情報処理月例会資料 33

KEIŌ－TŌSBAC
タイムシェアリング システム

土　居　範　久

1968年2月20日

社団法人 情 報 処 理 学 会

KEIŌ－TŌSBAC
タイムシエアリング　システム

慶応義塾大学　　土　居　範　久

0　はじめに

慶応義塾大学では，計算機に基づく協同体系の開発を目的として，1965年ごろからタイムシエアリング・システムの研究にとりかかり，TŌSBAC 3400－30 を中心としたタイムシエアリング・システムを開発し，1967年6月から実験的に稼動しはじめた．

まず，会話型FŌRTRAN システム（通称KEIŌ システム）およびdesk calculator を開発．ファイル　システムは簡単なもので磁気テープを使用している．端局には，通常の電動タイプライタを改造したものを使用し，その台数は2台である．端局制御装置（SIŌC）も本格的な制御装置の開発を目的としたパイロット　モデルである．

1967年6月16日　TŌSBAC 研究会で，また，1967年9月20日一般に披露したものが，このVersion 0で，システムの機器構成はFig.1 の通り．

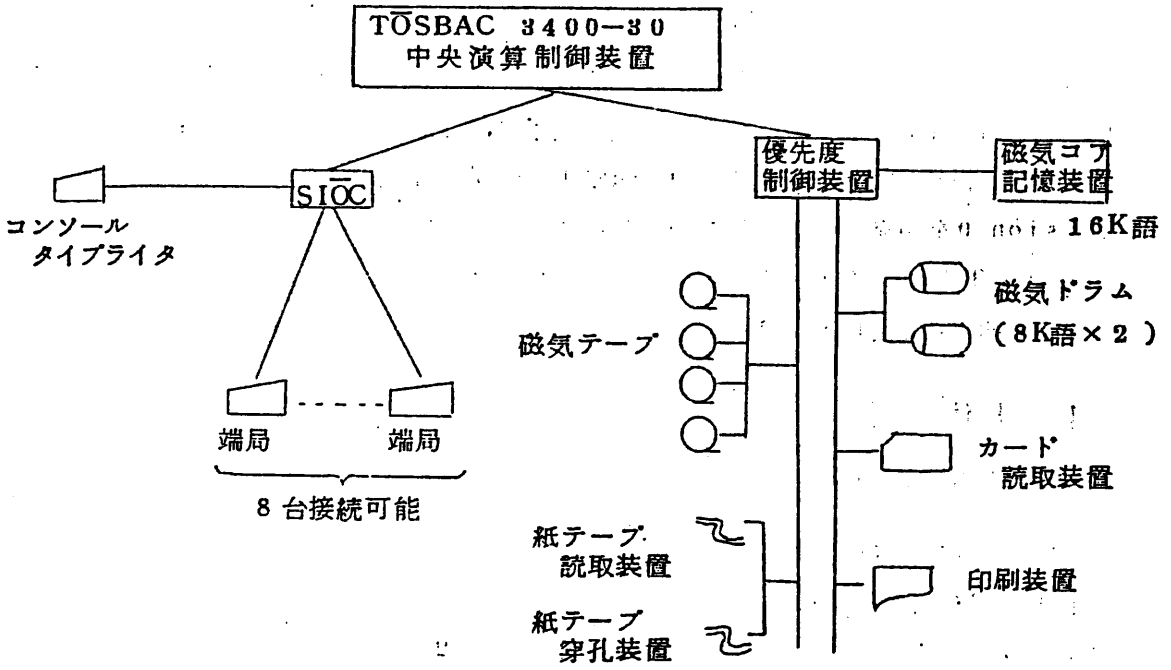


Fig.1 KEIŌ－TŌSBAC タイムシエアリング　システム
Version 0 機器構成

現在，Version 0には前記の2システムの他に，LISP，SNOBOL，KFMS（Keio Formula Manipulating System）が開発されて居り，さらに，カーブ・プロッタを操作するための言語Autoplot を開発中である．これらのシステムは，すべて純手続である．

以上が現在のシステムVersion 0の概要であるが，周辺機器の使用法を次に簡単に解説しておこう．

何分コア記憶装置が16K語しかなく，さらにスーパーヴァイザ，コマンド処理ルーテイン，KEIO 等のサブシステムが入ると，使用者の領域は4K～6K語ということになる．そこで，現在コア内にある使用者のプログラムは，別の使用者の処理を行うときには必然的にコアから追い出さなければならなくなる．このswapの作業には磁気ドラムを使用する．

KEIO，desk calculator等のサブシステムは一本の磁気テープに格納しておき，必要となる度にコアに読み込まれる．

Vorsion 0では使用者のプログラムだけはファイルとして保存できるようになつて居り，このために磁気テープを1本使用し，ファイル用の辞書は磁気ドラムに保持している．

