が良い。

そのためには、SIMBOL 自身がファイルを持ち、ユーザ要求時に自分のファイルの一部をユーザ用ファイルとして与える方法が考えられる。

これに依れば、SIMBOL の中でファイルに関する一切の処理が行

なえるようになりユーザも手軽にファイルを使用する事が出来る。

実際にこの様な方法でファイルの使用を許せれば都合が良いのであるが、現在 SIMBOL がインプリメントされているオペレーティングシステムの制約上実現するのは困難である。

SIMBOLが開設したファイル空間を多数のユーザに分割し、その各々にプロテクションをかける必要がある。

これはパスワードの様なユーザ個別のコードに依り簡単に行ない得る。

ユーザ個別に分割されたサブファイル空間を更にユーザの作成したモデルないしはプログラム単位に再分割しなければならない。

これを分割型順編成ファイルとして実現しようとすると、現在のOSの制約上、一度フリーにされた空間を再度利用する事が出来ないし、ユーザ用に分割した空間を拡張するには、より広い別のファイル空間へコピーする事によりこれを行なわなければならない。

更には、 SIMBOL のファイルを分割提供する事に依り、障害が発生した場合ファイルの内容を再現出来ない事も起り得る。

直接編成ファイルとして構成した場合でも、空間の分割管理はうまく行なわれても、障害に対し、現在使用中でないユーザズファイルさえも破壊してしまう危険性を回避し得ない。

この様な情況である事から、 SIMBOL ではユーザにファイルの開設を個別に実行させる事で危険性を回避している。

勿論 SIMBOL 開始後でもファイルを創成する機能があり、且上述の問題を全て解決し得れば、その方法を採用すべきであろう。

バッチシミュレーションシステムでは、一時的なファイルとして例えば GPSS－V が用いる様な磁気テープファイル等があるだけで、これもステータスセーブ用のファイルであり、それ以外はOSの機能を用いてファイルを使用する程度である。

ソースファイルと云うものを導入しても、それをメインテナンスする

機能をバッチシミュレーションシステムでは提供していないので、無駄なものに過ぎない訳である。

3.4.2 ファイル操作

ユーザの創成以降は SIMBOL がオールドファイルとしてメイテナンスする分割型順編成ファイルのメンバーの扱いに関しては、基本的にはメンバーのセーブとロード・デリートの3種類のものがある。

セーブとロードを適当に組み合わせて用いる事に依り、メンバーの追加・削除・修正が行なえるようになっている。

●セ ー ブ

セーブしたいプログラム単位に適当なメンバー名を付けて SAVE 又は CHECK コマンドを実行させると、対応するファイルにメンバーとして登録される。

SAVE はソースプログラム単位や実行形式プログラム単位をそれぞれ、ユーザが創成したソースファイル、R. B ファイルにメンバーとして登録する。

CHECK コマンドは、現在実行中のモデルの状態をセーブファイルに格納する。

この時対応するファイル上に同じメンバー名のものが存在する場合、同じ名前のものが存在する旨告知され、ユーザのアクションを求める。

実行してかまわないと云う事になれば先のメンバーの削除に引き続き、新メンバーが追加される。

同一メンバーが無い時は新規登録である。

●ロ ー ド

ソースファイル・R. B ファイルから主記憶へメンバーを呼び出すに

はLOADコマンドを、セーブファイルからの呼出しはRESTARTコマンドに依る。

ソースファイルから呼び出したメンバーに限って修正を施す事が出来、この結果を再度セーブする事に依り実質的にファイル内のメンバーの修正が出来るようになっている。

●デリート

DELETEコマンドに依り、全てのファイルから希望するメンバーを削除出来る。

DELETEされたメンバーが占有していた空間をその場で利用する事はOSの制限に依り出来ない

この様に極めて限られた機能しか提供していないにもかかわらず、実用上は何ら差しつかえない。

又既存のプログラムを変更したり消去するたびにファイルにアクセスするので、ファイルアクセスが高速に行なえるようなファイル編成を取ったり、ユーザ単位に完全にファイルプロテクションが行なえるようになっている点などユーザに対する配慮がうかがわれる。

3.5 SIMBOLエンバイロンメント

SIMBOL は FACOM 230 － 60／MONITOR－V の下で動作するシステムである。

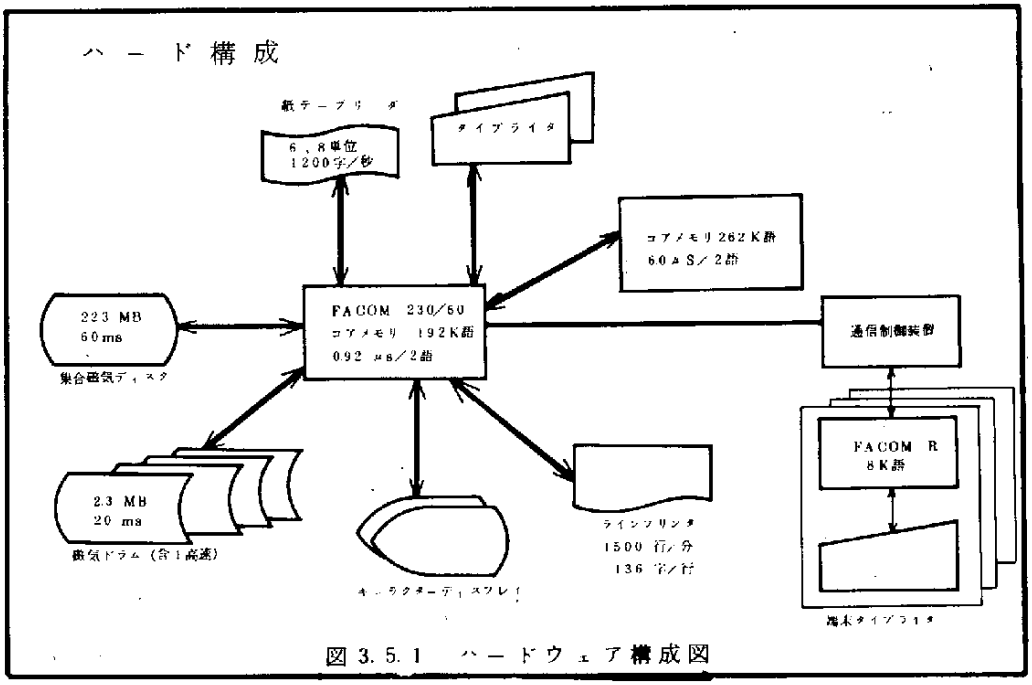


図 3.5.1 ハードウェア構成図

SIMBOLは、端末として普通のタイプライタ端末とキャラクターディスプレイを組合わせて、２台一組にして使用している。

モデルビルディングやプログラムのテスティング・エディティング等その９割がたはキャラクターディスプレイを使用して行なう。

ところが現在提供されているキャラクターディスプレイは一般 I／O 機器と何ら変わる所が無く、ジョブの起動や実行中のプログラムに対する Quit 機能を持っていない。

従って現在の SIMBOL システムのハードウェア構成は、図3.5.1に示した様な形態を取っており、ジョブの起動や実行中のジョブへの割込みをタイプライタ端末から行なっており、合わせてキャラクターディスプレイのハードコピー用としても使用している。

ジョブの起動はともかくとして、 Quit の為にタイプライタ端末を操作しなければならないという現在のハードウェアから来る制約は、対話型システムの本命とも云える手軽な操作に依る円滑なシミュレーション実験の効果を半減してしまっている。

ハードウェアとして Quit 可能なキャラクターディスプレイを使用出来るなら絶対に変更すべきである。

対話型システムに於て人間系と機械系の窓口を形成する装置の操作性が劣ると云う事は致命的な欠陥である。

更には、現在のディスプレイ装置にはファンクショナルキーに相当するものが無いので、一段と操作性を低めている。

SIMBOLの場合、使用可能な機器の機能が劣る為いたしかたないと云わざるを得ないが、対話型システムを構成する場合のインターフェス機器はその重要性を強調しても強調しすぎる事は無いであろう。

実用性を狙いとするならば、まず人間工学的な立場から端末の在るべき姿を追求する姿勢が必要であろう。

SIMBOLで使用しているキャラクターディスプレイは、文字サイズが一通りしかなくグラフィカルな機能が非常に劣り、高速なタイプライタ端末の方が文字の表示に限っていえば勝れているとも云える。

このように現在使用中のディスプレイ装置はいくらかの難点を持っており、キャラクターディスプレイ特性を SIMBOL システムに組み込む事に多少の無理があるにもかかわらず、他のディバイスを導入するよりは効果的である側面をいくつか持っている。

まずキャラクターディスプレイ装置は、グラフィックディスプレイ装置に比べ、コスト的に数十分の一程度で済み、比較的手軽に使用出来る。

画面操作のための特殊なソフトウェアを必要とせず一般ユーザが十分使用可能であり、ソフトウェアの開発も比較的簡単である。

グラフィックディスプレイ程の機能を持ってはいないが、バーチャート

や折線グラフの類は一応表現出来るし、複雑な関係をより速やかに、より理解しやすい形で表現出来る。

プロッターやタイプライタと違って、中間結果をプロットしたりタブレイト表示するにも時間がかからないし、ダイナミックな変化の様子をリアルに表現する事も可能である。

ソフトコピーアウトプットの便宜があり高速表示・表示部分の修正による完全なグラフの再表示等が簡単に可能である。

ソフトコピーと云う言葉をCRT装置上への表示の意味で用いている。

タイプライタ等にグラフを表示しようとすると、空白を取るためのスペーシングの為に何度も何度もプリントヘッドが遊ぶ形になり非常に時間がかかるし修正のためには、グラフ全体を再構成しなければならず、又修正結果を即座に確かめる事も出来ない。

更にSIMBOLではキャラクターディスプレイから＜COPY＞コマンドをキーインする事により簡単にハードコピーが得られる。

タイプライタ端末をハードコピー用に用いているため、多少の時間はかかるが、全画面一杯のコピーでも10～20秒程度であり問題にはならない。

但し低速タイプライタ端末の場合（F1592A）、印字速度が375字／分である事から全画面コピーに2～3分程度は必要になる。

現在のキャラクターディスプレイ規格は、FACOM 6221B解説書に従うもので以下の通りである。

- 表示字数　　50字×20行（1000字）
- 文字サイズ　　4㎜（高）×2.8㎜（幅）
- 表示フレーム数　毎秒約40フレーム
- 表示色　　緑色
- 字　種　　128種（カタカナ、アルファベット、数字、記号、

スペース)

他特殊記号　4種

- コード　　ＩＳＯコード
- 転送速度　　約4Ｋ字／秒

Ｆ6221Ｂ　の機能は次の通りである。

- 打鍵表示
- 編集機能
- 受信データの表示
- 表示データの送出
- スプリット・スクリーン
- アドレッシング
- 部分転送

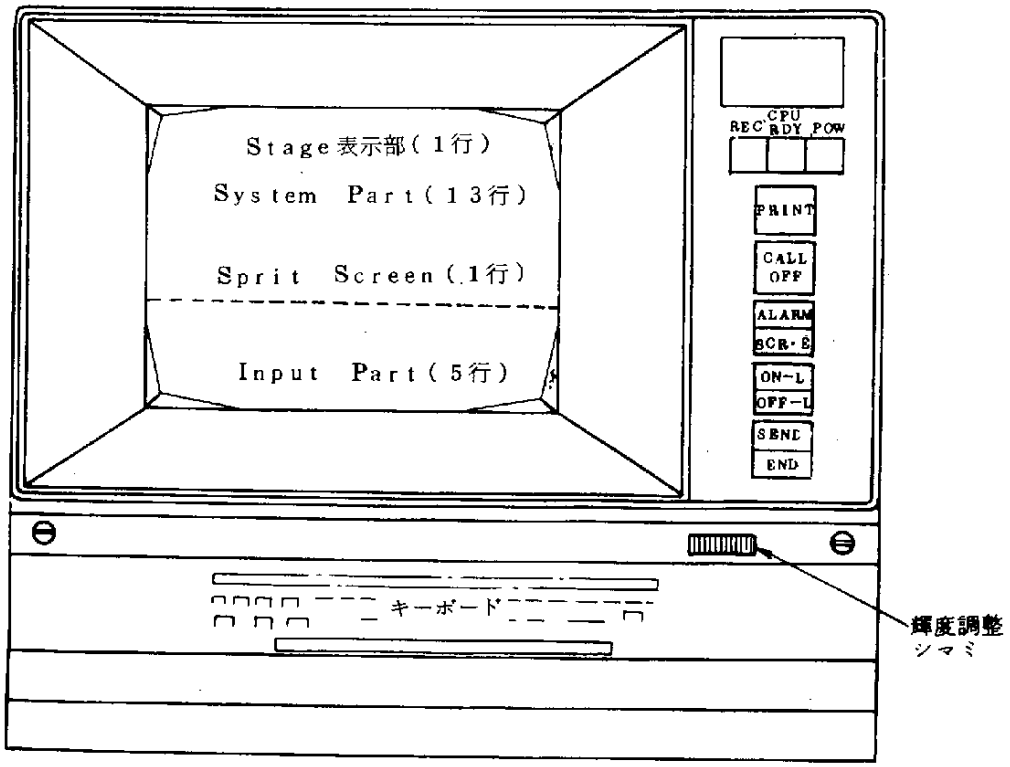


図3.5.2　キャラクターデイスプレイ構成図

SIMBOLでは有効画面をシステム側とユーザ側とに2分割して用いている。

ユーザ側をインプット部と呼び5行×50字（250字）迄が一度にキーイン可能な最大文字数となっている。

システム部は、最上段を SIMBOL が使用しており、ステージ、インプットモード、コマンド実行中等の状態を表示するために用いており、残り13行をユーザがインプットしたステートメントの表示や、データ表示に使用している。

13行ではグラム表示もし難いし、表示ステートメントも不足であり、少なくとも20行程度の有効表示画面が取れる方が好ましいようである。

スクローリングを有効的に使用する事に依り行方向の制限を無限大に拡大し得る訳だが、1回のパターン表示の内容が乏しい感を禁じ得ない。

しかしながらテキストエディティング機能が勝れており、修正が頻繁に行なわれる SIMBOL に於ては、修正が簡便に行なえ、修正結果を即座に確認出来る機能を持ったキャラクターディスプレイの使用は、多くの難点にもかかわらず対話型システムの目的を助長すると云う意味で効果的である。

更に図形データの表示と合わせてシミュレーションを行なう必要がない場合には、 SIMBOL が提供している程度の機能と性能でも十分であると云っても過言ではないであろう。

必要最小限の基本的機能を提供するだけでも、その効果がバッチシステムとは比べものにならない程大きなものである事がいっそうその感を強くする。

第4章　モデルビルディング

4章　モデルビルディング

この章では、ソースプログラムインプットや、ソースプログラムアップディト、エラーチェック、プログラム構成法等に関する問題を検討する。

4.1 ソースプログラムのインプット

対話型システムの場合はバッチシステムと異なり、プログラムのインプットの仕方も、コンパイルの仕方或はアップディトの方法等も段階的に行なわれる。

そのためにインプットの順序に制約を設けない事であるとか、ラインナンバーを必要とするとか、或はプログラム単位を明確に区分するためにインプットモードを設ける事であるとか、キーインのし易さ或は画面表示されたプログラムの扱いに関する事であるとか等々、バッチシステムでは見られなかった様な新しい問題に直面せざるを得ない。

この種の問題は対話型システムのまさに対話的である事を顕著に示すものでもあり、それ故にどういう思想的背景に基づいて対処したのか、対処のし方は妥当であったのかという視点で以下検討を進めている。

4.1.1 インプットシーケンス

〝SIMBOLでは、モデル作成が試行錯誤的に成される事に鑑み、入出力機器の制約である物理的なインプット順序に無関係に論理的順序が構成できるように、任意の順番でインプットしてさしつかえない便宜を提供している。〟

SIMBOLでは、ユーザがキャラクターディスプレイからインプットする最小単位をラインと呼んでいる。

ラインは SIMULA や SIMSCRIPT の言葉で云えばステートメ

ントに、GPSS で云うならブロック命令等と同様の単位であり、固有の機能を持っている。

先にも述べた通り、ラインにはコレクトステートメントとダイレクトステートメント（或はコマンド）との２種類のものがある。

コレクトステートメント（以下ステートメントと略記）は、キーインと同時にコンパイルが成され（インクリメンタルコンパイレーション）オブジェクトコードを含むステートメントインフォメーションが作成され、コードブロックに蓄積される。

後に指示が成された時始めて実行される。

コードブロック内では、キーインされたステートメント順にステートメントインフォメーションが蓄積されて居る訳だが、論理シーケンスは必ずしもインプット順に成ってはいない。

ステートメントのインプットと同時に即座に実行してしまう場合は物理的な入力順序が論理シーケンスと等価でなければ意味を成さないが、SIMBOLのようにインクリメンタルコンパイレーションと実行との単位が分かれているシステムでは、原理的には実行直前に正しい論理シーケンスが構成されているならば途中の入力順序というものは問題にならない。

そうは云っても、入力順序を正しいプログラムシーケンスと一致させなければならないという、バッチシステムと同様の制約をユーザに課した方が、プロセッサ構成上は非常に楽であり、開発工数も短縮出来る。

投入ステートメント単位にその場でオブジェクトコードを伴うブロックを作成しないならば、必要な情報が全て出揃った時点で効率良くコンパイルを行なえる訳で、オプティマイズされた形のオブジェクトを提供する事も可能となる。視点を変えると、これを押し進めた所に、SIMSCRIPT や SIMULA のバッチシステムが在るとも云えよう。

SIMBOLが、何故効率的な方法を取り得なかったのかという所にオ

ンラインシステムの特徴が顕著に現われている。

ＭＩＴのＯＰＳ－４やＳＩＭＰＬ、或はＣＭＵ（Carnegie-Mellon University）で行なったインタラクティブ ＳＩＭＵＬＡ 等は、シミュレーション研究に於けるステップの比較的早い時機からユーザと関わり合いを持つ事に依り、人間の負担を軽くし、問題解決の効率化を図る事を目的に計画されたものである。

ＳＩＭＢＯＬ自身も概念的には、ＯＰＳ－４等と同様の設計思想に立脚している。

比較的早い時機からサポートする事になると、システムの柔軟性が大きな問題となる。

実際にはシミュレーション研究のモデル作成のステップから関与しようとする傾向が一般的であり、モデル化以前のステップにスポットをあてようとするシステムは今の所まだ見あたらない。

モデル作成のステップは従来のバッチシュレーションシステムに於ては、ほとんどの部分が人間だけにまかされており、システムの関与する余地は無かった。

モデル作成のステップはアルゴリズムが確定している訳では無く、人間の創造的な思考活動に頼らざるを得ない部分である。

この様に不確定な部分からシステムが関与するので、人間の試行錯誤に依る行動を十分吸収し得る柔軟性が是非とも必要とされる。

その様な試行錯誤は、システム側から見れば次の様に解釈される。

即ち、人間が物理的な時間軸上で取るアクションは、真に人間が希望する所の論理関係の表現とは直接的な関わりを持たない。

この様な解釈に基づき物理的なアクションに論理的順序を規定する識別コード、即ちラインナンバーを付随させている。

このラインナンバーをキーとして分類を行なえば、所望の論理シーケンスが達成される事になる。

従って人間が試行錯誤的に成す行為を、人間が再編集する事なく、前回の試行を生かして核心に迫る事が可能となり、人間の思考活動の自由度を増し様々な試行を簡単に実現させる上で非常に強力なシステムと成り得る。

OPS－4 のようにオブジェクトコードに即座に変換しないシステムでは、インプット順序というものは上述の考え方だけで問題なく解決し得る。

所が SIMBOL のようにオブジェクトコードを作成するシステムでは、順序関係が乱された事に依り、オブジェクトコードを再編集しなければならない場合が生じる。

インプット順が攪乱される度に、オブジェクトコード群の再編集即ちリコンパイルが必要になってしまうのでは、バッチ汎用シミュレーションシステムと同等の性能を要求する SIMBOL にあっては全く実用にならない。

インクリメンタルコンパイレーションに依り実行可能なオブジェクトを作成する限りに於ては、人力順の攪乱を全くリコンパイル無しで吸収してしまう事は不可能に近い。可能ではあっても、プロセッサ側の負担が大きく効率は望み得ない訳である。

SIMBOLでは宣言ステートメントの修正を施した場合は、追加を除けば現在構成中のプログラム単位の全ステートメントがリコンパイルされる。

しかしそれ以外のステートメントの追加・削除・修正に関してはプログラム単位に渡るリコンパイルは成されない。

リコンパイルに関しては後節で詳述するが、対話型システムが何故対話を必要とし、又その対話の過程を通して何を解決しようとするのかと云う事を明確にする中で、物理的順序の制約の排除という事が考察された訳である。

そのような対話型システムに於けるインプットシーケンスの意味付けという問題のほかに、 SIMBOL 固有の問題もあった訳で次にそれを考察しよう。

先にも多少触れているように、 SIMBOL ではインクリメンタルコンパイレーションを行なっており、1ステートメントごとに実行可能なオブジェクトを作成する。

実行可能であるためには、コンパイルにさいして使われているデータの属性は全て明らかにされていなければならない。

すなわちそのデータは、グローバルなのかローカルなのか或はイクスターナルかと云ったデータ属性と、リアル・インティジャー・ロジック・セット等々と云ったデータタイプ、配列なのか単純変数なのか、或はプロセデュアのパラメータなのかと云った事を知らなければならない。

完全なプログラム単位を構成するライン群のインプット順序をバッチシステムのように制限しない理由の1つは先に述べた対話型システムの思想に依るものであるが、もう1つの理由はイクスターナルリファランスと云うデータベースの参照形態を SIMBOL が導入している事による。

イクスターナルリファランスとは、プロセス相互間のデータベースの参照形態を指す呼称である。アクティビティは SIMBOL で云う所の1つのプログラム単位を構成するものであり、たとえ同一のアクティビティから発生したプロセスどうしであっても、別のプロセスの参照はイクスターナルリファランスによらなければならない。

アクティビティAの記述中にアクティビティBのデータを参照しなければならない場合は、Bを無視してAのプログラム単位を完結させる訳にはいかない。

ミクロな話になってしまうがBのプログラム単位のデータbをAが参照する時データモードが分からないまま命令コードを作成する事は出来

ない。

インテジャーの場合なら FACOM では"LA"、リアルタイプなら"FL"の記号命令が出されるが、その命令語を構成するビット構成は全く異なるからである。

更にはベースレジスターモディファイ、インデクスレジスターモディファイ等でデータ属性の差異に依り命令語のビット構成は異なって来るため、SIMBOL のように、ライン投入時点でオブジェクトコードの作成を行なうシステムでは、不明な情報をユーザに補なってもらわざるを得ない。

このような状況であるから、現在記述中のプログラム単位のキーインを中断し、別のプログラム単位のデータベースの記述をしなければならない。

通常のプログラム構成法を取った場合でも、同時に別のプログラム単位のインプットが不可能である以上、それと等価なインプットが必要となるイクスターナルリファランスを導入する限りは、インプットシーケンスというものを物理的な入力順序の制約から解除しなければならない訳である。

インプットシーケンスを自由にした事により、インプットモードの問題であるとか、ラインナンバーを付す必要性やリコンパイルに対する配慮等々様々な問題が派生したが以下それらを検討しよう。

4.1.2 インプットモードの意義

先のインプットシーケンスの恣意性に引き続いて、インプットモードの便益について述べる必要があるであろう。

インプットモードという考え方は、キーインの順序に囚われない論理シーケンスの確立を保証するためのフレーム設定と云う方向から論ぜられる。

前にも触れた通り SIMBOL にはメイン・アクティビティ・プロセ

デュアと云う３つの独立したプログラム単位があり、これらプログラム単位のキーインの順序は勿論の事、プログラム内のラインの順序も一応任意の順序で構成出来る事を示した。

ＳＩＭＢＯＬでは、制約としての意味ではなく、人間の思考の一区切り、部分最適化の対象となる思考単位というものを保証するためのフレームと云う意味合いに於てインプットモードというものを設定している。

これにより少なくとも各プログラム単位間をどのように構造的に結合させてゆくかと云った問題と、各プログラム個別の構造の最適化と云った問題とを別のレベルで考え得る機会が与えられるであろう。

全ての状況を確認した上でプログラムを構成する事が最善である事は云う迄もないが、複雑な論理関係を常時脳裡に焼きつけておき、その最適解からして現在構成中のプログラムは一断面として如何程の意味付けが成され得るかという形でフィルタリングする事は至難の業であろう。

人間にとっては部分最適化が全体最適化に連がる道の方が遥かに歩み易く混乱も少ないのではなかろうか。

現局面では、これだけの限られた状況認識に注意を集中させているだけで９割がたは十分である。即ちメインプログラムを構成する時にはメインの事だけに注意を集中させなさい。メインで使用しないようなプロセデュアの細部に迄注意を払う必要はありませんと云う事は、ユーザの負担を非常に軽くするであろう。

ユーザは現在キャラクターディスプレイに表示されているものを、自分の脳裡の影像であるという形で把握し、明確に部分を意識してプログラミングに傾注出来る訳である。

先に述べたインプットシーケンスと云うのは、この独立した一思考断面のライン構成に於て自由である事、枠組みの中でのレイアウトの多様性という形でとらえ直す事も出来る。

従ってモデル全体の構成法の多様性というものは、モデルを構成する

独立したプログラム単位の並べ方の多様性、即ちインプットモード切替えの多様性として見る事も出来るであろう。

以上思想的背景を述べて来た訳だが、システムの設計という立場から見ても独立したプログラム単位を導入している事から見て、その様な独立性を保証する枠組みを与える事は自然な発想である。

枠組みはユーザを束縛するためのものでは断じてない、という裏付けがあればその枠組みを取り入れて非常に楽にプロセッサを構成出来る訳である。

何故なら枠組みとしてのインプットモードをシステムに導入する事に依り、プロセッサ側はユーザのプログラミングに対して、投入されたプログラム単位と云うものだけを対象として処理し得るからである。

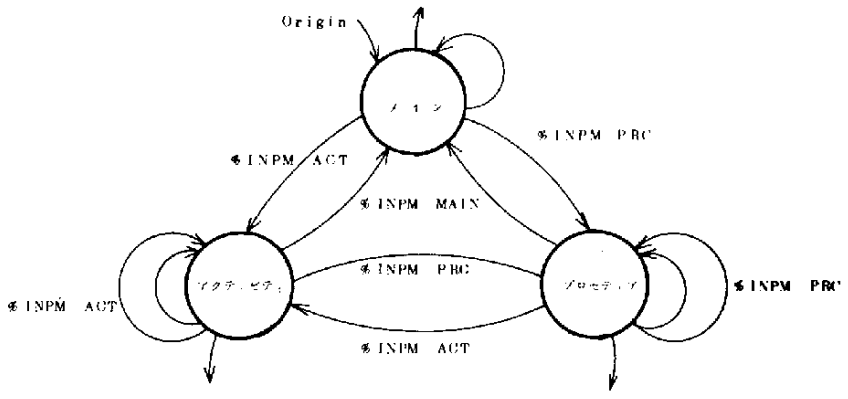


図4.1.1　プログラム単位遷移図

図4.1.1に示したように SIMBOL 起動直後のインプットモードは自動的にメインに設定される。

プログラミングの進行に際し、プログラム単位の遷移を示すインプットモード切替えコマンド"%INPM" のキーインが成されない限り、現在指示されているプログラム単位が処理されているものと見做せばよい。

ユーザからインプットモード変更の指示が成されない限り自分の城壁の中だけで処理を進めていればよく、任意のインプットシーケンスが他の城壁を破壊する心配も全くない。

他のプログラム単位への変更が指示されれば、その状態に移行し、初めてインプットされるプログラムなら新たに城壁を設けるし、既に途中迄作成済みのものであれば、城門を合言葉で開けて貰って中に入ればよい。

このようにインプットモードを設ける事により、投入されたラインがどのプログラム単位に属するものであるかを識別する必要は無くなり、且プログラム単位間の独立性も簡単に保証出来るように成っている。

4.1.3 ラインナンバーの問題

先に述べたインプットモードは、プログラミングに際してのフレームワークを与えるものであり、SIMBOL が提供するモデル化概念のプログラム単位というものを名実ともに規定するものであった。

ここでは、そのプログラム単位を構成するラインのレイアウトを規定するラインナンバーというものを考察する。

SIMBOLではステートメント本体に先立ってラインナンバー（4桁迄の数字）の記述を必要とする。

ラインの完備形は次図4.1.2に示すような構成を取っている。

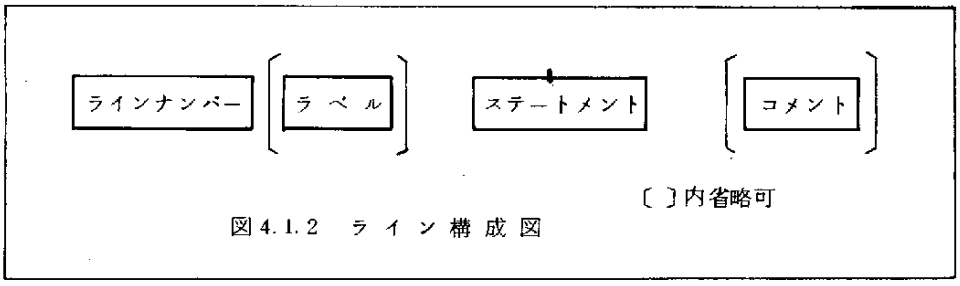


図4.1.2 ライン構成図

前述のインプットモードとの関係でラインナンバーは、プログラム単位内ローカルと云う形態を取っている。

ラインナンバーは、カード形式のアイデンティフィケーションナンバーと云われるものと同様のものであるが、より積極的な意味合を持つものである。

即ち SIMBOL のラインナンバーは、インプットシーケンスの自由を獲得するために払った代価のようなものであり、これなくしては論理シーケンスを構成し得ない重要な要素となるものである。

ラインナンバーをその数値の昇順にソートしたものがプログラムの論理シーケンスを表現する事になる。

数値は4桁迄可能であるから、0から始めて 10^4-1 迄の10000行が1つのプログラム単位の最大ステップ数となり、大概のプログラムはこれで十分である。

ラインナンバーは、ステートメントを代表する識別コードでもあるから、プログラム修正の時にもこの数値を用いて必要なステートメントの呼び出しをしたりするのに用いられる。

ラインナンバーをステートメント本体の頭に毎回付さなければならない事はキーイン作業を更に煩わしいものにする。

プログラムインプットに先立ってユーザがキザミ幅のみを指定すれば後はシステム側が自動的にキザミ幅を加算した数値を表示してくれればキーイン作業も比較的楽になるのではなかろうか。

ステートメントに関しては、キーワードの省略形を許したり、あれかこれかと云う二値関係で表現し得るものの標準的な方を省略形で表現すると云う様にキーイン作業の簡便さを図っている。

そうであれば、ラインナンバーの簡略化も当然考慮されるべきであると考えるのが自然であろう。

事実 SIMBOL に於ても開発段階でその様な検討が成されたのであ

るが、否定的な結果に終った。

と云うのは、インプットシーケンスに制限があるシステムならいざ知らず、ランダムインプットの観を呈するシミュレーションのプログラミング作業に於て、自動的にシステム側からラインナンバーを表示しキーイン要求をしたのでは、物理的にインプットシーケンスを規制されるのと等価になってしまいラインナンバーを付す意味が基本的にくずれてしまうからである。

プログラムメインテナンスとしての意味だけを強調する場合であっても、消極的な意味しか持ち得ないであろう。

即ち作成そのものに付随するアップディティングと云う積極的な意味合いを持つ云わばダイナミックなメインテナンスに対し、事後処理的なスタティックメインテナンスとでも云うべき機能の一担しか担えないであろう。

修正しつつプログラムを完成させて行く事を特徴とする SIMBOL に於ては、やはりラインナンバーと云うものはユーザに付して貰うようにする仕方しか無いようである。

これはキーイン作業を多少面倒臭いものとして印象付けるが、ユーザはモデル作成中に頻繁に修正を繰り返す中で、第一印象で受けるイメージからの即断がまさに即断すぎたと云う感を深くするであろう。

ユーザは何のトラブルもなく困難なモデル作りに専心出来る影の力として、ラインナンバーの威力を改めて認識するであろう。

それも全て、ラインナンバーの導入を、試行錯誤によるモデル作りの困難さを軽減させると云う方向で検討し、具体化しているがためである。

CMUの対話型 SIMULA の試みに於ては、小数点付きのラインナンバーと云うものが用いられている。

これによれば整数部分で独立したプログラム単位を表現する事も可能であり、細部に渡る問題を無視するなら、プログラム単位を区分するイ

ンプットモードと云うものは不要であったかも知れない。

又部分的な実行に際しても、整数部が同一のものとか、小数点第何桁目がいくつのものの実行と云うような指定も簡単に行なえるであろうし、修正に際してもグループとしての扱いをかなり簡便なものにするであろうと考えられる。

SIMBOLでは、その点まで考えてはいないので小数点付ラインナンバーと云うものは今後の対話型システムの開発に際し一考の余地があるであろう。

その場合でもラインナンバーのキーインのし易さ、或は見易さと云った事も勿論必要な事柄ではあるが、その使われ方の可能性の検討が第一義であろう。

```
    DISPLAY PART 40

40.1: ΔACTIVITY JOB.  THIS IS THE CODE BLOCK FOR THE SINGLE SERVER QUEUE.
40.2: NEW INTIME← ↘JOB[ACTIVE,5]; NEW SERVICE← ↘JOB[ACTIVE,5]
40.3: GO TO STEPNEXT(JOB[ACTIVE,1])
40.4: NUMJOBS←NUMJOBS+1;IF NUMJOBS > 100 THEN {TYPE 'END OF RUN';PAUSE}
40.5: NEXTJOB←CREATE(↘JOB[NUMJOBS])
40.6: ACTIVATE(NEXTJOB,TIME+RAND1)
40.61: INTIME←TIME;SERVICE←RAND2
40.615: JTRACE; TYPE 'TIME IS 'ITIME
40.62: IF SERVERBUSY THEN {WAIT(QUEUE);CANCEL(CURRENT)}
40.63: SERVERBUSY←TRUE
40.64: JTRACE; TYPE 'TIME IS 'ITIME
40.7: HOLD(SERVICE)
40.8: IF EMPTY(QUEUE) THEN {SERVERBUSY←FALSE;GO TO STAT}
40.81: POINTER←FIRST(QUEUE)
40.82: REMOVE(POINTER)
40.83: ACTIVATE(POINTER,TIME)
40.9: STAT: TYPE 'TRANSIT TIME FOR JOB WITH ID= 'IACTIVE' IS 'ITIME-INTIME
40.91: CANCEL(CURRENT)

    Δ I WILL NOW START EXECUTING THIS PROGRAM BY TYPING THE COMMAND INITIALIZE
    INITIALIZE
```

図4.1.3　CMU interactive SIMULA

```
0010 ACTIVITY MESSAGE;
0020 INTEGER CURIMP,ENDIMP;
0030 INTEGER NXTIMP,USLINE,LN;
0040 LOOP. LET NXTIMP=RMATRIX(ENDIMP,CURIMP);

0050 LET USLINE=LMATRIX(NXTIMP,CURIMP);
0060 IF LSTATE(USLINE) # 1 THEN BRANCH LAB1;
0070 IF BUFC(NXTIMP) >= BSIZE THEN WAIT -> LQUE(USLINE);
0080 LAB1. LET BUFC(NXTIMP)=BUFC(NXTIMP)+1;
0090 LET LSTATE(USLINE)=1;
0100 DELAY TRANSTM;
0110 LET LSTATE(USLINE)=0;
0120 LET BUFC(CURIMP)=BUFC(CURIMP)-1;
0130 IF LN # 0 THEN BRANCH ACTM;
0140 IF LSTATE(LN) # 0 THEN WAKE <- LQUE(LN) DELAY 0;
0150 ACTM. IF EMPTY(LQUE(USLINE)) THEN BRANCH LAB2;
0160 IF BUFC(NXTIMP) >= BSIZE THEN BRANCH LAB2;
0170 WAKE <- LQUE(USLINE) DELAY 0;
0180 LAB2. LET CURIMP=NXTIMP;
0190 DELAY 1;
0200 LET LN=USLINE;
0210 IF CURIMP /# ENDIMP THEN BRANCH LOOP;
0220 LET BUFC(CURIMP)=BUFC(CURIMP)-1;
0230 END;
```

図4.1.4 SIMBOL-Line Number Sample

4.2 プログラムのアップデイト

この節ではモデル・プログラムタイプ・ライン群等各レベルに於ける変更や、プログラムの編集機能（エディティング）、変更に伴う再編成（リコンパイレーション）等に関する検討を行なう。

4.2.1 ライン単位のアップデイト

ラインの追加・削除・修正は、人間の判断に基づき随時行なわれる場合もあるし、エラー発生時に適宜行なわれる場合もある。

通常のバッチシュミレーションシステムに於ては、ステートメントであるかプログラムタイプであるかにかかわらず全てカードの差し替えにより追加・削除・修正をマニュアルで行なわなければならない

変更後の確認はカードを1枚1枚繰り乍ら人間の目で行なう。

それ以上に或る程度プログラムとしての体裁を整えない限りは、アップディトは起らないと云っても良いであろう（例えば一度モデルプログラムを計算機にかけてからリストをチェックし誤ちを訂正すると云う具合に）。

モデルのランニングが主眼であるから、モデル構成時に於ては、システムから一応切離されて人間が置かれる。

そのような立場に置かれた人間は、一応プログラムとしての体裁を整えたモデルをシステムに投入する仕方でしかシステムと結合し得ない。

構成時にエラーがあったとしてもその場でそれをアップディトする事は余程熟練した注意深い人間であったとしても稀にしか見られないであろう。

プログラムの体裁を取って投入されるので、一括して解析され、エラーは一括して出される。

従って修正も一時期に済ませる事が出来る。

カードの抜き替えや差し替えを行なって再度入力すると、然るべき解析結果が示され、更に変更が必要なら同様の作業を順次続行する事となる。

変更と云う状況は SIMBOL でも同様の手続きが成されなければならないが、プログラムの体裁を整えている必要がない点や、ライン単位にその場で行動が起せる点など、バッチシミュレーションシステムとはかなり異った局面を持っている。

投入単位がプログラムではなく、1ラインであるため モデル作成の過程そのものが追加・削除・修正作業を包含する事になり、アップディ

トがリアルタイムに行なえるようになっている。

エラーに依りリジェクトされた場合は再表示されるので、もう一度全情報をキーインする事なく必要な修正を施すだけで済む。

通常のキーインは追加作業と等価であり、ラインナンバーが異なるものであれば全て追加と云うアップディトの単位と見做される。

今キーインしたラインがエラーも無く、受け付けられた場合であろうと、既に受け付けられているラインであろうと修正を施す場合は、そのラインをディスプレイ上のユーザインプット部に表示させればよい。

%の後に修正を施したいラインのラインナンバーをキーインすれば、そのラインの情報がユーザインプット部に表示される。

必要な修正を施した後 SEND キーを押すと、先にキーインされている修正前のラインの先頭に＊印が付され、置き替えてよいかどうかの確認信号をシステムが要求するので、Y（YES） をキーインする事に依り修正ラインと置換される。

削除の場合も同様に%の後にラインナンバーを付しその後にセミコロン（;）をキーインすればよい。

削除指定が成されたラインの前に＊印が付され、削除してよいか否かの確認が同様に成される。

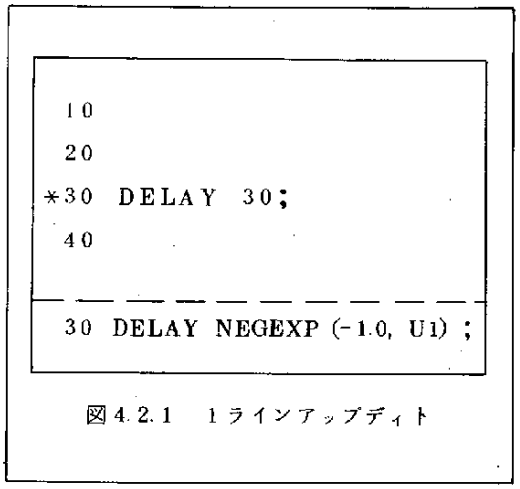


図4.2.1 1ラインアップディト

アップディションはこの様に全て対話型を取り確認し合って行なわれるようになっている。

従って同一のラインナンバーをもつラインをキーインした場合にも自動的な置き替えはせずに、既に登録されている旨告げてキーインを取り上げないように設計されている。

慣れたユーザは若干の煩雑さを覚えるかも知れないが、既にキーインされたものを尊重し、より少ない修正でプログラムを構成して行くためには、情報の消去は慎重な取り扱いをした方がユーザに取っても得策であると云う認識に基づいている訳である。

修正を必要とする場合の再表示機能はプログラム構成上極めて効果的なものである。

無駄な事をせずに出来る限り短時間で修正を済ませ、より積極的な目的に目を向ける時間を増やせるのも対話型システムならではの大きな利点の1つである。

複雑なモデルプログラムになればなる程、ソースプログラムのコンパイルエラーを取り除く作業に時間を費す事になり、論理的なチェックを遅らせる結果、ランタイムに吐き出されるロジックエラー等を解析するエネルギーさえも消耗してしまうと云った状況は、不慣れなシステムの下でシミュレーションを行なう場合にはありがちな事である。

そう云った意味でも、エラーチェックと修正と云うものは密接な関連をもって、システムの中で機能付けられている必要がある。

この種の問題を解決するのは、人間と機械とのインターフェイスを密ならしめる対話型システムの領域に於てであり、従来のバッチシステムの不備の一端もこの辺りに端緒があるように見受けられる。

4.2.2 ライン群のアップデイト

前項では1ライン単位のアップディトに関して述べた訳だが、この場合あくまでもアップディトが中心となるものではなく、モデルの構成中である事が主眼であった。

従って1ラインの修正に関しては必ずコンパイル作業が付随するものであった。

修正ラインのみのコンパイルだけで済んでいる場合は余り問題もないが、全体のリコンパイルを誘発するような場合にはユーザを苛立たせる

事にもなりかねない。

この様な場合、モデル構成中と云うよりは、修正と云うものをより意識したものとしてエディティング機能が用意されている。

エディット機能を用いたプログラムの修正は、プログラム単位内ローカルに行なわれる。

数ラインを一括して修正する事が出来、コンパイルはその場では成されない。

エディティングモードから抜け出した時点で、最後に扱われていたプログラム単位のコンパイルが自動的になされる。

ここでコンパイルエラーが発生した場合は、前節のライン単位の項で述べたような方法で修正を行なえばよい。

エディット機能を用いたライン群の修正は容易であり、形の上ではライン単位の修正と何ら変わる所がない。

一括した修正を1回のコンパイルで済ませる所に大きな意義がある。

ライン群のデリート機能を使用して数ラインを纏めて消去してしまう事も簡単に行なえる。

1ライン単位に確認し乍らキーインするのが煩わしい場合にも、エディット機能を用いて効率良くキーイン出来るし、ラインナンバーの再割りあても RENUM 機能を用いて簡単に行なえる。

4.2.3 プログラム単位の変更

プログラム単位は各々独立した単位であり、アクティビティをメインに変更すると云っても、プログラム単位を宣言するラインを変更すれば済むと云った様な簡単なものではない。

プログラム単位につけられた名前を変更するのは比較的簡単であり、この場合は、プログラム単位の識別名が変わるだけで、プログラム単位そのものの性格が変わる訳ではないからである。

しかし乍らプログラム単位名の変更であっても前述のラインの変更と

同じように扱う事は出来ない

プログラム単位宣言ステートメントの修正は、本来独立であるとするプログラム単位の独立性を損ねる危険性を孕んでいるからである。

従って明らかに修正を意識させる意味合も含めて、前述のエディティング機能の一部としてプログラム単位の修正が出来るようになっている。

特に SIMBOL では、部分的な実行をメインプログラム単位だけに許している関係で、アクティビティ、プロセデュア等が任意にメインプログラム単位に変更出来なければならないと云う事もあって、RENUM 機能により簡単に行なえるようになっている

プログラム単位が変更された場合には、必らずリコンパイルが行なわれる。

この場合には、特定のプログラム単位にしか許されないステートメントと言うものがあり、プログラム単位の変更に先立っていくつかの修正を必要とする場合もある。

この様な修正は、前項で述べたプログラム内の修正機能を用いて行なえば良い。

先に修正が成されない場合は、リコンパイルにより使用不可能なステートメントがすべてチェックされるので、その結果に基づいて必要な修正を行なえば良い。

一般のバッチシミュレーションシステムでは、この様なプログラム単位の変更に対する要求はまれであり、単にプログラム単位宣言ステートメント一行の差し替えだけで済む場合が多い。

その場合であっても、プログラム単位個有のステートメントと言うものがあり、結局その様なステートメントすべてを修正する作業は必要になる。

4.3 エラーチェック

SIMBOLでは、1ラインのインプットに際し解析を行なう所謂インクリメンタルコンパイレーション方式を取っているため、バッチの言語と異なり、ユーザがラインをキーインするのと時を同じくしてエラーチェックが成される。

修正しつつプログラムを構成し得る事が非常に有利である事は既に何度も述べて来た所でありここでは省略する。

エラー修正を対話形式で行なうSIMBOLでは、コンパイル時のエラーチェックを2つのフェーズで考えなければならない。

1つはライン単位にインクリメンタルコンパイレーションを行なう時のチェックでありあくまでも独立したラインである事を前提とするフェイズもう1つは、ライン相互に依存し合う情報に関するものであり、ラン直前にチェックされるものである。

バッチシステムでは全プログラムを一括してチェックの対象とするため、ステートメント固有の記述ミスのみならず、ステートメント相互に関連し合うもののチェックもコンパイル時に行なわれるが、SIMBOLのインクリメンタルコンパイレーションでは、ライン相互に関連をもつ記述のチェックは全プログラムが出揃った時点でなければ行なえないのでラン直前に廻さざるを得ない。

その場合でもラン直前のチェックから漏れる部分もあり、ランタイムチェックに持ち込まれているものもある。

ラン直前のチェックの主要なものは次のようなものである。

BRANCH, TRANS等のステートメントと飛び先側ステートメント間の対応関係やDO, LOOPの入れ子や対応関係、IFステートメントの中に書かれた前述のステートメントに関するもの等である。

ランタイムに持ち込まれているものはプロセデュア、或はファンクションの呼び出しであり、呼び出しステートメントのキーイン時に相手側の存

在を確認し得ないし、又ダイナミックリンキングを行なっている事からも、ランタイムのチェックでなければ意味を成さないからでもある。

SIMBOLでは、1ラインの解析途上エラーを発見した場合は即座にコンパイル処理を中止してエラー情報を出しユーザの修正を求めるようになっている。

1ラインの最後迄エラーを蓄積し乍ら解析する事は余り意味がないからである。

キーインしたラインが正しくない場合は、実効的にキーインは受け付けられず、ディスプレィ画面上のユーザインプット部にキーインされた形のまま残りエラー情報が示される。

ユーザはカーソルを移動して然るべき修正を施した後再度 SEND キーを押せばよい。

当初キーインした情報そのものが残っているので、エラー発見時に処理を中止し、ユーザの修正を求めても支障はないし、無意味な派生エラーの表示が成されないだけユーザも納得し易いとも云えるであろう。

バッチシステムではこの様な方法は取り得ないため、見つけたエラーは蓄積して行き後に一括して吐出す事になる。

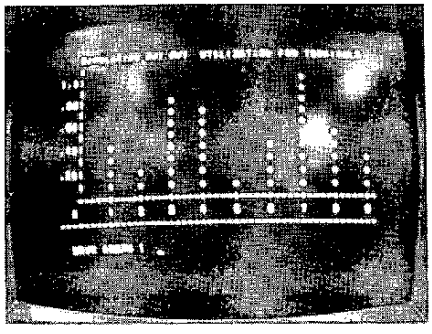


図4.3.1　SIMBOL のエラーチェック

修正に至る迄の時間が長い事から、バッチシステムでは執拗にチェックを厳しくし、細部にまでくまなく解析の眼を光らせる努力を払っているものが多い。

例えば SIMSCRIPT－Ⅱ に於けるソースプログラムのエラーチェックは非常に徹底しており、表4.3.1に示すように、1字1句も漏らさずにチェックしている。

SIMSCRIPT－Ⅱ の場合はステートメントの記述がフリーフォーマットで行なえるようになっているので、途中でエラーを発見してもそこで処理をやめてしまい次のステートメントに移る事が出来ない。

そのために、 ステートメントが要求する本来のシンタックスが完結する迄エラーチェックを続行するため、余分なチェックを行なわなければならない。

従って応々にして派生エラーが生じ易く、ユーザはまどわされる事もしばしばある。

又表4.3.1からも明らかな様に何のエラーか識別がつきにくいし、無駄な情報が多く、有効情報が埋もれがちであると云った側面もあり、適切なエラー情報を出力するような工夫が望まれる。

