会話型 FORTRAN : KEIO システム*

土居　範久**　原田　賢一***　中西　正和**

Abstract

The KEIO (Keio Elementary Instructive Operating) system has been developed for the conversational FORTRAN processor, one of the systems of the KEIO-TOSBAC Time-Sharing System at Keio University. The purpose of this system development is experimental research for software system with man-machine interaction. The followings are the functions required on the system; immediate error diagnoses, easy correcting and modifying without recompilation, debugging aids at source language level, accessibility of user program information, etc. On the implementation we are imposed such restrictions as a shortage of small size main memory, a lack of random access mass-storage and no dynamic relocation hardware. Furthermore, the user is allowed to use only 4K words drum area, in which all information of the user program must be hold. Considering these situation, we adopted the incremental compilation and interpretive-execution technique into the processor.

In this paper, we describe the organization and internal design of the system and user program, after the outline of KEIO system.

0. 概　説

KEIO システム (Keio Elementary Instructive Operating system) は KEIO-TOSBAC タイムシェアリング システムに開発した会話型 FORTRAN システムである．タイムシェアリングの特徴を大いに発揮させるために，システムを設計する際に考慮した点は次のとおり．

- 使用者から入力があった時点で，できる限りの診断を行ない，もし誤りがあったときには折りか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．しかも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こと．
- 少なくともプログラミング言語と同一レベルの言語でディバッグできること．

また，KEIO-TOSBAC タイムシェアリング システムの開発目的および性質から，次のような事柄についても考慮しなければならなかった．

- このタイムシェアリング システムの開発目的は，電子計算機に基づく協同体系 (computer utility) の実験，研究である．
- 人間と機械との緊密な通信機能を備えたソフトウェアを開発すること．
- このシステム (Version 0) は小規模の機器構成〔1〕であり，大容量ランダム呼出し記憶装置を備えていない．
- 使用者のプログラム交換には磁気ドラムを使用するが，その容量は端末当り約 4 K語 (24ビット/語) しかない．端末で作業をしている使用者のプログラムに関する情報はすべて，この中に納めなければならない．

FORTRAN 言語の処理方式には，大きく分けて従来のバッチ処理システムで広く採用されていた変換・実行方式と解読・通訳実行方式とがあるが，KEIO システムでは上記の条件を満足し，制約内で最大の機能をもたせるために後者を採用した．この方式は変換・実行方式にくらべて，使用者のプログラムの実行速度が遅くなることと，実行時には通訳ルーチンが必要であることから，使用者のプログラムの大きさは，かなり制約されるという欠点がある．しかし使用者と計算機との高度な交互作用 (interaction) を達成するため

* The conversational FORTRAN: KEIO system, by Norihisa Doi, Ken'ich Harada and Masakazu Nakanishi (Faculty of Engineering, Keio University)
** 慶応義塾大学工学部管理工学科
*** 慶応義塾大学工学部中央試験所計算センター

には，会話型の処理プログラムが必要であり，これには解読・通訳実行方式の方が何かと好都合である．

1. KEIO の言語仕様

KEIO の言語仕様は，FORTRAN II が基本ではあるが，使用者が利用できる入出力装置は端末しかないこと，および実験的なシステムであることから，多少の追加，修正が施してある．KEIO の言語仕様は**Table 1**のとおり．INPUT ステートメントは端末からデータを読み込むのに対して，READ ステートメントは DATA ステートメントで与えられたデータのリストからデータを読み込む〔8〕．CRLF ステートメントは復帰改行をm回指定するステートメントである．

プログラミング時に修正をしたり，追跡をしたり，あるいはプログラムに関する情報を獲得したりするための手段として用意したコマンドを**Table 2**に示す．ステートメントの位置を指定する必要があるコマンドでは，システムまたは使用者によって，各ステートメントにつけられている行番号（l；通常10から10きざみ）を用いて指定する．

Table 1　KEIO Statements

$v=e$
GO TO n
GO TO $(n_1, n_2, \cdots, n_m)$, i
IF (e) n_1, n_2, n_3
DO n $i=m_1$, m_2, m_3　　(m_3 may be omitted)
CONTINUE
STOP k
PAUSE k
READ v_1, v_2, $\cdots$, v_n
INPUT v_1, v_2, $\cdots$, v_n
PRINT n, v_1, v_2, $\cdots$, v_n
CRLF m
FORMAT $(S_1, S_2, \cdots, S_m/\cdots/\cdots, S_n)$
FORMAT * $j_1(S_1)\, j_2(S_2) \cdots j_n(S_n)$
DIMENSION a_1, a_2, $\cdots$, a_n
COMMON b_1, b_2, $\cdots$, b_n
DATA $(c_1, c_2, \cdots, c_n)$
ASSIGN $(v_1/c_1, v_2/c_2, \cdots, v_n/c_n)$
FUNCTION Name $(d_1, d_2, \cdots, d_n)$
SUBROUTINE Name $(d_1, d_2, \cdots, d_n)$
RETURN
CALL Name $(e_1, e_2, \cdots, e_n)$
Name $(d_1, d_2, \cdots, d_n)=e$
END

v: variable, e: arithmetic expression, n: statement number, i: integer variable, m: integer constant or variable, j: integer constant (may be omitted), k: octal constant, a: array declarator, b: variable or array declarator, s: field descripter, c: constant, d: variable or array, Name: subprogram name.

