

J A E R I - M  
93-155

FORTRANプログラム静的解析ツール  
ANALYSIS/EXの開発

1993年8月

長内 誠志\*・横川三津夫

JAERI-Mレポートは、日本原子力研究所が不定期に公刊している研究報告書です。  
入手の問い合わせは、日本原子力研究所技術情報部情報資料課（〒319-11 茨城県那珂郡東海村）あて、  
お申しこみください。なお、このほかに財団法人原子力弘済会資料センター（〒319-11 茨城県那珂郡  
東海村日本原子力研究所内）で複写による実費領布をおこなっております。

JAERI-M reports are issued irregularly.  
Inquiries about availability of the reports should be addressed to Information Division Department  
of Technical Information, Japan Atomic Energy Research Institute, Tokaimura, Naka-gun, Ibaraki-  
ken 319-11, Japan.

© Japan Atomic Energy Research Institute, 1993

編集兼発行 日本原子力研究所  
印 刷 ニッセイエプロ株式会社

FORTRANプログラム静的解析ツール ANALYSIS/EX の開発

日本原子力研究所東海研究所情報システムセンター

長内 誠志\*・横川三津夫

(1993年7月14日受理)

ANALYSIS は、FORTRAN プログラムの木構造や COMMON 領域の静的解析を行うためのソフトウェア・ツールである。新 FORTRAN コンパイラ FORTRAN77EX(V12)の導入とともに、従来の ANALYSIS の解析機能を拡張した ANALYSIS/EX を開発した。

ANALYSIS/EX では、従来の ANALYSIS の解析機能に加え、新コンパイラの文法に対応するとともに、大規模な原子力コードの解析を可能にした。また、COMMON 領域の詳細解析機能を追加した。

本報告書では、FORTRAN プログラム静的解析ツール ANALYSIS/EX について、改良方法や拡張機能について述べる。

Development of the Static Analyzer ANALYSIS/EX  
for FORTRAN Programs

Seiji OSANAI\* and Mitsuo YOKOKAWA

Computing and Information Systems Center  
Tokai Research Establishment  
Japan Atomic Energy Research Institute  
Tokai-mura, Naka-gun, Ibaraki-ken

(Received July 14, 1993)

The static analyzer "ANALYSIS" is the software tool for analyzing tree structure and COMMON regions of a FORTRAN program statically. With the installation of the new FORTRAN compiler, FORTRAN77EX(V12), to the computer system at JAERI, a new version of ANALYSIS, "ANALYSIS/EX", has been developed to enhance its analyzing functions.

In addition to the conventional functions of ANALYSIS, the ANALYSIS/EX is capable of analyzing of FORTRAN programs written in the FORTRAN77EX(V12) language grammar such as large-scale nuclear codes. The analyzing function of COMMON regions are also improved so as to obtain the relation between variables in COMMON regions in more detail.

In this report, results of improvement and enhanced functions of the static analyzer ANALYSIS/EX are presented.

Keywords: ANALYSIS/EX, Analyzer, Fortran, Nuclear Codes, Software Tool

---

\* On leave from FUJITSU, Ltd.

## 目 次

1. はじめに .....	1
2. FORTRAN77EX への対応 .....	3
2.1 ANALYSIS の構造 .....	3
2.2 ANALYSIS の EX 化 .....	3
2.3 ANALYSIS の AE 化 .....	6
3. 解析機能の拡張 .....	15
3.1 FORTRAN77EX での機能強化項目 .....	15
3.2 FORTRAN77EX 文法対応 .....	16
3.3 ANALYSIS の解析機能拡張 .....	20
4. 保守性・操作性の改善 .....	41
4.1 インクルードファイルの整備 .....	41
4.2 入力データの書式整備 .....	41
4.3 ANALYSIS コマンド .....	42
5. おわりに .....	46
謝 辞 .....	47
参考文献 .....	47
付録A ANALYSIS/EX v1.2 使用手引書 .....	48
付録B ANALYSIS/EX 木構造図 .....	105
付録C ANALYSIS コマンド・プロシジャ .....	119
付録D ANALYSIS カタログド・プロシジャ .....	129

## Contents

1. Introduction .....	1
2. Adaptation of ANALYSIS to the FORTRAN77EX Compiler .....	3
2.1 Structure of ANALYSIS .....	3
2.2 Conversion of Non-compatible Statements .....	3
2.3 Adaptation of ANALYSIS to Address Extension Function .....	6
3. Extension of ANALYSIS Function .....	15
3.1 Enhanced Function of FORTRAN77EX Compiler .....	15
3.2 Adaptation of FORTRAN77EX Grammer to ANALYSIS .....	16
3.3 Enhanced Function of ANALYSIS .....	20
4. Improvement of Management and Utility .....	41
4.1 Reorganization of Include Files .....	41
4.2 Improvement of Input Data Format .....	41
4.3 ANALYSIS Command .....	42
5. Conclusion .....	46
Acknowledgements .....	47
References .....	47
Appendix A ANALYSIS/EX v1.2 Manual .....	48
Appendix B Tree Structure of ANALYSIS/EX v1.2 .....	105
Appendix C Command Procedure List of ANALYSIS/EX v1.2 .....	119
Appendix D Catalogued Procedure List of ANALYSIS/EX v1.2 .....	129

# 1. はじめに

FORTRANプログラム静的解析ツールANALYSISは、日本原子力研究所・情報システムセンター(旧計算センター)が開発、保守を行ってきた、原子力コードの開発や改良のためのソフトウェア・ツールである。ANALYSISは、最新のFORTRAN文法に則した解析を行うことを前提としているため、原研大型計算機のFORTRANコンパイラの変遷とともに、ANALYSISの改良が実施されてきた。ANALYSISの開発状況は以下の通りである(Table 1)。<sup>(1) (2) (3) (4)</sup>

最初のANALYSISは1977年に開発され、当時の標準コンパイラFORTRAN-H、AP-FORTRANの文法で記述されたプログラムの静的解析を可能にした。1980年にはFORTRAN77コンパイラが導入され、ANALYSISの機能拡張が行われた。これはANALYSIS-77として、ユーザに公開されている。

1982年、1984年には、解析オプション等を追加する機能拡張が行われた。これ以降は、小さな改良が行われたが、コンパイラの大きな変更がなかったため、ANALYSIS-77が情報システムセンターの標準ツールとして使用されていた。

一方、1989年には、特定の大規模原子力コード(約7万行)を解析するために、いくつかの解析オプションを追加したANALYSIS専用バージョン(以下、専用バージョン)が開発された。しかし、他の原子力コードへの適用性については十分に検討されておらず、一般ユーザには公開されなかった。

1992年4月、大型計算機の新コンパイラFORTRAN77EX(V12)が原研に導入された。FORTRAN77EXでは、従来のFORTRAN77コンパイラと比較して、10数箇所及ぶ文法上の機能拡張が行われるとともに、FORTRAN77の標準文法(JIS規格)に反するFORTRAN-HE関連の機能が排除されている。このため、原研のFORTRAN77コンパイラ上で動作していたプログラムでも、FORTRAN77EX上では動作しない場合が生じる。また、ANALYSIS-77は、FORTRAN77EX文法の強化機能に対する静的解析機能は持っていない。このため、ANALYSIS-77では解析できない場合があることが分かった。

そこで、ANALYSIS-77の解析機能の見直し、FORTRAN77EXコンパイラへの対応、専用バージョンとの統一、現在の計算機運用環境との整合性をはかるため、ANALYSIS-77に対して改良と解析機能の拡張を施したANALYSIS/EXが開発された。ANALYSIS/EXは既に一般ユーザに公開されている。

本報告書では、第2章でFORTRAN77EXへの対応、第3章で解析機能の拡張、第4章で保守性と操作性の改良について述べる。また、付録には、ANALYSIS/EX V1.2の使用手引書、プログラムの木構造図、ANALYSIS実行用のコマンド・プロシジャ、および、カタログド・プロシジャを添付した。

なお、次章以降では、ANALYSIS全てのバージョンについての共通事項を述べる場合には一般のANALYSISで引用し、バージョン固有の内容について述べる場合に限り、ANALYSIS-77やANALYSIS/EXと表すことにする。

Table 1. History of the Fortran static analyzer "ANALYSIS"

年	A N A L Y S I S 開 発 の 歴 史	標 準 コ ン パ イ ラ	大 型 計 算 機
1970		FORTTRAN-H FORTTRAN-GE FORTTRAN-HE	F230-60
1975		AP-FORTTRAN	F230-75APU
1977	<ul style="list-style-type: none"> <li>最初のANALYSISが発表される。</li> <li>オプション数は8。ANALYSISの総行数約4000。</li> <li>この間、小変更。</li> </ul>		
1980	<ul style="list-style-type: none"> <li>ANALYSIS-77 発表。文法はFORTTRAN77 レベル。</li> </ul>	FORTTRAN77	M200
1982	<ul style="list-style-type: none"> <li>ANALYSIS-77 の機能拡張。オプション数は10。</li> </ul>		M380
1983			
1984	<ul style="list-style-type: none"> <li>この間、小変更の結果、オプション数は14。</li> <li>このバージョンが1992年までセンター標準ツールとして登録される。</li> </ul>		VP100
1987			M780
1989	<ul style="list-style-type: none"> <li>特定大規模プログラム解析専用バージョン開発。</li> <li>オプション数は16へ。一般公開されず。</li> </ul>		
1990			VP2600
1992	<ul style="list-style-type: none"> <li>ANALYSIS/EX v1.1発表。文法はFORTTRAN77。</li> </ul>	FORTTRAN77EX	
1993	<ul style="list-style-type: none"> <li>ANALYSIS/EX v1.2発表。文法はFORTTRAN77/EX。</li> <li>オプション数は19。ANALYSISの総行数約18000。</li> </ul>		



## 2. FORTRAN77EXへの対応

ここでは、ANALYSISを新コンパイラFORTRAN77EX上で動作させるために行った改良について述べる。

### 2.1 ANALYSISの構造

ANALYSISは、FORTRANとアセンブリ言語で書かれたプログラムである。非解析プログラムをFT01F001から、被解析インクルード・ファイルをFT02F001から読み込んで、FT05F001(SYSIN)から与えられるオプションをもとに静的解析を行う。このとき、オプションによってはFT03F001、FT04F001に割り当てられたデータセットを中間ファイルとして用いる。最終的な解析結果はFT06F001に出力される。そのジョブフローをFig. 2.1に示す。

原研の大型計算機が扱うFORTRANプログラムのファイル編成は、順編成(PS)、区分編成(PO)、GEMの3種類である。これらの編成ファイルは、ANALYSIS内でA T - J (Access Tool JAERI)データセットアクセスユティリティ<sup>(5)</sup>を用いて読み込まれる。

A T - Jとは、区分編成・順編成ファイル・アクセスツールである。指定されたファイルのV T O C (Volume Table Of Contents)リストの情報を調査し、それが区分編成ファイル(GEMを含む)の場合は、そのディレクトリ部の情報を調査する。具体的には、ファイルのオープン、クローズ、D C B情報の取得、入出力、メンバ名の取得、メンバの削除、メンバ名変更などの機能を持つ。

次に、ANALYSISのA T - J利用部分を明示し、全体の概要を記す。また、Fig. 2.2にその処理構造のフローチャートを示す。

- ①: 解析オプション(入力データ)を読み込む。
- ②: AT-Jを用いて被解析プログラムのD C B情報を得る。
- ③: ②の情報に応じてそのメンバを1レコードずつ読み込む。
- ④: 名標を種類別にテーブルに格納する。  
②~④は最末尾のレコードまで繰り返し行われる。
- ⑤: ④のテーブルを検索しながら①のオプションに応じた解析をする。
- ⑥: ⑤の編集および出力処理を行う。

### 2.2 ANALYSISのEX化

EX化とは、富士通の従来のコンパイラであるFORTRAN77やFORTRAN-HE等のFORTRAN言語で記述されたプログラムをFORTRAN77EX上で動作するように変換することをいう。

## 2.2.1 FORTRAN77とFORTRAN77EX実行上の非互換項目

FORTRAN77EXでの互換性の基本的な考え方は、FORTRAN77文法書に記載されている仕様は全てサポートするというものである。また、FORTRAN77文法書に記載されていない旧仕様の内、明らかに退化機能と見なされる仕様については、ユーザプログラムの将来に向けての保護、流通性及び処理系開発の軽量化の観点から、サポートを中止している。

FORTRAN77EXでは、以下の仕様を基本的に非互換としている。

- ① 明らかな文法エラーである仕様。
- ② ファイル破壊の可能性のある仕様。
- ③ 旧仕様から新仕様への移行を強く推奨する仕様。

しかし、実運用上はFORTRAN77互換性を維持する必要があることを認識し、次の内容に関してはサポートしている。

- ④ 慣例として処理系が認めており、非互換にすると自由度が大きく下がるもの。
- ⑤ FORTRAN77との媒体非互換を発生させるもの。
- ⑥ FORTRAN77と実行結果が異なるもの。ただし、FORTRAN77では値が不定になるものやFORTRAN77EXで機能拡張したものは除く。

詳細は、平成4年2月に情報システムセンターが催したFORTRAN77EX講習会の資料JAERI-M 92-105<sup>(6)</sup>を参照されたい。

## 2.2.2 ANALYSISのソースプログラムに関する非互換項目

ANALYSISは第1章で述べた通り、今までに複数のコンパイラの仕様に対応してきた。また、ANALYSIS自身もFORTRAN言語で記述されているため、常に最新のコンパイラ仕様で記述され、かつ最新の文法解析をサポートしてきている。このため、ANALYSIS-77のソースプログラムに非互換項目が存在し、FORTRAN77EXコンパイラで動作しなかったため、以下の修正を行った。

### (1) 翻訳時の制限事項に関する修正

FORTRAN77で使用されていたプログラム中の\*INCLUDE行の最後に付けられるオプション(FIXED, INSOURCE等)は、FORTRAN本来の文法に含まれないため許されていない。このため、それらのオプションを削除した。

```

例.   *INCLUDE incmem, FIXED, INSOURCE } オプションを削除
      ↓
      *INCLUDE incmem

```

この修正はサブルーチン ¥PBRAT, ANAL77 に対して行われた。変更内容をFig. 2.3に示す。

## (2) JCL上のSYSINから与えられるオプション制御行

FORTRAN77EXでは、実行JCL上の入力ストリームから与えられるデータを一度読み込んだ後に、入力ストリームに対して、REWIND文やBACKSPACE文の実行はできない。これは、ファイルを破壊する恐れがあるためである。

例.	JCL上	プログラム上
	//SYSIN DD *	READ(5, '(A)') OPT
	SU(1), TR(2), CO(0), CP(1)	.....
	T=MAIN	REWIND 5
	/*	READ(5, '(A)') OPT

ANALYSISでは、FT05F001(SYSIN)から与えられる入力データ（オプション制御行）は、データの種類や用途に応じて、REWINDやBACKSPACEを繰り返しながら、合計3ヶ所で読み込まれている。このままでは入力エラーを出力して途中終了してしまうため、入力データ読み込みルーチン READCD を作成した。

サブルーチンREADCDは、装置機番FT05F001(SYSIN)からの入力データを1行ずつ読み込み、コメント行（1カラム目が\*で始まる場合、空行の場合など）を省いた後、必要なオプション部分のみを配列RECORDへ格納する機能を持つ。このサブルーチンREADCDはANALYSISが最初に入力データを読み込むサブルーチンSETPRMの、入力データを読み込むためのREAD文の直前に組み込んだ。

また、他の入力データを読み込むREAD文をコメントアウトして、サブルーチンREADCDから出力される配列RECORDを、READ文で使用されている変数に渡すように変更した。

これらの作業により、BACKSPACEやREWINDを省略することに成功した。READCDをFig. 2.4にSETPRMへの組み込み部分をFig. 2.5に示す。

## (3) エラーセットルーチンについて

エラーセット(ERRSET)ルーチンは、富士通FORTRANのサービスルーチンの1つで、指定されたエラー識別番号に対応するエラー項目の種々の設定を変更したいときに使用する。このルーチンをFORTRAN77EX対応にするときの問題点は、FORTRAN77とFORTRAN77EXではエラー識別番号が異なることである。ERRSETで指定しているFORTRAN77のエラー識別番号と対応したFORTRAN77EXのエラー識別番号に変更しなければならない。

ANALYSIS内でのERRSETルーチンの設定では、装置機番FT03F001やFT04F001の中間ファイルのスペース量が足りなくなったときに、エラーメッセージを出力させている。このメッセージは、同時にシステムがアウトプットスプール（SORPまたはOUTLISTとも言う）に出力するので、ERRSETルーチンで設定しなくてもよいことが分かった。したがって、ANALYSISの機能から削除（CALL ERRSET の行をコメントアウト）した。ERRSETルーチン呼び出しを削除したルーチンは

サブルーチンANAL77, ERR232と, エントリERR131である。

以上(1)~(3)の変換作業により, FORTRAN77EX上で問題無く動作するようになった。また, EX化版の解析結果は, FORTRAN77版であるANALYSIS-77と全く同一であることを確認した。

## 2.3 ANALYSISのAE化

大規模な原子力コードの静的解析を可能とするためAE化を行った。

### 2.3.1 AE化の定義

AE化とは, 主記憶上の16MB以下の基本領域 (実際にはシステムが8MBを使用するため, ユーザが使用できる領域は8MB)だけで実行していたプログラムに対し, 16MBを越えた拡張領域を使用できるように変換することである。具体的には, アドレス拡張機能を利用することにより,

- ① ロードモジュールの拡張領域での実行
- ② 巨大データの拡張領域への割り付け

が可能となる。

①は, システム制御プログラム (オペレーティング・システム) がアドレス拡張機能を備えている場合に可能であり, 原研の大型計算機システムにおいても可能である。

②は, コンパイラオプションの『AE』を指定して翻訳することにより, 8MBを越えるような特定の巨大配列群を使用できるようになる。拡張領域に割付けられるデータの条件は, COMMONブロックに属する変数及び配列と属さない変数及び配列で, かつ, BLOCKDATAやDATA文で初期値設定されていない変数及び配列である。

### 2.3.2 AE化のための変換

ANALYSIS-77が解析できるプログラムの大きさは, 数万行程度であり, 実行時のリージョンサイズは1MB弱であった。したがって, 原研のM780の最下位クラスやTSS上で比較的簡単に実行できるという利点があった。

ANALYSIS/EXでは, 非解析プログラムの大きさを10万行以上にし, 各名標の長さを31文字対応にするため, リージョンサイズが増加するためAE化を行った。作業としては, 翻訳オプションのAEをつけて翻訳し, リンク時に拡張域に割り当てられないサブルーチンやCOMMON領域を調べ, 拡張域に割り当てられるように変更した。

#### (1) BLOCKDATAのサブルーチン化

BLOCKDATA文が原因で拡張域に割り付けられないCOMMON領域は以下の通りである。

COMMONラベル名

```

  BUFF00,  CMREF$,  COMCHA,  COMINT,
  HEADER,  LENGTH,  SETUP

```

これらのCOMMONは、インクルードファイルのメンバであるBUFFER, DIMENS, SETUPの中で定義されており、BLOCKDATAでDATA文による初期化が行われていることが分かった。そこで、このBLOCKDATAの代わりにサブルーチンBLKDATを作成し、全てのDATA文をFig. 2.6に示すように代入文化した。サブルーチンBLKDATは、Fig. 2.7に示すようにメインプログラムの最初に呼び出すようにした。

## (2) DATA文の代入文化

COMMON変数やCOMMON配列のDATA文による初期値設定を全てコメント化した。コメント化した配列で、整数型と実数型のものは0で、文字型のは空白で、論理型のは.FALSE.で初期化されていたため、代わりにCOMMON領域の初期化ルーチンを作成し、それぞれを代入文で初期化した。作成したルーチン名は以下の通りである。

```

  INITS,  INIT01 (INIT02, INIT03, INIT04)

```

括弧()で示したルーチン名はENTRY名である。サブルーチンINITSをFig. 2.8に、サブルーチンINIT01をFig. 2.9に示す。サブルーチンINITSは、インクルードファイルのメンバPARAMT内で定義されている各COMMON内の変数や配列の長さを採用して、その配列と長さをサブルーチンINIT01およびエントリINIT02~INIT04に渡す。INIT01では整数型変数及び配列を、INIT02では実数型変数及び配列をそれぞれ0で初期化し、INIT03では文字型変数及び配列を空白で初期化する。また、INIT04では、論理型変数及び配列を.FALSE.で初期化する。

なお、ローカルな変数や配列のDATA文による初期化に関しては、大きな配列を初期化している部分がなかったので、拡張域に割り当てる必要が無いため、変更していない。

## (3) AT-J関連サブルーチンのAE化

AT-Jはアセンブリで記述されており、これをそのままアセンブルすると、RMODE=24, AMODE=24の基本域動作モードになってしまう。拡張域を使用するためには、RMODE=ANY, AMODE=31の拡張域動作モードにする必要がある。実際の作業としては、そのアセンブリソースにRMODE命令及び、AMODE命令をそれぞれ挿入した。

## (4) AE化の効果と考察

BLOCKDATAと各サブルーチン内のDATA文を変更し、ANALYSIS内で作業用に使用している各配

列の大きさを試験的に2～3倍にした時のプログラムのサイズは以下ようになった。

ORIGINAL NOAE翻訳版 : 基本域 2.52MB, 拡張域 ----MB  
 AE化 AE翻訳版 : 基本域 0.44MB, 拡張域 2.16MB

現在一般公開しているANALYSIS/EXは、約18万行のプログラムの解析を可能としているが、基本域で動作可能であるためNOAE翻訳である。しかし、将来的に更に大規模プログラムの解析を行う場合でも、作業用配列の大きさを拡大し、AE翻訳することにより解析可能となる。

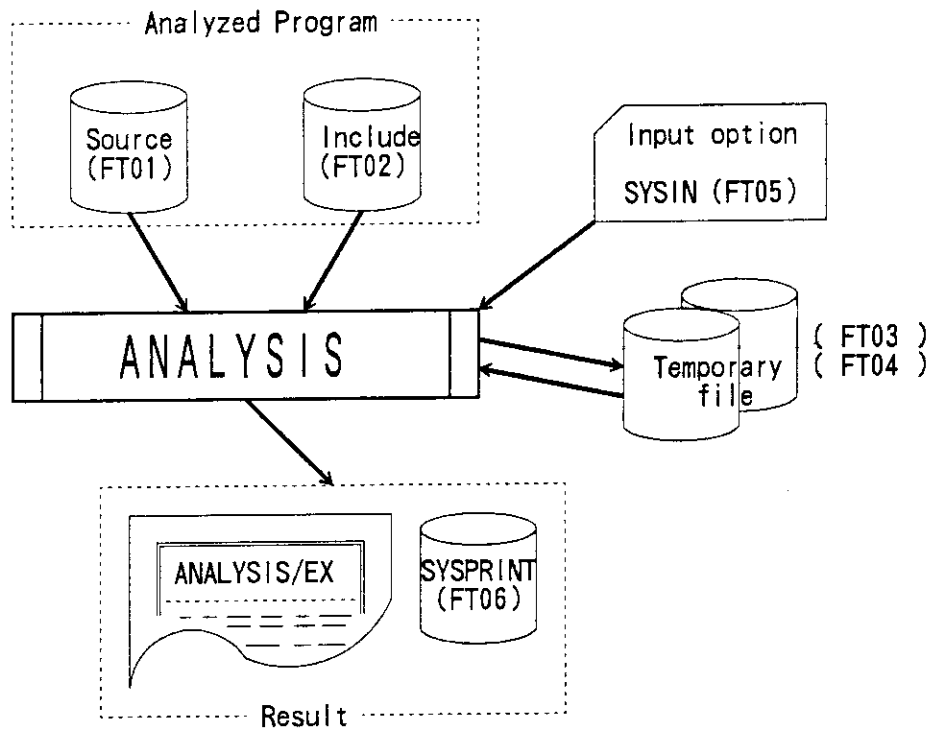


Fig. 2.1 Job flow chart of ANALYSIS

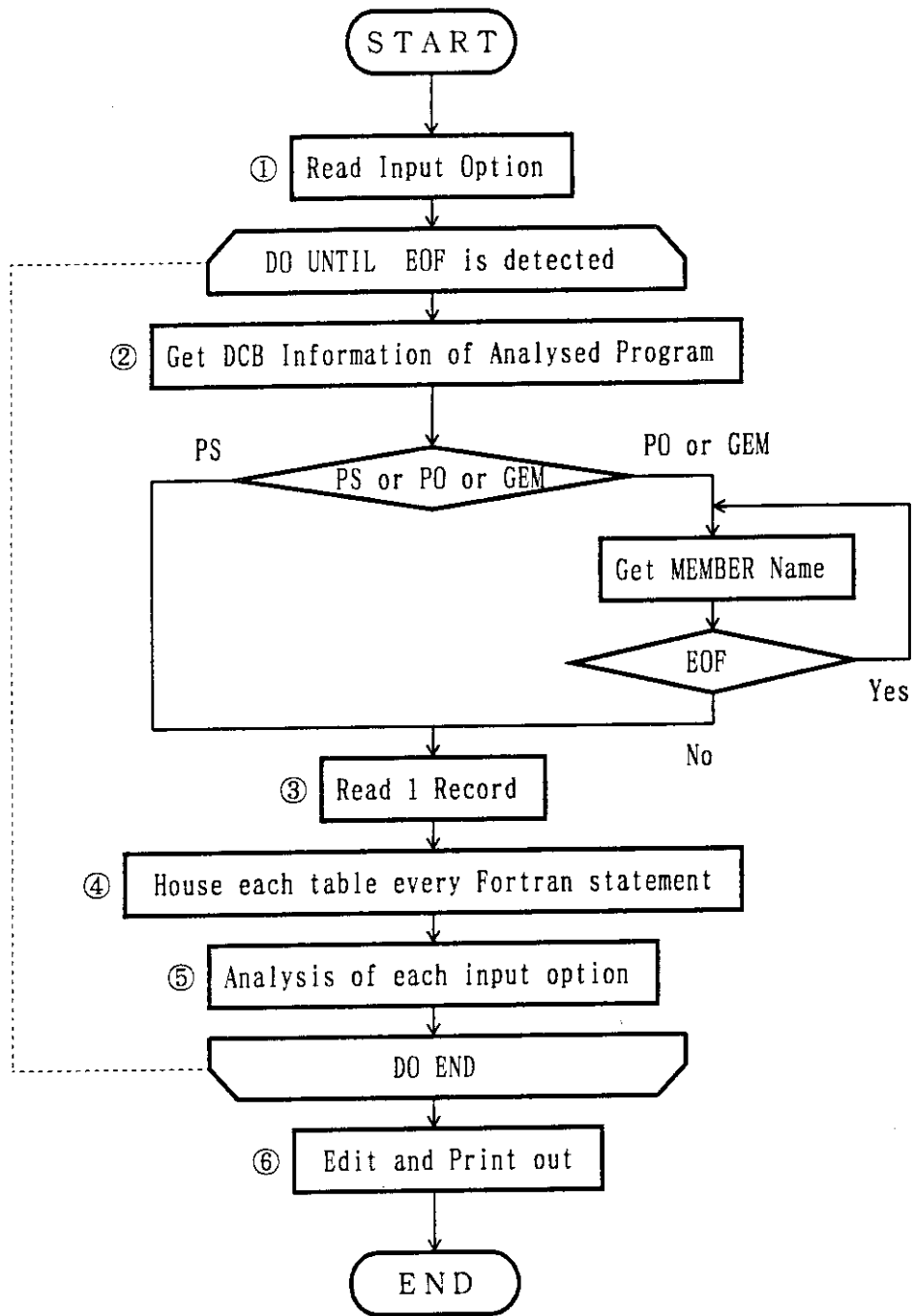
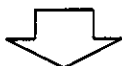


Fig. 2.2 Program flow chart of ANALYSIS

```

SUBROUTINE ANAL77( )
*INCLUDE DIMENS, INSOURCE
*INCLUDE BUFFER, INSOURCE
*INCLUDE SETUP, INSOURCE
*INCLUDE MIX, INSOURCE
*INCLUDE PO, INSOURCE

```



```

SUBROUTINE ANAL77( )
*INCLUDE DIMENS
*INCLUDE BUFFER
*INCLUDE SETUP
*INCLUDE MIX
*INCLUDE PO

```

Fig. 2.3 Deletion of options in \*INCLUDE statement

```

SUBROUTINE READCD( RECORD, NREC )
C
PARAMETER ( NLMAX=80 )
CHARACTER*72 RECORD(NLMAX), REC, OPTCHK
C
C----- READ INPUT CARD ( FT05F001 : SYSIN ) -----
NREC = 0
100 CONTINUE
READ( 5, '(A72)', END=200 ) REC
NREC = NREC + 1
C----- FOR CL(1) OPTION -----
IF( REC(1:2) .EQ. 'C=' ) THEN
RECORD(NREC) = REC
GOTO 100
ENDIF
RECORD(NREC) = OPTCHK( REC, 72 )
C----- FOR MA(1) OPTION -----
C IF( RECORD(NREC)(1:2) .EQ. 'M=' )
C & RECORD(NREC) = REC
C----- DELETE COMMENT LINE -----
IF( RECORD(NREC)(1:1) .EQ. '*' .OR. RECORD(NREC) .EQ. ' ' )
& NREC = NREC - 1
GOTO 100
200 CONTINUE
.....
RETURN
END

```

Fig. 2.4 Outline of subroutine READCD



```

SUBROUTINE SETPRM
.....
CALL FOPSET(5 , IWOPT , CARD , IRT )
IF ( IRT .GE. 8 ) THEN
  WRITE(*,*) ' **** ERROR **** SETPRM INVALID INPUT CARD '
  STOP
.....

REWIND 5

.....

DO 4990 WHILE (.TRUE.)
  READ(5,'(A72)',END=9999) CARD
  I = I+1
  WRITE(6,'(10X, I2, 2X, A72)') I, CARD
4990 CONTINUE
.....