又ここまで多くの情報を出力するのであれば、エラータイプ番号と云ったものでは無く適切なエラー理由を文章の形で出すべきであろう。

表4.3.1　SIMSCR=PT-Ⅰ　エラーサンプル

```
ROUTINE  FOR  CHECK  GIVEN  CRTIMP  AND  BLGCRT
         LET  SW=0
         LET  I=CIRC.ENT1(CRTIMP)            IF  CRCT.STATE(I)=1
              LET  SW=1  GO TO NXTSRV
   ELSE  LET  I=CIRC.ENT2(CRTIMP)            IF  CRCT.STATE(I)=1
   ELSE  LET  I=CIRC.ENT3(CRTIMP)            IF  I=0  GO TO NEXT
              ELSE IF CRCT.STATE(I)=1  LET SW=1  GO TO NXTSRV
   ELSE  LET  I=CIRC.ENT4(CRTIMP)            IF  I=0  GO TO NEXT
              ELSE IF CRCT.STATE(I)=1  LET SW=1  GO TO NXTSRV
'NXTSRV'  ADD 1 TO CURR.ENTRY(CRTIMP)
          SCHEDULE  ENDSERV(I)  AT  TIME.V+TRANS(I)          RETURN
**** ERROR OF TYPE 1 INVOLVING 'SCHEDULE' AT STATEMENT 11.  **********************
**** ERROR OF TYPE 1 INVOLVING 'ENDSERV' AT STATEMENT 11.  ***********************
**** ERROR OF TYPE 1 INVOLVING '(' AT STATEMENT 11.  *****************************
**** ERROR OF TYPE 1 INVOLVING 'I' AT STATEMENT 11.  *****************************
**** ERROR OF TYPE 1 INVOLVING ')' AT STATEMENT 11.  *****************************
**** ERROR OF TYPE 1 INVOLVING 'AT' AT STATEMENT 11.  ****************************
**** ERROR OF TYPE 1 INVOLVING 'TIME.V' AT STATEMENT 11.  ************************
**** ERROR OF TYPE 1 INVOLVING '+' AT STATEMENT 11.  *****************************
**** ERROR OF TYPE 1 INVOLVING 'TRANS' AT STATEMENT 11.  *************************
**** ERROR OF TYPE 1 INVOLVING '(' AT STATEMENT 11.  *****************************
**** ERROR OF TYPE 1 INVOLVING 'I' AT STATEMENT 11.  *****************************
**** ERROR OF TYPE 1 INVOLVING ')' AT STATEMENT 11.  *****************************
'NEXT'    IF  CRCT.QUE(BLGCRT)  IS  EMPTY                    RETURN
          ELSE  ADD 1  TO  CURR.ENTRY(CRTIMP)
          SCHEDULE ENDSERV(BLGCRT)  AT  TIME.V+TRANS(BLGCRT)
**** ERROR OF TYPE 1 INVOLVING 'SCHEDULE' AT STATEMENT 14.  **********************
**** ERROR OF TYPE 1 INVOLVING 'ENDSERV' AT STATEMENT 14.  ***********************
**** ERROR OF TYPE 1 INVOLVING '(' AT STATEMENT 14.  *****************************
**** ERROR OF TYPE 1 INVOLVING 'BLGCRT' AT STATEMENT 14.  ************************
**** ERROR OF TYPE 1 INVOLVING 'I' AT STATEMENT 14.  *****************************
**** ERROR OF TYPE 1 INVOLVING 'AT' AT STATEMENT 14.  ****************************
**** ERROR OF TYPE 1 INVOLVING 'TIME.V' AT STATEMENT 14.  ************************
**** ERROR OF TYPE 1 INVOLVING '+' AT STATEMENT 14.  *****************************
**** ERROR OF TYPE 1 INVOLVING 'TRANS' AT STATEMENT 14.  *************************
**** ERROR OF TYPE 1 INVOLVING '(' AT STATEMENT 14.  *****************************
**** ERROR OF TYPE 1 INVOLVING 'BLGCRT' AT STATEMENT 14.  ************************
**** ERROR OF TYPE 1 INVOLVING 'I' AT STATEMENT 14.  *****************************
RETURN    END
**** ERROR OF TYPE 9 AT STATEMENT 9.  ********************************************
```

4.4 プログラムの構成法

SIMBOLの対話機能は、計算機上でのモデルの構成を比較的早い時期から行なう事を可能にした。

ユーザは、混沌としたモデル化作業を計算機の助けをかりて、より明確にする中で、徐々にモデルを拡大してゆく事が可能となった。

明確に規定された細部から積木細工のようにモデルを積み上げて行く事

も可能であるし、全体の関連をグローバルに把えつつ、細部の構造を明らかにする構成法も可能であるし、荒っぽく云えば、思いつくままに構成する中で時にはより細に入り徴に渡った構造規定を、或る時はフレームワークの設定をと云うような具合に体型化を進める事も可能である。

モジュール構造としてモデルを拡大する事も可能であるし、実験体型のの中にモデルプログラムを構成する便宜も提供されている。

従来のバッチシミュレーションシステムでサポートされていなかったこれらの特徴を順次検討しよう。

4.4.1 プログラム単位の構成

SIMBOLのプログラム構造は、GPSS や SIMSCRIPT と異なり、擬似並行処理を基調とする SIMULA と同じような構造を取っている。

GPSS や SIMSCRIPT のように独立したプログラム構造よりも複雑な様相を呈している反面、ユーザは自由に構造化出来るようになっている。

SIMBOLにはメイン・アクティビティ・プロセデュアの3つのプログラム単位があり、それぞれ異なる構造と機能を具備している。

アクティビティはイベント相互を関連付けて記述する機能を持っている。

SIMSCRIPT 等では本来独立なものとしてイベントを別々に記述しなければならない訳だが、アクティビティの記述と云うものは、或る結晶軸の周囲に関連イベントを吸着するような機能を有している。

従ってイベント中心の考え方からするとアクティビティは、いくつかのイベントが時間的に或は状態条件によって結合されたもののようにも見える。

事実プロセスの類を規定するものがアクティビティであり、プロセス自身が一連の事象の追跡過程そのものの表現である事を考えれば至極当

然の話しでもある。

この様な考え方を更に一般化・抽象化したものとしてSIMULA67のクラスと云う概念が既に言語の中に取り入れられ、プログラムの構成と云うものがより広義に解釈されようとする傾向も見受けられる。

階層的なプログラム構成、例えばヒューリスティックなシミュレーションに於て多段の判断過程を階層的に構造化しようとする場合等、各レベル間の規定を極めて図式的な形でイメージ設定しようとするものである。

プログラム単位を集合を構成する1つの要素と見做し、要素相互間の関係として極めて抽象的な形でプログラム構成を把えようとする訳である。

これにより、当初ユーザは混乱の淵に押しやられる事にもなるが、この様な概念がより普遍的なものであり、単に待行列解析のためにシミュレーションがあるかの如き狭隘な考え方から救われるようになるであろう。

この事はプログラムの組み方の特殊性を強調する訳ではなく、極めて概念的な形で把握される事物の関係をプログラムとして表現する可能性を示すものであり、通常のプログラム構成に於けるモジュール化の考え方を包含するものである。

集合要素の多次元の関連をプログラム単位として実現する確とした方法論は未だ存在しないが、プログラム単位が或る種の機能単位として認識され、且対象システムがそのような機能の複雑なかかわり合いとして規定される事を認めるなら、アナロジーから得られるものは決して無駄にはならないはずである。

この種の推論が途方もない夢事で無い事はモデル化過程が抽象化の過程である事を思い浮べていただければ十分であろう。

モデル構成として上述して来た事項は今後の課題としてシミュレーシ

ョンシステムに課せられた問題であり、現時点でこれ以上詳細に語れるものでもない。

その方向性は決してプログラムの構成をいたずらに複雑化しようとするものでは断じてあり得ないと云う事をつけ加えておこう。

4.4.2 各種構成法

SIMBOLであろうと他のシミュレーションシステムであろうとプログラム構成の原理は基本的には変わりがない。

但し、対話型である事から、構成のし易さと云う点ではかなり様相を異にする。

ここで取り上げたアプローチも、対話型であるが故により効果的なものとして意味付けられるものである。

2章の SIMBOL の紹介で述べたように、 SIMBOL ではプログラミングの仕方がかなり自由になっており、プログラミングして来た過程は極めて整理された形でシステムが保存している事を示した。

必要なプログラムを個別に構成してゆき、それらの関係を取りながら核心に迫って行く様な方法（便宜上ボトムアップアプローチと称する）、一般のプログラム構成に見られるサブルーチンから構成してゆき、コーリングシーケンスを整えながら積み上げて行く方法に従ってモデルを構成して行く事は、見通しが得にくい反面構成は比較的簡単である。

末端プログラムを有機的に結合させて、上位レベルのプログラム単位を構成して行く訳で、構成中のプログラムに対する展望が取れないと混乱してしまうであろう。

SIMBOLでは構成中の未完全なプログラムを保存する事は勿論の事、構成中のプログラム単位を一瞥出来るよう配慮されている。

SIMBOLシステムが用意しているプロセデュアの一覧表や、ユーザが作成したプログラムに対する一覧表は TEACH コマンドに依り簡便に得られる。

一覧表の中の特定のプログラムの進行状況が見たければ、LOAD コマンドに依りそのプログラム単位を画面上に表示させればよい。

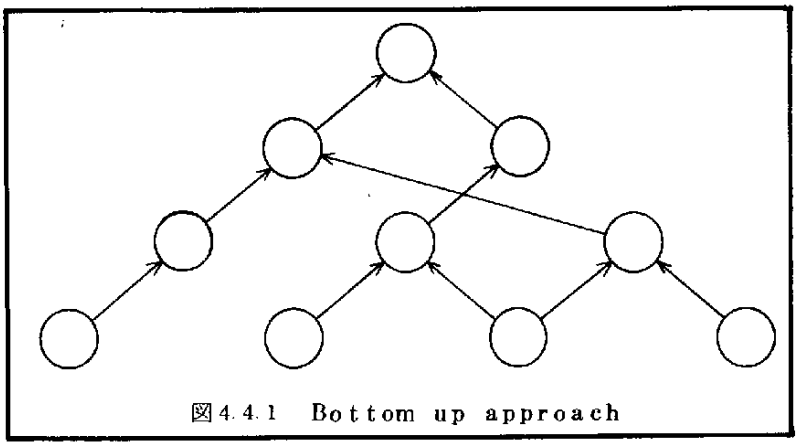


図4.4.1　Bottom up approach

図4.4.1　にボトムアップアプローチに依るプログラム構成の概念図を示してある。

図からも明らかな様に、この種の構成法は一般的なものであり、且確実に積み上げていけるので漏れや重複と云った事も少ない。

これとは全く違った方向からプログラムを構成する方法もあり、トップダウンアプローチと称している。

この方法は、まず核となるべきプログラム単位を作成し、以降それに従うプログラム単位と云うものを順次作成して行くもので、章・節・項目と云うように細部への展開と云う形で進められるものである。

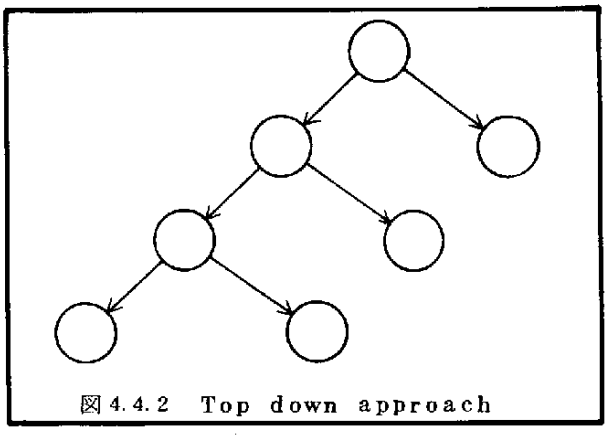


図4.4.2　Top down approach

この種の構成法は状況に応じて使い分けられるものであり、どちらか一方を使うべきだと云ったものでは決してない。

実際にプログラムを構成してゆく段階では、全体の見通しを持つ事も必要であるし、細部の構造規定も必要な訳で、結局の所、作成済みのプログラムに肉付けをして行く事になる訳で、作成済みのプログラム単位が保持出来且、見たい時に一瞥出来展望を明晰にし得る事がプログラム構成を容易にする。

構成法と云ったものが、実際の方法論として有効に活かせるための便宜が提供される事に依り、 SIMBOL に於けるプログラムの構成は比較的容易に行なわれ得る。

プログラム自体を機能と云う形で捉えるにしろ、一連の事物の流れとして捉える、或は状態変化と云うもので捉えるにしろ、プログラム単位の構造規定とプログラム間の構造規定と云うものはある訳で、これらの作業を見通し良く、効率的に行なえるような配慮が積極的に成されるべきであろう。

SIMBOL自身は従来のバッチシステムよりはいくぶんかプログラム構成と云う事を考えてはいる訳だが、ファイルを設けた事の副次効果と云う側面が強くもっと積極的にプログラム構成を問題にする余地があるようである。

4.4.2 プログラムの結合

SIMBOLのプログラムの結合はランタイムに行なわれるので、プログラム構成上極めて自由度が高い。

これを SIMBOL ではダイナミックリンキングと称しており、ランタイムに必要なプログラム単位が主記憶装置上に無い場合に、ユーザズファイルを探して所望プログラムを主記憶にロードし、結合させる。

ファイル上にも見あたらない場合、リンクフォールトを起こし、ユーザのアクションを待つ。

その時に必要なプログラム単位名をキーインしてやればよい。

プログラム構成法でも若干触れた様に、モデルはプログラム単位に依り構造規定されるものであるが、スタティックに全てが規定されてしまうものではない。

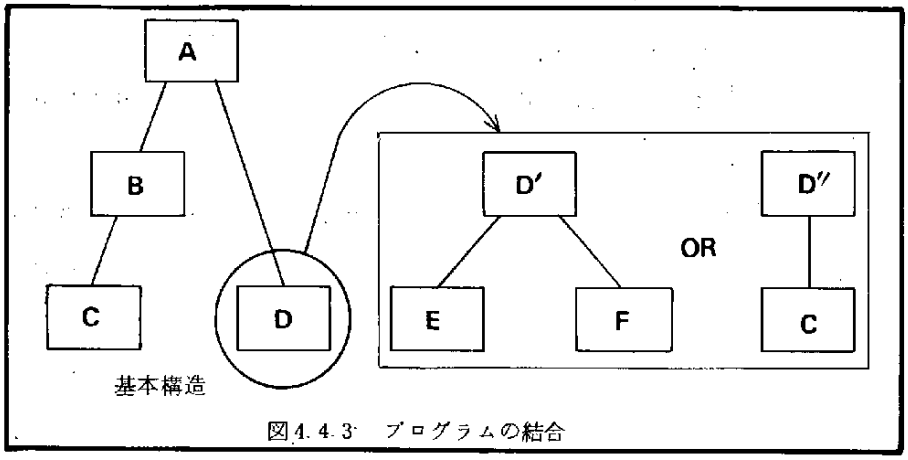


図4.4.3　プログラムの結合

図4.4.3に示すように、プログラム構成法に依る構造規定はあくまでも基本構造を規定するものである。

従来のバッチシミュレーションシステムでは、ランタイムにユーザのアクションをまってプログラムを結合して行く事が無かったために、異なるプログラム構造を得るにはランを別にする以外に方法が無かった。

従って基本構造と云うものがそのままランタイムの構造を表現している訳だが、SIMBOLの場合には事情が異なり、リンクフォールト機能を用いる事に依り別の構造規定が簡単に行なえるようになっている。

図4.4.3に示すDと云うプログラム単位を、あらかじめリンクフォールトを起こす様に存在しないプログラム名で表現しておく。

ランタイムにはプログラムのリンクを取ろうとしても対象プログラムが存在しない訳でリンクフォールトを起こす。

この時ユーザはUSEコマンドを用いてD′或はD″と云うものを指定する事に依り全く異なるプログラム構造を実現出来る。

D′と云うプログラムとリンクされた後Eと云うプログラム単位が又

リンクフォールトを起こすと云う様にしておき、基本構造を縦横に展開して行く事が可能である。

このリンクフォールト機能を用いたプログラム構造の規定の仕方は、いくつかの代替アルゴリズムを簡便に比較する事を可能にする。

D′とD″は例えば在庫管理手法に関する相異なるアルゴリズムで構成されたプログラムであるような場合ユーザは積極的にリンクフォールト機能を用いて、基本構造に手を加える事なく両アルゴリズムの比較を行なう事が出来る。

更にリンクフォールトを用いる事に依り、未完全なモデルであってもシミュレーションランの対象とする事が出来、チェックし乍らモデルを構成してゆく事が可能となっている。

これは従来のバッチシミュレーションには見られなかった大きな特徴でありシミュレーション実験を充実させる上で極めて効果的な手法である。

第5章　モデルの記述

第5章　モデルの記述

5.1　ワールドビュウ

シミュレーション言語に於ては、対象システムをモデル化する際の抽象化過程を容易にするために、対象システムをどう捉えればモデル化し易いかといった考え方、概念としてワールドビュウを提供しているものが多い。どういう方向から分析を進めれば、システムをモデル化し易いかといった物の見方、世界観とでも云うべきものである。

GPSSはトランザクションオリエンティッドなワールドビュウを規定しているし、SIMSCRIPTはイベントオリエンティッドなワールドビュウを提供しているというような事が云われる。何々オリエンティッドという事が、対象システムの物の見方、分析の仕方を方向づける概念であることはいうまでもない。シミュレーション言語で云われるワールドビュウは、概念であると同時に、技法としての性格も持っている。概念としてのワールドビュウだけではモデル化可能性を検証するにとどまるが、更に技法としての意味づけに依り、モデル化可能性が、モデル作成につながっている。

計算機に依るシミュレーションの場合、実際にモデルを作成する段階で、対象シミュレーション言語の提供するワールドビュウが大きな役割を果たす。対象システムを捉える方向の正当性が検証されれば、その捉え方に基づいて、モデル要素を構成する事になるが、その構成の仕方、便宜という事がモデルの作り易さと密接な関連を持つわけである。汎用のシミュレーション言語というものは、対象システムの捉え方と同時に、対象システムをその捉え方に基づいて具体的なモデルとするための道具を提供している。

従って汎用シミュレーション言語を用いたモデル化とシミュレーションという状況下では、ワールドビュウに関する概念としての普遍性といった事と、技法としての使い易さといったような事が問題にされる。例えばGPSSはトランザクションというテンポラリーな"物"が、様々な条件に

依りシステムの構成要素間を移行し、その移行の集大成がシステムの特性を表現するものであるとしてモデル化が進められる。システムの構成要素をモデル化するために、装置であるとか、スイッチ、待合室、等々のモデル化材料が用意されている。ストレイジ、ファシリティ、キュー、ロジックスイッチ、等々といったものがそれである。システムの構成要素をどういう形で使用していくのか、どう移行していくかという方向で論理を組み上げることに依りモデル化が成される。即ちシステムを事物の機能的な流れとしてモデル化しようとする訳である。この時、機能的な流れとして対象を捉えることがどの程度一般性があるのか、換言するなら、そのような捉え方がどこ迄通用するのかという問題と、そのような把握の仕方に無理が無いか、モデル化材料は有用か、それで十分かといったような問題が生じる。そこでGPSSのワールドビュウは具体性があり、待行列型のシステムに強いが即物的すぎて抽象的なシステムの複雑な関係をモデル化するには不向きであるとか、或はSIMSCRIPTの場合には、概念が抽象的すぎて素人向きではないといったような事が云われたりもする。

現在までの所、離散系シミュレーション言語に於けるワールドビュウに関してはGPSSのトランザクションを基調とするもの、SIMSCRIPTに於けるイベントを基調とするもの、SIMULA、SIMBOLのプロセスを基調とするものの3種類に大別されるであろう。どの言語の提供するワールドビュウがより普遍的であり且つ有用かということになると一概に論じ得ない。概念であると同時に技法でもあることに依り、人間の資質にも依存した問題提起になってしまう訳である。人間の慣れと、好み、考え方の相違が、ワールドビュウを一面からだけでは論じ得なくしている。しかもワールドビュウが論じられる時には、背後に必ずといってよい程、それが具体化されたシミュレーション言語の柔軟性や、操作性といったものがありモデル化材料の不備なのか、或は捉え方の限界というものなのかという事が不明確にされる。

ユーザの立場からすれば、問題になるのは、どちらの捉え方が今対象としているシステムのモデル化に際し便利かという捉え方のテクニックに意味があるのであろう。その場合であっても、どのワールドビュウが優れているかを決めることは非常に困難な事である。なぜならば結局総体として見たシミュレーション言語の中で、ワールドビュウが如何程の意味を持ってモデル化材料として表現されているのかを測る尺度が一意に定まらないということに起因するのであろう。

SIMBOLがプロセスオリエンティッドなワールドビュウを基調に開発されたことが、SIMBOLの性格を大局的に規制していることは事実で、それがユーザにとってどの程度の便宜を与え得るのかということが、シミュレーション言語としてのSIMBOLの評価ということになるであろう。評価対象としてのワールドビュウが論じ難い性格のものであることは前述の通りである。従ってここでは、各々の視点の相違というものを中心にワールドビュウを検討するにとどめる。

GPSSはトランザクションの流れを中心にモデルを構成することを特徴とする言語であり、現在のところ最もポピュラー且つ有用なシミュレーション言語であると云われている。システムを事物の機能的な流れとして直接記述できるところにこの言語の特徴がある。

SIMSCRIPTはイベントの相互作用としてシステムを記述するものであり、汎用言語 FORTRAN 等にも劣らぬ機能を有し、システム記述用言語としての側面をかなり強く持った言語である。システムの状態変化をイベントの生起として捉えるところに特徴がある。

SIMBOL、SIMULAは共にプロセスの相互作用としてシステムを記述するものであり、概念が一般化されたことに比例して、具体的なアナロジーが得にくく非常に強力な言語であるといわれている反面、まだまだ馴染が薄いようである。論理的に整合する一連のイベントの広範囲に渡る生起の様子をグループ化したものとしてプロセスは捉えられる。

事物の機能的な流れとしてモデル化を進めるGPSSは、かなり主観的な立場から対象を把握しようとする。即ち、対象システムに於ける流れの要素を見い出し得たならば、その中れの中で次はどの要素と関わるか、その次はどの要素と関わるかというように、流れてゆくもの自身をユーザの分析の眼とし、これをそのままトランザクションの流れに置き替えることでモデル化が成される。

システムを物相互の因果関係としてモデル化を進めるSIMSCRIPTでは、システムの状態変化を客観的に捉えようとする。或る事象の生起がどういうシステム変化と結びつき、他の事象の生起と如何に関わりを持つか、その関わりの仕方を因果則という形で客観化することでモデル化が成される。

GPSSでは事象ということは、トランザクションの行動軌跡の断面として与えられ、因果則をそこに見い出すことができるが、事象そのものはあくまでも従としてあるにすぎない。ユーザが行動過程を記述する事に依り、対象システムをモデル化する際に、GPSS流の概念に従う限りは客観的な因果関係というものは、個々のトランザクションの行動軌跡の総体として意識されるものと見る事もできよう。主体はあくまでも個々の活動そのものにある。

逆にSIMSCRIPT流の概念で捉えようとするなら、事象と事象との相互関係、因果関係といったことが主であり、そのように表現された因果則の適用として活動、行動過程というものがある。個々の活動そのものは、客観的に記述されている因果則の多様な適用の産物として"従"としてあるとも云える。動的過程をモデル化するに際して、GPSS流に個々の行動過程を中心に捉えてもよいし、SIMSCRIPT流に行動パターンを個々に適用するという形で捉えてもかまわない。いずれにしろ事象というものと、事象と事象とを結びつける活動とが表現される事に変わりはない。

活動を矢印→で、事象を○印で図式的に表現すると、3種のワールドビ

ュウは図5.1.1に示すようなモデルとして表現されるであろう。

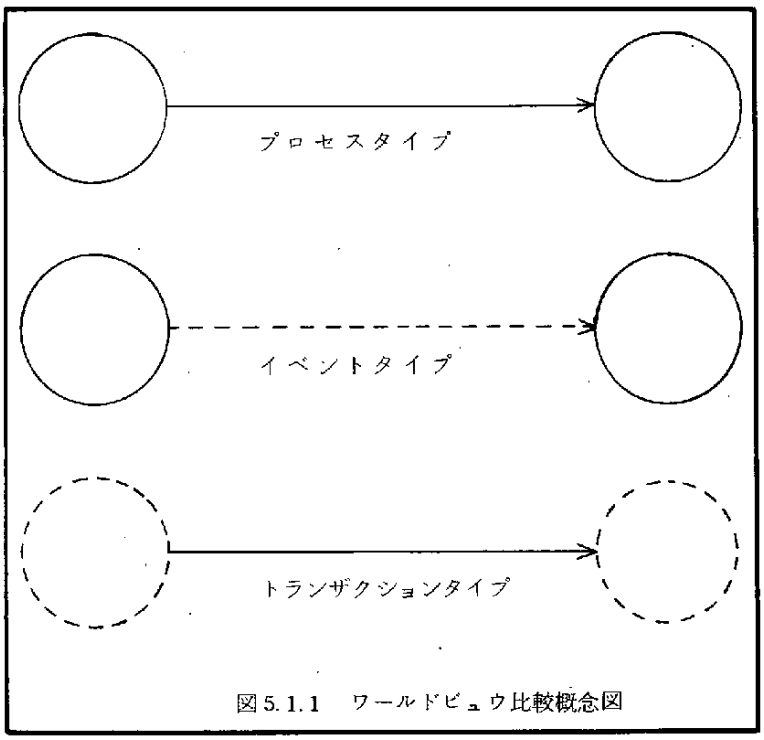


図5.1.1　ワールドビュウ比較概念図

SIMBOL、SIMULAのプロセスオリエンティッドな考え方は、GPSSの活動中心的な見方と、SIMSCRIPTの事象中心的な見方を総合するものである。この場合、プロセスはどちらかの性格を主とし従とするといった形で考えられるものではなく両者を同等に扱い得る事、主観性、客観性をユーザの好みに依り使い分けられる、そうした意味での概念の一般化として考えられるものである。

5.2　特殊データの取り扱い

シミュレーションシステムに於ては、対象システムをモデル化し易いようにという配慮から、特殊なデータ構造を提供しているものが多い。勿論通常の汎用言語といわれるものが扱い得るデータと同様のものも使用でき

るわけで、工夫しだいでユーザが望むデータ構造を実現することも可能である。この節では、モデル化のために用意されている、特殊データ構造、セット、キューの扱いやプログラム単位間の参照を可能にするためのイクスターナルリファランスに関する検討を行なう。

5.2.1 セット／キューの扱い

実際のシミュレーションに於ては、現在でもなお待行列管理のモデルを扱う事が極めて多い。

GPSS がよく利用されているのは、親しみ易さもさることながら、現実にシミュレーションとして取り上げられる対象が、待行列といったものを少なからず含んでおり、GPSSがそのように使用頻度の高い現象を標準的なモデルとして規格化し、簡便に利用できる便宜を提供していることに由来するものであろう。GPSSでは待行列の表現をキューとしてモデル化し、統計収集機能をも含ませている。待行列の長さや、待時間等に関する統計量は自動的に収集される。

SIMSCRIPTやSIMULAは、この種の便宜が無いため、概念の取りつきにくさに加えて、手軽に使用できない面があり、簡単なモデル等の表現であってもユーザが敬遠してしまうのも無理からぬことである。SIMBOLでは、待行列を標準的なものとして規格化し、SIMULA等が抱えていた不備を補なっている。

SIMBOLにはキュー・セットといわれる特殊データ構造を持ったエンティティが用意されている。キューはセットの特殊表現であり、集合体を管理するためのものである。キューやセットに関する基本的な処理は、特定集団へのメンバーの参画と脱退であり、後はその仕方の多様性が特定集団の多様性を物語る事になる。GPSSでいわれるキューは最も単純な集団構成法であり、年功序列型の会社組織を連想させるものである。ENTERブロックにより特定の組織のメンバーとなる。組織に加入したばかりのメンバーは最下位の平社員として扱われる。DEPARTブ

ロックにより組織から脱退し、昇進して行く訳だが、この時最初に抜擢されるのは、平社員が構成する団体の中でも年令がいった人、即ち最初に組織に加入したメンバーである。これをFIFO（First In First Out）のキューと呼んでいるが、この種の話はシミュレーションの場に於ては最も多く登上するものである。

前例で述べるなら、抜擢された平社員に関して、どの位の期間平社員であったのか、加入していた組織は常時何人位で構成されているのか、最盛期には一体どの位の平社員がいたのか、その組織から送り出された者はどの位の数に達するのか、要領よく出世の波に乗平社員を素通りした者は何人位いるのかといった経歴データが調べられる。その結果、課長のポストが窮屈すぎるから少し間口を広げて、平社員の数を減らそうとかいう話にもなっていく。或はどうも平社員の数が少なすぎ、全体のバランス上好ましくないから主任クラスへの昇格を適当に押えるかという話になったりする。この例のようにFIFOのキューとしてモデル化できる現象は、我々の身の廻りにいくらでも見い出すことができる。

SIMBOLではキューへの加入をINSERTで、脱退をREMOVEでモデル化する。GPSSでは、無条件にFIFOの取り扱いが成されるが（正確には、FIFO以外の扱いをキューに対して許していない）、SIMBOLでは位置指定といったものがあり、キューとして捉えられる現象がより広範囲に渡っている。勿論FIFOの現象の使用頻度から、位置指定を省略したときには、自動的にFIFOとして取り扱うよう考慮されている。

図5.2.1に示すようにFIFOの扱いはトコロ天の押し出しのようなものであり、これをキューとしてモデル化した時は最大待行列長、平均待行例長、キューに入った総数、零時間で通過した数と総数に占める割合、平均待時間等が自動的に集積される。（統計収集に関しては次節で検討する）

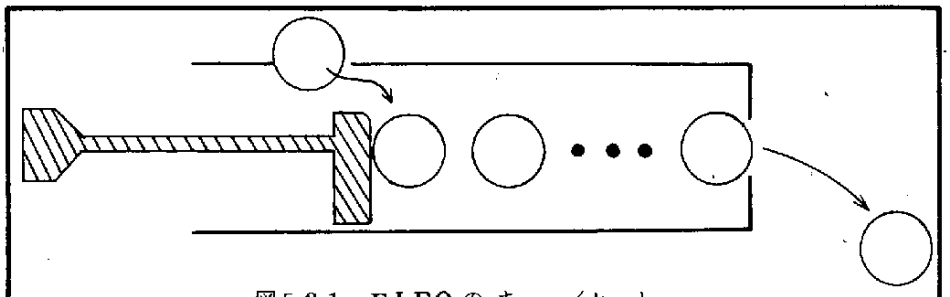


図5.2.1　FIFOのキュー／セット

SIMSCRIPT、SIMULA、或はSIMBOLでもセットいう概念を用いてGPSS、SIMBOL等が提供するキューを表現することは容易である。その場合には統計収集をユーザが記述しなければならない。SIMSCRIPTの場合1.5、Ⅱともにメンバーの登録をFILEステートメントで、脱退をREMOVEで行なう。S－1.5の場合には、セットをFIFOの性格をもつものとして定義して用いなければならない。

REMOVEに関してはREMOVE　FIRSTといった位置指定が必要である。S－Ⅱの場合は、あらかじめ集団の性格を定義しなくてもよいかわりに、メンバーとして加入する場合は、FILE　FIRST、脱退する場合はREMOVE　FIRSTとしてFIFOを構成しなければならない。SIMULAの場合PRCDとREMOVEステートメントによりこれを行なう。PRCD(Z, Y)に依りZが属すセットにメンバーYを登録するが、その位置はZの直後である。FIFOを構成するには、Zを所属すべき集団とすると次のようになる。

```
PRCD(LAST(Z), Y)
REMOVE(FIRST(Z))
```

SIMBOLの場合はセットもキューと同様の扱いが成されるので省略する。（但しキューと異なり統計収集機能は無い）

次にプロセッサを構成する場合の数式の解析にリバースポーリッシュ

法として用いられるプッシュダウンスタックや、エレベータの乗降に見られる現象等のモデル化のために特殊データ構造がどのように使われるかを検討してみよう。

この現象は、エレベータの例でいえば先に乗った人は奥の方に押しやられてしまい、後から乗った人が先に降りるようなＬＩＦＯ(Last In Fast Out)の現象としてモデル化される。この場合でもＬＩＦＯ構成の待行列モデルであり、現象としてはかなり多く見られるものである。従って統計収集に於ても自動的に収集される便宜があれば都合が良い。

所がＧＰＳＳでは先にも述べた通り、キューとして自動的に統計収集を行なってくれるものは、ポピュラーなＦＩＦＯ構成のみである。
ＳＩＭＢＯＬでは、ＬＩＦＯのキューも簡単に表現できるようになっている。

```
INSERT
REMOVE  LAST
 或 は
INSERT  FIRST
REMOVE
```

ＳＩＭＳＣＲＩＰＴ 1.5 の場合はディフィニションカードに依りＬＩＦＯの指示を与える。

- ＳＩＭＳＣＲＩＰＴ－Ⅱの場合

```
FILE  FIRST      / FILE  LAST
REMOVE  FIRST   /  REMOVE  LAST
```

- ＳＩＭＵＬＡ－６５の場合

```
PRCD(FIRST(Z), Y)     / PRCD(LAST(Z), Y)
REMOVE(FIRST(Z))     /  REMOVE(LAST(Z))
```

• GPSS－Vの場合には、キューとしてLIFO構成を実現できなかったが、セットに相当するものとしてユーザーズチェーンが用意されており、キューと同様統計処理を自動的に行なうようになっている。FIFOもこのチェーンを用いて表現できる。

LINK LIFO	LINK FIFO
UNLINK	UNLINK

• SIMULA－67の場合はFIFOであれ、LIFOであれプロセデュアでこれを行なうが、クラスという概念に基づいているため、セット操作の記述に関してもSIMULA－65とはかなり様子を異にしている。メンバーの脱退はシステムが用意したリンケージクラスに属するOUTプロセデュアにより行なう。メンバーの登録は、INTO、或はFOLLOWプロセデュアで行なう。

FIFO

Y.FOLLOW(S.LAST)	Y.INTO(S)
S. FIRST.OUT	S.LAST.OUT

LIFO

Y.FOLLOW(S.LAST)	Y.INTO(S)
S.LAST.OUT	S.FIRST.OUT

LIFOの扱いは図5.2.2に示すような自動的にボールが現われるゴルフ練習場のボール送り装置のようなものであり、このような現象もFIFO同様馴染み深いものである。それだけに、標準的なモデルとして規格化しておく方がはるかに便利であり、システム自身を身近なものとして感じさせることもできるであろう。

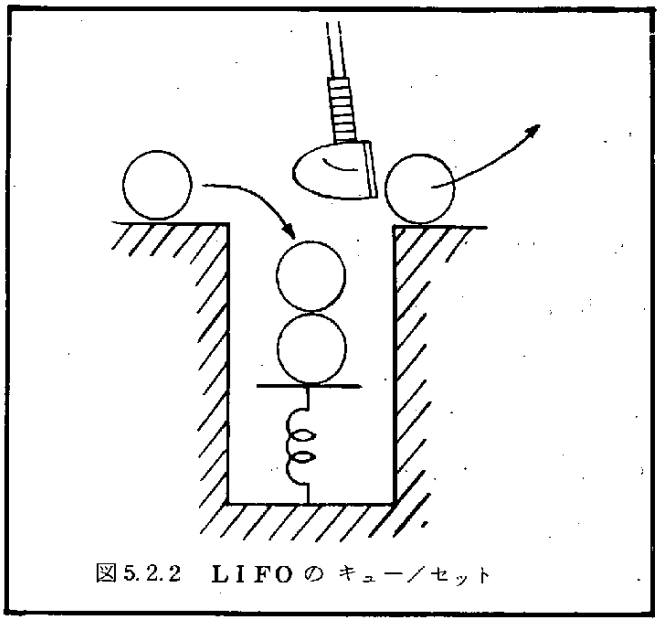


図5.2.2　LIFOのキュー／セット

更に複雑なデータ操作も可能であり、GPSSやSIMSCRIPT 1.5でいわれるランク付け、その拡張としてのメンバー名指しでその前後に特定の条件付けで構成するSIMSCRIPT－Ⅱ、SIMULA，SIMBOLの位置指定、BEFORE，AFTERといったものがある。これによりメンバーとなった順番とは全く違った条件、例えば集団構成員の中から年若い順に取り出したり、女性だけを高令者から脱退させたり、親族同志を年長者順に順序づけるとかいった事も簡単に行なえるようになっている。

このように多様な性格を集団に与え且つその集団の特性を自動的に収積してゆくSIMBOLのキューの概念は、GPSS 同様の手軽さと、SIMSCRIPT－Ⅱ、或はSIMULAが持っている表現能力との両者を兼ね備えているため、非常に強力且つ有効なモデル化の材料といえるであろう。

5.2.2　イクスターナルリファランス

シミュレーションに於ては、或るプログラム単位に固有なデータを別のプログラム単位から参照したいという要求がしばしば起る。全く異な

るプログラム単位のデータであるために直接参照することができない訳で、各システムとも特別な工夫がなされている。

SIMBOLに於ては、参照しようとするプログラム単位とは別のプロセスのローカルデータを参照するために、イクスターナルリファランスという機能が用意されている。これはSIMULAのコレクションメカニズムと等価なものであり、互に異なるプログラム単位間でのコミュニケーションを可能にする。SIMBOLのイクスターナルリファランスは、単に2つのプロセス間のコミュニケーションを取るだけではなく、逐次結合関係を求めて連鎖を辿り、目的とするプロセスのローカルデータを参照することも簡単に行なえる。

イクスターナルリファランスの一般型は、

＜エレメント＞：＜クラス名＞．＜ローカルデータ＞

で表わされる。

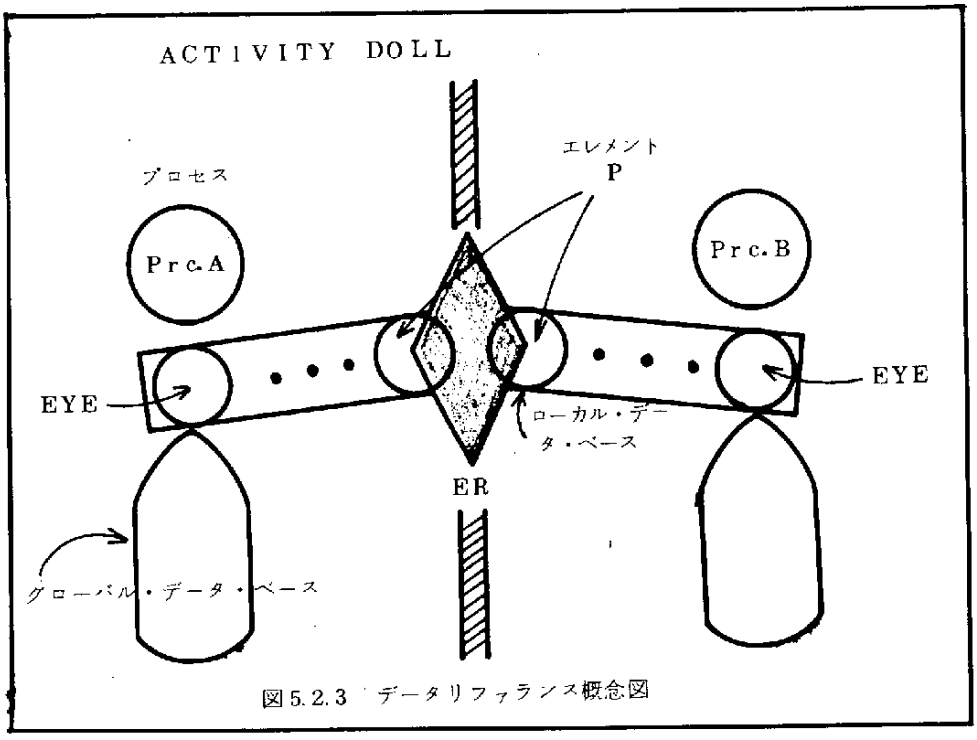


図5.2.3 データリファランス概念図

図5.2.3はイクスターナルリファランスを説明するための概略図である。PrcA，Bはそれぞれアクティビティ DOLL から発生したプロセスであり、EYE という名で呼ばれるデータを各々持っている。一般の汎用言語に於ては、名前によって直接データが参照されるのが普通である。

プロセスが自分自身にローカルなデータを参照する場合、或はグローバルデータベースといわれる共用データを参照する場合は単に名前だけで直接そのデータを参照できる。ところがプロセス相互にデータを参照する場合には、同じアクティビティから発生したプロセスであれば同じ名前のローカルデータを持つため、単に名前だけでは識別が不可能である。そこで何らかの形でプロセスを識別する必要がある。更に、別のアクティビティ例えばHUMANに於ても同じ名前例えばEYEを持ち得るので、アクティビティ或はブロックといったクラスの識別も必要になってくる。

こうして、どのクラスから発生したどのプロセスの何という名前のデータかという形でイクスターナルリファランスという参照形態が可能となる。図5.2.3でプロセスAからプロセスBのデータ EYE を参照する場合には、プロセスを識別するエレメント変数Pと、クラスを識別するためのアクティビティ名 DOLL と、目的とするデータ EYE とを次のように組合わせて行なえばよい。

P：DOLL．EYE

この記述のしかたはアクティビティ本体でのものであり、アクティビティ DOLL からプロセスA，Bが発生され、プロセスAが実行された時に、プロセスを識別するためのエレメント変数Pが、イクスターナルリファランスの参照に先立ってプロセスBを指し示していれば、そのプロセスのデータ EYE 例えば青色というものがイクスターナルリファランスで参照されることになる。

別のアクティビティHUMANから生み出されたプロセスデータ EYE をプロセスBから参照することも可能であり、アクティビティ DOLL で定義されたエレメントQがプロセスの実行時に、プロセスBを指しており、プロセスBのエレメントPがHUMANから生み出されたプロセスを指している場合には

Q：DOLL.P：HUMAN.EYE

とすればよく何重にもエレメントを介してイクスターナルリファランスによりデータを探索して行くことも簡単に実現できる訳である。ここで示したQはエレメントと呼ばれる属性をもつものであり、エレメント関数呼出しの形を取ることもできる。

同一アクティビティから発生したプロセスが同一のグループに属している場合等に於て

SUC〔Q〕：DOLL.P：HUMAN.EYE

という形で、プロセスAから、グループの中のプロセスBの後続プロセス例えばCというものを参照し更にそこから引続き別のプロセスを参照するような場合にエレメント関数呼出しは頻繁に使われる。

SIMULAに於ては、2つのコネクションバーブによってシステム内の種々のエレメント間の相互作用が表わされる。

コネクション・バーブ
INSPECT
EXTRACT

INSPECTは、SIMBOLのイクスターナルリファランスと同様のものであり EXTRACTは、もしEXTRACT の次に書かれたエレメントがセットのメンバーならセットから取りはずす操作も行なう。

SIMBOLの＜P：DOLL.EYE＞ に相当する参照は、次のようになる。

```
INSPECT P WHEN DOLL DO EYE ･･･
          WHEN  Ai  DO  Si
          OTHERWISE S ;
```

Aiはアクティビティクラスであり、Pというエレメントが Ai というアクティビティに属する時はDO以下のステートメントSiを実行する、そうでなければ次のWHEN以下に書かれたアクティビティとの対応関係を調べてゆき、どのアクティビティにも属さない場合は OTHERWISE 句以下のステートメントSを実行する形態をとる。

```
Inspect  P  When  Doll  Do  Eye = Red
                When  Human Do  Eye = Black ;
```

コネクションにより結合関係が取れれば、DO以下のステートメントの中で直接参照したいデータの名前を記述し処理をすることができる。SIMSCRIPTの場合は、ダイナミックに作成、消去されるブロックは、直接スケジューリングメカニズムと関連して扱われるものではなく、テンポラリーエンティティとして特別な形で与えられる。イベントの中でテンポラリーエンティティをSIMBOLのローカルリファランスと同じように参照する場合は名前だけで直接行なえる。SIMSCRIPT-1.5の場合のように同じ名前を別のテンポラリーエンティティで定義することを許さなければ、名前だけで識別できるわけである。テンポラリーエンティティの作成された世代を問題にする場合には、名前だけでなくテンポラリーエンティティの名前も必要になる。

テンポラリーエンティティの名前に対応して1つのグローバル変数が取られ、最新のテンポラリーエンティティを指し示しているので特に世代を問題にしなければエンティティの名前と関連づけた参照のしかたは不要である場合もある。世代を問題にする場合は、テンポラリーエンティティの作成に際し名前をつけておかなければならない。

CREATE DOLL CALLED P(I)

名前をあえてつけなければ、テンポラリーエンティティと同じ名前のものが作成されるので、イベントノーティスにその名前をパラメータで受け渡して世代に関する参照を行なう。

EYE(P(I))
EYE(Q(I))

GPSSの場合は、トランザクションのパラメータ相互の参照がイクスターナルリファランスに相当する。この場合には参照したいトランザクションのパラメータをあらかじめグローバルな性格を持つセイブバリューに入れておくか、グループに所属させておかなければならない。トランザクションの番号を問題にする場合には、単にセイブバリューにパラメータの値をコピーしておくだけではすまない。参照し合うトランザクション同志でうまく同期を取る事も必要になったりする。

SIMBOLのように存在するプロセスのデータ全てを参照できるようにするには、GPSSでは現存するトランザクションのパラメータ全てをセイブバリューに対応させるような事も必要になるであろう。このためにGROUPエンティティというものが導入されて、JOIN, REMOVEによりグループを構成し、ALTERによるパラメータの変更が行なえるようにはなっている。いずれにしろイクスターナルリファランスに近いことが行なえるようにはなっているが、簡便さからいえば、問題があろう。

5.3 フローコントロール

シミュレーションに於ては、物相互の関係が時間的な流れの中で意味付けられるように記述される。時間が刻々と更新されてゆくに従って、物相互が互に影響を及ぼし合ってモデルの状態を変えてゆく。従って、スケジ

ュールの仕方や、同期の取り方、クロックの進め方、同時並行現象の扱い方、等々が必ずシミュレーションシステムに於ては問題になる。これは特にSIMBOLに限った話しではなく、シミュレーションシステム一般が抱えている問題であり、結局このあたりのシミュレーションにとって本質的である問題をどう扱うか或は扱い得るかという事でそのシステムの柔軟性といったようなものが規定されてしまうようである。

特にGPSSのようにシステムが自動的に取り扱う部分が多いシステムでは、いろいろな工夫が成されなければ実用的でなくなってしまうであろう。スケジュールのしかた1つをとってみても、単純に時間軸上での計画通りに全てが移行するだけで済む訳ではなく、割り込みや、計画変更、緊急事態といった現象を簡便にモデル化できなければ、シミュレーションシステムの実用性というものは空論にすぎなくなってしまう。クロックの進め方も、もともと連続量であるところを、離散的に扱って階段上に進ませて行くわけだし、同時並行現象というものも、擬似パラレルという形で取り扱う現状であるわけで、違った角度からこのような問題が捉え直される事も必要であろう。ハイブリッドシミュレーションシステムというものが或は1つの解決を与えるものと成り得るかも知れない。

5.3.1 スケジュールの仕方

SIMBOLでは、SIMULA同様スケジュールの機能が豊富に用意されている。1つは時間値を対象とするスケジュールでありもう1つは、事物の関係に依るスケジュールである。時間値を対象とするスケジュールの仕方は一般的であり、どのシミュレーションシステムに於ても見受けられるものである。

GPSS-Vでは、トランザクションの発生に関するスケジュールをGENERATEブロックで行なう。このブロックは、平均発生時間間隔、分散、最初の発生時刻等を主要パラメータとし、シミュレーションランの前に最初に実行すべきブロックがGENERATEブロックであるとして擬

似トランザクションが発生される。

擬似トランザクションはスケジュールテーブルの1つFEC(Future Event Chain)に連鎖され、BDT(Block Departure Time)と称する生起時刻は、GENERATEブロックのパラメータから計算した値を持っている。シミュレーションが進行し、シミュレーションクロックがBDTと等しくなると、このトランザクションがFECからCEC(Current Event Chain)に移される。

FECへの擬似トランザクションの登録は、GENERATEブロックを実行する前に行なわれるのでGENERATEブロックは1つ先のものを絶えず計画するような働きを持つ事になる。従ってユーザは、発生過程をGENERATEブロックとして記述するだけで後は自動的に先の計画が成されていくことになる。

他方SIMULA, SIMSCRIPT, SIMBOLでは、スケジュール用のステートメントを単に記述するだけではなく、その命令が何度も実行されるように記述しなければ発生過程を表現できない。

SIMSCRIPT-1.5では計画すべきイベントノーティスを先にCREATE命令を用いて作成し、CAUSE命令に依りスケジュールテーブルに登録する。

発生時刻をCAUSEステートメントに記述する形をとる。

SIMBOLではSCHEDULEステートメントに依りイベントノーティスの発生と登録を行なう。発生時刻の指定には、何時何分になったらという絶対時刻による指定と、今から何時間後にといった相対時刻によるものとの2つがあり再スケジュールを容易にしている。

```
0.0924,1/0.1787,2/0.2386,3/0.4174,4/0.5566,5/0.6307,6/0.8461,7/0.903,8
0.9731,9/1.0,10
 12    FUNCTION    RN1,C6
.0,0/.2511,50/.411,75/.6349,125/.7783,175/1.0,300
****      INITIAL SET  ****
        INITIAL     X60,200
        INITIAL     X61,10
****      MATRIX DEFINE  ****

1               GENERATE    X61,FN12                *** MESSAGE GENERATE ***
2               ASSIGN      2,FN$STION              *** ASSIGN DISPATCH STATION ***
3               SAVEVALUE   P2+,1,XH
4               ASSIGN      3,FN*2                  *** ASSIGN TERMINAL STATION ***
5               SAVEVALUE   P3+,1,XB
6               ASSIGN      4,P2                    *** ASSIGN RELAY STATION ***
7       NET01 GATE   SNF    P4,NET06                *** OVERFLOW ***
        ********** IMP PROCESS ********************************************
8               ENTER       P4                      *** ENTER BUFFER ***
9       NET10 QUEUE         P4
10              SEIZE       P4                      *** SEIZE IMP ***
11              DEPART      P4                      *** DEPART BUFFER ***
```

図5.3.1　GPSSのスケジュール例

```
      ENDOGENOUS  EVENT  MSGCL
C     /*  MESSAGE  GENERATION  */
        CREATE  MSG
              CALL  MSSGE(*IORG,*IDST,*ARVT)
                  LET  SORCE(MSG)=IORG
                  LET  DSTIN(MSG)=IDST
                  LET  GENT(MSG)=TIME
              CALL  SELCT(IORG,IDST,*ICRTN,*IMPN)
C     /*  OCCUPANCY LINE NO */
                  LET  OCCPY(MSG)=ICRTN
                  LET  CURT(MSG)=IMPN
        CAUSE  MSGCL  AT  TIME + ARVT
              CALL       SRVST(IORG,MSG)
        RETURN
      END
```

図5.3.2　SIMSCRIPTのスケジュール例

図5.3.1，5.3.2にGPSSとSIMSCRIPTの場合のスケジュールの例を示してある。

図5.3.1のGPSSの場合、GENERATEブロックのX61，FN12

が生起時刻を規定するものである。X61はセイブバリューの61番目であり、INITIALステートメントに依り10に初期設定されている。この10という値が平均生起時間間隔を示すものであり、FN12は6点を結ぶ連続関数であり経験分布を表現している。この関数値が分散に使用され、平均時間間隔X61に分散FN12が乗ぜられ、小数点以下で切捨てられた値が実際の発生時刻ということになる。

図5.3.2はSIMSCRIPT－1.5の場合であり、MSGCLというイベントが自分自身を計画するようになっている。このイベントはメッセージの発生を司るものであり、システムクロックTIME＋ARVTという形で次の発生時刻を決定している。ARVTは到着時間間隔を表わすものであり、ARVTの時間間隔でこのイベントが実行されることに依り発生過程が表現される。

```
166
167          COMMENT ************** GENERATING NEW MESSAGE ********
168
169          ORGD:  X:=UNIFORM(0.0,1.0,U1);
170          K:=1;
   B14
171          FOR I:=1 STEP 1 UNTIL 10 DO BEGIN
172               K:=I;
   E14
173               IF ORGDIST(I) GTR X THEN GO TO ENDD; END;
174          ENDD:  X:=UNIFORM(0.0,1.0,U2);
175          J:=1;
   B15
176          FOR I:=1 STEP 1 UNTIL 10 DO BEGIN
177               J:=I;
   E15
178               IF MDIST(I,K) GTR X THEN GO TO NEWMSG; END;
179          NEWMSG:  MESC:=MESC+1;
180          HISTO(H1,B1,K,1);
181          IF BUFC(K) GEQ BSIZE THEN REJC:=REJC+1
   B16
182          ELSE BEGIN
   E16
183          ACTIVATE NEW ACCEPT(K,J,0,1,TIME) DELAY 0.0;  END;
184          HOLD(NEGEXP(1.0/ARRAIVT,U3));
185          IF TIME LSS ETIME THEN GO TO ORGD;
```