Table 2　Commands

一　般　形	機　　能
ALTER l_1〔, l_2〕	ステートメント l_1 から l_2 までを削除して，モード〔1〕を修正モードに変更する．そこにステートメントを挿入することもできる．l_2 がないときには l_1 以降にステートメントを挿入することを意味する．
EXIT	モードを修正モードからプログラミング　モードにもどす．
LIST〔l_1〔, l_2〕〕	ステートメント l_1（から l_2 まで）のリストをとる．行番号の指定がなければプログラム全部をリストする．
RESEQUENCE n_1〔, n_2〕	ステートメントの行番号を n_1 から n_2 きざみで再定義し，そのリストをとる．
SCRATCH	それまで扱っていたプログラムを抹消する．
START〔l_1〕	プログラムを先頭または中断点または l_1 から実行開始する．
TRACE〔(l_1〔, l_2〕)〔$v_1, v_2, \cdots, v_n$〕〕	実行時にステートメント l_1 から l_2 の範囲を変数 v_1, v_2, $\cdots$, v_n について，その値を追跡する．何の指定もないときにはプログラム全体にわたって全変数を追跡する．
TRAP〔l_1〔, l_2〕〕	ステートメント l_1 から l_2 に含まれる制御ステートメントについてコントロールの流れを追跡する．
RELEASE〔l_1〔, l_2〕〕	ステートメント l_1 から l_2 までの範囲に指定した TRACE または TRAP の効果を解除する．
DUMP〔$v_1, v_2, \cdots, v_n$〕	変数に与えられている値をダンプする．
EDIT s	DUMP および TRACE における数値の印字形式を指定する．s は実数型または整数型の欄記述子．

l は行番号，v は変数名，s は欄記述子である．〔 〕は省略可能であることを示す．

2. 使用者のプログラムの内部構成

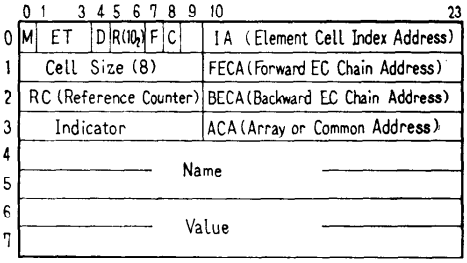
KEIO システムでは，解読・通訳実行方式の採用によって生じるプログラム実行速度の低下を防ぐために，ステートメントを走査（scan）して中間言語を生成するルーチン（構文解読ルーチン）において，入力された1ステートメントについての文法チェックを行なうと同時に，実行時の通訳ルーチン（解読実行ルーチン）に対する負荷が最小になるように，ステートメントの構成要素を整理した上で各種の“セル”を生成する．使用者のプログラムとしてはこれらセル化した中間言語だけを保持し，ソース　イメージは捨ててしまう．

したがって，使用者からプログラムのリストを要求されたときには，この中間言語からソース　イメージを再生して出力する．

使用者のプログラムを構成するセルには，エレメント　セルとステートメント　セルとがある．エレメント　セルはプログラムの構成要素に関する情報を構成要素ごとにまとめたものであり，ステートメント　セルはソース　ステートメントをその構成要素に分解して，解読ルーチンで処理しやすい順序に並べかえたもので，1ステートメントについてステートメント　セルを1つ生成する．

2.1 エレメント セル

エレメント セル(EC) は8語からなっており，英字ではじまる構成要素については，先頭の文字についてアルファベット順に26個のチェインを作り，この他に実定数，整定数，ステートメント番号に対してもそれぞれチェインを作る．EC の構成は **Fig. 1** のとおり．1つのエレメント セルには1つのインデックス(1語)を与え（プログラム ファンダメンタル セル(2.3 参照) 中にある)，ステートメント セルで構成要素を参照する場合にはこのインデックスを通して行なう．



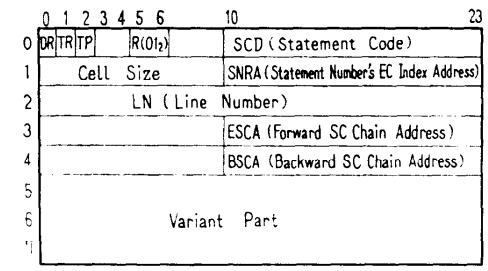
FECA 次のエレメント セルの番地.
BECA 1つ前のエレメント セルの番地.
IA このセルに与えられたインデックスの番地.
ACA 配列あるいは共通領域に割り付ける変数の場合にだけ使用し，その割付番地を保持する.
M 構成要素の型（実数型か整数型）を示す.
ET 構成要素の種類（変数名，配列名，定数，ライブラリ関数名，サブプログラム名またはステートメント関数名）を示す.
D 配列名のとき，その次元数を示す.
R エレメント セルであることを示す.
C 共通領域に割り付けるかどうかを示す.
F 構成要素の定義の有無を示す.
RC 各ステートメント セルで，このセルを参照している回数を示す.
Name 構成要素名(定数のときは内部表示形).
Value 変数および関数名のときには，実行時にその値を保持する．配列名のときには配列要素の数と各次元の大きさを保持する.
Indicator 文法チェックあるいは実行時のチェックに必要な補助的な情報を保持する.