```



```

SUBROUTINE SETPRM
.....
C ADD >>
C----- READ INPUT CARD (FT05F001) -----
          CALL READCD( RECORD, NCARD )
C ADD <<
          CALL FOPSET( IWOPT , CARD , IRT, RECORD, NCARD )
          IF ( IRT .GE. 8 ) THEN
            WRITE(*,*) ' **** ERROR **** SETPRM INVALID INPUT CARD '
            STOP
          .....

C      REWIND 5
          .....

C ADD >>
          ICLFLG = 0
          DO 500 II = 1, NCARD
            .....

          CARD = CARDRC(II)
            .....

C@      DO 4990 WHILE (.TRUE.)
C@      READ(5,'(A72)',END=9999) CARD
C@      I = I+1
          WRITE(6,'(10X, I2, 2X, A72)') I, CARD
C@4990 CONTINUE
          .....

500 CONTINUE
          .....

```

Fig. 2.5 Modification of subroutine SETPRM

```

BLOCK DATA
IMPLICIT INTEGER*4(A-Z)
*INCLUDE DIMENS
*INCLUDE BUFFER
*INCLUDE SETUP
  DATA CLASS1/70*0/, CLASS2/70*0/
C+++++ CHANGE NEDAC 1989 12.25 ++++++
C  DATA IDCVR¥ / 0 ,MXCVR¥*0 /
  DATA CVREF¥ / MXCVR¥*0 ,MXCVR¥*0, MXCVR¥*0, MXCVR¥*0/
  DATA NCV¥ / 0 /
C+++++ END ++++++
END

```



```

SUBROUTINE BLKDAT
IMPLICIT INTEGER*4(A-Z)
*INCLUDE PARAMT
*INCLUDE DIMENS
*INCLUDE BUFFER
*INCLUDE SETUP
  DO 10 I=-1, 68
    CLASS1(I)=0
    CLASS2(I)=0

10  CONTINUE

  DO 20 J=1, MXCVR¥
    DO 20 I=1, 4
      CVREF¥(I, J)=0
20  CONTINUE
    NCV¥=0

C  RETURN
END

```

Fig. 2.6 Conversion of BLOCKDATA into subrouitne BLKDAT

```

PROGRAM MAIN
C ADDITION >>
  CALL INITS
  CALL BLKDAT
C ADDITION <<
  CALL ANAL77
END

```

Fig. 2.7 Calling sequence of subroutine BLKDAT in main program

```

SUBROUTINE INITS
C *****
C * COMMON INITIALIZE ROUTINE
C *****
*INCLUDE PARAM
C
C FOR /ALIAS /
  PARAMETER ( NL01=0, NL11=NALIAS )
  COMMON /ALIAS / L01( NL01 : NL11 )
C FOR /BUFF00/
  PARAMETER ( NC01=NSTRNG+NBF*80+2*8 )
  COMMON /BUFF00/ C01( NC01 )
C FOR /CMREF¥/
  PARAMETER ( NI01=4*MXC¥+1+IDXMAX )
  COMMON /CMREF¥/ I01( NI01 )
C FOR /COMCHA/
  PARAMETER ( NC02=EXTMAX*6+COMMAX*6+2*CRFMAX*6+CVMAX*3+2*CVMAX*6
&           +CLMAX*6+2*MYMAX*6+SUBMAX*6+CALMAX*6+ROTMAX*6
&           +VARMAX*6+2*ENTMAX*6+ARGMAX*6+ARGMAX*3+ERMAX*6
&           +SYSMAX*6+CSMAX*1+IDXMAX*6+2*MAXARG*6+NCMB*80 )
  COMMON /COMCHA/ C02( NC02 )
  .....

  LOGICAL *4      L01, L02
  CHARACTER*1     C01, C02, C03, C04, C05, C06, C07, C08, C09, C10
C
C ***** INITIALIZE OF EACH COMMON *****
  CALL INIT01( I01, 1, NI01 )
  CALL INIT01( I02, 1, NI02 )
  .....

  CALL INIT02( R01, 1, NR01 )
  CALL INIT02( R02, 1, NR02 )
  .....

  CALL INIT03( C01, 1, NC01 )
  CALL INIT03( C02, 1, NC02 )
  .....

  CALL INIT04( L01, NL01, NL11 )
  CALL INIT04( L01, 1, NLO2 )
C
  RETURN
  END

```

Fig. 2.8 Initialization of subroutine INITS

```

SUBROUTINE INIT01( INT, N1, NN )
INTEGER*4  INT(*)
REAL*4    REL(*)
CHARACTER*1 CHA(*)
LOGICAL*4  LOG(*)
C =====
DO 101 I = N1, NN
  INT(I) = 0
101 CONTINUE
RETURN
C =====
ENTRY INIT02( REL, N1, NN )
DO 102 I = N1, NN
  REL(I) = 0.0
102 CONTINUE
RETURN
C =====
ENTRY INIT03( CHA, N1, NN )
DO 103 I = N1, NN
  CHA(I) = ' '
103 CONTINUE
RETURN
C =====
ENTRY INIT04( LOG, N1, NN )
DO 104 I = N1, NN
  LOG(I) = .FALSE.
104 CONTINUE
RETURN
C =====
END

```

Fig. 2.9 Initialization of variables

## 3. 解析機能の拡張

ANALYSIS/EXは従来版の機能に加え、以下に示す機能拡張を行った。

- ① 被解析プログラムの文法レベルをFORTRAN77EX対応にした。
- ② 18万行までの巨大プログラム解析可能にした。
- ③ COMMON関連の詳細解析機能を追加した。
- ④ インクルード文解析機能を追加した。
- ⑤ 作成途中のプログラム解析機能を追加した。

本章では、上記の機能拡張に関する概要について説明する。

### 3.1 FORTRAN77EXでの機能強化項目

FORTRAN77EXでの文法上の機能強化項目をTable 3.1に示す。詳細は富士通マニュアルFORTRAN77EX使用手引書<sup>(7)</sup>を参照されたい。ここで注意を必要とするのは、英小文字名標の扱い、プログラム名標の長さ、7文字を越えるCOMMONラベル名を使用する場合の動作である。以下にその概要を示す。

#### 3.1.1 英小文字名標の扱い

FORTRAN77EXでは名標に英小文字が使用できるようになった。しかし、英大文字名標と英小文字名標を完全に区別して扱うということではない。綴りが同じであれば同一変数として扱われる。例えば変数WORDの扱いは以下ようになる。

$$\text{WORD} = \text{WORD} = \text{WORD} = \text{Word} = \dots\dots = \text{word}$$

#### 3.1.2 プログラム名標の長さ

FORTRAN77EXではプログラム名標を最大31文字まで使用可能となり、実際にコンパイラはプログラム名標を格納する領域を31文字分用意している。しかし、その31文字分全てをプログラム名として認識するというわけではなく、通常は31文字の先頭7文字までしか認識しない。したがって、先頭7文字までが同一名であれば、同一のルーチンとして扱われる。

またFORTRAN77EXからは、プログラム名標の扱いに関して、翻訳オプションCOMPATが設けられた。これを指定することにより、プログラム名標の先頭4文字と末尾3文字の計7文字を認識させることが可能である。

以上から、任意の複数のプログラム名の先頭7文字と末尾3文字が同一の場合は、無条件に

同一プログラム名として扱われるため、実行時に実際に呼ばれるのは、リンク時に最初に現れた方となる。Fig. 3.1にそのプログラム例と実行結果を示す。

### 3.1.3 7文字を越えるCOMMONラベル名を使用する場合の動作

FORTRAN77EXから、COMMONラベル名の長さの制限は7文字まで拡張されたが、Fig. 3.2に示すように、8文字以上の場合でも、同一ラベルであれば、FORTRAN77の文法通り実行される。しかし、7文字までが同一で8文字以降が異なる場合は、Fig. 3.3に示すように、先に宣言した内容が後に宣言した内容に上書きされ、FORTRAN77の文法通りには実行されない。例えば、以下の2つの例を考えてみる。

#### (1) あるルーチン内のCOMMONラベルが同一の場合

プログラム上		メモリ上
COMMON /COM00001111/A		COMMON SECTION /COM0000/ の中身は、
COMMON /COM00001111/B	⇒	A, B, C と連結され、順番に並ぶ。
COMMON /COM00001111/C		

#### (2) あるルーチン内のCOMMONラベルが異なる場合

プログラム上		メモリ上
COMMON /COM0000AAA/A		COMMON SECTION /COM0000/ の中身は、
COMMON /COM0000BBB/B	⇒	Cのみである。
COMMON /COM0000CCC/C		A, Bの値を引用することはできない。

## 3.2 FORTRAN77EX文法対応

3.1節に示した機能強化項目を考慮し、ANALYSISをFORTRAN77EX文法で記述されたプログラムも解析できるように解析機能を拡張した。

### 3.2.1 31文字名標対応

FORTRAN77EXでは、Table 3.1で示されるように、COMMONラベル以外の名標は、31文字まで使用可能である。そこで、ANALYSIS内部での各名標格納のための配列やテンポラリ変数等の宣言を、6バイト文字型から31バイト文字型へ変更した。また、COMMONラベルに関しても3.1.3節に示したように、同一ラベル名の場合はFORTRAN77文法通りに動作するので、便宜上31文字の文字型へ格納した。

なお、解析結果の出力に関しては、COMMONラベルとプログラム名は7文字まで、変数及び配列名は10文字までに制限した。出力フォーマット変更例をFig. 3.4に示す。Fig. 3.4で変数

PRGNAMはプログラム名を格納した配列であり、変数COMNAMはコモンラベルを格納した配列である。変更したルーチンは以下の通りである。

¥CVLCT,	¥DIME,	¥IMPLI,	¥NMSET,
¥NMST2,	¥PRMCT,	¥RECK2,	¥REFCK,
¥SETCV,	¥SRTCL,	¥TSGRT,	¥TYSTM,
¥LPCL,	¥LPCM,	ARGCHK,	ARGLST,
ASSIGN,	CALL,	COMCHK,	COMTAB,
CONTRL,	DOCHK,	ENTRY,	EXTCHK,
FRTLIB,	GENTAB,	GETNAM,	IFCHK,
IOCHK,	LOOKUP,	LPARG,	LPCALL,
LPCHAR,	LPCOME,	LPCOMM,	LPCOM2,
LPEDIT,	LPNEST,	LPTREE,	MOVEX6,
OUTREF,	REFCHK,	SUBNAM,	SUBPRO,
TABLE,	TABLEI,	TREE	

### 3.2.2 英小文字及びアンダースコア付き名標対応

ANALYSISでは、各々の名標を判断し専用配列に格納する際に、富士通FORTRANの文法で名標に許される文字（英大文字、数字、¥）かどうかをINDEX関数を利用して調べている。ANALYSIS/EXでは、そのチェック部分に英小文字とアンダースコア『\_』を加え、文字型変数ALPBETとALPNUMへ格納した。変更したルーチンを以下に示す。また、変更例をFig. 3.5へ示す。

¥PERAT,	ARGCHK,	MATCH,	TOKEN
---------	---------	--------	-------

### 3.2.3 名標以外の英小文字対応

FORTRAN77EXで英小文字の使用が許されたのはユーザが定義する名標だけではない。FORTRAN命令やIMPLICIT関数など（予約語という）への使用も許された。実際のプログラム上では、英大文字と英小文字がどのような組み合わせで与えられてもよい。ところが、ANALYSISが解析する場合に、これらの組み合わせを全て網羅するわけにはいかない。

例えば、複数の変数や配列の各要素の領域合わせを宣言する『EQUIVALENCE』は、

EQUIVALENCE, EQUIVALENce, EQUIVALENce, EQUIVALEnce, ....., equivalence

と2<sup>11</sup>個の可能性があるため、これらひとつひとつを比較の対象にするわけにはいかない。このようなFORTRANの予約語をあらかじめ英大文字で配列に格納しておき、比較対象文字列を全て英大文字に変換してから比較するようにした。

そこで、英小文字を英大文字に変換するファンクションCAPTLFとサブルーチンCAPTLSの2種類を作成した。これら変換ルーチンを組み込んだルーチンを以下に示す。また、CAPTLFをFig. 3.6に、CAPTLSをFig. 3.7に示す。

変更ルーチン			
¥LSPCY,	CALL,	CHKNUM,	DCLARE,
FRTLIB,	INCMEM,	INCSF,	KEYWRD,
LPTREE,	MATCH,	NEST,	NUMBER,
POSF77,	READSF,	SUBNAM,	TYPDCL

EBCDICコード表内での英大文字ABCDE...Z, 英小文字abcde...zは、コードが連続しており、かつ、英大文字は英小文字に対して16進数表現で64だけ足した位置にある<sup>(7)</sup>。また、文字も数字と同様に大小比較できることを利用して、『a以上z以下の文字』だけを対象に変換することにした。図中の16進数で与えてある『Z40』は『空白』を表すが、変数を初期化している。また『Z81』は英小文字の『a』を、『ZA9』は『z』を表す。

### 3.2.4 インライン・コメントへの対応

FORTRAN77では、1カラム目に『C』か『\*』を記述することにより、その行全体をコメントとして扱うことができる。FORTRAN77EXでは更に、文中にコメントを付けられるようになった。これをインラインコメントといい、エクスクラメーションマーク『!』を任意のカラムに記述すれば、そのカラム以降、第72カラムまでがコメント行として扱われる。そこで、インライン・コメントを判定するサブルーチン EXCOMT を作成した。EXCOMTをFig. 3.8に示す。

コメント文以外に『!』が使われるのは、HOLLERITH型編集、継続行指定文字、クォーテーションとクォーテーションで囲まれた間の文字列内である。したがってEXCOMTでは、これらの場合を判定させ、インライン・コメントの場合にだけ、そのカラム位置を返す機能を持たせた。

ANALYSISでの解析は、コメント行以外のFORTRAN文のみを対象にしている。そのため、コメント行はコメント行用の配列COMBUFに格納しているが、インライン・コメントについても同様にCOMBUF格納するようにした。EXCOMTは、FORTRAN文を1行ずつ仕分けするサブルーチン READSF内に組み込んだ。



### 3.2.5 ダブルクォーテーションへの対応

FORTRAN文中で文字列を囲むには、シングルクォーテーション『』しか使えなかったが、FORTRAN77EXからは、ダブルクォーテーション『"』を使用できるようになった。そのため、従来は、シングルクォーテーションの判定をする場合には、

```
IF( SINGLE .EQ. ''' ) .....
```

としなければならなかったが、ダブルクォーテーションを使うことにより、

```
IF( SINGLE .EQ. "" ) .....
```

とすることも可能である。

ANALYSIS/EX内では、シングルクォーテーションの判定をしているところに、ダブルクォーテーションの判定部を追加した。変更ルーチンを以下に示し、サブルーチンAPOSTR内の判定部ENTRY DQUATEの追加部分をFig. 3.9へ示す。

<pre>APOSTR (DQUATE), CLASS, EXCOMT, SLASLA, TOKEN</pre>
--

### 3.2.6 継続行数の拡張

FORTRAN77EXでは、継続行数は従来の19行から99行まで拡張され、開始行を合わせると100行まで記述可能になった。そこで、ANALYSIS内での各FORTRAN行を格納するバッファの配列数NBFを、Fig. 3.10に示すように100まで拡張した。Fig. 3.10のINCLUDEメンバはPARAMTは、ANALYSIS/EX全体の変数の大きさを制御するパラメータ設定部分である。この部分の詳細は第4章で述べる。変更インクルードファイルは、PARAMT, BUFFERである。

### 3.2.7 英大文字と英小文字の各名標の区別について (DISTオプション)

大型計算機上では、英大文字と英小文字の区別は行われていない。しかし一般に、UNIXマシン上のFORTRANコンパイラでは、翻訳オプションによって、英大文字と英小文字を区別させることが可能である。

ANALYSIS/EXでは、UNIXマシン上で開発されたプログラムに対応するため、英大文字と英小文字の区別をして解析できるオプション『DIST』を追加した。

『DIST』指定した場合と指定しない場合では、例えば、

『DIST』を指定した場合	:	Distinguish ≠ DISTINGUISH
指定しない場合	:	Distinguish = DISTINGUISH

となる。新オプションの認識部分をFig. 3.11に示す。また、認識部分を組み込んだルーチンは以下の通りである。

¥OPSET,	¥OPTV,	¥LSPCY,	CALL,
CHKNUM,	DCLARE,	FRTLIB,	INCMEM,
INCSF,	KEYWRD,	LPTREE,	MATCH,
NEST,	NUMBER,	POSF77,	READSF,
SUBNAM,	TYPDCL		

### 3.3 ANALYSISの解析機能拡張

ANALYSIS-77には14種類の解析オプションが存在したが、COMMON配列を含むFORTRAN文の解析をする場合や、作成途中のプログラムの解析にはあまり有効ではなかった。以下にその理由と、追加した解析機能を示す。

#### 3.3.1 COMMONの解析強化 (CS, CPオプション)

ANALYSIS-77のCOMMON文解析は、『CO』と『VA』オプションによる簡易解析が主である。『CO』は、各ルーチンで任意のCOMMON変数が定義または参照されると、その変数が含まれるCOMMONラベルに相当する場所にFig. 3.12に示すように、『X』で表示し、その次にCOMMON文の一覧表を出力する機能である。『VA』は、COMMON変数とそのルーチン内で定義されているか、下位ルーチンへ引数で渡されているかといった情報を、Fig. 3.13に示すように出力する機能である。したがって、ANALYSIS-77でCOMMON内の個々の変数や配列の詳細情報を得るためには、『CO』と『VA』の結果を見較べて判断しなければならなかった。更に、同一のCOMMONラベルでもルーチンによってCOMMON変数名が異なる場合、同一のCOMMONラベルでもルーチンによってCOMMONブロックの長さが異なる場合、また、COMMONのリンクが行われている場合などは、個々のCOMMON変数の定義参照状況を知るには、プログラムを直接目で追わなければならなかった。

ANALYSIS/EXでは、『CS』と『CP』の2つのオプションを追加した。CSオプションは各ルーチン内での個々のCOMMON変数や配列の定義参照関係表を出力する(Fig. 3.14)。『CP』オプションは『CS』オプションの機能に加え、個々の変数や配列の長さをByte単位で明示しながら出力する機能である(Fig. 3.15)。

解析方法は以下の通りである。

- ① 他のオプションと同様に宣言部を確認した後、その中からCOMMON変数や配列とその長さをCOMMON格納配列に格納する。EQUIVALENCEがあればその情報も取り込む。

- ② 各行のFORTRAN文から変数を読み取り、①のCOMMON格納配列内の変数名とマッチングする。
- ③ マッチングした場合に、代入文内の『=』の前後どちらかにあるか、入出力文に含まれているかなどを調べ、それぞれのフラグを立てる（フラグの具体的な意味は付録A『ANALYSIS/EX V1.2仕様手引書の3.2節参照のこと）。
- ④ ①～③の処理を繰り返し、各COMMON変数と配列の定義参照情報テーブルを完成させる。
- ⑤ プログラムの最終行まで読み終わり、前述のテーブルが完成されたら、出力処理を行う。変更追加ルーチンは以下の通りである。

¥CNTLN,	¥CVLCT	¥DIME,	¥IMPLI,
¥ITEM,	¥LPCL,	¥LPCM,	¥LSPCY,
¥NMSET,	¥NMST2	¥OPSET	¥OPTCK,
¥OPTV,	¥PERAT	¥PRMCT,	¥PRTVR,
¥RECK2,	¥REFCK,	¥SETCV	¥SRTCL,
¥TSGRT,	¥TYSTM		

### 3.3.2 INCLUDE行解析強化

#### (1) INCLUDE文解析対応

従来のANALYSISは、\*INCLUDE行は認識できたが、INCLUDE文は認識できなかった。INCLUDE文はFORTRANの標準の文法なので、ANALYSIS/EXにINCLUDE文を認識する機能を加えた。

FORTRANでは一般に、\*INCLUDE行などの一部の命令を除いて、命令文や名標の間に空白が入っても翻訳に問題は無い。そこで、『INCLUDE(』の文字列を探し出すために、文字間の空白を詰めるファンクションCHCHKを作成し、INCLUDEを判定しているサブルーチンREADSFとSETPRMへ組み込んだ。CHCHKをFig. 3.16へ、組み込み部分の例をFig. 3.17へ示す。

CHCHK内では、文字列格納変数VALUE内の文字列の長さを調べ、VALUE内に空白が含まれるか否かを判断しながら、CHCHKに格納している。

#### (2) NOINCオプションについて

ANALYSIS-77ではソースプログラムを読み込んでいくとき、『\*INCLUDE』『INCLUDE(』の文字列を見つけると、次に続く名標がINCLUDEファイルのメンバ名であると認識し、そのメンバをオープンする。もしそのメンバが存在すれば、解析のために読み込まれ、無ければオープンエラーのメッセージを出力して、途中で解析が終了する。このため、作成途中のプログラムの木構造図や、IF文やDOループのネスト構造を知りたい場合には、INCLUDE呼び出し部分をコメントアウトしてから、ANALYSISによる解析をしなければならなかった。

このため、ANALYSIS/EXでは、\*INCLUDE行やINCLUDE文を無視する『NOINC』オプションを加えた。この機能では、INCLUDEファイルを読むのは、Fig. 3.17に示すように、

「OPTION(26)=0の時」で、かつ「INCLUDEが検出された時」

に限定されている。『NOINC』オプションを指定しない場合にはOPTIONに0が設定される。変更ルーチンはサブルーチンINCMEMとREADSFである。また、サブルーチンINCMEMには、Fig. 3.18に示すようなINCLUDE文検出時のメンバ名抽出機能を加えた。

### 3.3.3 巨大プログラム解析対応

原子力コードには、

- ① プログラム行数が10万行を越えるようなコード、
- ② COMMON変数やCOMMON配列が4万個以上もあるコード、
- ③ CALL文が5000個以上あるコード、
- ④ ENTRY副プログラムが200個以上もあるコード、

などのような大規模プログラムが存在する。

ANALYSIS/EXでは、これらの解析を行えるように、FORTRANの各ステートメントに対応した名標を格納する作業配列の大きさを大きくした。この配列の大きさは、INCLUDE内のPARAMETER文で指定することにより一括して行えるようにした。また、COMMON変数の個数は、ANALYSIS-77では2バイト整数型でカウントされていたため、32767個しか解析できなかったが、これを4バイト整数型で宣言することにより解析できる変数の個数が大幅に増加した。

個々のパラメータはFig. 3.10に示すPARAMETER文の一覧に示してある。拡張したパラメータを含むINCLUDEメンバは以下の通りである。

ANAMDF,	BYTE,	DIMENS,	PARAMT,
PO,	WORK		

### 3.3.4 エラーメッセージの追加

#### (1) FORTRANで許されない文字検出時のメッセージ

FORTRANで許されない文字とは、NULL文字、制御記号、演算子以外の記号、名標に許されない記号などのことである。外国で開発されたプログラムを、原研の大型計算機で動作するように変換するときなどに、外国のFORTRANとJIS-FORTRANの規定の違いから、許されない文字が検出される場合がある。そのため、ANALYSISでは各々の名標を分類するときに、その一つ一つの文字がFORTRANで許される文字か否かの確認も同時に行っている。

許されない文字が検出された場合は、その旨のメッセージを出力し、更にその文字を文字型(A形式)と、16進数型(Z8形式)の両方の形式で表示するようにサブルーチンTOKENを変更した。また、その判定部をFig. 3.19に、NULL文字(HEX:00)検出の際のメッセージ例をFig. 3.20に示す。

## (2) ファイルオープン時のメッセージ

ANALYSISが被解析プログラムを読み込むときは、AT-JのサブルーチンFREADを用いている。もし、INCLUDE文などで指定されたメンバがオープンできないなどの障害が生じた場合には、リターンコードIOCONDに1を返す。この場合はオープンエラーとエラーを起こしたメンバ名を表示するようにした。エラーメッセージはサブルーチンINCSF, POSF77, READSFに組み込んだ。また、その判定部をFig. 3.21に、メッセージ例をFig. 3.22に示す。

Table 3.1 The comparison of grammar level between FORTRAN77 and FORTRAN77EX

文法上のサポート機能	FORTRAN77(V10)	FORTRAN77EX(V12)
名標の長さ 変数, 配列名 COMMONラベル名 プログラム名	6文字以内 6文字以内 6文字以内	31文字以内 (日本語は15文字以内) 7文字以内 31文字以内 (通常は先頭から7文字以内*)
日本語名標の使用	不可	可
英小文字の使用 FORTRAN命令 名標	不可 不可	可 可
名標中のアンダースコアの使用	不可	可
引数の個数	255	1000
文字定数の長さ	255	32767
一文の最大継続行数	19	99
8Byte整数型変数の使用	不可	可
インラインコメントの使用	不可	可
ダブルクォーテーションの使用	不可	可
インクルード行のオプション使用	可	不可(オプションを指定しても無視)

\* FORTRAN77EXでは, 翻訳オプションのCOMPATを指定した場合, 8文字以上の名標は前4文字と末尾3文字の計7文字で認識される.

```

PROGRAM MAIN
CHARACTER*8 CHAR1, CHAR2
CALL SUB00001AAA(CHAR1)
CALL SUB00002AAA(CHAR2)
WRITE(*,*) 'CHAR1 = ', CHAR1
WRITE(*,*) 'CHAR2 = ', CHAR2
ENDS

```

```

SUBROUTINE SUB00001AAA(CHAR1)
CHARACTER*8 CHAR1
CHAR1 = 'SUB00001'
RETURN
END

```

```

SUBROUTINE SUB00002AAA(CHAR2)
CHARACTER*8 CHAR2
CHAR2 = 'SUB00002'
RETURN
END

```



EXECTION RESULT

```

CHAR1 = SUB00001
CHAR2 = SUB00001

```

Fig. 3.1 Example of linkage mechanism of subroutines

```

C      PROGRAM MAIN
          ← 8 →
COMMON /COM00001/ MEM1
COMMON /COM00001/ MEM2
MEM1 = 1
MEM2 = 2
WRITE(*,*) 'MAIN(1):', MEM1, MEM2
CALL SUB0001
CALL SUB0002
WRITE(*,*) 'MAIN(2):', MEM1, MEM2

SUBROUTINE SUB0001
COMMON /COM00001/ MEM1
COMMON /COM00001/ MEM2
WRITE(*,*) 'SUB0001:', MEM1, MEM2
END

SUBROUTINE SUB0002
COMMON /COM00001/ MEM1, MEM2
WRITE(*,*) 'SUB0002:', MEM1, MEM2
END

```



## EXECUTION RESULT

```

MAIN(1):1 2
SUB0001:1 2
SUB0002:1 2
MAIN(2):1 2

```

Fig. 3.2 Example of linkage mechanism of common variables in the case that the common label names are same.



```

C      PROGRAM MAIN
          ← 8 →
COMMON /COM00001/ MEM1
COMMON /COM00002/ MEM2
MEM1 = 1
MEM2 = 2
WRITE(*,*) 'MAIN(1):', MEM1, MEM2
CALL SUB0001
CALL SUB0002
WRITE(*,*) 'MAIN(2):', MEM1, MEM2

SUBROUTINE SUB0001
COMMON /COM00001/ MEM1
COMMON /COM00002/ MEM2
WRITE(*,*) 'SUB0001:', MEM1, MEM2
END

SUBROUTINE SUB0002
COMMON /COM00001/ MEM1, MEM2
WRITE(*,*) 'SUB0002:', MEM1, MEM2
END

```



## EXECTION RESULT

```

MAIN(1):1 2
SUB0001:2 2
SUB0002:1 0
MAIN(2):2 2

```

Fig. 3.3 Example of linkage mechanism of common variables in the case that letters after the 7th letter in common label name are different.

```

WRITE(6,6808) '+', ('-', J=1, 14), '+', ('-', J=1, IE-IS+1), '+ '
WRITE(6,6800) 'I*****', (PRGNAM(J)(1:1), J=IS, IE), 'I'
WRITE(6,6800) 'I*****', (PRGNAM(J)(2:2), J=IS, IE), 'I'
WRITE(6,6800) 'I*****', (PRGNAM(J)(3:3), J=IS, IE), 'I'
WRITE(6,6800) 'I*****', (PRGNAM(J)(4:4), J=IS, IE), 'I'
WRITE(6,6800) 'I*****', (PRGNAM(J)(5:5), J=IS, IE), 'I'
WRITE(6,6800) 'I*****', (PRGNAM(J)(6:6), J=IS, IE), 'I'
WRITE(6,6808) '+', ('-', J=1, 14), '+', ('-', J=1, IE-IS+1), '+ '
ENDIF
CALL PAGE(1)
WRITE(6,6804) M, COMNAM(COM), (LINE(J:J), J=IS, IE), 'I'
.....