図5.3.3　SIMULAのスケジュール例

SIMBOLやSIMULAの場合は何時刻後にといった時間指定が許されるので、図5.3.3に示すようにACTIVATEに依り発生を計画し、HOLDにより次の発生時間迄待機し又ACTIVATEするというような発生過程の表現も可能であるし、SIMSCRIPT流に発生時刻を指定したACTIVATEも可能である。

事物に関係付けたスケジュールの仕方というものを次に検討しよう。

SIMBOLやSIMULA、SIMSCRIPT-Ⅱでは、プロセス或はイベントノーティスに関連付けたスケジュールの仕方というものがある。

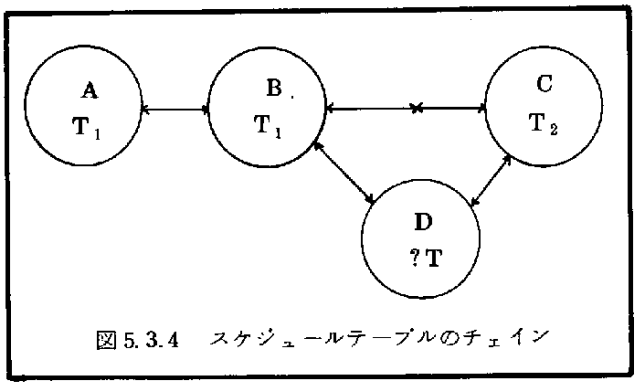


図5.3.4　スケジュールテーブルのチェイン

図5.3.4に示すようにスケジュールテーブル上に時刻T_1に生起すべきプロセス（或はイベントノーティス）A・BとT_2時刻に生起すべきものCとが計画されているとする。生起時刻Tは通常値の昇順にチェインされており、先に述べた時間指定によるスケジュールが成されると生起時刻の小さい方からサーチしてゆき登録すべき時刻より後のものが見つかった時点、或はスケジュールテーブルの登録アイテムの最後にゆきあたった時点で登録されるのが普通の方法である。

この指定に際し、同一生起時刻の先頭にスケジュールするか、最後にするか或はスケジュール即実行という形にするかといった指定ができるシステムと、GPSSのようにあらかじめプライオリティというものを付随させる事により同一時刻に於ける登録位置を決めてしまうシステムとが

ある。いずれにしろ時間属性での登録の自由度は大差ない。

ところが図5.3.4に示すようにBというプロセスの後にDというプロセスを計画したい、或はCというプロセスがいつ生起するか知らないが、その前にDというプロセスを行なう必要があるといった状況がしばしば発生する。このような場合、SIMBOLでは"AFTER B"であるとか"BEFORE C"といったような指定によりプロセスを計画できる。

DをBの後に計画したい場合には、SIMBOLのステートメントでは次のように記述される。

SCHEDULE NEW D AFTER B;

これによりDの時間属性はBの値T_1に等しくされる。

或るプロセスが生起してはじめてDというプロセスをBの後に計画する必要性が生じるといった状況をGPSSで表現するためにはGATEブロックを用いて条件待ちをするとか、ユーザーズチェイン上でスケジュールテーブルと同じような構造を作り出して制御するといったような工夫が必要になる。

GENERATEブロックで発生されたトランザクションに関して、どのGENERATEブロックであるかを識別しなければならない場合には、別々のユーザーズチェインに登録して所在を明らかにするような工夫が成されなければならないわけである。

SIMBOLでは個々のプロセスに名前がつけられるので、スケジュールテーブル上に登録されている場合でも個別に呼び出すことができ、前述の時間属性に依らないスケジュールが簡単に行なえるわけである。これによって単に時間属性に依らないスケジュールのみならず、特定のプロセスに関連した様々なスケジュールが可能になっている。

例えば図5.3.4に示すAというプロセスをCの後にリスケジュールするとか、まだ生起していないBを△T時刻後にリスケジュールし直すとか、或は特定な条件を満足するプロセスの計画を取りやめるとかいった

操作が非常に簡単に行なえるようになっている。

SIMBOLのRESCHEDULEやCANCELといったステートメントは、そのために用意されているものであり、プロセス名を指定することにより簡単に用いることができるものである。

5.4 統 計 収 集

シミュレーションに於ける統計収集の問題は、モデル全体の行動の総括として重要な意味を持っている。時々刻々の経時変化に関連してシステム特性が変化してゆく時系列過程を追跡し、その状態変化を総合して、平均とか分散、最大といった統計量として表現することは大変な作業である。

システムを構成する要素に関する使用率であるとか、待行列長であるとかいった特性値に関する量を把握したいという要求は、かなり一般的であるにもかかわらず、簡便な手段を用意しているシステムは余り多くはない。

SIMBOLに於ては、一般的であると思われる統計量に関する収集を自動的に行なう機能を用意している。時間に関する統計と分布に関する統計量は簡便に収集できる。GPSSが提供しているキューと同じ機能が用意されており、待行列に関する統計量は自動的に収集される。自動統計収集を行なうためにキューのヘッドに対して然るべきエリアが割当てられている。

QUEUE HEAD

+0	NOM ： 現在登録されているメンバー数	
+1	FMEP：先頭メンバーのポインター	LMEP：最終メンバーのポインター
+2	NEM ： 登録されたメンバーの総数	
+3	MAXNOM：最大待ち数	
+4	NONW ： 待ち無しで出たメンバー数	
+5	MAXWT ： 最大待ち時間	
+6	TWT ： 待ち時間の総和	

図5.4.1 キュー ヘッド

QUE宣言ステートメントにより使用するキューを定義しておく。

Queue Headに与えられた統計変数用のエントリーは、通常のグループ処理ステートメントにより新たにメンバーがキューに挿入されたり、取り出されるたびに自動的に更新される。

グループのメンバーとなるものはプロセスであるが、SIMBOLでは直接プロセスをメンバーとせずエレメントを介して間接的に行なっている。このエレメントをグループメンバーエレメント（GME）と称しており、キュー、セット両者がありそれぞれQME，SMEに依り識別している。

QME

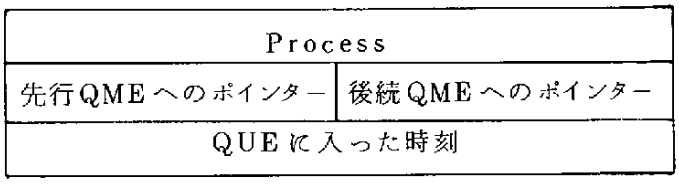


図5.4.2　メンバーの構造

GME（Group Member Element）には、プロセスを指し示すポインターと、グループに於ける先行・後続メンバーを示す2つのポインターがある。グループの先頭メンバーの先行メンバーは存在しないし、グループの最終メンバーの後続メンバーも存在しないので、共にメンバー無し＜NONE＞という形で表現している。

QMEには、グループへ入った時刻というエリアが割りあてられている。

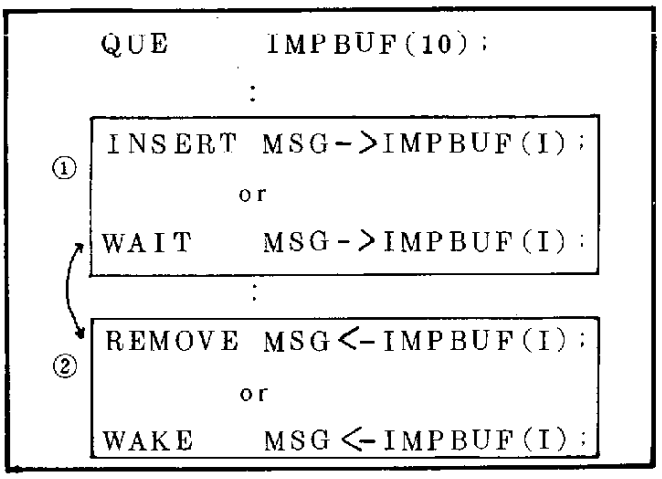


図5.4.3 統計収集（ SIMBOL ）

図 5. 4. 3 に示す①に依りメンバーはグループに挿入され、Q－Head IMPBUF(I)の現在数＋1，総数＋1，最終メンバーへのポインターの更新と、QME であるMSG のグループ加入時刻への現在システム時刻の代入等が自動的に成される。

②に依りグループからメンバーが取り除かれる時に、現在数と最大待数との比較を行ない最大待数の方が少なければ現在数と入れ替え、現在数－1、現時刻から QME の加入時刻を差引き、待時間の総和に加え、待時間が零であれば待ち無しで出たメンバー数を＋1し、最大待時間との比較を行ない必要なら入れ替えを行なう。

シミュレーション終了時に PRINT コマンドにより統計諸量を出力する。これによりユーザは GPSS の待行列統計を得るのと同じ手軽さでモデルを操作できる。この種の手段に欠ける SIMSCRIPT、SIMULA に於ては、ユーザは、自分で統計収集用のステートメントや数式処理ステートメントを使用して行なわなければならない。これが如何に大変な作業であるかは表 5. 4. 1、表 5. 4. 2 に示す SIMSCRIPT－1.5 の場合の統計収集例からも明らかであろう。

表に於て四角い枠で囲った部分が統計処理のために費したものである。

イベントルーチンの6割近い比率を統計処理のためにさいており、且つこれだけでは済まず、出力に先立ち表5.4.2に示すように利用率を改めて計算したり、次のシミュレーションランのために各種統計変数をクリアしたりせねばならず非常に面到である。と同時に間違いを起す確率も高く、出力結果を見ても、モデル自体の論理が誤まっているのか、或は統計処理を誤まったのかという問題に絶えず悩まされる事になり、ユーザにとって非常に煩わしいものでもある。

表 5.4.1　SIMSCRIPTに於ける統計収集－1

```
      SUBROUTINE  SRVST(IOG,MSG)
                  LET  IJ=OCCPY(MSG)
      LET X = ENTRY(IOG)
      LET  AMQUE = MQUE(IJ)
                  LET  ENTRY(IOG)=ENTRY(IOG) + 1
                  LET  QINT(MSG)=TIME
                  LET  MQUE(IJ)=MQUE(IJ) + 1
                  LET  TOTAL(IOG) = TOTAL(IOG) + 1
                  LET  ENTNO(IJ) = ENTNO(IJ) + 1
            IF  BUF(IJ) GR MQUE(IJ),GO TO L0
                  LET  LOST(IJ) = LOST(IJ) + 1
                  DESTROY  MSG
                  LET  MQUE(IJ) = MQUE(IJ)-1
                  GO TO L1
L0                FILE  MSG  IN  QUE(IJ)
      LET  KK  = CURT ( MSG )
      IF  ENTRY ( KK ) GE BUFS ( KK ) , GO TO L3
                  IF MQUE(IJ) LS 2,GO TO L1
                  GO TO L3
L1                      LET  ENTRY(CURT(MSG))=ENTRY(CURT(MSG)) + 1
                  CREATE  ENDSV
                  LET  NUMBR(ENDSV)=IJ
                  CAUSE  ENDSV  AT  TIME + TRANS(IJ)
      LET  A11 = 0
      LET  QQQ = QUEIN(IJ)
      LET  EEE = ENTM(IOG)
      ACC A11, AMQUE INTO UTIL(IJ) , AVGQ(IJ) SINCE QUEIN(IJ)
     1,QQQ
      ACC X , X INTO AVAIL(IOG) , AVGQL(IOG) SINCE ENTM(IOG)
     1,EEE
L3    RETURN
      END

      ENDOGENOUS  EVENT  ENDSV
         LET  NUM=NUMBR(ENDSV)
              DESTROY  ENDSV
         REMOVE FIRST  MSG  FROM  QUE(NUM)
              LET  IND1=BLIMP(NUM)
      LET  X = ENTRY(IND1)
      LET  Y = QUEIN(NUM) - QINT(MSG)
      IF Y GR MAXWT(NUM), LET MAXWT(NUM) = Y
      LET  AMQUE = MQUE(NUM)
      LET  A11 = 1.
      LET  QQQ = QUEIN(NUM)
      LET  EEE = ENTM(IND1)
      ACC A11, AMQUE INTO UTIL(NUM) , AVGQ(NUM) SINCE QUEIN(NUM)
     1,QQQ
      ACC X , X INTO AVAIL(IND1) ,  AVGQL(IND1) SINCE ENTM(IND1)
     1,EEE
      IF MAXQL(IND1) LS ENTRY(IND1) , LET  MAXQL(IND1) = ENTRY(IND1)
              LET  IND2=ENTRY(BLIMP(NUM))
                   LET  ENTRY(BLIMP(NUM))=IND2 - 1
              LET  IND3=CURT(MSG)
              LET  IND4=OCCPY(MSG)
              LET  Y=MQUE(IND4)
         IF MAXQQ(IND4) LS MQUE(IND4),LET MAXQQ(IND4)=MQUE(IND4)
              LET  MQUE(NUM)=MQUE(NUM) - 1
      LET  ENTRY( IND2) = ENTRY( IND2 ) -1
              IF  CURT(MSG) EQ DSTIN(MSG),,GO TO L1
C     /*  NEXT  SERVICE  START  */
              LET  M=DSTIN(MSG)
                   CALL  SELOT(IND3,M,*ICRTN,*IMPN)
                   LET  OCCPY(MSG)=ICRTN
                   LET CURT(MSG)=IMPN
      CALL  SRVST(IND3,MSG)
              GO TO L2
L1    DESTROY  MSG
L2    CALL  CHECK ( IND3, NUM )
      RETURN
      END
```

自動的に統計収集を行なう機能が用意されていない場合は、統計収集を行なうための基本的なステートメントが用意されている訳だが、簡単なモデルの場合でさえ、必ず統計収集ステートメントの記述が必要で、その量は又決して少なくないのでユーザの負担は非常に大きいものである。

SIMBOL或はGPSSのように自動統計収集機能が用意されれば、十分な表現力と相俟って更にポピュラー且つ使い易いシステムとなり、SIMSCRIPT或はSIMULAに対して抱くイメージも、従来のような実用性に欠けるといったものではなくなるであろう。

表5.4.2　SIMSCRIPT－1.5 統計収集－2

```
      SUBROUTINE  PRINT
C
C     /*  STATISTICAL  QUANTITIES  PRINT  OUT  */
C
      DO TO L3, FOR I=(1)(NNTWK)
      LET  UTIL(I) = UTIL(I) / 0.3
      LET  AVGCQ(I) = AVGCQ(I) / 0.3
   L3 LOOP
      DO TO L4, FOR I=(1)(NIMP)
      LET  ABUFS = BUFS(I)
      LET  AVAIL ( I ) = AVAIL ( I ) / 0.3  * ABUFS
      LET  AVGQL(I) = AVGQL(I) / 0.3
   L4 LOOP
         CALL  STATC
      RETURN
      END

      SUBROUTINE  RESET
C
C     /*  STATISTICAL  VALUE  RESET  */
C
 L7       DO TO L8,FOR I=(1)(10)
               LET  TOTAL(I)=0
      LET  ENTM(I) = 0
               LET  ENTRY(I)=0
               LET  MAXQL(I)=0
               LET  AVAIL(I)=0.0
               LET  AVGQL(I)=0.0
 L8   LOOP
          DO TO L9,FOR I=(1)(29)
               LET  ENTNO(I)=0
               LET  MAXCQ(I)=0
      LET  UTIL(I) = 0
               LET  LOST(I) =0
               LET  QUEIN(I)=0.0
               LET  AVGCQ(I)=0.0
               LET  MAXWT(I)=0.0
 L9   LOOP
         RETURN
      END
```

多くのモデルでは、キューに関する統計や、データの平均・分散・標準偏差といった基本的な統計量のほかに、データ値の完全な分布が必要とされる場合がある。

先のキューに関する統計収集同様、SIMSCRIPTに於て分布を得ようとするとかなりの量のプログラムを書かなければならない。

SIMBOLでは簡便に分布を得るために、3つのオペレーターが用意されている。

(1) テーブルの名前、下限・上限・間隔を与えて分布収集用のテーブルを定義するためのオペレーター。

```
DTABLE   IMPUTIL(10)〔10, 100, 10〕;
         (テーブル名   (テーブル・コントロール
```

DTABLE(Distribution Table)は、整数型、実数型の両者が許されており、その識別は、テーブルコントロールの中に実数型が現われたか否かによる。これにより、10～20, 20～30 --- 95～100 までの離散間隔をもつテーブルが用意される。

これはGPSSのQTABLE宣言と同様の機能である。

この9つの間隔の他に、アンダーフローとして値10以下のもの及びオーバーフローとして値100以上のものの2つが自動的に定義される。更に表5.4.3に示すような項目が付加されてDTABLEが構成される。

表5.4.3　DTABLEの項目

1.	RANK数　　n
2.	$RANK_0 \sim RANK_{n+1}$
3.	代表値の最少
4.	〃　　最大
5.	RANK幅
6.	サンプル数
7.	Σv
8.	Σv^2
9.	max v
10.	min v
11.	サンプル開始時刻
12.	最新サンプル時刻

(2)　テーブルへのデータ書き込み用オペレーター

```
COLLECT  X, IMPUTIL(I);
          ↑
        データ
```

Xを待時間を示すデータとすれば、IMPUTILは待ち時間の分布を示すことになるし、発生時間間隔を示すデータであれば発生分布が、度数分布として簡単に求まるようになっている。この機能はGPSSのTABULATEブロックの機能に相当するものであり、ヒストグラムが簡単に得られるので非常に便利なものである。

(3)　テーブルを表示するためのオペレーター

```
DISPLAY  IMPUTIL(I);
```

DISPLAYによりテーブルの内容表5.6.　が表示される。

棒グラフを得たい時にはGRAPHプロセデュアを用いる。

棒グラフ表示は、SIMBOLのようにキャラクターディスプレイを用いたシステムでは非常に効果があがる。

図5.4.4は、キャラクターディスプレイ上に表示された棒グラフを示したものである。

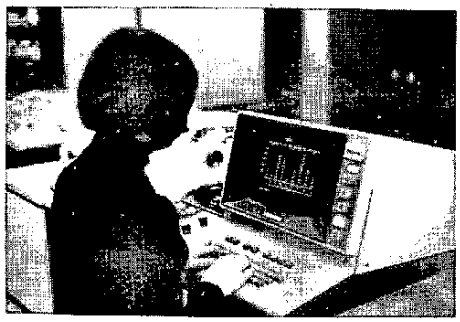


図5.4.4　グラフ表示

5.5 乱数と確率分布

例えば同一のモデルを別の計算機でランさせた場合、或は同じ系統のシミュレーションシステムを用いて別々の計算機でランさせた場合等出力結果が相当違ったものになるという事はよく経験する所である。これは乱数の違いに基づくものであり、どちらの結果がより正しいものかという判定は本来意味が無いわけである。そうは云っても計算機で表現する乱数というものは、本来の意味での乱数とは云い難く擬似乱数にすぎないわけで、1つの乱数系列上にある擬似乱数を適当に配分してしまうと思わぬトラブルが発生する場合もある。或は種々の乱数系列から適当に選択して用いていると、一様乱数の性格を損ねた確率変数の取り扱いをしてしまったりする場合もある。

ユーザはシミュレーションに際し十分な確率の知識を持つ事が必要であり、その上で正しく乱数というものを使用しないと何をやっているのか全くわからなくなってしまうであろう。どのシミュレーションシステムでも専用の乱数系列を何通りか用意しておりユーザが手軽に使用できる便宜を提供している。GPSSやSIMSCRIPTでは、いくつかの乱数系列というものが用意されており、ユーザは適宜系列を所い分けることにより互い

に独立な乱数を使用できる。

乱数の初期値はシステム側で各系列固有に用意しており、システム初期設定に際しその値を自動的に設定する。

SIMBOLやSIMULAでは、初期値をユーザに与えてもらうことにより別の乱数系列が得られるようになっている。いずれの方法が使い易いかということは一概に論ぜられないが、系列を用意する場合には10～20程度を必要とするであろうし、初期値を設定させる場合には代表的な初期値をシステム側で明示すべきであろう。

SIMBOLでは乗積合同法

$$x_{n+1} = k \cdot x_n \quad (\mathrm{Mod}\ M)$$

により得られる数列として乱数を得ている。

この方法ではkとMの値をどのように選択すれば、真の乱数に近い数列が得られるかという事、或は初期値 x_0 をどう選ぶかということが問題であり、いろいろな組み合せに対して検定を行ない望ましい初期値というものをどう設定したらよいかを示す事が必要である。

初期値の選び方に関しては、一般に

1＜（奇数）＜M（モジュールを取る数）

とされているが、これだけの情報から最適な初期値を求める事は非常に困難である。

初期値を（10^k+1）（k＝1～14）と1の15種類選択し係数kとMの組み合わせに対して度数検定、連検定、相関検定、組み合わせ検定を行なった所、全てを満足するものは稀にしかなかった。ただ余り小さすぎる数値や大きすぎる数値は好ましくなく、できれば除数Mの半分位のオーダの任意の奇数というものが比較的良い結果を示すようである。

GPSSでは一様乱数しか用意されていないので、一般の確率分布をFUNCTIONという形で定義せざるを得ないが、SIMBOLでは基本的な確率分布を表現するための変換乱数発生ルーチンが用意されているので

手軽に使用できる。

平均或は、平均と分散を初期値と一緒にパラメータとして与えるだけで理論分布を簡単に得ることができる。

表5.5.1にSIMBOLで提供している理論分布の一覧表を示す。

表5.5.1　理論分布一覧表

プロセデュア	説　　明	パラメータモード	関数モード	備　　考
RANDI (e1,e2,U)	e1～e2区間の一様乱数を値とする。	integer	integer	e2＞e1
RANDR (e1,e2,u)	e1～e2区間の一様乱数を値とする。	real	real	e2＞e1
POISSON (e, u)	平均eのポアソン分布乱数を値とする。	real	integer	e≦10.0
NORMAL (e1,e2,u)	平均e1，標準偏差 e2の正規分布乱数を値とする。	real	real	
NEGEXP (e, u)	平均 $1/e$ の負の指数乱数を値とする。	real	real	
ERLANG (e1,e2,u)	平均 $1/e1$，標準偏差 $1/(e1\ e2)$ のアーラン乱数を値とする。	real	real	e1,e2＞0

GPSSでは理論分布をも経験分布のように累積分布関数として表現しなければならずモデルの単純化に際し不便であった。しかし乍ら実際に対象とするシステムをモデル化しようとする場合に、理論分布が最初から適応できないことも多く、経験分布を簡単に使用できることも必要であろう。

GPSSでは

- 連続数値関数
- 離散数値関数
- リスト数値関数
- 離散属性値関数
- リスト属性値関数

の５つの関数形が用意されており経験分布が簡単に得られるという点では、ＳＩＭＳＣＲＩＰＴよりも優れている。

ＳＩＭＢＯＬやＳＩＭＵＬＡでも、ＧＰＳＳの連続数値関数、離散数値、リスト数値関数に相当するプロセデュアが用意されており、配列を用いて簡単に取り扱うことができる。

```
DISCRETE（A，B，U）
        配列　A，B：整数 U；
```

（整数型プロセデュア）

Ａは添字１の実数型の配列で図 5.5.1 のような離散的な累積分布関数の値が与えられる。

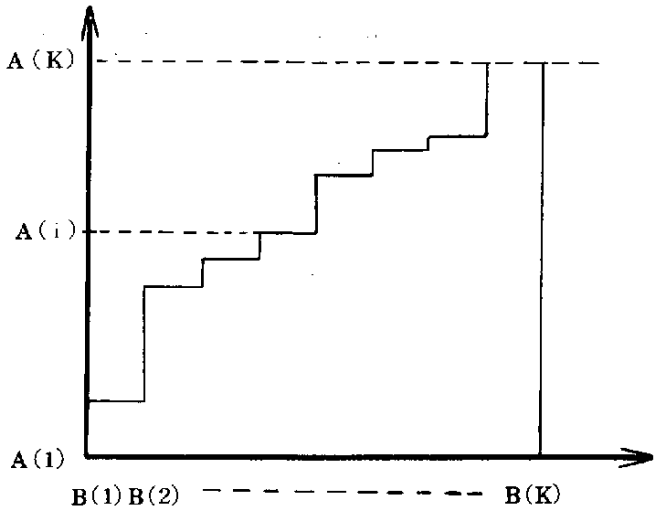


図5.5.1

（Ａ（K）は必ず１）

〔０，１.〕の一様乱数ｕがＡ（ｉ）＞ｕを満す最小の、Ｂ(ｉ)即ち（Ａ（ｉ）－Ａ（ｉ－１））の確率でＢ(ｉ)を得る。

LINEAR(A1, A2, u) 配列 A1, A2 ; 整数 U

（実数型プロセデュア）

A（i）とB（i）の関係は次のようになっている。

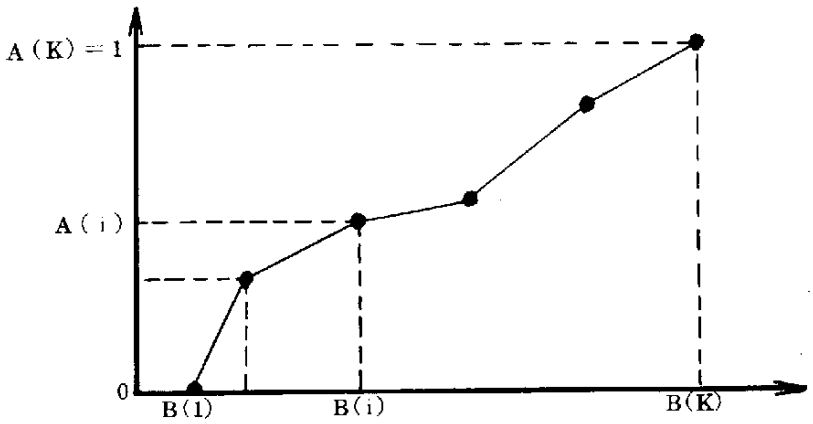


図 5.5.2

累積分布関数Fに従って〔B（1），B（K）〕の値が得られる。

第6章　シミュレーションラン

第6章　シミュレーションラン

6.1 初　期　設　定

モデルのチェックであるとシミュレーションランであるとを問わず、モデルプログラムを実行するためには、シミュレーションデータベースに必要な値を与えなければならない。これを初期設定と呼んでいるが、シミュレーションに於ては1回限りのランで済む場合は極めて稀な事なので、何回もランさせる場合に初期設定のし易さというものが問題になる。

SIMBOLでは、簡単に与えられるものはACCEPTステートメントを用いてキャラクターディスプレイから与えるようにしている。ACCEPTステートメントは、ダイレクトステートメントとしても使用できるので初期値の再設定に関してはかなり効力を発揮するものである。

データ個数が多いとか、複雑なデータなのでその都度キーインすることが非常に煩わしいというものに関しては、紙テープから読み込ませるようになっている。しかし乍らシミュレーションランごとに変わり得る性格のデータならば、何らさしつかえないであろうが、基礎データともいうべき変わりようのない性格のデータをランタイムに毎回設定してゆくのは、余り好ましいことではない。

FORTRANのDATAステートメントが持つような機能があれば、基礎データを1度設定するだけで済ませることができ、その後のランに際し例えば紙テープをかけかえるとかいった煩わしさも味わうずにすむわけである。これにより対応関係を取りながら値を与えられるので初期値設定ミスといったものも比較的少なくなることが予想される。

簡単にキーインできる基礎的なデータを恒久的に与える手段が用意されていればユーザは手軽に何度もシミュレーションランを行なえるであろう。SIMULA，SIMSCRIPT－Ⅱに於ても事情は同じであり、ランタイムにデータを読み込ませなければならない。

SIMSCRIPT－1.5の場合は、イニシャルコンディションズ・デックから必要な初期値をパーマネントエンティティに対して与える。

初期設定なのか再初期設定なのか、再初期設定は次のランのためのものなのか、エラーアボートされてプログラムが停止した時に行なうべきものなのかといった指定を初期設定カードに先立ってシステムスペシフィケーションカードにより行なう。これによりSIMSCRIPT－1.5では再ロードなしに1つ以上のデータデックの実行が可能となっている。

システムスペシフィケーションカードを表6.1.1に示す。

表6.1.1 System Specification Card Form

column	標準値	内容
1	1	カードの認識番号
2		もし0でなかったらinitializationカードのimageがoutput fileにリストされる。
3～6		不要
7～12	プログラマが与える	array numberの最大値
13～18	60	1時間当たりの分
19～24	24	1日当たりの時間
25		program停止後に再び初期値を与えるためのもの
26～30		不要
31～36	60	initial condition deckのためのfile番号
37～42	60	exogenous events deckのためのfile番号
43～48	61	reportのためのfile番号
49～54	55	1ページの制限行数

配列番号の最大値以外は必ずしも記述する必要はなく、その場合には標準値が自動的にセットされる。

ランごとに異なる乱数を使用したい場合、前回のランで使用した乱数をリセットしたい場合等の便宜のためコラム26以降の機能が用意されている。

表 6.1.2 に示した初期条件デックの四角で囲んだ部分がシステムスペシフィケーションカードであり、その後に配列番号に従ってパーマネントエンティティに与えるべき初期値が記述されている。

表6.1.2 Initial Condition Deck

```
INITIALIZATION CARDS
11          42    100      10
  1       0 R                                                    10
  2       1 R      10     1                                      10(I3)
10 10  10 10 10  10 10 10 10 10
  3       1 R      10     1                                      10(I3)
27 13  14 11  9   8  6  5  3  1
  4       1 R      10     1                                      10(I3)
28 24  26 12 21  10  7 15  4  2
  5       1 R      10     1                                      10(I3)
 0 25   0 23 22  19 18 17 15  0
  6       1 R      10     1                                      10(I3)
0  0   0  0  0   20 0  0  0  0
  7   12  1  Z     10     1
 13       0 R                                                    28
 14       1 R      28    13                                      28(I2)
9 8 8 7 6 6 5 5 4 2 2 3 1 11011 9 9 8 7 7 6 5 6 4 4 2 3
 15       1 R      28    13                                      17(I4)
 32       0 R                                                    5
 33   36  2  Z      5    32   28   13
 37       0 R                                                    1
 38   39  1  Z     28    13
 40       0 R                                                    1
 41   42  1  Z     28    13
```

同一データを配列要素に一括して与える便宜はない。但しゼロクリアする場合だけは特別にそのような機能を提供している。

配列番号は必ずしも連続している必要はないが、パーマネントエンティティを定義した配列番号は必ずここで値を与えなければならない。

GPSS−Vの場合は、グローバルなデータとしてセイブバリューといわれるものがあり、これにINITIALステートメントを用いて初期値を与

える。INITIALステートメントは同時にロジックスイッチをセットするためにも使用される。ロジックスイッチは、INITIALステートメントで指定されない場合はリセットされた状態である。

セイブバリューには単純変数と配列の2種類があり、配列の方はマトリックスセイブバリューと呼ばれている。セイブバリューには整数タイプでも実数タイプのデータでも入れられるようになっている。

INITIALステートメントの使用に実数タイプデータの扱いが許されたのは、GPSS−Vに改良されてからの事である。

INITIALステートメントは1セパレータで区切る事によりデータ並びを記述できるし、ハイフンで連結して一括した初期値の設定も可能となっている。

表6.1.3は、ネットワークフローモデルのシミュレーションに際し用いたセイブバリューの初期設定の様子である。GPSS−V, SIMSCRIPT−1.5の場合には、システム側で初期設定の便宜を提供している訳だが、見

表6.1.3　GPSS初期設定

```
****          INITIAL  SET   ****
              INITIAL     X1,18
              INITIAL     X2,24
              INITIAL     X3,2C
              INITIAL     X4,18
              INITIAL     X5,18
              INITIAL     X6,3C
              INITIAL     X7,20
              INITIAL     X8,18
              INITIAL     X9,20
              INITIAL     X10,16
              INITIAL     X6C,20C
****          MATRIX DEFINE   ****
**   ROUT MATRIX   **
 1        MATRIX      MX,1C,10
              INITIAL     MX1(1,1),0
              INITIAL     MX1(2,1),1
              INITIAL     MX1(3,1),1

              INITIAL     MX1(8,10),1C
              INITIAL     MX1(9,10),10
              INITIAL     MX1(10,10),0
**   CIRCUIT MATRIX   **
 2        MATRIX      MX,20,10
              INITIAL     MX2(1,2),23
              INITIAL     MX2(1,3),24
              INITIAL     MX2(2,4),21
```

方を変えると、別の手段に欠けるためであるとも云える。GPSS－Vの場合、INITIALステートメントが無かった場合の初期設定は、初期設定用のトランザクションを発生させ、SAVEVALUE或はMSAVEVALUEブロックを実行させて行なわなければならない。SIMSCRIPT－1.5の場合は、パーマネントエンティティの大きさを実行に先立ち指定してデータ領域を確保しなければならない。いずれにしろ初期設定という事が簡便

表6.1.4　SIMSCRIPT－Ⅱに於ける初期設定

```
ROUTINE  FOR  INITIALIZATION
''
''     /*  IMP  INFORMATION  SETTING-UP  */
''
          LET  N.IMP=10          CREATE  EVERY  IMP
               FOR  EACH  IMP , DO
                    READ  BUF.SIZE,CIRC.ENT1,CIRC.ENT2
               LOOP
''
''     /*  CIRCUIT INFORMATION SETTING-UP */
''
          LET  N.CRCT=28         CREATE  EVERY  CRCT
               FOR  EACH  CRCT, DO
                    READ  BELONG.IMP,TRANS.TIME
               LOOP
''
''      /* NETWORK  PATH  SELECTION  TABLE */
''
        RESERVE  NETWORK(*,*) AS 10 BY 20
         FOR  I=1 TO 10,FOR J=1 TO 20,DO
               READ  NETWORK(I,J)
          LOOP
''
''      /* NETWORK SELECTION PROBABILITY TABLE */
''
          RESERVE  PROB(*,*)  AS  10 BY 10
          FOR  I=1 TO 10,FOR  J=1 TO 10,DO
               READ  PROB(I,J)
          LOOP
''
''     /*  MESSAGE  CALL  OCCURRENCE  */
''
RETURN         END
```

に行なえる方が良いわけで、そのし易さの相違というものは初期値の再設定に於て明白に見てとれるであろう。

SIMSCRIPT－Ⅱに於ける初期設定をSIMBOLやSIMULA等の代表例として表6.1.4に示す。

初期設定は、ユーザの定義するデータのみならず、各種統計量とも関連するわけで、1回限りのランであるならば、各種統計量を全てクリアーしておくといった単純なものでもよい。しかも、SIMSCRIPT，SIMULAのように自動的に統計収集を行なう機能を持たないシステムでは、初期設定もユーザにまかせる以外に方法がない。

SIMSCRIPT－1.5の場合には、統計量に関しては、ユーザが全て責任をもって初期設定しなければならないが、パーマネントエンティティに対する初期設定の機能があり、非常に不便なものではあるが一応の目的は達成できるよう配慮されている。

統計量を取るためのエリアはパーマネントエンティティに取るのが普通であるから、初期設定カードでこれを行なうことができる。

GPSSではINITIALステートメントに依り行なわれ、RESET，CLEARステートメントを組み合わせて用いることにより、最初に設定した値をそのままにしておくこともできるし、一部分だけをそのままにしておき、残りをINITIALステートメントを用いて再設定をすることも簡単に行なえる。

SIMBOLもGPSS同様RESET，CLEARコマンドが用意されており、初期値の再設定を簡易化しているが、GPSSほど自動的に取り扱う対象がないので、SIMULAやSIMSCRIPTよりは便利になっている程度であろう。

6.2 トレース処理とモニタリング

ユーザが非常に頭を悩ませるものの1つに、モデルの妥当性のチェック

がある。実際にモデルを組んだ経験を持つユーザは、それがどの位大変な作業であるか身につまされて感じるであろう。モデルの構成とその妥当性のチェックに費す労力は、全体のシミュレーション実験に於ける50％近い比率を占めるというデータもある。

プログラム自体に論理的なミスがあるのか、モデル自体の論理ミスなのか、或は両方なのかといった状況が発生し、妥当性のチェックを複雑なものにしてしまう場合が多い。特にSIMBOLのようにステートメントを用いてプログラムを構成するようなシステムに於ては、ユーザが望むロジックを構成し得たか否かをチェックするために費す労力は相当のものになるであろう。設計する立場から見ると、一体何がモデルの妥当性をチェックする上で効果的な情報と成り得るかということが大きな問題である。ユーザの作成するモデルが多様であり、ユーザ自身の妥当性のチェックの仕方そのものが多様であることからすると一体どの程度の機能が必要なのかを見極めることはむずかしいことである。

更にトレースはモデルのデバッグの為に使用される場合が極めて多く、どういうタイミングで、何を見ようとするのかというユーザ特性も考慮する必要がある。

トレースは本来情報が与えられるだけではなく、その情報の解析結果により何らかのアクションに連なって行くべきものであり、ランが終了した時点で一括して提示されるようであってはランを無駄にする事にもなりかねない。トレース状態を解析し乍ら異常に気づいた時点で修正を施すなり、やめてしまうとかいったアクションが取れれば非常に効果的にデバッグが行なえる。トレース範囲を狭めたり、種類を変更したりといった操作も簡単に行なえなければ無意味な情報を収集することにもなりかねないし、トレースの開始、終了に関する指示も自由に行なえなければならないであろう。

SIMBOLではTRACEコマンドを用いてトレース種類の指定や、実行

の指定を行なっている。トレース種類の指定と、トレースの実行・停止とは全く別の機能としてあるために、SIMULAやGPSSで提供しているトレースよりも操作性に於て優れている。

6.2.1 トレースコマンド

SIMBOLに於けるトレースは、TRACEコマンドを用いて行なう。TRACEのサブコマンドとして5種類のものが用意されている。

トレースをかけるタイミングの問題（開始・終了）と、何をトレースするかという情報指示の問題（設定・解除）という単位を、SIMBOLでは明確に区分している。

タイミングに関するサブコマンドとしてトレースの開始・終了指示がある。

```
TRACE   START   [PAUSE   *AFTER ]
                        BEFORE]
                  ↑option

TRACE   STOP
```

STARTサブコマンドはトレースの実行開始を指示するものであり、オプションとしてPAUSE指定ができる。PAUSEはその名の通り、プログラムの実行を一時停止することを指示する。

AFTERはトレース対象にシステムが遭遇した時点で、トレース処理実行直後の停止を、BEFOREはトレース実行直前の停止を、＊印は直前直後の停止を意味する。

ユーザは、トレース対象の実行直前、直後にプログラムの実行を一時中断し、各種モデルステータスのモニタリングを行なうことができ、トレースと併合して用いられることにより極めて強力なデバッグを可能にしている。

STARTサブコマンドはシミュレーションランの任意時点でキーイン

でき、且つこのコマンドがキーインされなければトレースの実行は行なわない。

STOPサブコマンドは、トレース処理を終了させるためのものであり、STARTと組み合わせて、非常にダイナミックなトレース処理が可能となっている。

何をトレースするかというトレース対象の指示は、ON, OFFサブコマンドにより行なわれる。

```
TRACE   ON
TRACE   OFF
```

ON, OFFサブコマンドはトレーススイッチの操作を行なう。トレーススイッチがON（セット）されてもSTARTサブコマンドが入れられなければトレース処理は行なわない。OFF（リセット）サブコマンドは、トレーススイッチにセットされた対象をリセットする場合に用いられる。ダイナミックなトレーシングは、上記4種のサブコマンドを適当に組み合わせて行なわれる。

図6.2.1はダイナミックトレースの様子を図式化したものである。

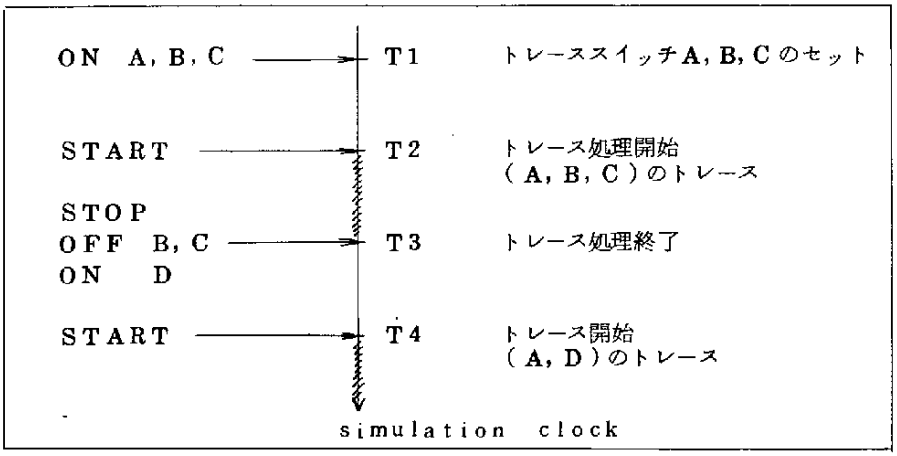


図6.2.1 ダイナミックトレース

トレーススイッチのセット、リセットはシミュレーションランを通じて何度でも行なうことができ、従ってトレース対象を変えたり、範囲を絞ったり或は広げたりということが容易にできる。

トレーススイッチのONサブコマンドでセットされたものは、START, STOPにより時間軸を分割して実行される時に始めて意味を成すわけだが、OFFにされない限りどういうタイミングでトレース処理が行なわれてもトレース対象は変わらない。

スイッチを無効にするためにOFFサブコマンドが用意されているわけだが、以前スイッチをどうセットしたかを知っていた方が都合が良い。スイッチの状態を表示させるためにVIEWサブコマンドが用意されている。

TRACE　VIEW

VIEWサブコマンドで表示されたスイッチの状態を見て、次のスイッチ操作を行なえば便利であろう。

6.2.2　トレース対象

トレーススイッチの操作はON, OFFサブコマンドに依り行なわれる。トレース対象は、表6.2.1に示すような4種類が用意されている。

表6.2.1　トレース対象

- プロセストレース（PROC）
  - プロセスの発生・消滅（PCRD）
  - スケジュール関係（PSCHD）
  - グループ操作（GROUP）
- イベントトレース（EVENT）
- ステータストレース（STATE）
- シーケンストレース（PSEQ）

プロセストレースと総称されるものには、プロセスの発生、消滅をトレースするもの、プロセスのスケジュール・リスケジュール・キャンセル・ターミネイト・インタラプト・レジューム等のスケジューリングに関するもの、更にプロセスをセットやキューといったグループ内で操作するものといった3種類がある。

イベントトレースはスケジュールテーブルから取り出されて実行されて行くプロセスをチェックするためのものである。ここでは実行されたプロセス全てをチェックするものと、特定のアクティビティに属するプロセスの実行のみに限ったものとがある。

ステータストレースは、モデルの状態変化をグローバル変数に関連させてチェックするためのものであり、アサインステートメント等により変数の値が変わる度にチェックを可能にする。

シーケンストレースはプログラムの流れを追跡するもので、BRANCH, DOといった制御ステートメント、或はCALLステートメントによるプログラムのコーリングチェック、スケジューリング関係のチェック、ラインナンバーによるもの、プログラムの一部分のチェック等ができる。

プロセストレースは、全ての関連するプロセスという表現と、特定のアクティビティに属するものだけという表現とが許され、トレース対象を絞ることができる。

```
TRACE   ON  PROC PSCHD   Activity list
        or
        OFF
                 PSCHD  *
```

*印は全指定、Activity listという形で特定のアクティビティ名を記述すればそのアクティビティに属するプロセスのみに限定される。

イベントトレースもプロセストレース同様、*かアクティビティリス

トの記述が可能である。

ステータストレースはグローバル変数の並びを記述する。シーケンストレースは、全ステートメントのトレース指示＊から始まり、プログラム単位を絞り、更にプログラム単位内で対象ステートメントを制限したり、範囲を制限したりできる。

PSEQをキーインするとプログラム単位を絞るか否かの問い合わせが成され、プログラム単位を制限すれば、その中で更に対象を制限するか否か問い合わせが来る。コントロールステートメント（DO，IF，BRANCH等）だけに限るとかいった指定をすることによりプログラム内でトレースすべき対象が限定される。

トレースコマンドがON, OFFのレベルから、PSEQ, EVENT等のトレース種類、PSEQからは更にプログラム単位、特定ステートメントというように階層的に構成されており、トレース情報を有効に絞り込むことが可能となっているために、システム側はユーザの指定が完了する迄、問い合わせを続けるようになっている。

キャラクターディスプレイからSENDキーを押す事により、ON、OFFのサブコマンドレベルに戻る。

トレーススイッチの操作を終了したい場合に¥マーク、或はSTARTサブコマンドのキーインを行なえばよい。このようにトレース対象を自由に変更し、有効な情報だけを収集できればユーザの負担は非常に軽減されるであろう。

GPSSが提供するTRACE，UNTRACEの指示は開始、終了はスタティックに決まってしまうわけだし、ブロックの範囲は制限できるが、変更できない。更にトレース対象はSIMBOLのシーケンストレース指定における全指定（即ち実行されたブロック全てが対象となる）と等価なものであり、トレース情報を減らすことはできない。その中から有効な情報を得ようとするのは大変な労力を要するし、トレース結果を見て

フィードバックをかける事もできないので効果的ではない。

6.2.3 トレース表示方法

6.2.2節でどういう種類のトレースがSIMBOLで行なえるかを示したので、ここではトレース対象がどう表示されるかという事を検討しよう。プロセストレースの情報表示は図6.2.2に示すように、アクションを起こした側と起こされた側の情報が得られるようになっている。

<トレース種類を示す記号>	<対象となったプロセスの情報>	<アクションを起こした側の情報>	<現在時刻>
4	18	8	10

図6.2.2 プロセストレース表示情報－1

トレース種類を示す記号は、プロセスの発生・消滅・スケジューリング・グループ操作といった種類を識別するためのもので表6.2.2に示すような10種類の記号がある。 4桁の省略記号で10種類が識別される。

表6.2.2 プロセストレース表示情報－2

種　　類	表示記号（4ケタ）
○ プロセスの発生	GNRT
○ プロセスの消滅	DSTY
○ プロセスがスケジュールされた	SCHD
○ プロセスがリスケジュールされた	RSCH
○ プロセスがキャンセルされた	CACL
○ プロセスがターミネイトした	TRMT
○ プロセスがインタラプトされた	INTR
○ プロセスがレジュームされた	RESM
○ プロセスがセット又はキューに入れられた	INST
○ プロセスがセット又はキューから取り出された	REMV

トレース情報は、対象となったプロセスに関するものとして、アクティビティ名、プロセスシリアルナンバー、セットカウント、アクション

を起こした側のものとしてアクティビティ名、更にその時のシミュレーションクロックの値が表示される。これによりどの時刻にどのアクティビティが、どのアクティビティに属す何番目のプロセスを対象にどういうアクションを起こしたかが一目してわかるようになっている。そしてもし、セットやキューといったグループのメンバーになっている時には、いくつのグループに属しているものであるかわかるようセットカウントを示している。

イベントトレースの表示情報は、スケジュールテーブルと関連したものであり、図6.2.3に示すような4種類の情報がある。

<現在時刻><アクティビティ名><プロセスシリアルナンバー><リアクティベーションポイント>

図6.2.3　イベントトレース表示情報

プロセスは、一連のイベントの集合体であり、各イベントの実行に際してはリアクティベイションポイントの参照により行なわれる。従ってどのプロセスかがわかっただけでは、プロセスを構成しているどの部分のイベントかが明らかではない。

リアクティベイションポイントをライン番号で表示する事に依り、どのイベント部を実行したのかが分かる様になっている。

ステータストレースの表示情報はグローバル変数の変化を示すものである。グローバルデータはアクティビティからも直接参照できるので、メインプログラムをも含めてどのプログラム単位の、どのステートメントにより変化させられたのか識別できるようになっている。

<変数名><変化後の値><プログラム名><ラインナンバー>

図6.2.4　アサインステートメントに関する
ステータストレース表示方法

シーケンストレースの表示情報は単純であり、プログラム名、ライン

ナンバー、ステートメント種類である。

＜プログラム名＞＜ラインナンバー＞＜ステートメント種類＞

図6.2.5　シーケンストレース表示情報

GPSSのトレースは、SIMBOLのシーケンストレースの一部分程度の機能しかない。その表示情報は、時刻何時何分にどのトランザクションがどのブロック命令を実行し、ターミネーションカウント（終了予定トランザクション数の残り数）がいくつかを示す。

SIMBOLのように特定のブロックに限ってトレースを行なうとか、或は1つのトランザクションの動きだけをトレースすることが困難であり、トレース対象を絞って効率良くデバッグを行なう便宜に欠けている。

トレース対象が豊富であり、対象を絞る事ができ、且つ自由なタイミングコントロールの下でトレース可能であるSIMBOLのトレースは、モデルの妥当性のチェックのためにも十分利用し得るものであろう。

6.2.4　モニタリング

SIMBOLでは、STARTサブコマンドにPAUSEフレーズを設けトレース情報とモニタリング情報とを有機的に結合して、モデルの振舞いを観察できるようになっている。その他にシミュレーションランの任意時点でクイットをかけてモニタリングする事も可能であるし、PAUSEコマンドによるモニタリングも可能である。

PAUSEコマンドには2種類のものがあり、予定時間が経過したらPAUSEさせるものと、サイクリックにPAUSEさせるものとである。

サイクリックなPAUSEとは例えば100クロック毎にプログラムランを一時中断させるような機能であり、その時間が経過する度に、ユーザはモデルの状態を確認することができる。

PAUSE状態になった時のモニタリングはCALLコマンドによりプロセデュアを用いて行なう。或はDISPLAYコマンドにより表示する事

も可能である。

SIMULA、65には、"Full Trace", "Selection Trace"がありトレース情報を制限できる。

ネットワークモデルに於て"Full Trace"機能を使用したので、その結果を示すと共に若干のコメントを付加しておこう。

SIMULAの"Full Trace"はSIMBOLのプロセストレースの機能即ち

TRACE ON PROC ＊

と等価であり、プロセスの発生・消去・スケジューリング・グループ操作に関するトレースを行なう。

表示コードによりプロセスに対して成された操作が識別される。表6.3.1に表示コードとその意味を示す。（SIMULA−65 Manualより）

"Full Trace"に関する情報は表6.3.2に示すような形で出力される。表6.3.2の先頭の英字コードが表6.3.1に対応するものである。1行には2つのプロセスに関するトレース情報が出力される。

左側がアクションを起した側のプロセスに関する情報であり、右側がアクションを取られた側、或はアクションの結果制御が移った側のプロセスに関する情報である。

左側半分に関して説明しよう。

まず先頭の表示コードがアクションの種類を表6.3.1に従って示す。

次の数値はイベントの切替え総数、その後にアクティビティ名が出力される。

次の3ブロックでプロセスのシリアルナンバー（同一アクティビティから発生された何番目のプロセスかを示す）、リアクティベイションライン、トレースシステムの呼出しラインが表示される。

