

JAERI - M
86-186

軽水炉安全性評価コードRELAP5の変換支援ツール
——CDC版からFACOM版へ——

1987年1月

篠沢 尚久*・藤崎 正英*・牧野 光弘*
近藤 一也*・石黒美佐子

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。
入手の間合わせは、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、お申しこしてください。なお、このほかに財団法人原子力弘済会資料センター（〒319-11 茨城県那珂郡東海村日本原子力研究所内）で複写による実費頒布をおこなっております。

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, Tokai-
mura, Naka-gun, Ibaraki-ken 319-11, Japan.

©Japan Atomic Energy Research Institute, 1987

編集兼発行 日本原子力研究所
印刷 いばらき印刷機

軽水炉安全性評価コード RELAP 5 の変換支援ツール
—— CDC 版から FACOM 版へ ——

日本原子力研究所東海研究所計算センター
篠沢尚久*・藤崎正英*・牧野光弘*
近藤一也*・石黒美佐子

(1986年12月12日受理)

軽水炉安全性解析コード RELAP 5 は、米国アイダホ国立研究所 (INEL) で、CDC-CYBER 176 計算機用に開発された。INEL では、RELAP 5 コードを機能拡張やエラー修正のため、頻繁にバージョンアップしている。日本原子力研究所では、このコードを CDC 版から FACOM 版に変換して使用している。この変換作業は、コードの規模が大きいこと及び CDC と FACOM 間の計算機のハードウェア及びソフトウェアの違いがあるため多大な時間を費やしてきた。そこで、この変換作業を支援するためのソフトウェアツールを開発した。このツールによって、RELAP 5 コードの変換作業の効率が改善された。変換作業の生産性は、手作業で行っていたときと比べて、2.0~2.6倍である。ツールを使用したときの変換作業の手続きと各ツールのオプション・パラメータについても記述される。

Conversion Tool for the LWR Transient Analysis Code RELAP5
from the CDC Version to the FACOM Version

Naohisa SHINOZAWA*, Masahide FUJISAKI*, Mitsuhiro MAKINO*
Kazuya KONDOU* and Misako ISHIGURO

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

(Received December 12, 1986)

The LWR transient analysis code RELAP5 has been developed on the CDC-CYBER 176 at Idaho National Engineering Laboratory (INEL), the RELAP5 code has been often updated in order to extend the analyzing model and correct the errors. At Japan Atomic Energy Research Institute the code has been converted from the CDC version to the FACOM version and the converted code has been used. The conversion is the task which consumes a lot of time, because the code is large and there is the difference between CDC's machines and FACOM's ones. In order to convert the RELAP5 code automatically, the software tool has been developed. By using this tools the efficiency for converting the RELAP5 code has been improved. Productivity of the conversion is increased about 2.0 to 2.6 times by the tools in comparison with in manual. The procedure of conversion by using the tools and the option parameters of each tool are described.

Keywords: RELAP5, Conversion, Software Tool, CDC, FACOM

*) on leave from FUJITSU Ltd.

目 次

1. はじめに	1
2. 従来の方法と問題点	3
2.1 抽出作業	4
2.2 比較作業	7
2.3 コンバージョン作業	9
2.4 アップデート作業	9
2.5 補正作業	9
3. 開発ツールの機能	10
3.1 抽出作業 —— SELECTX	11
3.2 比較作業 —— FORTCOMP	12
3.3 コンバージョン作業 —— STREAM 77	13
3.4 アップデート作業 —— SEQNUM, UPDATE	15
3.5 手作業による補正作業	17
3.6 ビット操作関数の最適化 —— FREDUCT	17
4. ツール使用による効果	19
5. おわりに	21
謝 辞	21
参考文献	21
付録A SELECTX	22
付録B SEQNUM	43
付録C FORTCOMP	55
付録D UPDATE	70
付録E FREDUCT	83
付録F STREAM 77	103

CONTENTS

1. Introduction	1
2. Problems in previous conversion	3
2.1 Extraction	4
2.2 Comparison	7
2.3 Conversion	9
2.4 Update	9
2.5 Revision	9
3. The procedure of conversion by using the tool	10
3.1 Extraction — SELECTX	11
3.2 Comparison — FORTCOMP	12
3.3 Conversion — STREAM77	13
3.4 Update — SEQNUM,UPDATE	15
3.5 Revision in manual	17
3.6 Reduction of bit operation functions — FREDUCT	17
4. The effect by using the tool	19
5. Concluding remarks	21
Acknowledgements	21
References	21
Appendix A SELECTX	22
Appendix B SEQNUM	43
Appendix C FORTCOMP	55
Appendix D UPDATE	70
Appendix E FREDUCT	83
Appendix F STREAM77	103

1. はじめに

RELAP 5 コード¹⁾ は、米国アイダホ国立研究所 (INEL) で、CDC-CYBER 176 計算機用に開発されたコードである。このコードは、加圧水型原子炉 (PWR) システムの過渡解析のためのコードであり、大・中・小破断時の冷却材喪失事故、プラント制御、燃料系の事故などの軽水炉安全性に関する各種の PWR システムのシミュレーションに使用されている。

RELAP 5 コードを日本原子力研究所 (原研) に導入して使用するためには、開発元と原研の計算機システムが異なるために以下に述べることを考慮する必要がある。

RELAP 5 コードは、非常に大規模なコードである。最新版の RELAP 5 / MOD 2 / C 36.02 の場合には、実行ステートメント約 6 万ステップ、サブルーチン数 262 個である。また、様々な計算機システムで実行できるように、各計算機システムに対する全プログラムを保存、管理すると、ファイル量が増えるだけでなくメンテナンスが複雑になる。そこで、RELAP 5 では一つのファイルに計算機ごと、オペレーティングシステムごとの非互換な部分を重複して持ち、使用時に CDC 計算機用のユーティリティを用いて各計算機システムに合ったプログラムを選択するという一元管理手法を用いている。このため、RELAP 5 を導入する時は一元化されたプログラムのなかから使用者の計算機及びオペレーティングシステムに合ったプログラムを抽出する必要がある。原研では、FACOM-M シリーズの計算機を使用しているため、CDC 計算機用のプログラム管理ユーティリティを直接使用できない。このため、このユーティリティの機能を FACOM-M シリーズの計算機上で模擬するユーティリティが必要である。

また、原研では、開発元で最も良く使用され、エラーの少ないと思われる CDC 版 RELAP 5 コードを抽出して、FACOM-M シリーズ版 RELAP 5 コードへ変換して使用している。この変換作業を行うためには、RELAP 5 のプログラムとデータ構造及び CDC 計算機と FACOM 計算機のハードウェアの違い及び FORTRAN の文法の違いを熟知していることが必要である。²⁾ さらに、INEL において、CDC 版に対して、その機能拡張が MOD 0, MOD 1, MOD 2 と順次なされ、また機能拡張がされた各 MOD に対するエラー修正が CYCLE 1, CYCLE 2, CYCLE 3, …… (このエラー修正を CYCLE-UP と呼ぶ) という形でなされている。このため、これらの修正のたびに、CDC 版から FACOM 版への変換が必要になっている。また計算機の処理時間が長いから、変換後のプログラムの妥当性を調べるのにも長い時間が必要である。

原研における RELAP 5 コードの導入の手順をまとめると、

1. 抽出：複数の計算機、オペレーティングシステムに対応する一元化されたプログラムから、使用者の実行環境に合ったプログラムを抽出する。
2. 比較：新・旧 CDC 版ソースを比較し、異なる部分を取り出し編集する。
3. コンバージョン：2. で取り出した部分について、CDC 版から FACOM 版に変換する。
4. アップデート：3. で変換されたカードを旧 FACOM 版に挿入・削除し、新 FACOM 版を作成する。
5. 補正：4. で作成した新 FACOM 版においてサブルーチン間の関係などを考慮にいたった大

域的な変換（CDC版からFACOM版）をする。

という5つのステップからなっている。

従来の作業では、ソフトウェアツールを一部のステップで使用していただけであった。また、これらのソフトウェアツールは、汎用的なものであったり、ほかのプログラムに対して使用する目的で開発されたものであったため、手作業による補助作業が必要であった。この補助作業には、かなりの時間がかかるのと同時に不注意によるミスが入ることも多かった。また、このミスは最終段階が終わって実行テストを行って初めて判明するため、その原因調査にも不必要な時間を要していた。

これらの問題点を解決するために一連の作業を機械的に行うツール（SELECTX, FORTCOMP, SEQNUM, UPDATE）を開発し、コンバージョンには既存のSTREAM 77を使用して変換作業をシステム化した。第1ステップ—抽出ではSELECTX, 第2ステップ—比較ではFORTCOMP, 第3ステップ—コンバージョンではSTREAM 77, 第4ステップ—アップデートではSEQNUM, UPDATEというツールを使うことによって第1ステップから第4ステップまでは、手作業による補助作業なしに自動的に行うことができるようになった。また、これらの一連の作業には関係ないが、第1ステップから第5ステップまで終了した後、機械的操作によって生じたビット操作ファンクションのオーバーヘッドの最適化を行い、プログラムの処理速度を向上させるツールFREDUCTも開発した。

これらの一連のツールを用いることによって、主に次のような3つの効果が得られた。第1に従来の方法に比べて、生産性（1人月で変換できるアップデートカードの枚数）が向上した。実際には、変換作業経験のある者の場合で2.6倍、変換作業経験のない者の場合で2.0倍向上した。第2に手作業による変換ミスを減少させ、作業の信頼性が向上した。第3に、従来は、作業の5つのステップのすべてにおいて手作業が入っていたため、どのステップでどのような誤りが起きているかを見つけるのが困難であったがツールを使用した場合では、ほとんどの場合、手作業による補正ステップだけのチェックで済ませることができるようになった。

本報告書の構成は、第2章に従来の変換作業の方法と問題点をまとめ、第3章でその問題点を解消するために開発したツールについて述べる。第4章にツールを使用した場合の作業効率の改善とその他の効果について述べる。最後に付録として、今回開発したツールの使用方法を詳述する。

2. 従来の方法と問題点

従来、FACOM 版 RELAP 5 コードの変換作業は、Fig.2.1 に示すように第 1 ステップー抽出、第 2 ステップー比較、第 3 ステップーコンバージョン、第 4 ステップーアップデート、第 5 ステップー補正という 5 つのステップで行われていた。この章では、各ステップにおける作業の内容および問題点について述べる。

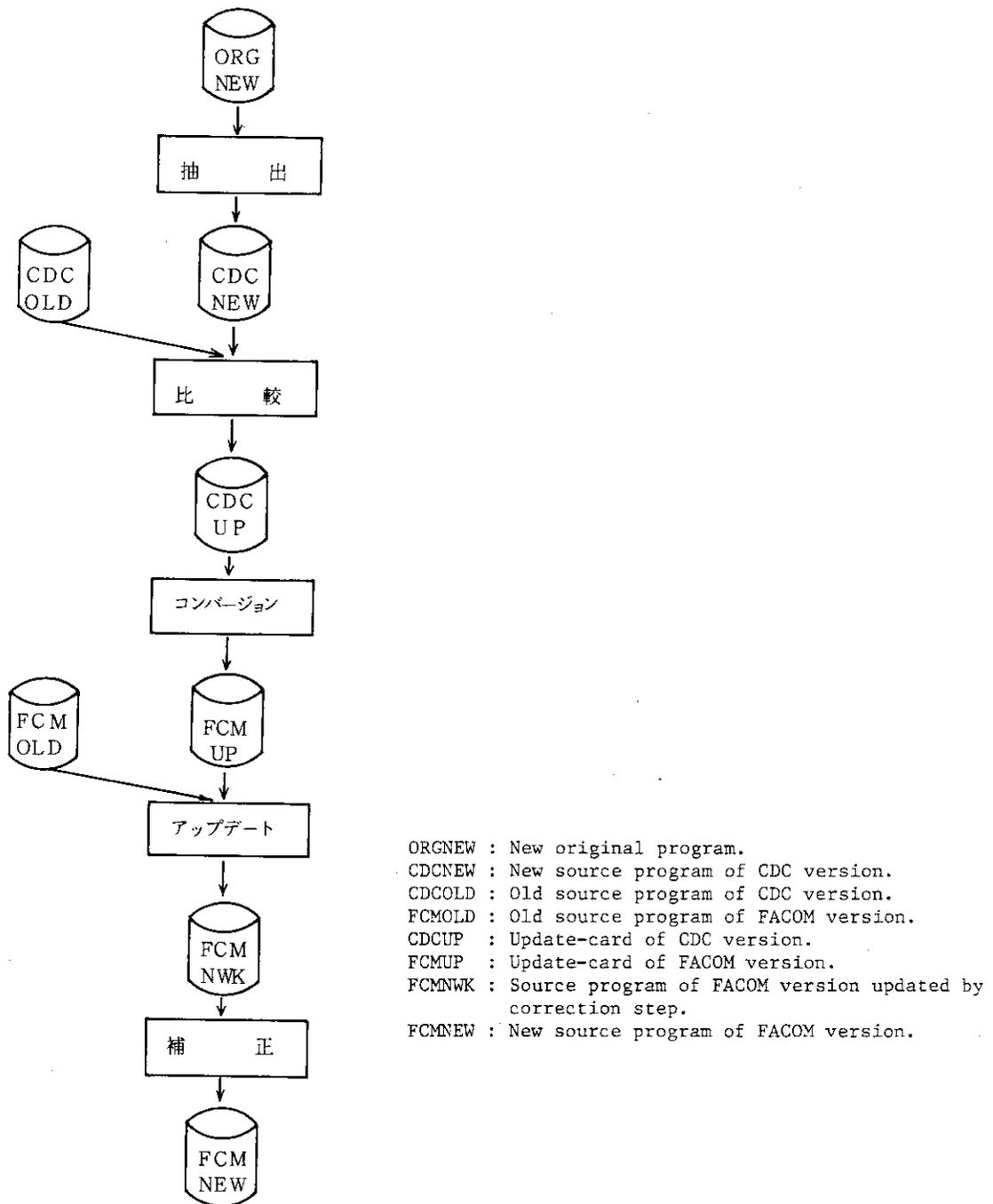


Fig. 2.1 General flow chart for conversion of RELAP5 in manual.

2.1 抽出作業

通常、計算機ハードウェアやオペレーティング・システムには、それぞれ非互換な点がある。あるプログラムをいくつかの計算機で使用するためには、その非互換な点に対応したソース・プログラムをそれぞれ作成する必要がある。各ソース・プログラムを独立したプログラムとして別々のファイルに保管すると、ファイル容量が増えるだけでなく、共通部分のプログラム修正が、あるプログラムには反映されないというような誤りがおこりやすくなり、プログラムの管理という点から問題がある。これらのことを避けるために、RELAP5コードでは、ソース・プログラム中にオプションとして非互換の部分を適宜選択して使用することができるようになっている。Fig.2.2にA、Bという2種類の計算機またはオペレーティング・システムに対するソース・プログラム保持の例を示す。OPTION A、Bに関するソースは、それぞれ計算機システムA、Bに依存したソース・プログラムを示す。共通部分は、計算機システムに依存しない部分のソース・プログラムである。利用者は、使用する計算機システムによって、OPTION Aと共通部分または、OPTION Bと共通部分を抽出して使用する。RELAP5/MOD2のオリジナル・プログラム内には、実際に選択すべきオプションを示す文及びステートメント単位の展開あるいは削除を指示するために*で始まる制御文(*DEFINE, *IF, *ENDIF)と¥で始まる制御文(¥DEFINE, ¥IF, ¥ENDIF)の2種類の制御文が混在している。前者の制御文は、CDCのUPDATEユーティリティ³⁾のものであり、後者の制御文は、RELAP5シリーズのSELECTXユーティリティのものである。この2つのユーティリティによって、選択したオプションの環境でコンパイル可能なソース・プログラムを抽出することができる。

FACOM上では、CDCのUPDATEユーティリティも、RELAP5シリーズのSELECTXユーティリティも使用できないため、この2つのユーティリティを模擬するプログラムPRPR1、PRPR3を作成して抽出をしていた。²⁾この抽出手順をFig.2.3に示す。まず、第1に前処理では、PRPR1が処理できるような形にオリジナル・プログラムをならべかえ、指定するオプションを示す文をソース・プログラムの先頭に付け加える。なお前処理は手作業で行う。第2にPRPR1によって、*IF制御文で指定された部分を抽出し中間ファイル(WORK)に出力する。第3にPRPR3によって、WORKから¥IF制御文で指定された部分を抽出し、変換対象になるソース・プログラム(SF)とインクルードソース・プログラム(INC)を作成するようになっている。実際には、CDC版のソース・プログラムとインクルードソース・プログラムを抽出している。

問題点としては、第1に操作手続きが3ステップ(前処理、*IF制御文の展開・削除、¥IF制御文の展開・削除)必要であった。第2に前処理ステップは手作業で行われるため誤りを犯す可能性があった。第3に*IF制御文の展開・削除や¥IF制御文の展開・削除を行うプログラムは、制御文が入れ子構造になっているとき正確に作動しなかった。

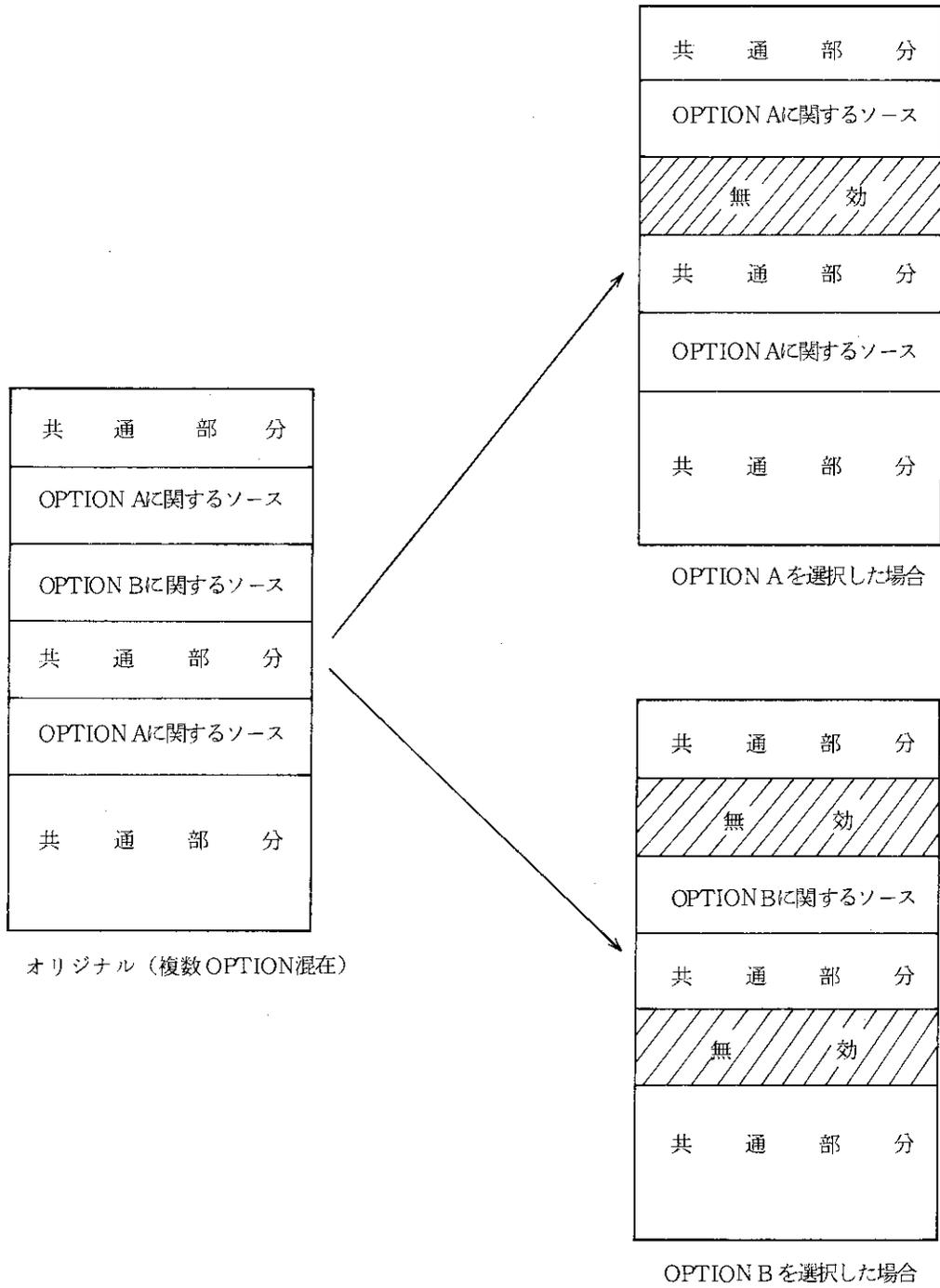


Fig. 2.2 An example of original source program including source programs correspond to some options and of source program for option A and option B.