その他の周辺機器は一斉使用しないし，端局からも使用できない．

現在，このVersion 0での実験に基づき，パイロット・モデルに種々の改良を加えた本格的な端局および端局制御装置の開発が完了し，さらに周辺機器も交換，追加した機器構成に移行の中途にあり，この結果システムの中心となるスーパーヴァイザも徹底的に改良している段階にあるが，このVersion 1の詳細は別の機会にゆずることにして，Version 0を開発する際に我々が歩んで来た過程およびサブシステムの代表例として会話型FORTRAN をとりあげ，これを開発する際に考慮しなければならなかつた問題点を幾つか述べることにしよう．

1 本体の改造

本体の改造は必要最小限に止めた．

1.1 インターバル タイマ

TOSBAC 3400-30 のオプションとして，20ms単位に時を刻むインターバルタイマがあるので，これを採用．

1.2 モードとPrivileged Operation

計算機を使用者が勝手に停止させたり，特別なレジスタを操回したり，あるいは，周辺機器の管理はスーパーヴァイザが一括管理した方が何かと都合がよいことなどから，ある

種の命令はPrivileged Operationと定め，使用者のプログラムでは使用できないようにした．その結果，Privileged Operationが実行できる状態とできない状態の2種のモードをもつよう改造した．この2種のモードを，それぞれマスタ モード，スレイブ モードと呼ぶ．

次の命令はPrivileged Operationであり，マスタ モードにある時だけ実行可能である．

SEIP	(Select Input)	コンソール関係入出力命令
SEOP	(Select Output)	
RCS	(Read Console)	
WCS	(Write Console)	
SER	(Select Read)	チャネル関係入出力命令
SEW	(Select Write)	
RLCN	(Reset and Load Channel)	
LCN	(Load Channel)	
SECN	(Select Channel)	
ETM	(Enter Trapping Mode)	割込関係命令
LTM	(Leave Trapping Mode)	
PM	(Protect Memory)	メモリ プロテクト
WTM	(Write Tape Mark)	磁気テープ関係命令
ERA	(Erase)	
BSR	(Back Space Record)	
BSF	(Back Space File)	
RWD	(Rewind)	
HJ	(Halt Jump)	停止命令
HP	(Halt and Proceed)	
SII	(Set Indicator from Interruption)	インディケータ

これらの命令がスレイブ モードで出された時は原則として，その命令はNOP(No Operation)となりGroup 1 の割込み(演算結果等にもとづく割込み等が属し，割込みの強さは2番目のもの)が起る．

各モードに入る条件は次に述べるメモリ プロテクトと関連する．

1.3 メモリ プロテクション

一般にシステム プログラムには"虫"はいないものと考えられるが，一般の使用者のプログラムにはどのような悪質な"虫"がいないとも限らない．この"虫"が"聖域"を

犯すようなことがあつては一大事であるから"聖域"は完全防備されていなければならない。

そこで，2つの境界レジスタを設け，プロテクトしなくてもよい領域の上限と下限を指定するようにした。この領域は256語単位で指定する。マスタ　モードにある時には，このメモリ　プロテクションは無効であるが，スレイブ　モードにある時には，コアを読み書きする度に境界レジスタの内容を読み書きする番地が比較される。その番地がプロテクトされるべき領域になければ，そのまま読み書きが行なわれるが，プロテクトされるべき領域にある場合には，その読み書きは行なわないでその命令を終り，その後でGroup 1の割込みを生ずる。

さらに，各モードに入る条件は次の通り。

- マスタ　モードに入る条件

  割込みが生じたとき，および，本体のコンソールにあるCONT　CLEARキーを押したときにはマスタ　モードとなる。

- スレイブ　モードに入る条件

  プロテクトされている領域から，プロテクトされていない領域へJumpした時にスレイブ　モードとなる。

通常，割込処理は割込飛先番地にJTL（Jump and Transfer：n番地に番地部にmをもつJTL命令があるとき，この余令を実行すると，m番地に"n＋1"を入れ"m＋1"番地に分岐する）命令を入れておき，その番地部に割込処理先を指定しておくことにより行なうが，この割込処理はマスタ，スレイブいずれのモードでも行なえるということになる。すなわち，割込みが起るとマスタ　モードに入り，その飛先番地の命令を実行することになるが，このとき，割込飛先番地に番地部にプロテクトされていない領域内にある場所を指定したJTL命令を入れておけば，その命令の実行後スレイブ　モードとなり，プロテクトされている領域内にある場所を指定しておくと当然マスタ　モードのままその場所に移る。ところで，割込飛先番地にJTL命令以外の命令が入つている場合には，その命令を実行後，再び割り込まれたプログラムにもどるが，このときにはマスタ　モードのままである。ただし，これがJump関係の命令があり，その番地部がプロテクトされていない領域内の場所を指している場合はスレイブ　モードに入る。