表6.2.3 SIMULA-65に於けるトレース表示コード

Generation and deallocation
Operations on the sequencing set
Connection
Set operations

CODE	NAME	SECOND PROCESS DESCRIPTOR DESCRIBES:
A	activate	Process designated by activate statement
B	event	Next active process (see Table E-2)
C	connect	Connected process
D	deallocate	Process to be deallocated
E	extract	Extracted process
F	sequencing	Operations on the sequencing set (described by code printed – see Table E-2)
G	generate	Generated process
H	hold	Next active process
I	inspect	Inspected process
J	storage	(Described by code printed – see Table E-2)
P	passivate	Next active process
Q	define RP	Process getting a reactivation point, which is an implicit passivate at the end of all declarations in an activity.
R	reactivate	Process designated by reactivate statement. Reactivate (current) is a special case – an extra process descriptor then designates the next active process.
S	cancel	Cancelled process. Cancel (current) is a special case – an extra process descriptor then designates the next active process.
T	terminate	Terminated process. Terminate (current) is a special case – an extra process descriptor then designates the next active process.
W	wait	Next active process
X	activity end	Next active process
+	include	Process aspect of affected element
–	remove	Process aspect of affected element

次に時刻の表示が成され、最後の２ブロックでリファランスカウント（プロセスを参照するためのポインター数）、カーディナル（セットに属している個数）の表示が成される。

メインブロックはシリアルナンバーは常に０である。

```
A  12 ACCEPT          5  113  122  1.4458,+00   1  0   SENDACT          5   59  122  1.4458,+00   1  0
G     SENDACT         5   59  126  1.4458,+00   1  0   ACCEPT           6   88  126  * * * * *    1  0
A     SENDACT         5   59   70  1.4458,+00   1  0   ACCEPT           6   88   70  * * * * *    1  0
H  13 SENDACT         5   71   ??  1.4458,+00   1  0   ACCEPT           2   88   71  5.2200,+00   1  0
R     ACCEPT          2   88  103  5.2200,+00   1  0   SENDACT          1   71  103  1.0100,+01   1  0
H  14 ACCEPT          2  111  111  5.2200,+00   1  0   SENDACT          1   71  111  5.2200,+00   1  0
X  15 SENDACT         1   71   84  5.2200,+00   1  0   ACCEPT           2  111   84  5.3200,+00   1  0
X  16 ACCEPT          2  111  122  5.3200,+00   1  0   MAIN BLK  1      0  185  122  5.9102,+00   1  0
G     MAIN BLK  1     0  185  183  5.9102,+00   1  0   ACCEPT           7   88  183  * * * * *    1  0
A     MAIN BLK  1     0  185  184  5.9102,+00   1  0   ACCEPT           7   88  184  * * * * *    1  0
H  17 MAIN BLK  1     0  185  185  5.9102,+00   1  0   ACCEPT           7   88  185  5.9102,+00   1  0
H  18 ACCEPT          7  111  111  5.9102,+00   1  0   ACCEPT           7  111  111  6.0102,+00   1  0
G     ACCEPT          7  111  112  6.0102,+00   1  0   SENDACT          4   59  112  * * * * *    1  0
A     ACCEPT          7  111  113  6.0102,+00   1  0   SENDACT          4   59  113  * * * * *    1  0
X  19 ACCEPT          7  113  122  6.0102,+00   1  0   SENDACT          4   59  122  6.0102,+00   1  0
G     SENDACT         4   59  126  6.0102,+00   1  0   ACCEPT           8   88  126  * * * * *    1  0
A     SENDACT         4   59   70  6.0102,+00   1  0   ACCEPT           8   88   70  * * * * *    1  0
H  20 SENDACT         4   71   71  6.0102,+00   1  0   ACCEPT           4   88   71  6.1196,+00   1  0
R     ACCEPT          4   88  103  6.1196,+00   1  0   SENDACT          2   71  103  1.1000,+01   1  0
H  21 ACCEPT          4  111  111  6.1196,+00   1  0   SENDACT          2   71  111  6.1196,+00   1  0
X  22 SENDACT         2   71   84  6.1196,+00   1  0   ACCEPT           4  111   84  6.2196,+00   1  0
X  23 ACCEPT          4  111  122  6.2196,+00   1  0   ACCEPT           6   88  122  6.5658,+00   1  0
R     ACCEPT          6   88  103  6.5658,+00   1  0   SENDACT          3   71  103  1.1446,+01   1  0
H  24 ACCEPT          6  111  111  6.5658,+00   1  0   SENDACT          3   71  111  6.5658,+00   1  0
X  25 SENDACT         3   71   84  6.5658,+00   1  0   MAIN BLK  1      0  185   84  6.6096,+00   1  0
G     MAIN BLK  1     0  185  183  6.6096,+00   1  0   ACCEPT           9   88  183  * * * * *    1  0
A     MAIN BLK  1     0  185  184  6.6096,+00   1  0   ACCEPT           9   88  184  * * * * *    1  0
H  26 MAIN BLK  1     0  185  185  6.6096,+00   1  0   ACCEPT           9   88  185  6.6096,+00   1  0
H  27 ACCEPT          9  111  111  6.6096,+00   1  0   ACCEPT           6  111  111  6.6658,+00   1  0
X  28 ACCEPT          6  111  122  6.6658,+00   1  0   ACCEPT           9  111  122  6.7096,+00   1  0
G     ACCEPT          9  111  112  6.7096,+00   1  0   SENDACT          5   59  112  * * * * *    1  0
A     ACCEPT          9  111  113  6.7096,+00   1  0   SENDACT          5   59  113  * * * * *    1  0
X  29 ACCEPT          9  113  122  6.7096,+00   1  0   SENDACT          5   59  122  6.7096,+00   1  0
G     SENDACT         5   59  126  6.7096,+00   1  0   ACCEPT          10   88  126  * * * * *    1  0
A     SENDACT         5   59   70  6.7096,+00   1  0   ACCEPT          10   88   70  * * * * *    1  0
H  30 SENDACT         5   71   71  6.7096,+00   1  0   MAIN BLK  1      0  185   71  7.0787,+00   1  0
G     MAIN BLK  1     0  185  183  7.0787,+00   1  0   ACCEPT          11   88  183  * * * * *    1  0
A     MAIN BLK  1     0  185  184  7.0787,+00   1  0   ACCEPT          11   88  184  * * * * *    1  0
H  31 MAIN BLK  1     0  185  185  7.0787,+00   1  0   ACCEPT          11   88  185  7.0787,+00   1  0
H  32 ACCEPT         11  111  111  7.0787,+00   1  0   ACCEPT          11  111  111  7.1787,+00   1  0
G     ACCEPT         11  111  112  7.1787,+00   1  0   SENDACT          6   59  112  * * * * *    1  0
A     ACCEPT         11  111  113  7.1787,+00   1  0   SENDACT          6   59  113  * * * * *    1  0
X  33 ACCEPT         11  113  122  7.1787,+00   1  0   SENDACT          6   59  122  7.1787,+00   1  0
G     SENDACT         6   59  126  7.1787,+00   1  0   ACCEPT          12   88  126  * * * * *    1  0
A     SENDACT         6   59   70  7.1787,+00   1  0   ACCEPT          12   88   70  * * * * *    1  0
H  34 SENDACT         6   71   71  7.1787,+00   1  0   MAIN BLK  1      0  185   71  8.7892,+00   1  0
G     MAIN BLK  1     0  185  183  8.7892,+00   1  0   ACCEPT          13   88  183  * * * * *    1  0
A     MAIN BLK  1     0  185  184  8.7892,+00   1  0   ACCEPT          13   88  184  * * * * *    1  0
H  35 MAIN BLK  1     0  185  185  8.7892,+00   1  0   ACCEPT          13   88  185  8.7892,+00   1  0
H  36 ACCEPT         13  111  111  8.7892,+00   1  0   ACCEPT          13  111  111  8.8892,+00   1  0
G     ACCEPT         13  111  112  8.8892,+00   1  0   SENDACT          7   59  112  * * * * *    1  0
A     ACCEPT         13  111  113  8.8892,+00   1  0   SENDACT          7   59  113  * * * * *    1  0
X  37 ACCEPT         13  113  122  8.8892,+00   1  0   SENDACT          7   59  122  8.8892,+00   1  0
W  38 SENDACT         7   59   64  8.8892,+00   1  0   ACCEPT           8   88   64  1.1130,+01   1  0
R     ACCEPT          8   88  103  1.1130,+01   1  0   SENDACT          4   71  103  1.6010,+01   1  0
A     ACCEPT          8  103  105  1.1130,+01   1  0   SENDACT          7   64  105  * * * * *    1  1
H  39 ACCEPT          8  111  111  1.1130,+01   1  0   SENDACT          4   71  111  1.1130,+01   1  0

G     ACCEPT        210  111  112  9.9078,+01   1  0   SENDACT        116   59  112  * * * * *    1  0
A     ACCEPT        210  111  113  9.9078,+01   1  0   SENDACT        116   59  113  * * * * *    1  0
X 760 ACCEPT        210  113  122  9.9078,+01   1  0   SENDACT        116   59  122  9.9078,+01   1  0
W 761 SENDACT       116   59   64  9.9078,+01   1  0   ACCEPT         195   88   64  9.9169,+01   1  0
R     ACCEPT        195   88  103  9.9169,+01   1  0   SENDACT        105   71  103  1.0405,+02   1  0
A     ACCEPT        195  103  105  9.9169,+01   1  0   SENDACT        116   64  105  * * * * *    1  1
H 762 ACCEPT        195  111  111  9.9169,+01   1  0   SENDACT        105   71  111  9.9169,+01   1  0
X 763 SENDACT       105   71   84  9.9169,+01   1  0   SENDACT        116   64   84  9.9169,+01   1  0
G     SENDACT       116   64  126  9.9169,+01   1  0   ACCEPT         211   88  126  * * * * *    1  0
A     SENDACT       116   64   70  9.9169,+01   1  0   ACCEPT         211   88   70  * * * * *    1  0
H 764 SENDACT       116   71   71  9.9169,+01   1  0   MAIN BLK  1      0  185   71  9.9198,+01   1  0
G     MAIN BLK  1     0  185  183  9.9198,+01   1  0   ACCEPT         212   88  183  * * * * *    1  0
A     MAIN BLK  1     0  185  184  9.9198,+01   1  0   ACCEPT         212   88  184  * * * * *    1  0
H 765 MAIN BLK  1     0  185  185  9.9198,+01   1  0   ACCEPT         212   88  185  9.9198,+01   1  0
H 766 ACCEPT        212  111  111  9.9198,+01   1  0   ACCEPT         195  111  111  9.9269,+01   1  0
X 767 ACCEPT        195  111  122  9.9269,+01   1  0   ACCEPT         212  111  122  9.9298,+01   1  0
G     ACCEPT        212  111  112  9.9298,+01   1  0   SENDACT        117   59  112  * * * * *    1  0
A     ACCEPT        212  111  113  9.9298,+01   1  0   SENDACT        117   59  113  * * * * *    1  0
X 768 ACCEPT        212  113  122  9.9298,+01   1  0   SENDACT        117   59  122  9.9298,+01   1  0
G     SENDACT       117   59  126  9.9298,+01   1  0   ACCEPT         213   88  126  * * * * *    1  0
A     SENDACT       117   59   70  9.9298,+01   1  0   ACCEPT         213   88   70  * * * * *    1  0
H 769 SENDACT       117   71   71  9.9298,+01   1  0   SENDACT         55   71   71  1.0004,+02   1  0
G     SENDACT        55   71  126  1.0004,+02   1  0   ACCEPT         214   88  126  * * * * *    1  0
A     SENDACT        55   71   70  1.0004,+02   1  0   ACCEPT         214   88   70  * * * * *    1  0
H 770 SENDACT        55   71   71  1.0004,+02   1  0   ACCEPT         197   88   71  1.0004,+02   1  0
R     ACCEPT        197   88  103  1.0004,+02   1  0   SENDACT        115   71  103  1.0841,+02   1  0
H 771 ACCEPT        197  111  111  1.0004,+02   1  0   SENDACT        115   71  111  1.0004,+02   1  0
X 772 SENDACT       115   71   84  1.0004,+02   1  0   MAIN BLK  1      0  185   84  1.0009,+02   1  0
```

表6.2.4 SIMULA－65 Full Trace Output

6.3 ランタイムチェック

SIMBOLではテンポラリーなデータのランタイムのチェックを厳密に行なっている。これもモデルの妥当性のチェックを容易にしようとする配慮からであり、その為に実行速度が落ちることは否めない。しかし乍ら常に厳密なチェックを行なう訳ではなく、モデルが妥当であるとの判断がなされるならばCHECKコマンドによりランタイムチェックをスキップさせる事もできるよう構成されている。ランタイムチェックを厳密にしないとエラーの原因が見失なわれてしまう場合が多く、修正のために多大な時間を費すはめになる。そうでなくともモデルの妥当性のチェックが困難な作業であるわけで、軽微なランタイムチェックはその困難さを助長するのみであり、実行速度を向上させた所で全体の効率から見れば逆にマイナスでしかない場合の方が多い。

SIMBOLでは、プロセス・ブロック等のテンポラリーデータに関するエラーチェックを非常に厳密に行なっている。プロセスやブロックはセットやキューといった集団のメンバーになるものでGPSSの場合のトランザクション、或はSIMSCRIPTの場合のテンポラリエンティティと同様のものである。

集団への加入を行なうプログラム単位が、そのメンバーを参照しようとするプログラム単位と別であり、且つ加入を解除するプログラム単位も別であるといった状況に於ては、データの参照に先立ち参照しようとする対象が存在するか否かをチェックする必要がある。そうでないと存在しないデータにアクセスすることになり、仮りにそのような事が行なわれたとしても無意味なデータを扱うことになり、別の所で別のエラー原因を誘発する。別のエラーとなって現われる場合はまだしも、何か無意味な事をやり続けてシミュレーションランを完了してしまった場合には、シミュレーション結果というものが意味を成さないにもかかわらず出力されるのでエラの発見を非常に遅らせることにもなる。

セットやキューの処理が比較的単純であり、固有の名前でメンバーを操作できないような場合には、セットやキューが空であるか否かといった簡単なチェックだけであっても十分であろう。

SIMBOLのようにプロセスやブロックに名前が何通りでも付けられるようなシステムに於ては、今参照しようとする名前とは別の名前で対象データが取り除かれていることがありそれをチェックしなければならない。そのためにSIMBOLでは、特別なキーを設けており、プロセスやブロックをエレメント変数を通して参照する際に両者のキーが一致するか否かチェックしている。

プロセスやブロックはスペース管理上、固定部分（PCBF或はBLKFと示してある）と可変部分（PCBV／BLKV）とに分かれており、固定部分はプロセス等の発生、消去を制御するために使用している。その固定長の部分にキーが設けられており、図6.3.1に示すように3つの部分から構成されている。

Activeスイッチは、プロセス或はブロックの存否を示す。

SME,QMEはセット又はキューのメンバーであるか否かを、グループカウンタは所属するグループの数を示す。

Key として示したものは世代を示す数値である。

参照のしかたを図6.3.2に示してある。

チェックは次の手順で行なわれる。まずエレメント変数が指しているデータがアクティブか否かチェックする。アクティブであれば、それが目的とするデータであるか否かは別としても、何らかのデータが存在するわけである。アクティブでなければ、データは存在しないわけで、そのエレメント変数を通して目的データを参照することはできない。

アクティブであれば次にそれがグループのメンバーとなっているか否かを調べる。グループのメンバーでない場合は、世代を示すキーが一致するか否かをチェックする。

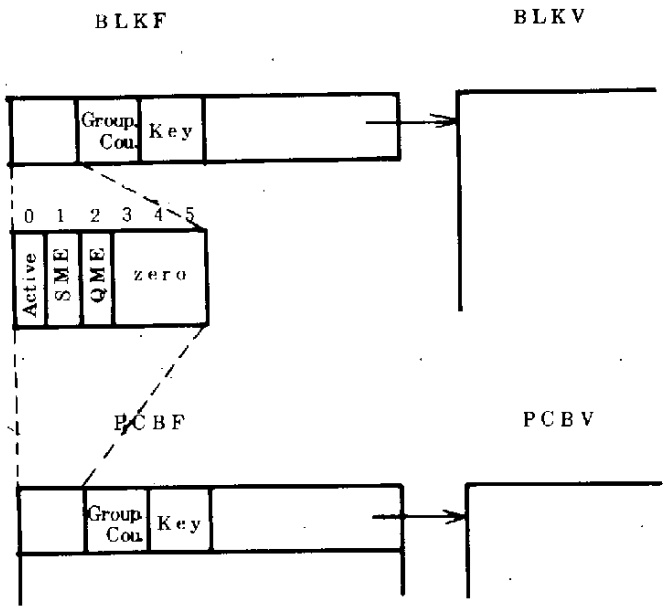


図6.3.1　キーの構造

PCBF又はBLKFのaccess

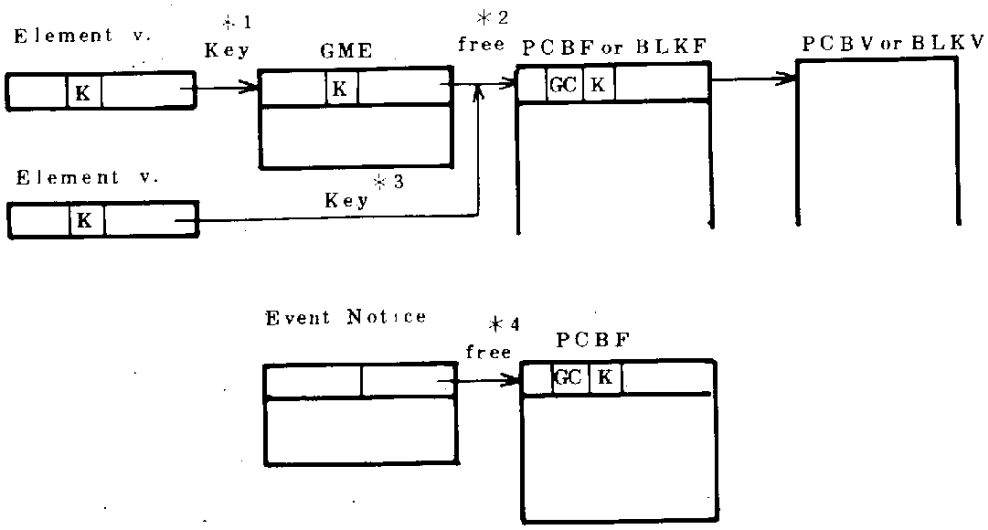


図6.3.2　プロセス／ブロックの参照

世代が一致しない場合はエラーとなる。

グループのメンバーである場合は、グループに属するメンバー名で間接的に参照する場合で、メンバーの世代を示すキーとの一致を確かめ、一致した場合は、メンバーから直接目的データを参照する。このようなチェックを毎回行なうことにより、目的とするデータは既に無くそのエリアを別のデータが占有している場合でも正しく処理が行なえるようになっている。

このようなチェックを可能にするために、プロセス又はブロックがグループに挿入されるたびにグループカウンターを＋1し、取り出すたびに－1し、プロセスやブロックの消去に際して、カウンターの値が零、即ちどのグループにも属さない場合以外はエラーとしている。

シミュレーションに於ては、そのダイナミックな振舞いは何らかの形でグループへのメンバの加入、脱退と関連を持つし、プロセスやブロックといったものは発生、消滅が頻繁に行なわれるので、SIMBOLが行なっているチェックが実行速度を低下させる事は否めない。そのためにチェックというコマンドが用意されており、チェックをしないで済ませるようランに先立って指定できるよう配慮されている。不安があるならば何の指定もしなければいい訳で、その場合には自動的にチェックが組み込まれる。

添字付変数に関するサイズオーバ或はアンダーのチェックや、計算型のジャンプ（TRANS）に於ける行先のチェック、DOステートメントに於けるキザミ幅のチェック等もオプションとなっており、実行速度を向上させたい、或はチェックは不要であるという場合には、これらのチェックも行なわずに済ませることができる。

ここで述べて来たようなチェックは、通常のテストではチェックできないようなエラー、頻繁には起らないがひとたび発生した場合には原因究明に大変な労力を必要とする為、ユーザがあえてチェックの取りやめを指定しない限りは自動的に行なうようになっている。

SIMSCRIPT－1.5の場合は配列の添字チェックをしていないがため

に、テンポラリーエンティティをセットから出そうとした時、セットへの登録が必ず行なわれているにもかかわらずメンバーが無いというエラーを起こしたりする。

添字つきのセットへの登録に際しサイズチェックをしていないがために全く関係ない場所にメンバーを登録してしまったりするわけである。或はデータ読み出し侵害としてアボートされてしまったりする。

サブルーチンへのパラメータの受け渡しを誤ったがために、オペレーティングシステムがエラーアボートした例を次表6.3.1～2に示す。この時は3回のラン結果が出力された後にアボートされており、1回限りで済ませておけば、エラーの発見は更に遅れていたであろう。

又このようなオペレーティングシステムが出すエラーだけでは、その原因究明も非常に困難であり、そのために種々のデータを出力させてもう一度ランさせたりしなければならない場合もあり、ランタイムのチェックの重要性が強調される由縁もこのあたりにあるのであろう。

表6.3.1　SIMSCRIPT－1.5 ランタイムエラー例－1

```
      SUBROUTINE  CHECK ( JCRT , IBJG )
NXT           IF  QUE(IBLG) IS EMPTY,GO TO EXIT
      IF  ENTRY ( JCRT ) GE BUFS ( JCRT ) , GO TO EXIT
            LET  ENTRY(JCRT)=ENTRY(JCRT) + 1
            CREATE  ENDSV
                 LET  NUMBR(ENDSV)=IBLG
      WRITE ON 61,IBLG,ENDSV
      FORMAT(*   CHECK*I10,010)
            CAUSE  ENDSV  AT  TIME + TRANS(IBLG)
EXIT      RETURN
      END

      ENDOGENDUS  EVENT  ENDSV
         LET  NUM=NUMBR(ENDSV)
              DESTROY  ENDSV
         REMOVE FIRST  MSG  FROM  QUE(NUM)
              LET  IND1=BLIMP(NUM)
      LET  X = ENTRY(IND1)
      LET  Y = QUEIN(NUM) - QINT(MSG)
      IF Y GR MAXWT(NUM), LET MAXWT(NUM) = Y
      LET  AMQUE = NQUE(NUM)
      LET  A11 = 1.
      ACC A11, AMQUE INTO UTIL(NUM) , AVGQ(NUM) SINCE QUEIN(NUM)
     1,QUEIN(NUM)
      ACC X , X INTO AVAIL(IND1) ,  AVGL(IND1) SINCE ENTM(IND1)
```

ENTRY	ADDRESS			REFERENCES
CMSG	000100	XMSGCL	000507	
DMSG	000101	XENDSV	001037	
		SRVST	003032	
GSORCE	000102			
PSORCE	000104	XMSGCL	000521	
FLQUE	000105	SRVST	003043	
RFQUE	000111	XENDSV	000577	
PMQUE	000271	XENDSV	000773	
		SRVST	003037	002777

表6.3.2　SIMSCRIPT－1.5 ランタイムエラー例－2

```
 02/27/73 *  C R C   SCOPE 3.3P  PSR 328 02/24/73


14.03.53.$ SIMS000 AT CTL.PT, 4 ---------- F600
14.03.53.$ 0060*10 CARDS READ. CR3G.
14.03.53.SIMS(CM50,T120,P4)
14.03.53.$IDA,113401,OZAWA.
14.03.54.COMMENT.UNREGISTER 0000
14.03.54.MIX(INPUT,COM)
14.03.55.REWIND(COM)
14.03.55.SIMS(I=COM)
14.04.25.COMPASS(I=MAPTP,L=0,B=SIMLGO)
14.04.48.  ASSEMBLY COMPLETE.    450008 SCM USED.
14.04.48.REWIND(SIMLGO)
14.04.48.SIMLGO(LC=40000)
14.05.16.CTL.PT.ERR.FLAG=02
14.05.16.ERROR MODE = (1)   ADDRESS =000271
14.05.17.FL CHANGES. OLD=50K,NEW=00K.
14.05.17.  TCP=00061,TPP=00009 SEC.   NR00344
14.05.17.   ST=00084
14.05.17.ELAPSED SYSTEM TIME= 001.1 MIN.
14.05.17.$ ACCT SYSTEM TIME= 001.1 MIN. (P=4*)
14.05.17.*** END OF JOB *** ERROR FLAG = 02

 ////////F600SIMS0100602116401          0011006100009142 OZAWA          730227
```

第7章　効　率

第７章　効　　率

7.1　メモリーの使用効率

コンピュータによるシミュレーションの大きな問題の１つはかなりの記憶容量を必要とする事である。

シミュレーションに於ては、テンポラリーなデータがダイナミックに発生・消滅を繰り返し、相互作用を及ぼし合うので作業用の豊富なメモリーが必要になる。

システムが提供するメモリー容量が少ないと、同時に存在するテンポラリーデータ数が制限される事になり大きなモデルが取り扱えないと云う事にもなる。

最近ではバーチャルメモリーの概念で二次記憶をも包含し容量を拡大する傾向も見受けられる。

これからのシミュレーションシステムは、積極的にこのような機能を導入してゆく事により、メモリー容量の制約に関して現在程には問題にされなくなってゆくかも知れない。

或いは又、必ずしも大型計算機でなくとも、例えばミニコンに依るシミュレーションさえ可能であるということになるべきかも知れない。

いずれにしろ、現状では計算機システムから来るメモリー容量の制約は大きな問題であり、領域の有効使用と云う事を速度を犠牲にしない範囲で考えてゆかなければならないであろう。

ＳＩＭＢＯＬでは、テンポラリーなデータ領域としてＦ２３０－６０のＬＣＭ（Ｌａｒｇｅ Ｃａｐａｃｉｔｙ Ｍｅｍｏｒｙ）を使い３２ＫＷ単位に最高１２８ＫＷ迄使用する事が出来る。

テンポラリーデータの代表的なものは、ＳＩＭＢＯＬ・ＳＩＭＵＬＡ ではプロセス、ＳＩＭＳＣＲＩＰＴではテンポラリーエンティティ、ＧＰＳＳ ではトランザクションである。

領域を占有するものは勿論これだけではなく、プログラム自身とか、グローバルデータ類も含まれるが、それらは一応スタティックに与えられシステムライフ中変化しないものであるからここでは問題にしない。

一般にテンポラリーデータは雪崩れ的に増加したり、過渡的に増加したりする性格のものであるから、モデルの動作特性を解析する上からも、そのような過渡状態から安定状態或は発散状態への移行の様子が捉えられるよう、容量が保証されている事が必要である。

テンポラリーデータが何組発生されるかと云う事もさる事ながら、結局は一時期に何組のデータが共存するか、しかもその最大共存予測数がいくつかで抜い得るモデルが限られてしまう訳である。

プログラム部分は小さいがテンポラリーデータを一時期に多数発生するタイプのモデルの場合には、短期間でシステムが用意した容量一杯を全て使い切ってしまう訳で、モデル自体の大きさと云うよりわ、扱うモデルの性格がむしろ問題となる場合が多い。

さて、SIMBOLに於けるテンポラリーデータであるプロセスのデータ部分は図 7.1.1 に示ような構造を有している。

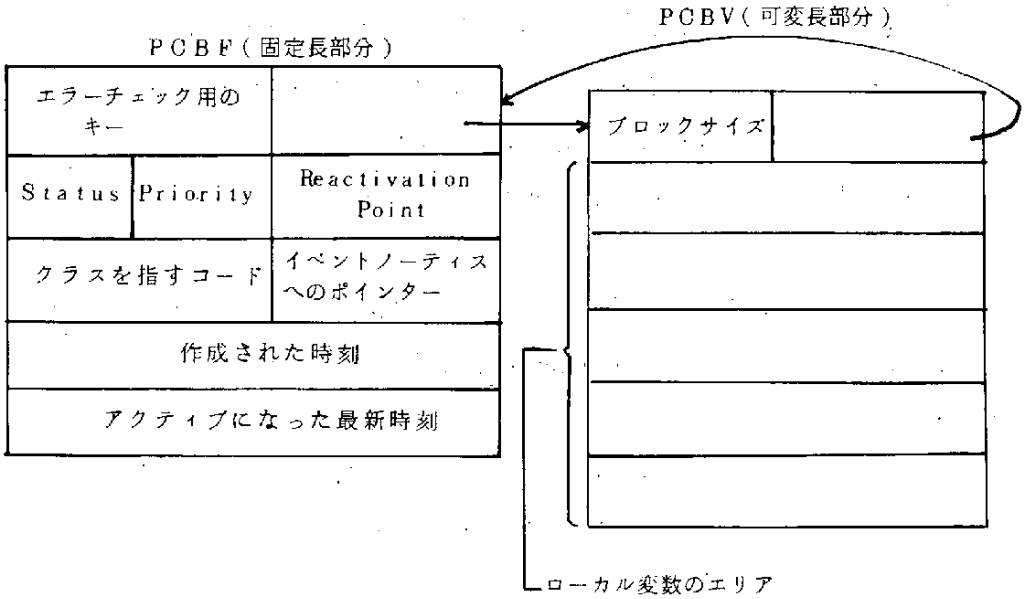


図 7.1.1　PCBの構造

プロセスはオブジェクトコード本体とデータ本体とに2分され、PCB (Process Control Block) に依って管理される。

この制御ブロックPCBは更に領域管理のためPCBF(固定長部分) とPCBV (可変長部分) とに2分されている。

ユーザが定義したデータそのものは、PCBVの領域に取られる。

プロセスがデータを持たなくとも少なくとも6語は、必要最小限のデータとして確保されてゆく。

SIMSCRIPT の様に時間の進行につれダイナミックに発生されるイベントとテンポラリーデータとが全く分離されているものでは、いくらか効率的にメモリーが使用出来る。

SIMBOLでは、ブロックと云うものを導入し、プログラム部分を持たないプロセスを効率的に扱うよう配慮している。

ブロックはBLOCK宣言により定義され、生成式に依り領域が確保される。

ブロックにはBCB (Block Control Block) が対応し、BCBVはPCBVと同じ構造であるが、BCBFの部分は単純で、PCBF の先頭一語で済む。

これによって、プロセス或はブロックで定義されるデータ個数をnとすると $(6+n)/(2+n)$ 倍の効率的な領域の使用が可能になっている。

$f(n)=1+4/(2+n)$ としてグラフに示すと図7.1.2 のようになり、平均テンポラリーデータ構成数がこの時に2倍の領域迄使用可能な事を示している。

10数語迄のデータ数でテンポラリーなデータブロックを構成する場合には、ブロックを用いる効果は大きい。

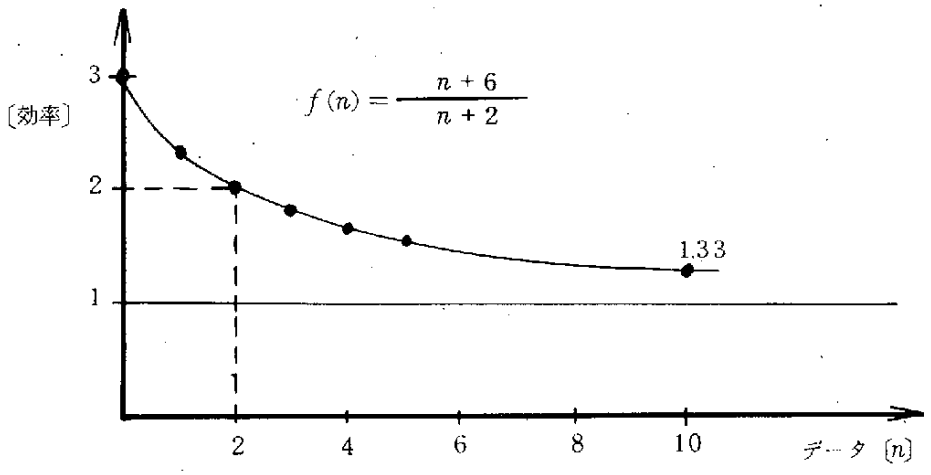


図7.1.2　メモリー使用効率〔プロセス、VS．ブロック〕

例として用いたネットワークモデルのように、発生局・目的局・現在局・使用回線・発生時刻と云うわずか5語のデータで済む様な場合、プロセスのデータとして扱った場合128K語そのままを使うものとしても1万個程度しか同時に扱い得ないのに反し、ブロックとして表現した場合2万個に近いものを同時に取り扱い得る事になる。

SIMBOLに於いてメモリー使用効率向上のもう1つのものは、セットやキューの扱いに見られるGME（Group Member Element)の導入である。

SIMSCRIPT に於ては、テンポラリーデータ本体が直接グループのメンバーに成る。

テンポラリーデータが同時には1つのグループのメンバーとしか成り得ないと云う場合は、非常に効率的にメモリーを使用出来る訳だが、同時に複数のグループのメンバーと成る場合（例えばAと云う家族に属し、Bと云う会社に通い、Cと云うクラブに加入し、Dと云う地方に住み、E…）、GPSSのようにトランザクションのコピーをSPLIT ブロックで何重にも作り出して処理するか、SIMSCRIPT のようにあらかじめテンポラリーエンティティに然るべきエリアを確保しておくと云った方法は効果的ではない。

データ本体を操作しなければならない事は、グループの扱いを鈍重なものとするのであろうし、あらかじめ作成しておく方法は、グループのメンバーと成らない場合でも無駄に領域を消費する事になる。

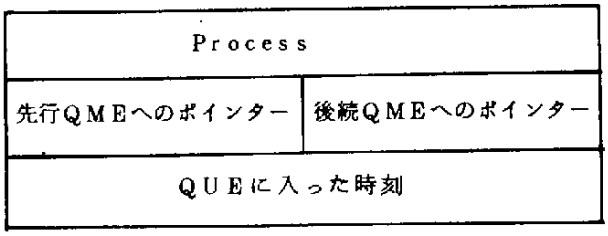
ＳＩＭＢＯＬやＳＩＭＵＬＡでは、グループへメンバーが加入する場合に始めて、メンバーエレメントを作成するように構成されている。

この様に必要性が生じて始めてメモリーを使用する事に依り、稀少資源であるメモリーの効率化を図っている。

図 7.1.3 にＳＩＭＢＯＬに於けるＧＭＥ の構造を示してあるが、一見して分かるように非常にシンプルな構造をとっている。

データ本体はこのＧＭＥから辿る事が出来、間接的にプロセスやブロックがグループのメンバーに成る訳である。

ＱＭＥ(Queue Member Element)



ＳＭＥ(Set Member Element)

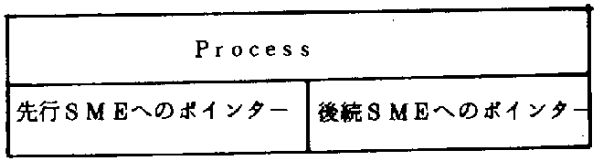


図7.1.3　ＧＭＥの構造

n 語のデータを持つプロセスが直接 m 個のグループに同時に所属する場合を考えてみると、ＰＣＢＦ の部分にグループの前後のメンバーを指し示

すために更に1語が必要となり、結局6m語のエリアが使われる事になる。

図7.1.4にGPSSのトランザクションのコアアロケーションを示す。

固定長部分は6語で構成されている。

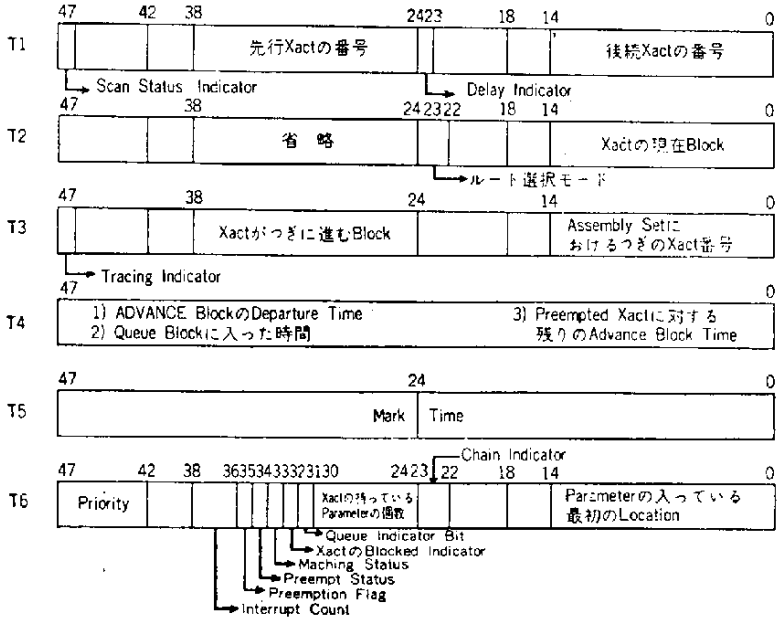


図7.1.4　GPSSのコア・アロケーション

SIMSCRIPTの場合には、1つのグループのメンバーとなるためには、Successor（後続）、Predecessor（先行）を示すポインターを定義する事が必要となる。

同時にm個のグループに所属する可能性がある場合は、テンポラリーエンティティにデータ本体の他に2m語の領域を定義しなければならない。

即ち2m＋n語がテンポラリーエンティティとして必要になる。

従ってグループを構成する場合には若干SIMSCRIPTの方がSIMBOL

よりも効率的にメモリーを使用する事になる。

しかし乍らテンポラリーデータを発生する度に2m語の領域が確保されて行く訳で、SIMBOLがブロックを使用した場合に比べ（2+n）／（2m+n）の効率でしかメモリーを使用出来ないと云う側面もあり、通常のシミュレーションに於ては、テンポラリーデータは相当量発生され、複数のグループに所属すると云う場合が多く、SIMBOLのメモリー効率化の効果は十分期待出来るであろう。

SIMBOLではインプリメントされている計算機がワードマシンである事から1語を再分割する様な使用は、システム側のルーチンしか行なっていないが、GPSSのハーフワードセイブバリュー、フルワードセイブバリューと云ったもののように、1語を1つ以上のデータで使用してメモリースペースの節約を図っているシステムもある。

F230−60に於ても1語の½分割、¼分割、ビット単位の分割等は可能であるが演算処理の命令が十分でなくなんらかの形で1語データと同じように変換し直す必要があり、処理効率を著しく低下させるおそれがあり採用していない。

SIMSCRIPTでは、GPSS以上にきめの細かい扱いが可能となっており、実数型のデータを扱わない場合には、非常に効率的にメモリーを使用する事が出来る。

1 語	ブランク または 1/1			
半 語	1/2		2/2	
3分の1語	1/3	2/3	3/3	
4分の1語	1/4	2/4	3/4	4/4

（3分の1語と4分の1語は整数のみ使用可能）

図7.1.5 SIMSCRIPT−1.5のPacking

図7.1.5は、SIMSCRIPT−1.5に於けるパッキングデータの構成である。

Field Packing Factor	Attribute Value Placement
1/2	first half of computer word
2/2	second half of computer word
1/4	first quarter of computer word
2/4	second quarter of computer word
3/4	third quarter of computer word
4/4	fourth quarter of computer word

Bit Packing Factor	Attribute Value Placement
n - m	bits n through m inclusive $1 \leq n \leq 32$ $1 \leq m \leq 32$ $n \leq m$

図7.1.6　SIMSCRIPT−Ⅱに於けるPacking

図7.1.6に示す様にSIMSCRIPT−Ⅱに於てはビット単位のパッキングも可能となっており、メモリーの効率化もさる事乍ら、又別の次元の可能性をも読み取る事が出来るであろう。

7.2 実行効率

シミュレーションランが1回限りで終了すると云う場合は極めて稀である。

パラメータを変えて見たり、乱数系列を変えたり、或はアルゴリズムを

変えてみたりして何度もランさせる事が一般的である。

シミュレーション実験は、通常の物理実験の場合に起こる様な実験誤差、即ち管理された因子或は類別された因子で説明し得ない変動分、により重要な因子の効果が隠されてしまうと云う事は稀であり、因子間の高次の相互作用が問題にされる場合が多い。

即ち、因子の水準の各組み合せに対してそれぞれ全然別個の乱数を用いて実験をしない限り、管理不可能な変動は存在しない事を前提に通常のシミュレーション実験は行なわれている。

しかし乍ら一般にシミュレーション結果は、母集団の標本点を与えるもので母集団の確率的な性格を明らかにするためには、何回かのサンプリングが必要になる。

母集団の特性を標本分布から推定しようとする場合には、母集団の特性を損ねない程度の標本点は少なくとも必要である。

母集団の平均や分散を区間推定する場合にも、求めようとする精度に応じてサンプリングサイズも多く取らなければならない。

母集団の平均値の区間推定を中心極限定理に基づいて行なう場合には、区間推定の精度は、シミュレーションランの繰り返し回数の平方根に比例する訳で、そのような場合にもラン回数は飛躍的に増大する訳である。

或はいくつかの代替案の比較を行なう場合であるとか、ラン結果に基づき最適化を図って行く場合であるとか、ＤＰに於ける資源配分のように何通りかの結果を必要とする場合であると云った建設的な方向に於ても多数回のランが必要になるのであろう。

もっと身近な問題として、ランタイムにエラーが発生した場合に修正を繰り返し乍ら何度も何度もランさせると云う事がある。

いずれにしろ、一回限りのランで済むと云う場合は例外中の例外であると思ってさしつかえないであろう。

そこで大きな問題となるのが実行速度であり、１回のランで１分程度の

ロスであっても数十回のランを行なえば数十分の差となって現われる訳で、特にSIMBOLの様にインタープリーティブに実行するシステムでは、実用性を保ち得るか否かと云った重要な問題となる。

SIMBOLでは実行速度を向上させるために多くの配慮が成されている。

例えば、その1つは中間コードを解釈しながら実行する形態を取らずに直接オブジェクトコードを作成する様なインクリメンタルコンパイレーションを行っている事である。

次の図7.2.1はオブジェクトコードブロックの構成を示している。

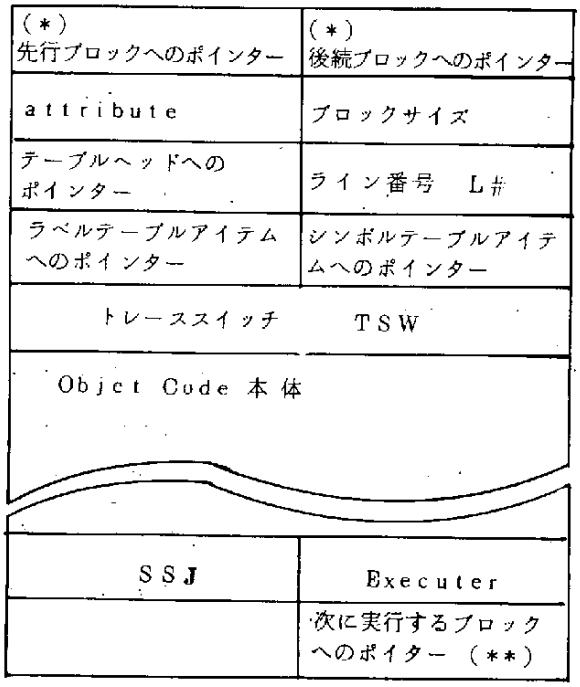


図7.2.1　Object Code Block

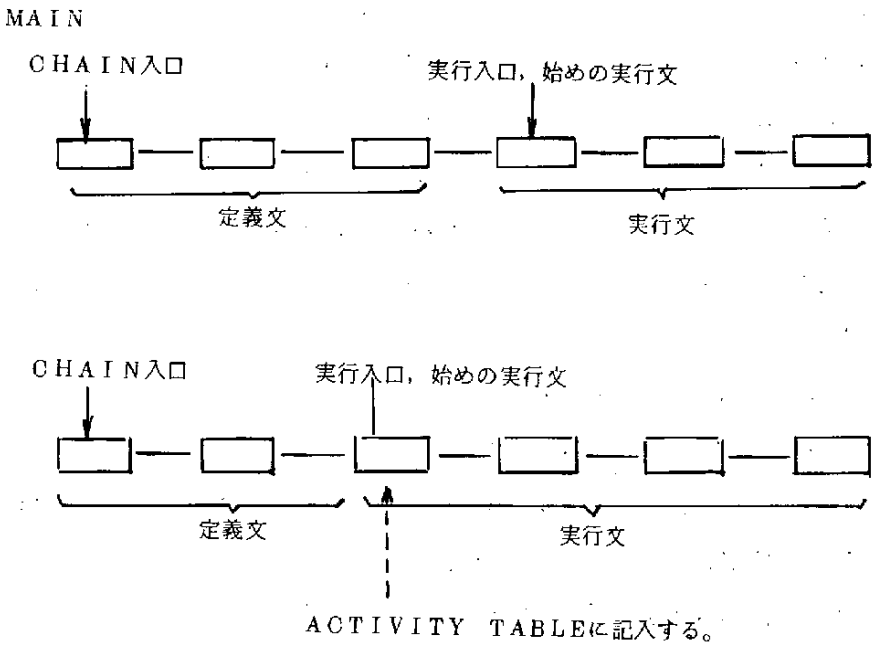


図7.2.2 Object blockのチエイン

図7.2.2 は各プログラム単位に於けるオブジェクトコードブロックの連鎖の様子を図示したものである。

図7.2.1 に於けるオブジェクトコード本体は、通常のコンパイラーが作成するものと大差はない。

オブジェクトの実行は1ステートメント単位にEXECUTER が介入して行なうが、1ステートメントのオブジェクト実行に費す時間は、本体だ

けに限って云えばコンパイルされたものとほとんど変わらない。

オブジェクト本体の実行後にFASP(F230−60のアセンブリー言語)のJump命令SSJを実行してEXECUTERに戻る事により、次に実行すべきオブジェクトブロックが自動的に示されるように成っているので、オブジェクトコードブロックをサーチする必要はなく効率的に次のブロックを実行し得る。

問題になるのはトレーススイッチであるが、それを次に述べよう。

SIMBOLのEXECUTERは順次オブジェクトコードブロックを取り上げてそこに制御を渡し乍ら実行して行く訳だが、時間処理に関するオブジェクトであればそのオブジェクトの中からランタイムルーチンが呼ばれ制御はSCHEDULERに渡される。

SCHEDULERが取り上げたプロセスを実行する場合には、制御は又EXECUTERに戻される。

EXECUTERは各オブジェクトブロックの実行に先立ちまずトレーススイッチがセットされているかどうかを調べる。

これは2命令の実行で済む。従って、これに関するEXECUTERのオーバーヘッドはほとんど問題にならない。

SIMBOLに於けるトレーススイッチは種類が豊富であり、実際にトレースを行なう場合にはその処理のために全体の実行効率が低下する訳だが、それはやむを得ないであろう。

トレースの複雑さにもよるが、平均的に見て50～100ステップの命令が1回のトレースに対し実行される。

GPSSなどバッチシステムの場合と異なりトレースの開始や終了をランタイム中でも自由にコントロール出来、トレース範囲や、対象を制限する事も簡単に行なえる。特に又、その結果を随時確認出来るようになっているので、目的に合わせた重点的なトレースを効率よく行なう事が可能である。

又ＳＩＭＢＯＬの場合のトレースはオンラインモニタリングの性格も兼ねており、ユーザがリアルタイムに反応し得る時間内に結果を表示する事が要請されるが、その点ではまず支障はない。

配列の添字等に関するランタイムのチェックも実行速度を低下させる原因となるが、シミュレーションの実行に先立ってチェックを取りはずす事が出来るようになっているので、チェックをはずす事自体に問題が無い限りは実行効率は保証される。

オブジェクトコード自体の効率化も十分考慮して、可能な限りオプティマイズされたオブジェクトを作成している。

ＤＯステートメントのキザミの処理に関しては、多少ステートメントの機能を犠牲にしてもかまわないと云う方向で効率化を図っている。

例えばキザミ幅をランタイムに変更してゆくと云う機能は削除してしまったが、この機能はランタイムのエラーチェックを必要とするので、多数回繰り返して実行されるＤＯステートメントの性格から、かなり処理効率に影響すると判断したためである。

シミュレーションに於ける処理は、時間軸を中心に展開される訳で、スケジューリングテーブル内の操作はポインターのつなぎ替えなどのリスト処理が中心である。

コントロールフローの章で述べている様に、シミュレーションシステムにはスケジュールテーブルがあり、トランザクションにしろイベント或はプロセスにしろ、この時間表の中に盛り込まれない限り実行の対象とはならい。

時間表への登録はスケジュール用のステートメントを用いて行ない、時間の進行によるシステム状態の変化に従ってテーブルの内容も変わってゆく。

既に作成済みの時間表も計画変更があればリスケジュールし直さなければならないし、割り込みがあれば計画を後ろに延期しなければならない。

計画が実行に移されれば、その計画はもはや時間表に取っては不要なものであるから取り除く必要がある。

この様にスケジュールテーブルに於ける登録アイテムは頻繁に並べ換えが行なわれる。

アイテムの登録・取りはずしには必ずポインター操作が付随する。

ＳＩＭＢＯＬではスケジュールテーブルへの登録アイテムは対象リスト構造をなしているので、先行メンバーと後続メンバーの両ポインターを処理しなければならない。

図 7.2.3 に、このポインター処理の例を示す。

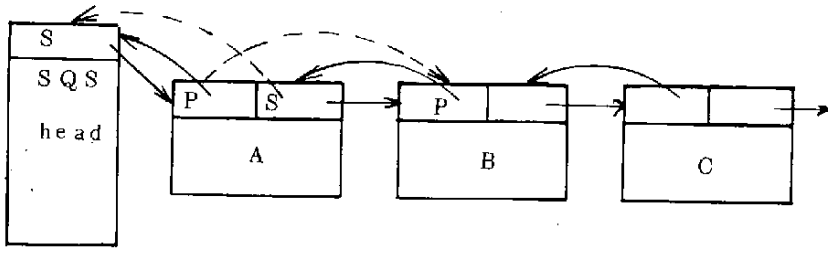


図 7.2.3　スケジュール・テーブルに於けるリスト処理

図 7.2.3 のＳＱＳの先頭アイテムＡを計画表からはずす場合には、Ａの後続ポインタＳの示すアイテム（図ではＢ）のポインターＰをＡのＰで置き替え、Ａの先行ポインターＰの示すアイテム（図ではＳＱＳの head）のポインターＳをＡ自身のＳで置き替える作業が必要である。

アイテムの取りはずし登録には、対象とするアイテムのアドレシングと前後アイテムへの自分のポインターのコピーと云う操作だけで済むが、実際にはサーチと云う作業が必要である。

何時間後に計画を予定すると云う表現は、スケジュールテーブル上での登録順序に変換される必要がある。

スケジュールテーブルに登録されているアイテムが多いとサーチにも時間がかかる訳で、何らかの工夫が必要であろう。

勿論対象モデルの性格に大きく依存するので、一概に最適なスケジューリングメカニズムを決め得ないが、例えばスケジュールテーブルを分割し、ハッシュテクニークを用いる事に依り効率の向上をはかる事が可能であろう。これについては現在検討中である。