```

```
6800 FORMAT(1X, A, 'I', 20(5A1, :, '.'))
.....
```

```
6804 FORMAT(1X, 'I', 14, './', A6, './', 1X, 'I', 20(5A1, :, '.'))
```



```

WRITE(6,6808) '+', ('-', J=1, 16), '--', ('--', J=1, IE-IS+1), '+ '
WRITE(6,6800) '|_ PROGRAM ', (PRGNAM(J)(1:1), J=IS, IE), '| '
WRITE(6,6800) '|_ NAME ', (PRGNAM(J)(2:2), J=IS, IE), '| '
WRITE(6,6800) '|_ ', (PRGNAM(J)(3:3), J=IS, IE), '| '
WRITE(6,6800) '|_ ', (PRGNAM(J)(4:4), J=IS, IE), '| '
WRITE(6,6800) '|_ ', (PRGNAM(J)(5:5), J=IS, IE), '| '
WRITE(6,6800) '|_ COMMON _', (PRGNAM(J)(6:6), J=IS, IE), '| '
WRITE(6,6800) '|_ LABEL _', (PRGNAM(J)(7:7), J=IS, IE), '| '
WRITE(6,6809) '|', ('=', J=1, 16), '|=', ('=', J=1, IE-IS+1), '=| '
WRITE(6,6804) M, COMNAM(COM), (LINE(J:J), J=IS, IE), '| '
.....

```

```
6800 FORMAT(1X, A, '|', 20(5A2, :, '|'))
.....
```

```
6804 FORMAT(1X, '|', 14, './', A7, './', 2X, '|', 20(5A2, :, '|'))
```

Fig. 3.4 Modification of output format of symbol names, or program names, variables, and so on

```

SUBROUTINE TOKEN(STRING, ENDSTR, START, CODE, CODMAX, CODLEN)
.....

CHARACTER*1  STRING(ENDSTR), LETTER, LET
.....

C...FETCH ONE CHARACTER
    LETTER=STRING(LCV)
C..IF BLANK CHARACTER ,FETCH THE NEXT CHARACTER
    IF(LETTER.EQ.' ') THEN
        BLANK=. TRUE.
C=====
C
C                ALPHABET CHARACTER
C
    ELSE IF(INDEX(' ABCDEFGHIJKLMNOPQRSTUVWXYZΨ', LETTER).NE. 0) THEN
        STRFST=LCV
.....

```



```

SUBROUTINE TOKEN(STRING, ENDSTR, START, CODE, CODMAX, CODLEN)
.....

CHARACTER*1  STRING(ENDSTR), LETTER, LET
C ADD >>
CHARACTER*54 ALPBET
CHARACTER*65 ALPNUM
DATA ALPBET
& /' ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyzΨ_ '/
DATA ALPNUM(:55)
& /' ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyzΨ_ '/
DATA ALPNUM(56:)
& /' 0123456789' /
C ADD <<
.....

C...FETCH ONE CHARACTER
    LETTER=STRING(LCV)
C..IF BLANK CHARACTER ,FETCH THE NEXT CHARACTERBL
    IF(LETTER.EQ.' ') THEN
        BLANK=. TRUE.
C=====
C
C                ALPHABET CHARACTER
C
    ELSE IF(INDEX( ALPBET, LETTER ) .NE. 0 ) THEN
        STRFST=LCV

```

Fig. 3.5 Example of modification in the letter matching procedure

```

CHARACTER*(*) FUNCTION CAPTLF(SMALL)
INTEGER * 4 F/Z40/, ZZ/Z81/, ZZZ/ZA9/
CHARACTER*(*) SMALL
CAPTLF=' '
DO 100 I=1,LEN(SMALL)
  F=ICHAR(SMALL(I:I))
  IF( F.LE.ZZZ .AND. F.GE.ZZ ) THEN
    CAPTLF(I:I)=CHAR(F+64)
  ELSE
    CAPTLF(I:I)=SMALL(I:I)
  ENDIF
100 CONTINUE
RETURN
END

```

Fig. 3.6 Conversion function program of letters into capital ones

```

SUBROUTINE CAPTLS(SMALL, LARGE, LENG)
INTEGER *4 F/Z40/, ZZ/Z81/, ZZZ/ZA9/
CHARACTER*1 SMALL(LENG), LARGE(LENG)
DO 100 I=1,LENG
  LARGE(I)=' '
  F=ICHAR(SMALL(I))
  IF( F.LE.ZZZ .AND. F.GE.ZZ ) THEN
    LARGE(I)=CHAR(F+64)
  ELSE
    LARGE(I)=SMALL(I)
  ENDIF
100 CONTINUE
RETURN
END

```

Fig. 3.7 Conversion subroutine program of letters into capital ones

```

SUBROUTINE EXCOMT(BUFFER, IEXCL)
CHARACTER BUFFER*80, HNUM*30, SPC30*30, NUM*10/' 1234567890' /, FDAT*1
.....

C ***** INTIALIZE *****
.....

C ***** BUFFER CHECK START *****
DO 1000 WHILE( I.LE.72 )
  IF( I.EQ.6 ) I=I+1
  IDATYP = INDEX( NUM, BUFFER(I:I) )
  IF(IDATYP.EQ.0) FDAT=BUFFER(I:I)
C ----- BUFFER IS NUMERICAL DATA -----
  IF( IDATYP.NE.0 ) THEN
    DO 90 WHILE( .TRUE. )
      IF( I.EQ.6 ) I=I+1
      .....

    90 CONTINUE
  ENDIF
C ----- BUFFER IS '??H.....' OR NOT -----
  91 CONTINUE
  IF(( BUFFER(I:I).EQ.'H' ) .AND.
    & ( FDAT.EQ.'=' .OR. FDAT.EQ.',' .OR.
    & ( FDAT.EQ.'/' .OR. FDAT.EQ.')' ) ) THEN
    .....

    IF( I.GT.72 ) RETURN
  ENDIF
C ----- BUFFER IS " " " -----
  IF( BUFFER(I:I).EQ.'"' .AND. ISFLG.EQ.0 ) THEN
    DO 100 WHILE( .TRUE. )
      IF( I.EQ.6 ) I=I+1
      .....

      I=I+1
    100 CONTINUE
  ENDIF
C ----- BUFFER IS ' ' ' -----
  IF( BUFFER(I:I).EQ.'"' .AND. IWFLG.EQ.0 ) THEN
    DO 110 WHILE( .TRUE. )
      IF( I.EQ.6 ) I=I+1
      .....

      I=I+1
    110 CONTINUE
  ENDIF
C ----- BUFFER IS '! ' -----
  IF( BUFFER(I:I).EQ.'!' .AND. IEXCL.EQ.0 ) THEN
    IEXCL=I
    .....

  ENDIF
1000 CONTINUE
C
  RETURN
END

```

Fig. 3.8 Detection procedure of inline comment line

```

SUBROUTINE APOSTR(STRING, ENDSTR, POSIT)
.....

CHARACTER*1  STRING(ENDSTR), APOS
DATA        APOS /'"/

.....

DO 100 WHILE (LCV, LE. ENDSTR)
  IF(STRING(LCV). NE. APOS) THEN
    LCV=LCV+1
  ELSE IF((STRING(LCV+1). NE. APOS) .OR. (LCV+1. GT. ENDSTR)) THEN
    .....

    END IF
100 CONTINUE
C
END

```



```

SUBROUTINE APOSTR(STRING, ENDSTR, POSIT)
.....

CHARACTER*1  STRING(ENDSTR), APOS, DQUA
.....

DO 100 WHILE (LCV, LE. ENDSTR)
  IF(STRING(LCV). NE. APOS) THEN
    LCV=LCV+1
  ELSE IF((STRING(LCV+1). NE. APOS) .OR. (LCV+1. GT. ENDSTR)) THEN
    .....

    END IF
100 CONTINUE
.....

RETURN

ENTRY DQUATE(STRING, ENDSTR, POSIT)
DATA        DQUA /'"/

.....

DO 200 WHILE (LCV, LE. ENDSTR)
  IF(STRING(LCV). NE. APOS) THEN
    LCV=LCV+1
  ELSE IF((STRING(LCV+1). NE. APOS) .OR. (LCV+1. GT. ENDSTR)) THEN
    .....

    END IF
200 CONTINUE
RETURN
END

```

Fig. 3.9 Example of detection procedure of double quotation symbol

```

C PARAMT *** PARAMETER FOR SCDAP, SRAC, DREAM.... ***
C
C (ANAMDF) FOR COMMON /MDFNM%/ ,/MDFVR%/ ,/MDFLN%/
  PARAMETER ( MXPR% = 200, MXVN% = 300, MXEX% = 20 )
  PARAMETER ( MXNC1% = 500, MXNC2% = 5000, MXLC% = 100 )
  PARAMETER ( MXNU% = 200, MXDM% = 300 )
C
C (PARM)
  PARAMETER ( MAXINT = 31, MAXFLD = MXPR%, MAXREC = 2000,
1      MAXCMD = 50, MAXPRM = 50, MAXTBL = 120,
2      MAXANL = 300, MAXANS = 1, LINESZ = 80,
3      MAXSTR = 240, MAXOSP = 10, MAXPLT = 10,
4      LCMD = 8, LPRM = 4, LFNM = 33,
5      LINT = 31, IO1 = 1, IO2 = 2,
6      IO3 = 3, IO4 = 4, IO5 = 5,
7      IO6 = 6, IO70 = 70, IO71 = 71 )
C
C (BUFFER) FOR COMMON /BUFF00/ ( ,/LENGTH/)
  PARAMETER ( NSTRNG = 6600, NBF = 100 )
C
C (BYTE) FOR COMMON /DCLBYT/ ,/DCLNUM/
  PARAMETER ( CHAMAX = 200, DCLMAX = 200, MAXPAR=20 )
  PARAMETER ( IMPMAX = 28 )
C
C (DIMENS) FOR COMMON /COMCHA/ ,/COMINT/
  PARAMETER ( SUBMAX = 500, CRPMAX = 4000, COMMAX = 500
&      , CLMAX = 100, MAXCOD = 1003, CODMAX = MAXCOD-3
&      , MYMAX = 8000, IOMAX = 7000, CALMAX = 4000
&      , ROTMAX = 8000, VARMAX = 1600, ARGMAX = 1000
&      , CVMAX = 900, ENTMAX = 300, ERMAX = 2000
&      , EXTMAX = 100, SYSMAX = 600, CSMAX = 8000
&      , IDXMAX = 800, MAXARG = 800 )
  PARAMETER ( MXCV% = 6000, NCMB = 4000 )
C
C (MIX) FOR COMMON /MIXCHA/ ,/MIXINT/
  PARAMETER ( NORGIN = 20, NSBSTR = 20, NSUBLE = 20
&      , NTPST = 66, NSTRID = 4000, NKEY = 7000 )
C
C (PO) FOR COMMON /POCHAR/ ,/PONUM /
  PARAMETER ( MAXLH = 20 )
C
C (SETUP) FOR COMMON /SETUP / ,/HEADER/
  PARAMETER ( NCLASS = 68, NOPTMX = 50, NTPMAX = 50 )
C
C (WORK) FOR COMMON /WORKC%/ ,/WORKD%/
  PARAMETER ( MXTK% = 2000, MXIM% = 500 )
C
C ( ) FOR OTHER COMMON /ALIAS / ,/SAMEXX/....
  PARAMETER ( NALIAS = 11, NSAMEX = 400 )

```

Fig. 3.10 Assignment of array variable size

```

INTEGER FUNCTION ¥OPTCK(N ,COPT )
.....

CHARACTER * 2 CX
CHARACTER * 3 CE /'ELM' /
CHARACTER * 5 CA /'ASTER' /
CHARACTER * 6 CS /'STATIS' /
CHARACTER * 5 CN /'NOINC' /
CHARACTER * 6 CD /'DIST' /
CHARACTER * (*) COPT

C
  ¥OPTCK = 0
C===== CHECK LETTER
  IF ( N.EQ. 2 ) THEN
    CX = COPT(1:2)
    IF ( CX.EQ. 'SU' ) ¥OPTCK = 1
    .....

    IF ( CX.EQ. 'CS' ) ¥OPTCK = 12
    IF ( CX.EQ. 'CP' ) ¥OPTCK = 13
    IF ( CX.EQ. 'LC' ) ¥OPTCK = 14
    IF ( (CX.NE.'AS').AND.(CX.NE.'ST').AND.(CX.NE.'EL').AND.
&      (CX.NE.'NO').AND.(CX.NE.'DI').AND.(¥OPTCK.EQ.0) ) THEN
      ¥OPTCK = 999
    ENDIF
    GO TO 9999
  ENDIF
C===== STATISTIC OR ASTER OPTION
  IF ( (N.EQ.3) .AND. (CE.EQ.COPT(1:3)) ) THEN
    ¥OPTCK = 20
    GO TO 9999
  ENDIF
  .....

  IF ( (N.EQ.5) .AND. (CN.EQ.COPT(1:5)) ) THEN
    ¥OPTCK = 26
    GO TO 9999
  ENDIF
  IF ( (N.EQ.4) .AND. (CD.EQ.COPT(1:4)) ) THEN
    ¥OPTCK = 27
    GO TO 9999
  ENDIF
C CHANGE >>
  IF ( (N.GE.3) .AND. (COPT(1:N).NE.CE(1:N)) .AND.
&      (COPT(1:N).NE.CA(1:N)) .AND. (COPT(1:N).NE.CS(1:N)) .AND.
&      (COPT(1:N).NE.CN(1:N)) .AND. (COPT(1:N).NE.CD(1:N)) ) THEN
C CHANGE <<
  ¥OPTCK = 999
  GO TO 9999
  ENDIF
C
C>>>> RETURN
C
9999 CONTINUE
  RETURN
  END

```

Fig. 3.11 Detection part of new option DIST, NOINC, CS, CP



```

===== SUBPROGRAM NAME VS. COMMON BLOCK NAME =====
+-----+
I*****I¥¥¥¥¥. ¥¥¥¥¥. ¥¥¥¥¥. ¥¥¥¥¥. ¥¥AAA. AAABB. CCCCC. CCCCD. DDDEE. EEEEE. FFFF
I*****I CCDII. LLLNN. OOO PP. PRRSS. TTNPR. RSSLL. AAAHL. OOOOC. EOOLL. NRRXX. INNR
I*****I INVIMT. PPSMM. PPPER. REBER. GYAOG. GSTKN. LLLKA. MMMNL. CCSMM. TRRTT. LDDT
I*****I TLMPE. CCPSS. STTRM. TCFTT. SSLSC. LINDK. CCLNS. CPTTA. OHT C. RO2CI. ESXL
I*****I LCELM. LMCET. ECVAC. VKCCC. RT7TH. SGUAS. MO US. HUARR. DKN H. YR3HN. TXI
I*****I INT I . YT2. TK TT. R2KVL. TM7RK. TNMTP. NO M . KTBLE. E O K. 2KT. RXB
+-----+
I 1./ALIAS / I . . . . .
I 2./BUFFOO/ I .XX X. . XXX . X X.XX X . X .X XXX. X .XX XX. X
I 3./CMREF¥/ I .XX X. . XXXX. X X.XX X . X .X XXX. X .XX XX. X
I 4./COMCHA/ I .XX X. . XXXX. X X.XX X . X .X XXX. X .XX XX. X
I 5./COMINT/ I .XX X. . XXXX. X X.XX X . X .X XXX. X .XX XX. X
I.....I.....
I 6./CTBL / I . . . X . . . . X . . . .
I 7./DCLBYT/ I .XX X. . X . . . . .X XX . . .
I 8./DCLNUM/ I .XX X. . X . . . . .X XX . . .
I 9./FILD / I . . . X . . . . X . . . .
I 10./HEADER/ I .XX X. . XXX . X X.X X . . XXX. .X . X
I.....I.....
I 11./LENGTH/ I .XX X. . XXX . X X.XX X . X .X XXX. X .XX XX. X
I 12./MDFLN¥/ I XXX .X . XX. X. X . .XX . XX . . .

```

Fig. 3.12 Result of ANALYSIS-77 by CO option

```

(001) ***** SUBROUTINE ¥CNTLN(CREC , NC , ID , NID , IDEF , IANS )
      ** ¥CNTLN **
      *****

      . DUMMY ARGUMENT(CREC ) MAY BE PASSED TO ANOTHER SUBPROGRAM.
      . DUMMY ARGUMENT(IANS ) MAY BE ASSIGNED A VALUE.
      . DUMMY ARGUMENT(ID ) MAY BE PASSED TO ANOTHER SUBPROGRAM.
      . DUMMY ARGUMENT(NC ) MAY BE PASSED TO ANOTHER SUBPROGRAM.
      . COMMON VARIABLE(CITEM¥/WORKC¥/) MAY BE ASSIGNED A VALUE AND PASSE

(002) ***** SUBROUTINE ¥CVLCT(CREC , NC , IDX )
      ** ¥CVLCT **
      *****

      . DUMMY ARGUMENT(CREC ) MAY BE PASSED TO ANOTHER SUBPROGRAM.
      . DUMMY ARGUMENT(IDX ) MAY BE PASSED TO ANOTHER SUBPROGRAM.
      . DUMMY ARGUMENT(NC ) MAY BE PASSED TO ANOTHER SUBPROGRAM.
      . COMMON VARIABLE(ID¥ /WORKD¥/) MAY BE PASSED TO ANOTHER SUBPROGR

```

Fig. 3.13 Result of ANALYSIS-77 by VA option

=ANALYSIS/EX= LISTINGS OF THE COMMON VARIABLES REFERENCE TABLE DATE 1993/

		D:DEFINE OR EQUIVALENCE	R:REFERENCE	A
		@:READ	-:WRITE	#
		! ¥ ¥ ¥ ¥ ¥ ! ¥ ¥ ¥ ¥ ¥ ! ¥ ¥ ¥ ¥ ¥ ! ¥ ¥ ¥ ¥ ¥ ! ¥ ¥ ¥ ¥ ¥		
		! C C D I I ! L L L N N ! O O O P P ! P R R S S ! T T N P		
		! N V I M T ! P P S M M ! P P P E R ! R E E E R ! G Y A O		
		! T L M P E ! C C P S S ! S T T R M ! T C F T T ! S S L S		
		! L C E L M ! L M C E T ! E C V A C ! V K C C C ! R T 7 T		
		! N T I ! Y T 2 ! T K T T ! R 2 K V L ! T M 7 R		
V-NAME	LABEL-ID			
BUFFER	/BUFF00 -01/			
DD1	/BUFF00 -01/			
DD2	/BUFF00 -01/			
STRING	/BUFF00 -01/		R R	
CVREF¥	/CMREF¥ -01/	-R-X		Z W
IDXNO¥	/CMREF¥ -01/	-A		
NCV¥	/CMREF¥ -01/	R X		D R
ARGNAM	/COMCHA -01/			
CLOCAL	/COMCHA -01/			
CNAME	/COMCHA -01/			

Fig. 3.14 Result of ANALYSIS/EX by CS option

=ANALYSIS/EX= LISTINGS OF THE COMMON VARIABLES REFERENCE TABLE DATE 1993/

		D:DEFINE OR EQUIVALENCE	R:REFERENCE	A
		@:READ	-:WRITE	#
		! ¥ ¥ ¥ ¥ ¥ ! ¥ ¥ ¥ ¥ ¥ ! ¥ ¥ ¥ ¥ ¥ ! ¥ ¥ ¥ ¥ ¥ ! ¥ ¥ ¥ ¥ ¥		
		! C C D I I ! L L L N N ! O O O P P ! P R R S S		
		! N V I M T ! P P S M M ! P P P E R ! R E E E R		
		! T L M P E ! C C P S S ! S T T R M ! T C F T T		
		! L C E L M ! L M C E T ! E C V A C ! V K C C C		
		! N T I ! Y T 2 ! T K T T ! R 2 K V L		
POSITION	LENGTH			
COMMON / BUFF00	/			
STRING	( 1- 6600)			R R
BUFFER	( 6601- 8080)			
COMMON / CMREF¥	/			
CVREF¥	( 1- 96000)	-R-X		Z W
NCV¥	( 96001- 4)	R X		D R
IDXNO¥	( 96005- 3200)	-A		
COMMON / COMCHA	/			
COMNAM	( 3101- 15500)			R
CLOCAL	( 145301- 3100)			
PRGNAM	( 644401- 15500)	- -		

Fig. 3.15 Result of ANALYSIS/EX by CP option

```

CHARACTER*(*) FUNCTION  CHCHK( VALUE )
CHARACTER*(*) VALUE
CHCHK=' '
NV = LEN( VALUE )
M = 1
DO 100 N = 1, NV
  IF( VALUE(N:N) .NE. ' ' ) THEN
    CHCHK(M:M) = VALUE(N:N)
    M = M + 1
  ENDIF
100 CONTINUE
RETURN
END

```

Fig. 3.16 Deletion of space symbol

```

C ADD >>
CHARACTER*8 MEMERR
CHARACTER*80 BUFC, CHCHK, CAPTLF
*INCLUDE PARAMT
*INCLUDE SETUP
C ADD <<
.....

  BUFC = CHCHK( BUFFER(1) )
  IF((CAPTLF(BUFFER(1)(1:9)) .EQ. '*INCLUDE ' .OR.
&    CAPTLF(BUFC      (1:8)) .EQ. ' INCLUDE(') .AND.
&    OPTION(26)          .EQ.  0          ) THEN
C CHANGE <<
  INCLUD=.TRUE.
  COMBUF(COMENT) = BUFFER(1)(1:80)
C      ..... GET INCLUDE MEMBER NAME
.....

  ELSE IF(BUFFER(1)(1:1) .EQ.'C' .OR.
.....

  ELSE
  ENDIF
.....

```

Fig. 3.17 Analysis procedure for INCLUDE statements

```

      SUBROUTINE INCMEM(BUFFER, MEMBER, OLDMEM, MAXMEM, NOLD)
C
C CHARACTER BUFFER*80, MEMBER*8, OLDMEM(*)*8
C ADD >>
      CHARACTER CAPTLF*8
C ADD <<
C CHANGE >> *INCLUDE STATEMENT ( *INCLUDE ABCDE, INSOURCE..... )
      IF( CAPTLF(BUFFER(1:8)) .EQ. '*INCLUDE' ) THEN
C CHANGE <<
      I=9
      DO 100 UNTIL(BUFFER(I:I) .NE. ' ')
        I=I+1
        IF(I.GE.73) THEN
          .....
C INCLUDE STATEMENT ( INCLUDE (ABCDE) )
      ELSE
        N1=INDEX(BUFFER, '(')
        N2=INDEX(BUFFER, ')')
        .....
        DO 20 I=N1+1, N2-1
          IF(BUFFER(I:I).NE. ' ') THEN
            N=N+1
            MEMBER(N:N) = CAPTLF(BUFFER(I:I))
          ENDIF
20    CONTINUE
      ENDIF
      .....
      RETURN
      END

```

Fig. 3.18 Acquisition of member name of include file

```

CHARACTER*54 ALPBET
CHARACTER*65 ALPNUM
DATA ALPBET
& /' ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz_'/
.....

C=====
C          ALPHABET CHARACTER
      ELSE IF(INDEX( ALPBET, LETTER ) .NE. 0 ) THEN
          STRFST=LCV
          DO 10 UNTIL (OK.EQ.0 .OR. EOS)
.....

C=====
C          NUMERIC NUMBER / HOLLERITH CONSTANT ?
      ELSE IF(INDEX('0123456789', LETTER).NE.0) THEN
.....

C=====
C          FLOATING POINT NUMBER ?
      ELSE IF(LETTER.EQ.'.') THEN
.....

C=====
C          CHARACTER CONSTANT (') ?
      ELSE IF(LETTER.EQ.'''') THEN
.....

C=====
C          COLON ?
      ELSE IF(LETTER.EQ.':') THEN
.....

C=====
C          E R R O R   S E C T I O N
      ELSE
          CALL PAGE(2)
          WRITE(6, '(1X, A, A, A, I3)') 'LETTER... ', LETTER, ' LCV=', LCV
          WRITE(6, '(1X, 21H HEX CODE : LETTER = , Z8)') LETTER
          WRITE(6, '(1X, 15H STRING(LCV) = , A, 9H (HEX) = , Z8)')
          &          STRING(LCV), STRING(LCV)
          CALL ERROR(3, ' TOKEN LOGICAL ERROR ')
      ENDIF

```

Fig. 3.19 Detection of unpermitted letters on Fortran compiler

```

LETTER... LCV=3
STRING(LCV) =    (HEX) =    00

**** ANALYZED SUBROUTINE NAME IS SUBABC
** ERROR-Z ** TOKEN LOGICAL ERROR

```

Fig. 3.20 Detection messages of unpermitted letters

```

CHARACTER*8 MEMERR
.....

CALL INCSF (DDMEM, BUFFER, IOCOND, MEMERR)
.....

C -- N O R M A L   S T A T U S  --
C
IF(IOCOND.EQ.0) THEN
.....

ELSE IF(IOCOND.GT.0) THEN
C
C -- E R R O R  ---
C
C      /* I/O ERROR */
      WRITE(6,6000) MEMERR, IOCOND
6000  FORMAT(1X
&      /1X,'*** ERROR *** I/O ERROR IN AT-J LIBRARY (FREAD)'
&      /1X,'          < ANALYSIS MEMBER "READSF" >'
&      /1X,'          ANALYZED PROGRAM LOGICAL ERROR ?!'
&      /1X,'          MEMBER NAME = ',A,' I/O-CODE:',I3)
      STOP
ELSE IF(IOCOND.LT.0) THEN
C
C -- E N D   O F   F I L E  --
C
C      /* END-OF-FILE ENCOUNTERED */
      IF(INCLUD) THEN
.....