この結果，従来のPM命令は機能が変更されている。

なお，入出力装置との間でやりとりされるデータに対しては，メモリ　プロテクションは行なわれない。

14　以上の機能に対する追加修正（Version 1）

Version 1では前述の諸機能をさらに次のように修正を加えた（目下検討中のものも

含む）

・インターバル　タイマ

Overhead の算出その他システムを解析したり，使用者のaccounting を正確に算出しようとすると，20ms単位の時計では"100m 競争を砂時計で計時しているようなもの"で実に都合が悪い．そこで，20〜30μs 単位の時計を目下検討中である．さらにclonological clock があるとさらに便利であるので，これはVersion 0開発当初からその仕様を検討中である．

・メモリ　プロテクション

メモリ　プロテクションを行なう単位はVersion 0では256 語であるが，コマンド処理ルーテインや他のsupervisoryルーテインをドラムにもち，随時必要な度毎にコアに読み込む方法をとるようになると特に256語という単位は相対的に大きすぎるので128語単位に変更した．

・その他

ドラム等の周辺機器が完備してくると，かなり大きなプログラムも処理したくなるのが人情である．となるとそれに付随したoverhead を少しでも減少させなければならない．そこでrelocation register の必要性が生じたので，これも目下検討中である．

2　端局と端局制御装置

端局とのインターフエースとしては，プロセッサをもつようなものは使用しないという立場をとることにした．すなわち，すべての処理はTOSBAC 3400−30 で行なうということにしたわけである．この立場で，端局と本体の接続方式をいろいろな観点から検討した結果，端局は広い意味でのコンソール入出力機器の一つとし，各端局を一括管理するMultiplexor を使用することにした．たとえば，各端局をチヤネルを媒介とした場合には，入出力にはコマンドを利用することになり，端局の自由度が極端に落ちる等の欠点がある．

Multiplexorすなわち端局制御装置の仕様が，端局の仕様と並行して決まると，ソフトウエアシミユレータをつくり，これをもとにスーパーヴアイザを開発するという方法が，通常，よくとられるが，タイミング等の点において真のシミユレータの作成は非常にむずかしい．そこで我々は金物でパイロット　モデルをつくり，ソフトウエアおよびハードウエア両方の開発を進めることにした．

端局のパイロット　モデルとしては，通常の電動タイプライタを改造したものを採用し，端局制御装置のパイロット　モデルとしてはSIOC（Special Input/Output Controller）を開発した．

2.1 端局制御装置

端局とのデータ授受は，各端局の同時動作を許すために割込み機能を利用し，本体側では1字毎にＲＣＳ（Read Console）またはWCS.(Write Console) 命令を用いて処理をする．このとき，端局制御装置は割込みに対するＲＣＳ，WCS命令がどの端局に対して働くのかを自動的に決める．

ＳＩ$\overline{O}$Ｃ からの割込みはGroup 2 の割込み（入出力関係にもとづく割込みが属し，割込みの強さは3番目のもの）一種だけで，この割込みの結果，端局番号と割込みの種類を知ることができる．割込の種類としては次の3種のものがある．

- ＥＲＣ(End of Receiving of a Character)
  端局からのデータを受信完了
- ＥＴＣ(End of Transmission of a Character)
  端局へのデータを送信完了
- ＣＳ (Channel Signal)
  割込原因コードのセット完了

すなわち，ＥＲＣおよびＥＴＣの割込みを媒介として，ＲＣＳおよびWCＳ命令を用いて1字ずつデータの授受を行なうのである．

割込原因コードの主なものは次の通り．

- ＳＩ$\overline{O}$Ｃ の状態と出された命令とが矛盾している．
- データ受信動作中に次のデータがきた等，その時までに受信したデータは正しくない．
- 交信が不能になつた
- 端局からの受信が完了した．

端局へ送信したデータに関する情報，および端局の状態等は，端局への最後のデータを送信後，ＳＥＩＰ 命令（従来の命令に対し，機能を拡張）を出すことにより知ることができる．すなわち，この命令の結果として，端局からSTATUS コードを送つてくる．

STATUS コードの主なものは次の通り．

- 端局が受信したデータにエラーがあつた
- 端局がオフラインになつている．
- インタラプション　スイッチが＇$\overline{O}$Ｎ＇になつている．

端局制御装置を接続した結果，前述のＳＥＩＰ 命令の外にもＳＥ$\overline{O}$Ｐ 命令の機能も拡張されている．

2.2 端　局

端局には通常の電動タイプライタを改造したものを使用している．多行に渡る印刷を途中で中断させるためのインタラプション　スイッチ，演算実行中のプログラムの実行を中

断させるためのguit キー，打鍵可能であることを表示する緑色のランプ等が増設されている外，前述のSTATUS コードを生成するための機能をもつている．また，端局タイプライタの電源が"OFF"のときには，本体から電源を投入し不在通信も可能である．