検討段階であり詳細について述べる事は出来ないが、その概略は以下の通りである。

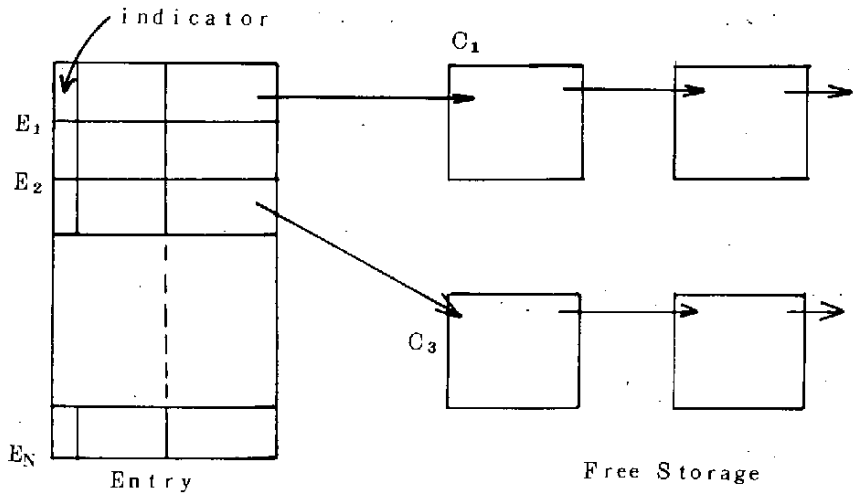


図7.2.4　スケジュール テーブルの構造

図7.2.4 に示すようにスケジュールテーブルを構成するチェインを1本ではなくN本に分割し、それぞれにチェインの入口を設ける。

新たに追加するアイテムがどのエントリーのチェインに対応するかは、その時間属性から簡単な計算で求められる様にする。

挿入すべきチェインが見つけられた後は、従来の方法に依り、つまりチェインの先頭から順に調べて挿入場所を決める。

この方法に依り、チェインをスキャンする範囲が狭まるので、挿入のためにサーチする時間の短縮が予想される。

但し、エントリーの設け方即ち時間属性の分割の仕方により効率は大きく影響されるものと思われる。

最後に実行効率に大きく作用するものとしてフリーストレージの管理法を検討する。

テンポラリーなデータは、全てフリーストレージを使用するが、この領域自身無限大の容量を持つ訳ではなく、完全に使い切ってしまえばそれ以上シミュレーションを続行する事は出来ない。

テンポラリーデータの発生・消滅が頻繁に繰り返えされる事に依り、フリーストレージは蜂の巣状に空きエリアが発生してゆく。

SIMBOLでは、処理の効率化のために、テンポラリーデータを固定長部分と可変長部分に分割したり、チェインを構成するものはヘッドを設けたりしている。

可変長部分のフリーストレージ上での空きエリアは勿論、新たなエントリーのために積極的に使用する訳だが、先頭から完全な形ですきまなく使用される事は考えていない。

SIMBOLに於けるテンポラリーデータは、あらかじめパターンが明らかにされ何語で構成されるものであるかを知る事が出来るので、フリーストレージの空きエリアの管理にこのパターンを結びつけて効率化を図っている。

アクティビティが定義されれば、それから発生するプロセスのデータ構成はアクティビティ内の変数定義から明らかにされる訳である。

図7.2.5 にフリーストレージ空きエリア管理の様子を示す。

テンポラリーデータのパターンに対応してエントリー $E_1 \sim E_N$ を設ける。

m 語構成のテンポラリーデータが必要になった場合、まず対応するエントリーのチェインを調べる。

そのエントリーに連鎖される空きエリアが無い場合は、フリーストレージの次の使用可能なエリアを示すNEXTにより新たにエリアを確保してゆく。

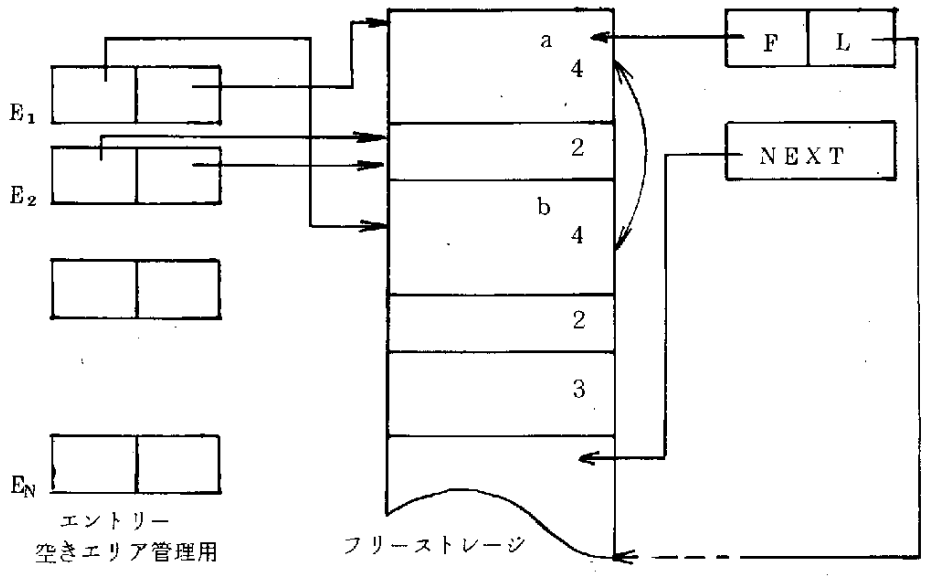


図7.2.5 フリーストレージ管理

対応エントリーに空きエリアのチェインがある場合は、（図7.2.5 のエントリー E_1 に対応するチェイン a, b）先頭の空きエリア（図の a）から使用して行く。

この時使用に先立って先頭空きエリアの後続ポインターの、エントリーへのコピーを行なっておく。

使用していたエリアがフリーにされた場合には、対応するエントリーのチェインの最後に連鎖してゆく。

この様な形で使用してゆき、n 語構成のデータエリアを確保したいが NEXT 以降のエリアでは不十分であり、且 n 語に対応するエントリーのチェインも存在しないと云う状況下で始めて、ガーベージコレクションが検討される。

そして空きエリアをチェインするエントリーが存在する場合に限って、ガーベージコレクションを行なう。

ガーベージコレクションに費す時間は、その時のフリーストレージの状態にも関係する訳で一概に論ぜられないものではあるが、各アイテム平均10語、128K語のエリア80％使用、と云う状況下では、各アイテムのデータ転送とポインターつなぎ替えに要する平均時間は1ms 以下であり、全体の時間見積りにして10数秒程度である。

これが頻繁に行なわれてしまうのでは、全く実用性を損ねてしまう訳だが、この種の状況が発生する確率は低いものと想定される。

何故なら、定常状態に於けるフリーストレージの使用状況は、エントリーヘッドからチェインされている空きエリアをサイクリックに使用している事が予想され、それ以上新たにNEXT を求めてエリアを確保していかなければならない状態ではないからである。

NEXT から新しくエリアを確保して行く状況は過渡状態に於て見られるが、これに依りガーベージコレクションが必要とされ、且その結果有効エリアが確保出来ない場合には、（この様な状況は、余程大きなモデルを取り扱うか或はプログラムミス例えば発生したデータで不要になったものを消去しないと云った場合にしか起らない）、シミュレーションを中止せざるを得ない。

実行効率を高めるための配慮は、実際効果的なものであり、例題モデルの実行に対しては、ガーベージコレクションは1度も起らなかった。

第8章　まとめ

第8章　まとめ

これまでの各章では、オンラインシミュレータおよび各種バッチ処理シミュレータの詳部に渡って検討して来た。最後に、これらをまとめる意味で、SIMBOL, SIMULA, GPSS, SIMSCRIPT との比較を表形式で整理し、表8.1以下に示してある。尚表中でGPSS, SIMSCRIPTについては、同じ系列のシステム、例えばGPSS－Ⅱ, GPSS－Ⅲ, GPSS－Ⅴ等で共通する性質とおもわれる場合には単にGPSSと表示し、特に異る場合のみGPSS－Ⅴの様に記してある。

又、性能の優劣を示すのにA, B, Cの記号を用い、この順に良さを表わしている。

表8.1　シミュレーション言語の機能比較　I

	SIMBOL	SIMULA	GPSS－V	SIMSCRIPT 1.5	SIMSCRIPT Ⅱ
言語の記述性	A	A	B	B	A ＊1
モデル化の容易さ	A	A	A	B	B
汎用言語との関係	どちらかと言えば FORTRAN 系	ALGOLを含む	無関係	FORTRAN 系	PL／I 系
プロヤッサの形式	インタープリター ＊2	コンパイラー	インタープリター	コンパイラー	コンパイラー

＊1　SIMSCRIPT Ⅱは SIMSCRIPT 1.5より機能が拡張されているが，シミュレーションを行う為の機能よりは，むしろシステム記述言語としての機能の拡張が中心になっている。

＊2　一部にコンパイラーとしての機能を含んでいる。

表8.2　統計データの収集機能の比較

	SIMBOL	SIMULA	GPSS-V	SIMSCRIPT
自動的にカウントするデータの有無	有　り	無　し	有　り	無　し
ユーザーが収集する為のサポートが充分に行われているか？	A	A	C	C

注）データ収集が自動化されている事は単に便利になっただけでなく、収集ミスの防止とか、モデルの組み方の簡易化に役立っている事を見逃せない。

表8.3　レポート作成機能の比較

	SIMBOL	SIMULA	GPSS-V	SIMSCRIPT
システムが自動的に作成してくれるレポートの　有・無	有　り	無　し	有　り	無　し
レポートジェネレータの　有・無	無　し	無　し	無　し	有　り
グラフ表示機能	有　り	有　り	有　り	無　し

表8.4　ステータス参照機能の比較

	SIMBOL	SIMULA	GPSS	SIMSCRIPT
モデルの実行中にステータスを参照できるか？	可	不　可	＊1 一部可	不　可
タイミング	随　時	—	随　時	—
参照出来る情報	変数の内容，システムサイドの情報など多種	—	時刻などの限られた情報だけ	—

＊1　一部のGPSSでだけ、コンソールからクイットを掛けて、ステータスが参照できるようになっている。

表8.5　ランタイム時のインタラクション機能の比較

	SIMBOL	SIMULA	GPSS	SIMSCRIPT
モデル実行時にインタラクションがとれるか？	可	不　可	一部のシステムで　可	不　可
タイミング	随　時	—	随　時	—
方　　法	端末からのクイット	—	コンソールからのクイット	—

表8.6　トレース機能の比較

	SIMBOL	SIMULA	GPSS	SIMSCRIPT
トレース機能の有・無	有　り	有　り	有　り	無　し
種　　類	豊　富	豊　富	限られている。	—
操 作 性	A トレース開始，終了，範囲の指定が自由に変えられる。	C プログラム内に組込んで行う必要がある。	C	—

表8.7　デバッグ機能の比較

	SIMBOL	SIMULA	GPSS	SIMSCRIPT
システムの特別なサポート	有　り	有　り	有　り	無　し
デバックの難易	A	B	A	C

表8.8　シミュレーション実験に関する比較

	SIMBOL	SIMULA	GPSS	SIMSCRIPT
モデルの実行開始や終了の指示方法	端末からの指示	コントロールカード 終了はプログラム内部で指示	コントロールカード 終了はプログラム又はコントロールカードで指示	コントロールカード 終了はプログラムで指示
繰り返し実行のし易すさ	A	B	B	B
シミュレーション途中のパラメータ変更	可	不　可	不　可	不　可
ステータスのイニシヤライズ機能	有　り	無　し	有　り	有　り
イニシヤライズの方法とタイミング	端末から 随　時	—	コントロールカード 実行の区切り	コントロールカード 実行の区切り
実行途中のステータスセーブ機能とリスタート機能の有・無	有　り	無　し	有　り	無　し
異るランの間での情報受け渡し可・不可と容易さ	可　能 A	可　能 C	可　能 A	可　能 C

表8.9 効率比較

	SIMBOL	SIMULA	GPSS	SIMSCRIPT
モデルの実行スピード	*1 C	B	B	A
メモリー使用効率 *2	C	B	C	A

*1 SIMBOLはインクリメンタルコンパイルを行っている関係で、オブジェクトプログラムの効率はいくぶん落ちている。シミュレーションでは一般にスケジューリングや、テンポラリーデータの取扱いなどのリスト処理が、モデル実行時間の大半を占め、この部分はバッチ処理の場合とまったく同じ条件なので、全体的に見た実行スピードはそれほど低下しない。

*2 SIMBOLやGPSSの場合、モデルプログラムだけでなくインタープリタータイプ故にプロセッサー自身も同居しているのでその分だけメモリーの使用効率が落ちてしまう。

表8からも明らかな様に、オンライン化した事によるメリットは次の点である。

モデルに対する連続的な介入操作が可能となったため、実行中のモデルのステータスが簡単に参照出来る。

トレース指示が自由に行なえるし、モニタリング機能と相まって強力なデバッキングエイドを提供出来る。

シミュレーション実験と呼称される、繰り返しモデルを実行する際のパラメータ変更、リイニシャライズ、異なるラン間での情報の受け渡し、実行中のモデルのセイブと云った操作性が向上し、シミュレーションが行ない易い。

更に従来ネックになっていたモデルの作成とテスト間の支障が取り除かれ、シミュレーション実験全体のパーフォーマンスが向上する。

デメリットは実行速度の若干の低下とメモリースペースの若干の低下が大きなものである。

しかし乍ら、ユーザ自身の思考と調歩的にシミュレーションを行なう目的のためには実行速度の低下は問題にならないであろう。

又メモリースペースにしても取り扱えるモデルは、バッチシミュレーションステムと比べても遜色ないと云える。

評価作業を行なっていて痛感したのは、機能と性能が十分でなければならない事は勿論であるが、ユーザサポート体制がどこ迄整っているのかと云う事であった。

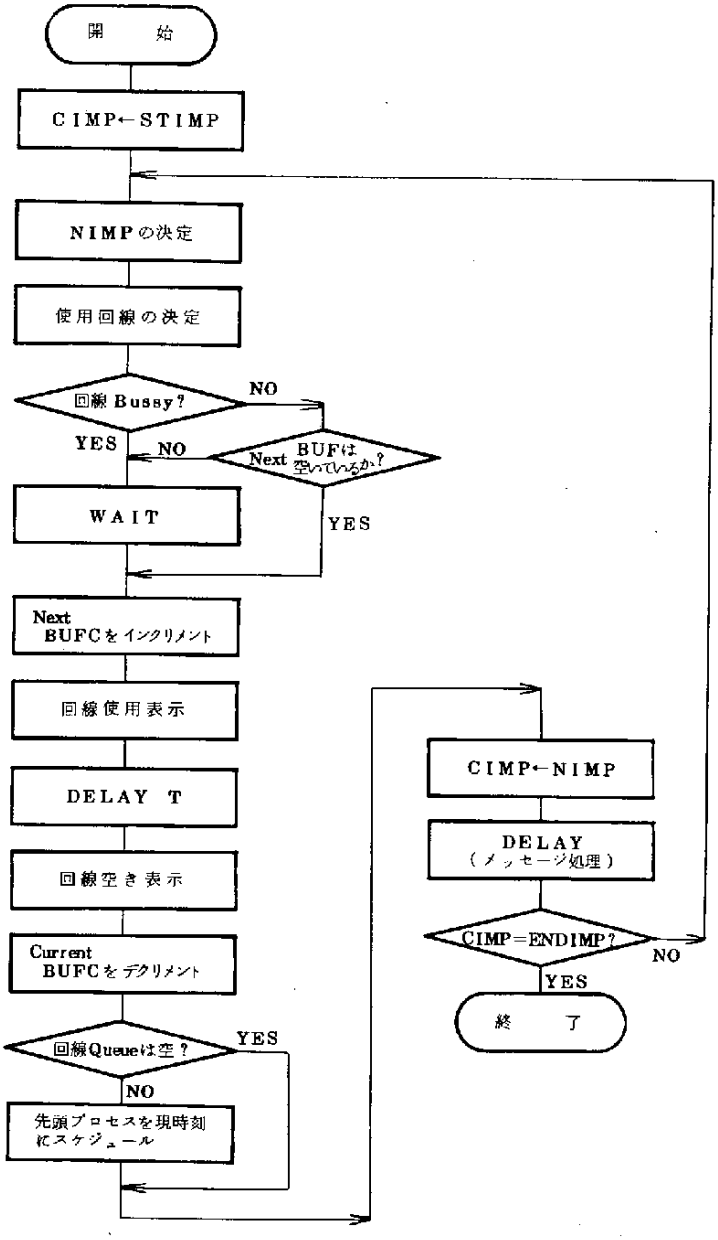
特にSIMBOLの様にキャラクターデイスプレイを使用している場合には、シミュレーションと本質的な関わりを持たない様な部分の問題、画面表示の機能であるとか、スクローリングの問題、ハードコピー、或はキーインを単純化するファンクションキーの機能と云ったサービスが十分行なわれる必要がある。

モニタリングにしてもプロセデユアも用意しており、必要に応じて拡張出来るようになっているが、もっと整備すべきであろう。

附　　録

1. プロセスタイプ（Ｉ）

付表1.1、1.2、1.3は、それぞれ付図1のフローチャートに基づくSIMBOL、SIMULA−65、SIMULA−67のプログラムリストである。



付図1.（メッセージ中心のプロセスオリエンディッドなフロー）

付表1.1　SIMBOL ソースリスト

```
0010 INTEGER RMATRIX(10,10),LMATRIX(10,10);
0020 INTEGER LSTATE(28),BUFC(10);
0030 INTEGER U1,U2,U3,I,J,K,MS,BSIZE;
0040 REAL MDIST(10,10),ORGDIST(20);
0050 REAL X,Y,ENDIMP,LSPEED,TRANSTM,ARRIVT;
0060 QUEUE LQUE(28);
0070 START GENERATE;
0080 END;
```

```
0010 ACTIVITY GENERATE;
0020 NEWMESG. LET X=UNIFORM(0.0,1.0,U1);
0030 LET K=1;

0040 DO L1 FOR I=1,10;
0050 LET K=I;
0060 IF ORGDIST(I) > X THEN BRANCH L2;
0070 L1. LOOP;
0080 L2. LET X=UNIFORM(0.0,1.0,U2);
0090 LET J=1;
0100 DO L3 FOR I=1,10;
0110 LET J=I;
0120 IF MDIST(I,K) > X THEN BRANCH CREATE;
0130 L3. LOOP;
0140 CREATE. IF BUFC(K) >= BSIZE THEN BRANCH L4;
0150 LET BUFC(K)=BUFC(K)+1;
0160 SCHEDULE NEW MESSAGE(CURIMP=K,ENDIMP=J) DELAY 0;
0170 L4. DELAY NEGEXP(1.0/ARRIVT,U3);
0180 BRANCH NEWMESG;
0190 END;
```

```
0010 ACTIVITY MESSAGE;
0020 INTEGER CURIMP,ENDIMP;
0030 INTEGER NXTIMP,USLINE,LN;
0040 LOOP. LET NXTIMP=RMATRIX(ENDIMP,CURIMP);

0050 LET USLINE=LMATRIX(NXTIMP,CURIMP);
0060 IF LSTATE(USLINE) # 1 THEN BRANCH LAB1;
0070 IF BUFC(NXTIMP) >= BSIZE THEN WAIT -> LQUE(USLINE);
0080 LAB1. LET BUFC(NXTIMP)=BUFC(NXTIMP)+1;
0090 LET LSTATE(USLINE)=1;
0100 DELAY TRANSTM;
0110 LET LSTATE(USLINE)=0;
0120 LET BUFC(CURIMP)=BUFC(CURIMP)-1;
0130 IF LN # 0 THEN BRANCH ACTM;
0140 IF LSTATE(LN) # 0 THEN WAKE <- LQUE(LN) DELAY 0;
0150 ACTM. IF EMPTY(LQUE(USLINE)) THEN BRANCH LAB2;
0160 IF BUFC(NXTIMP) >= BSIZE THEN BRANCH LAB2;
0170 WAKE <- LQUE(USLINE) DELAY 0;
0180 LAB2. LET CURIMP=NXTIMP;
0190 DELAY 1;
0200 LET LN=USLINE;
0210 IF CURIMP /# ENDIMP THEN BRANCH LOOP;
0220 LET BUFC(CURIMP)=BUFC(CURIMP)-1;
0230 END;
```

付表1.2　SIMULA－65

```
BB,ALG,ISCD    SIMS,SIMRB
CYCLE  000  COMPILED BY  12B4  0005  ON 03/27/73 AT 14:51:09
      B1
 1            BEGIN
      B2   S1   L1
 2            SIMULA  BEGIN
 3
 4            COMMENT ********************************************************
 5                    *                                                      *
 6                    *       THE SIMULATION MODEL OF COMPUTER NETWORK       *
 7                    *                                                      *
 8                    ********************************************************;
 9
10            INTEGER ARRAY  RMATRIX(1:10,1:10);
11            INTEGER ARRAY  LMATRIX(1:10,1:10);
12            REAL ARRAY  MD(ST(1:10,1:10),ORGD(ST(1:10);
13            INTEGER ARRAY LSTATE(1:28),BUFC(1:10);
14            INTEGER U1,U2,U3,I,J,K,MS,MESC,REGC,BSIZE;
15            REAL X,Y,ENDTIME,LSPEED,TRANSTM,ARRIVT;
16            REAL ARRAY STLQL(1:28),STTM(1:10),STLUS(1:28),STLTM(1:28);
17            REAL ARRAY STCUS(1:10),STLQT(1:28);
18            REAL ARRAY B1(1:10);
19            INTEGER ARRAY BZ(1:10),B3(1:10);
20            INTEGER ARRAY M1(1:1),MZ(1:1),M3(1:1);
21            INTEGER ARRAY LCNT(1:28),TCNT(1:10);
22            SET ARRAY LQUE(1:28);
23            FORMAT     EJK(E);
24            FORMAT     F1(X10,'ROUT MATRIX',A1,I);
25            FORMAT     F2(10I10,A);
26            FORMAT     F3(X10,'MESSAGE DISTRIBUTION',A1,I);
27            FORMAT     F4(10D10.4,A);
28            FORMAT     F5(X10,'INPUT PARAMETER',A1,I);
29            FORMAT     F6(3I10,D10.3,A);
30            FORMAT     F7(X10,'LINE MATRIX',A);
31            FORMAT     F8(X10,'CORE UTILIZATION',A5,5);
32            FORMAT     F9(X10,'IMP NUMBR:',I5,X10,'MEAN USAGE:',D10.2,' %',
33                       X10,'MESSAGE:',I10,A);
34            FORMAT     F10(X10,'LINE STATISTICS',A5,5);
35            FORMAT     F11(X10,'LINE NUMBER:',I5,X10,'MEAN USAGE:',
36                  D10.2,' %',X10,'MEAN QUEUE LENGTH:',D10.2,I10,A);
37            FORMAT     F13(X10,'SIMULATION RESULT',A10,5);
38            FORMAT     F14(X10,'BUFFER SIZE=',I5,X10,'LINE SPEED=',D10.1,
39                            ' B/S',X10,'ARRAIVAL RATE:'D10.2,' SEC(MEAN)',A3,3);
40            FORMAT     F15(X10,'SIMULATION END TIME=',D10.2,A);
41            FORMAT     F16(X10,' TOTAL MESSAGECOUNT:',I10,X10,
42                            'REJECT MESSAGE COUNT:',I10,A);
43
44            COMMENT ********************************************************
45                    *                  ACTIVITY MESSAGE                    *
46                    ********************************************************;
47
      S2    L2
48            ACTIVITY  MESSAGE(CURIMP,ENDIMP);
49                 INTEGER CURIMP,ENDIMP;
      B3
50                 BEGIN
51                 INTEGER NXTIMP,USLINE,LN,MMC;
```

```
52                  REAL LFT;
53                  LFT:=TIME;
54          LOOP:   NXTIMP:=RMATRIX(ENDIMP,CURIMP);
55                  USLINE:=LMATRIX(NXTIMP,CURIMP);
56                  ICNT(CURIMP):=ICNT(CURIMP)+1;
57                  MMC:=MMC+1;
58                  IF LSTATE(USLINE) EQL 1 THEN GO TO QUEUE;
59                  IF BUFC(NXTIMP) LSS BSIZE THEN GO TO TRANS;
60          QUEUE:  MS:=CARDINAL(LQUE(USLINE));
61                  ACCUM(STLQL(USLINE),STLQT(USLINE),MS,0);
62                  WAIT(LQUE(USLINE));
63          TRANS:  ACCUM(STCUS(NXTIMP),STTM(NXTIMP),BUFC(NXTIMP),0);
64                  BUFC(NXTIMP):=BUFC(NXTIMP)+1;
65                  LSTATE(USLINE):=1;
66                  LCNT(USLINE):=LCNT(USLINE)+1;
67                  ACCUM(STLUS(USLINE),STLTM(USLINE),0,0);
68                  HOLD(TRANSTM);
69                  LSTATE(USLINE):=0;
70                  ACCUM(STLUS(USLINE),STLTM(USLINE),1,0);
71                  ACCUM(STCUS(CURIMP),STTM(CURIMP),BUFC(CURIMP),0);
72                  BUFC(CURIMP):=BUFC(CURIMP)-1;
73                  IF LN EQL 0 THEN GO TO ACTM;
    B4
74                  IF LSTATE(LN) EQL 0 THEN BEGIN
75                  ACTIVATE FIRST(LQUE(LN)) DELAY 0.0;
76                  ACCUM(STLQL(LN),STLQT(LN),CARDINAL(LQUE(LN)),0);
77                  REMOVE(FIRST(LQUE(LN)));
    E4
78                  END;
79          ACTM:
80                  ACCUM(STCUS(CURIMP),STTM(CURIMP),BUFC(CURIMP),0);
81                  IF NOT EMPTY(LQUE(USLINE)) AND BUFC(NXTIMP) LSS BSIZE
82                  THEN
    B5
83                      BEGIN ACTIVATE FIRST(LQUE(USLINE)) DELAY 0.0;
84                      MS:=CARDINAL(LQUE(USLINE));
85                      ACCUM(STLQL(USLINE),STLQT(USLINE),MS,0);
86                      REMOVE(FIRST(LQUE(USLINE)));
    E5
87                      END;
88                  CURIMP:=NXTIMP;
89                  HOLD(0.1);
90                  LN:=USLINE;
91                  IF CURIMP NEQ ENDIMP THEN GO TO LOOP;
92                  ACCUM(STCUS(CURIMP),STTM(CURIMP),BUFC(CURIMP),0);
93                  BUFC(CURIMP):=BUFC(CURIMP)-1;
94              LFT:=TIME-LFT;
95              MMC:=MMC+1;
96              HISTO(H1,B1,LFT,1);
97              HISTO(H2,B2,MMC,1);
    E3   F2
98              END;
99
100         COMMENT ********************************************************
101                 *                MAIN PROGRAM START                    *
102                 ********************************************************;
103
```

```
104             COMMENT ********** PARAMETER READ **********;
105
106             READ(BSIZE,LSPEED,ARRIVT);
107             READ(U1,U2,U3,ENDTIME);
108             READ(RMATRIX);
109             READ(LMATRIX);
110             READ(ORGDIST);
111             READ(MDIST);
112
113             COMMENT ********** INPUT PARAMETER DISPLAY *****;
114
115             WRITE(EJK);
116             WRITE(F5);
117             WRITE(F6,U1,U2,U3,ENDTIME);
118             WRITE(F14,BSIZE,LSPEED,ARRIVT);
119             WRITE(F1);
120             WRITE(F2,RMATRIX);
121             WRITE(F7);
122             WRITE(F2,LMATRIX);
123             WRITE(F3);
124             WRITE(F4,ORGDIST);
125             WRITE(F9,MDIST);
126
127             COMMENT ********** INITIALIZE **********;
128
129             TRANSTM:=1024.0/LSPEED;
130             FOR I:=1 STEP 1 UNTIL 10 DO
131             B2(I):=B3(I):=I;
132             FOR I:=1 STEP 1 UNTIL 10 DO
133             B1(I):=5.0+I;
134             FOR I:=1 STEP 1 UNTIL 10 DO
135                 BUFC(I):=0;
136             FOR I:=1 STEP 1 UNTIL 28 DO
137                 LSTATE(I):=0;
138         NEWMESG: X:=UNIFORM(0.0,1.0,U1);
139             K:=1;
140             FOR I:=1 STEP 1 UNTIL 10 DO
    B6
141             BEGIN  K:=I;
142                 IF ORGDIST(I) GTR X THEN GO TO ENDTERM;
    E6
143             END;
144
145             COMMENT ******* GENERATING NEW MESSAGE **********;
146
147         ENDTERM: X:=UNIFORM(0.0,1.0,U2);
148             J:=1;
149             FOR I:=1 STEP 1 UNTIL 10 DO
    B7
150             BEGIN  J:=I;
151                 IF MDIST(I,K) GTR X THEN GO TO CREATE;
    E7
152             END;
153         CREATE:  MESC:=MESC+1;
154             HISTO(H3,B3,K+1);
155             IF BUFC(K) GEQ BSIZE THEN REGC:=REGC+1
    B8
```

```
156             ELSE  BEGIN
157             ACCUMISTCUS(K),STTM(K),BUFC(K),0);
158             BUFC(K):=BUFC(K)+1;
159             ACTIVATE NEW MESSAGE(K,J) DELAY 0.0;
   E8
160             END;
161             HOLD(NEGEXP(1.0/ARRIVT,U3));
162             IF TIME LSS ENDTIME THEN GO TO NEWMESG;
163
164             COMMENT ********************************************************
165                       *          GENERATING SIMULATION REPORT              *
166                     ********************************************************;
167
168             WRITE(EJK);
169             WRITE(F13);
170             WRITE(F16,MESG,RFGC);
171             WRITE(F8);
172             FOR I:=1 STEP 1 UNTIL 10 DO
   B9
173             BEGIN
174             X:=STCUS(I)/TIME/BSIZE*100.0;
175             WRITE(F9,I,X,ICNT(I));
   E9
176             END;
177             WRITE(F10);
178             FOR I:=1 STEP 1 UNTIL 28 DO
   B10
179             BEGIN
180             X:=STLUS(I)/TIME*100.0;
181             Y:=STLQL(I)/TIME;
182              WRITE(F11,I,X,Y,LCNT(I));
   E10
183             END;
184             WRITE(EJK);
185             WRITE('MESSAGE LIFE TIME DISTRIBUTION');
186             HPRINT(H1,R1);
187             WRITE('IMP COUNT OF MESSAGE');
188             HPRINT(H2,R2);
189             WRITE('MESSAGE DISTIBUTION OF ORG.IMP');
190             HPRINT(H3,R3);
191             WRITE(F15,TIME);
192             WRITE(EJK);
193             WRITE('SQS PRINT');
194             SQSPRINT;
195             WRITE('SET PRINT');
196             FOR I:=1 STEP 1 UNTIL 28 DO
197             WRITE(LQUE(I));
   E2   F1
198             END;
   E1
199             END;
COMPILATION COMPLETE
```

INPUT PARAMETER

977531 9765831 9765431 1000.000

BUFFER SIZE= 5 LINE SPEED= 200.0 M/S ARRAIVAL RATE: 1.00 SEC(MEAN)

HOUT MATRIX

0	2	3	3	3	7	2	7	2	7
1	0	1	4	4	6	6	6	6	6
1	1	0	4	4	4	4	4	4	4
3	2	3	0	5	5	5	5	5	5
4	4	4	4	0	6	7	6	7	7
2	2	2	5	5	0	7	8	8	8
6	6	5	5	5	6	0	9	9	9
6	6	6	6	6	6	9	0	9	10
7	8	7	7	7	8	7	8	0	10
8	8	8	8	8	8	9	8	9	0

LINE MATRIX

0	13	14	0	0	0	0	0	0	0
27	0	0	11	0	10	0	0	0	0
28	0	0	12	0	0	0	0	0	0
0	25	26	0	9	0	0	0	0	0
0	0	0	23	0	8	7	0	0	0
0	24	0	0	22	0	6	5	0	0
0	0	0	0	21	20	0	0	4	0
0	0	0	0	0	19	0	0	3	2
0	0	0	0	0	0	18	17	0	1
0	0	0	0	0	0	0	16	15	0

MESSAGE DISTRIBUTION

.0920	.1767	.2386	.4174	.5566	.6307	.8461	.9030	.9731	.9995
0.0000	.2327	.3625	.5803	.6701	.7250	.9006	.9337	.9776	.9995
.2470	.2470	.4705	.6587	.7528	.7763	.9174	.9409	.9761	.9996
.2033	.5250	.5250	.6951	.7798	.8136	.9322	.9491	.9829	.9998
.1022	.1931	.2490	.2990	.4999	.5737	.9089	.9429	.9826	.9996
.7290	.1312	.1476	.4887	.4887	.5908	.8973	.9337	.9794	.9992
.0883	.0968	.1731	.3011	.4920	.4920	.8763	.9173	.9720	.9993
.0759	.1320	.1660	.4933	.6914	.7730	.7730	.8535	.9667	.9997
.0535	.0892	.1070	.2141	.3033	.3568	.6490	.6490	.9353	.9888
.0579	.1013	.1302	.2316	.3185	.3764	.7242	.9560	.9560	.9774
.0769	.1538	.1922	.3075	.4228	.4997	.7669	.8842	.9995	.9995

SIMULATION RESULT

TOTAL MESSAGECOUNT: 997 REJECT MESSAGE COUNT: 763

CORE UTILIZATION

IMP NUMBR:	1	MEAN USAGE:	12.23 %	MESSAGE:	42
IMP NUMBR:	2	MEAN USAGE:	11.31 %	MESSAGE:	45
IMP NUMBR:	3	MEAN USAGE:	9.98 %	MESSAGE:	35
IMP NUMBR:	4	MEAN USAGE:	15.81 %	MESSAGE:	47
IMP NUMBR:	5	MEAN USAGE:	16.86 %	MESSAGE:	46
IMP NUMBR:	6	MEAN USAGE:	15.95 %	MESSAGE:	46
IMP NUMBR:	7	MEAN USAGE:	16.60 %	MESSAGE:	47
IMP NUMBR:	8	MEAN USAGE:	10.79 %	MESSAGE:	31
IMP NUMBR:	9	MEAN USAGE:	13.93 %	MESSAGE:	38
IMP NUMBR:	10	MEAN USAGE:	8.09 %	MESSAGE:	15

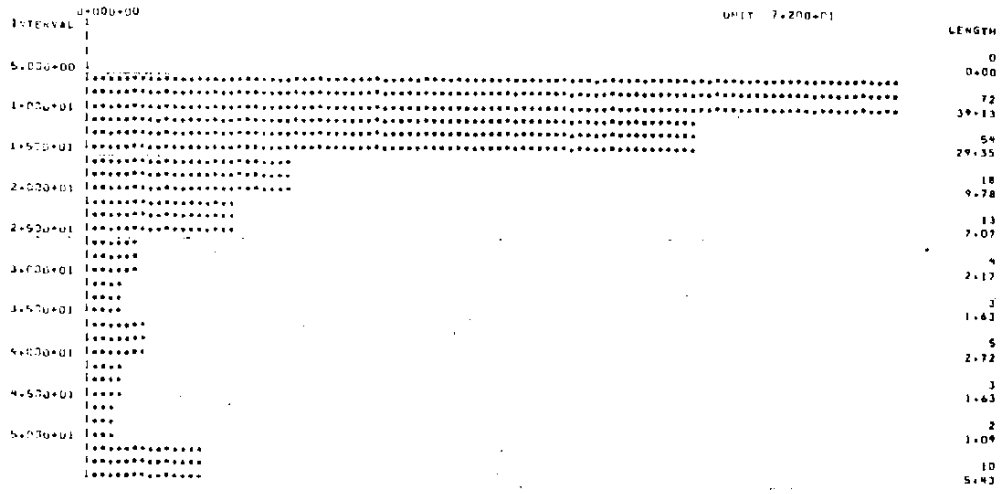
LINE STATISTICS

LINE NUMBER:	1	MEAN USAGE:	2.56 %	MEAN QUEUE LENGTH:	0.00	5
LINE NUMBER:	2	MEAN USAGE:	1.53 %	MEAN QUEUE LENGTH:	0.00	3
LINE NUMBER:	3	MEAN USAGE:	7.16 %	MEAN QUEUE LENGTH:	.06	14
LINE NUMBER:	4	MEAN USAGE:	7.67 %	MEAN QUEUE LENGTH:	.01	15
LINE NUMBER:	5	MEAN USAGE:	3.07 %	MEAN QUEUE LENGTH:	0.00	6
LINE NUMBER:	6	MEAN USAGE:	7.16 %	MEAN QUEUE LENGTH:	.05	14
LINE NUMBER:	7	MEAN USAGE:	5.63 %	MEAN QUEUE LENGTH:	.00	11
LINE NUMBER:	8	MEAN USAGE:	2.05 %	MEAN QUEUE LENGTH:	0.00	4
LINE NUMBER:	9	MEAN USAGE:	10.23 %	MEAN QUEUE LENGTH:	.37	20
LINE NUMBER:	10	MEAN USAGE:	6.65 %	MEAN QUEUE LENGTH:	.12	13
LINE NUMBER:	11	MEAN USAGE:	3.07 %	MEAN QUEUE LENGTH:	.03	6
LINE NUMBER:	12	MEAN USAGE:	3.58 %	MEAN QUEUE LENGTH:	.17	7
LINE NUMBER:	13	MEAN USAGE:	11.77 %	MEAN QUEUE LENGTH:	.03	23
LINE NUMBER:	14	MEAN USAGE:	7.16 %	MEAN QUEUE LENGTH:	.16	14
LINE NUMBER:	15	MEAN USAGE:	2.05 %	MEAN QUEUE LENGTH:	.03	4
LINE NUMBER:	16	MEAN USAGE:	3.07 %	MEAN QUEUE LENGTH:	.15	6
LINE NUMBER:	17	MEAN USAGE:	6.14 %	MEAN QUEUE LENGTH:	.00	12
LINE NUMBER:	18	MEAN USAGE:	8.19 %	MEAN QUEUE LENGTH:	.37	16
LINE NUMBER:	19	MEAN USAGE:	4.60 %	MEAN QUEUE LENGTH:	.06	9
LINE NUMBER:	20	MEAN USAGE:	5.63 %	MEAN QUEUE LENGTH:	.00	11

LINE NUMBER:	21	MEAN USAGE:	8.19 %	MEAN QUEUE LENGTH:	.39	16
LINE NUMBER:	22	MEAN USAGE:	4.60 %	MEAN QUEUE LENGTH:	.28	9
LINE NUMBER:	23	MEAN USAGE:	13.30 %	MEAN QUEUE LENGTH:	.35	26
LINE NUMBER:	24	MEAN USAGE:	6.14 %	MEAN QUEUE LENGTH:	0.00	12
LINE NUMBER:	25	MEAN USAGE:	2.05 %	MEAN QUEUE LENGTH:	0.00	4
LINE NUMBER:	26	MEAN USAGE:	9.21 %	MEAN QUEUE LENGTH:	.00	18
LINE NUMBER:	27	MEAN USAGE:	10.74 %	MEAN QUEUE LENGTH:	.02	21
LINE NUMBER:	28	MEAN USAGE:	11.77 %	MEAN QUEUE LENGTH:	.01	23

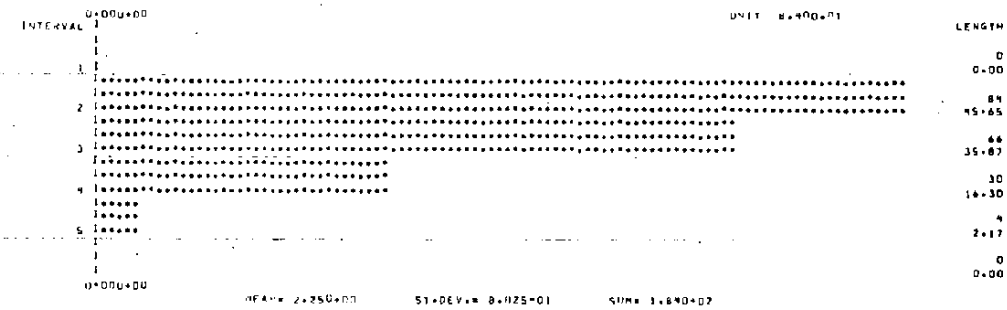
MESSAGE LIFE TIME DISTRIBUTION

H I S T O G R A M



IMP COUNT OF MESSAGE

H I S T O G R A M



MESSAGE DISTRIBUTION OF ORIGINP

HISTOGRAM

INTERVAL	0.000+00	UNIT 2.030+00	LENGTH
			85
1			8.53
			84
2			8.43
			62
3			6.22
			188
4			18.84
			193
5			19.39
			86
6			8.63
			203
7			20.36
			54
8			5.42
			67
9			6.72
			26
10			2.61
			0
			0.00

付表1.3　SIMULA－67

```
SIMULA-67 V1.0 P328                                          02/27/73   11.54 1RS

    0          ≠BEGIN≠
    0          SIMULATION  ≠BEGIN≠
    0
    0          ≠COMMENT≠ ************************************************...;
    1
    1                                        ≠COMMENT≠ SIMULATION MODEL
    1                                        OF COMPUTER NETWORK;
    2
    2          ≠INTEGER≠ ≠ARRAY≠ LMATRIX,RMATRIX[1:10,1:10];
    3          ≠INTEGER≠ I,J,K,BSIZE,LSPEED,U1,U2,U3,MESC,REJC;
    4          ≠REAL≠ X,Y,TRANST,RTRYT,ARRIVT,ETIME;
    5          ≠REAL≠  CN0,CN1;
    6          ≠REF≠ (HEAD) ≠ARRAY≠ LQUE[1:28];
    7          ≠INTEGER≠ ≠ARRAY≠ LSTATE,MCL[1:28];
    8          ≠ARRAY≠ ORGDIST[1:10],MDIST[1:10,1:10];
    9          ≠INTEGER≠ ≠ARRAY≠       MCIMP,B3,B2[1:10];
   10          ≠ARRAY≠ BUFC[1:10];
   11          ≠ARRAY≠ B1,ST3,TM3[1:10],ST1,ST2,TM1,TM2[1:28];
   12          ≠INTEGER≠ ≠ARRAY≠ M1,M2,M3[1:11];
   13
   13          ≠COMMENT≠ ************************************************;
   14
   14                                        ≠COMMENT≠    CLASS DECLALATION
   14                                        OF MESSAGE;
   15
   15          PROCESS ≠CLASS≠ MESSAGE(IMP,ENDIMP);
   16          ≠INTEGER≠ IMP,ENDIMP;
   17          ≠BEGIN≠
   17          ≠INTEGER≠ NXTIMP,UL,PL,MMC;
   18          ≠REAL≠ GTM;
   19          GTM:=TIME;
   20          NEWIMP:  NXTIMP:=RMATRIX[IMP,ENDIMP];
   21          UL:=LMATRIX[IMP,NXTIMP];
   22          MCIMP[IMP]:=MCIMP[IMP]+1;
   23          MMC:=MMC+1;
   24          ≠IF≠ LSTATE[UL] ≠EQUAL≠ 1 ≠THEN≠ ≠GO TO≠ QUEUE;
   26          ≠IF≠ BUFC[NXTIMP] ≠LESS≠ BSIZE ≠THEN≠ ≠GO TO≠ ULINE;
   28          QUEUE: X:=LQUE[UL].CARDINAL;
   29          ACCUM(ST2[UL],TM2[UL],X,0.0);
   30          WAIT(LQUE[UL]);
   31          ULINE:  ACCUM(ST3[NXTIMP],TM3[NXTIMP],BUFC[NXTIMP],0.0);
   32          BUFC[NXTIMP]:=BUFC[NXTIMP]+1;
   33          LSTATE[UL]:=1;
   34          MCL[UL]:=MCL[UL]+1;
   35          ACCUM(ST1[UL],TM1[UL],CN0,0.0);
   36          HOLD(TRANST);
   37          LSTATE[UL]:=0;
   38          ACCUM(ST1[UL],TM1[UL],CN1,0.0);
   39          ACCUM(ST3[IMP],TM3[IMP],BUFC[IMP],0.0);
   40          BUFC[IMP]:=BUFC[IMP]-1;
   41          ≠IF≠ PL ≠EQUAL≠ 0 ≠THEN≠ ≠GO TO≠ ACTM;
   43          ≠IF≠ LQUE[PL].CARDINAL ≠EQUAL≠ 0 ≠THEN≠ ≠GO TO≠ ACTM;
   45          ≠IF≠ LSTATE[PL] ≠EQUAL≠ 0 ≠THEN≠ ≠BEGIN≠
   46          ≠ACTIVATE≠ LQUE[PL].FIRST ≠DELAY≠ 0.0;
   47          X:=LQUE[PL].CARDINAL;
   48          ACCUM(ST2[PL],TM2[PL],X,0.0);
   49          LQUE[PL].FIRST.OUT;
   50          ≠END≠;
```