```

Fig. 3.21 Treatment of file open error

```

*** ERROR *** I/O ERROR IN AT-J LIBRARY (FREAD)
          < ANALYSIS MEMBER "READSF" >
ANALYZED PROGRAM LOGICAL ERROR ?!
MEMBER NAME = SUB001 I/O-CODE: 1

```

Fig. 3.22 Messages of file open error

## 4. 保守性・操作性の改善

ANALYSISは、1977年の公開以来、複数の人による数回の改良や機能追加が施された。1977年には4000行だったプログラムが、現在は約18000行を越えている。その間に、プログラムの統一性や保守性が欠如してきた。また、入力データはJCL上のSYSINから与えているが、オプションを書き込むときの制限事項が多く、ユーザにとって使い易いツールではないなどの問題点があった。

本章では、ANALYSISの保守性・操作性に関する改良について述べる。

### 4.1 インクルードファイルの整備

ANALYSISで用いられている配列の大きさなどの変更を容易に行うために、以下の改良を行った。

- ① ANALYSISのインクルードファイルに配列のサイズだけを宣言するメンバPARAMTを追加した(Fig. 3.10)。
- ② プログラム中で同じ情報を管理する配列のサイズ宣言を全てPARAMTに移動した。
- ③ 全COMMON配列の宣言をPARAMTに移動した。
- ④ インクルードファイル中でPARAMETER文で宣言されている部分をPARAMT移動した。
- ⑤ COMMON領域を初期化するルーチンINITSではPARAMT内で定義した値を参照するようにした。

変更ルーチンは以下の通りである。

ANAMDF,	BUFFER,	BYTE,	CTBL,
DIMENS,	MIX,	PARAMT,	PO,
SETUP,	SIZE,	WORK	

### 4.2 入力データの書式整備

ANALYSISはJCL上のFT05F001(SYSIN)の入カストリームから入力データ（オプション制御行、補助制御行、コメント行）を与えるようになっている(Fig. 4.1)。ところがANALYSIS-77では、以下のような使用上の制限があった。

- ① オプション制御行を複数指定する場合は、全オプションを1行に書く。複数行にまた

がってはいらない。

- ② 補助制御行は、必ず1カラム目から開始し、空白を含んではならない。
- ③ コメント行は、必ず1カラム目にアスタリスク(\*)を置かなければならない。
- ④ 専用バージョンでは、オプションを複数指定する場合に、オプション同士を必ず空白で区切らなければならない。

ANALYSIS/EXでは上記の点について次のように改良した。

- ① オプションは1行にいくつ書いても構わない。50行以下なら何行にまたがってもよい。
- ② 補助制御行やコメント行はどのカラム目から始めてもよく、空白を含んでもよい。
- ③ オプションを複数指定する場合は区切らなくてもよいし、従来のように、空白かカンマで区切ってもよい。
- ④ 入力データ中にエクスクラメーションマークから始まるインラインコメントを付けることができる。

これは、FT05F001(SYSIN)からの入力データを整理して、ANALYSISの制限に合った形式に変換するサブルーチンREADCDとファンクションOPTCHKを作成し、それらを入力データをチェックするサブルーチン¥OPSET, ¥OPTCK, SETPRMに組み込むことにより可能となった。入力データの与え方の例をFig. 4.1に示す。

### 4.3 ANALYSISコマンド

ANALYSIS/EXは、ANALYSIS-77の解析機能に5種類の機能を追加し、FORTRAN77EX文法仕様としたため、使用法に若干の違いがある。このため、ユーザが容易にANALYSIS/EXを実行できるように、センターコマンドANALYSISを開発した。このコマンドを用いれば、TSSセッションで対話形式でANALYSIS/EXを起動することができる。実行方法は、まず、大型計算機のREADYモードで、

```
ANALYSIS <RET>
```

と入力し、動作モード(TSSかBATCH)を選択する。

TSS動作モードを選択した場合は、約5万行、BATCH動作モードを選択した場合は、約18万行までのプログラムの解析が可能である。いずれのモードを選択しても、被解析プログラム名をと解析結果出力ファイル名と入力データを指定するだけでANALYSIS/EXが起動されるようになっている。さらにこのコマンドでは、プログラム名を誤って入力したとき(ファイルが存在しないとき)に、エラーメッセージと共に再入力の要求をする機能や、ANALYSISの入力データのオプション制御行と補助制御行に関するHELP機能も用意されている。

もし、このコマンドに慣れてきて、対話的に動作させるのが面倒な場合は、同様にREADYモードから、



```
ANALYSIS DSN(SAMPLE.FORT77) INC(SAMPLEINC.FORT77)
          OPT(ANA.INDAT) OUT(SAMPLE.ANALIST) <RET>
```

と指定し、動作モードを選択する。もし、このときに指定したファイル名に誤りがある場合は、エラーメッセージと共に再入力 of 要求をしてくる。

最終的に、TSSの場合は各ファイルを指定装置機番に割り付け、ANALYSISのロードモジュールを呼び出す。BATCHの場合は実行JCLが作成されるので、SUBMITするかどうかを聞いてくるようになっている。実行時の一連の画面イメージをFig. 4.2に示す。

なお、使用法の詳細は付録Aの『ANALYSIS/EX V1.2使用手引書』の4.2節を参照されたい。

```
TWCEI(03 04 06 00 04) SRP NOTIFY(JXXXX)
// EXEC ANALYSIS,
// DSN=' JXXXX.SAMPLE.FORT77', INC=' NULLFILE'
//PT06F001 DD SYSOUT=G
//SYSIN DD *
  LC(62) ELM(*)
  SU(0)  AR(0)  RE(1)  IO(0)
  VA(0)  CO(0)  CS(1)  CP(0)
  NE(1P) C*(0)
  MA(0)  TR(1)  CL(0)
  ! STATIS ASTER NOINC DIST
  * M=MATCHING
    T=MAIN
    T=SUBI
/*
//
```

} オプション制御行  
( Option control line )

} コメント行  
( Comment line )

} 補助制御行  
( Assistance line )

Fig. 4.1 Example of input data

READY  
ANALYSIS

=== THIS COMMAND IS EXECUTED ONLY IN THE READY MODE !! ===

ANALYSIS / EX << VER 1.2 >>				
NO.	MODE	GRAMMAR LEVEL	PROGRAM SIZE	LOGON SIZE
1.	TSS	FORTAN77/EX	50,000 STEP	3MB OR MORE
2.	BATCH	FORTAN77/EX	180,000 STEP	ANY

( 1/2. OTHER=EXIT ) ==> 2

ANALYZED DSNAME ? ==> SAMPLE.FORT77  
 ANALYZED INCLUDE ? ==> SAMPLE.INC.FORT77  
 SYSOUT CLASS OR NAME ?  
 (\*,C,F,G,H,...,FILE) ==> SAMPLE.LIST  
 \* DO YOU READ "OPTION HELP" OF ANALYSIS/EX ? (Y/N) ==> Y

```

*****
* ELM(* ! M1,M3-M5 )           ! MEMBER ELEMENT                *
* SU( 0 ! 1 )                 ! SUBPROGRAM CALL            *
* AR( 0 ! 1 )                 ! ARGUMENTS LIST            *
* RE( 0 ! 1 )                 ! REFERENCE TABLE          *
* IO( 0 ! 1 )                 ! I/O STATEMENTS            *
* CO( 0 ! 1 )                 ! COMMON REFERENCE TABLE   *
* VA( 0 ! 1 )                 ! ARGUMENT AND COMMON VALUE *
* CS( 0 ! 1 ! 2 )             ! DETAIL REFERENCE TABLE   *
* CP( 0 ! 1 )                 ! POSITION AND LENGTH TABLE *
* NE( 0 ! 1 ! 1P ! 1C ! 1PC ) ! SOURCE WITH NEST-STRUCTURE *
* C*( 0 ! 1 )                 ! COMMENT AND ASTERISK      *
* STATIS                      ! STATISTICS OF ANALYSIS/EX *
* LC( NNN ) :N>0             ! LINE COUNT                 *
* ASTER                       ! IGNORE ASTERISK LINE      *
* NOINC                       ! IGNORE INCLUDE STATEMENT  *
* DIST                        ! DISTINCTION BETWEEN SMALL *
* MA( 0 ! 1 )                 ! MATCHING CHARACTER        *
*M=MATCHING                   ! FOR "MA" OPTION (M=:<20)  *
* TR( 0 ! 1 ! 2 ! 3 )        ! TREE STRUCTURE            *
*T=ENTRY ROUTINE             ! FOR "TR" OPTION (T=:<20)  *
* CL( 0 ! 1 )                 ! CLASSIFIED FORTRAN STATEMENT *
*C= ID
*-----1-----2-----3-----4-----5-----6-----*
* AAGACALIDDEEFOCRWBEPFRWICLIDREDNIPCSDEIDPBFSECRSPEDEADIT!FOR *
* SSOSOROFOLLNOPLERANRUEIANOONAOEQAAMAOAIXNERLUUNAETANOENETIN!"CL" *
* SS SMIG SDRFOAICDINWNIQMGTHUAUTMPRMVMTTFOONBTLTOUDNCCB SIA!OPT *
* IIT P T UWEE MNSDTKFNCIDTUPIERBLIAELAMEBERIGCCRRLUPS TOOU PTC! *
* NNOGGIIH NH IA E E ITHN ILCGAL L LIMO NRINRKTUY R E IDDG L E! *
* M OOFFE TI IFT SL D REAECE I ICEN SNNEA IT N NEE A ! *
* N IL F PE EXLRT V SIT IAS MDOI U Y ! *
*****
    
```

\*\*\*

Fig. 4.2 Execution log of ANALYSIS command (continued)

```

EDIT --- JXXXX.@ANAEX.T230911.CNTL ----- COLUMNS 001 072
COMMAND ==> =X                               SCROLL ==> CUR
***** ***** TOP OF DATA *****V10L30*****
000001  TWCE1(03 04 06 00 04) NOTIFY(JXXXX)
000002 // EXEC ANALYSIS,
000003 //      DSN=' JXXXX. SAMPLE. FORT77' ,
000004 //      INC=' JXXXX. SAMPLE. INC. FORT77'
000005 //FT06F001 DD DSN=JXXXX. SAMPLE. LIST,
000006 //      DISP=(NEW, CATLG), UNIT=TSSWK, SPACE=(TRK, (50, 20), RLSE),
000007 //      DCB=(RECFM=FBA, LRECL=137, BLKSIZE=19043)
000008 //SYSIN    DD *
000009  LC( 62) ELM(*)
000010  SU(O)   AR(O)   RE(O)   IO(O)
000011  VA(O)   CO(O)   CS(O)   CP(O)
000012  NE(O)   C*(O)
000013  MA(O)   TR(O)   CL(O)
000014 ! STATIS  ASTER   NOINC   DIST
000015 ! T=MAIN
000016 ! M=MATCHING
000017 !C=
000018 *** LOOK OPTION HELP DATASET. : ' JXXXX. ANA. HELP' .   ***
000019 /*
000020 //
***** ***** BOTTOM OF DATA *****

```

\* DO YOU SUBMIT THIS ANALYSIS JOB ? (Y/N) ==> Y  
JOB (FXXXX001) SUBMIT

Fig. 4.2 Execution log of ANALYSIS command

## 5. おわりに

ANALYSISの機能拡張の目的は、

- ① FORTRAN77EX上で動作させること、
- ② 被解析プログラムの解析行数を増加させること、
- ③ FORTRAN77EX文法で記述されたプログラムも解析可能とすること、
- ④ 操作性、保守性、解析機能の向上を図ること、

である。①と③は、コンパイラの変更に伴う拡張である。②は、20万行にも及ぶ原子力コードの解析を可能とするためである。

FORTRAN77EX上で動作させるためには、FORTRAN77EXの互換・非互換項目の、JIS-FORTRAN77の仕様の確認をする必要があった。これは、FORTRAN77が旧コンパイラの仕様をサポートしているのに対し<sup>(8)</sup>、FORTRAN77EXでは旧コンパイラの仕様の一部を削除しているため、ANALYSIS自身が翻訳できないことと、またFORTRAN77EXでは、プログラミングの自由度を増すための新機能を備えたためである。<sup>(6) (7)</sup>

被解析プログラムの解析行数を増加させるために、解析のための作業配列の宣言サイズを変更するとともに、AE化を施した。この結果、ANALYSIS/EXでは、どのような大規模なプログラムでも解析可能である。

FORTRAN77EX文法で記述されたプログラムも解析可能とするために、FORTRAN77EX文法の機能強化内容を抽出しANALYSISを拡張した。この結果ANALYSIS/EXでは、旧仕様を含むFORTRAN77EX文法の解析が可能となった。

操作性、保守性、解析機能の向上を図るために、ANALYSIS-77では解析不十分な項目を抽出し、それらを解決する方向へ機能拡張を行った。この機能拡張では、ANALYSISの大型計算機以外への移植性を意識し、JIS-FORTRAN77仕様範囲内でのプログラミングを行うとともに、変数の宣言方法等の基本的な部分を修正、統一した。また、使用法を容易にするために、新しく対話形式のANALYSISコマンドを作成した。これらの改良の結果、ANALYSIS/EXは、原研の標準ツールとして、今後広く利用されることが期待される。

## 謝 辞

本コードの機能，並びに使用法に関するご指摘をして下さいました情報システムセンタープログラム相談室の鶴岡卓哉氏（現 那珂研炉心プラズマ研究部），本コードの改良，並びに本報告書の作成に貴重な意見を賜いました情報システムセンター外来研究員の根本俊行氏，また，センター運用方法に関して貴重な意見を賜いました情報システムセンターの富山峯秀氏，畑山智氏（現，那珂研炉心プラズマ研究部），システム環境その他に貴重な意見を賜いました富士通㈱の相沢広氏，事務処理関係を担当していただきました富士通㈱の山田圭子氏，他の諸氏に感謝致します。

本報告書を書く機会を与えて下さいました情報システムセンター室長秋元正幸氏，並びに富士通㈱科学システム統括部担当部長橋本道夫氏に深く感謝致します。

## 参 考 文 献

- 1) 計算センター，原子力コード開発・整備作業の効率化（計算センターにおける第一種外来研究員の活動について），私信，1979
- 2) 計算センター，FORTRANプログラム変換・解析ソフトウェアツール使用手引書，私信，1979
- 3) 計算センター，原子力コード開発・整備作業報告（計算センター第一種外来研究員の活動について），私信，1981
- 4) 計算センター，原子力コード開発・整備作業報告（計算センター第一種外来研究員の活動について），私信，1983
- 5) 計算センター，AT-J仕様手引書(V02/L01用)，1983
- 6) 根本俊行，鈴木孝一郎，渡辺健二，町田昌彦，長内誠志，磯辺信雄，原田裕夫，横川三津夫，新FORTRANコンパイラの導入とベクトル計算機の効果的利用法，JAERI-M 92-105，1992
- 7) 富士通㈱，FORTRAN 77 EX使用手引書 V12用，マニュアル 79SP-5031-1，1991
- 8) 富士通㈱，FORTRAN 77文法書 1990年6月版，マニュアル 99SP-8031，1991
- 9) 浦 昭二，FORTRAN 77入門，株式会社培風館，1982

## 謝 辞

本コードの機能，並びに使用法に関するご指摘をして下さいました情報システムセンタープログラム相談室の鶴岡卓哉氏（現 那珂研炉心プラズマ研究部），本コードの改良，並びに本報告書の作成に貴重な意見を賜いました情報システムセンター外来研究員の根本俊行氏，また，センター運用方法に関して貴重な意見を賜いました情報システムセンターの富山峯秀氏，畑山智氏（現，那珂研炉心プラズマ研究部），システム環境その他に貴重な意見を賜いました富士通㈱の相沢広氏，事務処理関係を担当していただきました富士通㈱の山田圭子氏，他の諸氏に感謝致します。

本報告書を書く機会を与えて下さいました情報システムセンター室長秋元正幸氏，並びに富士通㈱科学システム統括部担当部長橋本道夫氏に深く感謝致します。

## 参 考 文 献

- 1) 計算センター，原子力コード開発・整備作業の効率化（計算センターにおける第一種外来研究員の活動について），私信，1979
- 2) 計算センター，FORTRANプログラム変換・解析ソフトウェアツール使用手引書，私信，1979
- 3) 計算センター，原子力コード開発・整備作業報告（計算センター第一種外来研究員の活動について），私信，1981
- 4) 計算センター，原子力コード開発・整備作業報告（計算センター第一種外来研究員の活動について），私信，1983
- 5) 計算センター，AT-J仕様手引書(V02/L01用)，1983
- 6) 根本俊行，鈴木孝一郎，渡辺健二，町田昌彦，長内誠志，磯辺信雄，原田裕夫，横川三津夫，新FORTRANコンパイラの導入とベクトル計算機の効果的利用法，JAERI-M 92-105，1992
- 7) 富士通㈱，FORTRAN 77 EX使用手引書 V12用，マニュアル 79SP-5031-1，1991
- 8) 富士通㈱，FORTRAN 77文法書 1990年6月版，マニュアル 99SP-8031，1991
- 9) 浦 昭二，FORTRAN 77入門，株式会社培風館，1982

JAERI-M 93-155

付録A

# *ANALYSIS/EX*

*Ver. 1.2*

## 使用手引書

(第2版)

平成 5年 3月21日

日本原子力研究所

情報システムセンター

©Computing and Information Systems Center,  
Japan Atomic Energy Research Institute, 1993

この使用手引書は、日本原子力研究所におけるFORTRAN77EX静的解析ツール『ANALYSIS/EX Ver1.2』並びに、それを動作させる『ANALYSIS』コマンドのために作成されたものです。複製、転載の際は著者の許可が必要となります。

日本原子力研究所 情報システムセンター



## 1 概要

『ANALYSIS/EX Ver. 1.2』は、FORTRAN 77 言語によって記述されたソースプログラムの静的解析を行うツールである。これは、従来公開されていた『ANALYSIS-77』『ANALYSIS/EX v1.1』のFORTRAN 77 EX 対応バージョンである（コンパイラ、文法）。

従来の『ANALYSIS-77』は、FORTRAN 77 EX コンパイラに対応していない他、巨大プログラムの解析は不可能であった。それを『ANALYSIS/EX Ver. 1.1』ではFORTRAN 77 EX コンパイラに対応させるとともに、最大約18万行の巨大プログラムの解析を可能とした。今回は更にFORTRAN 77 EX で新規に追加された文法に対応させ、従来の機能にCOMMON中の配列や変数に関する詳細な定義参照関係を出力する機能、インクルード行を無視する機能、英小文字名標を判別する機能等を設けた。

以降『ANALYSIS/EX Ver. 1.2』を、単に『ANALYSIS/EX』と表記することとする。

## 2 ジョブフロー

『ANALYSIS/EX』は、被解析プログラムをFT01F001から入力し、インクルードファイルがあればFT02F001から読み込み、SYSIN(FT05F001) から与えられるオプションを元に解析する。その際にFT03F001、FT04F001を作業用のファイルとして使い、FT06F001に結果を出力する。そのジョブフローを図1に示す。

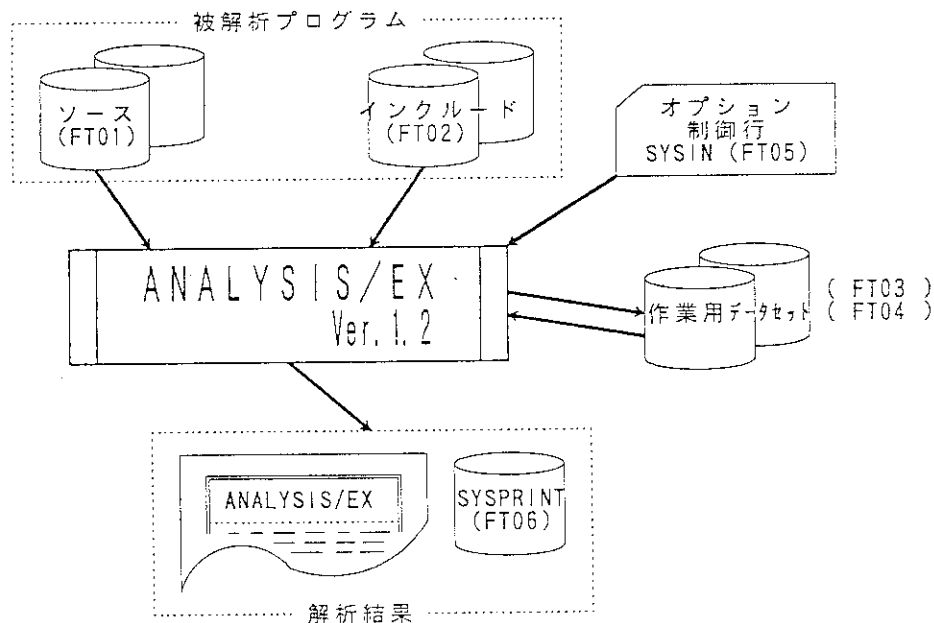


図1 ANALYSIS/EXのジョブフロー

### 3 入力データ (オプション制御行)

入力データは、『ANALYSIS/EX』の解析オプションを指定するもので、オプション制御行、補助制御行、コメント行に大別される。基本的な使用法は、従来版の『ANALYSIS-77』と同様であるが、COMMONの定義参照詳細図を出力するオプション、第1カラム目の『\*』を非コメント文として扱うオプション、インクルード文やインクルード行を無視するオプション、英大文字小文字名標を区別して扱うオプションが追加されている。

入力データはFT05F001から入力するが、センターで用意したJCL上ではSYSINから入力することになっている。尚、入力データは最大50行まで指定できる。

```

例. //SYSIN DD *
      NB(1P) TR(3) CL(1)  ! オプション制御行
      T = MAIN           ! 補助制御行 ( TR(3)に対応 )
      C=  X      X      X      ! 補助制御行 ( CL(1)に対応 )
      ! 各オプションの詳細は、3.2を参照のこと。
      * コメント行は、*または、! から始まる。詳細は3.3
/*

```

#### 3.1 入力データの概要

オプションの与え方は、オプション制御行の指定だけでよいものと、補助制御行の指定を伴うものの2種類に分けられる。いずれも、オプションを記述しない場合は、指定しないのと同じである。表1にオプション制御行の指定だけでよいものを、表2に補助制御行を伴うオプションの概要を説明する。

各制御行の詳細は3.2に示してあるので、この項はクイックリファレンスとして参照していただきたい。

表1. オプションの機能概要 (続く)

No	オプションの形式	機能概要
1	ELM(X)	被解析プログラムがPOの場合、Xに解析対象ノブを指定する。全ノブを指定する場合はELM(*)とする。詳細は3.2参照。
2	SU(1)	各ルーチンから呼ばれているサブルーチンと関数を出力。
3	AR(1)	プログラム中で使用されている関数の出力。
4	RE(1)	各ルーチンと関数の相互参照表を出力。
5	IO(1)	各ルーチン毎に使用している入出力文関連行を出力。
6	CO(1)	COMMONラベルと各ルーチンの参照関係表を出力。
7	VA(1)	COMMON変数の定義、サブルーチンへの引数渡しの状況出力。

表 1. オプションの機能概要 (続き)

No	オプションの形式	機能概要
8	CS(1) CS(2)	COMMON変数と各ル-フでの定義参照関係図をCOMMONレベル順出力。 COMMON変数と各ル-フでの定義参照関係図をCOMMON変数順出力。
9	CP(1)	COMMON変数の長さとは各ル-フでの定義参照関係図を出力。
10	NE(1X)	ソース中の IF-THEN-ELSE, DO-CONTINUE等をリスト付で出力。 ・ X=P の時はサブ-ル-フ毎に改ページする。 ・ X=C の時 *INCLUDE をコメントと見なす。 ・ X=PCの時 *INCLUDE をコメントと見なし、サブ-ル-フ毎に改ページする。
11	C*(1)	注釈文(C, *, ! ) の行出力。
12	STATIS	解析時に使用する ANALYSIS/EX 内の配列のサイズの情報を出力。
13	LC(X)	解析結果の1ページの出力行数を指定する。指定しない場合は62行。
14	ASTER	1カラム目の *を空白と置き換え、非コメントとして扱う。
15	NOINC	*INCLUDE行, INCLUDE文を無視する。
16	DIST	英大文字名標と英小文字名標を別変数として扱う。

表 2. 補助制御行の必要なオプションの機能概要

No	オプションの形式	機能概要
17	MA(1) 補助制御行	補助制御行で指定した変数や配列を使用しているル-フの出力。 『M=変数名or配列名』を指定。 例. M=ABC
18	TR(1) TR(2) TR(3) 補助制御行	各ル-フ間の参照関係を木構造図で出力。 TR(1) からFORTRAN の基本外部関数を除いて出力。 TR(2) に加え呼び出し回数の多いル-フを分けて出力。 『T=入口ル-フ名』を指定。 例. T=MAIN
19	CL(1) 補助制御行	補助制御行で指定したFORTRAN 文を使用している行の出力。 『C=』から始め、各FORTRAN 文に対応したカラムに任意の英数字を指定する。 詳細は 3.2 と表 3 参照

### 3.2 オプション制御行と補助制御行の詳細説明

ここでは、各オプションの意味と使用方法について具体的に説明する。

オプション制御行は、『ANALYSIS/EX』の解析オプションを指定する行である。各オプションは、空白またはカンマで区切るか、べた書きして与える。また、1行1オプションで与えてもよい。ELM、STATIS、LC、ASTER、NOINC、DISTを除く各オプションには、そのオプション自体を有効にするか無効にするかを、括弧『(n)』内の数字nで指定できる。オプションを指定しない場合の括弧内の標準値は

『(0)』である。

補助制御行は、オプション制御行の補助指定するものである。補助制御行が必要なオプションは、MA（指定した変数や配列をチェック）、TR（プログラム構造を木構造で出力）、CL（指定したFORTRAN文をチェック）の3種類である。①7～①9を参照されたい。

#### (1) ELM(X) オプション

被解析プログラムが区分編成ファイル(PO)の場合、そのデータセット中の解析対象メンバを指定する。順次編成ファイル(PS)の場合は指定しても無効となる。

指定方法は、Xの部分にメンバ名、或いはその範囲を指定する。メンバの範囲は先頭のメンバ名から、最後のメンバ名までを『-』でつなぐ。複数のメンバを指定する場合は、各メンバをカンマ『 , 』で区切って指定する。全メンバを対象とする場合は『\*』とする。これは、あくまでデータセット中のメンバ名を指定するオプションであって、プログラム名標を指定してはならない。

例. ELM(MEM1, MEM2-MEM7, MEM10)  
ELM(\*)

#### (2) SUオプション

- ①SU(0) : SUオプションを無効にする。
- ②SU(1) : 各ルーチン毎に、副プログラムを引用しているFORTRAN行(CALL文など)を出力する。下位ルーチンの引数等の確認ができる。図2に出力例を示す。

#### (3) ARオプション

- ①AR(0) : ARオプションを無効にする。
- ②AR(1) : 各ルーチン毎に、そのルーチンが他のルーチンからCALL文等で呼ばれている場合にそのFORTRAN行を出力する。引数の確認もできる。図3に出力例を示す。

#### (4) REオプション

- ①RE(0) : REオプションを無効にする。
- ②RE(1) : 各ルーチンの親ルーチンと子ルーチンを出力(クロスリファレンス)する。図4に出力例を示す。

#### (5) IOオプション

- ①IO(0) : IOオプションを無効にする。
- ②IO(1) : 各ルーチン毎の入出力関連文(READ, WRITE, FORMAT, REWIND, OPEN, CLOSE等)を出力する。図5に出力例を示す。

## (6) COオプション

- ①CO(0) : COオプションを無効にする。
- ②CO(1) : COMMON文と各ルーチンの参照関係の一覧(使用されている場合は「\*」で表示される)と、COMMON文の宣言の内容を表形式で出力する。但し、同じCOMMONラベルでもCOMMON変数の並びが異なる場合は、それらを分けて印刷する。図6に出力例を示す。

## (7) VAオプション

- ①VA(0) : VAオプションを無効にする。
- ②VA(1) : 各ルーチン毎に、COMMON変数に値を陽に定義しているもの、他のサブルーチンへ引数として渡しているものがあれば、その旨のメッセージを出力する。図7に出力例を示す。

## (8) CSオプション

- ①CS(0) : CSオプションを無効にする。
- ②CS(1) : 各COMMON変数の定義参照等の詳細な関係図を出力する。この際、COMMONラベルが同じでも変数名が異なる場合は、ラベルIDで判別できるようになっている。出力はCOMMONラベル順にソートされて出力される。図8に出力例を示す。
- ③CS(2) : CS(1)と同様だが、COMMON変数順にソートされて出力される。

(注) 関係図内のCOMMON変数の使用状況を示す記号は、

- D (DEFINE) : 定義されているか、EQUIVALENCEにより定義されている。  
 R (REFERENCE) : 参照されている。  
 A (ARGUMENT) : 下位ルーチンに引数として渡されている。  
 @ (READ) : READ文によって入力されている。  
 - (WRITE) : WRITE文によって出力されている。  
 # (I/O STMT) : I/O関連の制御情報に使用されている。

の6項目で分別しており、更にこれらを組み合わせて、

W : D + R,      X : R + A,      Y : D + A,      Z : D + R + A  
 \* : @ + #,      % : # + -,      & : @ + -,      ¥ : @ + - + #

の8種類を加えた計14種類の記号で表している。尚、これらの記号の意味は、出力された表の右上に示されるので参考にされたい。

## (9) CPオプション

- ①CP(0) : CPオプションを無効にする。
- ②CP(1) : CSオプションと同様の出力であるが、COMMONラベル中のCOMMON変数(配列)の位置(ポジション)と大きさ(長さ)を表示する。図9に出力例を示す。