2.3 以上の機能に対する追加修正（Version 1）

パイロット モデルを実際に使用した経験にもとづき，さらに可成りの改良を加えた本格的な端局（TOSBAC DN—510 およびDN—511）および端局制御装置（TOSBAC DN—230）を開発した結果前述の機能は大幅に改良された．

SIOCは半二重方式にもとづいていたのに対し，DN—230 は全二重，半二重いずれの方式をも採用できる様になつた他，データ伝送上のメッセージ フォーマットを標準型にしたことにより，多行に渡るデータの授受が容易になり，端局の機能がかなり増大した．この結果，割込原因コード，STATUS コードの種類を増加した他，STATUS コードもCSの割込を用いるように変更した．したがつてSEIPおよびSEOP 命令の機能もここに記述したものとは異なつている．その他，DN—230 が本体に割込みをかける際にはシステムのoverhead を減少させるように，SIOC の割込み方式を徹底的に改良した．

これらDN—230，DN—510 およびDN—511 に関する詳細は別の機会に行なうことにしてここでは行なわない．

3 スーパーヴアイザ

Version 0 でのスーパーヴアイザの機能を大別すると次の通り．

- 端局相手の入出力の管理
- スケジユーリング
- サブシステムの管理
- accounting
- 各種割込の管理

3.1 端局相手の入出力の管理

端局相手の入出力の管理は，さらに次の3種のものにわけることができる．

- ERC, ETC, CS の割込み処理
- 入力データの編集
- 割込原因コード，およびSTATUS コードの処理

ERC, ETC, CSの割込み処理

我々のシステムのように1字毎の割込によつて端局とデータの授受を行なうようなシステムでは，処理の仕方にもよるが，特に端局からの割込の処理速度ははやければはやい方

が良い。

端局コードと本体コードとが異るので通常のデータはコード変換の必要がある。特に入力時には，出力時よりも処理すべきデータ量が多いので，pool buffer を設け，割込処理時には，端局コードは変換せず生のまま，端局番号と共にここへ格納するだけの処理しかしていない。

出力時には，出力用のbufferにあるデータを伝送するだけのことであるから，コード変換はETC 割込処理の際に行なつている。

割込原因コードおよびSTATUS コードに対する処理はスケジューリングおよびその他に関連していることから，コード受信時に直ちに対応する処置をとるのはどうもうまくなく，これらの処置は遅ければ遅い程簡単に行なうことができる。もちろん，ある種のものは直ちに適切な処置をとらなければならないものもある。そこで，急を要するもの以外はロジカルなシグナルに変換し，通常のデータと一緒にpool bufferに格納し，処理を遅らしている。

入力データの編集

一定のtime sliceの後に，pool buffer内のデータを調べ，シグナル以外のものであればコード変換し，各使用者ごとに振り分けデータを組み立てる。このとき，1字抹消とか全文抹消という指示があれば，これに従つて対応する使用者のデータを修正する。

この時点でも，また，データ伝送時のパリテイ　エラー等のチエックも行なう。

割込原因コードとSTATUSコードの処理

pool buffer内のデータを編集している際にシグナルに遭遇したときには，シグナル別に適切な処理を行なう。通常，いくつかのロジカル　シグナルの組合せにより，とるべき処置が決まり，その結果，使用者の状態は別の状態に遷移することになる。pool buffer　を用いて一括管理する方法を用いると全体のロジックは一応簡単になるが，シグナル処理に関しては前述の処理時点等の点でまだ最適ではないのでVersion 1 ではさらに改良する積りである。

3.2　スケジューリング

入力データの完了および出力の完了に従つて，使用者の内部での状態が定まるので，その状態に従つて，次に処理すべき使用者を決定する。使用者決定の際にはVersion 0では単純なRound Robin 方式をとつている。

Version 1では，システム内部での使用者の状態はまつたくVersion 0とは異なり，従つて，使用者決定のアルゴリズムもVersion 0とは完全に異つており，変則的なRound Robin 方式を用いている。

3.3 サブシステムの管理

KEIOシステム(Keio Elementary Instructive Operating system：会話型FORTRAN), desk calculator, LISP, SNOBOL, KFMS (Keio Formula Manipulating System)はすべて純手続であるから，使用者の交替の際には，現在コア内にあるサブシステムが，次に必要とするシステムと異なるときにだけ，磁気テープから読み出せばよい．効率その他の点から純手続というものに批判があるが，我々は，コアおよび補助記憶装置の容量の点から，および，この種のサブシステムにおいて純手続を採用したときの効率の悪さは前述の理由等に比べるとまつたく問題にならないという観点からサブシステムはすべて純手続にした．もちろん，システム プログラムは効率および容量の点から純手続にはしていないし，Version 1においても純手続にする積りは毛頭ない．