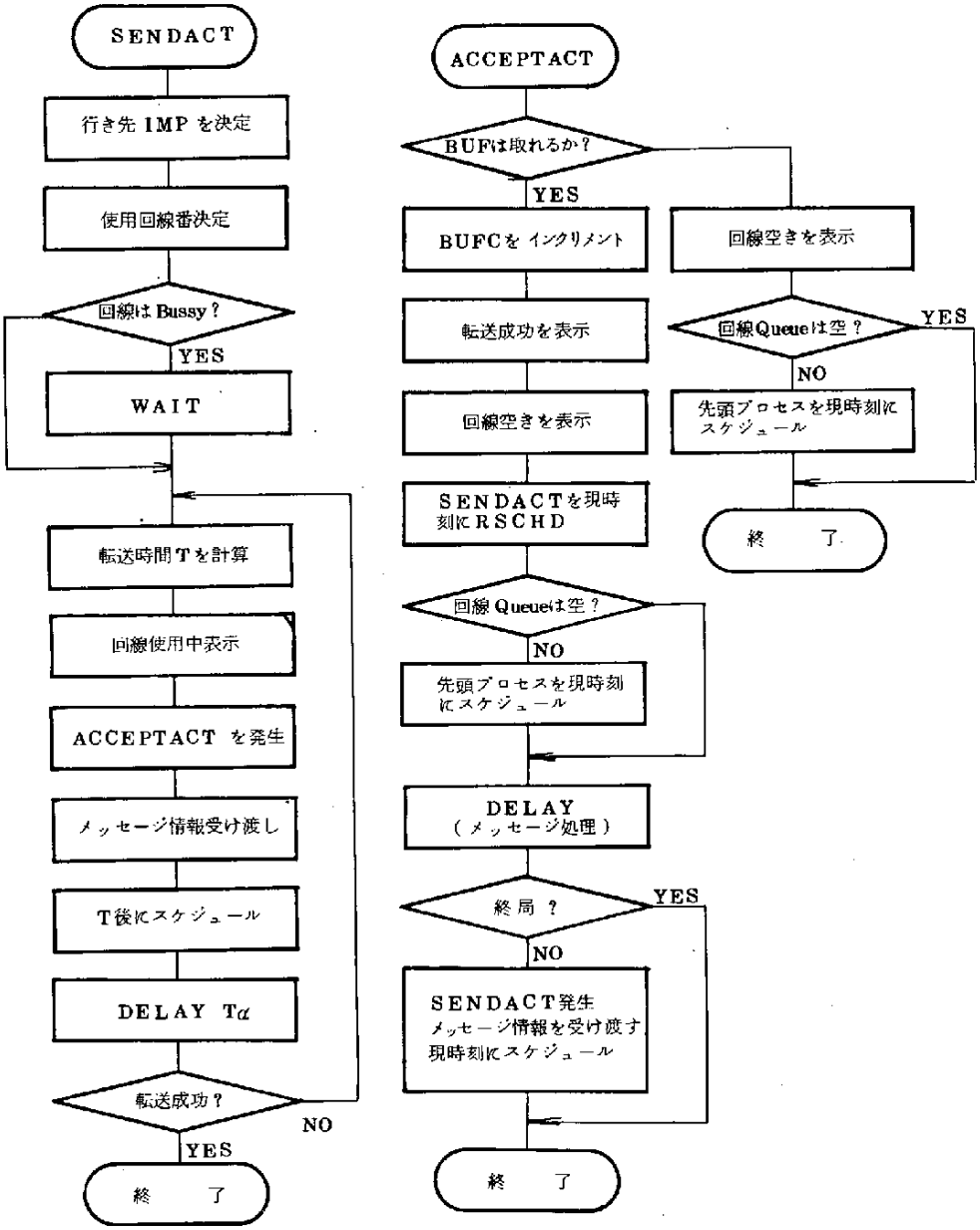
```
    53          ACTM#  ACCUM(ST3[IMP],TM3[IMP],BUFC[IMP],0.0);
    54          #IF# #NOT# LQUE[UL].EMPTY #AND# BUFC[NXTIMP] #LESS# BSIZE
    54          #THEN# #BEGIN#
    55          #ACTIVATE# LQUE[UL].FIRST #DELAY# 0.0;
    56          X1=LQUE[UL].CARDINAL;
    57          ACCUM(ST2[UL],TM2[UL],X,0.0);
    58          LQUE[UL].FIRST.OUT;  #END#;
    62          HOLD(0.1);
    63          PL1=UL;
    64          #IF# IMP #NOT EQUAL# ENDIMP #THEN# #GO TO# NEWIMP;
    66          ACCUM(ST3[IMP],TM3[IMP],BUFC[IMP],0.0);
    67          BUFC[IMP]1=BUFC[IMP]-1;
    68          GTM1=TIME-GTM;
    69          MMC1=MMC+1;
    70          HISTO(H1,B1,GTM,1);
    71          HISTO(H2,B2,MMC,1);
    72          #END#;
    75
    75          #COMMENT# ************************************************************;
    76
    76                                          #COMMENT# MAIN PROGRAM
    76                                          INITIALIZE PARAMETERS
    76                                          AND CONTROL SIMULATION RUN
    76                                          GENERATESIMULATION REPORT;
    77
    77          EJECT(5);
    78          OUTTEXT(*INPUP PARAMETER LIST*);
    79          SPACING(3);
    80          OUTIMAGE;
    81
    81                                          #COMMENT#  INPUT  PARAMETERS;
    82          CN01=0;
    83          CN11=1;
    84          BSIZE1=ININT;
    85          LSPEED1=ININT;
    86          ARRIVT1=INREAL;
    87          U11=ININT;
    88          U21=ININT;
    89          U31=ININT;
    90          ETIME1=INREAL;
    91          OUTTEXT(*SIMULATION END TIME1*);
    92          OUTFIX(ETIME,2,10);  OUTIMAGE;
    94          OUTTEXT(*BUFFER SIZE*);
    95          OUTINT(BSIZE,10);  OUTIMAGE;
    97          OUTTEXT(*LINE SPEED1*);
    98          OUTINT(LSPEED,10);
    99          OUTTEXT(* B/S*);  OUTIMAGE;
   101           OUTTEXT(*MESSAGE ARRATVAL RATE1*);
   102          OUTFIX(ARRIVT,2,10);  OUTTEXT(* SEC(MEAN)*);
   104          OUTIMAGE;
   105          EJECT(5);
   106          OUTTEXT(*POUT MATRIX*);
   107          SPACING(3);
   108          OUTIMAGE;
   109          #FOR# I1=1 #STEP# 1 #UNTIL# 10 #DO# #BEGIN#
   110          #FOR# J1=1 #STEP# 1 #UNTIL# 10 #DO# #BEGIN#
   111               RMATRIX[I,J]1=ININT;
   112               OUTINT(RMATRIX[I,J],10);  #END#;
```

```
116             OUTIMAGE;  #END#;
120         EJECT(5);
121         OUTTEXT(*LINE MATRIX*);
122         SPACING(3);
123         OUTIMAGE;
124         #FOR# I:=1 #STEP# 1 #UNTIL# 10 #DO# #BEGIN#
125         #FOR# J:=1 #STEP# 1 #UNTIL# 10 #DO# #BEGIN#
126             LMATRIX[I,J]:=ININT;
127             OUTINT(LMATRIX[I,J],10);  #END#;
131             OUTIMAGE;   #END#;
135         EJECT(5);
136         OUTTEXT(*DISTRIBUTION OF ORG.IMP NUMBER*);
137         SPACING(3);
138         OUTIMAGE;
139         #FOR# I:=1 #STEP# 1 #UNTIL# 10 #DO# #BEGIN#
140             ORGDIST[I]:=INREAL;
141         OUTFIX(ORGDIST[I],4,10);  #END#;
145         SPACING(3);
146         OUTIMAGE;
147         OUTTEXT(*DISTRIBUTION OF END.IMP NUMBER*);
148         SPACING(3);
149         OUTIMAGE;
150         #FOR# I:=1 #STEP# 1 #UNTIL# 10 #DO# #BEGIN#
151         #FOR# J:=1 #STEP# 1 #UNTIL# 10 #DO# #BEGIN#
152             MDIST[I,J]:=INREAL;
153             OUTFIX(MDIST[I,J],4,10);  #END#;
157             OUTIMAGE;  #END#;
161         #FOR# I:=1 #STEP# 1 #UNTIL# 28 #DO#
162             LQUE[I]:-#NEW# HEAD;
163         OUTTEXT(*QUEUE*);  OUTIMAGE;
165         #FOR# I:=1 #STEP# 1 #UNTIL# 10 #DO#
166             BUFC[I]:=0;
167         #FOR# I:=1 #STEP# 1 #UNTIL# 28 #DO#
168             LSTATE[I]:=0;
169
169         #COMMENT# ********************************************;
170
170                                     #COMMENT#  GENERATE NEW MESSAGE;
171
171         ORGIMP:   X:=UNIFORM(0.0,1.0,U1);
172         K:=1;
173         #FOR# I:=1 #STEP# 1 #UNTIL# 10 #DO# #BEGIN#
174             K:=I;
175             #IF# ORGDIST[I] #GREATER# X #THEN# #GO TO# LASTIMP; #END#;
180         LASTIMP:  X:=UNIFORM(0.0,1.0,U2);
181         J:=1;
182         #FOR# I:=1 #STEP# 1 #UNTIL# 10 #DO# #BEGIN#
183             J:=I;
184             #IF# MDIST[I,J] #GREATER# X #THEN# #GO TO# NEWMSG; #END#;
189         NEWMSG:  MESC:=MESC+1;
190         #IF# BUFC[K] #NOT LESS# BSIZE #THEN#
191             REJC:=REJC+1   #ELSE#  #BEGIN#
192             BUFC[K]:=BUFC[K]+1;
193             #ACTIVATE# #NEW# MESSAGE(K,J) #DELAY# 0.0;  #END#;
197         HOLD(NEGEXP(1.0/ARRIVT,U3));
198         #IF# TIME #LESS# ETIME #THEN# #GO TO# ORGIMP;
200         #COMMENT# ********************************************;
201
```

```
201                                          ≠COMMENT≠ GENERATE SIMULATION
201                                                    REPORT;
202
202         EJECT(5);
203         OUTTEXT(+SIMULATION RESULT+);
204         SPACING(5);
205         OUTIMAGE;
206         OUTTEXT(+SIMULATION END TIME:+);
207         OUTFIX(TIME,2,10);
208         SPACING(2);
209         OUTIMAGE;
210         OUTTEXT(+TOTAL MESSAGE COUNT:+);
211         OUTINT(MESC,10);  OUTIMAGE;
213         OUTTEXT(+REJECTED MESSAGE COUNT:+);
214         OUTINT(REJC,10);  OUTIMAGE;
216         EJECT(5);
217         OUTTEXT(+THE STATISTICS ABOUT IMP+);   OUTIMAGE;
219         SPACING(5);
220         OUTIMAGE;
221         OUTTEXT(+IMP NUMBER    UTILIZATION  MESSAGE COUNT+);
222         SPACING(3);
223         OUTIMAGE;
224         ≠FOR≠ I:=1 ≠STEP≠ 1 ≠UNTIL≠ 10 ≠DO≠ ≠BEGIN≠
225         X:=ST3(I)/TIME/dSIZE*100.0;
226         OUTINT(I,10);  OUTFIX(X,2,13);
228         OUTINT(MSIMP(I),15);  SPACING(2);  OUTIMAGE;  ≠END≠;
234         EJECT(5);
235         OUTTEXT(+THE STATISTICS ABOUT LINE+);  OUTIMAGE;
237         SPACING(5);
238         OUTIMAGE;
239         OUTTEXT(+LINE NUMBER   UTILIZATION  MEAN LENGTH+);
240         OUTTEXT(+  MESSAGE COUNT+);  OUTIMAGE;
242         SPACING(3);
243         OUTIMAGE;
244         ≠FOR≠ I:=1 ≠STEP≠ 1 ≠UNTIL≠ 28 ≠DO≠  ≠BEGIN≠
245         X:=ST1(I)/TIME*100.0;
246         Y:=ST2(I)/TIME;
247         OUTINT(I,10);  OUTFIX(X,2,13);  OUTTEXT(+ +);
250         OUTINT(MSL(I),15);  SPACING(2);  OUTIMAGE;  ≠END≠;
256         ≠END≠;
259         ≠END≠;
259             ≠EOP≠
```

2. プロセスタイプ（Ⅱ）

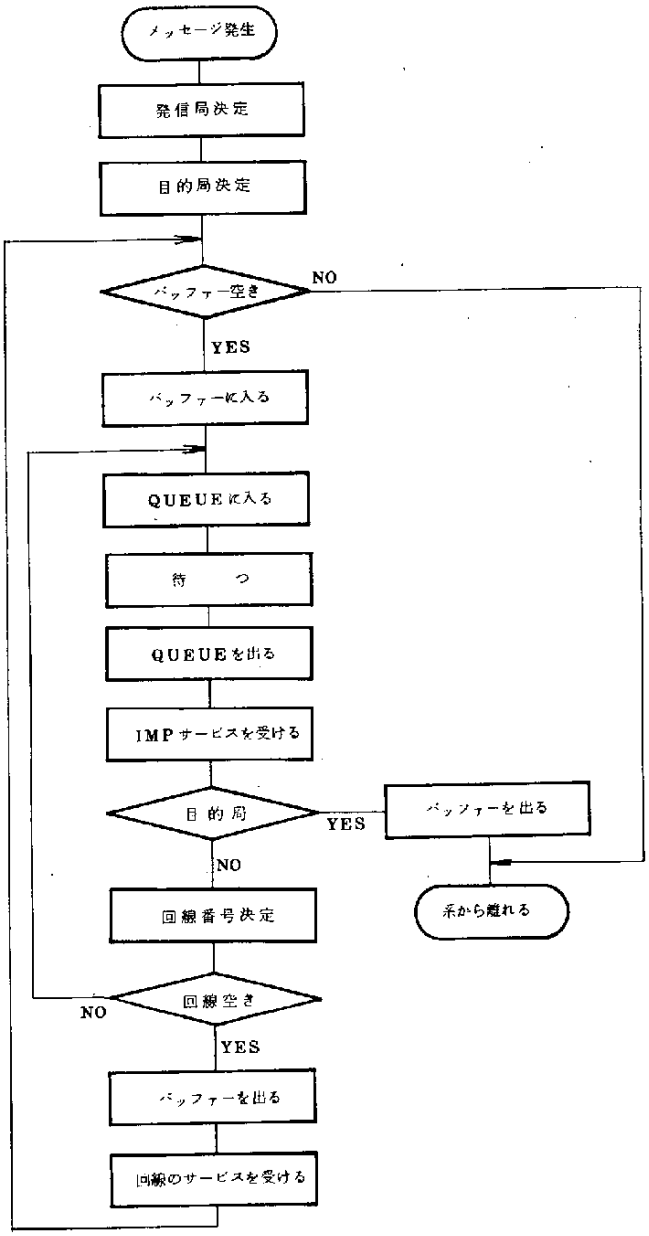
付表2.1、2.2は、付図2のIMP中心のフローチャートに基づく、SIMULA－65、SIMULA－67のプログラムリストである。



付図2.（IMP中心のプロセスオリエンティッドなモデル化フロー）

3. トランザクションタイプ

付表3.1は、付図3に基づくGPSS－Vのプログラムリスト、結果並びにトレース情報を示したものである。



付図3. メッセージ処理のフロー

付表2.1　SIMULA－65

```
@Q,ALG,ISCD    SIMS,SIMRB
CYCLE  000  COMPILED BY  1204  0006  ON 03/27/73 AT 22:51:18
     B1
 1           BEGIN
  B2    S1    L1
 2           SIMULA BEGIN
 3           COMMENT ************************************************************
 4                   *                                                          *
 5                   *        THE SIMULATION MODEL OF COMPUTER NETWORK          *
 6                   *                                                          *
 7                   ************************************************************;
 8           SET ARRAY LQUE(1:28);
 9           ARRAY MOIST(1:10,1:10),ORGDIST(1:10);
10           INTEGER ARRAY LMATRIX,RMATRIX(1:10,1:10);
11           INTEGER ARRAY BUFC,MCIMP,B1,B2(1:10);
12           INTEGER ARRAY LSTATE,MCL(1:28);
13           INTEGER ARRAY H1,H2,H3(1:11);
14           ELEMENT ARRAY LN(1:28);
15           ARRAY B3(1:10),ST1,ST2,TM1,TM2(1:28),ST3,TM3(1:10);
16           INTEGER I,J,K,MESC,BSIZE,LSPEED,U1,U2,U3,REJC;
17           REAL X,Y,TRANST,ETIME,RTRYT,ARRAIVT;
18           FORMAT     F1(E1,'INPUT PARAMETERS',A1,3);
19           FORMAT     F2('CONSTANTS FOR RANDOM DRAWING',A1,2,3I10,A1,3);
20           FORMAT     F3('BUFFER SIZE:',I10,A1,1,'TRANSFER SPEED OF LINE:',
21                      I10,' B/S',A1,1,'RETRY TIME:',D10,2,A1,1,
22                      'ARRAIVAL RATE:',D10,2,' SEC(MEAN)',A1,3);
23           FORMAT     F4('SIMULATION TIME:',D10,2,' SEC',A1);
24           FORMAT     F5(E1,'ROUT MATRIX',A3,2,(10I10,A2));
25           FORMAT     F6('LINE MATRIX',A6,2,(10I10,A2));
26           FORMAT     F7(E1,'CUMULATIVE DISTRIBUTION OF ORIGINAL IMP NUMBER',
27                      A1,2,10D10,2,A1,4);
28           FORMAT     F8('CUMULATIVE DISTRIBUTION OF END IMP NUMBER',
29                      A1,2,(10D10,4,A2));
30           FORMAT     F9('ND BUFFER; MESSAGE FROM',I5,' TO',I5,X10,
31                      'RETRY COUNT IS',I10,A1);
32           FORMAT     F10(E1,'SIMULATION RESULT',A3,3);
33           FORMAT     F11('SIMULATION END TIME:',D10,2,A2,'TOTAL MESSAGE:',
34                      I10,A2,'REJECTED MESSAGE COUNT:',I10,A1);
35           FORMAT     F12(E1,'THE STATISTICS ABOUT IMP',A3,3,
36                      'IMP NUMBER    UTILIZATION  MESSAGE COUNT',A1,2);
37           FORMAT     F13(I10,X3,D10,2,' %',X5,I10,A1,1);
38           FORMAT     F14(E1,'THE STATISTICS ABOUT LINE',A3,3,
39                      'LINE NUMBER   UTILIZATION  MEAN LENGTH  MESSAGE COUNT',
40                      A1,2);
41           FORMAT     F15(I10,X3,D10,2,' %',X3,D10,2,X5,I10,A1,1);
42           FORMAT     F16(E1,'DISTRIBUTION (ORGINAL IMP NUMBER)',A10);
43           FORMAT     F17(E1,'DISTRIBUTION (IMP COUNT OF MESSAGE)',A10);
44           FORMAT     F18(E1,'DISTRIBUTION (MESSAGE LIFE TIME)',A10);
45
46           COMMENT ***************VIRTUAL PROCEDURE **********************;
47
48           LOCAL ELEMENT PROCEDURE VACCEPT;
49
50           COMMENT ************************************************************
51                   *    ACTIVITY  SENDACT                                     *
52                   ************************************************************;
53
```

```
\2    L2
54          ACTIVITY SENDACT(IMP,ENDIMP,IC,GTM);
55          INTEGER IMP,ENDIMP,IC;
56          REAL GTM;
  B3
57          BEGIN
58          INTEGER NXTIMP,I,NC,TRS;
59          NXTIMP:=RMATRIX(ENDIMP,IMP);
60          I:=LMATRIX(NXTIMP,IMP);
  B4
61          IF LSTATE(I) NEQ 0 THEN BEGIN
62               ACCUM(ST2(I),TM2(I),CARDINAL(LQUE(I)),0);
  E4
63               WAIT(LQUE(I));  END;
64          LN(I):=CURRENT;
65          NC:=0;
66          TRS:=0;
67          SEND:  LSTATE(I):=1;
68          ACCUM(ST1(I),TM1(I),0,0);
69          ACTIVATE     VACCEPT(NXTIMP,ENDIMP,I,IC,GTM) DELAY TRANST;
70          HOLD(RTRYT);
  B5
71          IF TRS NEQ 1 THEN BEGIN
72               NC:=NC+1;
  E5
73               GO TO SEND;  END;
74          ACCUM(ST3(IMP),TM3(IMP),BUFC(IMP),0);
75          BUFC(IMP):=BUFC(IMP)-1;
76          MCL(I):=MCL(I)+1;
77          IF NC NEQ 0 THEN WRITE(F9,IMP,NXTIMP,NC);
  E3    F2
78          END;
79
80          COMMENT ************************************************
81                *       ACTIVITY   ACCEPT                        *
82                ************************************************;
83
  \3    L2
84          ACTIVITY ACCEPT(IMP,ENDIMP,L,IC,GTM);
85          INTEGER IMP,ENDIMP,L,IC;
86          REAL GTM;
  B6
87          BEGIN
88          IC:=IC+1;
89          MCIMP(IMP):=MCIMP(IMP)+1;
90          IF L EQL 0 THEN GO TO ACT;
  B7
91          IF BUFC(IMP) GEQ BSIZE THEN BEGIN
92               LSTATE(L):=0;
93               ACCUM(ST1(L),TM1(L),1,0);
  B8
94               IF NOT EMPTY(LQUE(L)) THEN BEGIN
95                    ACTIVATE FIRST(LQUE(L)) DELAY 0.0;
96                    ACCUM(ST2(L),TM2(L),CARDINAL(LQUE(L)),0);
  E8
97                    REMOVE(FIRST(LQUE(L)));  END;
  E7
```

```
                    GO TO TERM;  END;
  S4      L4    F4
                INSPECT LN(L) WHEN SENDACT DO TRSi=1;
                LSTATE(L):=0;
                ACCUM(ST1(L),TM1(L),1,0);
                REACTIVATE LN(L) DELAY 0.0;
    B9
                IF NOT EMPTY(LQUE(L)) THEN BEGIN
                    ACTIVATE FIRST(LQUE(L)) DELAY 0.0;
                    ACCUM(ST2(L),TM2(L),CARDINAL(LQUE(L)),0);
                    REMOVE(FIRST(LQUE(L)));
    E9
                    END;
                ACT:  ACCUM(ST3(IMP),TM3(IMP),BUFC(IMP),0);
                BUFC(IMP):=BUFC(IMP)+1;
                HOLD(0.1);
                IF IMP NEQ ENDIMP THEN
                    ACTIVATE NEW SENDACT(IMP,ENDIMP,IC,GTM) DELAY 0.0
    B10
                ELSE BEGIN
                    ACCUM(ST3(IMP),TM3(IMP),BUFC(IMP),0);
    E10
                    BUFC(IMP):=BUFC(IMP)-1;  END;
                HISTO(H2,B2,IC,1);
                HISTO(H3,B3,TIME-GTM,1);
    E6    F3
                TERM:  END;

                COMMENT ********** VIRTUAL PROCEDURE DECLALATION **********;

    S5    L2
                ELEMENT PROCEDURE VACCEPT(IMP,ENDIMP,L,IC,GTM);
                INTEGER IMP,ENDIMP,L,IC;
                REAL GTM;
    B11
                BEGIN
                VACCEPT:-NEW ACCEPT(IMP,ENDIMP,L,IC,GTM);
    E11   F5
                END;

                COMMENT **********************************************
                *          MAIN PROGRAM START                          *
                **********************************************;

                COMMENT ************ PARAMETER READ ***************************;

                READ(BSIZE,LSPEED,ETIME,RTRYT,ARRAIVT);
                READ(U1,U2,U3);
                READ(RMATRIX);
                READ(LMATRIX);
                READ(ORGDIST);
                READ(MDIST);

                COMMENT ************,INPUT PARAMETER DISPLAY*****************;

                WRITE(F1);
                WRITE(F2,U1,U2,U3);
```

```
146             FULLTRACE;
147             WRITE(F3,BSIZE,LSPEED,RTRYT,ARRAIVT);
148             WRITE(F4,ETIME);
149             WRITE(F5,RMATRIX);
150             WRITE(F6,LMATRIX);
151             WRITE(F7,ORGDIST);
152             WRITE(F8,MDIST);
153
154             COMMENT **************INITIALIZE*******************************;
155
156             TRANST:=1024.0/LSPEED;
   B12
157             FOR I:=1 STEP 1 UNTIL 10 DO BEGIN
158                  B1(I):=B2(I):=I;
159                  B3(I):=5.0*I;
160                  BUFC(I):=MCIMP(I):=0;
   E12
161                  ST3(I):=TM3(I):=0.0; END;
   B13
162             FOR I:=1 STEP 1 UNTIL 28 DO BEGIN
163                  LSTATE(I):=MCL(I):=0;
   E13
164                  ST1(I):=ST2(I):=TM1(I):=TM2(I):=0.0;  END;
165             MESC:=REJC:=0;
166
167             COMMENT ************** GENERATING NEW MESSAGE *****************;
168
169             ORGD:  X:=UNIFORM(0.0,1.0,U1);
170             K:=1;
   B14
171             FOR I:=1 STEP 1 UNTIL 10 DO BEGIN
172                  K:=I;
   E14
173                  IF ORGDIST(I) GTR X THEN GO TO ENDD; END;
174             ENDD:  X:=UNIFORM(0.0,1.0,U2);
175             J:=1;
   B15
176             FOR I:=1 STEP 1 UNTIL 10 DO BEGIN
177                  J:=I;
   E15
178                  IF MDIST(I,K) GTR X THEN GO TO NEWMSG; END;
179             NEWMSG:  MESC:=MESC+1;
180             HISTO(H1,B1,K,1);
181             IF BUFC(K) GEQ BSIZE THEN REJC:=REJC+1
   B16
182             ELSE BEGIN
   E16
183                  ACTIVATE NEW ACCEPT(K,J,0,1,TIME) DELAY 0.0;  END;
184             HOLD(NEGEXP(1,0/ARRAIVT,U3));
185             IF TIME LSS ETIME THEN GO TO ORGD;
186             COMMENT ************************************************************
187                  *       GENERATING  SIMULATION REPORT                        *
188                  ************************************************************;
189             WRITE(F10);
190             WRITE(F11,TIME,MESC,REJC);
191             WRITE(F12);
   B17
```

```
   192              FOR I:=1 STEP 1 UNTIL 10 DO BEGIN
   193                   X:=ST3(I)/TIME/BSIZE*100.0;
     E17
   194                   WRITE(F13,I,X,MCIMP(I)); END;
   195              WRITE(F14);
     B18
   196              FOR I:=1 STEP 1 UNTIL 28 DO BEGIN
   197                   X:=ST1(I)/TIME*100.0;
   198                   Y:=ST2(I)/TIME;
     E18
   199                   WRITE(F15,I,X,Y,MCL(I)); END;
   200              WRITE(F16);
   201              HPRINT(H1,B1);
   202              WRITE(F17);
   203              HPRINT(H2,B2);
   204              WRITE(F18);
   205              HPRINT(H3,B3);
     E2     F1
   206              END;
     E1
   207              END;
COMPILATION COMPLETE
```

INPUT PARAMETERS

CONSTANTS FOR RANDOM DRAWING

9765777 9765777 68421371

BUFFER SIZE: 5

TRANSFER SPEED OF LINE: 200 B/S

RETRY TIME: 10.00

ARRAIVAL RATE: 1.00 SEC(MEAN)

SIMULATION TIME: 100.00 SEC

ROUT MATRIX

0	2	3	3	5	2	2	2	2	2
1	0	1	4	4	6	6	6	6	6
1	1	0	4	4	4	4	4	4	4
3	2	3	0	5	5	5	5	5	5
4	4	4	4	0	6	7	6	7	7
2	2	2	5	5	0	7	8	8	8
6	6	5	5	5	6	0	9	9	9
6	6	6	6	6	6	9	0	9	10
7	8	7	7	7	8	7	8	0	10
8	8	8	8	8	8	9	8	9	0

LINE MATRIX

0	13	14	0	0	0	0	0	0	0
27	0	0	11	0	10	0	0	0	0
28	0	0	12	0	0	0	0	0	0
0	25	26	0	9	0	0	0	0	0
0	0	0	23	0	8	7	0	0	0
0	24	0	0	22	0	6	5	0	0
0	0	0	0	21	20	0	0	4	0
0	0	0	0	0	19	0	0	3	2
0	0	0	0	0	0	18	17	0	1
0	0	0	0	0	0	0	16	15	0

CUMULATIVE DISTRIBUTION OF ORIGINAL IMP NUMBER

.09	.18	.24	.42	.56	.63	.85	.90	.97	1.00

CUMULATIVE DISTRIBUTION OF END IMP NUMBER

0.0000	.2307	.3625	.5603	.6701	.7250	.9008	.9337	.9776	1.0000
.2470	.2470	.4705	.6587	.7528	.7763	.9174	.9409	.9761	1.0000
.2033	.5250	.5250	.6951	.7798	.8136	.9322	.9491	.9829	1.0000
.1022	.1931	.2490	.2490	.4999	.5737	.9089	.9429	.9826	1.0000
.7290	.1312	.1676	.4887	.4887	.5908	.8973	.9337	.9794	1.0000
.0683	.0958	.1231	.3011	.4920	.4920	.8763	.9173	.9720	1.0000
.0754	.1320	.1650	.4433	.6414	.7730	.7730	.8535	.9667	1.0000
.0535	.0892	.1070	.2141	.3033	.3568	.6490	.6490	.9353	1.0000
.0579	.1013	.1302	.2316	.3185	.3764	.7242	.9560	.9560	1.0000
.0769	.1538	.1922	.3075	.4228	.4997	.7689	.8842	.9995	1.0000

SIMULATION RESULT

SIMULATION END TIME: 100.09

TOTAL MESSAGE: 92
REJECTED MESSAGE COUNT: 6

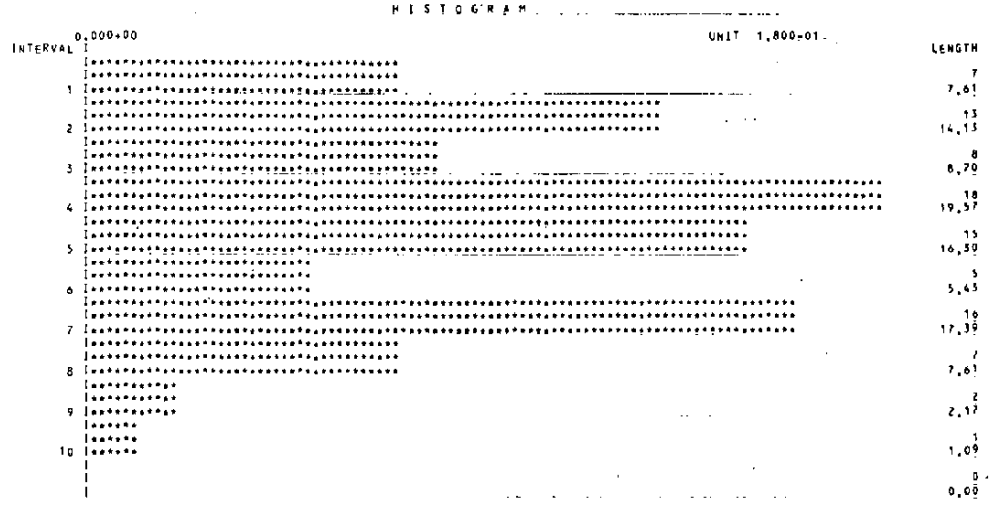
THE STATISTICS ABOUT IMP

IMP NUMBER	UTILIZATION	MESSAGE COUNT
1	20,73 %	34
2	15,88 %	21
3	36,38 %	20
4	58,84 %	39
5	43,32 %	31
6	5,58 %	15
7	18,40 %	20
8	6,73 %	8
9	3,27 %	11
10	1,08 %	3

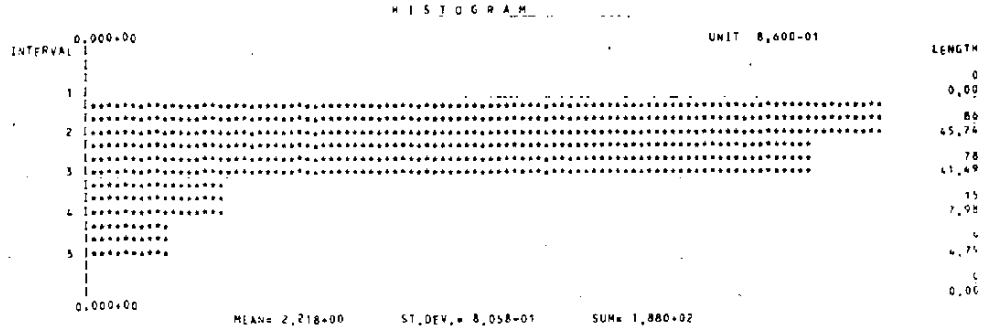
THE STATISTICS ABOUT LINE

LINE NUMBER	UTILIZATION	MEAN LENGTH	MESSAGE COUNT
1	10.23 %	0.00	2
2	0.00 %	0.00	0
3	30.69 %	.02	6
4	10.23 %	.07	2
5	0.00 %	0.00	0
6	25.58 %	.01	5
7	0.00 %	0.00	0
8	0.00 %	0.00	0
9	48.24 %	.06	13
10	0.00 %	0.00	0
11	0.00 %	0.00	0
12	0.00 %	0.00	0
13	40.92 %	.53	8
14	0.00 %	0.00	0
15	5.12 %	0.00	1
16	0.00 %	0.00	0
17	5.12 %	0.00	1
18	0.00 %	0.00	0
19	0.00 %	0.00	0
20	51.16 %	.21	10
21	0.00 %	0.00	0
22	0.00 %	0.00	0
23	19.84 %	.19	9
24	0.00 %	0.00	0
25	0.00 %	0.00	0
26	61.39 %	.12	12
27	61.39 %	.16	12
28	76.73 %	1.02	15

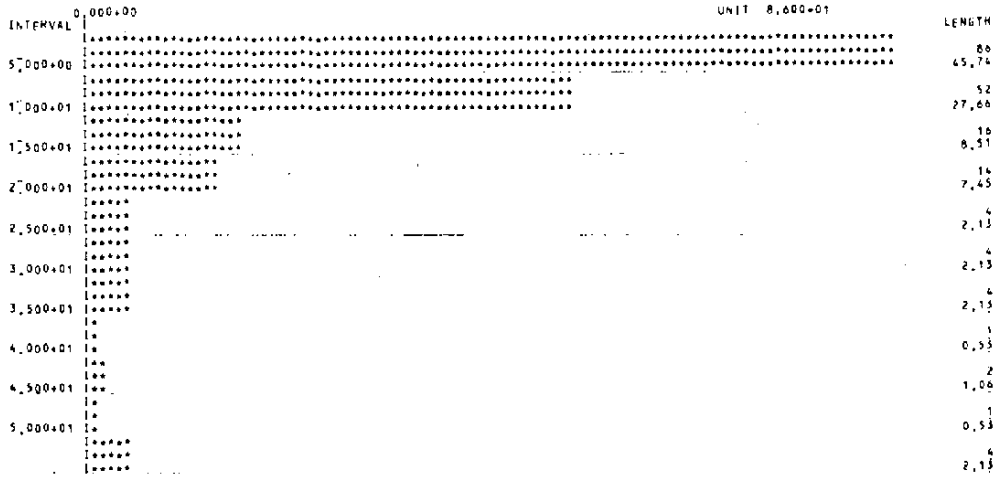
DISTRIBUTION (ORGINAL IMP NUMBER)



DISTRIBUTION (IMP COUNT OF MESSAGE)



DISTRIBUTION (MESSAGE LIFE TIME)



付表2.2　SIMULA－67

```
SIMULA-67 V1.0 P328                                        12/27/73   13.12 HRS

    0         %BEGIN%
    0         SIMULATION  %BEGIN%
    0
    0         %COMMENT% ************************************************************;
    1
    1                                   %COMMENT% SIMULATION MODEL
    1                                   OF COMPUTER NETWORK;
    2
    2         %INTEGER% %ARRAY% LMATRIX,KMATRIX(1:10,1:10);
    3         %INTEGER% I,J,K,BSIZE,LSPEED,U1,U2,U3,MESG,REJC;
    4         %REAL% X,Y,TRANST,RTRYT,ARRIVT,CTIME;
    5         %REAL%  CN0,CN1;
    6         %REF% (SENDACT) %ARRAY% LN(1:28);
    7         %REF% (HEAD) %ARRAY% LQUE(1:28);
    8         %INTEGER% %ARRAY% LSTATE,MCL(1:28);
    9         %ARRAY% ORGDIST(1:10),MDIST(1:10,1:10);
   10         %INTEGER% %ARRAY% BUFC,MCIMP,B3,B2(1:10);
   11         %ARRAY% B1,ST3,TM3(1:10),ST1,ST2,TM1,TM2(1:28);
   12         %INTEGER% %ARRAY% M1,M2,M3(1:11);
   13
   13         %COMMENT% ************************************************************;
   14
   14                                   %COMMENT% CLASS DECLALATION
   14                                   OF SENDACT.
   14                                   SEND MESSAGES FROM IMP TO NXTIMP;
   15
   15         PROCESS %CLASS% SENDACT(IMP,ENDIMP,IC,GTM);
   16         %INTEGER% IMP,ENDIMP,IC;
   17         %REAL% GTM;
   18         %BEGIN%
   18         %INTEGER% NXTIMP,I,NC,TRS;
   19         NXTIMP:=KMATRIX(IMP,ENDIMP);
   20         I:=LMATRIX(IMP,NXTIMP);
   21         %IF% LSTATE(I) %NOT EQUAL% 0 %THEN% %BEGIN%
   22              ACCUM(ST2(I),TM2(I),LQUE(I).CARDINAL,0);
   23              WAIT(LQUE(I));  %END%;
   27         LN(I):-CURRENT;
   28         NC:=0;
   29         TRS:=0;
   30         SEND:  LSTATE(I):=1;
   31         ACCUM(ST1(I),TM1(I),CN0,0.0);
   32         %ACTIVATE% %NEW% ACCEPT(NXTIMP,ENDIMP,I,IC,GTM) %DELAY% TRANST;
   33         HOLD(RTRYT);
   34         %IF% TRS %NOT EQUAL% 1 %THEN% %BEGIN%
   35              NC:=NC+1;
   36              %GO TO% SEND;  %END%;
   40         ACCUM(ST3(IMP),TM3(IMP),BUFC(IMP),0);
   41         BUFC(IMP):=BUFC(IMP)-1;
   42         MCL(I):=MCL(I)+1;
   43         %IF% NC %NOT EQUAL% 0 %THEN% %BEGIN%
   44              OUTTEXT(*NO BUFFER*);
   45              OUTINT(NC,10);  %END%;
   49         %END%;
   52
   52         %COMMENT% ************************************************************;
   53
   53         %COMMENT% ************************************************************;
   54
```

```
                                        #COMMENT# CLASS DECLALATION
                                        OF ACCEPT.
                                        ACCEPT MESSAGES FROM ANTHER IMP;

      PROCESS #CLASS# ACCEPT(IMP,ENDIMP,L,IC,GTM);
      #INTEGER# IMP,ENDIMP,L,IC;
      #REAL# GTM;
      #BEGIN# #REF# (HEAD) E;
      IC#=IC+1;
      MCIMP[IMP]#=MCIMP[IMP]+1;
      #IF# L #EQUAL# 0 #THEN# #GO TO# ACT;
      #IF# BUFC[IMP] #NOT LESS# BSIZE #THEN# #BEGIN#
           LSTATE[L]#=0;
           E#-LQUE[L];
           ACCUM(ST1[L],TM1[L],CN1,0.0);
           #IF# #NOT# E.EMPTY #THEN# #BEGIN#
                #ACTIVATE# E.FIRST #DELAY# 0.0;
                ACCUM(ST2[L],TM2[L],E.CARDINAL,0);
                E.FIRST.OUT;  #END#;
           #GO TO# TERM;  #END#;
      LN[L].IRS#=1;
      LSTATE[L]#=0;
      ACCUM(ST1[L],TM1[L],CN1,0.0);
      #REACTIVATE# LN[L] #DELAY# 0.0;
      E#-LQUE[L];
      #IF# #NOT# E.EMPTY #THEN# #BEGIN#
           #ACTIVATE# E.FIRST #DELAY# 0.0;
           ACCUM(ST2[L],TM2[L],E.CARDINAL,0);
           E.FIRST.OUT;  #END#;
      ACT:  ACCUM(ST3[IMP],TM3[IMP],BUFC[IMP],0);
      BUFC[IMP]#=BUFC[IMP]+1;
      HOLD(0.1);
      #IF#  IMP #NOT EQUAL# ENDIMP #THEN#
           #ACTIVATE# #NEW# SENDACT(IMP,ENDIMP,IC,GTM) #DELAY# 0.0
      #ELSE# #BEGIN#
           ACCUM(ST3[IMP],TM3[IMP],BUFC[IMP],0);
           BUFC[IMP]#=BUFC[IMP]-1;
           #END#;
      HISTO(H2,B2,IC,1);
      HISTO(H3,B3,TIME-GTM,1);
      TERM: #END#;

      #COMMENT# ********************************************************;

                                        #COMMENT# MAIN PROGRAM
                                        INITIALIZE PARAMETERS
                                        AND CONTROL SIMULATION RUN
                                        GENERATESIMULATION REPORT;

      EJECT(5);
      OUTTEXT(*INPUR PARAMETER LIST*);
      SPACING(3);
      OUTIMAGE;

                                        #COMMENT# INPUT PARAMETERS;
      CN0#=0;
      CN1#=1;
      BSIZE#=ININT;
```

```
115         LSPEED:=ININT;
116         ETIME:=INREAL;
117         RTRYT:=INREAL;
118         ARRIVT:=INREAL;
119         U1:=ININT;
120         U2:=ININT;
121         U3:=ININT;
122         OUTTEXT(*SIMULATION END TIME:*);
123         OUTFIX(ETIME,2,10);  OUTIMAGE;
125         OUTTEXT(*BUFFER SIZE:*);
126         OUTINT(BSIZE,10);  OUTIMAGE;
128         OUTTEXT(*LINE SPEED:*);
129         OUTINT(LSPEED,10);
130         OUTTEXT(* B/S*);  OUTIMAGE;
132          OUTTEXT(*MESSAGE ARRAIVAL RATE:*);
133         OUTFIX(ARRIVT,2,10);  OUTTEXT(* SEC(MEAN)*);
135         OUTIMAGE;
136         EJECT(5);
137         OUTTEXT(*ROUT MATRIX*);
138         SPACING(3);
139         OUTIMAGE;
140         #FOR# I:=1 #STEP# 1 #UNTIL# 10 #DO# #BEGIN#
141         #FOR# J:=1 #STEP# 1 #UNTIL# 10 #DO# #BEGIN#
142              RMATRIX[I,J]:=ININT;
143              OUTINT(RMATRIX[I,J],10);  #END#;
147              OUTIMAGE;  #END#;
151         EJECT(5);
152         OUTTEXT(*LINE MATRIX*);
153         SPACING(3);
154         OUTIMAGE;
155         #FOR# I:=1 #STEP# 1 #UNTIL# 10 #DO# #BEGIN#
156         #FOR# J:=1 #STEP# 1 #UNTIL# 10 #DO# #BEGIN#
157              LMATRIX[I,J]:=ININT;
158              OUTINT(LMATRIX[I,J],10);  #END#;
162              OUTIMAGE;  #END#;
166         EJECT(5);
167         OUTTEXT(*DISTRIBUTION OF ORG.IMP NUMBER*);
168         SPACING(3);
169         OUTIMAGE;
170         #FOR# I:=1 #STEP# 1 #UNTIL# 10 #DO# #BEGIN#
171              ORGDIST[I]:=INREAL;
172         OUTFIX(ORGDIST[I],4,10);  #END#;
176         SPACING(3);
177         OUTIMAGE;
178         OUTTEXT(*DISTRIBUTION OF END.IMP NUMBER*);
179         SPACING(3);
180         OUTIMAGE;
181         #FOR# I:=1 #STEP# 1 #UNTIL# 10 #DO# #BEGIN#
182         #FOR# J:=1 #STEP# 1 #UNTIL# 10 #DO# #BEGIN#
183              MDIST[I,J]:=INREAL;
184              OUTFIX(MDIST[I,J],4,10);  #END#;
188              OUTIMAGE;  #END#;
192         #FOR# I:=1 #STEP# 1 #UNTIL# 28 #DO#
193              LQUE[I]:-#NEW# HEAD;
194         #FOR# I:=1 #STEP# 1 #UNTIL# 10 #DO#
195              BUFC[I]:=0;
196         #FOR# I:=1 #STEP# 1 #UNTIL# 28 #DO#
197              LSTATE[I]:=0;
```

```
198
198         #COMMENT# ************************************************;
199
199                                         #COMMENT# GENERATE NEW MESSAGE;
200
200         ORGIMP#   X#=UNIFORM(0.0,1.0,U1);
201         K#=1;
202         #FOR# I#=1 #STEP# 1 #UNTIL# 10 #DO# #BEGIN#
203               K#=I;
204               #IF# ORGDIST[I] #GREATER# X #THEN# #GO TO# LASTIMP; #END#;
209         LASTIMP#  X#=UNIFORM(0.0,1.0,U2);
210         J#=1;
211         #FOR# I#=1 #STEP# 1 #UNTIL# 10 #DO# #BEGIN#
212               J#=I;
213               #IF# MDIST[K,I] #GREATER# X #THEN# #GO TO# NEWMSG; #END#;
218         NEWMSG#   MESG#=MESG+1;
219         #IF# BUFC[K] #NOT LESS# BSIZE #THEN#
220               REJC#=REJC+1   #ELSE#  #BEGIN#
221               #ACTIVATE# #NEW# ACCEPT(K,J,0,1,TIME) #DELAY# 0.0;  #END#;
225         HOLD(NEGEXP(1.0/ARRIVT,U3));
226         #IF# TIME #LESS# cTIME #THEN# #GO TO# ORGIMP;
228         #COMMENT# ************************************************;
229
229                                         #COMMENT# GENERATE SIMULATION
229                                               REPORT;
230
230         EJECT(5);
231         OUTTEXT(+SIMULATION RESULT+);
232         SPACING(5);
233         OUTIMAGE;
234         OUTTEXT(+SIMULATION END TIME:+);
235         OUTFIX(TIME,2,10);
236         SPACING(2);
237         OUTIMAGE;
238         OUTTEXT(+TOTAL MESSAGE COUNT:+);
239         OUTINT(MESG,10);  OUTIMAGE;
241         OUTTEXT(+REJECTED MESSAGE COUNT:+);
242         OUTINT(REJC,10);  OUTIMAGE;
244         EJECT(5);
245         OUTTEXT(+THE STATISTICS ABOUT IMP+);   OUTIMAGE;
247         SPACING(5);
248         OUTIMAGE;
249         OUTTEXT(+IMP NUMBER    UTILIZATION  MESSAGE COUNT+);
250         SPACING(3);
251         OUTIMAGE;
252         #FOR# I#=1 #STEP# 1 #UNTIL# 10 #DO# #BEGIN#
253         X#=ST3[I]/TIME/BSIZE*100.0;
254         OUTINT(I,10);  OUTFIX(X,2,13);
256         OUTINT(MQIMP[I],15);  SPACING(2);  OUTIMAGE;  #END#;
262         EJECT(5);
263         OUTTEXT(+THE STATISTICS ABOUT LINE+);  OUTIMAGE;
265         SPACING(5);
266         OUTIMAGE;
267         OUTTEXT(+LINE NUMBER   UTILIZATION  MEAN LENGTH+);
268         OUTTEXT(+  MESSAGE COUNT+);  OUTIMAGE;
270         SPACING(3);
271         OUTIMAGE;
272         #FOR# I#=1 #STEP# 1 #UNTIL# 28 #DO#  #BEGIN#
```



```
273         X#=ST1[I]/TIME*100.0;
274         Y#=ST2[I]/TIME;
275         OUTINT(I,10);  OUTFIX(X,2,13);  OUTFIX(Y,2,13);
278         OUTINT(MQL[I],15);  SPACING(2);  OUTIMAGE;  #END#;
284         #END#;
287         #END#;
287             #EOP#
```