(注) 関係図内のCOMMON変数の使用状況を示す記号は、CSオプションと同様で、

D, R, A, @, -, #, W, X, Y, Z, \*, %, &, ¥の14種類で表される。各々の意味はCSオプションと同じである。

## (10) NE オプション

- ① NE ( 0 ) : NE オプションを無効にする。
- ② NE ( 1 X ) : 『 IF 』や『 DO 』のネスト構造を明示しながら全ての FORTRAN 文を出力する。図10に出力例を示す。  
     X = P : ルーチン毎に改ページして出力する。  
     X = C : \* INCLUDE 文をコメントと見なす。  
     X = PC : P と C の両方指定。

## (11) C\* オプション

- ① C\* ( 0 ) : C\* オプションを無効にする。
- ② C\* ( 1 ) : 注釈文を出力する。但し、(10)の ASTER を指定したときは \* から始まる FORTRAN 文は、\* INCLUDE、\* VOCL、\* EJECT のみの出力となる。図11に出力例を示す。

## (12) STATIS オプション

STATIS オプションは、『 ANALYSIS / EX 』内部で取られている配列のサイズと、解析時に実際に使用した配列のサイズを出力する。

解析途中で配列の宣言サイズを越えた場合は、指定しなくてもエラーメッセージと STATIS を出力して解析が途中終了するようになっている。図12に出力例を示す。

## (13) LC ( n ) オプション

解析結果の 1 ページの出力行数を指定する。標準値は  $n = 62$  である。

## (14) ASTER オプション

翻訳オプションの ASTER と同意のオプションである。

プログラムの任意の行の第 1 カラム目が 『 \* 』から始まる時に、その 『 \* 』を空白に置き換え、その行を開始文か継続行と見なす。但し、『 \* INCLUDE ~ ~ 』はインクルード行として扱い、『 \* EJECT 』『 \* VOCL ~ ~ 』等はコメントとして扱う。

## (15) NOINC オプション

プログラム中のインクルード文及び、インクルード行を無視する。開発途中のプログラムの解析（木構造図、引数チェックなど）に利用できる。当然、インクルードファイル内にコモンの宣言がある場合は、少なくともコモンの解析は正確に出力されないし、インクルードファイル内に CALL 文がある場合は TREE 構造図も正確に出力されないので注意されたい。

## (16) DIST オプション

FORTRAN 77 / EX から『英小文字名標』が使用可能になった。実際には大型計算機上での翻訳時には、英大文字でも英小文字でも綴りが同じ場合は同じ名標（変数

名、プログラム名等)として扱うが、UNIXマシン等では、翻訳オプションで別名標として扱うことも可能である。

そこで、このオプションを指定すると、プログラム中の名標に英大文字と英小文字の双方が含まれる場合、それぞれを区別して扱う。

但し、UNIXマシン上で開発されたコードなどは、英大文字小文字の区別をしているかどうかを確認した上で使用しないと、他のオプションの解析結果が実際と異なる場合が出てくるので注意されたい。また、日本語名標はサポートしていない。

#### (17) MAオプション (補助制御行が必要)

- ① MA(0) : MAオプションを無効にする。
- ② MA(1) : 補助制御行で入力した文字列を検索し、それを含むFORTRAN行とそのルーチン名を出力する。

補助制御行 : ②を指定したときに、『M=name』のように指定する。nameは31文字までの空白を含まない任意の文字列である。『M=name』は必ず1行に1つだけ指定し、最大20個まで指定できる。図13に出力例を示す。

(注) 通常、補助制御行は英大文字で指定するが、(16)のDISTオプションを指定した場合は、英大文字と小文字を正確に指定しないと正確に解析できない。

補助制御行に空白を含む文字列を指定すると、その文字列の空白より前の文字列に対する検索を行う。つまり、

```
M=ANALYSIS /EX V1.2
```

の場合は、空白より前の文字列である『ANALYSIS』だけが有効になる。

補助制御行に何も指定しない場合は、何も出力されない。

#### JCL指定例

```
//SYSIN DD *
MA(1)
M=VAR001
M=DIM001
M=VAR002
/*
```

#### (18) TRオプション (補助制御行が必要)

- ① TR(0) : TRオプションを無効にする。
- ② TR(1) : 被解析プログラム内の副プログラムの木構造を全て出力する。
- ③ TR(2) : FORTRANの基本外部関数を除いた木構造を出力する。
- ④ TR(3) : TR(2)に加え、2個以上の親ルーチンがあり、かつ、2個以上の子ルーチンを持つサブルーチンを木の本体から切り離して出力する。

補助制御行 : ②~④を指定したときに木構造の入口ルーチン名を『T=name』のように指定する。『T=name』を複数枚指定すると、そのルーチン以下が別の木構造で示される。『T=name』は必ず1行に1つだけ指定し、最大20種類まで指定できる。入口ルーチン名は必ずしもMA





表3. CLオプション上の補助制御行の各カラムと対応FORTRAN文

No	FORTRAN 文	No	FORTRAN 文	No	FORTRAN 文
1	C	31	LOGICAL	61	AT
2	=	32	INTEGER	62	DISPLAY
3	ASSIGNMENT	33	CHARACTER	63	INIT
4	ASSIGN	34	DOUBLE PRECISION	64	TRACE
5	GO TO	35	REAL	65	--- ROGUE ---
6	ASSIGNED GO TO	36	EQUIVALENCE		
7	COMPUTED GO TO	37	DATA		
8	ARITHMETIC IF	38	NAMelist		
9	LOGICAL IF	39	IMPLICIT		
10	IF(-----) THEN	40	PARAMETER		
11	DO	41	COMMON		
12	DO UNTIL(-----)	42	SAVE		
13	DO WHILE(-----)	45	DIMENSION		
14	ELSE	44	EXTERNAL		
15	ELSE IF(-----) THEN	45	INTRINSIC		
16	ENDIF	46	DEFINE FILE		
17	FORMAT	47	PROGRAM		
18	OPEN	48	BLOCK DATA		
19	CLOSE	49	FUNCTION		
20	READ	50	SUBROUTINE		
21	WRITE	51	ENTRY		
22	BACK SPACE	52	CALL		
23	ENDFILE	53	RETURN		
24	PRINT	54	STOP		
25	PUNCH	55	PAUSE		
26	REWIND	56	END		
27	FIND	57	CONTINUE		
28	WAIT	58	DECODE		
29	INQUIRE	59	ENCODE		
30	COMPLEX	60	DEBUG		

※ 以上の(2)~(11)、(17)~(19)までのオプションを1枚以上指定すると、被解析プログラム統計情報が出力される。統計情報には、プログラムの解析行数や、各FORTRAN文の使用数等が含まれる。図16に出力例を示す。

### 3.3 コメント行

3.1、3.2のオプション制御行中にコメントを任意行挿入することができる。挿入方法は、FORTRAN 77 EXのコメントのように、オプション制御行中に『!』から始まる文字列(空白を含む)を記述する。『!』を行の先頭に記述すると、その行全体がコメント行となる。また、行頭を『\*』で始めるコメント行も、従来通りサポートされている。

尚、『\*』行と『!』以降の文字列は『ANALYSIS/EX』の解析結果(リスト)には出力されない。

以下の例では、オプションのCO(1)とSU(0)とNE(1P)のみが反映され、残りはコメントとして認識される。

```

例 0----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
//SYSIN DD *
    * THIS IS A COMMENT.
    CD(1),SU(0) ! COMMON
    NE(1P), , ! SOURCE WITH NEST !
    ! ANALYSIS/EX COMMENT LINE !!
/*

```

### 3.4 入力データの制御行の指定例

ここでは、いくつかの使用例についての入力データを示す。

- (1) 入出力文と副プログラムの親ルーチン、子ルーチンを知りたい。また、FORTRAN変数 'VAR1' と 'VAR2' がどの副プログラムのどのFORTRAN文で引用されているか知りたい。

```

例 0----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
//SYSIN DD *
    IO( 1 ) MA(1)
    RE(1 ) ! CALLER AND CALLEE
    * FOR MA(1) OPTION
    M = VAR1
    M= VAR2
/*

```

- (2) COMMON文の定義参照関係とサイズを知りたい。

```

例 0----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
//SYSIN DD *
    ELM( * ) ! PO FILE
    CP ( 1 ) ! COMMON REFERENCE
/*

```

- (3) プログラムの木構造図を知りたい。ただし、副プログラムCONTROLとEDT01はたくさんのルーチンから呼ばれて、かつ子ルーチンも多いので、別エントリーとしたい。

```

例 0----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
//SYSIN DD *
    TR( 1 )
    ! FOR TR(1) OPTION
    T = MAIN
    T = CONTRL
    T = EDT01
/*

```



## 4 実行方法

### 4.1 カタログド・プロシジャを利用しての実行

「ANALYSIS/EX」を動作させるのに必要なJCL上のパラメータは、表4のようになっている。

表4. ANALYSIS/EX実行JCL上のパラメータ

パラメータ	用途	標準値
DSN	被解析対象プログラム名	ユーザが指定
INC	被解析対象インクルード名	'NULLFILE'
UNIT	FT03、FT04の作業データセットのUNIT名	'WK10'
SPC3	FT03のファイルのスペース量	'300'
SPC4	FT04のファイルのスペース量(GEM解析時使用)	'100,50'
BLK4	FT04のブロック長(GEM解析時使用)	'6000'
OUT	SYSPRINT, FT06F001 の出力クラス	'SYSOUT=*'
ORECFM	SYSPRINT, FT06F001 のレコード形式	'FBA'
OB	SYSPRINT, FT06F001 のブロック長	'19043'
SYSIN	FT05に対するDD名 (ANALYSISのオプション入力)	'DDNAME=SYSIN'

#### ● センターのカタログド・プロシジャを利用しての実行例

```

TWC(3 4 3 0 4) SRP
// EXEC ANALYSIS.
//   DSN='J####.SAMPLE.FORT77', INC='NULLFILE'
//SYSIN DD *
BLM(*) SU(0) CO(1) RE(0) AR(0) IO(0) MA(0) TR(5) VA(0) CS(0) CP(1)
NE(IP) CL(1) C*(1) LC(108) STATIS ASTER NOINC DIST
T=MAIN
/*
//

```

※ オプション制御カードの指定方法は3.参照。

※ カタログド・プロシジャに関しては図17参照。

## 4.2 ANALYSISコマンドによる実行

「ANALYSIS/EX」を円滑に利用していただくために、センターコマンドを用意した。使用法はREADYプロンプトの画面から、

ANALYSIS

と打つだけでよく、後は聞かれる内容に逐一返答するだけでよい。

- (1) 先ず「ANALYSIS」コマンドを用いて起動。

```
READY
ANALYSIS
```

- (2) この後に、TSSで実行 (FORG GROUND)するか、BATCH (BACK GROUND)で実行するかを聞いてくる。この時、このメニューに示されるメッセージを読んでもらいたい。

```

=== THIS COMMAND IS EXECUTED ONLY IN THE READY MODE !! ===
-----
          A N A L Y S I S / E X << VER 1.2 >>
-----
NO.      MODE      GRAMMAR LEVEL  PROGRAM SIZE  LOGON SIZE
-----
  1.     TSS       FORTRAN77/EX   50.000 STEP   3MB OR MORE
  2.     BATCH     FORTRAN77/EX   180.000 STEP   ANY
-----
( 1/2. OTHER=EXIT ) ==>

```

※ 1のTSSを選択する場合、被解析プログラムは比較的小規模のものに限る、また、この時のLOGON時のリージョンサイズは3MB以上でなければならない。3MB未満の場合、ロードモジュールを呼ぶ際に106C (リージョンサイズが不足している場合のエラーコード)で異常終了するので注意が必要である。もし、リージョンサイズが足りない場合は、1と2以外の文字を入力するか、何も入力しないで実行キーを打つとその場で終了する。

```
READY
LOGON JXXXX S(4M) ← Sはサイズパラメータでここでは4MBを指定している。
```

これで、リージョンサイズ4MBでLOGONし直すことができる。

2のBATCHを選択した場合、約18万行のプログラムの解析が可能である。これを選択した場合は、最終的にJCL (Job Control Language)を作成するので、LOGON時のリージョンサイズは関係ない。

ここで、1 (TSS)を選択すると、更に次のメッセージが出力される。

```
*** LOGON REGION SIZE MUST BE 3MB OR MORE !! ***
```

- (3) 次に被解析プログラム名、同インクルード名を聞いてくるので、自分のID下に存在するデータセット名を指定する。但し、フルネームでは指定できないので、PERMITのかかった他ID下のデータセットを指定したい場合は、あらかじめ自分のID下にコピーしておく必要がある。

```
ANALYZED DSNAME ? ==> SAMPLE.FORT77
ANALYZED INCLUDE ? ==> SAMPLE.INC.FORT77
```

- ※ この時に、誤って存在しないデータセット名を指定するとエラーが返ってくるので、再度指定し直す。インクルードファイル指定時の場合を示す。

```
ANALYZED INCLUDE ? ==> AAMPLE.INC.FORT88
*** FILE ALLOCATION ERROR OCCURED << JXXXX.AAMPLE.INC.FORT88 >> ***
ANALYZED INCLUDE ? ==>
```

- (4) 次に「ANALYSIS/EX」の解析結果の出力先を指定する。1文字しか指定しない場合は、出力クラスの場合もあり得るので、再度確認してくる。そのクラスでよければ「Y」を入力すればよい。

```
SYSOUT CLASS OR NAME ?
(*, C, F, G, H, ..., FILE) ==> H
* SYSOUT CLASS IS 'H'-CLASS ? (Y/N) ==> Y
```

- ※ よくない場合や誤って入力した場合は「N」を指定すると、再度聞いてくる。

```
SYSOUT CLASS OR NAME ?
(*, C, F, G, H, ..., FILE) ==> *
* SYSOUT CLASS IS '*'-CLASS ? (Y/N) ==> N
SYSOUT CLASS OR NAME ?
(*, C, F, G, H, ..., FILE) ==>
```

- ※ 2文字以上を指定すると、データセット名として判断する。このデータセット名は新たに作成するものでもいいし、既に存在するものでもいい。

```
SYSOUT CLASS OR NAME ?
(*, C, F, G, H, ..., FILE) ==> SAMPLE.ANALIST
```

- ※ 仮に1文字のデータセット名を作成したい場合は、「FILE-NAME」か単に、「FILE」と指定すると、その後再度聞いてくるので、そこで例えば「G」と入力するとJXXXX.Gというデータセットが指定されることになる。勿論、もっと長い名前を指定してもよい。

```
SYSOUT CLASS OR NAME ?
(*, C, F, G, H, ..., FILE) ==> FILE
SYSOUT FILE NAME ? ==> G      ← JXXXX.G というデータセットの指定
```

- (5) (2)で1(TSS)を選んだ場合は、次のようなメッセージが出力される。

「OPTION HELP」とは、「ANALYSIS/EX」のオプションの説明が

記述されたファイル「ANA.HELP」で、「Y」を指定することにより見ることができる。(2 (BATCH) を選択した場合の説明は(9)からの説明を参照)

```

=====
LET'S OPEN "OPTION INPUT FILE (SYSIN)"
PLEASE, CHANGE ANALYSIS/EX INPUT OPTION !!
=====
* DO YOU READ "OPTION HELP" OF ANALYSIS/EX ? (Y/N) ==> Y

*****
* ELM(* ! M1.M3-M5 )          ! MEMBER ELEMENT          *
* SU( 0 ! 1 )                ! SUBPROGRAM CALL      *
* AR( 0 ! 1 )                ! ARGUMENTS LIST      *
* RE( 0 ! 1 )                ! REFERENCE TABLE   *
* IO( 0 ! 1 )                ! I/O STATEMENTS     *
* CO( 0 ! 1 )                ! COMMON REFERENCE TABLE *
* VA( 0 ! 1 )                ! ARGUMENT AND COMMON VALUE ASSIGNMENT *
* CS( 0 ! 1 ! 2 )           ! DETAIL REFERENCE TABLE OF COMMON *
* CP( 0 ! 1 )                ! POSITION AND LENGTH TABLE OF COMMON *
* NE( 0 ! 1 ! 1P ! 1C ! 1PC ) ! SOURCE WITH NEST-STRUCTURE *
* C*( 0 ! 1 )                ! COMMENT AND ASTERISK *
* STATIS                      ! STATISTICS OF ANALYSIS/EX *
* LC( NNN ) :N>0             ! LINE COUNT          *
* ASTER                       ! IGNORE ASTERISK LINE *
* NOINC                       ! IGNORE INCLUDE STATEMENT *
* DIST                        ! DISTINCTION BETWEEN SMALL AND CAPITAL *
* MA( 0 ! 1 )                ! MATCHING CHARACTER  *
* M=MATCHING                  ! FOR "MA" OPTION (M=:<20) *
* TR( 0 ! 1 ! 2 ! 3 )       ! TREE STRUCTURE      *
* T=ENTRY ROUTINE            ! FOR "TR" OPTION (T=:<20) *
* CL( 0 ! 1 )                ! CLASSIFIED FORTRAN STATEMENT *
* C= ID                       *
*-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----*
* AAGAC|DDDEE|FOCRWB|EPPRF|WICL|CDREDN|PCSDE|DPBFSECR|SPECED|ADIT|FOR *
* SSOSOR|FOO|LLNOPLER|ANRUE|ANOONAO|EQAA|MAO|IXNERLU|UNAET|ANOENET|NR|"CL"*
* SSSMIG SSDREOA|CD|NWN|IQMG|THUAUT|MPRMV|MTTF|FOONB|TLT|OUDN|CCB|SIA|OPT *
* IIT P T UWEE MNSDT|KFNC|IDTUP|IERBL|AELAMEE|ERIG|CCRR|LUPS TOOU|PTC| *
* NNOGG|IH NH IA E E ITHN ILCGAL L LIMO NR|NRKTUY R E IDDG L E| *
* M OOFFE TI IFT SL D REAECE |ICEN SNNEA IT N NEE A | *
* N IL F PE EXLRT V SIT IAS MDOI U Y | *
*****

```

(6) (2)で1 (TSS) を選んだ場合は、次にオプションを指定する画面が開く。データセット名は、このコマンドが任意につける名前である。ここで指定したいオプションを指定し、その後に画面を閉じると自動的に実行される。指定の仕方はこの使用手引書の3.を参照するか、次のEDIT画面のCOMMANDラインから「LIST」コマンドで「ANA.HELP」を見るとよい。

```

COMMAND ==> X LIST ANA.HELP
           ↑
          または、
COMMAND ==> /LIST ANA.HELP
           ↑

```

LISTコマンドの前の「/」 「X」は、  
PPD コマンドのLISTと区別するため  
に必ずつける。

と指定すると、(5)の「OPTION HELP」が表示される。





```

BROWSE - JXXXX.SAMPLE.ANALIST ----- LINE 00000 COLS 001 080
COMMAND ==>                               SCROLL ==> CUR
***** TOP OF DATA *****-CAPS ON-***
=ANALYSIS/EX= STATIC ANALYSIS OF THE FORTRAN/77 SOURCE PROGRAM      DATE 1993/
    
```

```

+----- VERSION 1.1 (1993.03.01) -----+
!
!      ###   ###   #   ###   ##   ##   ##   #####   ####   #####
!      ## #   ## #   #   ## #   ##   ##   ##   #   ##   #
!      ## #   ## #   #   ## #   ##   ##   ##   #####   ##   #####
!      ## #####   ##   #   ## #####   ##   ##   #   ##
!      ##   #   ##   ## ##   #   #####   ##   #####   ####   #####
!
+-----+
!
! SU(0) CO(0) RE(0) AR(0) IO(0) MA(0) TR(0) VA(0) CS(0) CP(0) C*(0) NE(
! SU(1) CO(1) RE(1) AR(1) IO(1) MA(1) TR(1) VA(1) CS(1) CP(1) C*(1) NE(
!                                     TR(2)          CS(2)          NE(
    
```

※ ANALYSISコマンド終了処理は、このBROWSE画面を閉じた後に行われる。

(9) (2)で2 ( BATCH ) を選択した場合も、(5)のようなメッセージが表示された後で、JCLが開かれる。JCLのデータセット名は、このコマンドが任意につける名前である。ここで、実際にSYSINからのオプションを指定する。

\* DO YOU READ "OPTION HELP" OF ANALYSIS/EX ? (Y/N) ==> N

```

EDIT --- JXXXX.@ANAEX.T094619.CNTL ----- COLUMNS 001 072
COMMAND ==>                               SCROLL ==> CUR
***** *****V10L30*****
000001 TWCEI(03 04 06 00 04) NOTIFY(JXXXX)
000002 // EXEC ANALYSIS,
000003 //      DSN='JXXXX.SAMPLE.FORT77',
000004 //      INC='JXXXX.SAMPLE.INC.FORT77'
000005 //FT06F001 DD DSN=JXXXX.SAMPLE.ANALIST,DISP=SHR,
000006 //      DCB=(RECFM=FBA,LRECL=137,BLKSIZE=19043)
000007 //SYSIN DD *
000008 LC( 62) ELM(+ )
000009 SU(0) AR(0) RE(0) IO(0)
000010 VA(0) CO(0) CS(0) CP(0)
000011 NE(0) C*(0)
000012 MA(0) TR(0) CL(0)
000013 ! STATIS ASTER NOINC DIST
000014 ! T=MAIN
000015 ! M=MATING CHARACTER
000016 !C=
000017 *** LOOK OPTION HELP DATASET.: 'JXXXX.ANA.HELP', ***
000018 /*
000019 //
***** *****
    
```

※ このEDIT画面の中でも「OPTION HELP」を見たい場合は、(6)の要領で

見ることができる。

上の例の FT06 のデータセットは既に存在するため、DISP は「SHR」指定であるが、存在しない場合は「(NEW, CATLG)」指定となる。また、出力クラスを指定した場合も、「SYSOUT=\*」のような指定で JCL が作成される。

```
000005 //FT06F001 DD DSN=JXXXX.SAMPLE.ANALIST,
000006 //      DISP=(NEW,CATLG),UNIT=TSSWK,SPACE=(TRK,(50,20),RLSE),
000007 //      DCB=(RECFM=FBA,LRECL=137,BLKSIZE=19043)
```

※ もし、UNIT 等を変更したい場合や、センターのカタプロのパラメータ値を変更したい場合は、ここで変更するとよい。変更内容は 4.1 を参照のこと。

(10) オプションの指定を終了したら、(7)の要領で JCL の画面を閉じると、そのジョブをサブミットするかどうかを聞かれてくるので、「Y」を入力すれば自動的にサブミットして ANALYSIS コマンドの終了処理に入る。

```
* DO YOU SUBMIT THIS ANALYSIS JOB ? (Y/N) ==> Y
```

◎ コマンドの補足

ANALYSIS コマンドは、表 5. に示す幾つかのパラメータを持っている。

表 5. ANALYSIS コマンドのパラメータ

パラメータ	用途	標準値
DSN	FT01: 被解析対象プログラム名	空白
INC	FT02: 被解析対象インクルード名	空白
OPT	FT05: オプション入力データセット名	空白
OUT	FT06: 出力データセット名 (クラスではない)	空白
SPC3	FT03のファイルのスペース量	'300'
SPC4	FT04のファイルのスペース量 (GEM解析時使用)	'100 50'
SPC6	FT06のファイルのスペース量	'50 20'

よって、コマンドに逐一答えるのが面倒な場合は、次のようにも指定できる。

```
READY
ANALYSIS DSN(SAMPLE.FORT77) INC(SAMPLE.INC.FORT77) SPC6('100 50') .....
```

このパラメータは指定したいもの、変更したいもののみでよい。パラメータの指定忘れや不備がある場合は、必ず(2)~(5)のようなメッセージを伴って再度聞いてくる。この指定は不備が無い限り、実行時のコマンドや JCL に必ず生かされる。

## 5 実行時の注意点

### (1) 被解析プログラムの文法レベルについて

被解析プログラムの文法レベルは、TSS実行する場合は『FORTRAN77』、  
BATCHで実行する場合は『FORTRAN77』か『FORTRAN77/EX』  
で正常翻訳できるものとする。但し、オプションのNOINCを指定した場合を除く。

『ANALYSIS/EX』の解析結果は、

プログラム名、コモンラベル名 : 前から数えて7文字目まで  
変数名、配列名 : 前から数えて10文字目まで

しか表示しない。

また、日本語名標(変数名等)はサポートしてない。

### (2) 被解析対象プログラムの規模について

『ANALYSIS/EX』では、解析可能な被解析対象プログラムの規模を拡張し  
た。実際は約18万行のプログラムまで実行確認済である。但し、ある特定の命令の使  
用頻度が多過ぎる等の特徴のあるプログラムの場合、冒頭で述べた18万行より少なく  
ても解析不可能な場合があり得るので注意されたい。その場合は、『ANALYSIS  
/EX』内部でエラーメッセージを出力して終了するようになっているので相談いただ  
きたい。以下に被解析プログラムの大きさを示す主なパラメータを示す。

・ サブルーチン数MAX = 1000  
・ CALL文数MAX = 6000  
・ COMMON数MAX = 1000  
・ 入出力文数MAX = 10000

### (3) プログラム中の英大文字および英小文字の取扱について

『FORTRAN77/EX』から、英小文字名標(プログラム名、コモン名、変数  
名、配列名、FORTRAN命令、等)が使用できるようになった。しかし実際に大型  
計算機上で翻訳すると、英大文字でも英小文字でも綴りが同じであれば、以下のよう  
に同じ変数名として認識される。

SubRoutine Program ==> SUBROUTINE PROGRAM  
Dimension Computing(Ncom) ==> DIMENSION COMPUTING(NCOM)

『ANALYSIS/EX』では、DISTオプション(3.2(6)参照)を指定しない  
限り、入力される被解析プログラムに英小文字が存在しても、全て英大文字に変換して  
読み込むようになっている。よって、解析結果も全て英大文字で出力される。

DISTオプションを指定した場合は、英大文字小文字に関係なく読み込まれ、綴り  
が同じでも別変数として扱う。よって、このオプションを使用した場合と使用しない場  
合では、解析結果に差が出るので注意されたい。

(4) 入力データについて

『ANALYSIS/EX』のオプションは前述のように、BATCHで動作させる場合はJCL上のSYSINから、TSSで動作させる場合はオプション・データセットから入力できるようになっている。入力方法の詳細は、3.1、3.2、3.3を参照していただきたい。

補助制御行は、従来版では必ず第1カラム目から始めなければならなかったが、『ANALYSIS/EX』では第64カラム目以下であれば、どこから始めてもよい。但し、1行に1個の補助制御行の制限は変わらない。

尚、BATCHで実行する際のオプション入力カードは、JCL上で

“『//SYSIN DD \*』から始めて、必ず『/\*』で閉じる”

ことを忘れてはならない。

例

```
//SYSIN DD * ←
SU(0), CO(1), TR(3), ELM(*)
AR(1)RE(0) VA(0) CS(1)          C*(0)
CP(0)
T=MAIN
                                T=SUB001
/* ←
```

(5) BATCHでの実行時のREGIONサイズとTWCEIの値について

『ANALYSIS/EX』の実行時のREGIONサイズは約5300KBである。よって、TWCEIのCの値は6、Eは0で充分であり、M780の最下位のBクラスで動作可能である。他のT、W、Iの値は、被解析プログラムの大きさとオプション制御行の種類と数によるので、それに応じた値が必要となる。5万行以上のプログラムをフルオプションで実行したりすると、CPU時間やI/O回数が多くかかり、上位のクラスでの実行を余儀なくされる。そういう場合はオプションを何回かに分けて実行するとよい。また、解析結果の出力行数が多くなる場合には、出力行数オーバーでジョブが異常終了するのを防ぐために、解析結果の機番(PT06F001)をデータセットに割り当てるのが望ましい。簡単なJCL例を以下に示す。

```
T(3) W(4) C(3) I(4) SRP
// EXEC ANALYSIS.
// DSN='J####.SAMPLE.FORT77'.INC='NULLFILE'
//PT06F001 DD DSN=J####.SAMPLE.ANALYST.DISP=(NEW,CATLG).UNIT=TSSWK.
// SPACE=(TRK,(30,10),RLSE),DCB=(RECFM=FBA,LRECL=137,BLKSIZE=19043)
//SYSIN DD *
ELM(*)
SU(1) CO(1) RE(1) AR(1) TR(3)
* IO(1) MA(1) VA(1) CS(2) CP(1)
! NE(IP) CL(1) C*(1) NOINC ASTER DIST
T=MAIN
! M=ABCDE
!C= X X X
/*
//
```