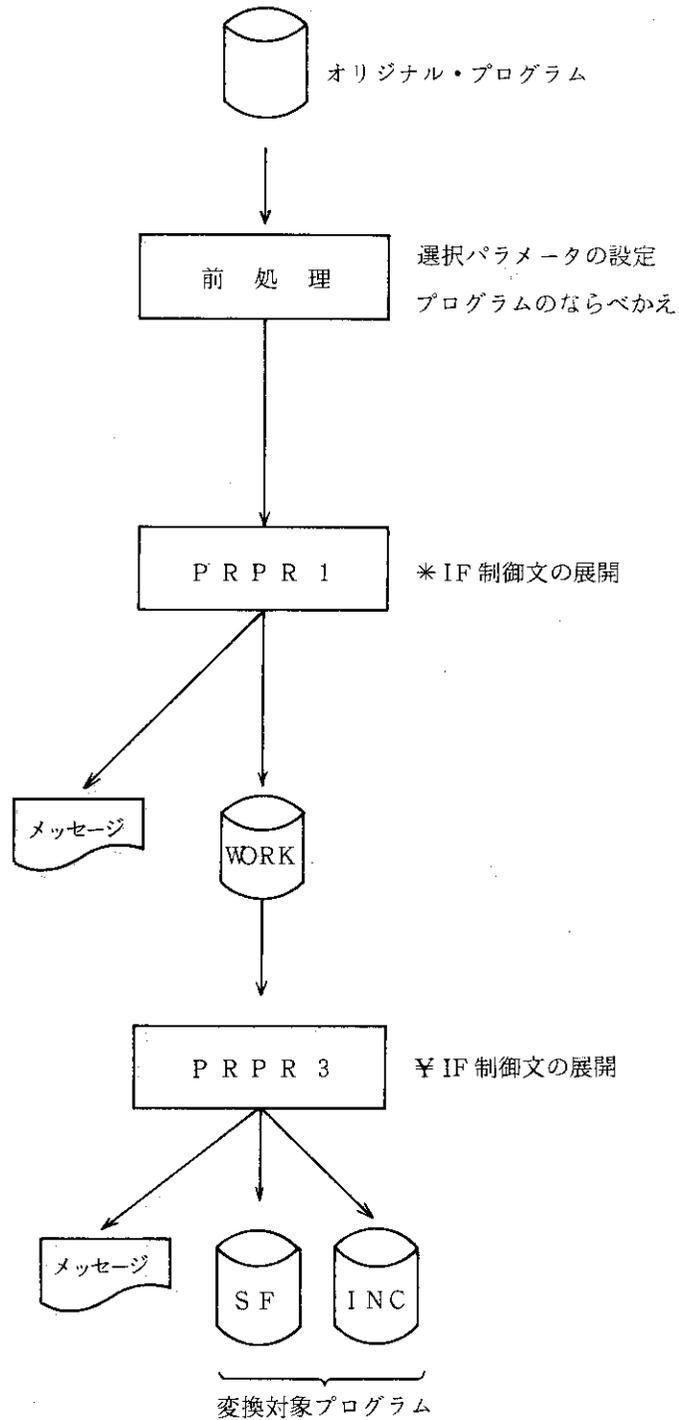


Fig. 2.3 Flow chart for extraction of source program by PRPR1 and PRPR3.

2.2 比較作業

この作業は、新・旧 CDC 版ソース・プログラムを比較して、修正（追加・削除）部分を抜き出しアップデートカードを作成する作業で、従来は主に原研ツール SFCOMP を使用して比較²⁾を行ってきた。アップデートカードとは、プログラムの修正変更を行うための指示（挿入・削除）及び修正データの集合をいう。修正前のソース（旧版 CDC ソース）に対してアップデートカードの指示に従って挿入・削除を行えば、修正後のソース（新版 CDC ソース）が完成するようになっている。Fig.2.4 に示すように新版 CDC ソース（SF 1）、旧版 CDC ソース（SF 2）、制御データ（CONTROL）を SFCOMP に入力し、比較結果（COMP）を出力する。つぎに、編集ステップで、この比較結果に手作業で修正位置（シーケンス番号）をつけ、アップデートカードを作成する。Fig.2.5 には、SFCOMP のジョブ制御文（JCL）を示す。SF 1 を FT 01 Fnnn, SF 2 を FT 02 Fnnn, CONTROL を FT 05 F 001, COMP を FT 06 F 001 に割り当てる。ただし、SF 1, SF 2 は順編成ファイル（PS ファイル）のみ取り扱う。また、SF 1, SF 2 の組を複数指定することは可能である。

問題点として、まず SFCOMP は、RELAP 5 専用のツールとして作成されたわけではないので区分編成ファイル（PO ファイル）を考慮に入れていない。このため、RELAP 5 のように通常 1 メンバ 1 サブルーチンの PO ファイルとして管理しているプログラムを入力として SFCOMP を実行する場合、各メンバごとにファイル名を指定しなければならない。これは、Fig.2.5 のようにサブルーチン数が多いときにはファイル名の指定が複雑になる。さらに、SFCOMP では比較対象ファイル間の違いのみ出力される。この出力ファイルをもとにして、修正位置（シーケンス番号）のついたアップデートカードを手作業で作成していたため作業時間が長く、シーケンス番号などを誤る可能性があった。

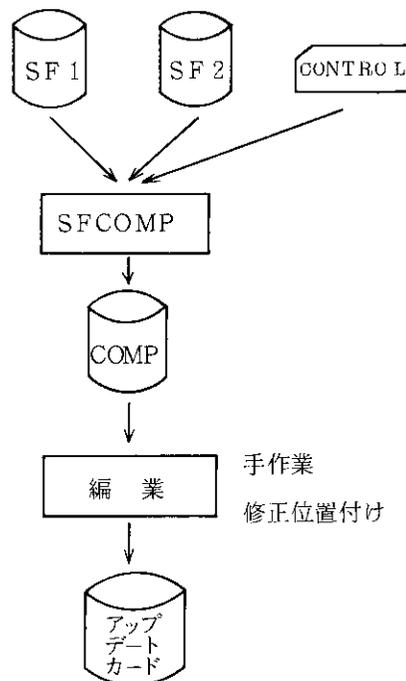


Fig. 2.4 Flow chart for comparison by SFCOMP.

```

/**
/**          *****
/**          ***** SFCOMP *****
/**          *****
/**
//SFCOMP EXEC PGM=SFCOMP
//STEPLIB DD DSN=J████.TOOL.LOAD,DISP=SHR
//FT06F001 DD SYSOUT=*
//FT05F001 DD *
13 18
/*
//FT01F001 DD DSN=J████.NEW.FORT77(DTLIST),DISP=SHR,LABEL=(,IN)
//FT02F001 DD DSN=J████.OLD.FORT77(DTLIST),DISP=SHR,LABEL=(,IN)
//FT01F002 DD DSN=J████.NEW.FORT77(ERROR),DISP=SHR,LABEL=(,IN)
//FT02F002 DD DSN=J████.OLD.FORT77(ERROR),DISP=SHR,LABEL=(,IN)
//FT01F003 DD DSN=J████.NEW.FORT77(GRAPH),DISP=SHR,LABEL=(,IN)
//FT02F003 DD DSN=J████.OLD.FORT77(GRAPH),DISP=SHR,LABEL=(,IN)
//FT01F004 DD DSN=J████.NEW.FORT77(IHEAT),DISP=SHR,LABEL=(,IN)
//FT02F004 DD DSN=J████.OLD.FORT77(IHEAT),DISP=SHR,LABEL=(,IN)
//FT01F005 DD DSN=J████.NEW.FORT77(IJTRAN),DISP=SHR,LABEL=(,IN)
//FT02F005 DD DSN=J████.OLD.FORT77(IJTRAN),DISP=SHR,LABEL=(,IN)
//FT01F006 DD DSN=J████.NEW.FORT77(INP1),DISP=SHR,LABEL=(,IN)
//FT02F006 DD DSN=J████.OLD.FORT77(INP1),DISP=SHR,LABEL=(,IN)
//FT01F007 DD DSN=J████.NEW.FORT77(INP2),DISP=SHR,LABEL=(,IN)
//FT02F007 DD DSN=J████.OLD.FORT77(INP2),DISP=SHR,LABEL=(,IN)
//FT01F008 DD DSN=J████.NEW.FORT77(INP3),DISP=SHR,LABEL=(,IN)
//FT02F008 DD DSN=J████.OLD.FORT77(INP3),DISP=SHR,LABEL=(,IN)
//FT01F009 DD DSN=J████.NEW.FORT77(INP4),DISP=SHR,LABEL=(,IN)
//FT02F009 DD DSN=J████.OLD.FORT77(INP4),DISP=SHR,LABEL=(,IN)
//FT01F010 DD DSN=J████.NEW.FORT77(INP6),DISP=SHR,LABEL=(,IN)
//FT02F010 DD DSN=J████.OLD.FORT77(INP6),DISP=SHR,LABEL=(,IN)
//FT01F011 DD DSN=J████.NEW.FORT77(INP7),DISP=SHR,LABEL=(,IN)
//FT02F011 DD DSN=J████.OLD.FORT77(INP7),DISP=SHR,LABEL=(,IN)
//FT01F012 DD DSN=J████.NEW.FORT77(PLDATA),DISP=SHR,LABEL=(,IN)
//FT02F012 DD DSN=J████.OLD.FORT77(PLDATA),DISP=SHR,LABEL=(,IN)
//FT01F013 DD DSN=J████.NEW.FORT77(RODSUB),DISP=SHR,LABEL=(,IN)
//FT02F013 DD DSN=J████.OLD.FORT77(RODSUB),DISP=SHR,LABEL=(,IN)
++

```

Fig. 2.5 An example of JCL for SFCOMP.

2.3 コンバージョン作業

FORTRAN ステートメント単位で CDC 版ソース（アップデートカード）から FACOM 版ソース（アップデートカード）に変換する。

変換内容としては、主に次の3つがあげられる。

- ① CDC と FACOM-M シリーズの 1 語の長さ（ビット数）の違いから、実数型変数の倍精度化が必要になる。

（例） REAL VAR ⇔ REAL*8 VAR

CDC の実数型変数は60ビットであるので FACOM の倍精度実数64ビットで模擬する。

- ② CDC と FACOM-M シリーズの間に FORTRAN 文法の違いがあるため、型宣言文、変数名の長さ及び文字の違いの変換が必要になる。

（例） FORMAT 文, DATA 文

CDC では FORMAT 文, DATA 文中の文字列は、二重引用符（"）でくくられているが、FACOM ではこれを単重引用符（'）に変換する必要がある。

```
DATA LPLOT/" PLOTREC" /
1000 FORMAT ("0 * * * * PFN NOT SUPPLIED")
```

↓

```
DATA LPLOT/' PLOTREC' /
1000 FORMAT ('0 * * * * PFN NOT SUPPLIED')
```

- ③ CDC と FACOM-M シリーズの間に FORTRAN ライブラリ, システム・ライブラリの違いがあるため、同機能をもつライブラリに変換する必要がある。

問題点としては、手作業のために作業時間が長くなり不注意による間違いが入りやすかった。

2.4 アップデート作業

旧版の FACOM ソースにコンバージョン後のアップデートカードを挿入したり、新版には必要なくなったステートメントを削除する。

問題点として、アップデートカードが数千枚もある場合には、作業時間が長くかかり、誤りを犯す率が高くなっていた。

2.5 補正作業

コンバージョン作業などは、すべてステートメント単位の変換である。ここでは、これまでの作業ではわからないステートメント間の大域的な変換・修正を行う。

問題点として、ブーリアン変数のビット構成を意識した変換や、サブルーチン間での関係を認識できるツールの不備があげられる。

3. 開発ツールの機能

2章で述べたような問題点を解決するため、Fig.3.1に示すような手順によって修正作業をシステム化した。そこで、このシステムの各ステップにおいて、問題点がどのように解決されたかについて述べる。さらに、RELAP5/MOD2/CYCLE 36の修正に使用したジョブ制御文(JCL)の例を示す。なお、このシステムの詳細な機能、使用法などについては、付録を参照されたい。

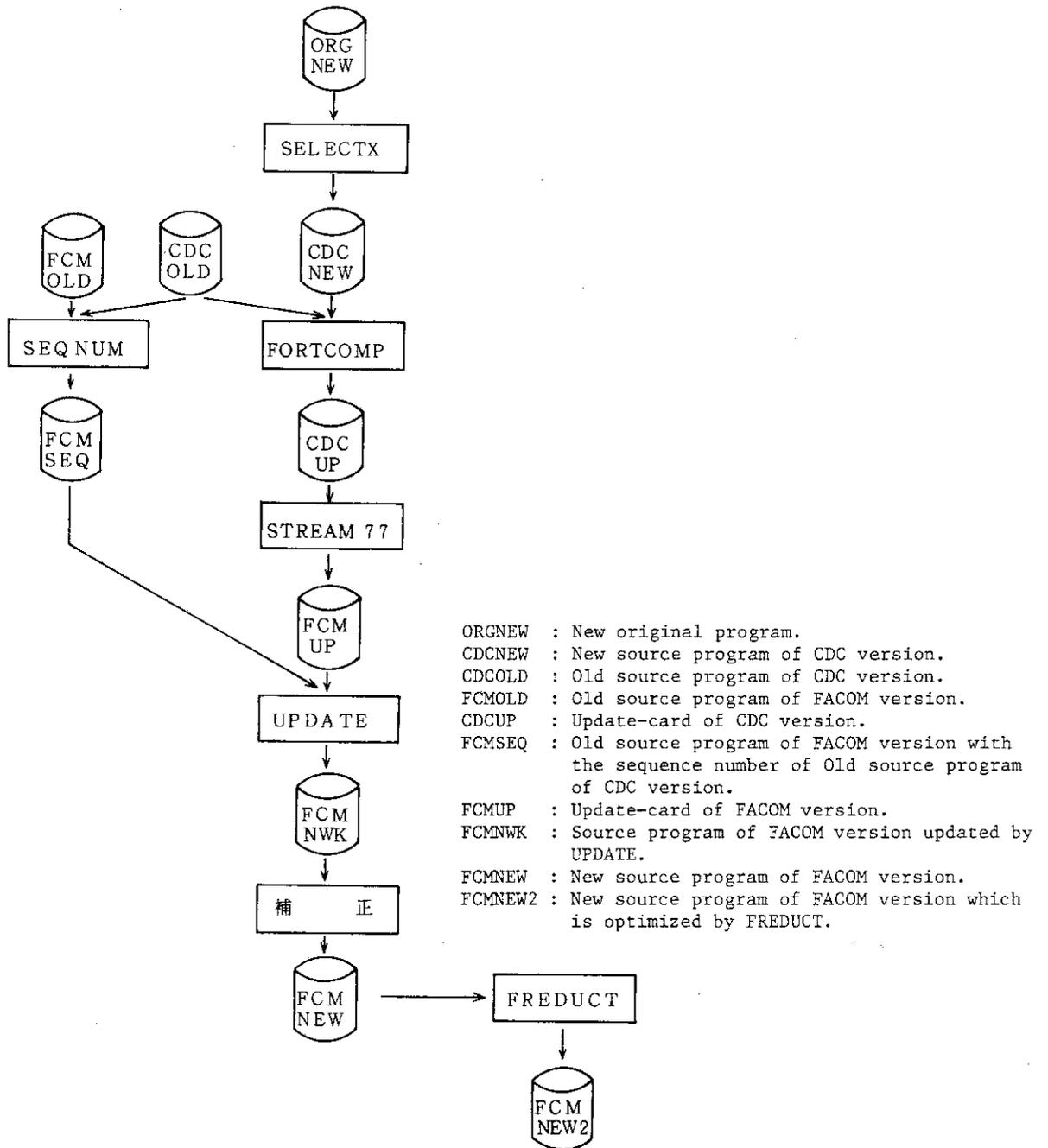


Fig. 3.1 General flow chart for conversion of RELAP5 by using the tools.