Fig. 1 The structure of the Element Cell

2.2 ステートメント セル

ステートメント セル(SC) は可変長のブロックであるが，先頭の5語はステートメントの種類，行番号，前後関係を示す情報を保持している固定部分である．SC は行番号の順にチェインする．SC の構成は **Fig. 2** のとおり．CONTINUE, RETURN, および END ステートメントの場合には固定部分だけで可変部分はない.

各ステートメントの構成要素は，一般に，変数名，定数，関数名，およびステートメント番号の場合には，その EC のインデックス番地で表わし，オペレータの場合にはモニタが保持しているオペレータ基本セル（後述）の番地で表わす（**Fig. 3** 参照）．算術式はすべて逆ポーランド記法による表現形式をとる.



LN ステートメントの行番号.
FSCA 次の SC の番地.
BSCA 1つ前の SC の番地.
SCD ステートメントの種類を示す表示子．実際には後述のステートメント基本セルの番地がはいる.
SNRA そのステートメントにステートメント番号がつけられているとき，対応する EC のインデックスの番地がはいる.
DR DO の端末ステートメントであるかどうかを示す.
TR このステートメントの TRACE を行なうかどうかを示す.
TP このステートメントの TRAP を行なうかどうかを示す.
R このセルが SC であることを示す特別な情報 (01_2).
Variant Part 可変部分．ステートメントの種類によって形式が定まっている.

Fig. 2 The Stucture of the Statement Cell

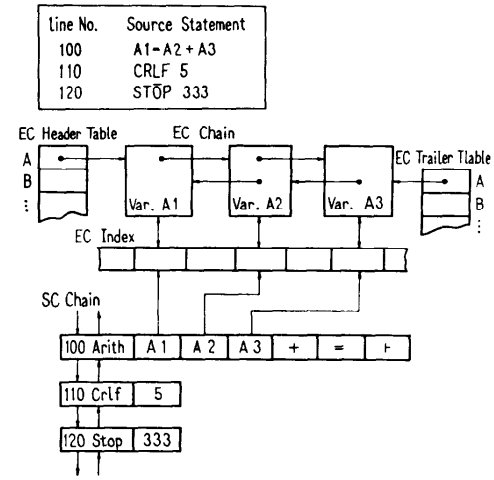
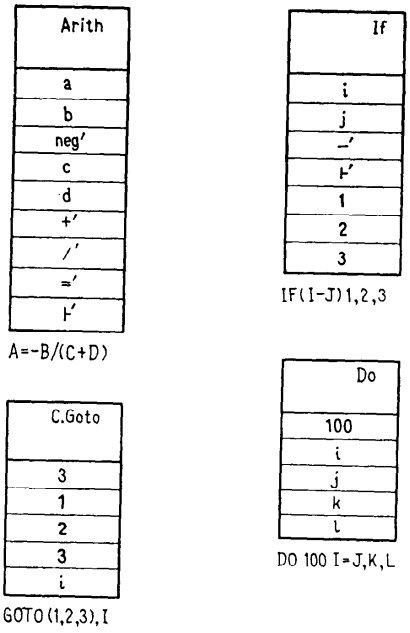


Fig. 3 The interrelation of SC and EC

FORMAT ステートメントおよびコメントはソースイメージのままで可変部分に入れておく．DATA ステートメントに羅列された定数は内部表示形に変換して可変部分に入れる．ステートメントの内部表現形式の例を **Fig. 4** に示す.

2.3 プログラム 領域とファンダメンタル セル

使用者の1つのプログラムは，プログラム ファン



A lower case and a bold face letter stand for the index address of EC, and a prime stands for the address of the operater basic cell

Fig. 4　Examples of the internal form of some statements

ダメンタル　セル(FC) とプログラム領域とから構成される．FC の大きさは一定で，次のようなプログラム全体に関する情報から成る．

・オペレーティング　モード　インディケータ　プログラミング　モードか修正　モードかを示す．
・行番号　次の入力ステートメントに，対する行番号．
・空間表　プログラム領域の途中にできた使用可能な記憶場所を登録．
・追跡表　TRACE コマンドによって特定の変数について追跡が指示されたときに，その EC の番地を登録．この表は実行時に，解読実行ルーチンが使用する．
・EC の先頭表と最後尾表　EC チェイン（頭文字別）の先頭と最後の番地．
・EC インデックス (2.1 参照)
・プログラム領域指示子　プログラム領域の最後の番地．
・共通領域割付けカウンタ
・外部参照表　使用者のプログラムで参照している外部手続名を登録．
・ディバッグ用欄記述子　ディバッグのために，システムが数値を編集して出力する際に用いる欄記述子．