3.4 accounting

使用者のプログラムを処理するのに要した時間，コマンド処理に要する時間，および使用者のファイルを読み込んだり，書き込むのに要する時間の和を使用者が使用した時間として計上しているが，Version 1では本格的なaccounting を行なう積りであるので，ファイル保持のための補助記憶装置の使用等もaccounting の対象とする積りである．

これまでに述べた端局制御装置(SIOC) およびスーパーヴァイザの詳細に関しては参考文献〔1〕を参照されたい．

サブシステムの開発の際には，各サブシステムが会話型である故に，いろいろと苦心しなければならない事柄が生ずる．これらのうちで，とくに苦心しなければならない事柄はサブシステムの言語体系に依存しており，一般的な問題として取り上げることはできない．

そこで，ここではサブシステムの代表例としてKEIO システム(会話型FORTRAN)のVersion 0をとりあげ，これを開発する際にいろいろと面倒をみなければならなかつた問題のうちのいくつかを述べることにする．

4 会話型FORTRANを開発する際に考慮しなければならない幾つかの問題

FORTRANに限らず，一般に，会話型のプロセッサを開発する際の基本的精神としては，次のようなものがある．

- 使用者からの入力があつた時点で，できる限りの診断を行ない，もし誤りがあつたときには，折り返えしその旨を知らせること．
- プログラムの修正が簡単にできること．システムがその作業に要する時間は，できる

限り短くなければならない。

- プログラミングに必要な情報は，即座にとり出せるようになつていること。
- 少くともプログラミング言語と同一レベルの言語で，プログラムのデイバツクができること。

4.1 KEIO システムの構成(Version 0)

KEIOシステムを構成しているプログラムを大別すると次のようになる。

- モニタ……以下の各ルーテインで共通に使用する各種テーブル，分岐ベクトル等から成る。
- 構文解読ルーテイン……ステートメントの文法チエツクと各種セルの作成
- つなぎ処理ルーテイン……構文解読ルーテインにより生成された1ステートメントに関する各種セルを，それまでに作られた使用者のプログラムにつなげる。
- 解読実行ルーテイン……構文解読ルーテインおよびつなぎ処理ルーテインによつて変換されたプログラムを解読し実行する。
- コマンド処理ルーテイン……使用者から出された各種のコマンドに対するサービスを行なう。プログラムの修正，ソース プログラムのリステイング等はこのルーテインが行なう。