3.1 抽出作業 ——— SELECTX

従来まで、Fig.2.3 に示すように3段階で処理していた*IF制御文の展開・削除と¥IF制御文の展開・削除を1本のツールで、まとめてできるようにした。また、制御文が入れ子構造になっているとき正確に作動するようにした。

SELECTXの入出力概略図をFig.3.2に、JCLをFig.3.3に示す。SELECTXでは、オリジナル・プログラム (ORGNEW) と選択するオプション (CONTROL) を入力して変換対象のソース・プログラム (SF), インクルードソース・プログラム (INC), そして、チェックリスト (LIST) を出力する。JCLでは、ORGNEW データセットを DD 名 INSOC, CONTROL データセットを DD 名 SYSIN, SF データセットを DD 名 OUTSOC, INC データセットを DD 名 OUTINC, LIST データセットを DD 名 SYSPRINT にそれぞれ割り当てる。

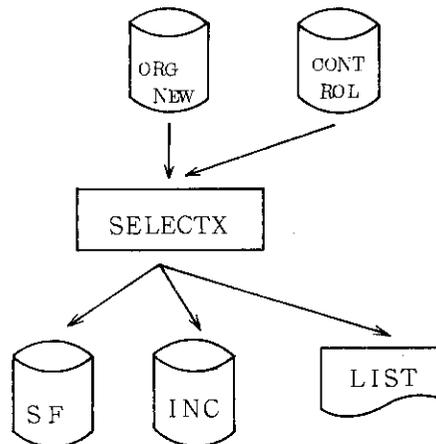


Fig. 3.2 INPUT/OUTPUT configuration for SELECTX.

```

/*
/*
*****
*****      RELAP5/MOD2/C36 SELECT-X PROGRAM      *****
*****
/*
/*
//      EXEC PGM=SELECTX,PARM='ELM(*),MSGLEVEL(9) '
//STEPLIB DD DSN=J████████.R5TOOL.LOAD,DISP=SHR
//INSOC   DD DSN=J████████.R5ORG02.FORT77,DISP=SHR,LABEL=(,/,IN)
//OUTSOC  DD DSN=J████████.R5CDC02.FORT77,DISP=SHR
//OUTINC  DD DSN=J████████.R5CDC02.INCLUDE,DISP=SHR
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=137,BLKSIZE=6850)
//SYSIN   DD *
*DEFINE CDC,CDCRA,NOSBE,FTN4,TIMED,LCM,CH8,SCOPE1
¥DEFINE CDC,CDCRA,NOSBE,FTN4,TIMED,LCM,CH8,SCOPE1
/*
++
//

```

Fig. 3.3 An example of JCL for SELECTX.

3.2 比較作業 —— FORTCOMP

2.2節で示したような手作業で行っていたアップデートカードの作成を自動化した。また扱えるデータセットは、POでもPSでも可能であるため、FORTCOMPを実行するJCLでは各メンバ毎にDD文を指定する必要がない。

FORTCOMPの入出力概略図をFig.3.4に、JCLをFig.3.5に示す。FORTCOMPでは、新・旧CDC版ソース・プログラム(CDCNEW・CDCOLD)を入力してアップデートカード(CDCUP)とチェックリスト(LIST)を出力する。JCLでは、CDCNEWデータセットをDD名NEWSOC、CDOLDデータセットをDD名OLDSOC、CDCUPデータセットをDD名UPDATECD、LISTデータセットをDD名SYPRINTにそれぞれ割り当てる。

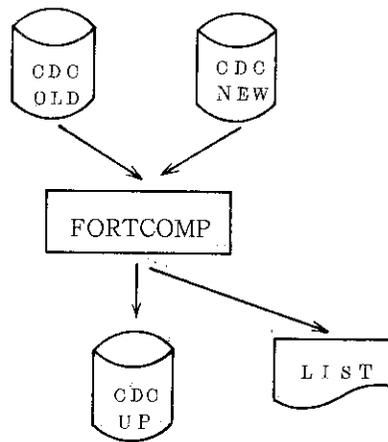


Fig. 3.4 INPUT/OUTPUT configuration for FORTCOMP.

```

//* *****
//* ***** RELAP5/MOD2/C36 FORTRAN COMPARE PROGRAM *****
//* *****
//*
//COMPARE EXEC PGM=FORTCOMP
//STEPLIB DD DSN=J████.R5TOOL.LOAD,DISP=SHR
//OLDSOC DD DSN=J████.R5CDC00.FORT77,DISP=SHR,LABEL=(,IN)
//NEWSOC DD DSN=J████.R5CDC02.FORT77,DISP=SHR,LABEL=(,IN)
//UPDATECD DD DSN=J████.@UPDTC.DATA,DISP=(NEW,CATLG),
// UNIT=TSSWK,SPACE=(TRK,(100,10,50))
//SYSPRINT DD SYSOUT=*,
// DCB=(LRECL=137,BLKSIZE=15344,RECFM=FBA,DSORG=PS)
++
//
  
```

Fig. 3.5 An example of JCL for FORTCOMP.

3.3 コンバージョン作業 ——— STREAM 77

コンバージョン作業は、(富士通^株製品) STREAM 77⁴⁾ を使用した。STREAM 77 はプログラムの変換及びプログラム構造の解析のための C-ENGINE と変換時の前後処理及び補助処理のための P-ENGINE で構成されている。ここでは、主な変換を C-ENGINE で行い、その後、P-ENGINE で後処理を行った。これを使用することによってほとんど 2.3 節で述べた問題点については自動化される。しかし、この段階ではステートメント単位での変換だけなので、自動化の困難な点も多々ある。これらの点については、手作業によって最後の補正ステップで補っている。

STREAM 77 の C-ENGINE と P-ENGINE の JCL をそれぞれ Fig.3.6, Fig.3.7 に示す。STREAM 77 (C-ENGINE) では、CDC 版アップデートカード (CDCUP) と CDC 版インクルードソース (CDCINC) とオプションを入力して FACOM 版アップデートカード (FCMUP1) と FACOM 版インクルードソース (FCMINC1) とチェックリストを出力する。STREAM 77 (P-ENGINE) では、FACOM 版アップデートカード (FCMUP1) と FACOM 版インクルードソース (FCMINC1) とオプションを入力して後処理済み FACOM 版アップデートカード (FCMUP) と後処理済み FACOM 版インクルードソース (FCMINC) とチェックリストを出力する。C-ENGINE の JCL では、CDCUP データセットを DD 名 SYSIN, CDCINC データセットを DD 名 SYSINC, FCMUP1 データセットを DD 名 FT 12 F 001, FCMINC1 データセットを DD 名 FT 14 F 001, チェックリストを DD 名 FT 15 F 001 にそれぞれ割り当てる。P-ENGINE の JCL では、FCMUP1 データセットを DD 名 SYSIN, FCMINC1 データセットを DD 名 SYSINC, オプションの補助入力データセットを DD 名 READ, FCMUP データセットを DD 名 FT 12 F 001, FCMINC データセットを DD 名 FT 14 F 001, チェックリストを DD 名 FT 15 F 001 にそれぞれ割り当てる。

```

//*****
//*          STREAM77 ( C-ENGINE )
//*****
//CENG      EXEC CENGINE,
//          A='NOSUB,NOTAB,NOT,NOGEN,CONV',INC=INCLUDE
//FT12F001 DD DSN=J██████.QUPDTC2.FORT77,DISP=(NEW,CATLG),UNIT=TSSWK,
//          SPACE=(TRK,(50,10,30))
//FT14F001 DD DSN=J██████.QUPDTC2.INCLUDE,DISP=(NEW,CATLG),UNIT=TSSWK,
//          SPACE=(TRK,(20,5,30))
//FT15F001 DD SYSOUT=*
//SYSIN    DD DSN=J██████.QUPDTC.FORT77,DISP=SHR,LABEL=(,,IN)
//SYSINC   DD DSN=J██████.QUPDTC.INCLUDE,DISP=SHR,LABEL=(,,IN)
//

```

Fig. 3.6 An example of JCL for STREAM77 (C-ENGINE)

```

//*****
//*                STREAM77 ( P-ENGINE )
//*****
//PENG      EXEC PENGINE,
//          INC=INCLUDE,A=READ
//FT12F001 DD DSN=J████.①UPDTC3.FORT77,DISP=(NEW,CATLG,DELETE),
//          UNIT=TSSWK,SPACE=(TRK,(10,10,10),RLSE),
//          DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB)
//FT14F001 DD DSN=J████.①UPDTC3.INCLUDE,DISP=(NEW,CATLG,DELETE),
//          UNIT=TSSWK,SPACE=(TRK,(10,10,10),RLSE),
//          DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB)
//READ      DD DSN=J████.①OPTION.DATA,DISP=SHR
//SYSIN     DD DSN=J████.①UPDTC2.FORT77,DISP=SHR
//SYSINC    DD DSN=J████.①UPDTC2.INCLUDE,DISP=SHR
//
//

```

Fig. 3.7 An example of JCL for STREAM77 (P-ENGINE)

3.4 アップデート作業 —— SEQNUM, UPDATE

アップデート作業は、SEQNUM と UPDATE の2つのステップで行われる。

初めに、SEQNUM によって旧 CDC 版ソースに対応するシーケンス番号を旧 FACOM 版ソースにつける。その後 UPDATE ステップで、この旧 FACOM 版ソース（旧 CDC 版のシーケンス番号付き）をアップデートカード（コンバージョン済み・旧 CDC 版のシーケンス番号付き）を使って、自動的にアップデート（挿入・削除）が行えるようにした。

SEQNUM の入出力概略図と JCL をそれぞれ Fig.3.8, Fig.3.9 に示す。SEQNUM では、旧版 CDC ソース・プログラム（CDCOLD）と旧版 FACOM ソース・プログラム（FCMOLD）を入力して、CDC のシーケンス番号のついた旧版 FACOM ソース・プログラム（FCMSEQ）とチェックリスト（LIST）を出力する。JCL では、CDCOLD データセットを DD 名 CHKSOC, FCMOLD データセットを DD 名 INSOC, FCMSEQ データセットを DD 名 OUTSOC, LIST データセットを DD 名 SYSPRINT にそれぞれ割り当てる。

UPDATE の入出力概略図と JCL をそれぞれ Fig.3.10, Fig.3.11 に示す。UPDATE では、SEQNUM で出力されたシーケンス番号付き旧版 FACOM ソース・プログラム（FCMSEQ）と FACOM 版のアップデートカード（FCMUP）を入力して、新版 FACOM ソース・プログラム（FCMNWK）とチェックリスト（LIST）を出力する。JCL では、FCMSEQ データセットを DD 名 OLDSOC, FCMUP データセットを DD 名 UPDATECD, FCMNWK データセットを DD 名 NEWSOC, LIST データセットを DD 名 SYSPRINT にそれぞれ割り当てる。

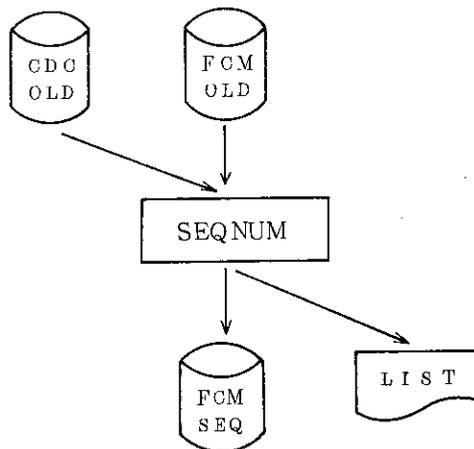


Fig. 3.8 INPUT/OUTPUT configuration for SEQNUM.

```

//*
//*
//*          *****
//*          ***** RELAP5/MD2/C36 SEGNUMBER PROGRAM *****
//*          *****
//*
//*
// EXEC PGM=SEQNUM
//STEPLIB DD DSN=J████.R5TOOL.LOAD,DISP=SHR
//CHKSOC DD DSN=J████.R5CDCOO.FORT77,DISP=SHR,LABEL=(,,,IN)
//INSOC DD DSN=J████.R5FCMOO.FORT77,DISP=SHR
//OUTSOC DD DSN=J████.R5FCMOOW.FORT77,DISP=SHR
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=137,BLKSIZE=6850)
++
//

```

Fig. 3.9 An example of JCL for SEQNUM.

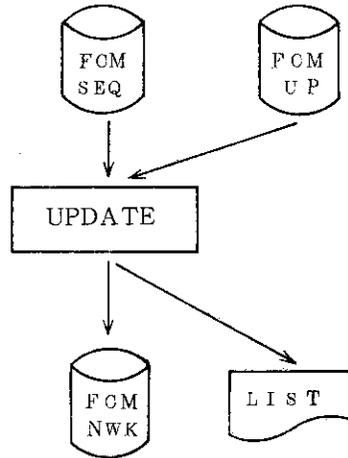


Fig. 3.10 INPUT/OUTPUT configuration for UPDATE.

```

//*
//* *****
//* ***** RELAP5/MOD2/C36 UPDATE PROGRAM *****
//* *****
//*
//UPDATE EXEC PGM=UPDATE
//STEPLIB DD DSN=J████████.R5TOOL.LOAD,DISP=SHR
//UPDATECD DD DSN=J████████.QUPDTC2.DATA,DISP=SHR,LABEL=(,/,IN)
//QLDSOC DD DSN=J████████.R5FCM00W.FORT77,DISP=SHR,LABEL=(,/,IN)
//NEWSOC DD DSN=J████████.R5FCM02W.FORT77,DISP=SHR
//SYSPRINT DD SYSOUT=*,
// DCB=(LRECL=137,BLKSIZE=15344,RECFM=FBA,DSORG=PS)
//
//
//

```

Fig. 3.11 An example of JCL for UPDATE.

3.5 手作業による補正作業

このステップは、サブルーチン単位やサブルーチン間での関係を考慮にいたした変換、ブーリアン変数のチェックなどを行うが、様々な情報が必要なため手作業で行っている。問題点で述べたブーリアン変数のビット構成、サブルーチン間での関係を考慮したツールについては、今後の課題である。

3.6 ビット操作関数の最適化 —— FREDUCT

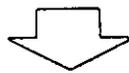
FREDUCT は、ここで述べた一連の修正作業には直接関係ないが、RELAP 5 の修正作業が終了した後、このコードの処理速度を向上させるためにビット操作関数の最適化をするツール⁵⁾である。

CDC 版計算機の FORTRAN コンパイラは、領域節約のためのパック語を取り扱うためのビット操作関数、論理演算子を持っている。富士通計算機では、1語のビット数の違いなどから、それらを完全に模擬する関数がない。このため、アセンブラでビット操作関数を作成し、すべてこれにおきかえている。関数の変更作業は、ツール (STREAM 77) を用いて機械的に行われるため、無駄な関数の組合せが生じる場合がある。このような場合にひとつずつ手作業で最適化することは、作業時間や信頼性の観点から問題があるため、機械的に最適化し無駄な関数の組合せを削除するツール FREDUCT を用意した。

この最適化の例を Fig.3.12 に示す。この例では、単純変換によって作られた 8 個の関数 (¥ID, ¥SUB, ¥ADD, ¥NOT, ¥SFT, ¥DI, ¥MASK) を FREDUCT によって 2 つの関数 ¥IDAND, ¥SFT に置き換えている。

FREDUCT の入出力概略図と JCL をそれぞれ Fig.3.13, Fig.3.14 に示す。FREDUCT では、補正ステップで完成した新版 FACOM ソース・プログラム (FCMNEW) と制御データ (CONTROL) を入力して最適化済み新版 FACOM ソース・プログラム (FCMNEW 2) とチェックリスト (LIST) を出力する。JCL では、FCMNEW データセットを DD 名 INSOC, CONTROL データセットを DD 名 SYSIN, FCMNEW 2 データセットを DD 名 OUTSOC, LIST データセットを DD 名 SYSPRINT にそれぞれ割り当てる。

```
KP = (.NOT.MASK(48) .AND. SHIFT(IJ1(I),30)) + IXSOP - 1
```



コンバージョン

```
KP = ¥ID( ¥SUB( ¥ADD( ¥AND( ¥NOT( ¥MASK(48)) ,
¥SFT( IJ1(I),30)) , ¥DI(IXSOP)),¥B0001))
```



FREDUCTによる最適化

```
KP = ¥IDAND( ¥MKN48 , ¥SFT( IJ1(I),30)) + IXSOP - 1
```

Fig. 3.12 An example of optimization of bit operation functions by FREDUCT.

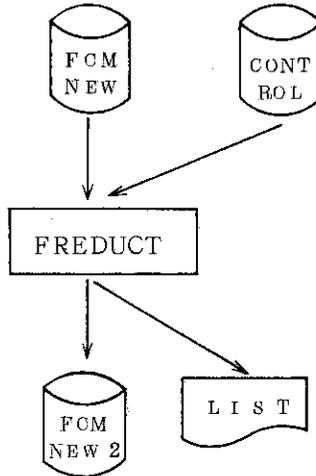


Fig. 3.13 INPUT/OUTPUT configuration for FREDUCT.

```

/*
/* *****
/* ***** FUNCTION REDUCTION PROGRAM *****
/* *****
/*
// EXEC PGM=FREDUCT,
// PARM='ELM(*),CHKLIST(INOUT),REPLACE'
//STEPLIB DD DSN=J████████.R5TOOL.LOAD,DISP=SHR
//INSOC DD DSN=J████████.R5FCM02.FORT77,DISP=SHR,LABEL=(,,,IN)
//OUTSOC DD DSN=J████████.R5FCM02R.FORT77,DISP=SHR
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=137,BLKSIZE=6850)
//SYSIN DD *
-MODIFY1
  %ID(%ADD,+
  %ID(%SUB,-
  %ID(%ADD,+
-MODIFY2
  %ID(%AND,%IDAND
  %ID(%DI
-MODIFY3
  %NOT(%MASK,%MKN
  %MASK,%MKP
-END
/*
//
  
```

Fig. 3.14 An example of JCL for FREDUCT.