## (6) TSSでの実行時の注意点

4.2で述べたANALYSISコマンドを使用した時は、被解析プログラムと出力データセット以外は、全て一時データセットでありカタログもされていない。よって、ANALYSISコマンドが終了すると、それらの一時データセット群は全て削除されてしまう。

ANALYSISコマンドで1のTSSを選択した場合、実行に際しての必要最低条件は、①現在のセッションがリージョンサイズ3MB以上で開設され、かつ、②実行CPU時間が2分以内で終了できる事である。①：リージョンサイズが3MB未満の場合は、実行直前にリージョンサイズが足りないことにより異常終了(CODE 106C)する。②：一般ユーザが1度LOGONすると、そのセッションを開いている間のCPU時間のMAXは2分である。通常、セッションを開いている間に行われる全ての動作(任意のコマンドを使用する、PFDを開く、PFキーを押す、等の動作)は必ず、わずかながらCPUを使用する。勿論、ANALYSISコマンドを実行してから、『ANALYSIS/EX』のロードモジュールが呼び出されるまでにも、わずかなCPUは使用される。そのため、TSS上で『ANALYSIS/EX』を実行する場合は、CPUを2分間使用するような解析をするのは事実上不可能である。実際に2分を越えるとそのセッションはLOGOFFされてしまうので、もう一度LOGONし直して、今度はBATCHで動作させるとよい。

## (7) その他

『ANALYSIS/EX』は既に情報システムセンターの標準ツールとして登録済みであり、標準のカタログドプロシジャ『ANALYSIS』を用いるか、標準のコマンドプロシジャ『ANALYSIS』を用いれば、『ANALYSIS/EX』が起動されるようになっている。それでも、従来の『ANALYSIS-77』を使用したい場合、20万行を越えるような超巨大プログラムの解析を行いたい場合、また改良や要望等に関しても別途相談いただきたい。

本使用手引書は、『ANALYSIS/EX v1.2』並びに『ANALYSIS』コマンドに関しての使用法を解説した手引書である。よって、ここに述べられている機能は、『ANALYSIS/EX v1.2』より以前のバージョンには適用できないものもあるので、ご注意されたい。

尚、今まで述べた仕様は予告なく変更する場合がある。その場合は、新たな仕様手引書を情報システムセンター受付に置くので、それを参照していただきたい。



<<<<<< INPUT CARD LIST >>>>>>

1 LC(108) ELM(\*) SU(1)

=ANALYSIS/EX= STATIC ANALYZER OF THE FORTRAN77/EX SOURCE PROGRAM DATE 1993/03/15(MONDAY) TIME 21:38:11 PAGE 0002

```

(001) *****
** %CNTLN **
*****
SUBROUTINE %CNTLN(CREC ,NC ,ID ,NID ,IDEF ,IANS )
00000040 .....( 6 ARGUMENTS).....
00001320 SUBROUTINE SUBPROGRAM(%ITEM )
00001330
00001390 SUBROUTINE SUBPROGRAM(MAXCHK )
00001480 SUBROUTINE SUBPROGRAM(CALCMN )
00001570 FUNCTION SUBPROGRAM (INT )
00000040 .....( 3 ARGUMENTS).....
(002) *****
** %CVLCT **
*****
CALL TOKEN (CREC ,NC ,ISTK ,IDX ,MXTK% ,NID )
CALL MAXCHK(NCLCK% ,MXLC% ,%CVLCT> AREA OVER FLOW LCLCT %!)
CALL MAXCHK(CICHECK ,MXNC2% ,%CVLCT-01> AREA OVER FLOW NCM% ! )
CALL %PRTVR ( ID% ,NID ,IST ,IEN )
CALL %CNTLN(CREC ,NC ,ID*(1,IST) ,NNID ,LEN ,IANS )
CALL MAXCHK(CICHECK ,MXNC2% ,%CVLCT-02> AREA OVER FLOW NCM% ! )
CALL MAXCHK( IDX ,MXNC1% ,%CVLCT-03> AREA OVER FLOW LCM% ! )
SUBROUTINE %DIME(CREC ,NC )
00000050 .....( 2 ARGUMENTS).....
(003) *****
** %DIME **
*****
CALL TOKEN (CREC ,NC ,ITEK ,IDX ,MXTK% ,NID )
CALL %PRTVR ( ID% ,NID ,IST ,IEN )
CALL %CNTLN(CREC ,NC ,ID*(1,IST) ,NNID ,IDEF ,IANS )
CALL MAXCHK( NDM% ,MXDM% ,%TYSTM> AREA OVER FLOW MXDM% ! )
SUBROUTINE %IMPLI(CREC ,NC )
00000040 .....( 2 ARGUMENTS).....
(004) *****
** %IMPLI **
*****
CALL TOKEN (CREC ,NC ,IST ,IDX ,MXTK% ,NID )
CALL %LSPCY ( CREC ,NC ,ILST ,IANS ,NCHA )
CALL MAXCHK(NEXT% ,MXEX% ,%NEXT%> AREA OVER FLOW MXVN% ! )
CALL MAXCHK(NEXT% ,MXEX% ,%NEXT%> AREA OVER FLOW MXVN% ! )
SUBROUTINE %ITEM(CREC ,NC ,ID ,NID ,CANS ,LC ,IANS )
00000040 .....( 7 ARGUMENTS).....
(005) *****
** %ITEM **
*****
CALL MAXCHK(LCHK ,LC ,%ITEM-01 ***** OVER LC ! )
CALL CALCMN( CREC(ID(2,IC+1)) ,LL ,IANS ,IRT )
CALL CALCMN( CREC(ID(2,IC)) ,LL ,IANS ,IRT )
IRIG = INT(CANS)
CALL CALCMN( CREC(ID(2,1)) ,LL ,IANS ,IRT )
SUBROUTINE SUBPROGRAM(MAXCHK )
SUBROUTINE SUBPROGRAM(CALCMN )
FUNCTION SUBPROGRAM (INT )
SUBROUTINE SUBPROGRAM(CALCMN )

```

図2 SU(1) オプション出力例 (続く)

```

=ANALYSIS/EX= STATIC ANALYZER OF THE FORTRAN77/EX SOURCE PROGRAM      DATE 1993/03/15(MONDAY)      TIME 21:38:11      PAGE 0002
CALL CALCMN(CREC(ID(2,1)),LL,ANS,IRT)
ILFT = INT(ANS)
SUBROUTINE YLPCL
*****
** YLPCL
*****
(006) *****
** YLPCL
*****
CALL %SRTCL
IF ( MOD((IE-IS+1),5) .EQ. 0 ) THEN
  CALL PAGE(O)
ELSEIF ( MOD(LLX,5).EQ.O) .AND. (LL.GE.(NUMX-1)) ) THEN
  ELSEIF( MOD(LLX,5).EQ.O ) THEN
SUBROUTINE YLPCM(ISRT)
*****
** YLPCM
*****
(007) *****
** YLPCM
*****
CALL %TGSRT ( ISRT ,NCV% ,CVREF% ,IDXNO% )
CALL %TGSRT ( IX2 ,ICNT,CVREF*(1,I-ICNT),IDXNO% )
CALL %TGSRT ( IX2 ,ICNT,CVREF*(1,I-ICNT),IDXNO% )
IF ( MOD((IE-IS+1),5) .EQ. 0 ) THEN
  CALL PAGE(O)
IF ( MOD(LL,5).EQ.O) .OR. (LL.EQ.NUMX) ) THEN
SUBROUTINE %LSPCY(CREC ,NC ,IL ,IDEF ,NCHA )
*****
** %LSPCY
**
*****
IF ( INDEX(' ',CREC(I)) .EQ. 0 ) THEN
IF ( INDEX('0123456789',CREC(I)) .NE. 0 ) THEN
  IF ( INDEX('0123456789',CREC(I)) .EQ. 0 ) THEN
  ELSEIF ( LOGCHK .AND. ( INDEX(' ',CREC(I)).EQ.O) ) THEN
  CALL CALCMN(CREC(IS) ,LL ,ANS ,IRT )
  IDEF = INT(ANS)
CHARACTER * 31 FUNCTION %NMSET ( STRING , ICODE ,IDX )
*****
** %NMSET
**
*****
%NMSET(I:I)=STRING(J)
CHARACTER * 31 FUNCTION %NMST2(NTYP,ID )
*****
** %NMST2
**
*****
CALL TOKEN ( COMSAV<IS>,NXL , IIXIN ,CODE ,CODMAX ,NCOD)
SUBROUTINE %OPSET( IOPT,ELMCD,IRT,CARDRC,NCARD )
*****
** %OPSET
**
*****
IF ( INDEX(CREC,'') .NE. 0 ) GO TO 4100
ICK = %OPTCK(J,COPT)
IVOPT = %OPTV<ICK>,J,COPT,INEC ,INEP,IRT )

```

図2 SU(1) オプション出力例



```

<<<<<<< INPUT CARD LIST >>>>>>
-----1-----2-----3-----4-----5-----6-----7-----+-----8
1 LC(108) ELM(*) AR(1)
=ANALYSIS/EX= LISTINGS OF THE ACTUAL ARGUMENTS(LIBRARY ROUTINE) DATE 1993/03/15(MONDAY) TIME 21:39:18 PAGE 0002

ABS
=====
ANS = ABS( ARG(1) )
IF( ABS( ARG(1) ) .GT. 2.0**18*3.141592 ) GO TO 8000
IF( ABS( ARG(1) ) .GT. 2.0**18*3.141592 ) GO TO 8000
IF( ABS( ARG(1) ) .GT. 2.0**18*3.141592 ) GO TO 8000
IF( ABS( ARG(1) ) .GT. 2.0**18*3.141592 ) GO TO 8000
IF( ABS( ARG(1) ) .GT. 1.0 ) GO TO 8000
IF( ABS( ARG(1) ) .GT. 1.0 ) GO TO 8000
IF( ABS( ARG(1) ) .GE. 175.366 ) GO TO 8000
IF( ABS( ARG(1) ) .GE. 175.366 ) GO TO 8000
IABSN = ABS(N)

ACOS
=====
ANS = ACOS( ARG(1) )

AINT
=====
ANS = AINT( ARG(1) )

AMT
=====
ANS = AMT( ARG(1) )

ANINT
=====
ANS = ANINT( ARG(1) )

ASIN
=====
ANS = ASIN( ARG(1) )

ATAN
=====
ANS = ATAN( ARG(1) )

ATAN2
=====
ANS = ATAN2( ARG(1), ARG(2) )

CBRT
=====
ANS = CBRT( ARG(1) )

CHAR
=====
CAPTLF(I:I)=CHAR(F+64)
LARGE(I)=CHAR(F+64)

COS
=====
ANS = COS( ARG(1) )

COSH
=====
ANS = COSH( ARG(1) )

00000390 CALLING PROGRAM=INTRNC .ARGUMENTS( 1)
00001040 CALLING PROGRAM=INTRNC .ARGUMENTS( 1)
00001090 CALLING PROGRAM=INTRNC .ARGUMENTS( 1)
00001140 CALLING PROGRAM=INTRNC .ARGUMENTS( 1)
00001190 CALLING PROGRAM=INTRNC .ARGUMENTS( 1)
00001240 CALLING PROGRAM=INTRNC .ARGUMENTS( 1)
00001290 CALLING PROGRAM=INTRNC .ARGUMENTS( 1)
00001440 CALLING PROGRAM=INTRNC .ARGUMENTS( 1)
00001490 CALLING PROGRAM=INTRNC .ARGUMENTS( 1)
00000390 CALLING PROGRAM=PAGE .ARGUMENTS( 1)

00001300 CALLING PROGRAM=INTRNC .ARGUMENTS( 1)
00000310 CALLING PROGRAM=INTRNC .ARGUMENTS( 1)
00001810 CALLING PROGRAM=INTRNC .ARGUMENTS( 1)
00000350 CALLING PROGRAM=INTRNC .ARGUMENTS( 1)
00001250 CALLING PROGRAM=INTRNC .ARGUMENTS( 1)
00001340 CALLING PROGRAM=INTRNC .ARGUMENTS( 1)
00001400 CALLING PROGRAM=INTRNC .ARGUMENTS( 2)
00000700 CALLING PROGRAM=INTRNC .ARGUMENTS( 1)
00000803 CALLING PROGRAM=CAPTLF .ARGUMENTS( 1)
          CALLING PROGRAM=CAPTLL .ARGUMENTS( 1)
00001100 CALLING PROGRAM=INTRNC .ARGUMENTS( 1)
00001500 CALLING PROGRAM=INTRNC .ARGUMENTS( 1)

```

図3 AR(1) オプション出力例 (続く)

```

=ANALYSIS/EX= LISTINGS OF THE ACTUAL ARGUMENTS(LIBRARY ROUTINE)      DATE 1993/03/15(MONDAY)      TIME 21:39:18      PAGE 0002

COTAN
=====
ANS = COTAN( ARG(1) )
                                .ARGUMENTS( 1 )
DATE
=====
CALL DATE(TODAYX)
                                .ARGUMENTS( 1 )
ERF
=====
ANS = ERF( ARG(1) )
                                .ARGUMENTS( 1 )
ERFC
=====
ANS = ERFC( ARG(1) )
                                .ARGUMENTS( 1 )
ERRTRA
=====
CALL ERRTRA
                                .ARGUMENTS( 0 )
EXP
=====
ANS = EXP( ARG(1) )
                                .ARGUMENTS( 1 )
EXP10
=====
ANS = EXP10( ARG(1) )
                                .ARGUMENTS( 1 )
EXP2
=====
ANS = EXP2( ARG(1) )
                                .ARGUMENTS( 1 )
FBCTL
=====
CALL FBCTL (DDMEM(1:8),1,RC02)
CALL FBCTL (DDNAME,1,RC0)
                                .ARGUMENTS( 3 )
                                .ARGUMENTS( 3 )
FDCB
=====
CALL FDCB(DD1,DSN1,REC1,BLK1,RECFM1,DSORG1,RC01)
CALL FDCB (DD2,DSN2,REC2,BLK2,RECFM2,DSORG2,RC02)
                                .ARGUMENTS( 7 )
                                .ARGUMENTS( 7 )
FGETM
=====
CALL FGETM ( DDNAME, MEMBER, RC00 )
                                .ARGUMENTS( 3 )
FKSET
=====
CALL FKSET (DDNAME,K,RC0K)
CALL FKSET (DDNAME,K,RC0K)
                                .ARGUMENTS( 3 )
                                .ARGUMENTS( 3 )
FOPEN
=====
CALL FOPEN(DD1,1,2,RC01)
CALL FOPEN(DD2,1,2,RC02)
                                .ARGUMENTS( 4 )
                                .ARGUMENTS( 4 )
FREAD
=====
CALL FREAD(DDMEM,DECK,LEN,RC0)
CALL FREAD(DDNAME, MEMBER, DECK, LEN, RC0)
CALL FREAD(DDNAME, MEMBLK, DECK, LEN, RC01)
                                .ARGUMENTS( 4 )
                                .ARGUMENTS( 4 )
                                .ARGUMENTS( 4 )
GAMMA
=====
ANS = GAMMA( ARG(1) )
                                .ARGUMENTS( 1 )

```

図3 AR (1) オプション出力例

```

<<<<<<< INPUT CARD LIST >>>>>>
-----1-----2-----3-----4-----5-----6-----7-----8
1 LC(108) ELM(*) RE(1)

=ANALYSIS/EX= LISTINGS OF CALL & CALLED-BY REFERENCE DATE 1993/03/15(MONDAY) TIME 21:40:29 PAGE 0002

*****
***** E N T R Y *****
***** " ASICHA" IS AN ENTRY POINT NAME OF THE USER SUBPROGRAM " ASTNUM".
CALLED FROM.....TYPCHA
*****
2. *ENTRY* THE NAME " DQUATE" IS AN ENTRY POINT NAME OF THE USER SUBPROGRAM " APOSTR".
CALLED FROM.....CLASS
*****
3. *ENTRY* THE NAME " ERR131" IS AN ENTRY POINT NAME OF THE USER SUBPROGRAM " ERR232".
4. *ENTRY* THE NAME " INITO2" IS AN ENTRY POINT NAME OF THE USER SUBPROGRAM " INITO1".
CALLED FROM.....INITS
5. *ENTRY* THE NAME " INITO3" IS AN ENTRY POINT NAME OF THE USER SUBPROGRAM " INITO1".
CALLED FROM.....INITS
6. *ENTRY* THE NAME " INITO4" IS AN ENTRY POINT NAME OF THE USER SUBPROGRAM " INITO1".
CALLED FROM.....INITS
7. *ENTRY* THE NAME " IOREAD" IS AN ENTRY POINT NAME OF THE USER SUBPROGRAM " IOWRIT".
CALLED FROM.....ARGLST
LPARG
LPCALL
LPCCHAR
LPIO
8. *ENTRY* THE NAME " NESTCM" IS AN ENTRY POINT NAME OF THE USER SUBPROGRAM " NEST".
CALLED FROM.....LPNEST
9. *ENTRY* THE NAME " SEARCH" IS AN ENTRY POINT NAME OF THE USER SUBPROGRAM " LOOKUP".
CALLED FROM.....*RECK2
*REFCK
ASSIGN
CALL
DOCHK
EXTINT
IFCHK
IOCHK
REFCHK

*****
***** USER ROUTINE *****
*****

-----1-----2-----3-----4-----5-----6-----7-----8
1. *CNTLN ( USER SUBROUTINE ) 237 STEPS
CALLS ==> *ITEM CALLED BY ==> *CVLCT
MAXCHK *DIME
CALCMN *TYSTM
INT

```

図4 RE ( 1 ) オプション出力例 ( 続く )

=ANALYSIS/EX= LISTINGS OF CALL & CALLED-BY REFERENCE DATE 1993/03/15(MONDAY) TIME 21:40:29 PAGE 0002

```

-----
2. *CVLCT ( USER SUBROUTINE ) 294 STEPS
CALLS ==>> TOKEN CALLED BY ==>> CONTRL
      MAXCHK
      *PRTVR
      *CNTLN

-----
3. *DIME ( USER SUBROUTINE ) 191 STEPS
CALLS ==>> TOKEN CALLED BY ==>> DCLARE
      *PRTVR
      *CNTLN
      MAXCHK

-----
4. *IMPLI ( USER SUBROUTINE ) 226 STEPS
CALLS ==>> TOKEN CALLED BY ==>> DCLARE
      *LSPCY
      MAXCHK

-----
5. *ITEM ( USER SUBROUTINE ) 98 STEPS
CALLS ==>> MAXCHK CALLED BY ==>> *CNTLN
      CALCMN
      INT

-----
6. *LPCL ( USER SUBROUTINE ) 413 STEPS
CALLS ==>> *SRTCL CALLED BY ==>> ANAL77
      MOD
      PAGE

-----
7. *LPCM ( USER SUBROUTINE ) 349 STEPS
CALLS ==>> *TGSRT CALLED BY ==>> ANAL77
      MOD
      PAGE

-----
8. *LSPCY ( USER SUBROUTINE ) 166 STEPS
CALLS ==>> INDEX CALLED BY ==>> *IMPLI
      CALCMN *TYSTM
      INT
    
```

図4 RE(1) オブション出力例











ANALYSIS/EX= COMMON SPECIFICATION PATTERN DATE 1993/03/15(MONDAY) TIME 21:44:28 PAGE 0005

COMMON SPECIFICATION PATTERN	SUB-PROGRAM NAME
ALIAS COMMON/ALIAS/LO1(NLO1:ML11) COMMON/ALIAS/ALIAS(O:NALIAS)	INITS LPEDIT TREE
BUFFOO COMMON/BUFFOO/STRING(NSTRNG),BUFFER(NBF),DD1,DD2	LMST2 ARGCHK COMTAB ENTRY ERROR IFCHK IOCHK LPCOMM LPTREE MATCH SUBPRO
CMREF* COMMON/CMREF*/IO1(NIO1) COMMON/CMREF*/CVREF*(C,MXCV)*,NCV*,IDXNO*(IDXMAX)	LMST2 ANAL77 ARGCHK COMCHK ENTRY FRTLIB LPCALL LPTREE SETPRM STATIS TYPDCL
COMCHA COMMON/COMCHA/CO2(NCO2) COMMON/COMCHA/EXTNAM(EXTMAX),COMNAM(COMMAX),CNAME(CRFMAX),COMTYP(CVMAX), CLOCAL(CLMAX),MYSON(2,MYMAX),PRGNAM(SUBMAX),COMLAB(CVMAX),SLAVES(CALMAX), ROOTS(ROTMAX),VARNAM(VARMAX),ENTNAM(ENTMAX),ARGNAM(ARGMAX),ARGTYP(ARGMAX), COMVAR(CVMAX),ENTPAR(ENTMAX),ERROOTS(ERMAX),SYSTEM(SYSMAX),COMSAV(CSMAX), CXNAME(CRFMAX),IDXCOM(IDXMAX),SUBFUN(2,MAXARG),COMBUF(NCMB)	LMST2 ANAL77 ARGCHK COMCHK ENTRY GETNAM LPCALL LPNEST LPNEST STATIS SUBPRO
COMINT COMMON/COMINT/IO2(NIO2) COMMON/COMINT/CFID(3,CRFMAX),COMID(2,SUBMAX),COMION(2,COMMAX), SUBID(CRFMAX),SYSTYP(SYSMAX),CODE(3,CODMAX),IDXTAB(IDXMAX),MYID(3,MYMAX), ENTTYP(ENTMAX),ENTID(2,ENTMAX),PRGTYP(SUBMAX),PRGSTP(SUBMAX),IOID(3, IOMAX),CLLOC(2,CLMAX),CALLS(2,SUBMAX),SLVTYP(CALMAX),CALLED(2,SUBMAX), ARGID(3,MAXARG)	LMST2 ANAL77 ARGCHK COMCHK ENTRY GETNAM LPCALL LPNEST LPNEST STATIS SUBPRO

図6 CO(1)オプション出力例(続く)

=ANALYSIS/EX= COMMON SPECIFICATION PATTERN DATE 1993/03/15(MONDAY) TIME 21:44:28 PAGE 0006

COMMON SPECIFICATION PATTERN		SUB-PROGRAM NAME	
CTBL	COMMON/CTBL/CO3(NC03) COMMON/CTBL/FNAM(MAXFLD),INTR(MAXINT)	CALCOO	MNINIT
DCLBYT	COMMON/DCLBYT/CO4(NC04) COMMON/DCLBYT/CHANAM(CHAMAX),DCLNAM(DCLMAX),IMPNAM(IMPMAX),ZLOW(MAXPAR), ZHIGH(MAXPAR),STX,CHABYT(CHAMAX),IMPTYP(IMPMAX)	%LPCM CONTRL	%NMST2 LPBYTE TYPCHA COMCHK TYPDCL
DCLNUM	COMMON/DCLNUM/IO3(NI03) COMMON/DCLNUM/DCLTYP(2,DCLMAX),DCLLEN,CHALEN,IMPLEN	%LPCM CONTRL	%SETCV TYPCHA COMCHK TYPDCL
FILED	COMMON/FILED/R01(NR01),I04(NI04) COMMON/FILED/FDAT(MAXFLD),NEXT	CALCOO	MNINIT
HEADER	COMMON/HEADER/CO5(NC05) COMMON/HEADER/TOP	%LPCM ANAL77 CONTRL INCSF LPCALL LPCHAR LPNEST POSF77 PAGE SUBPRO	%RECK2 ARGLST ENTRY LPARG LPCOME LPCOMM LPTREE MATCH REFCHK SETPRM
LENGTH	COMMON/LENGTH/IO5(NI05) COMMON/LENGTH/ENTLEN,SUBLEN,COMLEN,CALLEN,ROTLEN,MYLEN,LENARG,CLEN,IOLEN, EXTLEN,CVLEN,ARGLEN,VARLEN,ERLEN,IDXLEN,COOLEN,ENDSTR,NOBUF,FIRST,REFER, NCALL,START	%LPCM ANAL77 CALL DOCHK ENTRY FRTLIB LPASSI LPNEST REFCHK TYPCHA	%RECK2 ARGLST COMTAB ERROR GETNAM IFCHK LPCOME LPTREE STATUS SUBPRO
MDFLN*	COMMON/MDFLN*/RLPRM*(MXPR*),LV*(MXVN*),LEXT*(MXEX*),LCM*(MXNC1*), NCM*(CO:MXNC2*),LCLCK*(MXLCK*),NUCID*(MXNU*),LDM*(MXDM*) COMMON/MDFLN*/R02(NR02),I06(NI06)	%DIM %SRTCL CONTRL	%PERAT %LPLC %TYSTM MNCALCMN MNCSTATS CALCOO

図6 CO(1) オプション出力例

```

<<<<<<< INPUT CARD LIST >>>>>>
-----1-----2-----3-----4-----5-----6-----7-----8
1  LC(108) ELM(*) VA(1)

=ANALYSIS/EX= STATIC ANALYZER OF THE FORTRAN77/EX SOURCE PROGRAM   DATE 1993/03/15(MONDAY)   TIME 21:42:59   PAGE 0002

(001) *****
      ** %CNTLN **
      *****
      SUBROUTINE %CNTLN(CREC ,NC ,ID ,NID ,IDEF ,IANS )
      00000040 .....( 6 ARGUMENTS).....

      .DUMMY ARGUMENT(CREC ) MAY BE PASSED TO ANOTHER SUBPROGRAM.
      .DUMMY ARGUMENT(IANS ) MAY BE ASSIGNED A VALUE.
      .DUMMY ARGUMENT(ID ) MAY BE PASSED TO ANOTHER SUBPROGRAM.
      .DUMMY ARGUMENT(NC ) MAY BE PASSED TO ANOTHER SUBPROGRAM.
      .COMMON VARIABLE(CITEM%) /WORKC% /) MAY BE ASSIGNED A VALUE AND PASSED TO ANOTHER SUBPROGRAM.

(002) *****
      ** %CVLCT **
      *****
      SUBROUTINE %CVLCT(CREC ,NC ,IDX )
      00000040 .....( 3 ARGUMENTS).....

      .DUMMY ARGUMENT(CREC ) MAY BE PASSED TO ANOTHER SUBPROGRAM.
      .DUMMY ARGUMENT(IDX ) MAY BE PASSED TO ANOTHER SUBPROGRAM.
      .DUMMY ARGUMENT(NC ) MAY BE PASSED TO ANOTHER SUBPROGRAM.
      .COMMON VARIABLE(CID%) /WORKD% /) MAY BE PASSED TO ANOTHER SUBPROGRAM.
      .COMMON VARIABLE(CLCLCK%) /MDFLN% /) MAY BE ASSIGNED A VALUE.
      .COMMON VARIABLE(CLCM%) /MDFLN% /) MAY BE ASSIGNED A VALUE.
      .COMMON VARIABLE(CNCLCK%) /MDFNM% /) MAY BE ASSIGNED A VALUE AND PASSED TO ANOTHER SUBPROGRAM.
      .COMMON VARIABLE(CNCM%) /MDFLN% /) MAY BE ASSIGNED A VALUE.
      .COMMON VARIABLE(CVCLCK%) /MDFVR% /) MAY BE ASSIGNED A VALUE.

(003) *****
      ** %DIME **
      *****
      SUBROUTINE %DIME(CREC ,NC )
      00000050 .....( 2 ARGUMENTS).....

      .DUMMY ARGUMENT(CREC ) MAY BE PASSED TO ANOTHER SUBPROGRAM.
      .DUMMY ARGUMENT(NC ) MAY BE PASSED TO ANOTHER SUBPROGRAM.
      .COMMON VARIABLE(CID%) /WORKD% /) MAY BE PASSED TO ANOTHER SUBPROGRAM.
      .COMMON VARIABLE(LDM%) /MDFLN% /) MAY BE ASSIGNED A VALUE.
      .COMMON VARIABLE(NDM%) /MDFNM% /) MAY BE ASSIGNED A VALUE AND PASSED TO ANOTHER SUBPROGRAM.
      .COMMON VARIABLE(VDM%) /MDFVR% /) MAY BE ASSIGNED A VALUE.

(004) *****
      ** %IMPLI **
      *****
      SUBROUTINE %IMPLI(CREC ,NC )
      00000040 .....( 2 ARGUMENTS).....

      .DUMMY ARGUMENT(CREC ) MAY BE PASSED TO ANOTHER SUBPROGRAM.
      .DUMMY ARGUMENT(NC ) MAY BE PASSED TO ANOTHER SUBPROGRAM.
      .COMMON VARIABLE(CID%) /WORKD% /) MAY BE PASSED TO ANOTHER SUBPROGRAM.
      .COMMON VARIABLE(LEXT%) /MDFLN% /) MAY BE ASSIGNED A VALUE.
      .COMMON VARIABLE(NEXT%) /MDFNM% /) MAY BE ASSIGNED A VALUE AND PASSED TO ANOTHER SUBPROGRAM.
    
```