Fig.2 はKEIO システムの構成図を示す。

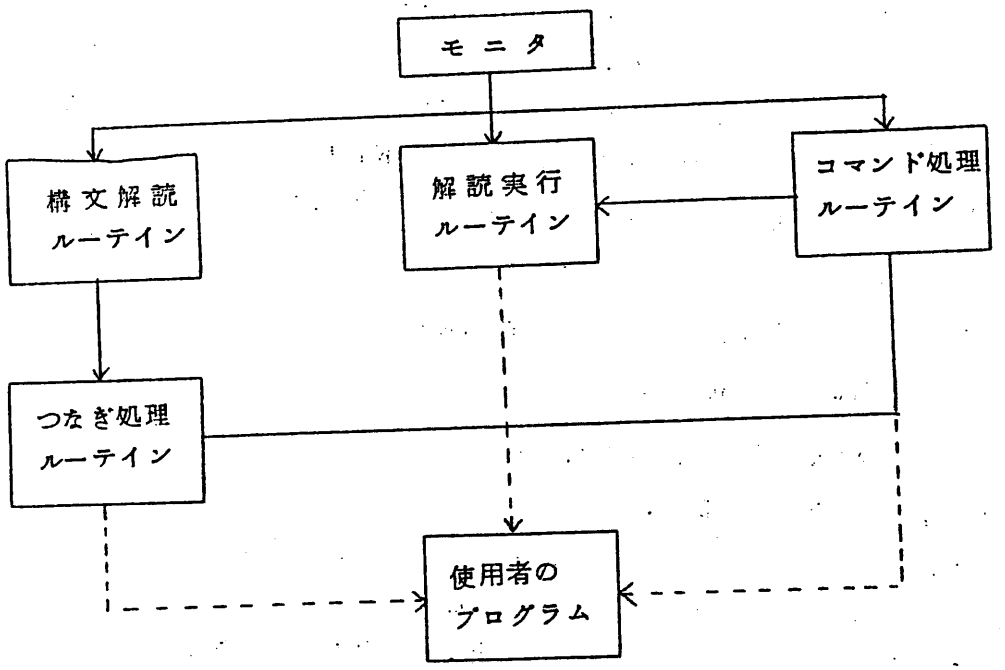


Fig.2 KEIO システムの構成図

結局，モニタで使用者からのメッセージをしらべ，使用者の内部での状態と照らし合わせて，必要なルーテインを呼び出し，そのルーテインにコントロールを移すことになる．

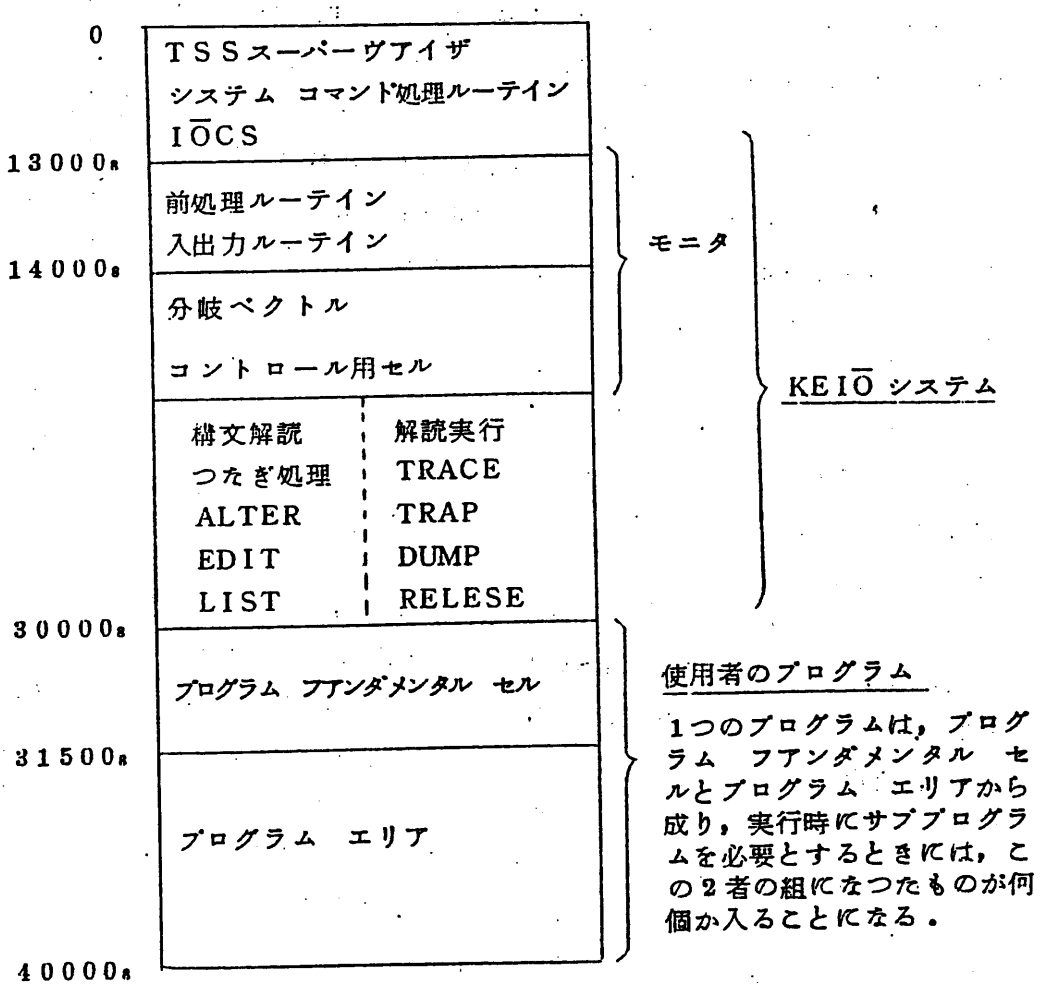
なお，KEIO システムはシステム テープには3レコードに分けて保持されている．各レコードの構成は次の通り．

レコード1： モニタ

レコード2： 構文解読ルーテイン，つなぎ処理ルーテイン，コマンド処理ルーテイン（LIST, ALTER, EDIT）

レコード3： 解読実行ルーテイン，コマンド処理ルーテイン（DUMP, TRACE, TRAP, RELEASE）

KEIO システムを使用しているとき，記憶装置はFig.3 のような構成になる．



4.2 幾つかの問題点

使用者の1つのプログラムは前述の通り，プログラム フアンダメンタル セルとプログラム エリアから成る．さらに，プログラム エリアは多数のステートメント セルとエレメント セルによつて構成されている．各ステートメント セルおよびエレメント セルはプログラム フアンダメンタル セルを中心としたチエインで結ばれている，