4. ツール使用による効果

修正作業用ツールを整備することによって、つぎのような効果が得られた。

第1の効果として、従来までの修正方法に比べて、作業工数を大幅に削減することが可能になった。また RELAP 5 コードのバージョンアップ作業は、作業者の経験・ノウハウによって作業時間が異なっていた。ところが、この修正作業用システムを使用することで、経験・ノウハウの少ない作業者は、必要な作業の内容・流れがわかりやすくなり、従来より円滑に作業を進めることができるようになった。そして、経験・ノウハウの少ない作業者においても、かなり作業工数を削減することができた。

次に、具体例で、経験・ノウハウのある作業者やない作業者が、どの程度作業工数が削減できたかを定量的に示す。一般に、作業工数と修正量は比例すると考えられるので、工数削減化の度合を示す指標として次の式で生産性を定義する。

$$\begin{aligned} \text{作業全体の工数 (人月)} &= \text{調査にかかった工数} + \text{変換にかかった工数} + \text{実行にかかった工数} \\ \text{生産性} &= \text{修正量 (アップデートカードのステップ数)} \div \text{作業全体の工数 (人月)} \\ &= 1 \text{ 人月で修正できる量 (アップデートカードのステップ数)} \end{aligned}$$

TABLE 4.1 は、従来の手作業による方法とツールを使用した方法で、修正作業を行ったときにかかった工数とその工数から算出される生産性を示している。TABLE. 4.1 の1列目は、RELAP 5 / MOD 2 / CYCLE 21 から CYCLE 36 への変換を従来の手作業による方法でノウハウのある者が作業を行ったときの工数と生産性である。2列目は、RELAP 5 / MOD 2 / CYCLE 36.00 から CYCLE 36.02 への変換をツールを使用してノウハウのある者が作業を行ったときの工数と生産性で、3列目は、2列目と同じコードをツールを使用してノウハウのない者が作業を行ったときの工数と生産性である。生産性は、最後の行で示され、各列での作業工数の合計（作業全体の工数）と修正量から算出される。また（従来方法比）の欄では、従来の方法での生産性を1.0倍としたときにツールを使用したときの生産性がどの程度向上したかを示す倍率である。

結果として、ノウハウのある者の場合、生産性が2.6倍向上した。（作業工数が1 / 2.6になった。）また、ノウハウのない者の場合、（従来の方法でノウハウのある者が作業した場合と比べて）2.0倍生産性が向上している。

これは、このツールの最大の利点であり、今後のバージョンアップに少なからぬ貢献をすることと思われる。

第2の効果として、従来作業フローとツールを使用したときのフローを比較すればわかるように、単純な作業は大部分自動化され、手作業が大幅に減少した。このことは、手作業による変換ミスを減少させ作業の信頼性を向上させるのに寄与している。

第3の効果として、変換済みの新しいソースを試験的に実行してデバッグする際に、従来フローでは、ほとんどのステップで手作業が入るため、どの時点で誤ったかを判断するのが困難であった。しかし、システム化されたフローでは最後のステップ（補正ステップ）以外自動化されている。このため、ほとんどの場合は補正ステップだけのチェックで済ませることができる。ま

た、補正ステップ以前のステップにおいてオプションなどを誤ってもすぐにやり直しができるので、STEP-BY-STEP でかつ迅速に作業を行える。

TABLE 4.1 An example of productivity of the conversion.

変換方法	従来方法	ツール使用	
変換コード名	RELAP 5 / MOD 2 CYCLE21 → CYCLE36	RELAP 5 / MOD 2 CYCLE36.00 → CYCLE36.02	
修正量	3,000ステップ	1,000ステップ	
ノウハウ	有り	有り	無し
調査	2.0人×1.0ヶ月	2.0人×0.5週間	1.0人×1.0週間
変換	2.0人×1.5ヶ月	2.0人×0.5週間	1.0人×2.0週間
実行	2.0人×0.5ヶ月	2.0人×0.5週間	1.0人×1.0週間
合計	6.00人月	0.75人月	1.00人月
生産性 (従来方法比)	500 (1.0倍)	1,300 (2.6倍)	1,000 (2.0倍)

5. お わ り に

RELAP5 コードの変換・修正作業は、現在でも頻繁に行われている。ところが、作業担当者は順次変わっていくため、作業の継承が困難である。このツールの整備によって、作業担当者のノウハウにあまり依存せず作業が継承できることが今後期待できる。

本ツールはRELAP5 コードの変換保守のために開発されているが、RELAP5 と同じようなソース・プログラムの管理方法をしているプログラムに対しても適用が可能である。

今後の課題としては、ブーリアン変数のビット構成、サブルーチン間での関係を考慮したツールを検討することが重要となるであろう。また、オリジナルコードの修正を、ベクトル計算機向きに書き直されたベクトル化版に反映させ、修正済みベクトル化版を作成する方法がある。これは、現在手作業で行っているが、本ツールの適用の検討も重要である。

謝 辞

計算センター室長浅井清氏、および富士通(株)科学システム部長田子精男氏には、本報告書を書く機会を与えて頂きました。心より感謝します。

計算センターの藤井実氏には、本報告書の投稿にあたり記述方法の御指導を頂きました。感謝いたします。

参考文献

- 1) V.H.Ramson, R-J.Wagner ; RELAP 5 / MOD 2 CODE MANUAL Volume 1 EGG-SAMM-6377, INEL (April 1984)
- 2) 奈良岡賢逸, 他 ; 軽水炉安全性解析コード RELAP 5 / MOD 2 / CYCLE21 の FACOM M-380 システムへの変換, 私信 (August 1985)
- 3) CDC ; UPDATE Reference Manual
CRC (株) ; サービスビューロー・ユーザーズ・ガイド
- 4) 富士通(株) ; FACOM OS IV / F 4 MSP STREAM77 説明書 (February 1985)
- 5) 篠沢尚久, 他 ; 軽水炉安全性解析コード RELAP 5 / MOD 2 / CYCLE36 のベクトル化, JAERI-M86-019 第 4 章 (February 1986)

5. お わ り に

RELAP 5 コードの変換・修正作業は、現在でも頻繁に行われている。ところが、作業担当者は順次変わっていくため、作業の継承が困難である。このツールの整備によって、作業担当者のノウハウにあまり依存せず作業が継承できることが今後期待できる。

本ツールは RELAP 5 コードの変換保守のために開発されているが、RELAP 5 と同じようなソース・プログラムの管理方法をしているプログラムに対しても適用が可能である。

今後の課題としては、ブーリアン変数のビット構成、サブルーチン間での関係を考慮したツールを検討することが重要となるであろう。また、オリジナルコードの修正を、ベクトル計算機向きに書き直されたベクトル化版に反映させ、修正済みベクトル化版を作成する方法がある。これは、現在手作業で行っているが、本ツールの適用の検討も重要である。

謝 辞

計算センター室長浅井清氏、および富士通（株）科学システム部長田子精男氏には、本報告書を書く機会を与えて頂きました。心より感謝します。

計算センターの藤井実氏には、本報告書の投稿にあたり記述方法の御指導を頂きました。感謝いたします。

参考文献

- 1) V.H.Ramson, R-J.Wagner ; RELAP 5 / MOD 2 CODE MANUAL Volume 1 EGG-SAMM-6377, INEL (April 1984)
- 2) 奈良岡賢逸, 他 ; 軽水炉安全性解析コード RELAP 5 / MOD 2 / CYCLE21 の FACOM M-380 システムへの変換, 私信 (August 1985)
- 3) CDC ; UPDATE Reference Manual
CRC (株) ; サービスビューロー・ユーザーズ・ガイド
- 4) 富士通 (株) ; FACOM OS IV / F 4 MSP STREAM77 説明書 (February 1985)
- 5) 篠沢尚久, 他 ; 軽水炉安全性解析コード RELAP 5 / MOD 2 / CYCLE36 のベクトル化, JAERI-M86-019 第 4 章 (February 1986)

5. お わ り に

RELAP5 コードの変換・修正作業は、現在でも頻繁に行われている。ところが、作業担当者は順次変わっていくため、作業の継承が困難である。このツールの整備によって、作業担当者のノウハウにあまり依存せず作業が継承できることが今後期待できる。

本ツールはRELAP5 コードの変換保守のために開発されているが、RELAP5 と同じようなソース・プログラムの管理方法をしているプログラムに対しても適用が可能である。

今後の課題としては、ブーリアン変数のビット構成、サブルーチン間での関係を考慮したツールを検討することが重要となるであろう。また、オリジナルコードの修正を、ベクトル計算機向きに書き直されたベクトル化版に反映させ、修正済みベクトル化版を作成する方法がある。これは、現在手作業で行っているが、本ツールの適用の検討も重要である。

謝 辞

計算センター室長浅井清氏、および富士通(株)科学システム部長田子精男氏には、本報告書を書く機会を与えて頂きました。心より感謝します。

計算センターの藤井実氏には、本報告書の投稿にあたり記述方法の御指導を頂きました。感謝いたします。

参考文献

- 1) V.H.Ramson, R-J.Wagner ; RELAP 5 / MOD 2 CODE MANUAL Volume 1 EGG-SAMM-6377, INEL (April 1984)
- 2) 奈良岡賢逸, 他 ; 軽水炉安全性解析コード RELAP 5 / MOD 2 / CYCLE21 の FACOM M-380 システムへの変換, 私信 (August 1985)
- 3) CDC ; UPDATE Reference Manual
CRC (株) ; サービスビューロー・ユーザーズ・ガイド
- 4) 富士通(株) ; FACOM OS IV / F 4 MSP STREAM77 説明書 (February 1985)
- 5) 篠沢尚久, 他 ; 軽水炉安全性解析コード RELAP 5 / MOD 2 / CYCLE36 のベクトル化, JAERI-M86-019 第 4 章 (February 1986)

付録A SELECTX

A.1 機能

A.1.1 機能の概要

SELECTX プログラムは、RELAP 5 コードの管理、保存に用いられるオリジナル・プログラムを入力とし、指定したオプションに従ってソースを抽出し、2つのデータセット、FORTRAN ソース・データセット及び FORTRAN インクルードソース・データセットを作成するツールである。

RELAP 5 コードは、2.1節で記述したようにプログラムの管理、保存の際、計算機のハードウェア、OS、コンパイラの種類などに依存する部分を重ねて持ち、そういったものに影響されない共通部分は一つのソースのみ保有するようになっている。

オリジナル・プログラム内には、選択オプションに対応して展開、削除を行うための*で始まる制御文(*DEFINE, *IF, *ENDIF)と¥で始まる制御文(¥DEFINE, ¥IF, ¥ENDIF)が混在している。前者は、CDCのUPDATEユーティリティのものであり、後者は、RELAP 5シリーズのSELECTXユーティリティのものである。*制御文も¥制御文も機能は同じである。ただし、¥制御文は削除対象のステートメントに対し、コメント行(先頭1カラムを'*'とする)として残すが、*制御文は完全に消去してしまう。以前から、この2つのユーティリティの代用ツールはあったが、制御文が入れ子構造になっている場合、正確に作動しない不完全なものであった。今回、新しく作成するに際して、一本のプログラムで*制御文、¥制御文の両方の展開、削除を行えるようにした。

A.1.2 *DEFINE (¥DEFINE) 制御文

ユーザーが使用可能なソースを取り出すために必要なオプションを、*DEFINE 制御文を使ってプログラムの先頭に定義する。Fig. A.1にその例を示す。また、この例に示すように、RELAP 5におけるUPDATEユーティリティの実質的な目的は、指定されたオプションをSELECTXユーティリティに必要な¥DEFINE 制御文で定義することである。

```

*DEFINE CDC
*DEFINE FTN4
*IF DEF CDC
¥DEFINE CDC
*ENDIF
*IF DEF CRA
¥DEFINE CRA
*ENDIF
*IF DEF IBM
¥DEFINE IBM
*ENDIF
*IF DEF FTN4
¥DEFINE FTN4
*END IF
.
.
.

```

```

¥DEFINE CDC
¥DEFINE FTN4
.
.
.

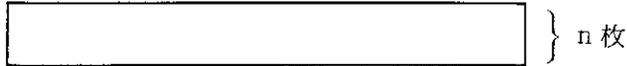
```

Fig. A.1 An example of *DEFINE control statement in the UPDATE utility program of CDC.

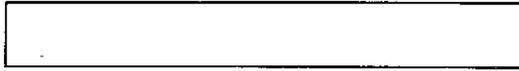
A.1.3 * IF, * ENDIF (¥IF, ¥ENDIF) 制御文

* IF 制御文は下記の4つのいずれかのパターンをとる。

- (1) ¥IF DEF, VAR, n

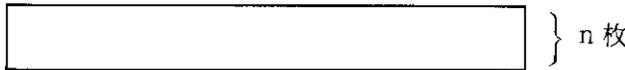


- (2) ¥IF DEF, VAR

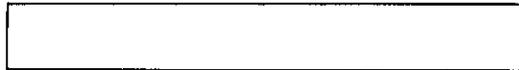


¥ENDIF

- (3) ¥IF -DFE, VAR, n



- (4) ¥IF -DEF, VAR



¥ENDIF

それぞれ、次のような意味を持つ。

- VAR が定義されているとき (¥DEFINE 文で定義する。)
 - (1) 引き続き n 枚が、ソース中に展開される。
 - (2) ¥ENDIF 文までの全ての文が展開される。
 - (3) 引き続き n 枚が省略される。
 - (4) ¥ENDIF 文までの全ての文が省略される。
- VAR が定義されていないとき (前述の展開と省略が逆になる。)
 - (1) 引き続き n 枚が省略される。
 - (2) ¥ENDIF 文までの全ての文が省略される。
 - (3) 引き続き n 枚が、ソース中に展開される。
 - (4) ¥ENDIF 文までの全ての文が展開される。

SELECTX ユーティリティによる使用例を Fig. A.2 に示す。

今回作成したプログラム SELECTX は、削除対象のソースのコメント化、消去、どちらも可能なようになっている。

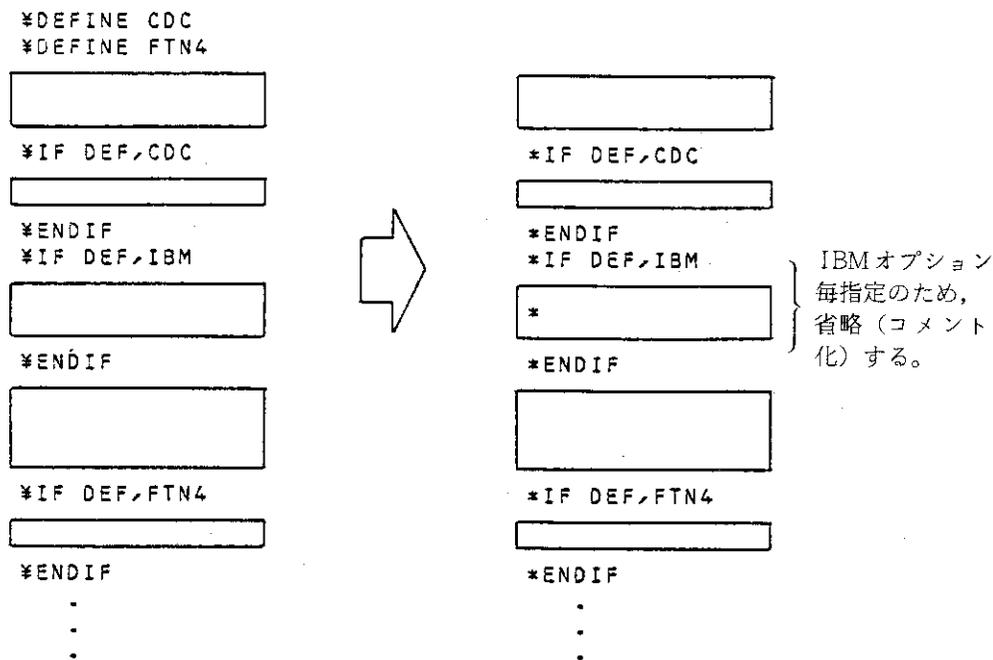


Fig. A.2 An example of %IF control line in SELECTX utility program of RELAP5.

A.2 実行方法

SELECTX プログラムの主な入力は、RELAP5 オリジナル・プログラムである。その他の入力として、制御データ、実行時オプションがある。Fig. A.3 に示すように、オリジナル・プログラムは INSOC データセットから、制御データは SYSIN データセット、実行時オプションは EXEC 文の PARM パラメータ、又は SYSIN データセットから入力され、抽出後の FORTRAN ソースは OUTSOC データセット、抽出後の FORTRAN インクルードソースは OUTINC データセット、実行時の印刷情報は SYSPRINT データセットに出力される。

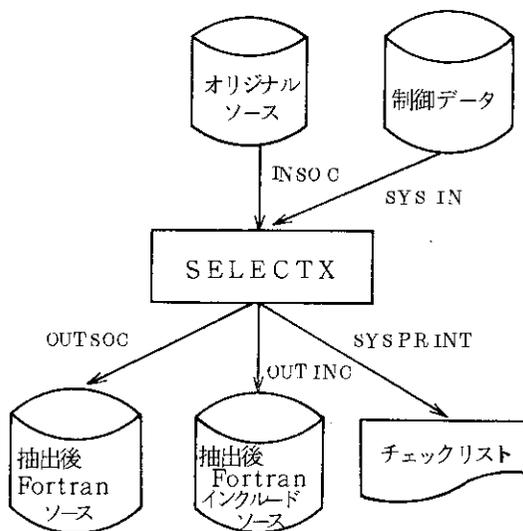


Fig. A.3 INPUT/OUTPUT configuration for SELECTX.

A.2.1 SELECTX の実行

SELECTX プログラムを実行するためには、1つの EXEC 文といくつかの DD 文を用意しなければならない。EXEC 文には通常、TABLE A.1 に示すパラメータを指定すれば良い。PARM パラメータの指定内容の詳細は、A.2.2実行時オプションを参照されたい。実行時には必要な DD 文はTABLE A.2で示されるものである。この中には、条件により用意しなくても良いものがある。実行時のジョブ制御文の例をFig.A.4に示す。

TABLE.A.1 Main parameters of EXEC statements of JCL for SELECTX.

PGM	プログラム名SELECTX を指定する。このパラメータは必須である。
PARM	実行時のオプションを指定する。

TABLE.A.2 DD statements of JCL for SELECTX.