付表3.1　GPSS－V

```
BLOCK                                                                    STATEMENT
NUMBER  *LOC   OPERATION  A,B,C,D,E,F,G,H,I          COMMENTS            NUMBER
        *      SIMULATE
        ************************************************************      1
        *                                                          *      2
        *          NETWORK  MODEL                                  *      3
        *                                                          *      4
        ************************************************************      5
                                                                          6
        ROUTE  EQU          1,M                                           7
        CRCIT  EQU          2,M                                           8
        STION  EQU          11,Z                                          9
        INTVL  EQU          1,V                                           10
        MATYM  EQU          1,F,Q,X,H,XB,S      *** MATSUYAMA ***         11
        OSAKA  EQU          2,F,Q,X,H,XB,S      *** OSAKA ***             12
        FKOKA  EQU          3,F,Q,X,H,XB,S      *** FUKUOKA ***           13
        TOTRI  EQU          4,F,Q,X,H,XB,S      *** TOTTORI ***           14
        KNAZW  EQU          5,F,Q,X,H,XB,S      *** KANAZAWA ***          15
        TOKYO  EQU          6,F,Q,X,H,XB,S      *** TKYO ***              16
        NIGAT  EQU          7,F,Q,X,H,XB,S      *** NIIGATA ***           17
        SENDI  EQU          8,F,Q,X,H,XB,S      *** SENDAI ***            18
        SAPRO  EQU          9,F,Q,X,H,XB,S      *** SAPPORO ***           19
        KSIRO  EQU          10,F,Q,X,H,XB,S     *** KUSHIRO ***           20
        ****      VARIABLE DEFINE  ****                                   21
         1      VARIABLE    1000*1000/X60                                 22
        ****      FUNCTION DEFINE  ****                                   23
         1      FUNCTION    RN1,D9                                        24
        0.2307,1/0.3625,3/0.5803,4/0.6701,5/0.7250,6/0.9008,7/0.9337,8/0.9776,9   25
        1.0,10                                                            26
         2      FUNCTION    RN1,D9                                        27
        0.2470,1/0.4705,3/0.6587,4/0.7528,5/0.7763,6/0.9174,7/0.9409,8/0.9761,9   28
        1.0,10                                                            29
         3      FUNCTION    RN1,D9                                        30
        0.2033,1/0.5253,2/0.6951,4/0.7798,5/0.8136,6/0.9322,7/0.9491,8/0.9829,9   31
        1.0,10                                                            32
         4      FUNCTION    RN1,D9                                        33
        0.1022,1/0.1931,2/0.2499,3/0.4999,5/0.5737,6/0.9089,7/0.9429,8/0.9826,9   34
        1.0,10                                                            35
         5      FUNCTION    RN1,D9                                        36
        0.0729,1/0.1312,2/0.1672,3/0.4887,4/0.5908,6/0.8973,7/0.9337,8/0.9794,9   37
        1.0,10                                                            38
         6      FUNCTION    RN1,D9                                        39
        0.0683,1/0.0958,2/0.1231,3/0.3011,4/0.4928,5/0.8763,7/0.9173,8/0.9720,9   40
        1.0,10                                                            41
         7      FUNCTION    RN1,D9                                        42
        0.0754,1/0.1320,2/0.1650,3/0.4433,4/0.6414,5/0.7734,6/0.8535,8/0.9667,9   43
        1.0,10                                                            44
         8      FUNCTION    RN1,D9                                        45
        0.0535,1/0.0892,2/0.1070,3/0.2141,4/0.3033,5/0.3568,6/0.6498,7/0.9353,9   46
        1.0,10                                                            47
         9      FUNCTION    RN1,D9                                        48
        0.0579,1/0.1013,2/0.1302,3/0.2316,4/0.3185,5/0.3764,6/0.7242,7/0.9560,8   49
        1.0,10                                                            50
         10     FUNCTION    RN1,D9                                        51
        0.0769,1/0.1538,2/0.1922,3/0.3075,4/0.4228,5/0.4997,6/0.7689,7/0.8842,8   52
        0.9995,9                                                          53
         11     FUNCTION    RN1,DL0                                       54
        0.0924,1/0.1787,2/0.2386,3/0.4174,4/0.5566,5/0.6307,6/0.8461,7/0.903,8
        0.9731,9/1.0,10                                                   56
         12     FUNCTION    RN1,C6                                        57
        .0,0/.2511,50/.411,75/.6349,125/.7783,175/1.0,300                 58
```

```
****      INITIAL SET  ****                                                  59
          INITIAL    X60,200                                                 60
          INITIAL    X61,10                                                  61
****        MATRIX DEFINE  ****                                              62
**  ROUT MATRIX  **                                                          63
  1       MATRIX     MX,10,10                                                64
          INITIAL    MX1(1,1),0                                              65
          INITIAL    MX1(2,1),1                                              66
          INITIAL    MX1(3,1),1                                              67
          INITIAL    MX1(4,1),3                                              68
          INITIAL    MX1(5,1),4                                              69
          INITIAL    MX1(6,1),2                                              70
          INITIAL    MX1(7,1),6                                              71
          INITIAL    MX1(8,1),6                                              72
          INITIAL    MX1(9,1),7                                              73
          INITIAL    MX1(10,1),8                                             74
          INITIAL    MX1(1,2),2                                              75
          INITIAL    MX1(2,2),0                                              76
          INITIAL    MX1(3,2),1                                              77
          INITIAL    MX1(4,2),2                                              78
          INITIAL    MX1(5,2),4                                              79
          INITIAL    MX1(6,2),2                                              80
          INITIAL    MX1(7,2),6                                              81
          INITIAL    MX1(8,2),6                                              82
          INITIAL    MX1(9,2),8                                              83
          INITIAL    MX1(10,2),8                                             84
          INITIAL    MX1(1,3),3                                              85
          INITIAL    MX1(2,3),1                                              86
          INITIAL    MX1(3,3),0                                              87
          INITIAL    MX1(4,3),3                                              88
          INITIAL    MX1(5,3),4                                              89
          INITIAL    MX1(6,3),2                                              90
          INITIAL    MX1(7,3),5                                              91
          INITIAL    MX1(8,3),6                                              92
          INITIAL    MX1(9,3),7                                              93
          INITIAL    MX1(10,3),8                                             94
          INITIAL    MX1(1,4),3                                              95
          INITIAL    MX1(2,4),4                                              96
          INITIAL    MX1(3,4),4                                              97
          INITIAL    MX1(4,4),0                                              98
          INITIAL    MX1(5,4),4                                              99
          INITIAL    MX1(6,4),5                                             100
          INITIAL    MX1(7,4),5                                             101
          INITIAL    MX1(8,4),6                                             102
          INITIAL    MX1(9,4),7                                             103
          INITIAL    MX1(10,4),8                                            104
          INITIAL    MX1(1,5),3                                             105
          INITIAL    MX1(2,5),4                                             106
          INITIAL    MX1(3,5),4                                             107
          INITIAL    MX1(4,5),5                                             108
          INITIAL    MX1(5,5),0                                             109
          INITIAL    MX1(6,5),5                                             110
          INITIAL    MX1(7,5),5                                             111
          INITIAL    MX1(8,5),6                                             112
          INITIAL    MX1(9,5),7                                             113
          INITIAL    MX1(10,5),8                                            114
          INITIAL    MX1(1,6),2                                             115
          INITIAL    MX1(2,6),6                                             116
          INITIAL    MX1(3,6),4                                             117
          INITIAL    MX1(4,6),5                                             118
          INITIAL    MX1(5,6),6                                             119
          INITIAL    MX1(6,6),0                                             120
          INITIAL    MX1(7,6),6                                             121
          INITIAL    MX1(8,6),6                                             122
          INITIAL    MX1(9,6),8                                             123
          INITIAL    MX1(10,6),8                                            124
          INITIAL    MX1(1,7),2                                             125
          INITIAL    MX1(2,7),6                                             126
          INITIAL    MX1(3,7),4                                             127
          INITIAL    MX1(4,7),5                                             128
          INITIAL    MX1(5,7),7                                             129
          INITIAL    MX1(6,7),7                                             130
          INITIAL    MX1(7,7),0                                             131
          INITIAL    MX1(8,7),9                                             132
          INITIAL    MX1(9,7),7                                             133
          INITIAL    MX1(10,7),9                                            134
          INITIAL    MX1(1,8),2                                             135
          INITIAL    MX1(2,8),6                                             136
          INITIAL    MX1(3,8),6                                             137
```

```
                INITIAL     MX1(4,8),5                                                                  138
                INITIAL     MX1(5,8),6                                                                  139
                INITIAL     MX1(6,8),8                                                                  140
                INITIAL     MX1(7,8),9                                                                  141
                INITIAL     MX1(8,8),0                                                                  142
                INITIAL     MX1(9,8),8                                                                  143
                INITIAL     MX1(10,8),8                                                                 144
                INITIAL     MX1(1,9),2                                                                  145
                INITIAL     MX1(2,9),6                                                                  146
                INITIAL     MX1(3,9),4                                                                  147
                INITIAL     MX1(4,9),5                                                                  148
                INITIAL     MX1(5,9),7                                                                  149
                INITIAL     MX1(6,9),8                                                                  150
                INITIAL     MX1(7,9),9                                                                  151
                INITIAL     MX1(8,9),9                                                                  152
                INITIAL     MX1(9,9),0                                                                  153
                INITIAL     MX1(10,9),9                                                                 154
                INITIAL     MX1(1,10),2                                                                 155
                INITIAL     MX1(2,10),6                                                                 156
                INITIAL     MX1(3,10),4                                                                 157
                INITIAL     MX1(4,10),5                                                                 158
                INITIAL     MX1(5,10),7                                                                 159
                INITIAL     MX1(6,10),8                                                                 160
                INITIAL     MX1(7,10),9                                                                 161
                INITIAL     MX1(8,10),10                                                                162
                INITIAL     MX1(9,10),10                                                                163
                INITIAL     MX1(10,10),0                                                                164
          **  CIRCUIT MATRIX  **                                                                        165
            2   MATRIX      MX,20,10                                                                    166
                INITIAL     MX2(1,2),23                                                                 167
                INITIAL     MX2(1,3),24                                                                 168
                INITIAL     MX2(2,4),21                                                                 169
                INITIAL     MX2(2,6),20                                                                 170
                INITIAL     MX2(3,4),22                                                                 171
                INITIAL     MX2(4,5),19                                                                 172
                INITIAL     MX2(5,6),18                                                                 173
                INITIAL     MX2(5,7),17                                                                 174
                INITIAL     MX2(6,7),16                                                                 175
                INITIAL     MX2(6,8),15                                                                 176
                INITIAL     MX2(7,9),14                                                                 177
                INITIAL     MX2(8,9),13                                                                 178
                INITIAL     MX2(9,10),11                                                                179
                INITIAL     MX2(8,10),12                                                                180
                INITIAL     MX2(12,1),37                                                                181
                INITIAL     MX2(13,1),38                                                                182
                INITIAL     MX2(14,2),35                                                                183
                INITIAL     MX2(16,2),34                                                                184
                INITIAL     MX2(14,3),36                                                                185
                INITIAL     MX2(15,4),33                                                                186
                INITIAL     MX2(16,5),32                                                                187
                INITIAL     MX2(17,5),31                                                                188
                INITIAL     MX2(17,6),30                                                                189
                INITIAL     MX2(18,6),29                                                                190
                INITIAL     MX2(19,7),28                                                                191
                INITIAL     MX2(20,8),26                                                                192
                INITIAL     MX2(19,8),27                                                                193
                INITIAL     MX2(20,9),25                                                                194
          *****   BUFFER DEFINE                                                                         195
                STORAGE     S1,10/S2,10/S3,10/S4,10/S5,10                                               196
                STORAGE     S6,10/S7,10/S8,10/S9,10/S10,10                                              197
          ********** MESSAGE GENERATE ***************************************                          198
1               GENERATE    X61,FN12            *** MESSAGE GENERATE ***                                199
2               ASSIGN      2,FN$STION          *** ASSIGN DISPATCH STATION ***                         200
3               SAVEVALUE   P2+,1,XH                                                                    201
4               ASSIGN      3,FN*2              *** ASSIGN TERMINAL STATION ***                         202
5               SAVEVALUE   P3+,1,XB                                                                    203
6               ASSIGN      4,P2                *** ASSIGN RELAY STATION ***                            204
7         NET01 GATE   SNF  P4,NET06            *** OVERFLOW ***                                        205
          ********** IMP PROCESS ***************************************                               206
8               ENTER       P4                  *** ENTER BUFFER ***                                    207
9         NET10 QUEUE       P4                                                                          208
10              SEIZE       P4                  *** SEIZE IMP ***                                       209
11              DEPART      P4                  *** DEPART BUFFER ***                                   210
12              ADVANCE     100                 *** DELAY 100 MS ***                                    211
13              RELEASE     P4                  *** RELEASE IMP ***                                     212
14              TEST NE     P4,P3,NET05         *** TERMINAL STATION ? ***                              213
15              ASSIGN      5,MX&ROUTE(P4,P3)   *** NO ASSIGN NEXT STATION ***                          214
16              TEST L      P4,P5,NET02         *** P4 < P5 ? ***                                       215
17              TRANSFER    ,NET03              *** YES ***                                             216
18        NET02 INDEX       4,10                *** NO ***                                              217
19              ASSIGN      6,MX$CRCIT(P1,P5)   *** ASSIGN CIRCUIT ***                                  218
20              TRANSFER    ,NET04                                                                      219
21        NET03 ASSIGN      6,MX&CRCIT(P4,P5)   *** ASSIGN CIRCUIT ***                                  220
22        NET04 GATE   NU   P6,NET10            *** CIRCUIT USE ? ***                                   221
          ********** TRANSMIT ***************************************                                   222
23              SEIZE       P6                  *** SEIZE CIRCUIT ***                                   223
24              LEAVE       P4                  *** LEAVE BUFFER ***                                    224
25              ASSIGN      4,P5                                                                        225
```

```
26          ADVANCE     V$INTVL                 *** DELAY ***                                    226
27          RELEASE     P6                      *** RELEASE CIRCUIT ***                          227
28          TRANSFER    ,NET01                  *** ENTER BUFFER ***                             228
          ********** TERMINATE ***************************************************              229
29        NET05 LEAVE       P4                                                                   230
30          INDEX       4,10                                                                     231
31          SAVEVALUE   P1+,1                   *     COUNTER UP          *                      232
32          SAVEVALUE   30+,1                   *                         *                      233
33          TRANSFER    ,NET07                  *                         *                      234
          ********** OVERFLOW ************************************************                   235
34        NET06 SAVEVALUE   P4+,1               *     COUNTER UP          *                      236
35          TRACE                                                                                237
36          SAVEVALUE   31+,1                   *                         *                      238
37        NET07 SAVEVALUE   32+,1               *************************                        239
38          UNTRACE                                                                              240
39          TERMINATE                                                                            241
          ********** TIME CONTROL ********************************************                   242
40          GENERATE    ,,600000,1              *** 600 S ***                                    243
41          TERMINATE   1                                                                        244
            START       1                                                                        245
            RESET                                                                                246
            CLEAR       MX1,MX2                                                                  247
            INITIAL     X60,1200                                                                 248
            INITIAL     X61,1                                                                    249
            RMULT       37                                                                       250
            START       1                                                                        251
            REPORT                                                                               252
            EJECT                                                                                253
          FMS   TITLE       1,ROUTE MATRIX                                                       254
          FMS   TITLE       2,CIRCUIT MATRIX                                                     255
            EJECT                                                                                256
          ************************************                                                   257
          **                                *                                                   258
          **        DISPATCH STATION        *                                                   259
          **                                *                                                   260
          ************************************                                                   261
            SPACE       3                                                                        262
          5 FORMAT      1-10/XH1,XH2                                                             263
            EJECT                                                                                264
            GRAPH       XH,1,10                                                                  265
            ORIGIN      50,10                                                                    266
            X           SYM,2,8                                                                  267
            Y           0,10,20,2                                                                268
          50 STATEMENT  3,32,*=*=*= DISPATCH STATION =*=*=*                                      269
            ENDGRAPH                                                                             270
            EJECT                                                                                271
          ************************************                                                   272
          **                                *                                                   273
          **        TERMINAL STATION        *                                                   274
          **                                *                                                   275
          ************************************                                                   276
            SPACE       3                                                                        277
          5 FORMAT      1-10/XB1,XB2                                                             278
            EJECT                                                                                279
            GRAPH       XB,1,10                                                                  280
            ORIGIN      50,10                                                                    281
            X           SYM,2,8                                                                  282

            Y           0,10,20,2                                                                283
          50 STATEMENT  3,32,*=*=*= TERMINAL STATION =*=*=*                                      284
            ENDGRAPH                                                                             285
            EJECT                                                                                286
          ************************************                                                   287
          **                                *                                                   288
          **        BUFFER UTILIZATION      *                                                   289
          **                                *                                                   290
          ************************************                                                   291
            SPACE       3                                                                        292
          STO INCLUDE     S$MATYM-S$KSIRO/1,2,3,4,5,6,7,8                                        293
            EJECT                                                                                294
            GRAPH       SR,1,10                                                                  295
            ORIGIN      50,10                                                                    296
            X           SYM,2,8                                                                  297
            Y           .000,.050,20,2                                                           298
          50 STATEMENT  3,33,*=*=*= BUFFER UTILIZATION =*=*=*                                    299
            ENDGRAPH                                                                             300
            EJECT                                                                                301
          ************************************                                                   302
          **                                *                                                   303
          **        IMP UTILIZATION         *                                                   304
          **                                *                                                   305
          ************************************                                                   306
            SPACE       3                                                                        307
          FAC INCLUDE     F$MATYM-F$KSIRO/1,2                                                    308
            EJECT                                                                                309
            GRAPH       FR,1,10                                                                  310
            ORIGIN      50,10                                                                    311
            X           SYM,2,8                                                                  312
            Y           .000,.050,20,2                                                           313
          50 STATEMENT  3,30,*=*=*= IMP UTILIZATION =*=*=*                                       314
            ENDGRAPH                                                                             315
            EJECT                                                                                316
```

```
**************************************                                  317
**                                  *                                  318
**        CIRCUIT UTULIZATION       *                                  319
**                                  *                                  320
**************************************                                  321
        SPACE       3                                                   322
 FAC    INCLUDE     F11-F38/1,2,3                                       323
        EJECT                                                           324
        GRAPH       FR,11,28                                            325
        ORIGIN      50,10                                               326
        X           ,2,2                                                327
        Y           .000,.050,2C,2                                      328
 50     STATEMENT   3,33,*=*=*= CIRCUIT UTILIZATION =*=*=*              329
        ENDGRAPH                                                        330
        EJECT                                                           331
**************************************                                  332
**                                  *                                  333
**        OVERFLOW CONTENTS         *                                  334
**                                  *                                  335
**************************************                                  336
        SPACE       3                                                   337
  5     FORMAT      1-10/X1,X2                                          338
        EJECT                                                           339

        OUTPUT                                                          340
        EJECT                                                           341
        END                                                             342
```

ROUTE MATRIX
FULLWORD MATRIX ROUTE

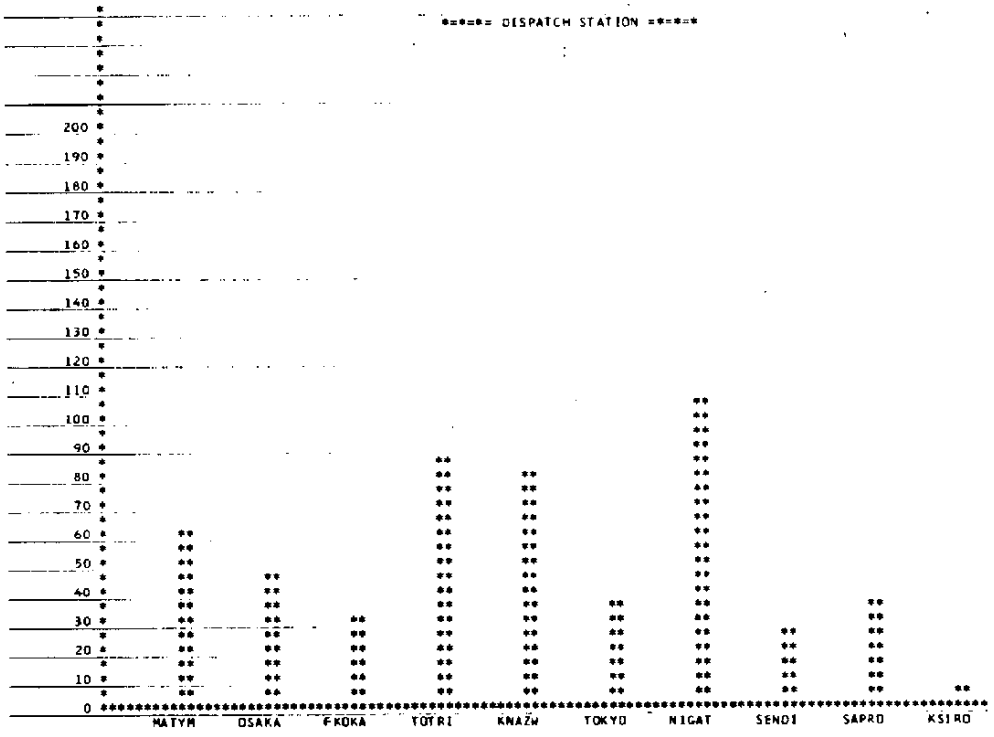
ROW/COLUMN	1	2	3	4	5	6	7	8	9	10
1	0	2	3	3	3	2	2	2	2	2
2	1	0	1	4	4	6	6	6	6	6
3	1	1	0	4	4	4	4	4	4	4
4	3	2	3	0	5	5	5	5	5	5
5	4	4	4	4	0	6	7	6	7	7
6	2	2	2	5	5	0	7	8	8	8
7	6	6	5	5	5	6	0	9	9	9
8	6	6	6	6	6	6	9	0	9	10
9	7	8	7	7	7	8	7	8	0	10
10	8	8	8	8	8	8	9	8	9	0

CIRCUIT MATRIX
FULLWORD MATRIX CKCIT

ROW/COLUMN	1	2	3	4	5	6	7	8	9	10
1	0	23	24	0	0	0	0	0	0	0
2	0	0	0	21	0	20	0	0	0	0
3	0	0	0	22	0	0	0	0	0	0
4	0	0	0	0	19	0	0	0	0	0
5	0	0	0	0	0	18	17	0	0	0
6	0	0	0	0	0	0	16	15	0	0
7	0	0	0	0	0	0	0	0	14	0
8	0	0	0	0	0	0	0	0	13	12
9	0	0	0	0	0	0	0	0	0	11
	ROWS 10-11, COLUMNS 1-10 ARE ZERO									
12	37	0	0	0	0	0	0	0	0	0
13	38	0	0	0	0	0	0	0	0	0
14	0	35	36	0	0	0	0	0	0	0
15	0	0	0	33	0	0	0	0	0	0
16	0	34	0	0	32	0	0	0	0	0
17	0	0	0	0	31	30	0	0	0	0
18	0	0	0	0	0	29	0	0	0	0
19	0	0	0	0	0	0	28	27	0	0
20	0	0	0	0	0	0	0	26	25	0

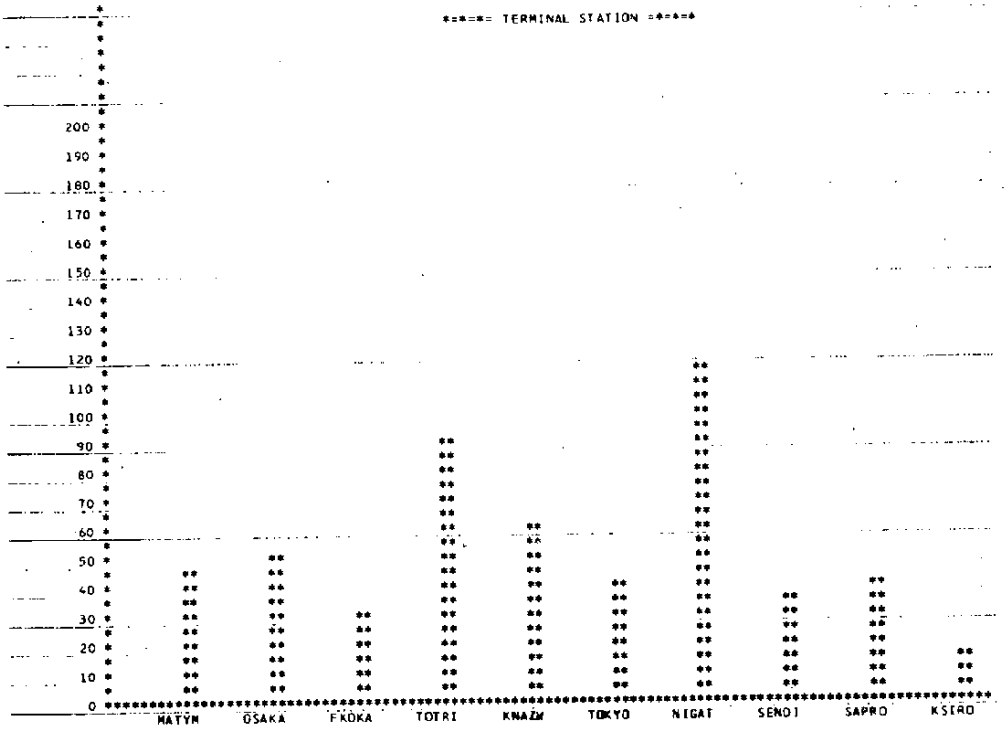
DISPATCH STATION

TYM	64
AKA	47
OKA	31
TRI	89
AZM	80
KYO	39
GAT	108
NDI	27
PRO	37
IRO	8



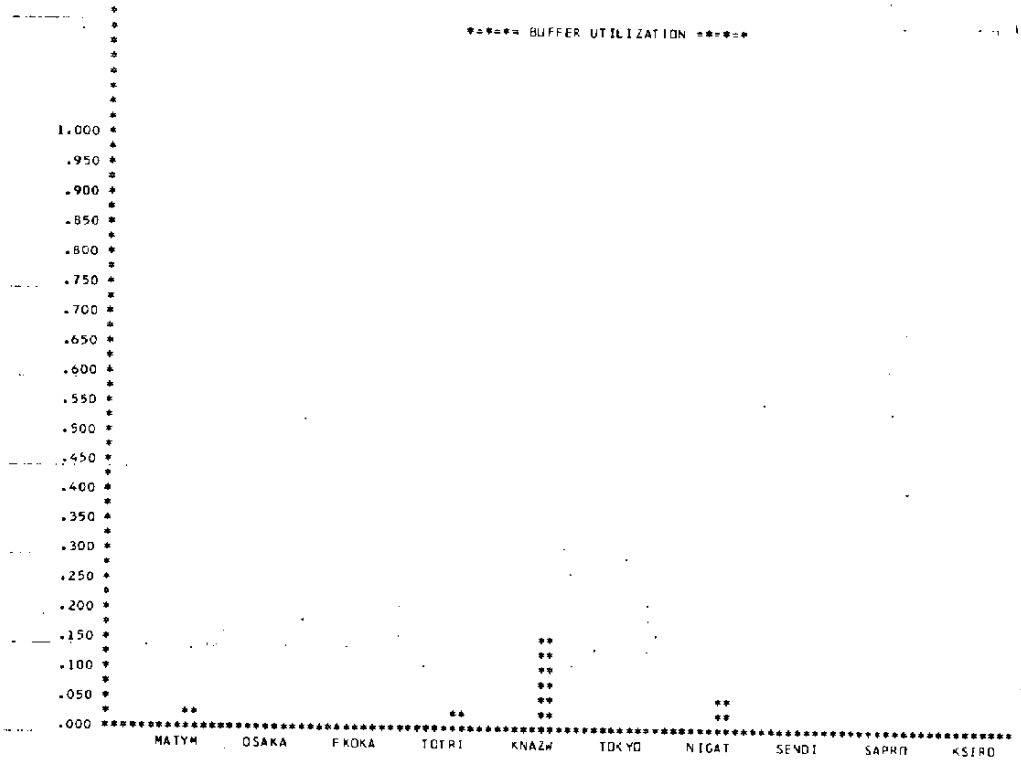
TERMINAL STATION

TYH	47
AKA	50
OKA	30
TRI	94
AZW	60
KYO	40
GAT	116
NDI	38
PRO	40
IRO	15



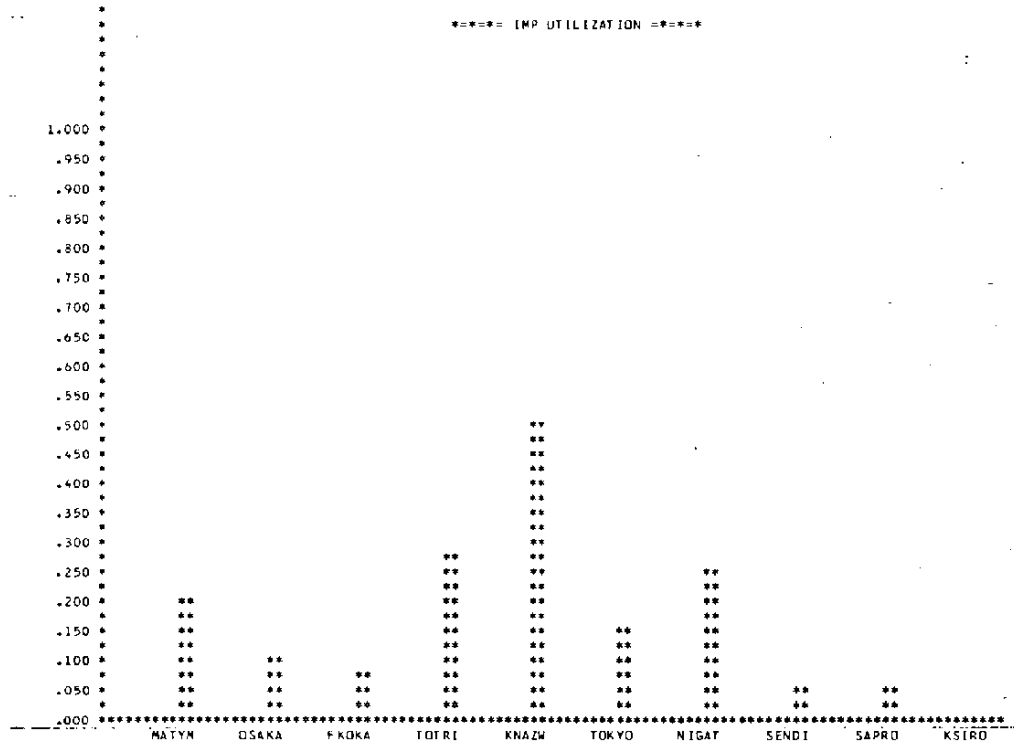
BUFFER UTILIZATION

STORAGE	CAPACITY	AVERAGE CONTENTS	AVERAGE UTILIZATION	ENTRIES	AVERAGE TIME/UNIT	CURRENT CONTENTS	MAXIMUM CONTENTS
MATYM	10	.318	.031	134	1427.619		4
OSAKA	10	.122	.012	143	513.699		2
FKOKA	10	.112	.011	85	791.093	2	4
TOTRI	10	.443	.044	206	1291.786		5
KNAZW	10	1.673	.167	230	4365.839		10
TOKYO	10	.230	.023	153	904.548	1	4
NIGAT	10	.548	.054	244	1349.253		7
SENDI	10	.091	.009	87	634.160		3
SAPRO	10	.085	.008	100	510.919		3
KSIRO	10	.003	.000	22	100.000		1



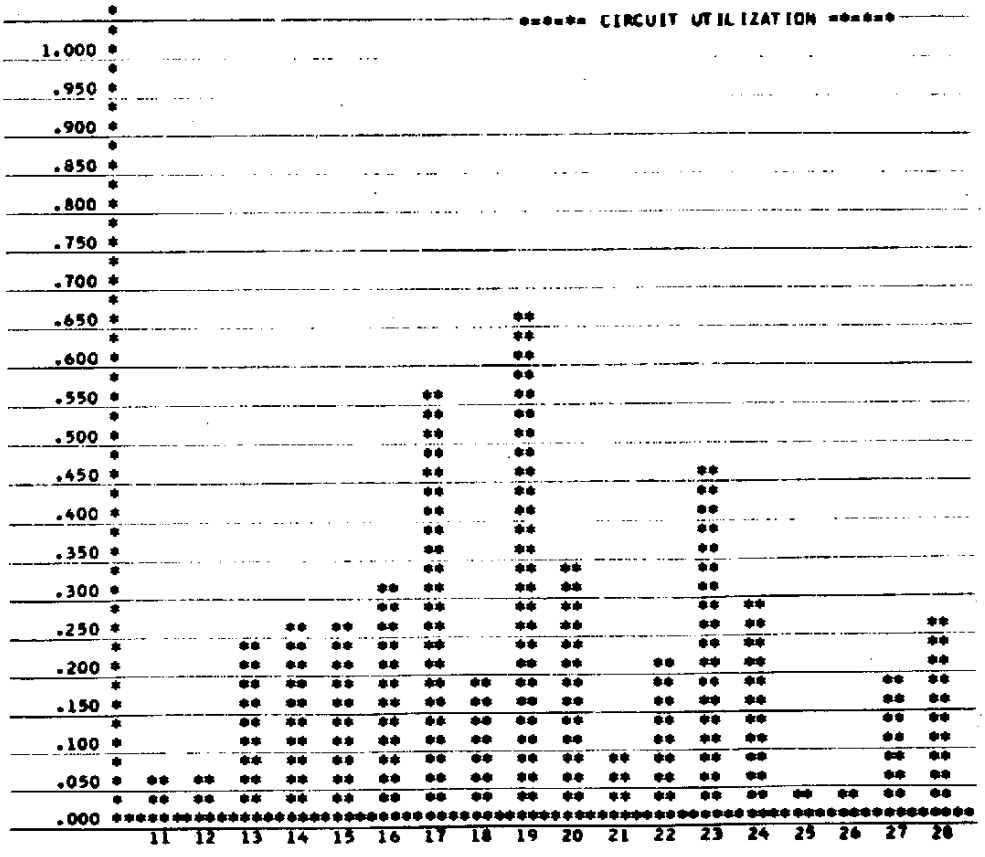
IMP UTILIZATION

FACILITY	AVERAGE UTILIZATION
MATYM	.212
OSAKA	.108
FKOKA	.088
TOTRI	.281
KNAZW	.503
TOKYO	.161
NIGAT	.261
SENDI	.054
SAPRO	.069
KSIRO	.003



CIRCUIT UTULIZATION

FACILITY	AVERAGE UTILIZATION	NUMBER ENTRIES
11	.058	7
12	.058	7
13	.233	28
14	.262	32
15	.274	33
16	.308	37
17	.566	68
18	.183	22
19	.663	80
20	.341	41
21	.099	12
22	.208	25
23	.458	55
24	.283	35
25	.033	4
26	.033	4
27	.191	23
28	.258	31
29	.116	14
30	.308	37
31	.496	60
32	.116	14
33	.670	81
34	.237	29
35	.108	13
36	.172	21
37	.337	41
38	.255	31



RELATIVE CLOCK 600000 ABSOLUTE CLOCK 600000

BLOCK COUNTS

BLOCK	CURRENT	TOTAL	BLOCK	CURRENT	TOTAL	BLOCK	CURRENT	TOTAL	BLOCK	CURRENT	TOTAL	BLOCK	CURRENT	TOTAL
1	0	530	11	0	10472	21	0	5570	31	0	516	41	0	1
2	0	530	12	2	10472	22	0	9954	32	0	516			
3	0	530	13	0	10470	23	0	885	33	0	516			
4	0	530	14	0	10470	24	0	885	34	0	2			
5	0	530	15	0	9954	25	0	885	35	0	2			
6	0	530	16	0	9954	26	9	885	36	0	2			
7	0	1406	17	0	5570	27	0	876	37	0	518			
8	0	1404	18	0	4384	28	0	876	38	0	518			
9	1	10473	19	0	4384	29	0	516	39	0	518			
10	0	10472	20	0	4384	30	0	516	40	0	1			

FACILITIES

FACILITY	NUMBER ENTRIES	AVERAGE TIME/TRAN	AVERAGE TOTAL TIME	UTILIZATION DURING AVAIL. TIME	UNAVAIL. TIME	CURRENT STATUS	PERCENT AVAILABILITY	TRANSACTION NUMBER SEIZING	PREEMPTING
MATYM	1277	100.000	.212				100.0		
OSAKA	651	100.000	.108				100.0		
FKOKA	531	99.636	.088				100.0	14	
TOTRI	1689	100.000	.281				100.0		
KNAZW	3022	100.000	.503				100.0		
TOKYO	969	99.914	.161				100.0	4	
NIGAT	1567	100.000	.261				100.0		
SENDI	326	100.000	.054				100.0		
SAPRO	418	100.000	.069				100.0		
KSIRO	22	100.000	.003				100.0		
11	7	5000.000	.058				100.0		
12	7	5000.000	.058				100.0		
13	28	5000.000	.233				100.0		
14	32	4915.093	.262				100.0	18	
15	33	5000.000	.274				100.0		
16	37	5000.000	.308				100.0		
17	68	5000.000	.566				100.0		
18	22	5000.000	.183				100.0		
19	80	4979.011	.663				100.0	17	
20	41	5000.000	.341				100.0		
21	12	5000.000	.099				100.0		
22	25	5000.000	.208				100.0		
23	55	5000.000	.456				100.0		
24	35	4861.597	.283				100.0	3	
25	4	5000.000	.033				100.0		
26	4	5000.000	.033				100.0		
27	23	5000.000	.191				100.0		
28	31	5000.000	.258				100.0		
29	14	5000.000	.116				100.0		
30	37	5000.000	.308				100.0		
31	60	4960.250	.496				100.0	13	
32	14	5000.000	.116				100.0		
33	81	4959.886	.670				100.0	15	
34	29	4919.343	.237				100.0	10	
35	13	5000.000	.108				100.0		
36	21	4924.808	.172				100.0	16	
37	41	4942.609	.337				100.0	5	
38	31	4946.257	.255				100.0	7	

STORAGES

STORAGE	CAPACITY	AVERAGE CONTENTS	ENTRIES	AVERAGE TIME/UNIT	AVERAGE TOTAL TIME	UTILIZATION DURING AVAIL. TIME	UNAVAIL. TIME	CURRENT STATUS	PERCENT AVAILABILITY	CURRENT CONTENTS	MAXIMUM CONTENTS
MATYM	10	.318	134	1427.619	.038				100.0		4
OSAKA	10	.122	143	513.699	.012				100.0		2
FKOKA	10	.112	85	791.093	.011				100.0	2	4
TOTRI	10	.443	206	1291.786	.044				100.0		6
KNAZW	10	1.673	230	4365.839	.167				100.0		10
TOKYO	10	.230	153	904.548	.023				100.0	1	4
NIGAT	10	.548	244	1349.253	.054				100.0		7
SENDI	10	.091	67	434.160	.009				100.0		3
SAPRO	10	.035	100	510.919	.008				100.0		3
KSIRO	10	.003	22	100.000	.000				100.0		1

QUEUES

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	PERCENT ZEROS	AVERAGE TIME/TRANS	$AVERAGE TIME/TRANS	TABLE NUMBER	CURRENT CONTENTS
MATYM	4	.106	1277	1250	97.8	49.805	2355.592		
OSAKA	2	.013	651	642	98.6	12.840	928.177		
FKOKA	4	.023	532	522	98.1	26.746	1422.859		
TOTRI	5	.162	1689	1642	97.2	57.553	2068.255		1
KNAZW	10	1.169	3022	2919	96.5	232.277	6814.988		
TOKYO	4	.069	969	948	97.8	42.909	1979.952		
NIGAT	7	.287	1567	1507	96.1	110.094	2875.299		
SENDI	3	.037	326	320	98.1	69.239	3762.000		
SAPRO	2	.015	418	408	97.6	22.229	929.199		
KSIRO	1	.000	22	22	100.0	.000	.000		

$AVERAGE TIME/TRANS = AVERAGE TIME/TRANS EXCLUDING ZERO ENTRIES

FULLWORD SAVEVALUES

NUMBER - CONTENTS	NUMBER - CONTENTS	NUMBER - CONTENTS	NUMBER - CONTENTS	NUMBER - CONTENTS	NUMBER - CONTENTS
KNAZW 2	11 44	12 49	13 27	14 92	15 59
16 39	17 115	18 38	19 39	20 14	30 516
31 2	32 518	60 200	61 10		

HALFWORD SAVEVALUES

NUMBER - CONTENTS	NUMBER - CONTENTS	NUMBER - CONTENTS	NUMBER - CONTENTS	NUMBER - CONTENTS	NUMBER - CONTENTS
MATYM 64	OSAKA 47	FKOKA 31	TOTRI 85	KNAZW 80	TOKYO 39
NIGAT 108	SENDI 27	SAPRO 37	KSIRO 8		

FULLWORD MATRICES

FULLWORD MATRIX ROUTE

ROW/COLUMN	1	2	3	4	5	6	7	8	9	10
1	0	2	3	3	3	2	2	2	2	2
2	1	0	1	4	4	6	6	6	6	6
3	1	1	0	4	4	4	4	4	4	4
4	3	2	3	0	5	5	5	5	5	5
5	4	4	4	4	0	6	7	6	7	7
6	2	2	2	5	5	0	7	8	8	8
7	6	6	5	5	5	6	0	4	9	9
8	6	5	6	6	6	6	9	0	9	10
9	7	8	7	7	7	8	7	8	0	10
10	8	8	8	8	8	8	9	8	9	0

FULLWORD MATRIX CREIT

ROW/COLUMN	1	2	3	4	5	6	7	8	9	10
1	0	23	24	0	0	0	0	0	0	0
2	0	0	0	21	0	20	0	0	0	0
3	0	0	0	22	0	0	0	0	0	0
4	0	0	0	0	19	0	0	0	0	0
5	0	0	0	0	0	13	17	0	0	0
6	0	0	0	0	0	0	16	15	0	0
7	0	0	0	0	0	0	0	0	14	0
8	0	0	0	0	0	0	0	0	13	12
9	0	0	0	0	0	0	0	0	0	11
	ROWS 10-11, COLUMNS 1-10 ARE ZERO									
12	37	0	0	0	0	0	0	0	0	0
13	38	0	0	0	0	0	0	0	0	0
14	0	45	36	0	0	0	0	0	0	0
15	0	0	0	33	0	0	0	0	0	0
16	0	34	0	0	32	0	0	0	0	0
17	0	0	0	0	31	30	0	0	0	0
18	0	0	0	0	0	29	0	0	0	0
19	0	0	0	0	0	0	28	27	0	0
20	0	0	0	0	0	0	0	26	25	0

BYTE SAVEVALUES

NUMBER - CONTENTS		NUMBER - CONTENTS		NUMBER - CONTENTS		NUMBER - CONTENTS		NUMBER - CONTENTS		NUMBER - CONTENTS	
MATYM	47	OSAKA	50	FKUKA	30	TOTRI	94	KNAZW	60	TOKYO	40
NIGAT	116	SENDI	38	SAPRO	40	KSIRO	15				

RESET
CLEAR MX1,MX2
INITIAL X60,1200
INETIAL K61,1
RMULT 37
START 1

TRANS *TRANS	17 FROM CUR BLOCK	35 TO NEXT BLOCK	36 CLOCK ADV BLK	8744 DEPART	TERMINATIONS TO GO PRIORITY	MARK TIME	1 ASSEM SET	SEL	TRACE	DELAY	CHAIN	PREEMPT	COUNT/FLAG
17	35	36		8744	0	7811	17		X		C		

HALFWORD PARAMETERS

	1	2	3	4	5	6	7	8	9	10
1 - 10	0	3	7	4	4	22	0	0	0	0
11 - 12	0	0								

TRANS *TRANS	17 FROM CUR BLOCK	36 TO NEXT BLOCK	37 CLOCK ADV BLK	8744 DEPART	TERMINATIONS TO GO PRIORITY	MARK TIME	1 ASSEM SET	SEL	TRACE	DELAY	CHAIN	PREEMPT	COUNT/FLAG
17	36	37		8744	0	7811	17		X		C		

HALFWORD PARAMETERS

	1	2	3	4	5	6	7	8	9	10
1 - 10	0	3	7	4	4	22	0	0	0	0
11 - 12	0	0								

TRANS *TRANS	17 FROM CUR BLOCK	37 TO NEXT BLOCK	38 CLOCK ADV BLK	8744 DEPART	TERMINATIONS TO GO PRIORITY	MARK TIME	1 ASSEM SET	SEL	TRACE	DELAY	CHAIN	PREEMPT	COUNT/FLAG
17	37	38		8744	0	7811	17		X		C		

HALFWORD PARAMETERS

	1	2	3	4	5	6	7	8	9	10
1 - 10	0	3	7	4	4	22	0	0	0	0
11 - 12	0	0								

TRANS *TRANS	7 FROM CUR BLOCK	35 TO NEXT BLOCK	36 CLOCK ADV BLK	8988 DEPART	TERMINATIONS TO GO PRIORITY	MARK TIME	1 ASSEM SET	SEL	TRACE	DELAY	CHAIN	PREEMPT	COUNT/FLAG
7	35	36		8988	0	8988	7		X		C		

HALFWORD PARAMETERS

	1	2	3	4	5	6	7	8	9	10
1 - 10	0	4	7	4	0	0	0	0	0	0
11 - 12	0	0								

TRANS *TRANS	7 FROM CUR BLOCK	36 TO NEXT BLOCK	37 CLOCK ADV BLK	8988 DEPART	TERMINATIONS TO GO PRIORITY	MARK TIME	1 ASSEM SET	SEL	TRACE	DELAY	CHAIN	PREEMPT	COUNT/FLAG
7	36	37		8988	0	8988	7		X		C		

HALFWORD PARAMETERS

	1	2	3	4	5	6	7	8	9	10
1 - 10	0	4	7	4	0	0	0	0	0	0
11 - 12	0	0								

TRANS *TRANS	7 FROM CUR BLOCK	37 TO NEXT BLOCK	38 CLOCK ADV BLK	8988 DEPART	TERMINATIONS TO GO PRIORITY	MARK TIME	1 ASSEM SET	SEL	TRACE	DELAY	CHAIN	PREEMPT	COUNT/FLAG
7	37	38		8988	0	8988	7		X		C		

HALFWORD PARAMETERS

	1	2	3	4	5	6	7	8	9	10
1 - 10	0	4	7	4	0	0	0	0	0	0
11 - 12	0	0								

TRANS *TRANS	7 FROM CUR BLOCK	35 TO NEXT BLOCK	36 CLOCK ADV BLK	9085 DEPART	TERMINATIONS TO GO PRIORITY	MARK TIME	1 ASSEM SET	SEL	TRACE	DELAY	CHAIN	PREEMPT	COUNT/FLAG
7	35	36		9085	0	9085	7		X		C		

HALFWORD PARAMETERS

	1	2	3	4	5	6	7	8	9	10
1 - 10	0	4	1	4	0	0	0	0	0	0
11 - 12	0	0								

TRANS *TRANS	7 FROM CUR BLOCK	36 TO NEXT BLOCK	37 CLOCK ADV BLK	9085 DEPART	TERMINATIONS TO GO PRIORITY	MARK TIME	1 ASSEM SET	SEL	TRACE	DELAY	CHAIN	PREEMPT	COUNT/FLAG
7	36	37		9085	0	9085	7		X		C		

HALFWORD PARAMETERS

	1	2	3	4	5	6	7	8	9	10
1 - 10	0	4	1	4	0	0	0	0	0	0
11 - 12	0	0								

TRANS #TRANS	7 FROM CUR BLOCK	37 TO NEXT BLOCK	38 CLOCK ADV BLK	9085 DEPART	TERMINATIONS TO GO PRIORITY	MARK TIME	1 ASSEM SET	SEL	TRACE	DELAY	CHAIN	PREEMPT COUNT/FLAG
7	37	38		9085	0	9085	7		X		C	

HALFWORD PARAMETERS

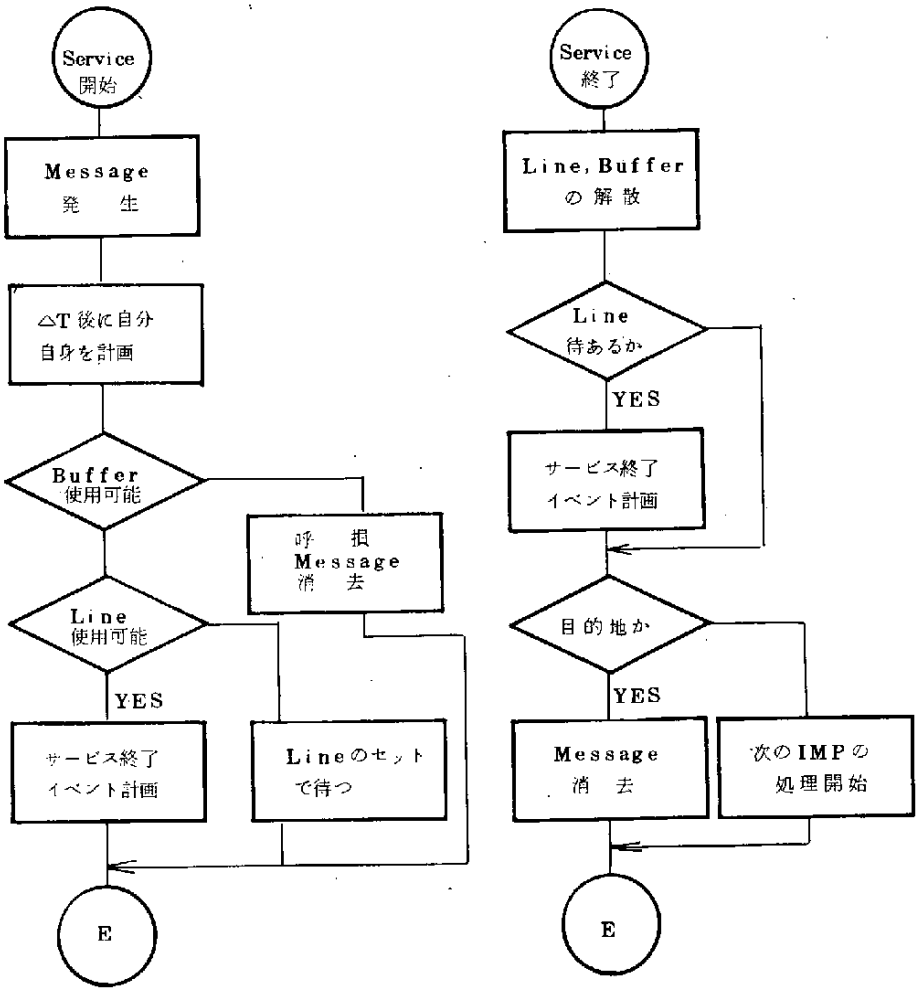
	1	2	3	4	5	6	7	8	9	10
1 - 10	0	4	1	4	0	0	0	0	0	0
11 - 12	0	0								

TRANS #TRANS	42 FROM CUR BLOCK	35 TO NEXT BLOCK	36 CLOCK ADV BLK	9429 DEPART	TERMINATIONS TO GO PRIORITY	MARK TIME	1 ASSEM SET	SEL	TRACE	DELAY	CHAIN	PREEMPT COUNT/FLAG
42	35	36		9429	0	9429	42		X		C	