プログラム領域は SC および EC のチェインを作成するための場所である．この領域にこれら2種類のセルが混り合って配置される．ステートメントが逐次入力されていくときには，通常，プログラム領域は連続的に使用していくが，ステートメントの削除によって空間ができたときには，以後の入力ステートメントに対して，それらの空間をできる限り有効に使用するように番地を割り付ける．

3. KEIO システムの構成

KEIO システムの構成要素を大別すると，次のとおり．

・モニタ　KEIO システムに対する入力の識別，コマンドの振分けを行なうルーチン，および以下の各ルーチンで共通に使用する各種サブルーチン，各種の表，分岐ベクトル等からなる．
・構文解読ルーチン　入力ステートメントの文法をチェックし，各種セルを生成する．
・つなぎ処理ルーチン　構文解読ルーチンによって生成された1ステートメントに関する各種のセルを，それまでに作られた使用者のプログラム領域につなげる．
・解読実行ルーチン　構文解読ルーチンおよびつなぎ処理ルーチンによって変換された使用者のプログラムを解読し実行する．このときプログラムの実行に関するチェックを行ない，TRACE あるいは TRAP コマンドによって指定された必要なディバッギング情報がある場合には，それを出力する．
・コマンド処理ルーチン群　使用者からの各種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ィング等はこのルーチンの内の1つが行なう．

KEIO システムの構成図を **Fig. 5** に示す．

上記のルーチンはすべて磁気テープ（あるいは磁気ドラム）に格納されている．KEIO システムが動いている限り，モニタはいつでも主記憶装置内にあり，その他のルーチンは必要に応じて主記憶装置に呼び出される．なお，モニタ以外のルーチンはプログラムの交換に必要な記憶容量を最小にし，システムの効率をよくするために純手続 (pure-procedure) になっている．KEIOシステム使用時の主記憶装置の割り付けを **Fig.**

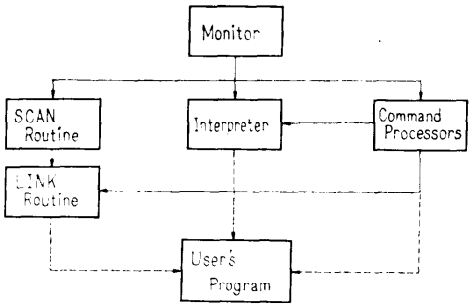


Fig. 5 The configuration of the KEIO system

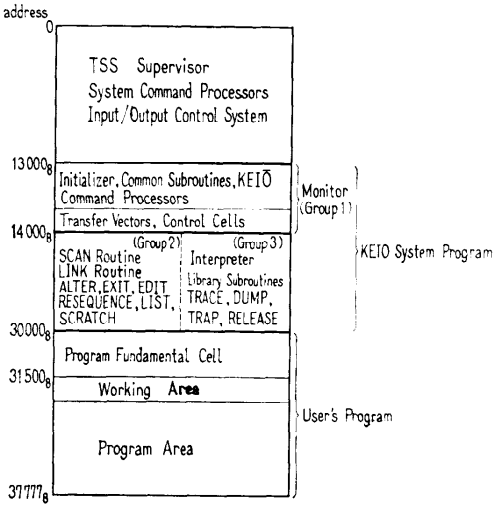


Fig. 6 The memory layout of the KEIO system

6に示す．システムを構成しているルーチンは3つのループに分けられ，1グループが1つのロード単位になっている．

4. モニタ

モニタは幾つかのルーチンと2種の基本セルにから構成されている．

・入力の識別とコマンド振分けルーチン

使用者の状態に従って，入力情報がコマンドか否かを調べ，コマンドの場合にはその正当性をチェックする．適切である場合には対応する処理ルーチンが属するグループを調べ，そのグループが主記憶装置にないときには，TSS スーパーバイザとつなぎをとり呼び込む．その後，分岐ベクトルを用いて対応する処理ルーチンにコントロールを移す．コマンド以外の情報の場合には，TSS スーパーバイザとつなぎをとり，その使用者の入力バッファ内の情報を所定の位置に移し，コントロールを使用者のプログラムにもどす．

・前処理ルーチン

使用者がプログラミングを開始するときや SCRATCH コマンドの処理に必要なルーチンで，プログラム領域および FC 内の表，指示子，スイッチ類を初期化する．

・共通サブルーチン

KEIO システムの入出力方式を標準化するために形式の決まった入出力ルーチン群が用意してある．この他に，エラー メッセージを出力したり，語から文字への分解，文字から語への収集を行なうサブルーチンがある．

・ステートメント基本セル

ステートメントの種類ごとに **Fig. 7** に示すようなセルがあり，全部で 24 個ある．このセルは，SC でその種類を示したり，解読実行ルーチンやコマンド処理ルーチンで各ステートメントに固有の処理をしたりするときに使用する．