DD名	必要の有無
INSOC	必ず用意する。
SYSIN	制御データを入力する際に必要
OUTSOC	必ず用意する。
OUTINC	抽出ソースについて、FORTRANソースとインクルードソースを別データセットにする際に必要
SYSPRINT	印刷情報を見たい場合に必要

```
// EXEC PGM=SELECTX
//STEPLIB DD DSN=J████████.R5TOOL.LOAD,DISP=SHR
//INSOC DD DSN=J████████.R5ORGO2.FORT77,DISP=SHR,LABEL=(,/,IN)
//OUTSOC DD DSN=J████████.R5CDCO2.FORT77,DISP=SHR
//OUTINC DD DSN=J████████.R5CDCO2.INCLUDE,DISP=SHR
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=137,BLKSIZE=6850)
//SYSIN DD *
*DEFINE CDC,CDDCRA,NOSBE,FTN4,TIMED,LCM,CH8,SCOPE1
*DEFINE CDC,CDDCRA,NOSBE,FTN4,TIMED,LCM,CH8,SCOPE1
/*
```

Fig. A.4 An example of JCL for SELECTX.

A.2.2 実行時オプション

TABLE A.3に実行時オプションを示す。これらは、EXEC 文の PARM パラメータ又は SYSIN データセットの先頭（制御データの前）に指定できる。複数個の実行時オプションを指定する場合は、各々のオプションをコンマ又は空白で区切って指定する。相反する複数のオプションが指定された場合は最後のオプションが有効となる。指定されなかったオプションの値は、標

TABLE.A.3 Options of JCL for SELECTX.

オプションの形式		
* ((COMMENT <u>DELETE</u> NO))	NO*	*制御文に関する抽出を行うかの指定
≠ ((COMMENT <u>DELETE</u> NO))	NO≠	≠制御文に関する抽出を行うかの指定
ELM (mem1 [-mem2] [,mem1 [-mem2]] ...) <u>ELM</u> (*)		区分データセット内の対象とするメンバ名の指定 区分データセット内のすべてのメンバを対象とする。
IN ((INSOC DD名))		オリジナルソース入力DD名の指定
INCLUDE (((OUTINC DD名)))	NOINCLUDE	インクルードソース出力DD名の指定
INC (((OUTINC DD名)))	NOINC	(省略形)
LINECOUNT ((60 n)) LC ((60 n))		印刷情報の1ページあたりの行数の指定 (省略形)
MSGLEVEL ((0 1 2 <u>3</u> 9))		印刷情報の出力レベルの指定
OUT ((OUTSOC DD名))		FORTRAN ソース出力DD名の指定
RENUMBER [(((100 n1)) [, (100 n2))]) RENUM [(((100 n1)) [, (100 n2))]) REN [(((100 n1)) [, (100 n2))])	<u>NORENUMBER</u> <u>NORENUM</u> <u>NOREN</u>	シーケンス番号付けの指定 (省略形) (省略形)
REPLACE REP	<u>NOREPLACE</u> <u>NOREP</u>	既に存在するメンバを置き換えるか否かの指定 (省略形)

注) 下線はオプション省略時に取られる標準値である。

標準値が取られる。

次に実行時オプションの詳細を説明する。

* [((COMMENT | DELETE | NO))]

NO*

オリジナル・プログラムについて、*制御文に従った展開・削除を行うかどうかを指示するものである。*(COMMENT)が指定されると、展開・削除を行うが、その際、削除対象となったステートメント(*制御文も含む)をコメント化してソース上に残すものである。COMMENTの代わりにCOM又はCを用いても良い。

*(DELETE)又は*指定された場合、展開・削除を行うが、その際削除対象となったステートメント(*制御文を含む)を完全に消去するものである。DELETEの代わりにDEL又はDを用いても良い。

*(NO)又はNO*が指定された場合、*制御文に関する展開・削除を行なわないことを指定する。この場合、*制御文はコメント文とみなされる。NOの代わりにNを用いても良い。

なお、このオプションを省略した場合は、*(DELETE)が指定されたものとみなされる。

¥ (({COMMENT | DELETE | NO}))

NO¥

オリジナル・プログラムについて、¥制御文に従った展開・削除を行うかどうかを指示するものである。

本プログラムでは、*制御文による展開・削除、¥制御文による展開・削除、少なくともどちらかは実行させなければならない。両方指定した場合、*制御文に関する展開・削除を行ったソースに対して、¥制御文に関する展開・削除を行う。

¥(COMMENT)又は¥が指定された場合、展開・削除を行うが、その際、削除対象となったソース・ステートメント(¥制御文も含む)をコメント化してソース上に残すものである。COMMENTの代わりにCOM又はCを用いても良い。

¥(DELETE)が指定された場合、展開・削除を行うが、その際、削除対象となったステートメント(¥制御文も含む)を完全に消去するものである。DELETEの代わりにDEL又はDを用いても良い。

¥(NO)又はNO¥が指定された場合、¥制御文に関する展開・削除は行なわないことを指定する。この場合、¥制御文は通常の文とみなされる。NOの代わりにNを用いても良い。

なお、このオプションを省略した場合は、¥(DELETE)が指定されたものとみなされる。

ELM (mem 1 [-mem 2] [, mem 3 [-mem]] ……)

ELM (*)

オリジナル・プログラムの入力を区分データセットから行う際のメンバ名を指示するものである。mem 1, mem 2, ……はオリジナル・プログラムが存在するメンバの名前であり、最大100組指定可能である。メンバの指定方法には直接、メンバ名を指定する方法と範囲で指定する方法がある。直接指定する方法は、対象とするメンバ名をコンマで区切って並べる。範囲で指定する方法は、'—'の前後にメンバ名を書く。

始まりと終りを示す名前は、必ずしも実在の名前でなくても良い。範囲指定の場合、EBCDICコードによる大小判断で行う。その際、始まりを示すメンバ名は後ろにLOW-VALUEを補い、終りを示すメンバ名にはHIGH-VALUEを補う。また、ELM(*)を指定した場合は、全メンバが対象となる。

(例)

ELM (A-C, Z) 先頭1文字がAまたはBまたはCで始まるメンバと名前がZで始まるメンバが対象となる。

順データセットの場合、この指定は意味を持たない。区分データセットのとき、このオプションを省略した場合は、ELM(*)が指定されたものとみなされる。メンバの処理は、EBCDICコード順に行われる。

IN ({INSOC | DD名})

オリジナル・プログラムの入力DD名を指定する。省略した場合、DD名はINSOCとなる。DD名としてINSOCが使用できない場合にこのオプションを用いる。

INCLUDE ({OUTINC | DD名}) または INC ({OUTINC | DD名})

NOINCLUDE または NOINC

FORTRAN インクルードソースの出力 DD 名を指定する。省略した場合、DD 名は OUTINC となる。DD 名として OUTINC が使用できない場合に、このオプションを用いる。

NOINCLDE が指定された場合、FORTRAN インクルードソースは、FORTRAN ソースと同じデータセットに出力される。

LINECOUNT ({60 | n}) または LC ({60 | n})

印刷情報を出力するとき、1 ページ当りの行数を指定するものである。この行数には、SELECTX の見出し行も含まれる。特に、n = 0 のときは、印刷情報は全てベタ打ちとなる。すなわち、見出し行は 1 回だけ出力され、その後は、編集のための改ページは行わない。省略時は、60 が指定されたものとみなす。

MSGLEVEL ({ 0 | 1 | 2 | 3 | 9 })

印刷情報の出力レベルを指定する。

MSGLEVEL (0) は、オリジナル・プログラムの全てに対して展開・削除の説明を付けて出力することを意味する。

MSGLEVEL (1) は、展開されたソース及び制御文について出力することを意味する。

MSGLEVEL (2) は、削除されたソース及び制御文について出力することを意味する。

MSGLEVEL (3) は、制御文のみ展開・削除の説明を付けて出力することを意味する。

MSGLEVEL (9) は、印刷情報は出力しないことを意味する。

このオプションを省略した場合は、MSGLEVEL (3) が指定されたものとみなされる。また、このオプションとは関係なく SYSIN 入力データの内容が最初に出力される。

OUT ({ OUTSOC | DD名 })

FORTRAN ソース出力 DD 名を指定する。省略した場合、DD 名は OUTSOC となる。DD 名として OUTSOC が使用できない場合にこのオプションを用いる。

RENUMBER (([{100 | n1}] [, {100 | n2}])

または RENUM (([{100 | n1}] [, {100 | n2}])

または REN (([{100 | n1}] [, {100 | n2}])

NORENUMBER または NORENUM または NOREN

ソースを出力する際に、シーケンス番号付けを行うかどうかを指定する。シーケンス番号は、出力メンバが代わるたびに初期値から付けなおす。

RENUMBER (n1, n2) は、n1 が初期値、n2 が増分値を意味する。

RENUMBER (n1) は、初期値を n1、増分値を 100 とすることを意味する。

RENUMBER (, n2) は、初期値を 100、増分値を n2 とすることを意味する。

RENUMBER は、初期値を 100、増分値を 100 とすることを意味する。

NORENUMBER は、シーケンス番号付を行わないことを意味する。

標準値は NORENUMBER である。

REPLACE 又は REP

NOREPLACE 又は NOREP

出力データセットに既に同じメンバ名が存在している場合に、それと置き換えるかどうかを指

定する。

REPLACE は置き換えることを意味する。古いメンバは消去される。

NOREPLACE は置き換えないことを意味する。新しいメンバの処理は行われない。

標準値は NOREPLACE である。

A.2.3 実行時のデータセット

SELECTX は TABLE A.4 に示されている入力装置上のデータセットに対してアクセスする。データセットを DD 文で定義する場合、基本的には定められた DD 名を使用する。DD 文では、DD 名、データセット名、装置情報、ボリューム情報、データセットの取り扱い及び DCB 情報等を記述する。データセットがカタログされている場合、DD 文のパラメータとして、データセット名 (DSNAME パラメータ) とデータセットの取り扱い (DISP パラメータ) だけを指定すればよい。

データセットの DCB 属性が与えられていない場合、SELECTX はある規約に従って DCB 属性を割り当てる。

TABLE.A.4 Attributes of dataset in SELECTX.

DD 名	入出力装置	D C B 属 性			
		D S O R G	RECFM	LRECL	BLKSIZE
INSOC ^井	直接アクセス装置 磁気テープ装置 カード読み取り装置	PS 又は PO	F,FB	80	80の整数倍
SYSIN	直接アクセス装置 磁気テープ装置 カード読み取り装置	PS	F,FB	80	80の整数倍
OUTSOC ^井	直接アクセス装置 磁気テープ装置 カード読み取り装置	PS 又は PO	F,FB	80	80の整数倍
OUTINC ^井	直接アクセス装置	PO	F,FB	80	80の整数倍
SYSPRINT	直接アクセス装置 磁気テープ装置 カード読み取り装置	PS	FBA	137 / 81	レコード長の整数倍
			VBA	137 / 81	レコード長 + 4 バイト
STEPLIB	直接アクセス装置	PO	U	0	任意

井) この DD 名は、実行時のオプションによって変更可能である。

備考 1) INSOC, SYSIN は、入力ストリームから入力できる。OUTSOC は、出力ストリームの SYSOUT クラスをカード穿孔装置として使用できる。SYSPRINT は、出力ストリームの SYSOUT クラスをラインプリンタとして使用できる。

備考 2) OUTSOC, OUTINC の DCB 属性が与えられていない場合、INSOC の DCB 属性が割り当てられる。

備考 3) INSOC, SYSIN, SYSPRINT は、端末装置に割り付けてもよい。

備考 4) PO データセットは、直接アクセスデータ装置のみ割り付けられる。

A.2.3.1 INSOC データセット

RELAP5 のオリジナル・プログラムが格納されているデータセットである。このデータセットは、基本的には順データセットであるが、区分データセットも使用可能である。区分データセットの場合、メンバ毎に処理を分けるのではなく、対象となるメンバ全てが1つの順データセットのように扱われる。DD 名は、実行時オプションの指定により変更が可能である。

DD 文の指定によって連結したデータセットを定義してもよい。この場合は、各データセットの DCB 属性は同じでなければならない。また、一つのデータセットの処理が終わってから次のデータセットの処理を行う。

INSOC データセットは、入力ストリームの中でデータを定義してもよい。この場合は、データの最後に区切り文を置く。

A.2.3.2 SYSIN データセット

オリジナル・プログラムを抽出・削除するための制御データが格納されているデータセットである。制御データが不要ならば、この DD 文は必要ない。詳細は A.3 制御データを参照されたい。

データセットの先頭、制御データより前に A.2.2 で示した実行時オプションを置くことができる。PARM パラメータのオプションよりこのオプションのほうが優先される。

SYSIN データセットは、入力ストリームの中でデータを定義してもよい。この場合は、データの最後の区切りとして区切り文を置く。

A.2.3.3 OUTSOC データセット

SELECTX が出力する FORTRAN ソースを格納するデータセットである。このデータセットは、FACOM 版へのコンバージョンの際のもとなるソースである。このデータセットは基本的には区分データセットでなければならないが、順データセットも使用可能である。この DD 名は、実行時オプションの指定により変更可能である。

区分データセットの場合、*DECK (または *COMDECK) 文で指定された名前をメンバ名としてメンバを作成する。

A.2.3.4 OUTINC データセット

SELECTX が出力する FORTRAN インクルードソースを格納するデータセットである。このデータセットは、FACOM 版へのコンバージョンの際のもとなるソースである。このデータセットは区分データセットでなければならない。この DD 名は、実行時オプションの指定により変更可能である。

この DD 文を省略した場合、または実行時オプションで NOINCLUDE を指定した場合、FORTRAN インクルードソースは OUTSOC データセットに出力される。その方式は、FORTRAN ソースの場合と同じである。

出力の際のメンバ名は、*COMDECK 文で与えられた名前が用いられる。

A.2.3.5 SYSPRINT データセット

実行の際の名種の印刷情報を出力するデータセットである。詳細はA.5印刷情報を参照されたい。

SYSPRINT データセットは、通常システムの出力装置を指定して (SYSOUTパラメータを用いる)、そのユニットレコード装置のクラスをラインプリンタ装置とする。

A.2.3.6 STEPLIB データセット

SELECTX が登録されているロードモジュールデータセットである。

A.3 制御データ

オリジナル・プログラムの先頭には * DEFINE 制御文及び ¥DEFINE 制御文の中に持つ * IF - * ENDIF 制御文がなければならない。ところが、INEL から送られるオリジナル・プログラムの中には、先頭に置かれていない場合がある。このとき、数万行もあるデータセットを EDIT して修正することは大変厄介なことである。

このような場合に、制御データから定義すべきオプションを指定することにより、オリジナル・プログラムを手作業で編集しなおすことは不要となる。

制御データの書き方は以下のとおりである。

1 2 3 4 5 6 7	8	9	72	73	80
* D E F I N E		option, option,			
¥ D E F I N E		option, option,			

制御データの9～72カラムの間に、選択オプションをコンマで区切って並べる。書ききれない場合は、同形式のデータを必要枚数置く。

ここで、* DEFINE データは * 制御文に関する選択オプションを、¥DEFINE データは ¥制御文に関する選択オプションを設定する。

現在、RELAP 5 のコンバージョンにおいて採用しているオプションは、

CDC, CDCRA, FTN4, NOSBE, TIMED, CHB, SCOPE1, LCM

の計8個である。実際の使用例をFig.A.4の入力データ (DD 名SYSIN) に示す。

A.4 出力ソースの形式

オリジナル・プログラムは、各プログラム単位の先頭に * DECK 文、コモンブロック単位の先頭に * COMDECK 文、プログラム中のコモンブロックを展開すべき位置に * CALL文が置かれている。Fig.A.5にオリジナル・プログラムの例を示す。

OUTSOC データセット、OUTINC データセットが共に区分データセットが割りつけられているとき、選択後のソースは、DECK 名をメンバ名として OUTSOC データセットに作成され

る。コモンブロック単位は、COMDECK 名をメンバ名として OUTINC データセットに作成される。* CALL 文は、* INCLUDE 文に置き換えられる。このとき、実行時オプションに * (DELETE) が与えられている場合は、DECK 文、COMDECK 文が共に消去され、* (COMMENT) が与えられている場合は、コメント行として各メンバの先頭に残る。(Fig.A.6, A.7 参照)

OUTSOC データセットが区分データセットに割りつけられ、実行時オプションに NOINCLUDE が与えられている場合 (又は OUTINC データセットが割りつけられていない場合)、コモンブロック単位の出力データセットが OUTSOC データセットに変更される以外は前述と同じ変換が行われる。

OUTSOC データセットが順データセットに割りつけられ、OUTINC データセットが区分データセットに割りつけられている場合、コモンブロック単位は、COMDECK 名をメンバ名として OUTINC データセットに作成される。

このとき、COMDECK 文は実行時オプションに従って消去されるか、コメント行として残される。ソースは、実行時オプションにかかわらず DECK をプログラムの先頭に残し順データセットに出力される。(Fig.A.8 参照)

OUTSOC データセットが順データセットに割りつけられ、OUTINC データセットが割りつけられていない場合 (又は NOINCLUDE オプションが与えられている場合)、コモンブロック単位も COMDECK 文を残したまま OUTSOC データセットに出力される。(Fig .A.9 参照)

```

*DECK ACCUM
  SUBROUTINE ACCUM(I,K,L,J,JJ)
C
C  ACCUMULATOR MASS, WALL HEAT TRANSFER AND MOMENTUM MODEL
C
C  COGNIZANT ENGR: DMK
*CALL CMPDAC
*CALL CMPDAT
*CALL COMCTL
*CALL CONTRL
*CALL FAST
*CALL GENRL
*CALL JUNDAT
*CALL SCRATCH
*CALL STATEC
*CALL TRNHLP
*CALL VOLDAT
C
C *** LOCAL VARIABLES
C
C  INTEGER OUTPUT
  DATA GRAV/9.80665/, PIE/3.141592654/, XN/0.3333333333/
*IF DEF,CRAY,1
  DATA OUTPUT/6LOUTPUT/
*IF DEF,CDC, 1
  DATA OUTPUT/L"OUTPUT"/
C
C *** CHECK FOR A SUCCESSFUL TIME STEP
C
C  IF (SUCCES .EQ. 0) GO TO 11
C
C --- UNSUCCESSFUL...RESET NEW TIME TERMS TO OLD TIME TERMS
C
*IF DEF,CRAY,1
  IF (SHIFT(ACCON(I),6).GE.0 .OR. SHIFT(ACCON(I),9).LT.0) GO TO 10
*IF DEF,CDC,1
  IF (SHIFT(ACCON(I),2).GE.0 .OR. SHIFT(ACCON(I),5).LT.0) GO TO 10
  QUALA(K) = 0.0
  QUALAQ(K) = 0.0
  VOIDF(K) = 1.0
  VOIDG(K) = 0.0
  10 CONTINUE
*IF DEF,CB205, 2
  ACCON(I) =OR( (AND(DCMPL(SHIFT(MASK(3),60) ), ACCON(I))) ,
+  SHIFT(AND(SHIFT(MASK(3),57) , ACCON(I)), 3))
*IF DEF,IBM, 2
  ACCON(I) =DOR( (DAND(DCMPL(SHIFT(MASK(3),60) ), ACCON(I))) ,
+  SHIFT(DAND(SHIFT(MASK(3),57) , ACCON(I)), 3))
*IF DEF,CRAY,2
  ACCON(I) = (.NOT.SHIFT(MASK(3),60) .AND. ACCON(I)) .OR.
+  SHIFT(SHIFT(MASK(3),57) .AND. ACCON(I), 3)
*IF DEF,CDC,2
  ACCON(I) = (.NOT.MASK(3) .AND. ACCON(I)) .OR.
+  SHIFT(SHIFT(MASK(3),57) .AND. ACCON(I), 3)
  GO TO 13
  11 CONTINUE
C
C --- SUCCESSFUL...RESET OLD TIME TERMS TO NEW TIME TERMS
C
  VDMO(I) = VDM(I)