HALFWORD PARAMETERS

4. イベントタイプ

付表 4.1、4.2は、付図4のイベント相互の関連図に基づく SIMSCRIPT－1.5、SIMSCRIPT－Ⅱのプログラムリストである。



付図4.　イベント相互の関連処理フロー

付表4.1　SIMSCRIPT－1.5

```
          6000 SIMSCRIPT   VERSION 2.1     MARCH 1970
 DEFINITION CARDS
          0         1         2         3         4         5         6         7         8
COLUMN    12345678901234567890123456789012345678901234567890123456789012345678901234567890

          +T MSG   8          T SORCE 1     I  1IMP    E          QUE 1 X          TRANSF
          +                   T QINT  2     F  2BUFS   1     I
          +                   T CURT  3     I  3ENT1   1     I
          +                   T DSTIN 4     I  4ENT2   1     I
          +                   T OCCPY 5     I  5ENT3   1     I
          +                   T GENT  6     F  6ENT4   1     I
          +                   T ELPST 7     F  7TOTAL  1     I
          +                   T SQUE  8     I  8ENTM   1     F
          +N MSGCL2                         I  9ENTRY  1     I
          +N ENDSV4           N NUMBR 3     I 10MAXQL  1     I
          +N SMOP12                           11AVAIL  1     F
          +N SMOP22                           12AVGQL  1     F
          +N SMOP32                           13NTWK   E
          +                                   14BLIMP  1     I
          +                                   15TRNST  1     I
          +                                   16ENTNO  1     I
          +                                   17MAXCQ  1     I
          +                                   18UTIL   1     F
          +                                   19LOST   1     I
          +                                   20STATE  1     I
          +                                   21MQUE   1     I
          +                                   22FQUE   1     I
          +                                   23LQUE   1     I
          +                                   24QUEIN  1     F
          +                                   25AVGCQ  1     F
          +                                   26MAXWT  1     F
          +                                   27BUF    1     I
          +                                   28CRCIT  E
          +                                   29FLOW   2     I
          +                                   30PROB   E
          +                                   31ODPRB  2     F
          +                                   32PRM    E
          +                                   33XPRM   2     I
          +                                   34YPRM   2     I
          +                                   35APRM   2     F
          +                                   36LPRM   2     I
          +                                   37CHAR1  E
          +                                   38MINA   1     F
          +                                   39MINB   1     I
          +                                   40CHAR2  E
          +                                   41MAXA   1     F
          +                                   42MAXB   1     I
```

```
EVENTS
    1       EXOGENOUS
            CNTL(1)
    5       ENDOGENOUS
                MSGCL
                ENDSV
            SMOP1
            SMOP2
            SMOP3
END
```

```
      EXOGENOUS  EVENT  CNTL
C
C     /*  SIMULATION  EXPERIMENTS  CONTROL  PROCEDURE  */
C
          SAVE
          READ  NO
          FORMAT(I6)
          IF (NO) EQ (0) , GO TO L5
          CALL  PRINT
          IF  (NO) EQ (1),GO TO L7
          LET  NO=NO-1
          DO TO L1, FOR I=(1)(28)
               LET XPRM(NO,I)=BUF(I)
               LET YPRM(NO,I)=TRNST(I)
      LET  APRM(NO,I) = UTIL(I)
               LET LPRM(NO,I)=LOST(I)
L1    LOOP
          IF (NO) EQ (3) , GO TO L3
          IF  (NO) EQ (1),GO TO L7
          DO TO L2, FOR I=(1)(28)
               LET  BUF ( I ) = 8
               LET  TRNST(I)=200
L2    LOOP
          GO TO L7
L3        DO TO L4,FOR I=(1)(28)
               LET  BUF ( I ) = 6
               LET  TRNST(I)=1200
L4    LOOP
          GO TO L7
L5        DO TO L6,FOR I=(1)(28)
               LET  BUF ( I ) = 8
               LET  TRNST(I)=1200
L6    LOOP
L11       CALL RESET
          CREATE  MSGCL
          CAUSE  MSGCL  AT TIME
          GO TO L10
L7        IF  (NO) NE (3),GO TO L11
          CREATE  SMOP1
          CAUSE  SMOP1  AT TIME+0.3
L10   CALL  BCON
      RETURN
      END
```

```
      ENDOGENOUS  EVENT  SMOP1
C
C     /*  DETERMINE  PARAM  MIN  +  MAX  */
C
                  DESTROY  SMOP1
      DO TO L8 ,FOR I=(1)(28)
                  LET  IDX=APRM(1,I)
                  IF  (IDX) LE (APRM(2,I)), GO TO L4
                  IF  (IDX) LE (APRM(3,I)), GO TO L1
                       LET  MINA(I)=APRM(3,I)
                       LET  MINB(I)=3
                       GO  TO  L2
 L1                    LET  MINA(I)=IDX
                       LET  MINB(I)=1
                  IF  (APRM(2,I)) LE (APRM(3,I)), GO TO L3
 L2                    LET  MAXA(I)=APRM(2,I)
                       LET  MAXB(I)=2
                       GO  TO  L8
 L3                    LET  MAXA(I)=APRM(3,I)
                       LET  MAXB(I)=3
                       GO  TO  L8
 L4                IF  (APRM(2,I)) LE (APRM(3,I)), GO TO L5
                       LET  MINA(I)=APRM(3,I)
                       LET  MINB(I)=3
                       GO  TO  L6
 L5                    LET  MINA(I)=APRM(2,I)
                       LET  MINB(I)=2
                  IF  (IDX) LE (APRM(3,I)), GO TO L7
 L6                    LET  MAXA(I)=IDX
                       LET  MAXB(I)=1
                       GO  TO  L8
 L7                    LET  MAXA(I)=APRM(3,I)
                       LET  MAXB(I)=3
 L8   LOOP
C
C     /*  REFLECTION  POINT  SELECTION  */
C
            CALL  RESET
           LET  IMPO=0
      DO TO L18,FOR I=(1)(28)
                  LET  J=MINB(I)
                  LET  MINY=YPRM(J,I)
                  LET  MINX=XPRM(J,I)
                  LET  NX=0
           DO TO L9,FOR L=(1)(3)
                  IF  (MINX) LS (XPRM(L,I)),LET NX=NX+1
 L9        LOOP
                  LET  NY=0
           DO TO L10,FOR L=(1)(3)
                  IF  (MINY) EQ (YPRM(L,I)),LET NY=NY+1
 L10       LOOP
                  IF  (NY) EQ (1),GO TO L12
                  IF  (NY) EQ (2),GO TO L13
                       IF  (NX) LS (2),GO TO L17
 L11                   LET  XPRM(4,I)=MINX-2
                  LET  BUF(I) = MINX-2
                       GO TO L15
 L12              IF  (NX) NE (0),GO TO L11
                       LET  XPRM(4,I)=MINX+2
                  LET  BUF(I) = MINX+2
                       GO TO L15
 L13              IF  (NX) EQ (0),GO TO L14
                       LET  XPRM(4,I)=MINX-1
                  LET  BUF(I) = MINX-1
                       GO TO L15
 L14                   LET  XPRM(4,I)=MINX+1
                  LET  BUF(I) = MINX+1
 L15              IF  (MINY) EQ (200),GO TO L16
                       LET  YPRM(4,I)=200
                  LET  TRNST(I) =200
                       GO  TO  L17
 L16                   LET  YPRM(4,I)=1200
                  LET  TRNST(I) =1200
                       GO  TO L18
 L17       LET   IMPO=IMPO+1
 L18  LOOP
           IF  (IMPO) GE (28), STOP
           CREATE  SMOP2
      CAUSE SMOP2 AT TIME + 0.3
      CALL  BCON
      RETURN
      END
```

```
      ENDOGENOUS  EVENT  SMOP2
C
C     /*  EXPANSION  POINT  SELECTION  */
C
          DESTROY  SMOP2
          CALL  PRINT
      DO TO L1,FOR I=(1)(28)
      LET  APRM(4,I) = UTIL(I)
               LET  LPRM(4,I)=LOST(I)
 L1   LOOP
      DO TO L6,FOR I=(1)(28)
               IF  (APRM(4,I)) GE (MAXA(I)),GO TO L2
               IF  (APRM(4,I)) LE (MINA(I)),GO TO L3
                    LET  XPRM(MINB(I),I)=XPRM(4,I)
                    LET  YPRM(MINB(I),I)=YPRM(4,I)
                    LET APRM(MINB(I),I)=APRM(4,I)
                    LET  LPRM(MINB(I),I)=LPRM(4,I)
                    GO TO L3
 L2        LET  J=0
      DO TO L3,FOR  L=(1)(3)
               IF  (XPRM(MINB(I),I)) LS (XPRM(L,I)),LET J=J+1
 L3   LOOP
          CALL  RESET
               IF  (J) EQ (0), GO TO L4
                    LET  XPRM(5,I)=XPRM(4,I)-1
               LET  BUF(I)  = XPRM(5,I)
                    GO TO L5
 L4                 LET  XPRM(5,I)=XPRM(4,I)+1
               LET  BUF(I)  = XPRM(5,I)
 L5                 LET  YPRM(5,I)=YPRM(4,I)
               LET  TRNST(I) = YPRM(5,I)
 L6   LOOP
          CREATE  SMOP3
      CAUSE SMOP3 AT TIME + 0.3
      CALL  BCON
          RETURN
      END
```

```
      ENDOGENOUS  EVENT  SMOP3
C
C     /*  CONSTRICTION  JUDGMENT  */
C
          DESTROY  SMOP3
          CALL     PRINT
          DO TO L1,FOR I=(1)(28)
      LET  APRM(5,I) = UTIL(I)
               LET  LPRM(5,I)=LOST(I)
 L1        LOOP
               LET  LOSS=0
          DO TO L3,FOR I=(1)(28)
               IF (LPRM(5,I)) GR (1),GO TO L2
               IF  (APRM(5,I)) LE (APRM(4,I)),GO TO L3
                    LET  XPRM(MINB(I),I)=XPRM(5,I)
                    LET  YPRM(MINB(I),I)=YPRM(5,I)
                    LET  APRM(MINB(I),I)=APRM(5,I)
                    LET  LPRM(MINB(I),I)=LPRM(5,I)
                    GO TO L3
 L2             LET  LOSS=LOSS+1
 L3        LOOP
               IF  (LOSS) EQ (28),STOP
               LET  MOR=0
          DO TO L4,FOR I=(1)(28)
               IF  (APRM(MINB(I),I)) LS (70),GO TO L4
                    LET  MOR=MOR+1
 L4        LOOP
               IF  (MOR) EQ (28),STOP
          CREATE  SMOP1
          CAUSE  SMOP1  AT TIME
       RETURN
      END
```

```
      ENDOGENOUS  EVENT  ENDSV
         LET  NUM=NUMBR(ENDSV)
              DESTROY  ENDSV
         REMOVE FIRST  MSG  FROM  QUE(NUM)
              LET  IND1=BLIMP(NUM)
      LET  X = ENTRY(IND1)
      LET  Y = QUEIN(NUM) - QINT(MSG)
      IF Y GR MAXWT(NUM), LET MAXWT(NUM) = Y
      LET  AMQUE = MQUE(NUM)
      LET  A11 = 1.
      LET  QQQ = QUEIN(NUM)
      LET  EEE = ENTM(IND1)
      ACC A11, AMQUE INTO UTIL(NUM) , AVGCQ(NUM) SINCE QUEIN(NUM)
     1,QQQ
      ACC X , X INTO AVAIL(IND1) ,  AVGQL(IND1) SINCE ENTM(IND1)
     1,EEE
      IF MAXQL(IND1) LS ENTRY(IND1) , LET  MAXQL(IND1) = ENTRY(IND1)
              LET  IND2=ENTRY(BLIMP(NUM))
                   LET  ENTRY(BLIMP(NUM))=IND2 - 1
              LET  IND3=CURT(MSG)
              LET  IND4=OCCPY(MSG)
              LET  Y=MQUE(IND4)
         IF MAXCQ(IND4) LS MQUE(IND4),LET MAXCQ(IND4)=MQUE(IND4)
              LET  MQUE(NUM)=MQUE(NUM) - 1
      LET  ENTRY( IND3) = ENTRY( IND3 ) -1
              IF  CURT(MSG) EQ DSTIN(MSG),GO TO L1
C     /*  NEXT  SERVICE  START  */
              LET  M=DSTIN(MSG)
                   CALL  SELCT(IND3,M,*ICRTN,*IMPN)
                   LET  OCCPY(MSG)=ICRTN
                   LET CURT(MSG)=IMPN
      CALL  SRVST(IND3,MSG)
              GO TO L2
 L1   DESTROY  MSG
 L2   CALL  CHECK ( IND3, NUM )
      RETURN
      END


      ENDOGENOUS  EVENT  MSGGL
C     /*  MESSAGE  GENERATION  */
         CREATE  MSG
              CALL  MSSGE(*IORG,*IDST,*ARVT)
                 LET  SORCE(MSG)=IORG
                 LET  DSTIN(MSG)=IDST
                 LET  GENT(MSG)=TIME
              CALL  SELCT(IORG,IDST,*ICRTN,*IMPN)
C     /*  OCCUPANCY LINE NO */
                 LET  OCCPY(MSG)=ICRTN
                 LET  CURT(MSG)=IMPN
         CAUSE  MSGGL  AT  TIME + ARVT
              CALL       SRVST(IORG,MSG)
         RETURN
      END
```

```
      SUBROUTINE  BCON
C     /*  IMP  BUFFER  CONSTRUCTION  */
            DO TO L1,FOR I=(1)(NIMP)
                  LET  BUFS(I)=0
 L1         LOOP
            DO TO L2,FOR I=(1)(NIMP)
                  LET  BUFS(I)=BUFS(I)+BUF(ENT1(I))+BUF(ENT2(I))
                  IF  (ENT3(I)) EQ (0),GO TO L2
                  LET  BUFS(I)=BUFS(I)+BUF(ENT3(I))
                  IF  (ENT4(I)) EQ (0),GO TO L2
                  LET  BUFS(I)=BUFS(I)+BUF(ENT4(I))
 L2         REPEAT
      RETURN
      END


      SUBROUTINE  PRINT
C
C     /*  STATISTICAL  QUANTITIES  PRINT  OUT  */
C
      DO TO L3, FOR I=(1)(NNTWK)
      LET  UTIL(I) = UTIL(I) / 0.3
      LET  AVGCQ(I) = AVGCQ(I) / 0.3
   L3 LOOP
      DO TO L4, FOR I=(1)(NIMP)
      LET  ABUFS = BUFS(I)
      LET  AVAIL ( I ) = AVAIL ( I ) / 0.3  * ABUFS
      LET  AVGQL(I) = AVGQL(I) / 0.3
   L4 LOOP
         CALL  STATC
      RETURN
      END


      FUNCTION  TRANS(LCRT)
                  LET  Y=TRNST(LCRT)
                  LET  TRANS = 1.0 / Y
         RETURN
      END


      SUBROUTINE  CHECK ( JCRT , IBLG )
  NXT             IF  QUE(IBLG) IS EMPTY,GO TO EXIT
      IF  ENTRY ( JCRT ) GE BUFS ( JCRT ) , GO TO EXIT
                  LET  ENTRY(JCRT)=ENTRY(JCRT) + 1
                  CREATE  ENDSV
                     LET  NUMBR(ENDSV)=IBLG
      WRITE ON 61,IBLG,ENDSV
      FORMAT(*   CHECK*I10,D10)
                  CAUSE  ENDSV  AT  TIME + TRANS(IBLG)
 EXIT     RETURN
      END
```

```
      SUBROUTINE  MSSGE(IORG,IOST,ARV)
           LET  VAL=RANDM
                IF  VAL GE 0.0924,GO TO L1
                    LET  IORG=1
                    GO TO NEXT
   L1           IF  VAL GE 0.1787,GO TO L2
                     LET  IORG=2
                     GO TO NEXT
   L2           IF  VAL GE 0.2386,GO TO L3
                     LET  IORG=3
                     GO TO NEXT
   L3           IF  VAL GE 0.4174,GO TO L4
                     LET  IORG=4
                     GO TO NEXT
   L4           IF  VAL GE 0.5566,GO TO L5
                     LET  IORG=5
                     GO TO NEXT
   L5           IF  VAL GE 0.6307,GO TO L6
                     LET  IORG=6
                     GO TO NEXT
   L6           IF  VAL GE 0.8461,GO TO L7
                     LET  IORG=7
                     GO TO NEXT
   L7           IF  VAL GE 0.9030,GO TO L8
                     LET  IORG=8
                     GO TO NEXT
   L8           IF  VAL GE 0.9731,GO TO L9
                     LET  IORG=9
                     GO TO NEXT
   L9                LET  IORG=10
  NEXT      LET  VAL = RANDM
            DO TO L20,FOR I = (1)(10)
                LET  Y=ODPRB(IORG,I)
                 IF  VAL  GR  Y,GO TO L20
                LET  IOST=I
                GO TO EXIT
  L20       REPEAT
  EXIT          LET  R = RANDM
                LET  ARV = -ALOG ( R ) /1000.
            RETURN
        END


      SUBROUTINE  SELCT(IORG,IOST,ICRT,IMPN)
           LET  ICRT=FLOW(IORG,IOST*2)
           LET  IMPN=FLOW(IORG,IOST*2-1)
        RETURN
      END
```

```
      SUBROUTINE  SRVST(IOG,MSG)
                  LET   IJ=OCCPY(MSG)
      LET X = ENTRY(IOG)
      LET  AMQUE = MQUE(IJ)
                  LET   ENTRY(IOG)=ENTRY(IOG) + 1
                  LET   QINT(MSG)=TIME
                  LET   MQUE(IJ)=MQUE(IJ) + 1
                  LET   TOTAL(IOG) = TOTAL(IOG) + 1
                  LET   ENTNO(IJ) = ENTNO(IJ) + 1
            IF  BUF(IJ) GR MQUE(IJ),GO TO L0
                  LET   LOST(IJ) = LOST(IJ) + 1
                  DESTROY  MSG
                  LET   MQUE(IJ) = MQUE(IJ)-1
                  GO TO L3
 L0               FILE  MSG  IN  QUE(IJ)
      LET  KK  = CURT ( MSG )
      IF  ENTRY ( KK ) GE BUFS ( KK ) , GO TO L3
                  IF MQUE(IJ) LS 2,GO TO L1
                  GO TO L3
 L1                     LET  ENTRY(CURT(MSG))=ENTRY(CURT(MSG)) + 1
                  CREATE  ENDSV
                  LET   NUMBR(ENDSV)=IJ
                  CAUSE  ENDSV  AT  TIME + TRANS(IJ)
      LET  A11 = 0
      LET  QQQ = QUEIN(IJ)
      LET  EEE = ENTM(IOG)
      ACC A11, AMQUE INTO UTIL(IJ) , AVGQ2(IJ) SINCE QUEIN(IJ)
     1,QQQ
      ACC X , X INTO AVAIL(IOG) , AVGQL(IOG) SINCE ENTM(IOG)
     1,EEE
   L3 RETURN
      END


      SUBROUTINE  RESET
C
C     /*  STATISTICAL  VALUE  RESET  */
C
 L7        DO TO L8,FOR I=(1)(10)
                  LET  TOTAL(I)=0
      LET  ENTM(I) = 0
                  LET  ENTRY(I)=0
                  LET  MAXQL(I)=0
                  LET  AVAIL(I)=0.0
                  LET  AVGQL(I)=0.0
 L8   LOOP
           DO TO L9,FOR I=(1)(28)
                  LET  ENTNO(I)=0
                  LET  MAXCQ(I)=0
      LET  UTIL(I) = 0
                  LET  LOST(I) =0
                  LET  QUEIN(I)=0.0
                  LET  AVGCQ(I)=0.0
                  LET  MAXNT(I)=0.0
 L9   LOOP
        RETURN
      END
```

```
     REPORT  STATC

FX C  R  CR      H  S3 NS  SP

+++++ N E T W O R K    F L O W    S I M U L A T I O N +++++
FX C  R  CR      H  S3 NS  SP

                                                    +++++ END TIME = ***.*** +++++
FX C  R  CR      H  S3 NS  SP

       TIME
F  CX R  CR      H  S  NS  SP

         +++++ STATISTICAL  QUANTITIES  OF  INTERMEDIATE  PROCESSOR +++++
FX C  R  CR      H  S2 NS  SP

IMP.NO  BUFFER  ENTER-CIRCUIT  TOTAL-ENTRY  AVAILABILITY  AVERAGE-QLENGTH  MAXIMUM-QLENGTH
FX C  R  CR      H  S3 NS  SP

    **    ****    ** ** ** **        *****       *.*****          ***.***            *****
FX C  R  CR      H  S2 NS  SP

    I  BUFS(I) ENT1(I) ENT2(I) ENT3(I) ENT4(I) TOTAL(I) AVAIL(I)   AVQ3(I) MAXQL(I)
F  CX R  CR      H  S  NS  SP

FOR  I=(1)(10)
F  C  RX CR      H  S  NS  SP

         +++++ NETWORK  CONECTING  CIRCUIT  STATISTICS +++++
FX C  R  CR      H  S3 NSX SP

CIRCUIT.NO  BELONG.IMP  SPEED(BAUD)  ENTRY-NO  AVRG.QL  MAX.QL     MAX.WAIT  LOST-CALL    UTILITY    BUFSIZE
FX C  R  CR      H  S3 NS  SP

       **          **          ****        ****  ***.***    ***       ***.***        ****    *.*****         **
FX C  R  CR      H  S  NS  SP

       J     BLIMP(J)    TRNST(J)    ENTNO(J)  AVQ3(J) MAXQ3(J)   MAXWT(J)    LOST(J)   UTIL(J)   BUF(J)
F  CX R  CR      H  S  NS  SP

FOR  J=(1)(28)
F  C  RX CR      H  S  NS  SP

FOR EACH CRCIT K
F  C  R  CR20    H  S  NS  SP

         +++++ ANOTHER  CIRCUIT  INFORMATION +++++
FX C  R  CR      H  S3 NS  SP

         +++++                                +++++
FX C  R  CR      H  S  NS  SP

         +  MESSAGE LENGTH=1000(BIT)   CONSTANT +
FX C  R  CR      H  S  NS  SP

         +  TRANSFER SPEED IS VARIABLE  (BAUD)  +
FX C  R  CR      H  S  NS  SP

         +  IMP BUFFER-SIZE IS VARIABLE(/PACKET)+
FX C  R  CR      H  S  NS  SP

         +  MSG CALL IS POISSON DISTRIBUTION    +
```

```
FX C  R  CR     H  S  NS  SP

         +          MEAN IS 1.0(NUMBER/SEC)       +
FX C  R  CR     H  S  NS  SP

         +++++                                    +++++
FX C  R  CR     H  S2 NS  SP

         +++++ NETWORK PATH SELECTION TABLE +++++
FX C  R  CR     H  S2 NS  SP

               (CIRCUIT NO.,PASSING IMP NO.)
FX C  R  CR     H  S2 NS  SP

COLUMN    **    **  **    **  **    **  **    **  **    **  **    **  **    **  **    **  **    **  **    **  **    **
FX C  R  CR     H  S  NS  SP

    20(K)
F  CX R  CR     H  S  NS  SP

   ROW
FX C  R  CR     H  S  NS  SP

    **    (**,**)  (**,**)  (**,**)  (**,**)  (**,**)  (**,**)  (**,**)  (**,**)  (**,**)  (**,**)
FX C  R  CR     H  S  NS  SP

    L    20(FLOW(L,K))
F  CX R  CR     H  S  NS  SP

FOR EACH IMP L
F  C  RX CR     H  S  NS  SP

FOR EACH PROB N
F  C  R  CR10   H  S  NS  SP

         +++++ NETWORK SELECTION PROBABILITY TABLE +++++
FX C  R  CR     H  S3 NS  SP

COLUMN    **        **        **        **        **        **        **        **        **        **
FX C  R  CR     H  S  NS  SP

     10(N)
F  CX R  CR     H  S  NS  SP

   ROW
FX C  R  CR     H  S  NS  SP

    **  *.****  *.****  *.****  *.****  *.****  *.****  *.****  *.****  *.****  *.****
FX C  R  CR     H  S  NS  SP

    M   10(100PRB(M,N))
F  CX R  CR     H  S  NS  SP

FOR EACH IMP M
F  C  RX CR     H  S  NS  SP

                    END                                                              END
F  C  R  CR     H  S  NS  SP
```

```
 INITIALIZATION CARDS
11          42    100     10
  1       0 R                                                    10
  2       1 R     10    1                                        10(I3)
 10 10 10 10 10 10 10 10 10 10
  3       1 R     10    1                                        10(I3)
 27 13 14 11  9  8  6  5  3  1
  4       1 R     10    1                                        10(I3)
 28 24 26 12 21 10  7 16  4  2
  5       1 R     10    1                                        10(I3)
  0 25  0 23 22 19 18 17 15  0
  6       1 R     10    1                                        10(I3)
 0  0  0  0  0  20 0  0  0  0
  7  12 1  Z    10    1
 13       0 R                                                    28
 14       1 R     28  13                                         28(I2)
 9 8 8 7 6 6 5 5 4 2 2 3 1 11010 9 9 8 7 7 6 5 6 4 4 2 3
 15       1 R     28  13                                         17(I4)
 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200 200
 200 200 200 200 200 200 200 200 200 200 200
 16  27 1  Z    28  13
 28       0 R                                                    20
 29       2 R     10    1  20  28      R  F                      20(I3)
 0  0   2 13  3 14  3 14  3 14  2 13  2 13  2 13  2 13  2 13
 1 27   0  0  1 27  4 11  4 11  6 10  6 10  6 10  6 10  6 10
 1 28   1 28  0  0  4 12  4 12  4 12  4 12  4 12  4 12  4 12
 3 26   2 25  3 26  0  0  5  9  5  9  5  9  5  9  5  9  5  9
 4 23   4 23  4 23  4 23  0  0  6  8  7  7  6  8  7  7  7  7
 2 24   2 24  2 24  5 22  5 22  0  0  7  6  8  5  8  5  8  5
 6 20   6 20  5 21  5 21  5 21  6 20  0  0  9  4  9  4  9  4
 6 19   6 19  6 19  6 19  6 19  6 19  9  3  0  0  9  3 10  2
 7 18   8 17  7 18  7 18  7 18  8 17  7 18  8 17  0  0 10  1
 8 16   8 16  8 16  8 16  8 16  8 16  9 15  8 16  9 15  0  0
 30       0 R                                                    10
 31       2 R-   10    1  10  30      R  F                      10(D2.4)
0.0     0.2307 0.3625 0.5603 0.6701 0.7250 0.9008 0.9337 0.9776 1.0
0.2470 0.0000 0.4705 0.6587 0.7528 0.7763 0.9174 0.9409 0.9761 1.0000
0.2033 0.5253 0.0000 0.6951 0.7798 0.8136 0.9322 0.9431 0.9829 1.0
0.1022 0.1931 0.2499 0.0000 0.4999 0.5737 0.9089 0.9429 0.9826 1.0
0.0729 0.1312 0.1676 0.4887 0.0000 0.5908 0.8973 0.9337 0.9774 1.0
0.0683 0.0958 0.1231 0.3011 0.4928 0.0000 0.8763 0.9173 0.9720 1.0
0.0754 0.1320 0.1650 0.4433 0.6414 0.7734 0.0000 0.8535 0.9667 1.0
0.0535 0.0892 0.1070 0.2141 0.3033 0.3568 0.6496 0.0000 0.9353 1.0
0.0579 0.1013 0.1302 0.2316 0.3185 0.3764 0.7242 0.9560 0.0000 1.0
0.0769 0.1538 0.1922 0.3075 0.4228 0.4997 0.7689 0.8842 1.0000 0.0
 32       0 R                                                    5
 33  36 2  Z     5  32  28  13
 37       0 R                                                    1
 38  39 1  Z    28  13
 40       0 R                                                    1
 41  42 1  Z    28  13
```

***** N E T W O R K F L O W S I M U L A T I O N *****

***** END TIME = 0.060 *****

***** STATISTICAL QUANTITIES OF INTERMEDIATE PROCESSOR *****

IMP.NO	BUFFER	ENTER-CIRCUIT	TOTAL-ENTRY	AVAILABILITY	AVERAGE-QLENGTH	MAXIMUM-QLENGTH
1	16	27 28 0 0	10	0.44444	0.028	1
2	24	13 24 25 0	10	0.86667	0.036	3
3	16	14 26 0 0	6	0.25667	0.017	1
4	24	11 12 23 0	11	1.96056	0.082	3
5	24	9 21 22 0	20	2.42027	0.101	3
6	32	8 10 19 20	9	1.23430	0.039	2
7	24	6 7 18 0	18	2.30325	0.096	3
8	24	5 16 17 0	4	0.26667	0.011	2
9	24	3 4 15 0	8	0.60010	0.025	2
10	16	1 2 0 0	1	0.04444	0.003	1

***** NETWORK CONECTING CIRCUIT STATISTICS *****

CIRCUIT.NO	BELONG.IMP	SPEED(BAUD)	ENTRY-NO	AVRG.QL	MAX.QL	MAX.WAIT	LOST-CALL	UTILITY	SURSIZE
1	9	1200	0	0.0	0	0.0	0	0.0	8
2	8	1200	1	0.003	1	0.0	0	0.00278	8
3	8	1200	2	0.003	1	0.0	0	0.00278	8
4	7	1200	2	0.008	2	0.001	0	0.00555	8
5	6	1200	2	0.016	1	0.0	0	0.00555	8
6	6	1200	3	0.008	1	0.0	0	0.00833	8
7	5	1200	7	0.019	1	0.0	0	0.01944	8
8	5	1200	4	0.011	1	0.0	0	0.01111	8
9	4	1200	9	0.031	2	0.001	0	0.02500	8
10	2	1200	3	0.008	1	0.0	0	0.00833	8
11	2	1200	0	0.0	0	0.0	0	0.0	8
12	3	1200	6	0.017	1	0.0	0	0.01667	8
13	1	1200	3	0.008	1	0.0	0	0.00833	8
14	1	1200	7	0.019	1	0.0	0	0.31944	8
15	10	1200	1	0.003	1	0.0	0	0.00278	8
16	10	1200	0	0.0	0	0.0	0	0.0	8
17	9	1200	2	0.036	1	0.0	0	0.00556	8
18	9	1200	6	0.017	1	0.0	0	0.01667	8
19	8	1200	1	0.003	1	0.0	0	0.00278	8
20	7	1200	4	0.011	1	0.0	0	0.01111	8
21	7	1200	12	0.036	2	0.0	0	0.03333	8
22	6	1200	0	0.0	0	0.0	0	0.0	8
23	5	1200	9	0.028	2	0.0	0	0.02500	8
24	6	1200	4	0.011	1	0.0	0	0.01111	8
25	4	1200	1	0.003	1	0.0	0	0.00278	8
26	4	1200	1	0.003	1	0.0	0	0.00278	8
27	2	1200	7	0.025	2	0.001	0	0.01944	8
28	3	1200	0	0.0	0	0.0	0	0.0	8

+++++ ANOTHER CIRCUIT INFORMATION +++++

```
+++++                                    +++++
+  MESSAGE LENGTH=1000(BIT)    CONSTANT  +
+  TRANSFER SPEED IS VARIABLE  (BAUD)    +
+  IMP BUFFER-SIZE IS VARIABLE(/PACKET)  +
+  MSG CALL IS POISSON DISTRIBUTION      +
+          MEAN IS 1.0(NUMBER/SEC)       +
+++++                                    +++++
```

+++++ NETWORK PATH SELECTION TABLE +++++

(CIRCUIT NO.,PASSING IMP NO.)

COLUMN / ROW	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	(0, 0)	(2,13)	(3,14)	(3,14)	(3,14)	(3,14)	(2,13)	(2,13)	(2,13)	(2,13)	(2,13)	(2,13)	(2,13)	(2,13)	(2,13)	(2,13)	(2,13)	(2,13)	(2,13)	(2,13)
2	(1,27)	(0, 0)	(1,27)	(4,11)	(4,11)	(4,11)	(6,10)	(6,10)	(6,10)	(6,10)	(6,10)	(6,10)	(6,10)	(6,10)	(6,10)	(6,10)	(6,10)	(6,10)	(6,10)	(6,10)
3	(1,28)	(1,28)	(0, 0)	(4,12)	(4,12)	(4,12)	(4,12)	(4,12)	(4,12)	(4,12)	(4,12)	(4,12)	(4,12)	(4,12)	(4,12)	(4,12)	(4,12)	(4,12)	(4,12)	(4,12)
4	(3,26)	(2,25)	(3,26)	(0, 0)	(5, 9)	(5, 9)	(5, 9)	(5, 9)	(5, 9)	(5, 9)	(5, 9)	(5, 9)	(5, 9)	(5, 9)	(5, 9)	(5, 9)	(5, 9)	(5, 9)	(5, 9)	(5, 9)
5	(4,23)	(4,23)	(4,23)	(4,23)	(0, 0)	(6, 8)	(7, 7)	(6, 8)	(7, 7)	(7, 7)	(7, 7)	(7, 7)	(7, 7)	(7, 7)	(7, 7)	(7, 7)	(7, 7)	(7, 7)	(7, 7)	(7, 7)
6	(2,24)	(2,24)	(2,24)	(5,22)	(5,22)	(0, 0)	(7, 6)	(8, 5)	(8, 5)	(8, 5)	(8, 5)	(8, 5)	(8, 5)	(8, 5)	(8, 5)	(8, 5)	(8, 5)	(8, 5)	(8, 5)	(8, 5)
7	(6,20)	(6,20)	(5,21)	(5,21)	(5,21)	(6,20)	(0, 0)	(9, 4)	(9, 4)	(9, 4)	(9, 4)	(9, 4)	(9, 4)	(9, 4)	(9, 4)	(9, 4)	(9, 4)	(9, 4)	(9, 4)	(9, 4)
8	(6,19)	(6,19)	(6,19)	(6,19)	(6,19)	(6,19)	(9, 3)	(0, 0)	(9, 3)	(10, 2)	(10, 2)	(10, 2)	(10, 2)	(10, 2)	(10, 2)	(10, 2)	(10, 2)	(10, 2)	(10, 2)	(10, 2)
9	(7,18)	(8,17)	(7,18)	(7,18)	(7,18)	(8,17)	(7,18)	(8,17)	(0, 0)	(10, 1)	(10, 1)	(10, 1)	(10, 1)	(10, 1)	(10, 1)	(10, 1)	(10, 1)	(10, 1)	(10, 1)	(10, 1)
10	(8,16)	(8,16)	(8,16)	(8,16)	(8,16)	(8,16)	(8,16)	(9,15)	(8,16)	(9,15)	(9,15)	(9,15)	(9,15)	(9,15)	(9,15)	(9,15)	(9,15)	(9,15)	(9,15)	(9,15)

+++++ NETWORK SELECTION PROBABILITY TABLE +++++

COLUMN / ROW	1	2	3	4	5	6	7	8	9	10
1	0.0	0.2307	0.3625	0.5603	0.6701	0.7250	0.9008	0.9337	0.9776	1.0
2	0.2470	0.0	0.4705	0.6587	0.7528	0.7763	0.9174	0.9409	0.9761	1.0
3	0.2033	0.5253	0.0	0.6951	0.7798	0.8136	0.9322	0.9491	0.9829	1.0
4	0.1022	0.1931	0.2499	0.0	0.4999	0.5737	0.9089	0.9429	0.9826	1.0
5	0.0729	0.1312	0.1676	0.4887	0.0	0.5908	0.8973	0.9337	0.9774	1.0
6	0.0683	0.0958	0.1231	0.3011	0.4928	0.0	0.8763	0.9173	0.9728	1.0
7	0.0754	0.1320	0.1650	0.4433	0.6414	0.7734	0.0	0.8635	0.9667	1.0
8	0.0535	0.0892	0.1070	0.2141	0.3033	0.3558	0.6496	0.0	0.9353	1.0
9	0.0579	0.1013	0.1302	0.2316	0.3185	0.3764	0.7242	0.9960	0.0	1.0
10	0.0769	0.1538	0.1922	0.3075	0.4228	0.4997	0.7689	0.8842	1.0	0.0

付表4.2　SIMSCRIPT－II

```
**                                                                      1
**    *=*=*     N E T W O R K     F L O W     S I M U L A T I O N     *=*=*  2
**                                                                      3
**                  /*  DESIGN & CODED BY T.OZAWA  */                   4
**                                                                      5
**    /*  NET--WORK  STRUCTURE  DESCRIPTION  */                         6
**                                                                      7
      PREAMBLE                                                          8
        NORMALLY, MODE IS INTEGER                                       9
**                                                                      10
**    /*  INTERMEDIATE  PROCESSOR  ABBREVIATION  /* IMP */ */           11
**                                                                      12
      PERMANENT  ENTITIES                                               13
          EVERY  IMP HAS A  BUF.SIZE     **/* IMP COMMON BUFFER SIZE     */**,  14
                         A  CIRC.ENT1    **/* ENTERING CIRCUIT NO.       */**,  15
                         A  CIRC.ENT2    ** /*ENTERING CIRCUIT NO..      */**,  16
                         A  CIRC.ENT3    ** /*ENTERING CIRCUIT NO..      */**,  17
                         A  CIRC.ENT4    ** /*ENTERING CIRCUIT NO..      */**,  18
                         A TOTAL         **/* IMP TOTAL ENTRY NUMBER     */**,  19
                         A  SAMPLE.IMP   **/* SAMPLING COUNTER           */**,  20
                         A  CURR.ENTRY   **/* CURRENT BUF.ENTRY NUMBER   */**,  21
                         A  BUF.AVAIL    **/* BUFFER   AVAILABILITY      */**,  22
                         A  MAX.BQL      **/* MAXIMUM BUF.QUEUE LENGTH   */**,  23
                        AN AVRG.BQL      **/* AVERAGE BUF.QUEUE LENGTH   */**   24
                    DEFINE  BUF.AVAIL    AS  REAL  VARIABLE                    25
**                                                                      26
**    /*  NETWORK  ORGANIZING  CIRCUIT  DESCRIPTION  */                 27
          EVERY CRCT HAS A  BELONG.IMP   **/*  BELONGING  IMP  NO        */**,  28
                         A  TRANS.TIME   **/*  CIRCUIT TRANSFER TIME     */**,  29
                         A  QUEIN        **/*  QUEUE ENTER TIME          */**,  30
                        AN AVRG.CQL      **/* AVERAGE CIRCUIT QUE LENGTH */**,  31
                         A  MAX.CQL      **/* MAXIMUM CIRCUIT QUE LENGTH */**,  32
                         A  MAX.WAIT     **/* MAXIMUM  WAITING  TIME     */**,  33
                         A NO.OF.ENTRY   **/* CIRCUIT TOTAL ENTRY        */**,  34
                         A SAMPLE.CRCT   **/* SAMPLING COUNTER IN CRCT   */**,  35
                         A  LOST.CALL    **/*  LOSTING  MESSAGE  CALL     */**   36
                AND OWNS A  CRCT.QUE     **/*  CIRCUIT OWN  QUEUE         */    37
                    DEFINE  TRANS.TIME, MAX.WAIT AS REAL VARIABLES              38
**                                                                      39
**    /*  OCCURRENCE  MESSAGE  DESCRIP.TION  */                         40
**                                                                      41
      TEMPORARY  ENTITIES                                               42
          EVERY  MSG HAS A     SOURCE     **/* OCCURRENCE TERMINAL IMP NO. */**,  43
                         A     QUEINTM    **/* CRCT QUEUE IN TIME */ **,          44
                         A     CURRENT    **/* CURRENT STAYING IMP NO.    */**,  45
                         A     DESTIN     **/* DESTINATION HOST IMP NO.   */**,  46
                         A     GEN.TIME   **/* GENERATION  TIME           */**,  47
                         AN    ELAPS.T    **/* ELAPSED  TIME              */**,  48
                         AN    OCCUPANCY  **/* OCCUPYING CIRCUIT NO.      */**   49
                    AND MAY BELONGS TO A CRCT.QUE                               50
                    DEFINE GEN.TIME,ELAPS.T  AS REAL VARIABLES                  51
**                                                                      52
      EVENT NOTICES INCLUDE MSGCALL          AND ENDSIM                 53
                    EVERY ENDSERV HAS A C                                54
**                                                                      55
```

```
        NORMALLY, DIMENSION = 2                                         56
         THE  SYSTEM  HAS  A  NETWORK(*/2)                              57
        NORMALLY, MODE IS REAL AND DIMENSION = 2                        58
         THE  SYSTEM  HAS  A  PROB                                      59
        NORMALLY, MODE IS INTEGER AND DIMENSION = 0                     60
**                                                                      61
      DEFINE  TRANS  AS A REAL  FUNCTION  GIVEN 1 ARGUMENT              62
END                                                                     63
```

```
**                                                                      1
**    /*  INITIALIZATION  &  START  SIMULATION  */                      2
**                                                                      3
      MAIN                                                            4
          'INITIALIZE'          PERFORM  INITIALIZATION               5
**  *******    LET  BETWEEN.V='TRACE'          *******                 6
          START  SIMULATION                                           7
**                                                                      8
**    /*  PERFORM  NEXT  SIMULATION  EXPERIMENTS  */                    9
**                                                                     10
**                                                                     11
**              /* CHECK POINT FOR DEBUGGING */                       12
**                                                                     13
                LIST       TIME.V,EVENT.V                           14
          START NEW PAGE                                             15
      WRITE  HOUR.F(TIME.V) AS B 30,                                 16
       "**=**= N E T W O R K    F L O W    S I M U L A T I O N *=**=",/,/,/,  17
         B 70," START TIME=",I 2,"HOUR"                              18
                START      NEW PAGE                                  19
      WRITE HOUR.F(TIME.V) AS B 70,"END TIME=",I 2,"HOUR",/,/,/,     20
      B 30,"*=* STATISTICAL QUANTITIES OF INTERMEDIATE PROCESSOR *=*",/,  21
      B 10,"IMP.NO",S 4,"BUFFER",S 4,"ENTER-CIRCUIT",S 2,"TOTAL-ENTRY",S 4,  22
      "AVAILABILITY",S 3,"AVG.QLENGTH",S 4,"MAX.QLENGTH",/,/         23
                FOR I = 1 TO 10,DO                                   24
                WRITE I ,BUF.SIZW,CIRC.ENT1,CIRC.ENT2,CIRC.ENT3,CIRC.ENT4,  25
                     IMPTOTAL,BUF.AVAIL,AVRG.BQL, AND MAX.BQL AS     26
           B 15,I 2,S 8,I 3,S 4,4 I 3,S 10,I 5,S 11,D(5,4),S 9,D(5,4),S 10,I 5,/  27
                LOOP                                                 28
                START      NEW PAGE                                  29
                PRINT      1 LINE AS FOLLOWS                         30
                ===  NETWORK CONECTING CIRCUIT STATISTICS ===
                WRITE AS B 10,                                       31
                "CIRCUIT",S 3,"BELONG.IMP",S 3,"SPEED(BAUD)",S 3,"NO.OF.ENTRY",  32
                S 3,"AVRG.QL",S 3,"MAX.QL",S 3,"MAX.WAIT",S 3,"LOST.CALL",/,/  33
                FOR        I=1 TO 28,DO                              34
                WRITE I,BELONG.IMP,TRANS.TIME,NO.OF.ENTRY,AVRG.CQL,MAX.CQL,  35
                     MAX.WAIT AND LOST.CALL AS B 15,                 36
                I 3,S 11,I 3,S 8,I 4,S 11,I 4,S 6,D(5,4),S 5,I 4,S 3,D(8,2),  37
                       S 8,I 4,/                                     38
                LOOP                                                 39
                START      NEW  PAGE                                 40
                PRINT      13  LINES AS FOLLOWS                      41

       ==============  ANOTHER  CIRCUIT  INFORMATION  ==========================
                =====                              =====
                =  MESSAGE LENGTH=1000(BIT)   CONSTANT =
                =  TRANSFER SPEED IS VARIABLE (BAUD)   =
                =  IMP BUFFER SIZE IS VARIABLE(/PACKET)=
                =  MSG CALL IS POISSON DISTRIBUTION    =
                =         MEAN IS 1.0(NUMBER/SEC)      =
                =====                              =====

                =====  NETWORK  PATH  SELECTION  TABLE  =====
                          (CIRCUIT NO.,PASSING IMP NO.)



          FOR I=1 TO 10, DO                                          42
      PRINT 1 LINE WITH                                              43
           NETWORK(I, 1),NETWORK(I, 2),NETWORK(I, 3),NETWORK(I, 4),  44
           NETWORK(I, 5),NETWORK(I, 6),NETWORK(I, 7),NETWORK(I, 8),  45
           NETWORK(I, 9),NETWORK(I,10),NETWORK(I,11),NETWORK(I,12),  46
           NETWORK(I,13),NETWORK(I,14),NETWORK(I,15),NETWORK(I,16),  47
           NETWORK(I,17),NETWORK(I,18),NETWORK(I,19),NETWORK(I,20)   48
      AS FOLLOWS                                                     49
(**,**) (**,**) (**,**) (**,**) (**,**) (**,**) (**,**) (**,**) (**,**) (**,**)
          LOOP                                                       50
                PRINT      3  LINES  AS  FOLLOWS                     51

                =====  NETWORK  SELECTION  PROBABILITY  TABLE  =====

                FOR        I=1 TO 10,DO                              52
                           FOR J = 1 TO 10,                          53
                                WRITE PROB(I,J) AS S  2,"(",D(5,4),")"  54
                LOOP                                                 55
STOP          END                                                    56
```

```
ROUTINE  FOR  INITIALIZATION                                                    1
**                                                                              2
**     /*  IMP  INFORMATION  SETTING-UP  */                                     3
**                                                                              4
          LET  N.IMP=10        CREATE  EVERY  IMP                               5
               FOR  EACH  IMP , DO                                             6
                    READ  BUF.SIZE,CIRC.ENT1,CIRC.ENT2,CIRC.ENT3,CIRC.ENT4     7
               LOOP                                                            8
**                                                                             9
**     /*  CIRCUIT INFORMATION SETTING-UP */                                   10
**                                                                            11
          LET  N.CRCT=28       CREATE  EVERY  CRCT                            12
               FOR  EACH  CRCT, DO                                           13
                    READ  BELONG,IMP,TRANS.TIME                              14
               LOOP                                                          15
**                                                                           16
**      /* NETWORK  PATH  SELECTION  TABLE */                                 17
**                                                                           18
          RESERVE  NETWORK(*,*) AS 10 BY 20                                  19
           FOR  I=1 TO 10,FOR J=1 TO 20,DO                                   20
                READ  NETWORK(I,J)                                           21
           LOOP                                                              22
**                                                                           23
**      /* NETWORK SELECTION PROBABILITY TABLE */                            24
**                                                                           25
          RESERVE  PROB(*,*)  AS  10 BY 10                                   26
          FOR  I=1 TO 10,FOR  J=1 TO 10,DO                                   27
               READ  PROB(I,J)                                               28
          LOOP                                                               29
**                                                                           30
**     /*  MESSAGE  CALL  OCCURRENCE  */                                     31
**                                                                           32
          SCHEDULE  A  MSGCALL  AT  0                                        33
          SCHEDULE A ENDSIM AT TIME.V + 360.0                                34
RETURN         END                                                           35
```

```
EVENT  FOR   MSGCALL                                                            1
**                                                                              2
**       /* CHECK POINT */                                                      3
**                                                                              4
              LIST  TIME.V,EVENT.V                                              5
          CREATE  A  MSG                  **/* MESSAGE OCCURRENCE */          6
          CALL     MESSAGE     YIELDING  ORIGIN , TERMINUS AND ARRIVAL          7
                     LET  SOURCE(MSG)=ORIGIN                                    8
                     LET  DESTIN(MSG)=TERMINUS                                  9
                     LET  GEN.TIME(MSG)=TIME.V                                 10
      CALL  CRCT.SELECT GIVEN ORIGIN AND TERMINUS YIELDING CRCT.NO AND IMP.NO   11
                     LET  OCCUPANCY(MSG)=CRCT.NO                               12
                     LET  CURRENT(MSG)=IMP.NO                                  13
**                                                                             14
**     /*  ENTER  IMP(ORIGIN)  &  NEXT  MSGCALL  PROVISION  */                 15
**                                                                             16
          SCHEDULE  A  MSGCALL  AT  TIME.V+ARRIVAL                           17
          CALL     ENTERIMP     GIVEN  ORIGIN                                 18
RETURN    END                                                                  19
```

```
ROUTINE  FOR  SRVSTART  GIVEN NO.IMP                                    1
         LET    II=OCCUPANCY(MSG)                                       2
         ADD 1 TO CURR.ENTRY(NO.IMP)                                    3
         LET  QUEINTM(MSG)=TIME.V                                       4
         FILE  MSG  LAST  IN  CRCT.QUE(II)                              5
         ADD 1 TO SAMPLE.IMP(NO.IMP)    **/* SAMPLING NUMBER */        6
         ADD 1 TO SAMPLE.CRCT(II)       **/* SAMPLING NUMBER */        7
         IF    N.CRCT.QUE(II) < 2                    GO TO NEXT        8
               ELSE          LET  QUEIN(II)=TIME.V    RETURN           9
'NEXT'   IF    BUF.SIZE(CURRENT(MSG)) > CURR.ENTRY(CURRENT(MSG))       10
              GO  TO  CRCTSERV                                         11
               ELSE  RETURN                                            12
'CRCTSERV'     ADD  1  TO  CURR.ENTRY(CURRENT(MSG))                    13
         SCHEDULE  AN  ENDSERV      AT  TIME.V + TRANS(II)              14
         LET C = II                                                     15
RETURN   END                                                            16
```

```
ROUTINE   FOR   ENTERIMP   GIVEN   IMPNO                               1
''                                                                      2
''    /*  COMMON BUFFER ENTER  & CIRCUIT SERVICE START  */              3
          ADD 1 TO TOTAL(IMPNO)     **/* TOTAL ENTRY IN BUFFER */      4
          LET  II=OCCUPANCY(MSG)                                        5
          ADD 1 TO NO.OF.ENTRY(II)      **/* TOTAL ENTRY IN CIRCUIT */ 6
          IF    BUF.SIZE(IMPNO) > CURR.ENTRY(IMPNO)    GO TO SERVICE    7
                ELSE           ADD 1 TO LOST.CALL(II)                   8
                               DESTROY  A  MSG          GO TO TERM      9
'SERVICE'       CALL  SRVSTART  GIVEN  IMPNO                            10
'TERM'          RETURN          END                                     11
```

```
EVENT  FOR  ENDSERV(C)                                                    1
**                                                                        2
**        /* CHECK POINT */                                               3
**                                                                        4
               LIST  TIME.V,EVENT.V                                       5
**                                                                        6
**     /*  CIRCUIT  SERVICE  END  &  NEXT  SERVICE  PROVISION  */         7
**                                                                        8
          REMOVE  THE  FIRST  MSG  FROM  CRCT.QUE(C)                      9
               LET  K = BELONG.IMP(C)    **/* IMP ORIGIN */              10
          ADD  CURR.ENTRY(K) TO AVRG.BQL(K)       **/* AVERAGE QUE LENGTH */ 11
          LET  BUF.AVAIL(K) = BUF.AVAIL(K) + CURR.ENTRY(K) / BUF.SIZE(K)  12
          IF  MAX.BQL(K) < CURR.ENTRY(K)                                  13
               LET  MAX.BQL(K) = CURR.ENTRY(K)  ELSE                      14
          SUBTRACT  1  FROM  CURR.ENTRY(BELONG.IMP(C))                    15
               LET  N=CURRENT(MSG)                                        16
               LET  L=OCCUPANCY(MSG)                                      17
          ADD  N.CRCT.QUE(L) TO AVRG.CQL(L)       **/* AVERAGE QUE LENGTH */ 18
          IF  MAX.CQL(L) < N.CRCT.QUE(L)                                  19
               LET  MAX.CQL(L) = N.CRCT.QUE(L)  ELSE                      20
**                                                                        21
**     /*  NEXT  SERVICE  PROVISION  */                                   22
**                                                                        23
          IF  CURRENT(MSG)=DESTIN(MSG)  **/* DESTINATION IMP */          24
               GO TO TERM                                                 25
     ELSE      LET  M = DESTIN(MSG)          ** /* MSG DESTINATION */     26
               CALL CRCT.SELECT(N,M)  YIELDING  CRT.NO AND IMP.NO         27
                    LET  OCCUPANCY(MSG)=CRT.NO                            28
                    LET  CURRENT(MSG)=IMP.NO                              29
          CALL  SRVSTART  GIVEN  N           GO TO EXIT                   30
*TERM*    DESTROY  THE  MSG                                               31
*EXIT*    CALL  CHECK  GIVEN  N AND C                                     32
RETURN    END                                                             33
```

```
ROUTINE  FOR  CRCT.SELECT(ORG,DST,I,J)                                    1
          LET  I=NETWORK(ORG,DST*2)          **/*  IMP NO. */           2
          LET  J=NETWORK(ORG,DST*2-1)        **/* CIRCUIT NO. */         3
RETURN    END                                                             4
```

```
EVENT  FOR  ENDSIM                                                          1
          FOR I=1 TO EVENTS.V,FOR EACH NOTICE IN EV.S(I),DO                 2
          REMOVE  THE  NOTICE FROM  EV.S(I)                                 3
          LOOP                                                              4
**                                                                          5
**      /*  TALLY  THE  STATISTICS  */                                      6
**                                                                          7
          FOR  EACH  IMP , DO                                               8
               LET  BUF.AVAIL = BUF.AVAIL / SAMPLE.IMP                      9
               LET  AVRG.BQL = AVRG.BQL / SAMPLE.IMP                        10
          LOOP                                                              11
          FOR  ALL  CRCT , DO                                               12
               LET  AVRG.CQL = AVRG.CQL / SAMPLE.CRCT                       13
          LOOP                                                              14
RETURN    END                                                               15


ROUTINE  FOR  CHECK  GIVEN  CRTIMP  AND  BLGCRT                             1
      DEFINE  X  AS  REAL  VARIABLE                                         2
          LET  SW=0                                                         3
          LET  I=CIRC.ENT1(CRTIMP)            IF  CRCT.STATE(I)=1           4
               LET  SW=1  GO TO NXTSRV                                      5
   ELSE   LET  I=CIRC.ENT2(CRTIMP)            IF  CRCT.STATE(I)=1           6
               LET  SW=1  GO TO NXTSRV                                      7
  ELSE    LET  I=CIRC.ENT3(CRTIMP)            IF  I=0  GO TO NEXT           8
               ELSE IF CRCT.STATE(I)=1  LET SW=1  GO TO NXTSRV              9
   ELSE   LET  I=CIRC.ENT4(CRTIMP)            IF  I=0  GO TO NEXT           10
               ELSE IF CRCT.STATE(I)=1  LET SW=1  GO TO NXTSRV  ELSE        11
'NXTSRV'  ADD 1 TO CURR.ENTRY(CRTIMP)                                       12
          LET  X = TIME.V - QUEINTM(MSG)          '*/* WAITING TIME */      13
          IF MAX.WAIT(BLGCRT) < X                                           14
               LET MAX.WAIT(BLGCRT) = X  ELSE                               15
          SCHEDULE  AN  ENDSERV       AT  TIME.V + TRANS(I)                 16
          LET C = I                                                         17
'NEXT'    IF  CRCT.QUE(BLGCRT)  IS  EMPTY                    RETURN         18
          ELSE  ADD 1  TO  CURR.ENTRY(CRTIMP)                               19
          SCHEDULE  AN ENDSERV          AT TIME.V + TRANS(BLGCRT)           20
          LET C = BLGCRT                                                    21
RETURN    END                                                               22

 ROUTINE  FOR  MESSAGE(ORG,TRM,ARV)                                         1
       DEFINE  X  AS  A  REAL  VARIABLE                                     2
**                                                                          3
**      /*  ORIGIN  IMP  DETERMINATION  */                                  4
**                                                                          5
           LET  X=UNIFORM.F(0.0,1.0,1)                                      6
                IF  X < .0924, LET  ORG=1      GO TO NEXT                   7
           ELSE IF  X < .1787, LET  ORG=2      GO TO NEXT                   8
           ELSE IF  X < .2386, LET  ORG=3      GO TO NEXT                   9
           ELSE IF  X < .4174, LET  ORG=4      GO TO NEXT                   10
           ELSE IF  X < .5566, LET  ORG=5      GO TO NEXT                   11
           ELSE IF  X < .6307, LET  ORG=6      GO TO NEXT                   12
           ELSE IF  X < .8461, LET  ORG=7      GO TO NEXT                   13
           ELSE IF  X < .9030, LET  ORG=8      GO TO NEXT                   14
           ELSE IF  X < .9731, LET  ORG=9      GO TO NEXT                   15
           ELSE                LET  ORG=10     GO TO NEXT                   16
 'NEXT'    FOR  I=1 TO 16,DO                                                17
                LET Y=PROB(ORG,I)                                           18
                IF  X < Y ,    LET  TRM=I      GO TO EXIT                   19
                ELSE      LOOP                                              20
 'EXIT'    LET  ARV=POISSON.F(1.0,2)                                        21
 RETURN    END                                                              22


ROUTINE  FOR  TRANS  GIVEN  NO.CRCT                                         1
          RETURN  WITH  1000.0/TRANS.TIME(NO.CRCT)                          2
END                                                                         3
```

参　考　文　献

1) SIMULA Reference Manual (60234800)
Control Data 6400/6500/6600 Computer Systems. (1971)

2) CDC 6600 SIMSCRIPT
(May 1971)

3) The Simscript II Programming Language
P. J. Kiviat et al. Prentice-Hall, Inc. (1968)

4) UNIVAC 1106/1108 SIMULA (1970)

5) General Purpose Simulation System V User's Manual
(IBM August 1961)

6) Incremental Simulation on a Time-Shared Computer
by MaLcolm Murray Jones (MAC-TR-48) January 1968.

7) SIMPL Reference Manual
Malcolm M. Jones et al. (October 1972)

8) SIMPL Primer
Malcolm M. Jones et al. (September 1972) Third Revision

9) オンラインシミュレーション言語 SIMBOL (46-S004)
日本情報処理開発センター (1971)

10) SIMULA A Language for programming and description of discrete event systems
Introduction and User's manual
by Ole-Johan Dahl & Kristen Nygaard
(Norwegian Computing Center) (September 1967)

11) On-line Computation and Simulation OPS-3
Martin Greenberger et al. (MIT May 1967)

12) Essentials of Simulation
J. H. Mize/J. G. Cox 1968 Prentice-Hall, Inc.

13) The Art of Simulation
(シミュレーションの理論) K. D. Tocher 著　上條史彦訳
日本生産性本部 (January 1971)

14) Computer Simulation – Discussion of the Technique and Comparison of Languages
(Daniel Teichroew & John Francis Lubin)
Vol.9 No. 10, October 1966 Communications of the ACM

15) The Past, Present and Future of General Simulation Languages
Reino A. Merikallio
(Proceedings of the IFIF Congress 1965)

昭和48年 3月発行

発行所 財団法人 日本情報処理開発センター

東京都港区芝公園3－5－8

機械振興会館内

TEL (434) 8211 (代表)

印刷所 東京都渋谷区代々木2丁目26番地

有限会社 秀 嶺 社

TEL (379) 1816

47—S004