図7 V A ( 1 ) オプシ ョ ン 出 力 例 ( 続 く )



<<<<<<<< INPUT CARD LIST >>>>>>

1 LC(108) ELM(\*) CS(1)

=ANALYSIS/EX= LISTINGS OF THE COMMON VARIABLES REFERENCE TABLE DATE 1993/03/15(MONDAY) TIME 21:45:48 PAGE 0002

V-NAME	LABEL-ID	D:DEFINE OR @:READ	R:REFERENCE	A:ARGUMENT	W:R+D	X:R+A	Y:D+A	Z:D+R+A
			:-WRITE	#:I/O STATEMENT	*:@#	X:#+	&:@+	Y:@+~#
CCDI	/BUFF00 -01/			A A A A A B B I	C C C C C C C C C C	C C C C C C C C C C	C C C C C C C C C C	C C C C C C C C C C
NVIMP	/BUFF00 -01/			A A A A A B B I	A A A A A A A A A A	A A A A A A A A A A	A A A A A A A A A A	A A A A A A A A A A
LCELM	/CMREF# -01/	-R-X	R R Z W	X I X D	X	R	A A	A A X
INDXNO	/CMREF# -01/	-A	D R	D				W
ARGNAM	/COMCHA -01/	R X		A				X
CLOCAL	/COMCHA -01/							D
CNAME	/COMCHA -01/							D
COMBUF	/COMCHA -01/							A
COMLAB	/COMCHA -01/		R				A	Z
COMNAM	/COMCHA -01/	X	R				A	A
COMSAV	/COMCHA -01/		-X				A	A
COMVAR	/COMCHA -01/		A A R					
IDXCOM	/COMCHA -01/	R						W
PRGNAM	/COMCHA -01/							R
SLAVES	/COMCHA -01/			Y				Y
SUBFUN	/COMCHA -01/			D I R A				A
VARNAM	/COMCHA -01/		A					D
ARGID	/COMINT -01/			D I R				W
CALLS	/COMINT -01/			W				R
CLLOC	/COMINT -01/			X				A
CODE	/COMINT -01/		R R R					A
COMID	/COMINT -01/							W
CREFID	/COMINT -01/		-X					W
IDXTAB	/COMINT -01/	R	R					D
PRGSTP	/COMINT -01/							R
SLVTYP	/COMINT -01/			D				R
FNAM	/CTBL -01/	A			D			W

図8 CS(1) オプション出力例 (続く)



<<<<<<<< INPUT CARD LIST >>>>>>>>

1 LC(108) ELM(\*) CP(1)

=ANALYSIS/EX= LISTINGS OF THE COMMON VARIABLES REFERENCE TABLE DATE 1993/03/15(MONDAY) TIME 21:47:40 PAGE 0002

	POSITION	LENGTH	D:DEFINE OR EQUIVALENCE @:READ	R:REFERENCE -:WRITE	A:ARGUMENT #:I/O STATEMENT	W:R+D *:@#	X:R+A X:#+-	Y:D+A Y:@+-	Z:D+R+A Z:@+-#
STRING	( COMMON / BUFFO	/ 6600)					X		R
BUFFER	( 6601-	8080)							
CVREF*	( COMMON / CMREF*	/ 96000)							
NCV*	( 96001-	4)							
IDXNO*	( 96005-	3200)							
COMNAM	( COMMON / COMCHA	/ 15500)							
CLOCAL	( 145301-	3100)							
PRGNAM	( 644401-	15500)							
COMLAB	( 659901-	27900)							
SLAVES	( 687801-	124000)							
VARNAM	( 1059801-	49600)							
ARGNAM	( 1118701-	12400)							
COMVAR	( 1132301-	27900)							
COMSAV	( 1250101-	8000)							
IDXCOM	( 1382101-	24800)							
SUBFUN	( 1406901-	49600)							
CREFLD	( COMMON / COMINT	/ 48000)							
COMID	( 48001-	4000)							
CODE	( 74401-	12000)							
IDXTAB	( 86401-	3200)							
CLLOC	( 277201-	800)							
CALLS	( 278001-	4000)							
SLVTYP	( 282001-	16000)							
ARGID	( 302001-	9600)							

図9 CP(1) オブション出力例 (続く)

=ANALYSIS/EX= LISTINGS OF THE COMMON VARIABLES REFERENCE TABLE DATE 1993/03/15(MONDAY) TIME 21:47:40 PAGE 0003

	POSITION	LENGTH	D:DEFINE OR EQUIVALENCE @:READ	R:REFERENCE --:WRITE	A:ARGUMENT #:I/O STATEMENT	W:R+D *:@+H	X:R+A *:#+-	Y:D+A &:@+-	Z:D+R+A *:@+H
FNAM	( COMMON / CTBL / 1-	6600)		A			D		
INTR	( COMMON / DCLBYT / 6601-	961)		A					
STX	( COMMON / DCLBYT / 8069-	6600)							2
FDAT	( COMMON / FILD / 1-	800)		R			D		
TOP	( COMMON / HEADER / 1-	50)							
SUBLEN	( COMMON / LENGTH / 5-	4)			R		R		
CALLN	( COMMON / LENGTH / 13-	4)					W		
LENARG	( COMMON / LENGTH / 25-	4)			Z				
CVLEN	( COMMON / LENGTH / 41-	4)		A					
ARGLEN	( COMMON / LENGTH / 45-	4)			X				A
VARLEN	( COMMON / LENGTH / 49-	4)		A			A		
IDXLEN	( COMMON / LENGTH / 57-	4)		R					
CODLEN	( COMMON / LENGTH / 61-	4)			X		X		
ENDSTR	( COMMON / LENGTH / 65-	4)			R		A		R
NOBUF	( COMMON / LENGTH / 69-	4)			R				
FIRST	( COMMON / LENGTH / 73-	4)							
NCALL	( COMMON / LENGTH / 81-	4)							
START	( COMMON / LENGTH / 85-	4)			D		A		
RLPRM*	( COMMON / MDFLN* / 1-	800)		Y					
LV*	( COMMON / MDFLN* / 801-	1200)			D				
LEXT*	( COMMON / MDFLN* / 2001-	80)							
LCM*	( COMMON / MDFLN* / 2081-	2000)			R				
NCM*	( COMMON / MDFLN* / 4081-	20004)			R				
LCLCK*	( COMMON / MDFLN* / 24085-	400)							
LDM*	( COMMON / MDFLN* / 25285-	1200)							
NPRM*	( COMMON / MDFLN* / 1-	4)		Z					
NVN*	( COMMON / MDFLN* / 5-	4)							
NEXT*	( COMMON / MDFLN* / 9-	4)			Z				
NCLCK*	( COMMON / MDFLN* / 13-	4)							
NDM*	( COMMON / MDFLN* / 21-	4)							

図9 CP (1) オブション出力例





=ANALYSIS/EX= SUBPROGRAM NEST STRUCTURES ( %CNTLN DATE 1993/03/15(MONDAY) TIME 21:50:18 PAGE 0003

```

C (BYTE) FOR COMMON /DCLBYT//DCLNUM/
PARAMETER ( CHAMAX = 200, DCLMAX = 200, MAXPAR=20 )
PARAMETER ( IMPMAX = 28 )
C
C (DIMENS) FOR COMMON /COMCHA//COMINT/
PARAMETER ( SUBMAX = 500, CRFMAX = 4000, COMMAX = 500
& ,CLMAX = 100, MAXCOD = 1003, CODMAX = MAXCOD-3
& ,MYMAX = 8000, IOMAX = 7000, CALMAX = 4000
& ,ROTMAX = 8000, VARMAX = 1600, ARGMAX = 400
& ,CVMAX = 900, ENTMAX = 300, ERMAX = 2000
& ,EXTMAX = 100, SYSMAX = 600, CSMAX = 8000
& ,IDXMAX = 800, MAXARG = 800 )
PARAMETER ( MXCV# = 6000, NCMB = 4000 )
C
C (MIX) FOR COMMON /MIXCHA//MIXINT/
PARAMETER ( NORGIN = 20, NSBSTR = 20, NSUBLE = 20
& ,NTPST = 66, NSTRID = 4000, NKEY = 7000 )
C
C (PO) FOR COMMON /POCHAR//PONUM /
PARAMETER ( MAXLH = 20 )
C
C (SETUP) FOR COMMON /SETUP //HEADER/
PARAMETER ( NCLASS = 68, NOPTMX = 50, NTPMAX = 50 )
C
C (WORK) FOR COMMON /WORKC//WORKD#/
PARAMETER ( MXTK# = 2000, MXIM# = 500 )
C
C ( ) FOR OTHER COMMON /ALIAS //SAMEXX/....
PARAMETER ( NALIAS = 11, NSAMEX = 400 )
*INCLUDE WORK
C WORK
C----- ADD 1990 01/01 ----- NEDAC -----
COMMON / WORKD# / ID*(3,MXTK#)
COMMON / WORKC# / CITEM#
CHARACTER * (MXIM#) CITEM#
C-----
C-
C- PARAMETER
C- MXTK# : MAXIMUM NUMBER OF ID#
C- ( = CODMAX : DEFINED AT DIMENS)
C-
C- / WORKD# / WORK DATA AREA (INTEGER)
C- ID# : SET RESULTS OF TO BE CALLED SUBROUTINE TOKEN
C- ( = CODE : DEFINED AT DIMENS)
C-
C- / WORKC# / WORK DATA AREA (CHARACTER*)
C- CITEM# : CHARACTER AREA
C-
C-----
CHARACTER * 1 CREC(NC)
REAL * 4 ANS
LOGICAL * 4 LOGDIM ,LOGLEN ,LOGSER ,LOGPAR
INTEGER IX(2,2) ,ID(3,NID) ,IA(2)
C
C>>>>>>> SERCH ASTER OR '(

```

図10 NE (1) オプション出力例 (続く)

=ANALYSIS/EX= SUBPROGRAM NEST STRUCTURES ( \*CNTLN DATE 1993/03/15(MONDAY) TIME 21:50:18 PAGE 0004

```

C
C DIMENSION IX(1,1) : DIMENSION AREA
C DIMENSION IX(1,2) : LENGTH AREA
C
LOGDIM = .FALSE.
LOGLEN = .FALSE.
LOGSER = .TRUE.
LOGPAR = .FALSE.
IX(1,1) = 0
IX(2,1) = 0
IX(1,2) = 0
IX(2,2) = 0
IP = 0
C
DO 4000 I=1,NID
C+++++
VAR / **** /
( LOGSER .AND. (ID(1,I).EQ.14.) ) THEN
GO TO 3500
C+++++
VAR (****)
ELSEIF ( LOGSER .AND. (ID(1,I).EQ.11) ) THEN
IX(1,1) = I
LOGDIM = .TRUE.
LOGSER = .FALSE.
IP = 1
C-----
IF ( ID(1,I+1) .EQ. 19 ) THEN
IANS = -999
GO TO 9999
ENDIF
C+++++
VAR(****)
ELSEIF ( LOGSER .AND. (ID(1,I).EQ.19) ) THEN
IX(1,2) = I + 1
LOGLEN = .TRUE.
LOGSER = .FALSE.
IP = 0
C-----
IF ( ID(1,I+2) .EQ. 19 ) THEN
IANS = -999
GO TO 9999
ENDIF
C+++++
SERCH '( ' OR ' )' FROM VAR(****) OR VAR( ** )
ELSEIF ( (LOGLEN).OR.(LOGDIM) .AND. (ID(1,I).EQ.11) ) THEN
IP = IP + 1
IF ( LOGLEN ) LOGPAR = .TRUE.
ELSEIF ( (LOGLEN).OR.(LOGDIM) .AND. (ID(1,I).EQ.12) ) THEN

```

図10 NE (1) オプション出力例 (続く)







STATISTICS INFORMATION	
ENTLEN =	9 ENTMAX = 500
SUBLEN =	125 SUBMAX = 1000
COMLEN =	26 COMMAX = 1000
CALLEN =	641 CALMAX = 6000
ROTLN =	242 ROTMAX = 10000
MYLEN =	1603 MYMAX = 30000
CSLEN =	760 CSMAX = 70000
IOLN =	410 IOMAX = 10000
EXTLEN =	0 EXTMAX = 100
CVLEN =	0 CVMAX = 900
ARGLEN =	4 ARGMAX = 500
VARLEN =	3 VARMAX = 1600
ERLEN =	21 ERMAX = 2000
IDXLN =	32 IDXMAX = 800
REFER =	42 CRFMAX = 4000
ARGLEN =	4 ARGMAX = 500
NCV* =	46 MXCV* = 13000
NPRM* =	0 MXPR* = 500
NVN* =	3 MXVN* = 1000
NEXT* =	1 MXEX* = 40
NUC* =	0 MXNU* = 400

(1)

(2)

※(1) 解析時に実際に使用した値  
 (2) 宣言しているパラメータの値

図12 STATIS オプション出力例







```

<<<<<<<< INPUT CARD LIST >>>>>>
-----1-----2-----3-----4-----5-----6-----7-----8
1 LC(108) ELM(*) TR(1)
2 T=MAIN
=ANALYSIS/EX= *** LISTINGS OF THE PROGRAM TREE *** DATE 1993/03/15(MONDAY) TIME 21:55:08 PAGE 0002

```

```

+--- TREE
|
| F O R Y O U R R E F E R E N C E
|
| * STANDS FOR THE LIBRARY ROUTINE.
| # STANDS FOR THE STATEMENT FUNCTION DEFINITION.
| X STANDS FOR THE ENTRY NAME.
| @ STANDS FOR THE EXTERNAL/INTRINSIC PROCEDURE NAME.
| === STANDS FOR THE ABBREVIATION OF TREE STRUCTURE.
+---

```

USER ROUTINE	XCMLN (SUBROUTINE)	XCVLCT (SUBROUTINE)	XTIME (SUBROUTINE)	XIMPLI (SUBROUTINE)	XITEM (SUBROUTINE)	(SUBROUTINE)	XLPCL (SUBROUTINE)	(SUBROUTINE)
=====	XLPCM (SUBROUTINE)	XLSPCY (SUBROUTINE)	XNMSET (SUBROUTINE)	XNMST2 (SUBROUTINE)	XOPSET (SUBROUTINE)	(SUBROUTINE)	XOPTCK (SUBROUTINE)	(SUBROUTINE)
	*OPTX (FUNCTION)	*PRAY (SUBROUTINE)	*PRMCT (SUBROUTINE)	*PRTRV (SUBROUTINE)	*RECK2 (SUBROUTINE)	(SUBROUTINE)	*REFCK (SUBROUTINE)	(SUBROUTINE)
	*SETCV (SUBROUTINE)	*SRTCL (SUBROUTINE)	*TGSRT (SUBROUTINE)	*TYSTM (SUBROUTINE)	ANAL77 (SUBROUTINE)	(SUBROUTINE)	APOSTR (SUBROUTINE)	(SUBROUTINE)
	ARGCHK (SUBROUTINE)	ARGLST (SUBROUTINE)	ASSIGN (SUBROUTINE)	ASTUM (SUBROUTINE)	BLKDAT (SUBROUTINE)	(SUBROUTINE)	BLNKSP (SUBROUTINE)	(SUBROUTINE)
	CALCMN (SUBROUTINE)	CALCOD (SUBROUTINE)	CALL (SUBROUTINE)	CAPLFL (FUNCTION)	CAPLTS (SUBROUTINE)	(FUNCTION)	CHCHK (FUNCTION)	(FUNCTION)
	CHKNUM (SUBROUTINE)	CLASS (SUBROUTINE)	COMCHK (SUBROUTINE)	COMPUT (SUBROUTINE)	COMTAB (SUBROUTINE)	(SUBROUTINE)	CONTRL (SUBROUTINE)	(SUBROUTINE)
	DECLARE (SUBROUTINE)	DECODE (SUBROUTINE)	DOCHK (SUBROUTINE)	DOSTNO (SUBROUTINE)	ELM (FUNCTION)	(FUNCTION)	ELMCHK (SUBROUTINE)	(SUBROUTINE)
	ENTRY (SUBROUTINE)	ERROR (SUBROUTINE)	ERR232 (SUBROUTINE)	EXCOMT (SUBROUTINE)	EXTCHK (SUBROUTINE)	(SUBROUTINE)	EXTINT (SUBROUTINE)	(SUBROUTINE)
	FNDSTR (SUBROUTINE)	FNOXXX (SUBROUTINE)	FRLIB (SUBROUTINE)	GENTAB (SUBROUTINE)	GETNAM (SUBROUTINE)	(SUBROUTINE)	GOTO (SUBROUTINE)	(SUBROUTINE)
	HOLLER (SUBROUTINE)	IDTABL (SUBROUTINE)	IFCHK (SUBROUTINE)	INCMEM (SUBROUTINE)	INCSF (SUBROUTINE)	(SUBROUTINE)	INITS (SUBROUTINE)	(SUBROUTINE)
	INITO1 (SUBROUTINE)	INTRNC (SUBROUTINE)	IOCHK (SUBROUTINE)	IOOPEN (SUBROUTINE)	IOWRIT (SUBROUTINE)	(SUBROUTINE)	KEYWRD (SUBROUTINE)	(SUBROUTINE)
	KOMP (FUNCTION)	LOOKC (SUBROUTINE)	LOOKF (SUBROUTINE)	LOOKUP (SUBROUTINE)	LPARG (SUBROUTINE)	(SUBROUTINE)	LPASSI (SUBROUTINE)	(SUBROUTINE)
	LPBLNK (SUBROUTINE)	LPBYTE (SUBROUTINE)	LPBALL (SUBROUTINE)	LPCHAR (SUBROUTINE)	LPCLAS (SUBROUTINE)	(SUBROUTINE)	LPCOME (SUBROUTINE)	(SUBROUTINE)
	LPCOMM (SUBROUTINE)	LPCOM2 (SUBROUTINE)	LPEDIT (SUBROUTINE)	LPIO (SUBROUTINE)	LPNEST (SUBROUTINE)	(SUBROUTINE)	LPFREE (SUBROUTINE)	(SUBROUTINE)
	MOVEX (SUBROUTINE)	MAIN (MAIN)	MATCH (SUBROUTINE)	MAXCHK (SUBROUTINE)	MNINIT (SUBROUTINE)	(SUBROUTINE)	MOVE (SUBROUTINE)	(SUBROUTINE)
	OPERAT (SUBROUTINE)	MOVEX6 (SUBROUTINE)	MOSSET (SUBROUTINE)	NEST (SUBROUTINE)	NOWTIM (SUBROUTINE)	(SUBROUTINE)	NUMBER (SUBROUTINE)	(SUBROUTINE)
	READSF (SUBROUTINE)	OPTCHK (FUNCTION)	OUTREF (SUBROUTINE)	PAGE (SUBROUTINE)	POSF77 (SUBROUTINE)	(SUBROUTINE)	READCD (SUBROUTINE)	(SUBROUTINE)
	SUBPRO (SUBROUTINE)	REFCHK (SUBROUTINE)	SEPRM (SUBROUTINE)	SETPRM (SUBROUTINE)	STATS (SUBROUTINE)	(SUBROUTINE)	SUBNAM (SUBROUTINE)	(SUBROUTINE)
	TREE (SUBROUTINE)	TABLE (SUBROUTINE)	TABLEI (SUBROUTINE)	TAGSRT (SUBROUTINE)	TODAY (SUBROUTINE)	(SUBROUTINE)	TOKEN (SUBROUTINE)	(SUBROUTINE)
		TYPCHA (SUBROUTINE)	TYPDCL (SUBROUTINE)	TYPIMP (SUBROUTINE)	YMDW (SUBROUTINE)	(SUBROUTINE)		

```

ENTRY NAME
=====
DQUATE ( MAIN ENTRY...APOSTR ) ASTCHA ( MAIN ENTRY...ASTNUM ) ERR131 ( MAIN ENTRY...ERR232 ) INITO2 ( MAIN ENTRY...INITO1 )
INITO3 ( MAIN ENTRY...INITO1 ) INITO4 ( MAIN ENTRY...INITO1 ) IOREAD ( MAIN ENTRY...IOWRIT ) SEARCH ( MAIN ENTRY...LOOKUP )
NESTCM ( MAIN ENTRY...NEST )

```

図14 TR (1) オプション出力例 (続く)

=ANALYSIS/EX= \*\* TREE STRUCTURE \*\* ENTRY POINT = MAIN

```

MAIN  ---INIT01
      +-%INIT02
      +-%INIT03
      +-%INIT04
      +---BLKDAT
      +---ANAL77
  
```

```

      +---SETPRM  ---READCD
      |          +---%OPSET  ---*INDEX
      |          |          +---%OPTV  +---*INDEX
      |          |          +---ELMCHK  ---*INDEX
      |          |          +---IOOPEN  ---*FDCB
      |          |          +---%FOPEN  +---*FOPEN
      |          |          +---PAGE    ---*DATE
      |          |          +---%YMDW  +---*MOD
      |          |          +---INDEX  +---*INDEX
      |          |          +---NOWTIM  ---*TIME
      |          |          +---ABS    +---*ABS
      |          |          +---LPBLNK
      |          |          +---INDEX
      |          |          +---%MOD
      |          +---PAGE    ---*DATE
      |          +---YMDW    ---*MOD
      |          +---INDEX  ---*INDEX
      |          +---NOWTIM  ---*TIME
      |          +---ABS    +---*ABS
      |          +---ERROR  ---*ERROR
      |          +---LEN    +---*LEN
      |          +---ERRTRA +---*ERRTRA
      |          +---FREAD  +---*FREAD
      |          +---FBCTL  +---*FBCTL
      |          +---FGETM  +---*FGETM
      |          +---FKSET  +---*FKSET
      |          +---ELM    +---*ELM
      |          +---FREAD  +---*FREAD
      |          +---FBCTL  +---*FBCTL
      |          +---INDEX  +---*INDEX
      |          +---MOD    +---*MOD
      |          +---INCMEM +---*INCMEM
      |          +---MOVE   +---*MOVE
      |          +---KEYWRD +---*KEYWRD
      |          +---CLASS  +---*CLASS
      |          |          +---CAPTLS ---*ICCHAR
      |          |          +---FNDSTR +---*FNDSTR
      |          |          +---APOSTR +---*APOSTR
      |          |          +---%DQUATE +---*DQUATE
      |          +---LPTYPE ---*LPTYPE
      |          +---PAGE    ---*PAGE
      |          +---YMDW    ---*YMDW
      |          +---INDEX  ---*INDEX
      |          +---NOWTIM  ---*NOWTIM
      |          +---ABS    +---*ABS
      |          +---SUBPRO  ---*SUBPRO
      |          +---MOD    +---*MOD
      |          +---SUBNAM  +---*SUBNAM
  
```

図14 TR (1) オプション出力例





THE CLASSIFICATION OF FORTRAN STATEMENTS		STATEMENTS	
COMMENT	7904( 40.33 %)	WAIT	0( 0.00 %)
CONTINUATION	1975( 10.08 %)	INQUIRE	0( 0.00 %)
ASSIGNMENT	2469( 12.60 %)	COMPLEX	0( 0.00 %)
ASSIGN	0( 0.00 %)	LOGICAL	52( 0.27 %)
GO TO	158( 0.81 %)	INTEGER	443( 2.26 %)
ASSIGNED GO TO	0( 0.00 %)	CHARACTER	548( 2.80 %)
COMPUTED GO TO	3( 0.02 %)	DOUBLE PRECISION	0( 0.00 %)
ARITHMETIC IF	1( 0.01 %)	REAL	22( 0.11 %)
LOGICAL IF	350( 1.79 %)	EQUIVALENCE	2( 0.01 %)
ASSIGNMENT	137	DATA	309( 1.58 %)
ASSIGN	0	NAMelist	0( 0.00 %)
GO TO	151	IMPLICIT	74( 0.38 %)
ASSIGNED GO TO	0	PARAMETER	917( 4.68 %)
COMPUTED GO TO	0	COMMON	439( 2.24 %)
ARITHMETIC IF	0	SAVE	0( 0.00 %)
OPEN	0	DIMENSION	10( 0.05 %)
CLOSE	0	EXTERNAL	0( 0.00 %)
READ	0	INTRINSIC	0( 0.00 %)
WRITE	0	DEFINE FILE	0( 0.00 %)
BACKSPACE	9	PROGRAM	1( 0.01 %)
ENDFILE	0	BLOCKDATA	0( 0.00 %)
IF (...) THEN	0	FUNCTION	9( 0.05 %)
DO UNTIL(...)	620( 3.16 %)	SUBROUTINE	115( 0.59 %)
DO WHILE(...)	264( 1.35 %)	ENTRY	9( 0.05 %)
ELSE IF	49( 0.25 %)	CALL	536( 2.73 %)
END IF	22( 0.11 %)	RETURN	134( 0.68 %)
FORMAT	253( 1.29 %)	STOP	21( 0.11 %)
OPEN	247( 1.26 %)	PAUSE	0( 0.00 %)
CLOSE	620( 3.16 %)	END	125( 0.64 %)
READ	97( 0.49 %)	CONTINUE	498( 2.54 %)
WRITE	1( 0.01 %)	DECODE	0( 0.00 %)
BACKSPACE	0( 0.00 %)	ENCODE	0( 0.00 %)
ENDFILE	14( 0.07 %)	DEBUG	0( 0.00 %)
PRINT	287( 1.46 %)	AT	0( 0.00 %)
PUNCH	0( 0.00 %)	DISPLAY	0( 0.00 %)
REWIND	0( 0.00 %)	INIT	0( 0.00 %)
FIND	0( 0.00 %)	TRACE	0( 0.00 %)
		-- ROGUE --	0( 0.00 %)
		NCHARACTER (JEF)	0( 0.00 %)

TOTAL STATEMENTS = 19600

図16 被解析プログラム統計情報出力例

```

/*-----
/* JAERI JAERI
/* JAERI FORTRAN PROGRAM STATIC ANALYZER JAERI
/* JAERI JAERI
/* JAERI = A N A L Y S I S / B X = (1992.12.21) JAERI
/* JAERI JAERI
/* JAERI FORTRAN SOURCE MAX = ABOUT 180000 STEP JAERI
/* JAERI JAERI
/*-----
//ANALYSIS PROC PNM=ANALYSIS.LM='J0000.TOOL',Q='LOAD',
// OB=19043,ORECFM=PBA,SPC3=300,SPC4='100.50',
// DSN=.OUT='SYSOUT=*'.BLK1=8000.INC=NULLFILE.
// UNIT=WK10
/*-----
//ANALYSIS EXEC PGM=&PNM
//SUBSYS DD SUBSYS=(VFCS,'SIZE=(00000K,00M)')
//STEPLIB DD DSN=&LM.&Q,DISP=SHR
/*-----
//SYSPRINT DD &OUT. * GEM3 PRINT FILE *
// DCB=(LRECL=137,BLKSIZE=&OB,RECFM=&ORECFM)
//FT06FOO1 DD &OUT. * ANALYSIS-77 PRINT FILE *
// DCB=(LRECL=137,BLKSIZE=&OB,RECFM=&ORECFM)
/*-----
//FT01FOO1 DD DSN=&DSN. * INPUT SOURCE PROGRAM(PS,PD,GEM)*
// DISP=SHR,LABEL=(...IN) * *
//FT02FOO1 DD DSN=&INC. * INPUT SOURCE PROGRAM(PO) *
// DISP=SHR,LABEL=(...IN) * *INCLUDE *
/*-----
//FT03FOO1 DD UNIT=&UNIT,DISP=NEW. * WORK SPACE FOR DIRECT ACCESS *
// SPACE=(6000,(&SPC3)). * FILE *
// DCB=(BLKSIZE=6000,RECFM=F)
/*-----
//FT04FOO1 DD UNIT=&UNIT,DISP=NEW. * THIS FILE IS USED IF INPUT *
// SPACE=(TRK,(&SPC4),RLSE). * SOURCE PROGRAM IS GEM. *
// DCB=(LRECL=80,BLKSIZE=&BLK1,RECFM=PB)
/*-----
//FT05FOO1 DD DDNAME=SYSIN