```

Fig. A.5 An example of original source program with the *IF control lines.

```

      SUBROUTINE ACCUM(I,K,L,J,JJ)
C
C   ACCUMULATOR MASS, WALL HEAT TRANSFER AND MOMENTUM MODEL
C
C   COGNIZANT ENGR: DMK
C   *INCLUDE CMPDAC
C   *INCLUDE CMPDAT
C   *INCLUDE COMCTL
C   *INCLUDE CONTRL
C   *INCLUDE FAST
C   *INCLUDE GENRL
C   *INCLUDE JUNDAT
C   *INCLUDE SCRTCH
C   *INCLUDE STATEC
C   *INCLUDE TRNHLP
C   *INCLUDE VOLDAT
C
C   *** LOCAL VARIABLES
C
C   INTEGER OUTPUT
C   DATA GRAV/9.80665/, PIE/3.141592654/, XN/0.3333333333/
C   DATA OUTPUT/L"OUTPUT"/
C
C   *** CHECK FOR A SUCCESSFUL TIME STEP
C
C   IF (SUCCES .EQ. 0) GO TO 11
C
C   --- UNSUCCESSFUL...RESET NEW TIME TERMS TO OLD TIME TERMS
C
C   IF (SHIFT(ACCON(I),2).GE.0 .OR. SHIFT(ACCON(I),5).LT.0) GO TO 10
C   QUALA(K) = 0.0
C   QUALAO(K) = 0.0
C   VOIDF(K) = 1.0
C   VOIDG(K) = 0.0
C   10 CONTINUE
C   ACCON(I) = (.NOT.MASK(3) .AND. ACCON(I)) .OR.
C   * SHIFT(MASK(3),57) .AND. ACCON(I), 3)
C   GO TO 13
C   11 CONTINUE
C
C   --- SUCCESSFUL...RESET OLD TIME TERMS TO NEW TIME TERMS
C
C   VDMO(I) = VDM(I)
C   RHONO(I) = RHON(I)
C   TVAPO(I) = TEMPG(K)
C   VLIQO(I) = VLIQ(I)
C   TTANKO(I) = TTANK(I)
C   QTANKO(I) = QTANK(I)
C   BORONO(K) = BORON(K)
C

```

Fig. A.6 An example of source program of Fig. A.5 processed by SELECTX with DELETE option.

```

*DECK ACCUM
  SUBROUTINE ACCUM(I,K,L,J,JJ)
C
C  ACCUMULATOR MASS, WALL HEAT TRANSFER AND MOMENTUM MODEL
C
C  COGNIZANT ENGR: DMK
*INCLUDE CMPDAC
*INCLUDE CMPDAT
*INCLUDE COMCTL
*INCLUDE CONTRL
*INCLUDE FAST
*INCLUDE GENRL
*INCLUDE JUNDAT
*INCLUDE SCRATCH
*INCLUDE STATEC
*INCLUDE TRNHLP
*INCLUDE VOLDAT
C
C *** LOCAL VARIABLES
C
      INTEGER OUTPUT
      DATA GRAV/9.80665/, PIE/3.141592654/, XN/0.3333333333/
*IF DEF,CRAY,1
*   DATA OUTPUT/6LOUTPUT/
*IF DEF,CDC, 1
      DATA OUTPUT/L"OUTPUT"/
C
C *** CHECK FOR A SUCCESSFUL TIME STEP
C
      IF (SUCCES .EQ. 0) GO TO 11
C
C --- UNSUCCESSFUL...RESET NEW TIME TERMS TO OLD TIME TERMS
C
*IF DEF,CRAY,1
*   IF (SHIFT(ACCON(I),6) .GE. 0 .OR. SHIFT(ACCON(I),9) .LT. 0) GO TO 10
*IF DEF,CDC,1
      IF (SHIFT(ACCON(I),2) .GE. 0 .OR. SHIFT(ACCON(I),5) .LT. 0) GO TO 10
      QUALA(K) = 0.0
      QUALAO(K) = 0.0
      VOIDF(K) = 1.0
      VOIDG(K) = 0.0
      10 CONTINUE
*IF DEF,CB205, 2
*   ACCON(I) =OR( (AND(COMPL(SHIFT(MASK(3),60) ), ACCON(I))) ,
*   +   SHIFT(AND(SHIFT(MASK(3),57) , ACCON(I)), 3))
*IF DEF,IBM, 2
*   ACCON(I) =DOR( (DAND(DCMPL(SHIFT(MASK(3),60) ), ACCON(I))) ,
*   +   SHIFT(DAND(SHIFT(MASK(3),57) , ACCON(I)), 3))
*IF DEF,CRAY,2
*   ACCON(I) = (.NOT.SHIFT(MASK(3),60) .AND. ACCON(I)) .OR.
*   +   SHIFT(SHIFT(MASK(3),57) .AND. ACCON(I), 3)
*IF DEF,CDC,2
      ACCON(I) = (.NOT.MASK(3) .AND. ACCON(I)) .OR.
      *   SHIFT(SHIFT(MASK(3),57) .AND. ACCON(I), 3)
      GO TO 13
      11 CONTINUE
C
C --- SUCCESSFUL...RESET OLD TIME TERMS TO NEW TIME TERMS
C
      VDMO(I) = VDM(I)
      RHONO(I) = RHON(I)
      TVAPO(I) = TEMPG(K)

```

Fig. A.7 An example of source program of Fig. A.5 processed by SELECTX with COMMENT option.

```

*DECK RELAP5
      PROGRAM RELAP5 (INPUT=128,OUTPUT=512,RSTIN=512,RSTPLT=512,
      * TALKER=10,PLOTFL=0,STH2XT=0,DEBUG=OUTPUT,TAPE6=OUTPUT)
C
C RELAP5, A COMPUTER PROGRAM TO SIMULATE A NUCLEAR REACTOR LOSS OF
C COOLANT ACCIDENT.
      .
      .
      .

1000 STOP
      END
*DECK ACCUM
      SUBROUTINE ACCUM(I,K,L,J,JJ)
C
C ACCUMULATOR MASS, WALL HEAT TRANSFER AND MOMENTUM MODEL
C
C COGNIZANT ENGR: DMK
*INCLUDE CMPDAC
*INCLUDE CMPDAT
*INCLUDE COMCTL
*INCLUDE CONTRL
*INCLUDE FAST
*INCLUDE GENRL
*INCLUDE JUNDAT
*INCLUDE SCRTCH
*INCLUDE STATEC
*INCLUDE TRNHLP
*INCLUDE VOLDAT
C
C *** LOCAL VARIABLES
C
      INTEGER OUTPUT
      DATA GRAV/9.80665/, PIE/3.141592654/, XN/0.3333333333/
      DATA OUTPUT/L"OUTPUT"/
C
C *** CHECK FOR A SUCCESSFUL TIME STEP
C
      IF (SUCCES .EQ. 0) GO TO 11
C
C --- UNSUCCESSFUL...RESET NEW TIME TERMS TO OLD TIME TERMS
C
      IF (SHIFT(ACCON(I),2).GE.0 .OR. SHIFT(ACCON(I),5).LT.0) GO TO 10
      QUALA(K) = 0.0
      QUALAO(K) = 0.0
      VOIDF(K) = 1.0
      VOIDG(K) = 0.0
10 CONTINUE
      ACCON(I) = (.NOT.MASK(3) .AND. ACCON(I)) .OR.
      * SHIFT(SHIFT(MASK(3),57) .AND. ACCON(I), 3)
      GO TO 13
11 CONTINUE
C
C --- SUCCESSFUL...RESET OLD TIME TERMS TO NEW TIME TERMS
C
      VDMO(I) = VDM(I)

```

Fig. A.8 An example of source program of Fig. A.5 processed by SELECTX allocated the sequential dataset to OUTSOC dataset and the partition dataset to OUTINC dataset.

```

*COMDECK SIUNIT
C
C *** SCALE BRITISH TO SI UNITS
C          VBR = BRITISH VARIABLE,
C          CONVYY = SI TO BRITISH CONVERSION FACTOR
C
          SIUNIT(VBR, CONVYY) = VBR * AMAX1(0.0, (1.0/CONVYY))
          * - AMIN1(0.0, SIGN(0.5555555556,CONVYY)) * (VBR - 1.8*CONVYY)
*COMDECK TRCBK
          CALL STRACE
*COMDECK ZALFAG
C COMMON INLINE FORMULA TO CALCULATE VOID FROM X, VG,VF
          ALFAG(QXX,VXXG,VXXF) = AMAX1(0.0, AMIN1(1.0, (QXX * VXXG
          * / (VXXF + QXX * (VXXG - VXXF))))))
C
C          ALFAG = VOIDG
C          QXX   = QUALITY
C          VXXG  = VAPOR SP. VOLUME
C          VXXF  = LIQUID SP. VOLUME
C
*DECK DEFINE
*DECK RELAP5
          PROGRAM RELAP5 (INPUT=128,OUTPUT=512,RSTIN=512,RSTPLT=512,
          * TALKER=10,PLOTPL=0,STH2XT=0,DEBUG=OUTPUT,TAPE6=OUTPUT)
C
          :
          :
          :
1000 STOP
          END
*DECK ACCUM
          SUBROUTINE ACCUM(I,K,L,J,JJ)
C
C ACCUMULATOR MASS, WALL HEAT TRANSFER AND MOMENTUM MODEL
C
C COGNIZANT ENGR: DMK
*INCLUDE CMPDAC
*INCLUDE CMPDAT
*INCLUDE COMCTL
*INCLUDE CONTRL
*INCLUDE FAST
*INCLUDE GENRL
*INCLUDE JUNDAT
*INCLUDE SCRTCH
*INCLUDE STATEC
*INCLUDE TRNHLP
*INCLUDE VOLDAT
C
C *** LOCAL VARIABLES
C
          INTEGER OUTPUT
          DATA GRAV/9.80665/, PIE/3.141592654/, XN/0.3333333333/
          DATA OUTPUT/L"OUTPUT"/
C
C *** CHECK FOR A SUCCESSFUL TIME STEP
C
          IF (SUCCES .EQ. 0) GO TO 11
C
C --- UNSUCCESSFUL...RESET NEW TIME TERMS TO OLD TIME TERMS
C
          IF (SHIFT(ACCON(I),2).GE.0 .OR. SHIFT(ACCON(I),5).LT.0) GO TO 10
          QUALA(K) = 0.0
          QUALAQ(K) = 0.0
          VOIDF(K) = 1.0
          VOIDG(K) = 0.0
          10 CONTINUE
          ACCON(I) = (.NOT.MASK(3) .AND. ACCON(I)) .OR.
          * SHIFT(SHIFT(MASK(3),57) .AND. ACCON(I), 3)
          GO TO 13
          11 CONTINUE
C
C --- SUCCESSFUL...RESET OLD TIME TERMS TO NEW TIME TERMS
C
          VDMO(I) = VDM(I)

```

Fig. A.9 An example of source program of Fig. A.5 processed by SELECTX allocated the sequential dataset to OUTSOC dataset with NOINCLUDE option.

A.5 印刷情報

SELECTX が出力する印刷情報には SYSIN データ・チェックリスト、ソースの出力情報、及びエラーメッセージの三種類がある。エラーメッセージの説明は、ここで省略する。

A.5.1 SYSIN データ・チェックリスト

SYSIN データセットから入力したデータの内容をそのまま出力するものである。Fig.A.10 にその例を示す。このリストは、実行時パラメータの MSGLEVEL の値に関係なく出力される。

```

                SELECT-X PROGRAM V10L10                                DATE 1986.04.26
<<< SYSIN DATA CHECK LIST >>>
.....".....1.....".....2.....".....3.....".....4.....".....5.....".....6.....".....7.....".....8
*DEFINE CDC,CDCRA,NOSBE,FTN4,TIMED,LCM,CH8,SCOPE1                00130000
¥DEFINE CDC,CDCRA,NOSBE,FTN4,TIMED,LCM,CH8,SCOPE1                00140000

```

Fig. A.10 An example of check list for SYSIN dataset.

A.5.2 ソースの出力情報

オリジナル・プログラムの各制御文による展開・削除の情報を出力するものであるが、実行時パラメータ MSGLEVEL 値によって出力する情報が変化する。

MSGLEVEL (0) の場合、オリジナル・プログラムの全内容について展開・削除の情報を付けて出力するものである。Fig.A.11にその例を示す。

MSGLEVEL (1) の場合、*IF 制御文によって展開されたソースと、全制御文が出力される。Fig.A.12にその例を示す。

MSGLEVEL (2) の場合、*IF 制御文によって削除されたソースと、全制御文が出力される。Fig.A.13にその例を示す。

MSGLEVEL (3) の場合、全制御文が出力される。Fig.A.14にその例を示す。

MSGLEVEL (9) の場合、チェックリストは全く出力されない。

また、これらの例のようにオプション LINECOUNT の値が 0 より大きい場合には、1 ページ当りの印刷行数が指定値を越えたとき以外にも、メンバの変更時に改ページを行う。この時は、ページの先頭に '*' で囲ったメンバ名を印刷する。

DATE 1986.04.28 TIME 09:24:29 PAGE.0117

SELECT-X PROGRAM VIOL10

```

*****
** ACCUM
**
<<DELETE>> SOC NO.0002111
<<SELECT>> SOC NO.0002112
<<SELECT>> SOC NO.0002113
<<SELECT>> SOC NO.0002114
<<SELECT>> SOC NO.0002115
<<SELECT>> SOC NO.0002116
<<INCLUDE>> SOC NO.0002117
<<INCLUDE>> SOC NO.0002118
<<INCLUDE>> SOC NO.0002119
<<INCLUDE>> SOC NO.0002120
<<INCLUDE>> SOC NO.0002121
<<INCLUDE>> SOC NO.0002122
<<INCLUDE>> SOC NO.0002123
<<INCLUDE>> SOC NO.0002124
<<INCLUDE>> SOC NO.0002125
<<INCLUDE>> SOC NO.0002126
<<INCLUDE>> SOC NO.0002127
<<SELECT>> SOC NO.0002128
<<SELECT>> SOC NO.0002129
<<SELECT>> SOC NO.0002130
<<SELECT>> SOC NO.0002131
<<SELECT>> SOC NO.0002132
<<DELETE>> SOC NO.0002133
<<DELETE>> SOC NO.0002134
<<DELETE>> SOC NO.0002135
<<SELECT>> SOC NO.0002136
<<SELECT>> SOC NO.0002137
<<SELECT>> SOC NO.0002138
<<SELECT>> SOC NO.0002139
<<SELECT>> SOC NO.0002140
<<SELECT>> SOC NO.0002141
<<SELECT>> SOC NO.0002142
<<SELECT>> SOC NO.0002143
<<DELETE>> SOC NO.0002144
<<DELETE>> SOC NO.0002145
<<DELETE>> SOC NO.0002146
<<SELECT>> SOC NO.0002147
<<SELECT>> SOC NO.0002148
<<SELECT>> SOC NO.0002149
<<SELECT>> SOC NO.0002150
<<SELECT>> SOC NO.0002151
<<SELECT>> SOC NO.0002152
<<DELETE>> SOC NO.0002153
<<DELETE>> SOC NO.0002154
<<DELETE>> SOC NO.0002155
<<DELETE>> SOC NO.0002156
<<DELETE>> SOC NO.0002157

I*DECK ACCUM
I SUBROUTINE ACCUM(K,L,J,JJ)
IC
IC ACCUMULATOR MASS, WALL HEAT TRANSFER AND MOMENTUM MODEL
IC
IC COGNIZANT ENGR: DMK
I*CALL CMPDAC
I*CALL CMPDAT
I*CALL COMCTL
I*CALL CONTRL
I*CALL FAST
I*CALL GENRL
I*CALL JUNDAT
I*CALL SCRICH
I*CALL STATEC
I*CALL TRNHLP
I*CALL VOLDAT
IC
IC *** LOCAL VARIABLES
IC
IC INTEGER OUTPUT
I DATA GRAY/9.80665/, PIE/3.141592654/, XN/0.333333333333/
I*IF DEF,GRAY,1
I DATA OUTPUT/6LOUTPUT/
I*IF DEF,CDC, 1
I DATA OUTPUT/L"OUTPUT"/
IC
IC *** CHECK FOR A SUCCESSFUL TIME STEP
I IF (SUCCES .EQ. 0) GO TO 11
IC --- UNSUCCESSFUL...RESET NEW TIME TERMS TO OLD TIME TERMS
I*IF DEF,GRAY,1
I IF (SHIFT(ACCON(I),6).GE.0 .OR. SHIFT(ACCON(I),9).LT.0) GO TO 10
I*IF DEF,CDC,1
I IF (SHIFT(ACCON(I),2).GE.0 .OR. SHIF(ACCON(I),5).LT.0) GO TO 10
I QUALA(K) = 0.0
I QUALAO(K) = 0.0
I VOIDF(K) = 1.0
I VOIDG(K) = 0.0
I 10 CONTINUE
I*IF DEF,CB205, 2
I ACCON(I) =OR( (AND(COMPL(SHIFT(MASK(3),60) ), ACCON(I))) ,
I + SHIFT(AND(SHIFT(MASK(3),57) , ACCON(I)), 3))
I*IF DEF,IBM, 2
I ACCON(I) =DOR( (DAND(DCMPL(SHIFT(MASK(3),60) ), ACCON(I))) ,