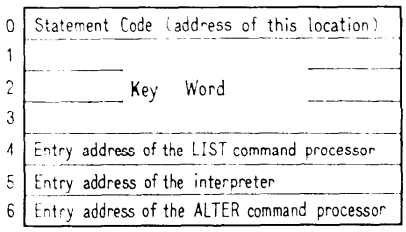


Fig. 7 The Statement Basic Cell

・オペレータ基本セル

プログラムで使用できるオペレータ，および SC への変換時にシステムが生成する特別なオペレータ（ステートメントの最後を示すものなど）に対して，**Fig. 8** に示すようなセルが19個用意してある．

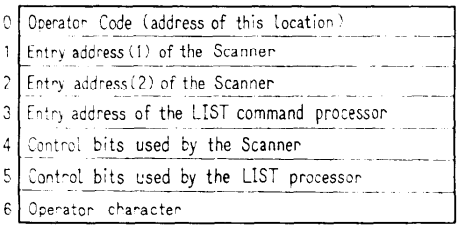


Fig. 8 The Operator Basic Cell

5. 構文解読ルーチン

構文解読ルーチンの機能は，オペレーティング モ

ードとは無関係に，使用者が入力したステートメントを文法チェックしながら SC と EC とに分解していくことである．このとき，構文解読ルーチンでは，生成した各セルを直接プログラム領域につなげる作業は行なわず，仮に設けた別の作業領域に順次生成していく．この作業領域には TSCA (Temporary Statement Cell Area) と TECA (Temporary Element Cell Area) の2種類のものがある．

TSCA には，プログラム FC にある EC インデックスやモニタ内のプログラム基本セルを仲介として，ステートメントのオブジェクト プログラムに相当する SC を生成する．

TECA には，初めて現われた構成要素に関する EC を生成する．このとき，SC で参照できるよう EC インデックスを確保する．構成要素が登録済みのものであれば，プログラム領域内にある EC を直接参照する．すなわち，EC のチェインの集合は通常のシンボル表に相当する．

1ステートメントが無事にセル化できたときには，つなぎ処理ルーチンにコントロールを移す．

エラーが生じたときには，TSCA および TECA 内にそれまでに生成したセルはすべて解消する必要がある．このとき，すでにあった EC を参照するときに，定義済みのビットをセットしたり，参照カウンタを修正しておいたならば，このステートメントがそれぞれの EC に与えた情報はすべてキャンセルしておく必要がある．この種の手続きとしては，各セルについて再度頭からの走査が必要になり，会話型システムではかなりの致命傷になる．そこで，KEIO システムでは，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って初めて与えるようにしている．

5.1 構文解読ルーチンの構成

構文解読ルーチンの概要を **Fig. 9** に示す．おもなルーチンの機能は次のとおり．

・分類処理ルーチン　ステートメントの種類を分類し，ステートメントの種類によってそれぞれの処理ルーチンにコントロールを移す．

・ステートメント処理ルーチン群　各ステートメントの処理を行ない TSCA にオブジェクト プログラム (SC) を生成する．

・ステートメント番号処理サブルーチン　ステートメント番号の処理を行なうもので，未出現のものについては TECA に EC を生成する．

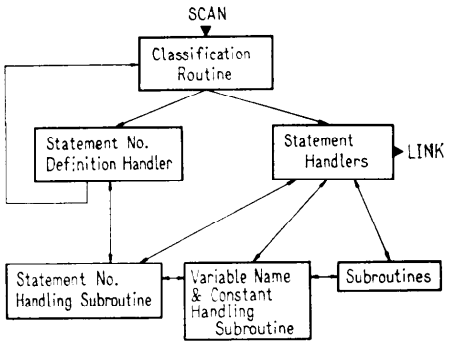


Fig. 9 The block diagram of SCAN routine

・変数および定数処理サブルーチン　構文解読ルーチン中最大のもので，変数と定数の処理を行ない，EC および SC を生成する．

6. つなぎ処理ルーチン

つなぎ処理ルーチンのおもな機能は，TSCA に作られている SC を行番号に関して大きさの順になるように SC チェインに接続することと，TECA に作られている EC を EC チェインに接続することである．各セルをプログラム領域に移す作業と同時に，このルーチンではステートメントの前後関係のチェックを行なう．つなぎ処理の概略的な流れを **Fig. 10** に示す．

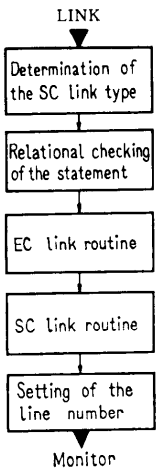


Fig. 10 The block diagram of LINK routine

6.1 SC のつなぎの型の決定

ステートメント間のチェックをし，SC を実際につなげるときのために，SC チェインにおける接続位置を定める．つなぎ処理ルーチンは ALTER コマンドによって挿入されるステートメントも扱うので，TSCA

に作られたSCを単にチェインの最後につなげていくだけでなく，オペレーティング　モードも考慮しなければならない．プログラミング　モードのときにはSCチェインの最後につなげるだけでよい．修正モードのときには，TSCAに作られたSCの行番号をSCチェインの各セルの行番号と比較して挿入位置を決定する．