ステートメントのセル化

まず，プログラムの開始に先だつて，プログラム フアンダメンタル セルを用意する．次に，使用者が入力したソース ステートメントを順次構文解読ルーテインでステートメント セルとエレメント セルに分解する．このとき，構文解読ルーテインでは，これら各セルをプログラム エリアにチエインする作業は行なわず，仮に設けた別のエリアに生成して行く．この仮のエリアにはTSCR (Temporary Statement Cell Area) とTECA (Temporary Element Cell Area)の2種類のものがある．

TSCA には，ステートメントの種類（算術ステートメントとかGO TOステートメント等），各ステートメントの逆ポーランド記法表示，ステートメント番号などを，プログラム フアンダメンタル セルにあるエレメント セル インデックスやモニタ内のプログラム ベイシック セルを仲介として，ステートメントのオブジエクト プログラムに相当するものを生成する．

TESR には，初めて現われたエレメント（変数名，定数，ステートメント番号等）に関するセルを生成する．このとき，ステートメント セルで引用できるようエレメント セル インデックスを確保する．エレメントがすでに登録済のものであれば，プログラム エリアにあるエレメント セルを直接引用する．すなわち，エレメント セルのチエインの集合は通常のシンボル テーブル等に相当する．

1ステートメントが無事にセル化できたときには，つなぎ処理ルーテインにコントロールが移り，このルーテインがTSCA およびTECA 内のセルをプログラム エリア内に移す．プログラム エリアは，通常，連続的に使用して行くが，ステートメントを挿入したり，削除したりされると次第に虫喰い状態になつて行く．そこで，その空間を能率よく使用するために，使用可能なエリアの表を作り，各セルをプログラム エリアに移すときには，この表をしらべて一番よく合う空間を捜し出して使用する．この表の中に適当なものがないときに限り，残りの広場を使用する．エリアを解除するときには，その前後が空間としてすでに登録されている場合には，表の内容を書きかえて1つの空間としてまとめることをする．もし表が一杯になつた場合，および空間の総語数が一定量をこえたとき，およびプログラム エリアの残りの場所がなくなつてしまつたときなどにはゴミ集めを行い，表をカラにする．

以上のような手順でオブジエクト　プログラムを生成していくとき，ステートメントのエラー　チエツクは，ステートメント自体に対するものと，ステートメント間のつながり方に対するものについて行い，それぞれ構文解読ルーテイン，つなぎ処理ルーテインで行なう．

また，使用者がプログラムを作成しているときのモードには，プログラム　モードと修正モードがある．前者はプログラムを普通に作成していくときのモードであり，後者はすでに作られているプログラムの途中にステートメントを挿入しているときのモードである．エラーのチエツクは，このモードによつても異なる．すなわち，プログラム　モードの場合には，通常のコンパイラと同じ方式でよいが，修正モードの場合には，余分なチエツクが必要になる．

構文解読時の問題点

このときには，TSCA およびTECA 中のセルはすべてないものとしなければならないが，これが案外やつかいな問題である．

すなわち，すでにあつたエレメント　セルを引用している場合には，そのセルに，このステートメントが与えた情報はキヤンセルする必要がある．このためには，各セルについて再度頭からスキヤンしなければならない．したがつて，この種の情報は，そのステートメントにエラーがないということが明確になつてはじめて与えるようにしなければならない．

ステートメントの挿入時の幾つかの問題点

基本的には，作業エリアに作られている各種のセルをプログラム　エリアに移して，挿入するセルの前後の関係のつなぎを正しくとるだけのことであるが，ステートメントの種類によつては少々問題になる．

・DIMENSIŌNステートメントの挿入

まだプログラムに一度も使用されていない配列名が宣言されたときには問題ないが，以前に宣言されていたものが削除されて未定義になつていたときには記憶場所の割り付け等に関して問題となる．たとえば，次元数が変更されたり，広さが変わつたりしたような事態を想像してみれば，ちよつとばかりいやらしいことが起ることが，わかつていただけることと思う．

・FUNCTIŌN　SUBRŌUTINEステートメントの挿入

このステートメントが先頭に挿入されると，プログラムは，突如として主プログラムからサブプログラムに変わることになる．さらに，仮引数があつてしかもその変数名がすでに使用されていた場合には，この変数名の性格も変えられたことになる．この種の変更に対す

る処置もうまくできるようにしておかなければならない．

・DOステートメント挿入

挿入に関しては一番いやなものがDO ステートメントに関するものである．たとえば，次のような場合を考えればよい．

例

```
      :
      DO  2  I=1, J
      DO  3  K=1, L   ← 挿入しようとしているステートメント
      :
2     A=B
      :
3     C=D
      :
```

例

```
      :
      DO   2   I=1, J
      :
2     CONTINUE
      :
      DO  2  K=1, L   ←挿入しようとしているステートメント
      :
```