```

Fig. A.11 An example of print out of source program with MSGLEVEL (0)

DATE 1986.04.28 TIME 09:34:24 PAGE.0117

SELECT-X PROGRAM V10L10

```

*****
** ACCUM
*****
<<DELETE>> SOC NO.0002111
<<SELECT>> SOC NO.0002112
<<SELECT>> SOC NO.0002113
<<SELECT>> SOC NO.0002114
<<SELECT>> SOC NO.0002115
<<SELECT>> SOC NO.0002116
<<INCLUDE>> SOC NO.0002117
<<INCLUDE>> SOC NO.0002118
<<INCLUDE>> SOC NO.0002119
<<INCLUDE>> SOC NO.0002120
<<INCLUDE>> SOC NO.0002121
<<INCLUDE>> SOC NO.0002122
<<INCLUDE>> SOC NO.0002123
<<INCLUDE>> SOC NO.0002124
<<INCLUDE>> SOC NO.0002125
<<INCLUDE>> SOC NO.0002126
<<INCLUDE>> SOC NO.0002127
<<SELECT>> SOC NO.0002128
<<SELECT>> SOC NO.0002129
<<SELECT>> SOC NO.0002130
<<SELECT>> SOC NO.0002131
<<SELECT>> SOC NO.0002132
<<DELETE>> SOC NO.0002133
<<DELETE>> SOC NO.0002135
<<SELECT>> SOC NO.0002136
<<SELECT>> SOC NO.0002137
<<SELECT>> SOC NO.0002138
<<SELECT>> SOC NO.0002139
<<SELECT>> SOC NO.0002140
<<SELECT>> SOC NO.0002141
<<SELECT>> SOC NO.0002142
<<SELECT>> SOC NO.0002143
<<DELETE>> SOC NO.0002144
<<DELETE>> SOC NO.0002146
<<SELECT>> SOC NO.0002147
<<SELECT>> SOC NO.0002148
<<SELECT>> SOC NO.0002149
<<SELECT>> SOC NO.0002150
<<SELECT>> SOC NO.0002151
<<SELECT>> SOC NO.0002152
<<DELETE>> SOC NO.0002153
<<DELETE>> SOC NO.0002156
<<DELETE>> SOC NO.0002159
<<DELETE>> SOC NO.0002162
<<SELECT>> SOC NO.0002163
<<SELECT>> SOC NO.0002164
<<SELECT>> SOC NO.0002165

I*DECK ACCUM
SUBROUTINE ACCUM(I,K,L,J,JJ)
IC
IC ACCUMULATOR MASS, WALL HEAT TRANSFER AND MOMENTUM MODEL
IC
IC COGNIZANT ENGR: DNK
I%CALL CMPDAC
I%CALL CMPDAT
I%CALL COMCTL
I%CALL CONTRL
I%CALL FAST
I%CALL GENRL
I%CALL JUNDAT
I%CALL SCRICH
I%CALL STATEC
I%CALL TRNHELP
I%CALL VOLDAT
IC
IC *** LOCAL VARIABLES
IC
IC INTEGER OUTPUT
DATA GRAV/9.80665/, PIE/3.141592654/, XN/0.33333333333/
I%IF DEF,CRAY,1
I%IF DEF,CDC, 1
DATA OUTPUT/"OUTPUT"/
IC
IC *** CHECK FOR A SUCCESSFUL TIME STEP
I IF (SUCCES .EQ. 0) GO TO 11
IC
IC --- UNSUCCESSFUL...RESET NEW TIME TERMS TO OLD TIME TERMS
I%IF DEF,CRAY,1
I%IF DEF,CDC,1
IF (SHIFT(ACCON(I),2).GE.0 .OR. SHIFT(ACCON(I),5).LT.0) GO TO 10
QUALACK = 0.0
QUALAO(K) = 0.0
VOIDF(K) = 1.0
VOIDG(K) = 0.0
10 CONTINUE
I%IF DEF,CB205, 2
I%IF DEF,IBM, 2
I%IF DEF,CRAY,2
I%IF DEF,CDC,2
ACCON(I) = (.NOT.MASK(3) .AND. ACCON(I)) .OR.
* SHIFT(SHIFT(MASK(3),57) .AND. ACCON(I), 3)
GO TO 13

```

Fig. A.12 An example of print out of source program with MSGLEVEL (1)

DATE 1986.04.28 TIME 09:40:15 PAGE.0106

SELECT-X PROGRAM V10L10

```

*****
** ACCUM
*****
SOC NO.0002111 <<DELETE>>
SOC NO.0002133 <<DELETE>>
SOC NO.0002134 <<DELETE>>
SOC NO.0002135 <<DELETE>>
SOC NO.0002144 <<DELETE>>
SOC NO.0002145 <<DELETE>>
SOC NO.0002146 <<DELETE>>
SOC NO.0002153 <<DELETE>>
SOC NO.0002154 <<DELETE>>
SOC NO.0002155 <<DELETE>>
SOC NO.0002156 <<DELETE>>
SOC NO.0002157 <<DELETE>>
SOC NO.0002158 <<DELETE>>
SOC NO.0002159 <<DELETE>>
SOC NO.0002160 <<DELETE>>
SOC NO.0002161 <<DELETE>>
SOC NO.0002162 <<DELETE>>
SOC NO.0002180 <<DELETE>>
SOC NO.0002181 <<DELETE>>
SOC NO.0002182 <<DELETE>>
SOC NO.0002187 <<DELETE>>
SOC NO.0002188 <<DELETE>>
SOC NO.0002189 <<DELETE>>
SOC NO.0002190 <<DELETE>>
SOC NO.0002191 <<DELETE>>
SOC NO.0002201 <<DELETE>>
SOC NO.0002202 <<DELETE>>
SOC NO.0002203 <<DELETE>>
SOC NO.0002204 <<DELETE>>
SOC NO.0002205 <<DELETE>>
SOC NO.0002206 <<DELETE>>
SOC NO.0002207 <<DELETE>>
SOC NO.0002208 <<DELETE>>
SOC NO.0002209 <<DELETE>>
SOC NO.0002210 <<DELETE>>
SOC NO.0002216 <<DELETE>>
SOC NO.0002217 <<DELETE>>
SOC NO.0002218 <<DELETE>>
SOC NO.0002219 <<DELETE>>
SOC NO.0002220 <<DELETE>>
SOC NO.0002222 <<DELETE>>
SOC NO.0002223 <<DELETE>>
SOC NO.0002224 <<DELETE>>
SOC NO.0002295 <<DELETE>>
SOC NO.0002296 <<DELETE>>
SOC NO.0002297 <<DELETE>>
SOC NO.0002302 <<DELETE>>

I*DECK ACCUM
I%IF DEF,CRAY,1
I DATA OUTPUT/6LOUTPUT/
I%IF DEF,CDC,1
I%IF DEF,CRAY,1
I IF (SHIFT(ACCOUN(1),6).GE.0 .OR. SHIFT(ACCOUN(1),9).LT.0) GO TO 10
I%IF DEF,CDC,1
I%IF DEF,CB205,2
I ACCOUN(I) =OR( DAND(COMPL(SHIFT(MASK(3),60) ), ACCOUN(I))),
+ SHIFT(AND(SHIFT(MASK(3),57) , ACCOUN(I)), 3))
I%IF DEF,IBM,2
I ACCOUN(I) =DOR( DAND(DCML(SHIFT(MASK(3),60) ), ACCOUN(I))),
+ SHIFT(DAND(SHIFT(MASK(3),57) , ACCOUN(I)), 3))
I%IF DEF,CRAY,2
I ACCOUN(I) = (.NOT.SHIFT(MASK(3),60) .AND. ACCOUN(I)) .OR.
* SHIFT(SHIFT(MASK(3),57) .AND. ACCOUN(I), 3)
I%IF DEF,CDC,2
I%IF DEF,CRAY,1
I IF (SHIFT(ACCOUN(1),6).LT.0 .OR. VLIQ(1).GT.0.0) GO TO 12
I%IF DEF,CDC,1
I%IF DEF,CB205,1
I ACCOUN(I) =OR( SHIFT(MASK(2) ,59) , ACCOUN(I))
I%IF DEF,IBM,1
I ACCOUN(I) =DOR( SHIFT(MASK(2) ,59) , ACCOUN(I))
I%IF DEF,CDCRA,1
I%IF DEF,CB205,2
I ACCOUN(I) =OR( DAND(COMPL(SHIFT(MASK(3),57) ), ACCOUN(I))),
+ SHIFT(AND(SHIFT(MASK(3),60) , ACCOUN(I)), 57))
I%IF DEF,IBM,2
I ACCOUN(I) =DOR( DAND(DCML(SHIFT(MASK(3),57) ), ACCOUN(I))),
+ SHIFT(DAND(SHIFT(MASK(3),60) , ACCOUN(I)), 57))
I%IF DEF,CDC,2
I ACCOUN(I) = (.NOT.SHIFT(MASK(3),57) .AND. ACCOUN(I)) .OR.
* SHIFT(SHIFT(MASK(3),60) .AND. ACCOUN(I), 57)
I%IF DEF,IBM,1
I KX =DAND( DCML(MASK(47) ), SHIFT(JC(J),28))
I%IF DEF,CDCRA,1
I KX =DAND( DCML(MASK(47) ), SHIFT(JC(J),28))
I%IF DEF,CDC,1
I KX = .NOT.MASK(47) .AND. SHIFT(JC(J),28)
I%IF DEF,CDC,1
I%IF DEF,CRAY,1
I IF (SHIFT(ACCOUN(1),6) .GE. 0) GO TO 22
I%IF DEF,CDC,1
I%IF DEF,CB205,1

```

Fig. A.13 An example of print out of source program with MSGLEVEL (2)

SELECT-X PROGRAM VIOL10 DATE 1986.04.28 TIME 14:43:12 PAGE.0106

```

*****
** ACCUM **
*****
<<DELETE>> SOC NO.0002111 I*DECK ACCUM
<<DELETE>> SOC NO.0002133 I%IF DEF,CRAY,1
<<DELETE>> SOC NO.0002135 I%IF DEF,CDC, 1
<<DELETE>> SOC NO.0002144 I%IF DEF,CRAY,1
<<DELETE>> SOC NO.0002146 I%IF DEF,CDC,1
<<DELETE>> SOC NO.0002153 I%IF DEF,CB205, 2
<<DELETE>> SOC NO.0002156 I%IF DEF,IBM, 2
<<DELETE>> SOC NO.0002159 I%IF DEF,CRAY,2
<<DELETE>> SOC NO.0002162 I%IF DEF,CDC,2
<<DELETE>> SOC NO.0002180 I%IF DEF,CRAY,1
<<DELETE>> SOC NO.0002182 I%IF DEF,CDC,1
<<DELETE>> SOC NO.0002187 I%IF DEF,CB205, 1
<<DELETE>> SOC NO.0002189 I%IF DEF,IBM, 1
<<DELETE>> SOC NO.0002191 I%IF DEF,CDCRA,1
<<DELETE>> SOC NO.0002201 I%IF DEF,CB205, 2
<<DELETE>> SOC NO.0002204 I%IF DEF,IBM, 2
<<DELETE>> SOC NO.0002207 I%IF DEF,CRAY,2
<<DELETE>> SOC NO.0002210 I%IF DEF,CDC,2
<<DELETE>> SOC NO.0002216 I%IF DEF,CB205, 1
<<DELETE>> SOC NO.0002218 I%IF DEF,IBM, 1
<<DELETE>> SOC NO.0002220 I%IF DEF,CDCRA,1
<<DELETE>> SOC NO.0002222 I%IF DEF,CRAY,1
<<DELETE>> SOC NO.0002224 I%IF DEF,CDC,1
<<DELETE>> SOC NO.0002295 I%IF DEF,CRAY,1
<<DELETE>> SOC NO.0002297 I%IF DEF,CDC,1
<<DELETE>> SOC NO.0002302 I%IF DEF,CB205, 1
<<DELETE>> SOC NO.0002304 I%IF DEF,IBM, 1
<<DELETE>> SOC NO.0002306 I%IF DEF,CDCRA,1
<<DELETE>> SOC NO.0002311 I%IF DEF,CB205, 1
<<DELETE>> SOC NO.0002313 I%IF DEF,IBM, 1
<<DELETE>> SOC NO.0002315 I%IF DEF,CDCRA,1
<<DELETE>> SOC NO.0002336 I%IF DEF,CB205, 1
<<DELETE>> SOC NO.0002338 I%IF DEF,IBM, 1
<<DELETE>> SOC NO.0002340 I%IF DEF,CDCRA,1
<<DELETE>> SOC NO.0002345 I%IF DEF,CB205, 1
<<DELETE>> SOC NO.0002347 I%IF DEF,IBM, 1
<<DELETE>> SOC NO.0002349 I%IF DEF,CDCRA,1
<<DELETE>> SOC NO.0002381 I%IF DEF,CRAY,1
<<DELETE>> SOC NO.0002383 I%IF DEF,CDC,1
<<DELETE>> SOC NO.0002409 I%IF DEF,CRAY,1
<<DELETE>> SOC NO.0002411 I%IF DEF,CDC,1
<<DELETE>> SOC NO.0002453 I%IF DEF,CB205, 1
<<DELETE>> SOC NO.0002455 I%IF DEF,IBM, 1
<<DELETE>> SOC NO.0002457 I%IF DEF,CDCRA,1
<<DELETE>> SOC NO.0002459 I%IF DEF,CB205, 1
<<DELETE>> SOC NO.0002461 I%IF DEF,IBM, 1
<<DELETE>> SOC NO.0002463 I%IF DEF,CRAY,1

```

Fig. A.14 An example of print out of source program with MSGLEVEL (3)

付録B SEQNUM

B.1 機能

B.1.1 機能の概要

SEQNUM プログラムは、CDC 版ソースのシーケンス番号（連番）を対応する FACOM 版ソースにつけるプログラムである。ここで言うシーケンス番号とは、プログラム単位（メンバ）の 1 枚目のステートメントから 1, 2, 3, ……と順に与えられた番号を意味し、実際のステートメントの 73~80 カラムに付いている番号（付いているとして）とは関係がない。

日本原子力研究所（原研）における RELAP5 のエラー修正（サイクルアップ）作業は、新・旧二つの CDC 版ソースを比較し変更部分を抽出し、それに対応する FACOM 版ソースを修正する、という形式で行われる。その際、修正部分の FACOM 版への対応付けは従来手作業で行われており、非常に手間のかかるのであった。

一方、RELAP5 の FACOM 版へのコンバージョンでは、CDC 版ソースをコメントとして残し、FACOM 版ソースをその後に付加するという形でおこなわれている。このことを利用し、CDC 版のシーケンス番号を FACOM 版の対応するソースの 73~80 カラムに付加し、実際のアップデート作業にはこの番号を使い機械的に行おうというものである。

B.1.2 シーケンス番号付けの基本

B.1.1 でも述べたように、CDC 版から FACOM 版へのコンバージョンの際変換を必要とするステートメントは、CDC 版ソースをコメントとして残し、その下に FACOM 版ソースを置いている。Fig.B.1 に CDC 版ソースの例、Fig.B.2 に対応する FACOM 版ソースの例を示す。コメント文への書き換えは、先頭から 3 カラムが用いられているので、4~72 カラムは CDC 版、FACOM 版完全に一致させることができる。このように比較を行い、一致するステートメントの CDC 版のシーケンス番号を FACOM 版に付ける。FACOM 版において、シーケンス番号の付かなかったステートメントはコンバージョンの際新たに追加されたステートメントである。これは、直前のステートメントのシーケンス番号をそのまま付ける。継続行がある場合は、正確な対応とはならず、最後のステートメントのシーケンス番号が対応する FACOM 版の文全部につけられることになるが、エラー修正は文単位に行ない継続行だけの修正は基本的に考えていないので、これで修正には十分なシーケンス番号が付くことになる。Fig.B.3 に考えかたの基本を示す。

```

SUBROUTINE ACCUM(I,K,L,J,JJ)
C
C ACCUMULATOR MASS, WALL HEAT TRANSFER AND MOMENTUM MODEL
C
C COGNIZANT ENGR: DMK
*INCLUDE CMPDAC
*INCLUDE CMPDAT
*INCLUDE COMCTL
*INCLUDE CONTRL
*INCLUDE FAST
*INCLUDE GENRL
*INCLUDE JUNDAT
*INCLUDE SCRATCH
*INCLUDE STATEC
*INCLUDE TRNHLP
*INCLUDE VOLDAT
C
C *** LOCAL VARIABLES
C
C INTEGER OUTPUT
DATA GRAV/9.80665/, PIE/3.141592654/, XN/0.3333333333/
DATA OUTPUT/L"OUTPUT"/
C
C *** CHECK FOR A SUCCESSFUL TIME STEP
C
C IF (SUCCES .EQ. 0) GO TO 11
C
C --- UNSUCCESSFUL...RESET NEW TIME TERMS TO OLD TIME TERMS
C
C IF (SHIFT(ACCON(I),2).GE.0 .OR. SHIFT(ACCON(I),5).LT.0) GO TO 10
QUALA(K) = 0.0
QUALAG(K) = 0.0
VOIDF(K) = 1.0
VOIDG(K) = 0.0
10 CONTINUE
ACCON(I) = (.NOT.MASK(3) .AND. ACCON(I)) .OR.
* SHIFT(SHIFT(MASK(3),57) .AND. ACCON(I), 3)
GO TO 13
11 CONTINUE
C
C --- SUCCESSFUL...RESET OLD TIME TERMS TO NEW TIME TERMS
C
VDMO(I) = VDM(I)
RHONO(I) = RHON(I)
TVAPO(I) = TEMPG(K)
VLIQO(I) = VLIQ(I)
TTANKO(I) = TTANK(I)

```

Fig. B.1 An example of source program of CDC version.