6.2 エラー　チェック

プログラミング　モードのときには，通常のコンパイラと同じ方式で，すでに処理済みのステートメントと入力されたステートメントとの関係についてチェックする．修正モードのときには**6.1**で決定した挿入位置の前後にあるステートメントとTSCAに作られたステートメントとの関係をチェックする．

6.3 ECの接続

TECAに作られているECを1個ずつECチェインに接続していく．まずECを1個つなげるためにプログラム領域に8語の空間を確保し，その要素の種類によって該当するチェインを決定する．次に，TECAに作られたECをそこへ移しつなぎをとる．つなげようとするECに対するインデックスは構文解読ルーチンで割り当て済みであるから，確保したプログラム領域の先頭番地をそのインデックスに入れる．つなげようとするECが配列名または共通領域に割り付けるべき変数であれば，記憶場所の割付けをし，その割付け番地をECの4語目に格納する．

6.4 SCの接続

SCチェインにつなげようとするセルの語数はその固定部分にはいっている．これをもとに，プログラム領域に必要な大きさの空間を確保し，TSCAに作られているセルを移す．**6.1**で決められたつなぎの型に従って，SCチェインの組みかえを行なう．ECを参照するようなSCのときには，その可変部分をしらべて，該当するECの参照カウンタを1上げる．

6.5 行番号のセット

次の入力ステートメントに対する行番号はプログラムFCの中に保持していて，この情報をもとに次のステートメントの入力を要求する．1ステートメントに関するセルのつなぎ処理が終わりしだい，次のステートメントに対する行番号を用意する．行番号はプログラミング　モードのときには10から10きざみで増加させていく．修正モードのときにはALTERコマンドによって指定された番号から始めて，1ステートメント挿入されるたびに1ずつ増加させていく．

以上の処理によって，1ステートメントが完全にシステムに受け入れられたことになり，次のステートメントの入力を要請するためにスーパーバイザにコントロールを移す．

6.6 プログラム領域の割付け

プログラム領域は，通常，連続的に使用していくが，ステートメントが削除されたり，挿入されたりすると，しだいに虫喰い状態になっていく．そこで，そのような空間を能率よく使用するために使用可能な空間の表(空間表)を作り，各セルをプログラム領域に移すときには，まず，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，残りの場所を使う．使用していた場所を解除するときには，その前後が空間として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．表が一杯になった場合，空間の総語数が一定量を越えたとき，およびプログラム領域の残りの場所がなくなってしまったときにはゴミ集めを行ない表をカラにする．

7. 解読実行ルーチン

解読実行ルーチンはプログラム領域内のSCを，ECを媒介にして解読し，実際に使用者のプログラムを実行するルーチンである．

オブジェクト　プログラム(SCのチェインとECのチェインの集合)は，ソース　イメージに関する情報をすべて保持していることから，実行時間は長くなるが，使用者との会話には都合がよい．追跡したり，分岐状態を調べたりするディバッグ用の機能は，簡単に組み込むことができる．解読実行ルーチンでは，特に，様々なスタックを使用するので，これら各スタックとその機能を次に述べる．

- オペランド用スタック　　逆ポーランド記法で書かれたSCを処理するのに使い，オペランドを積む．DOステートメントの各種パラメタもこれを用いて処理する．
- 一時記憶用スタック　　計算結果を，一時格納する．
- 引数用スタック　　仮引数と実引数との間で値の受け渡しをするのに用いる．
- 関数用スタック　　関数サブプログラムを呼び出すとき，そのときのオペランド用スタックの指示子を積む．このために再帰呼出しが可能である．
- ステートメント関数用スタック　　ステートメン

ト関数が引用される場合，そのもどり先を保持する．

・サブプログラム用スタック　サブプログラムを使用する際のそのもどり先を保持する．

7.1 解読実行ルーチンの構成

解読実行ルーチンの概要を **Fig. 11** に示す．各ルーチンの機能は次のとおり．

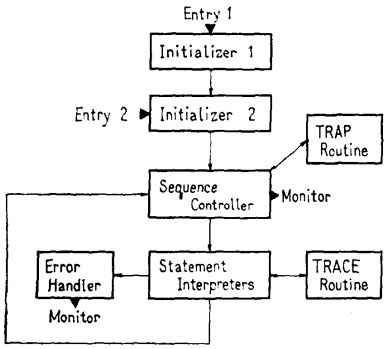


Fig. 11 The block diagram of INTERPRETER

・第1前処理ルーチン

主プログラムを最初から実行するときには，Entry 1 からこのルーチンにはいる．ここでは，すべてのカウンタ，ポインタ，スイッチをリセットするなどすべての前処理を行なう．

・第2前処理ルーチン

割込み，エラー，およびPAUSEステートメントの処理後，再び実行を続けるときには，Entry 2からこのルーチンに直接はいる．続行するために必要とするものだけを残して他のカウンタ，ポインタなどを初期化する．