例

```
      :
10    A=B   ←― 挿入しようとしているステートメント．
      :
      DO  10  I=1, J
      :
10    C=D   ←― 削除されたステートメント
      :
```

この他様々な場合が考えられるが，このような場合に対する処理方法は相当うまく考えておく必要がある．

ステートメントの削除時の幾つかの問題点

ステートメントが削除される場合には，単にステートメント　セルのチエインの変更だ

けに留まらず，プログラムの性格やエレメントの性質に影響を与えることがある．

・SUBROUTIN FUNCTION ステートメントの削除

この場合には，いままでサブプログラムであつたものが主プログラムに変わることからプログラムの性格を変更する必要がある．さらに，仮引数は普通の変数に変わり，また，FUNCTION ステートメントの場合には，関数名は変数名に変わることになる．

・ステートメント関数定義式の削除

ステートメント関数の定義式を削除する場合，その関数が他で引用されていなければ問題はないが，一回でも引用されていたときには，その関数名の処置が問題となる．定義式を削除した結果，プログラム全体を見渡せば，成程これまでのステートメント関数名はFUNCTIONサブプログラム名に変じたことになるが，そのエレメントの性質を変更することは処理上，かなりの問題となつてくる．そこでKEIO システムのVersion 0では，ステートメントの挿入，削除をどのように行なつた場合でも，ステートメントのエレメントの性質は変えないという方針をとつている．したがつて，すでに使用しているエレメントの性質を変更する場合には，引用しているステートメントも修正する必要がある．この精神は，あくまでも turn—around time を第1としていることであるから，Version 1 以降においては，変更することもあり得る．

・DIMENSION ステートメントの削除

この場合も，ステートメント関数の定義式を削除した理由も含み，すでに，プログラム内に現われている場合には，配列名は未定義の状態で放置している．ただし，削除された配列名がまつたくプログラム中に使用されていないときには，関係するものすべてを削除する．

・DOステートメントの削除

ステートメント番号のエレメント　セルには，DOの範囲に関するインデイケータがある．DOステートメントを削除したために，そのステートメント番号がDOの範囲を示すものではないものに変わつてしまつた場合には，そのインデイケータを解除しておく必要がある．このとき，何重にも入れ子になつて居り，しかも端末ステートメントを共通にしている場合もあることも当然あるので，むやみにこのインデイケータを解除することはできない．

・エレメントの削除

1つのステートメントは，ステートメント　セルとエレメント　セルに分解するということは前に述べたが，同一のエレメントに対しては1つのセルしか生成しない．ステート

メントを削除するときに，ステートメント　セルだけを解除していると，ステートメントの削除，挿入を繰返していくうちに，プログラムに存在しないようなエレメントのセルが残つてしまう．そこで，プログラム　エリアを有効に使用するためには，引用回数のカウンタをエレメント　セルに設け，このカウンタがゼロになつたときには，このセルを解除してしまうような方法をとる必要がある．

以上幾つかの問題点を述べたが，この他にも，バッチ処理方式のコンパイラでは考えられないような様々な事態が生ずる．さらに，このようなオブジエクト　プログラムを生成した場合には，幾つものサブプログラムを使用するようなときのリンケージのとり方などにもいろいろと工夫を要する．これらの点に関しては，また別の機会に述べることにして，ここでは省略させていただく．

5 むすび

成程，我々のシステムはタイムシエアリング　システムとしては小型のシステムではあるが，我々としては十分満足している．また，小型ではあるが，そこで得た経験は大型のシステムに対しても十分役立つものであると思う．すでに，DN−230（端局制御装置）およびDN−510，DN−511（端局）の開発が完了して居り，しかも，300K語の磁気ドラムをシステムに加え，スーパーヴアイザをはじめとし，各サブシステムもVersion 0での経験をもとに，徹底的に改良したVersion 1を開発の途上にある．このVersion 1では，端局の数も5〜10台に増加し，カーブ　プロッタも端局から使用でき，しかも，バッチ　ジョブをバック　グラウンドとしてもつ，本格的なシステムである．

この，KEIŌ−TŌSBAC タイムシエアリング　システム Version 1 は本年6月ごろ完成の予定である．

これらの経験が少しでも世の中の御役に立てばと思う次第である．

参考文献

〔1〕土居範久　他：'KEIŌ−TŌSBAC タイムシエアリング　システム：スーパーヴアイザと端局制御'，情報処理　1968年3月号

〔2〕浦昭二，土居範久，原田賢一：'タイムシエアリング　システム'，数学セミナー　1967年10月−12月

連　絡　先

社団
法人　情報処理学会

東京都港区芝公園第21号地1-5
機械振興会館内

TEL（434）8211 内347
直通（431）2808

発　行

1968-2-20(350)