```

図17 センターのカタログド・プロシジャ

付録B

ANALYSIS/EX木構造図





\*\*\* LISTINGS OF THE PROGRAM TREE \*\*\* DATE 1993/05/11(TUESDAY) TIME 16:46:48 PAGE 0002

```

+--- TREE FOR YOUR REFERENCE
+
+
+ * STANDS FOR THE LIBRARY ROUTINE.
+ ! # STANDS FOR THE STATEMENT FUNCTION DEFINITION.
+ ! % STANDS FOR THE ENTRY NAME.
+ ! @ STANDS FOR THE EXTERNAL/INTRINSIC PROCEDURE NAME.
+ ! === STANDS FOR THE ABBREVIATION OF TREE STRUCTURE.
+

```

USER ROUTINE

```

=====
%CNTLN (SUBROUTINE) %CVLCT (SUBROUTINE) %DIME (SUBROUTINE) %IMPLI (SUBROUTINE) %ITEM (SUBROUTINE) %LPCL (SUBROUTINE)
%LPCM (SUBROUTINE) %LSPCY (SUBROUTINE) %NMSET (SUBROUTINE) %NMST2 (FUNCTION) %OPSET (SUBROUTINE) %OPTCK (FUNCTION)
%SETCV (SUBROUTINE) %PERAT (SUBROUTINE) %PRMCT (SUBROUTINE) %PRTVR (SUBROUTINE) %RECK2 (SUBROUTINE) %REFCK (SUBROUTINE)
%RGCHK (SUBROUTINE) %RTRCL (SUBROUTINE) %TGSRT (SUBROUTINE) %TYSTM (SUBROUTINE) %ANAL77 (SUBROUTINE) %APOSTR (SUBROUTINE)
%ALCMM (SUBROUTINE) %CALCOO (SUBROUTINE) %CALL (SUBROUTINE) %CAPTLF (FUNCTION) %BKDAT (SUBROUTINE) %BLKSP (SUBROUTINE)
%CHKNM (SUBROUTINE) %CLASS (SUBROUTINE) %COMCHK (SUBROUTINE) %COMPT (SUBROUTINE) %COMTAB (SUBROUTINE) %CHCHK (FUNCTION)
%DECLARE (SUBROUTINE) %DDCODE (SUBROUTINE) %ERR232 (SUBROUTINE) %DOCHK (SUBROUTINE) %DOSTNO (SUBROUTINE) %ELM (FUNCTION)
%ENTRY (SUBROUTINE) %ERROR (SUBROUTINE) %FNDXXX (SUBROUTINE) %FRTLIB (SUBROUTINE) %EXCOMT (SUBROUTINE) %EXTCHK (SUBROUTINE)
%HOLLER (SUBROUTINE) %IDTABL (SUBROUTINE) %INTRNC (SUBROUTINE) %IOCHK (SUBROUTINE) %IOOPEN (SUBROUTINE) %INCMM (SUBROUTINE)
%INITO1 (SUBROUTINE) %LOOKC (SUBROUTINE) %LOOKF (SUBROUTINE) %LPCALL (SUBROUTINE) %LPEDIT (SUBROUTINE) %LPIO (SUBROUTINE)
%KOMP (FUNCTION) %LPBYTE (SUBROUTINE) %LPBYTE (SUBROUTINE) %LPCOM2 (SUBROUTINE) %LPCOMM (SUBROUTINE) %LPCOMM (SUBROUTINE)
%LPBLNK (SUBROUTINE) %LPBLNK (SUBROUTINE) %MAIN (MAIN) %MATCH (SUBROUTINE) %MAXCHK (SUBROUTINE) %MCHK (SUBROUTINE)
%LPTYPE (SUBROUTINE) %MOVEX6 (SUBROUTINE) %MSGSET (SUBROUTINE) %NEST (SUBROUTINE) %NEXT (SUBROUTINE) %NMNIT (SUBROUTINE)
%OPERAT (SUBROUTINE) %OPTCHK (FUNCTION) %OUTREF (SUBROUTINE) %PAGE (SUBROUTINE) %PAGE (SUBROUTINE) %PAGES (SUBROUTINE)
%READSF (SUBROUTINE) %REFCHK (SUBROUTINE) %SETPRM (SUBROUTINE) %TAGSRT (SUBROUTINE) %TAGSRT (SUBROUTINE) %TAGSRT (SUBROUTINE)
%SUBPRO (SUBROUTINE) %TABLE (SUBROUTINE) %TABLE1 (SUBROUTINE) %TAGSRT (SUBROUTINE) %TAGSRT (SUBROUTINE) %TAGSRT (SUBROUTINE)
%TREE (SUBROUTINE) %TYPCHA (SUBROUTINE) %TYPDCL (SUBROUTINE) %TYPIMP (SUBROUTINE) %TYPIMP (SUBROUTINE) %TYPIMP (SUBROUTINE)
=====

```

ENTRY NAME

```

=====
DQUATE ( MAIN ENTRY.....APOSTR ) ASTCHA ( MAIN ENTRY.....ASTNUM ) ERR131 ( MAIN ENTRY.....ERR232 ) INITO2 ( MAIN ENTRY.....INITO1 )
INITO3 ( MAIN ENTRY.....INITO1 ) INITO4 ( MAIN ENTRY.....INITO1 ) IOREAD ( MAIN ENTRY.....IDWRIT ) SEARCH ( MAIN ENTRY.....LOOKUP )
NESTCM ( MAIN ENTRY.....NEST )
=====

```

FORTRAN LIBRARY ROUTINE(S)

```

=====
INT
REAL
ICHAR
CHAR
AINT
AMINT
ABS
MOD
MAX
MIN
LEN
INDEX
=====

```

SORT  
CBRT  
EXP  
EXP2  
EXP10  
LOG  
LOG10  
LOG2  
SIN  
COS  
TAN  
COTAN  
ASIN  
ACOS  
ATAN  
ATAN2  
SINH  
COSH  
TANH  
ERF  
ERFC  
GAMMA  
LGAMMA  
IRE  
AMT  
DATE  
TIME  
ERRTRA























## 付録C

# ANALYSISコマンド・プロシジャ

```

PROC 0   DSN() +
        INC() +
        OPT() +
        OUT() +
        LOAD('J0000.TOOL.LOAD(ANALYSML)') +
        SPC3('300') SPC4('100 50') SPC6('50 20')

/*
CONTROL NOLIST NOMSG NOCONLIST NOFLUSH NOPROMPT
/*
/*== MENU ==*/
CALL 'SYS9.GENUTY.LOAD(CLDISP)'
WRITE      === THIS COMMAND IS EXECUTED ONLY IN THE READY MODE !! ===
WRITE
WRITE
WRITE      A N A L Y S I S / E X << VER 1.2 >>
WRITE
WRITE      NO.   MODE   GRAMMAR LEVEL   PROGRAM SIZE   LOGON SIZE
WRITE      -----
WRITE      1.   TSS    FORTRAN77/EX   50,000 STEP    3MB OR MORE
WRITE      2.   BATCH  FORTRAN77/EX   180,000 STEP   ANY
WRITE
WRITE      ( 1/2. OTHER=EXIT ) ==>
WRITE
READ &JOBNUM
  IF &JOBNUM = 1 && &JOBNUM = 2 THEN EXIT
  IF &JOBNUM=1 THEN +
    WRITE   *** LOGON REGION SIZE MUST BE 3MB OR MORE !! ***
WRITE
/*
/*== ATTENTION ==*/
ATTN EXEC 'SYS9.CMDPROC(FREEALL)'
/*
/*== INITIALIZE ==*/
FREE ATTR(ANA1 ANA2 ANA5 ANA6)
FREE DD(FT06F001 SYSPRINT FT05F001 SYSIN)
ATTR ANA3 RECFM(F)           BLKSIZE(6000)
ATTR ANA4 RECFM(F B)        LRECL(80) BLKSIZE(8000) DSORG(PS)
ATTR ANA5 RECFM(F B)        LRECL(80) BLKSIZE(3120) DSORG(PS)
ATTR ANA6 RECFM(F B A)      LRECL(137) BLKSIZE(19043) DSORG(PS)
SET &CTIME = &SUBSTR(1:2,&SYSTIME)&SUBSTR(4:5,&SYSTIME)&SUBSTR(7:8,&SYSTIME)
SET &FLG2 = 0 /* --- FLG2 : 0=DUMMY 1=SHR          ---*/
SET &FLG5 = 0 /* --- FLG5 : 0=NEW 1=SHR           ---*/
SET &FLG6 = 0 /* --- FLG6 : 0=NEW 1=SHR 2=CLASS  ---*/
/*
/*== ANALYZED DATASET, INCLUDE, AND OUTPUT DATASET CHECK ==*/
%DSN1:+
  IF &DSN= THEN +
    DO
      WRITENR ANALYZED DSNAME ? ==>
      READ &DSN
      ALLOC DS(&DSN) SHR REU
      IF &LASTCC > 0 THEN +
        DO
          WRITE *** FILE ALLOCATION ERROR OCCURED << &SYSPREF..&DSN >> ***
          WRITE
          SET &DSN=
          GOTO %DSN1
        END
      END
    ELSE +
      DO
        ALLOC DS(&DSN) SHR REU

```

```

IF &LASTCC > 0 THEN +
DO
WRITE *** FILE ALLOCATION ERROR OCCORED << &SYSPREF..&DSN >> ***
WRITE
SET &DSN=
GOTO ¥DSN1
END
END
/*
¥INC1:+
IF &INC= THEN +
DO
WRITENR ANALYZED INCLUDE ? ==>
READ &INC
IF &INC= | &INC=NULLFILE | &INC=DUMMY THEN +
DO
SET &FLG2=0
GOTO ¥OUT1
END
ELSE SET &FLG2=1
ALLOC DS(&INC) SHR REU
IF &LASTCC > 0 THEN +
DO
WRITE *** FILE ALLOCATION ERROR OCCORED << &SYSPREF..&INC >> ***
WRITE
SET &INC=
GOTO ¥INC1
END
GOTO ¥OUT1
END
IF &INC_= THEN +
DO
IF &INC= | &INC=NULLFILE | &INC=DUMMY THEN +
DO
SET &FLG2=0
GOTO ¥OUT1
END
ELSE SET &FLG2=1
ALLOC DA(&INC) SHR REU
IF &LASTCC > 0 THEN +
DO
WRITE *** FILE ALLOCATION ERROR OCCORED << &SYSPREF..&INC >> ***
WRITE
SET &INC=
GOTO ¥INC1
END
END
/*
¥OUT1:+
IF &STR(&OUT)= THEN +
DO
WRITE SYSOUT CLASS OR NAME ?
WRITENR (*, C, F, G, H, ..., FILE) ==>
READ &OUT
IF &STR(&OUT)= THEN SET &OUT=&STR(+)
IF &LENGTH(&STR(&OUT))=1 THEN +
DO
WRITENR * SYSOUT CLASS IS '&STR(&OUT)'.-CLASS ? (Y/N) ==>
READ &ASK1
IF &ASK1=Y THEN SET &FLG6=2

```



```

ELSE +
DO
  SET &OUT=
  GOTO %OUT1
END
IF &STR(&OUT)=&STR(*) THEN SET &FLG6=1
END
IF &LENGTH(&STR(&OUT))-1 THEN +
DO
  IF &STR(&OUT)=&STR(FILE) | &STR(&OUT)=&STR(FILE-NAME) THEN +
  DO
    WRITENR SYSOUT FILE NAME ? ==>
    READ &OUT
  END
  ALLOC DS(&OUT) SHR REU
  IF &LASTCC > 0 THEN SET &FLG6=0
  ELSE SET &FLG6=1
END
GOTO %MKHELP
END
IF &STR(&OUT)-= THEN +
DO
  ALLOC DS(&OUT) SHR REU
  IF &LASTCC > 0 THEN SET &FLG6=0
  ELSE SET &FLG6=1
END
END
/*
/* MAKING THE "OPTION HELP FILE".
%MKHELP:+
SET &HELP=ANA.HELP
ALLOC DD(HELPDD) DS(&HELP) NE DEL +
  SP(1,1) TR US(ANA5) UNI(TSSWK) REU
OPENFILE HELPDD OUTPUT
DO
  SET &HELPDD=&STR(*****
  PUTFILE HELPDD
  SET &HELPDD=&STR(* ELM(* ! M1,M3-M5 ) ! MEMBER ELEMENT *)
  PUTFILE HELPDD
  SET &HELPDD=&STR(* SU( 0 ! 1 ) ! SUBPROGRAM CALL *)
  PUTFILE HELPDD
  SET &HELPDD=&STR(* AR( 0 ! 1 ) ! ARGUMENTS LIST *)
  PUTFILE HELPDD
  SET &HELPDD=&STR(* RE( 0 ! 1 ) ! REFERENCE TABLE *)
  PUTFILE HELPDD
  SET &HELPDD=&STR(* IO( 0 ! 1 ) ! I/O STATEMENTS *)
  PUTFILE HELPDD
  SET &HELPDD=&STR(* CO( 0 ! 1 ) ! COMMON REFERENCE TABLE *)
  PUTFILE HELPDD
  SET &HELPDD=&STR(* VA( 0 ! 1 ) ! ARGUMENT AND COMMON VALUE ASSIGNMENT *)
  PUTFILE HELPDD
  SET &HELPDD=&STR(* CS( 0 ! 1 ! 2 ) ! DETAIL REFERENCE TABLE OF COMMON *)
  PUTFILE HELPDD
  SET &HELPDD=&STR(* CP( 0 ! 1 ) ! POSITION AND LENGTH TABLE OF COMMON *)
  PUTFILE HELPDD
  SET &HELPDD=&STR(* NE( 0 ! 1 ! 1P ! 1C ! 1PC ) ! SOURCE WITH NEST-STRUCTURE *)
  PUTFILE HELPDD
  SET &HELPDD=&STR(* C*( 0 ! 1 ) ! COMMENT AND ASTERISK *)
  PUTFILE HELPDD
  SET &HELPDD=&STR(* STATIS ! STATISTICS OF ANALYSIS/EX *)
  PUTFILE HELPDD

```

```

SET &HELPDD=&STR(* LC( NNN ) :N>0          ! LINE COUNT          *)
  PUTFILE HELPDD
SET &HELPDD=&STR(* ASTER                    ! IGNORE ASTERISK LINE  *)
  PUTFILE HELPDD
SET &HELPDD=&STR(* NOINC                    ! IGNORE INCLUDE STATEMENT *)
  PUTFILE HELPDD
SET &HELPDD=&STR(* DIST                    ! DISTINCTION BETWEEN SMALL AND CAPITAL *)
  PUTFILE HELPDD
SET &HELPDD=&STR(* MA( 0 ! 1 )              ! MATCHING CHARACTER    *)
  PUTFILE HELPDD
SET &HELPDD=&STR(*M=MATCHING                ! FOR "MA" OPTION (M=:<20) *)
  PUTFILE HELPDD
SET &HELPDD=&STR(* TR( 0 ! 1 ! 2 ! 3 )      ! TREE STRUCTURE       *)
  PUTFILE HELPDD
SET &HELPDD=&STR(*T=ENTRY ROUTINE           ! FOR "TR" OPTION (T=:<20) *)
  PUTFILE HELPDD
SET &HELPDD=&STR(* CL( 0 ! 1 )              ! CLASSIFIED FORTRAN STATEMENT *)
  PUTFILE HELPDD
SET &HELPDD=&STR(*C=          ID            *)
  PUTFILE HELPDD
SET &HELPDD=&STR(*-----1-----2-----3-----4-----5-----6-----*)
  PUTFILE HELPDD
SET &HELPDD=&STR(* AAGACALIDDBEEFOCRWBEPFRFWICLDREDNIPCSDEIDPBFSECRSPECDEADIT!FOR *)
  PUTFILE HELPDD
SET &HELPDD=&STR(* SSOSOROF00LLNOPLERANRUEIANOONA0EQAAMAOAIXNERLUUNAETANOENETINR!"CL"*)
  PUTFILE HELPDD
SET &HELPDD=&STR(* SS SMIG   SSDRBOAICDINWNQMGTHUAUTMPRMVMTTFOONBTLTOUNCCB SIA!OPT *)
  PUTFILE HELPDD
SET &HELPDD=&STR(* IIT P  T UWEE MNSDTKFNCIDTUPIERBLIAELAMEEERIGCCRRLUPS TOOUP TC! *)
  PUTFILE HELPDD
SET &HELPDD=&STR(* NNOGG!IH NH  IA E E ITHN  ILCGAL L LIMO NRINRKTUY R E IDDG L E! *)
  PUTFILE HELPDD
SET &HELPDD=&STR(* M OOFFE TI IFT  SL  D REAECE I ICEN SNNEA IT N  NEE A ! *)
  PUTFILE HELPDD
SET &HELPDD=&STR(*          N IL F    PE    EXLRT V SIT  IAS MDOI    U  Y ! *)
  PUTFILE HELPDD
SET &HELPDD=&STR(*****)
  PUTFILE HELPDD
END
CLOSEFILE HELPDD
/*
%SELECT:+
IF &JOBNUM=1 THEN GOTO %TSS%
IF &JOBNUM=2 THEN GOTO %BATCH%
/*
/*****/
/*** TSS JOB ***/
/*****/
%TSS%:+
          ALLOC DD(FT01F001) DS(&DSN) SHR REU
IF &FLG2=0 THEN ALLOC DD(FT02F001) DUMMY REU
IF &FLG2=1 THEN ALLOC DD(FT02F001) DS(&INC) SHR REU
          ALLOC DD(FT03F001)          NE DEL +
          SP(6000 &SPC3) B(6000) US(ANA3) UNI(WK10) REU
          ALLOC DD(FT04F001)          NE DEL +
          SP(&SPC4) TR US(ANA4) UNI(WK10) REU
IF &FLG6=0 THEN ALLOC DD(FT06F001) DS(&STR(&OUT)) NE CAT +
          SP(&SPC6) TR REL US(ANA6) UNI(TSSWK) REU
IF &FLG6=1 THEN ALLOC DD(FT06F001) DS(&STR(&OUT)) SHR REU
IF &FLG6=2 THEN ALLOC DD(FT06F001) SY(&STR(&OUT)) SHR REU

```

```

/*
/*== MAKE ANALYSIS OPTION DATA ==*/
%OPT1:+
  IF &OPT= THEN +
  DO
    SET &OPT =&STR(@ANAEX.T&CTIME..INDAT)
    SET &FLG5 =0
  /*
/*== OPEN OPTION DATASET ==*/
  ALLOC DD(OPTDD) DS(&OPT) NE KEEP +
    SP(1.1) TR US(ANA5) UNI(TSSWK) REU
  OPENFILE OPTDD OUTPUT
  DO
    SET &OPTDD=&STR( LC( 62) ELM(*) )
    PUTFILE OPTDD
    SET &OPTDD=&STR( SU(O) AR(O) RE(O) IO(O))
    PUTFILE OPTDD
    SET &OPTDD=&STR( VA(O) CO(O) CS(O) CP(O))
    PUTFILE OPTDD
    SET &OPTDD=&STR( NE(O) C*(O) )
    PUTFILE OPTDD
    SET &OPTDD=&STR( MA(O) TR(O) CL(O) )
    PUTFILE OPTDD
    SET &OPTDD=&STR(! STATIS ASTER NOINC DIST )
    PUTFILE OPTDD
    SET &OPTDD=&STR(! T=MAIN )
    PUTFILE OPTDD
    SET &OPTDD=&STR(! M=MATING CHARACTER )
    PUTFILE OPTDD
    SET &OPTDD=&STR(!C= )
    PUTFILE OPTDD
    SET &OPTDD=&STR(*** LOOK OPTION HELP DATASET.: '&SYSPREF..ANA.HELP'. ***)
    PUTFILE OPTDD
  END
/*
/*== CLOSE OPTION DATASET ==*/
  CLOSFILE OPTDD
  ALLOC F(FT05F001) DA(&OPT) OLD DEL
  WRITE =====
  WRITE LET'S OPEN "OPTION INPUT FILE (SYSIN)"
  WRITE PLEASE, CHANGE ANALYSIS/EX INPUT OPTION !!
  WRITE =====
  WRITENR * DO YOU READ "OPTION HELP" OF ANALYSIS/EX ? (Y/N) ==>
  READ &HASK
  IF &HASK=Y THEN +
  DO
    CALL 'SYS9.GENUTY.LOAD(CLDISP)'
    LIST &HELP
  END
  PFDE &OPT E
  END
/*
  ELSE +
  DO
  %OPT2:+
  ALLOC DD(FT05F001) DS(&OPT) SHR REU
  FLG5=1
  IF &LASTCC > 0 THEN +
  DO
    WRITE *** FILE ALLOCATION ERROR OCCORED << &SYSPREF..&OPT >> ***

```



```

IF &STRPOINT(&STR( ), &SPC6)=0 THEN SET &SPC6N1=
ELSE SET &SPC6N1=&SUBSTR(1:&STRPOINT(&STR( ), &SPC6)-1, &SPC6)
SET &SPC6N2=&SUBSTR(&STRPOINT(&STR( ), &SPC6)+1:&LENGTH(&SPC6), &SPC6)
SET &SPC6 =&SPC6N1, &SPC6N2
IF &SPC6N1= THEN SET &SPC6=&SPC6N2
/*
SET &JCLDSN=&STR(@ANAEX. T&CTIME. .CNTL)
/*
/*== OPEN JCL DATASET ==*/
ALLOC DD(JCLDD) DS(&JCLDSN) NE DEL +
      SP(1,1) TR US(ANA5) UNI(TSSWK) REU
OPENFILE JCLDD OUTPUT
DO
  SET &JCLDD=&STR( TWCEI(03 04 06 00 04) NOTIFY(&SYSPREF) )
  PUTFILE JCLDD
IF &SPC3=300 && &SPC4=&STR(100,50) THEN +
DO
  SET &JCLDD=&STR(// EXEC ANALYSIS, )
  PUTFILE JCLDD
END
ELSE +
DO
  SET &JCLDD=&STR(// EXEC ANALYSIS, SPC3=' &SPC3', SPC4=' &SPC4',
  PUTFILE JCLDD
END
SET &JCLDD=&STR(//      DSN=' &SYSPREF. .&DSN'. )
  PUTFILE JCLDD
/*
IF &FLG2=0 THEN +
DO
  SET &JCLDD=&STR(//      INC=NULLFILE)
  PUTFILE JCLDD
END
ELSE +
DO
  SET &JCLDD=&STR(//      INC=' &SYSPREF. .&INC' )
  PUTFILE JCLDD
END
/*
IF &FLG6=0 THEN +
DO
  SET &JCLDD=&STR(//FT06F001 DD DSN=&SYSPREF. .&OUT, )
  PUTFILE JCLDD
SET &JCLDD=&STR(//      DISP=(NEW, CATLG), UNIT=TSSWK, SPACE=(TRK, (&SPC6), RLSE), )
  PUTFILE JCLDD
SET &JCLDD=&STR(//      DCB=(RECFM=FBA, LRECL=137, BLKSIZE=19043)
  PUTFILE JCLDD
END
IF &FLG6=1 && &STR(&OUT)=-&STR(*) THEN +
DO
  SET &JCLDD=&STR(//FT06F001 DD DSN=&SYSPREF. .&OUT, DISP=SHR, )
  PUTFILE JCLDD
SET &JCLDD=&STR(//      DCB=(RECFM=FBA, LRECL=137, BLKSIZE=19043)
  PUTFILE JCLDD
GOTO NEXT2
END
IF &FLG6=2 | &STR(&OUT)=-&STR(*) THEN +
DO
  SET &JCLDD=&STR(//FT06F001 DD SYSOUT=&STR(&OUT))
  PUTFILE JCLDD

```

```

      GOTO NEXT2
    END
NEXT2:+
    SET &JCLDD=&STR(//SYSIN DD *)
    PUTFILE JCLDD
    IF &OPT_=- THEN GOTO $CLOSE
    SET &JCLDD=&STR( LC( 62) ELM(*) )
    PUTFILE JCLDD
    SET &JCLDD=&STR( SU(O) AR(O) RE(O) IO(O) )
    PUTFILE JCLDD
    SET &JCLDD=&STR( VA(O) CO(O) CS(O) CP(O) )
    PUTFILE JCLDD
    SET &JCLDD=&STR( NE(O) C*(O) )
    PUTFILE JCLDD
    SET &JCLDD=&STR( MA(O) TR(O) CL(O) )
    PUTFILE JCLDD
    SET &JCLDD=&STR(! STATIS ASTER NOINC DIST )
    PUTFILE JCLDD
    SET &JCLDD=&STR(! T=MAIN )
    PUTFILE JCLDD
    SET &JCLDD=&STR(! M=MATING CHARACTER )
    PUTFILE JCLDD
    SET &JCLDD=&STR(IC= )
    PUTFILE JCLDD
    SET &JCLDD=&STR(*** LOOK OPTION HELP DATASET.: '&SYSPREF..ANA.HELP'. ***)
    PUTFILE JCLDD
    SET &JCLDD=&STR(/* )
    PUTFILE JCLDD
    SET &JCLDD=&STR(// )
    PUTFILE JCLDD
  END
/*
/*== CLOSE JCL DATASET ==*/
CLOSEFILE JCLDD
IF &OPT_=- THEN +
DO
MERGE &OPT &JCLDSN
SET &JCLLST=&STR(@ANAEX.T&TIME..LAST)
ALLOC DD(LSTDD) DS(&JCLLST) NE DEL +
      SP(1,1) TR US(ANA5) UNI(TSSWK) REU
OPENFILE LSTDD OUTPUT
SET &LSTDD=&STR(*** LOOK OPTION HELP DATASET.: '&SYSPREF..ANA.HELP'. ***)
PUTFILE LSTDD
SET &LSTDD=&STR(/* )
PUTFILE LSTDD
SET &LSTDD=&STR(// )
PUTFILE LSTDD
CLOSEFILE LSTDD
MERGE &JCLLST &JCLDSN
END
/*
/*== EDIT JCL DATASET ==*/
PFDE &JCLDSN E
WRITENR * DO YOU SUBMIT THIS ANALYSIS JOB ? (Y/N) ==>
/*
CONTROL MSG
READ &SUB
IF &SUB=Y THEN +
DO
SUBMIT '&SYSPREF..&JCLDSN'

```

```
END
/*
/*== FREE ALL ALLOCATE DATASETS ==*/
¥FREEALL:+
LIB
EXEC 'SYS9.CMDPROC(FREEALL)'
/*
EXIT
```

付録D

ANALYSIS カタログド・プロシジャ



```

/*-----
/* JAERI JAERI
/* JAERI FORTRAN PROGRAM STATIC ANALYZER JAERI
/* JAERI JAERI
/* JAERI = A N A L Y S I S / E X = (1992.12.21) JAERI
/* JAERI JAERI
/* JAERI FORTRAN SOURCE MAX = ABOUT 180000 STEP JAERI
/* JAERI JAERI
/*-----
//ANALYSIS PROC PNM=ANALYSIS, LM='J0000.TOOL', Q='LOAD',
// OB=19043, ORECFM=FBA, SPC3=300, SPC4='100,50',
// DSN=, OUT='SYSOUT=*', BLK4=8000, INC=NULLFILE,
// UNIT=WK10
/*-----
//ANALYSIS EXEC PGM=&PNM
//SUBSYS DD SUBSYS=(VPCS, 'SIZE=(00000K,00M)')
//STEPLIB DD DSN=&LM, &Q, DISP=SHR
/*-----
//SYSPRINT DD &OUT, * GEM3 PRINT FILE *
// DCB=(LRECL=137, BLKSIZE=&OB, RECFM=&ORECFM)
//FT06F001 DD &OUT, * ANALYSIS-77 PRINT FILE *
// DCB=(LRECL=137, BLKSIZE=&OB, RECFM=&ORECFM)
/*-----
//FT01F001 DD DSN=&DSN, *INPUT SOURCE PROGRAM(PS, PO, GEM)*
// DISP=SHR, LABEL=(, , IN) *
//FT02F001 DD DSN=&INC, *INPUT SOURCE PROGRAM(PO) *
// DISP=SHR, LABEL=(, , IN) * *INCLUDE *
/*-----
//FT03F001 DD UNIT=&UNIT, DISP=NEW, * WORK SPACE FOR DIRECT ACCESS *
// SPACE=(6000, (&SPC3)), * FILE *
// DCB=(BLKSIZE=6000, RECFM=F)
/*-----
//FT04F001 DD UNIT=&UNIT, DISP=NEW, * THIS FILE IS USED, IF INPUT *
// SPACE=(TRK, (&SPC4), RLSE), * SOURCE PROGRAM IS GEM. *
// DCB=(LRECL=80, BLKSIZE=&BLK4, RECFM=FB)
/*-----
//FT05F001 DD DDNAME=SYSIN

```