・順序制御ルーチン

プログラムはリスト構造になっているので，いわゆるフェッチ　サイクルは，このリストをたどることによって行なう．分岐関係のステートメントのときには，各ステートメント通訳ルーチンで順序制御カウンタを修正しておく，そこで，ここでは対応するステートメントの SC の可変部分をさすように修正し，各ステートメント通訳ルーチンにコントロールを移す．TRAP が指定されているときには，ここで TRAP ルーチンとつなぎをとる．

・ステートメント通訳ルーチン群

各ステートメントを解読し実行する．

・エラー処理ルーチン

解読実行中エラーを検出したときには，実行続行が可能な状態でエラー　メッセージを出力することになるが，エラー　メッセージの出力要請自体は順序制御ルーチンがモニタとつなぎをとることによって行なう．そこで，このルーチンでは各種カウンタ，指示子の後始末を行ない．エラーが生じたという旗だけを立てて順序制御ルーチンにコントロールを移す．

7.2 サブプログラムの呼出し処理

各プログラム　セグメントはすべて構文解読およびつなぎ処理ルーチンにより，プログラム領域の先頭から物理的に割り付けられている．したがって，サブプログラムがある場合には，まず主プログラムをロードし，次にサブプログラムをロードすることから，サブプログラムはそれまでにロードしたプログラム　セグメント分だけ修正する必要がある．この修正量はローダーがロード時にサブプログラム名の EC 中に与える．そこで実際には，これらサブプログラムを呼び出す際に解読実行ルーチンが，この修正量だけベース　レジスタを修正することによってサブプログラムを実行する．ベース　レジスタとしてはインデックス　レジスタを使用している．

8. コマンド処理ルーチン

次におもなコマンドについて，その処理方法を述べる．

8.1 ALTER, EXIT コマンド

ステートメントの削除を指示する ALTER コマンドのときには，指定された範囲にある SC を解除した上で，セルの先頭番地と語数を空間表に登録する．削除する SC が EC の参照を含んでいるときには，その EC の参照カウンタを修正する．参照カウンタが零になったとき，すなわちその構成要素がプログラムに存在しなくなったときには，その EC も解除する．

配列宣言子とかステートメント関数定義式が削除され，しかも宣言されていた名前が他で参照されていたときには，各 EC の定義ビットをオフにし，再定義および実行中のチェックに備える．

ALTER　コマンドで指定された位置にステートメントを挿入できるようにするために，いままでの行番号を退避して，コマンドで指定された番号にセットし，オペレーティング　モードを修正モードにする．

セルの解除によってできた空間は，**6.6** のとおり空間表にもとづいて，適宜プログラム領域のゴミ集めを

し，整理する．このとき，実際にプログラム領域の先頭からセルを物理的に移動していくが，この移動によって各セルのチェイン番地とECインデックスの内容を修正する必要がある．SCでECを参照しているとき，そのインデックス番地は固定なので，ECを移動してもSCの内容を修正する必要はない．

EXITコマンドの処理は，退避しておいた行番号を復帰させ，モードをもとにもどすだけである．

8.2 LIST, RESEQUENCE コマンド

KEIOシステムでは，使用者のプログラムのソースイメージは保持しておかないので，リストが要求されると，SCからソースイメージを再生して出力する．

逆ポーランド記法表示になっている算術式を，ソースイメージに再生する手続は次のとおり．

（1） オペレータに着目して，基本演算項ごとにまとめ，木構造を作る．

（2） 最高レベルの演算項から走査を始め，オペランドがオペレータを含む項であれば，それをカッコでくくるかどうかを判定する．このようにして，オペランドとオペレータを並べかえて，ソースステートメントに書かれていた順序にする．．

（3） この並べかえたレコードを最初からながめていき，ECあるいはオペレータ基本セルから実際の文字をとり出して編集し出力する．

したがって，ソースステートメントにおいて意味のないカッコを書くと，リストしたときには省略されたり，定数の表現法が異なっていたりすることがある．

他のステートメントはSCがソースステートメントと同じような形になっているために，ステートメント基本セルからそのキーワードをとり出し，簡単な文字の処理と数値変換によってソースイメージを再生することができる．

RESEQUENCEコマンドの場合には，SCチェインを最初からたぐりながら，指定されたとおりに行番号をふりなおし，LISTコマンド処理ルーチンにコントロールを移して，プログラムのリストを出力する．

8.3 ディバッギング コマンド

TRACE, TRAPコマンドの場合には，指定された範囲にあるSCの固定部分の該当するビットをセットするにすぎない．このビットを解読実行ルーチンで参照する．TRACEコマンドで特定の変数についての追跡が指定されたときには，対応するECのインデックスの番地をプログラムFC内の追跡表に登録する．この表も解読実行ルーチンで参照する．

RELEASEコマンドは指定された範囲にあるSCのTRACEあるいはTRAPビットを解消すると共に，追跡表から指定された変数を削除する．

DUMPコマンドの場合には，ECに格納してある変数名とその値を，EDITコマンドの欄記述子に従って編集し出力する．配列名に関してはその次数と要素の個数とから添字つき変数の形で各要素の値を出力する．

9. むすび

会話型FORTRANシステムを開発していく段階で幾つかの問題に遭遇した．これらの問題は次の2種に大別できる．

・TSSの精神にもとづくもの

・FORTRAN言語仕様にもとづくもの

前者の例としては次のようなものがある．端末にいるものは気の変わりやすい“人間”であるから，いつでも端末に主導権が与えられるようになっている．これは端末にあるインタラプション スイッチ(Version 1ではアテンション キー）をオンにすることによって行なうことができる．この割込みがかけられると，FORTRANシステムはその使用者の実行を中断し，使用者の指示待ちにする必要がある．これはTSSスーパーバイザが勝手に実行を中断するだけではことが済まない．というのは，実行が続行される場合にはこれでもよいが，他のことが開始される場合にはうまくいかない．これは，純手続ではあっても各種状態を示すスイッチやレジスタの類が中途はんぱな状態にあることに原因する．特にわれわれのシステムでは使用者が利用できる領域が小さいので，構文解読，つなぎ処理の作業領域と解読実行の作業領域を共用しているので致命的である．そこで，この割込みはTSSスーパーバイザが論理的な割込みに変え，対応する使用者のプログラムFCの所定の位置にビットをセットしておき，KEIOシステムの方では，1ステートメントに関する作業が終了するごとにこの位置を参照するようにしてある．

後者の例としては，DOの端末ステートメントを削除してしまい，それにつけられていたDOの範囲の終わりを示すステートメント番号を対応するDOステートメントよりも前にあるステートメントにつけたりするような場合である．この種の事態をひき起こす犯人は決まっており，SUBROUTINE, FUNCTION, DI-

MENSION ステートメントと DO 関係およびステートメント関数定義式に限られている．どのような事態が発生しても驚きはしないが，処理時間が問題になる．したがって，この種のものに関しては，多少の制約があってもかまわないであろう．

最後に，この論文を書くに至るまでに直接ご指導いただいた慶応義塾大学工学部　浦昭二教授，およびシステムの開発に際し，ご協力をいただいた慶応義塾大学大学院工学研究科　大野義夫，石渡裕之，松本雅雄の諸兄および東京芝浦電気株式会社電子計算機事業部第二電子計算機課の諸氏に感謝する．

（昭和43年11月26日受付）

参考文献

1) 土居範久，他：KEIO-TOSBAC タイムシェアリング　システム　入出力制御とスーパーバイザ，情報処理 Vol. 9 No. 2 3月号 (1968)
2) T. M. Dunn and J. H. Morrissey: Remote Computing-An Experimental System Part 1: External Specifications SJCC, 1964.
3) J. M. Keller, E. E. Strum, and G. H. Yang: Remote Computing-An Experimental System Part 2: Internal Design
4) 土居範久，原田賢一：タイムシェアリング　システムに関する研究，慶応義塾大学大学院工学研究科修士論文 (1965)〔未発表〕
5) 中西正和，中村鋤也：タイムシェアリング　システム用コンパイラ．慶応義塾大学工学部管理工学科卒業論文 (1965)〔未発表〕
6) 土居範久，原田賢一：入門タイムシェアリングシステム，KEIO/TOSHIBA TSS. TR-2 (1967)
7) 土居範久：KEIO-TOSBAC タイムシェアリング　システム，情報処理月例会資料33 (1968)
8) Dartmouth College Computation Center: BASIC, June 1965.
9) 浦昭二，土居範久：KEIO-TOSBAC タイムシェアリング　システム，EDP アプリケーションハンドブック，日刊工業新聞社 (1968).

会　員　の　声

一松氏の二進加法のプログラムについて

西　村　恕　彦*

本誌9巻3号 160 ページの二進加法のプログラムの説明に，「COMMON IA, IB でリンクする」と述べてあるが，この 6805 のプログラムも，そのもとの 6801 のプログラムも，IA, IB が共通ブロックにはいっているという性質は，まったく利用されていないようだから，上記の文は取り除くべきでしょう．

ついでに気を回わすと，この原著者は，「副プログラムにおいては，仮引数の英字名を COMMON 文に書いてはならない」という規約を，無用の制限として無視したがっているのではあるまいか．

（昭和 43 年 10 月 2 日受付）

西村氏の指摘に答えて

一　松　　信**

私の作った二進加法のプログラムに関する西村恕彦氏の指摘 (COMMON IA, IB は不要) はまったくそのとおりです，別に深い理由があるわけでも，また制限を無視したがっているわけでもなく，COMMON の意味を誤解していたためです．もちろんこれは不勉強と早合点のせいですが，独学で文法書だけで勉強した人間は，このような誤解をよくやるものです．だからこそ，中級ないし高級プログラミングの解説を，本誌あたりで積極的にとりあげ，我流のプログラムで非能率的な使い方をしている大勢のプログラムを再教育して下さることを強く希望するものです．

（昭和 43 年 11 月 11 日受付）

* 通産省工業技術院
** 立教大学理学部