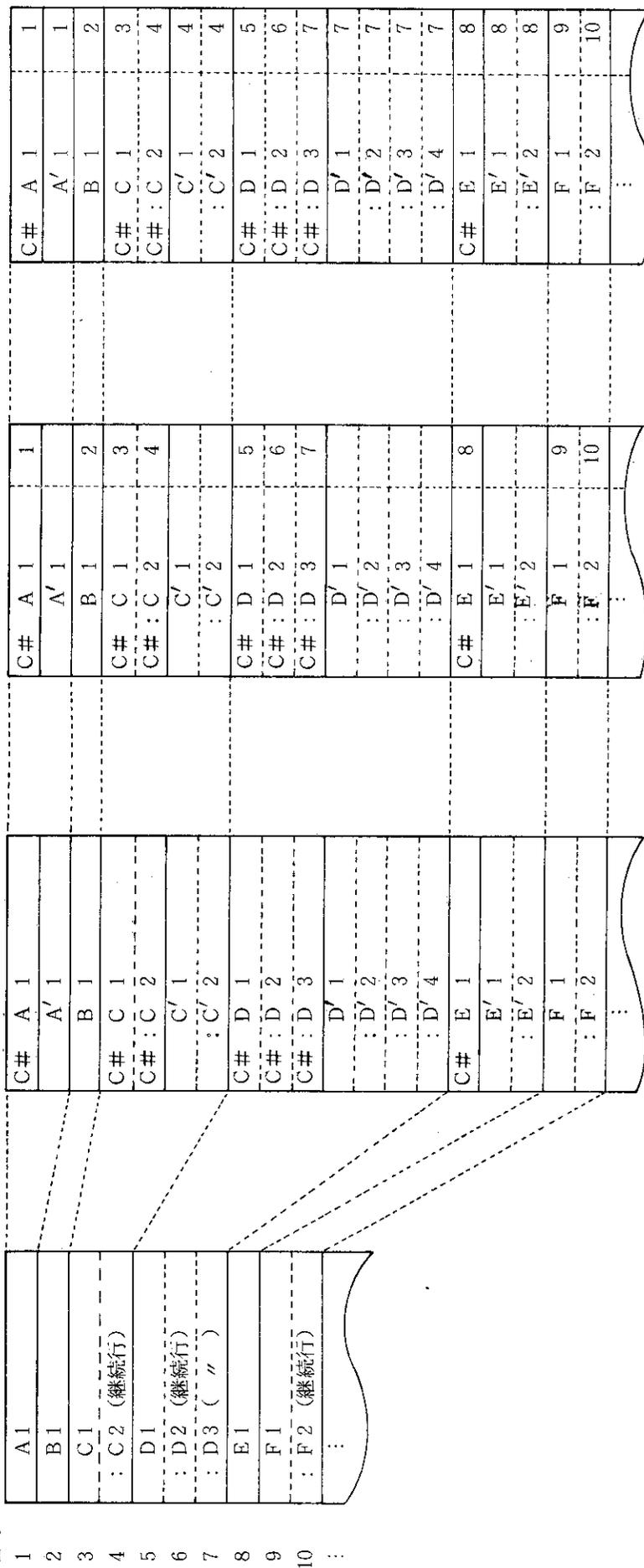
```

SUBROUTINE ACCUM(I,K,L,J,JJ)                                00000010
  IMPLICIT REAL*8 ( A-H,O-Z )                               00000020
  *INCLUDE %MASKD                                           C36VP
  *INCLUDE %FUNC                                             00000030
  REAL*8                                                     00000040
  DATA                                                     %B0001
  DATA                                                     %B0001 / Z 0000 0000 0000 0000 / 00000050
C
C ACCUMULATOR MASS, WALL HEAT TRANSFER AND MOMENTUM MODEL 00000060
C
C COGNIZANT ENGR: DMK                                       00000070
  *INCLUDE CMPDAC                                           00000080
  *INCLUDE CMPDAT                                           00000090
  *INCLUDE COMCTL                                           00000100
  *INCLUDE CONTRL                                           00000110
  *INCLUDE FAST                                             00000120
  *INCLUDE %IO                                              00000130
  *INCLUDE GENRL                                            00000140
  *INCLUDE JUNDAT                                           00000150
  *INCLUDE SCRTECH                                          00000160
  *INCLUDE STATEC                                           00000170
  *INCLUDE TRNHLP                                          00000180
  *INCLUDE VOLDAT                                           00000190
C
C *** LOCAL VARIABLES                                       00000200
C
C! INTEGER OUTPUT                                           00000210
CDC DATA GRAV/9.80665/, PIE/3.141592654/, XN/0.3333333333/ 00000220
CDC DATA GRAV/9.80665D0/, PIE/3.141592654D0/, XN/0.3333333333D0/ 00000230
C! DATA OUTPUT/6LOUTPUT/                                    00000240
C
C *** CHECK FOR A SUCCESSFUL TIME STEP                       00000300
C
C IF (SUCCES .EQ. 0) GO TO 11                               00000310
C
C --- UNSUCCESSFUL...RESET NEW TIME TERMS TO OLD TIME TERMS 00000320
C
CDC IF (SHIFT(ACCON(I),2).GE.0 .OR. SHIFT(ACCON(I),5).LT.0) GO TO 10 00000330
CDC IF ( %GE( %SFT(ACCON(I),2), %B0001 ).OR. %LT( %SFT(ACCON(I),5), 00000340
  * %B0001)) GO TO 10                                       00000350
CDC QUALA(K) = 0.0                                          00000360
CDC QUALA(K) = 0.0D0                                        00000370
CDC QUALAD(K) = 0.0                                         00000380
CDC QUALAD(K) = 0.0D0                                       00000390
CDC VOIDF(K) = 1.0                                          00000400
CDC VOIDF(K) = 1.0D0                                        00000410
CDC VOIDG(K) = 0.0                                          00000420
CDC VOIDG(K) = 0.0D0                                       00000430
  10 CONTINUE                                               00000440
CDC ACCON(I) = (.NOT.MASK(3) .AND. ACCON(I)) .OR.          00000450
CDC * SHIFT(SHIFT(MASK(3),57) .AND. ACCON(I), 3)           00000460
CDC ACCON(I) = %OR( %AND( %MKN3,ACCON(I)) , %SFT( %A00000550
  *ND( %SFT( %MKP3,57),ACCON(I)),3))                       00000560
CDC GO TO 13                                                00000570
  11 CONTINUE                                               00000580
C
C --- SUCCESSFUL...RESET OLD TIME TERMS TO NEW TIME TERMS 00000590
C
C
C VDMO(I) = VDM(I)                                         00000600
C RHONO(I) = RHON(I)                                       00000610
C TVAPO(I) = TEMPG(K)                                       00000620
C VLIQO(I) = VLIQ(I)                                        00000630
C TTANKO(I) = TTANK(I)                                     00000640

```

Fig. B.2 An example of source program of FACOM version converted from the program of Fig. B.1

シーケンス番号



(a) CDC 版

(b) FACOM 版

(c)

(d) シーケンス番号付き

注) a' は, CDC 版ソース a に
対応する FACOM 版ソースである。

Fig. B.3 Method of sequential numbering for source
program of FACOM version

B.2 実行方法

SEQNUM プログラムの主な入力は、旧 CDC 版（修正前）ソースと、それに対応する FACOM 版ソースである。

その他の入力として、実行時オプションがある。旧 CDC 版ソースは CHKSOC データセットから、旧 FACOM 版ソースは INSOC データセットから、実行時オプションは EXEC 文の PARM パラメータ、又は SYSIN データセットから入力される。Fig.B.4に入出力の概略を示す。

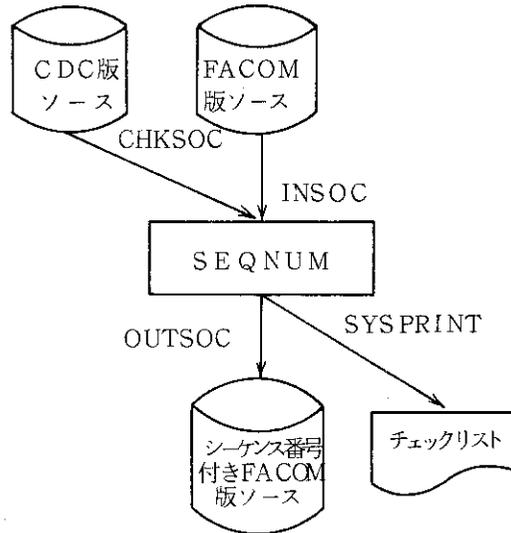


Fig. B.4 INPUT/OUTPUT configuration for SEQNUM.

B.2.1 SEQNUM の実行

SEQNUM プログラムを実行するためには、1つの EXEC 文といくつかの DD 文を用意しなければならない。EXECには通常、TABLE B.1に示すパラメータを指定すれば良い。PARM パラメータの指定内容の詳細は、B.2.2実行時オプションを参照されたい。実行時に必要な DD 文は、TABLE B.2で示されるものである。実行時のジョブ制御文の例をFig.B.5に示す。

TABLE.B.1 Main parameters of EXEC statements of JCL for SEQNUM.

PGM	プログラム名SEQNUMを指定する。このパラメータは必須である。
PARM	実行時のオプションを指定する。

TABLE.B.2 DD statements of JCL for SEQNUM.

DD名	必要の有無
CHKSOC	必ず用意する。
INSOC	必ず用意する。
OUTSOC	必ず用意する。
SYSPRINT	印刷情報を見たい場合に必要
STEPLIB	必ず用意する。

```
// EXEC PGM=SEQNUM
//STEPLIB DD DSN=J████.R5TOOL.LOAD,DISP=SHR
//CHKSOC DD DSN=J████.R5CDC00.FORT77,DISP=SHR,LABEL=(,///,IN)
//INSOC DD DSN=J████.R5FCM00.FORT77,DISP=SHR
//OUTSOC DD DSN=J████.R5FCM00W.FORT77,DISP=SHR
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=137,BLKSIZE=6850)
```

Fig. B.5 An example of JCL for SEQNUM.

B.2.2 実行時オプション

TABLE B.3 に実行時オプションを示す。これらは、EXEC 文の PARM パラメータ又は SYSIN データセットの先頭（制御データの前）に指定できる。複数個の実行時オプションを指定する場合は、各々のオプションをコンマ又は空白で区切って指定する。相反する複数のオプションが指定された場合は最後のオプションが有効となる。指定されなかったオプションの値は、標準値が取られる。

TABLE.B.3 Options of JCL for SEQNUM.

オプションの形式		
ELM (mem1 [-mem2] [,mem1 [-mem2]] ...)		区分データセット内の対象とするメンバ名の指定
ELM (*)		区分データセット内のすべてのメンバを対象とする。
IDENT (識別名)	<u>NOIDENT</u>	アイデント名を付けたい場合に指定する
LINECOUNT ((<u>60</u> n))		印刷情報の 1 ページあたりの行数の指定
LC ((<u>60</u> n))		(省略形)
REPLACE	<u>NOREPLACE</u>	既に存在するメンバを置き換えるか否かの指定
REP	<u>NOREP</u>	(省略形)
SEQNO [((<u>73</u> n1) , (<u>80</u> n2))]		シーケンス番号を付けるカラムの指定
SEQ [((<u>73</u> n1) , (<u>80</u> n2))]		省略値

注) 下線はオプション省略時に取られる標準値である。

*) LINECOUNT オプションは、現在、印刷情報としてエラーメッセージしか出力しないため意味を持たない

次に実行時オプションの詳細を説明する。

ELM (mem1 [-mem2] [,mem3 [-mem4]])

ELM (*)

ソースの入力を区分データセットから行う際のメンバ名を指示するものである。mem1, mem2, はソースが存在するメンバの名前であり、最大100組指定できる。メンバの指定方法には、直接メンバ名を指定する方法と、範囲で指定する方法がある。直接指定する方法は、対象とするメンバ名をコンマで区切って並べる。範囲で指定する方法は、'—'の前後にメンバ名を書く。始ま

りと終りを示す名前は、必ずしも実在の名前でなくても良い。範囲指定の場合、EBCDICコードによる大小判断で行う。その際、始まりを示すメンバ名は後ろに LOW-VALUE を補い、終りを示すメンバ名には HIGH-VALUE を補う。また、ELM (*) を指定した場合は、全メンバが対象となる。

(例)
ELM (A-C, Z) 先頭一文字がAまたはBまたはCで始まるメンバと、名前がZで始まるメンバが対象となる

順データセットの場合、この指定は意味を持たない。区分データセットのとき、このオプションを省略した場合は、ELM (*) が指定されたものとみなされる。メンバの処理は、EBCDICコード順に行われる。

IDENT (識別名)

NOIDENT

プログラム識別名を付加する際に指定する。シーケンス番号領域の左から文字数ぶん使用する為、プログラム識別名の文字数はシーケンス番号領域のそれより少なくなればならない。

NOIDENT は、プログラム識別名を付けないことを意味する。すなわち、シーケンス番号領域のすべてがシーケンス番号となる。

このオプションを省略した場合は、NOIDENT が指定されたものとみなされる。

(例)
IDENT (EXA) ……この例では、シーケンス番号領域を73~80とした場合には、Fig.B.6のような形式となる。

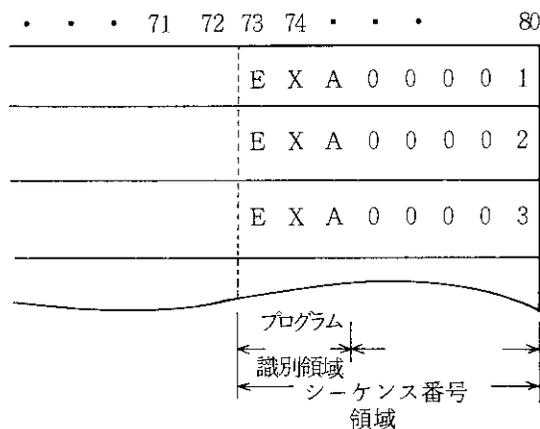


Fig. B.6 Method of storing name to the area in order to identify program in FORTRAN statement.

LINECOUNT (60 | n) または LC ({ 60 | n })

印刷情報を出力するとき、1 ページ当りの行数を指定するものである。特に、n = 0 のときは、印刷情報は全てベタ打ちとなる。省略時は、60 が指定されたものとみなします。

ただし、現在 SEQNUM プログラムは、印刷情報として実行時のエラーメッセージしか出力しないので、このオプションは実際には意味を持たない。

REPLACE または REP

NOREPLACE または NOREP

出力データセットに既に同じメンバ名が存在している場合に、それと置き換えるかどうかを指定する。REPLACE は置き換えることを意味する。古いメンバは消去される。

NOREPLACE は置き換えないことを意味する。新しいメンバの処理は行われぬ。

標準値は NOREPLACE である。

SEQNO [({ 73 | n1 }, { 80 | n2 })] または SEQ [({ 73 | n1 }, { 80 | n2 })]

シーケンス番号を打つ際のカラム位置を指定する。通常 FORTRAN のシーケンス番号は、73~80カラムが用いられるが、これを任意の位置、任意のカラム数に設定可能なようにしたものである。

n1 はシーケンス番号領域の始めのカラム位置、n2 は、終りのカラム位置を示す。n1 は 1 以上であること、また n2 はレコード長を越えることはできない。

省略時は、SEQNUM (73, 80) が指定されたものとみなす。

B.2.3 実行時のデータセット

SEQNUM は、TABLE B.4 に示されている入出力装置上のデータセットに対してアクセスする。データセットを DD 文で定義する場合、基本的には定められた DD 名を使用する。DD 文では、DD 名、データセット名、装置情報、ボリューム情報、データセットの取り扱い及び DCB 情報等を記述する。データセットがカタログされている場合、DD 文のパラメータとして、データセット名 (DSNAME パラメータ) とデータセットの取り扱い (DISP パラメータ) だけを指定すればよい。

出力データセットの DCB 属性が与えられていない場合、SEQNUM はある規約に従って DCB 属性を割り当てる。

B.2.3.1 CHKSOC データセット

CDC 版ソースが格納されているデータセットである。このデータセットは、基本的には区分データセット (SELECTX プログラムの出力データセット) であるが、順データセット (区分データセットのメンバ指定も含む) も使用可能である。

DD 文の指定によって連結したデータセットを定義してもよい。この場合は、各データセットの DCB 属性は同じでなければならない。また、1 つのデータセットの処理が終わってから次のデータセットの処理を行う。

CHKSOC データセットは、入力ストリームの中でデータを定義してもよい。この場合は、データの最後に区切として区切り文を置く。

TABLE.B.4 Attributes of dataset in SEQNUM.

DD名	入出力装置	D C B 属性			
		DSORG	RECFM	LRECL	BLKSIZE
CHKSOC	直接アクセス装置 磁気テープ装置 カード読み取り装置	PO 又は PS	F,FB	80	80の整数倍
INSOC	直接アクセス装置 磁気テープ装置 カード読み取り装置	PO 又は PS	F,FB	80	80の整数倍
OUTSOC	直接アクセス装置 磁気テープ装置 カード穿孔装置	PO 又は PS	F,FB	80	80の整数倍
SYSPRINT	直接アクセス装置 磁気テープ装置 ラインプリンタ	PS	FBA	137 / 81	レコード長の整数倍
			VBA	137 / 81	レコード長 + 4 バイト
STEPLIB	直接アクセス装置	PO	U	0	任意

備考1) CHKSOC, INSOCは、入力ストリームから入力できる。SYSPRINTは、出力ストリームのSYSOUTクラスをラインプリンタとして使用できる。

備考2) CHKSOCとINSOCのDCB属性は、DSORGは一致しなくてはならない。

備考3) OUTSOCのDCB属性が与えられていない場合、INSOCのDCB属性が割り当てられる。

備考4) SYSPRINTは、端末装置に割り付けてもよい。

備考5) POデータセットは直接アクセス装置にのみ割り付けられる。

B.2.3.2 INSOC データセット

CHKSOCで定義されたCDC版ソースに対応するFACOM版ソースが格納されているデータセットである。このデータセットの編成はCHKSOCデータセットと同じでなければならない。

DD文の指定によって連結したデータセットを定義してもよい。この場合は、各データセットのDCB属性は同じでなければならない。また、一つのデータセットの処理が終わってから次のデータセットの処理を行う。

INSOCデータセットは、入力ストリームの中でデータを定義してもよい。この場合は、データの最後に区切り文を置く。

B.2.3.3 OUTSOC データセット

SEQNUMが出力するシーケンス番号付きのFACOM版ソースを格納するデータセットである。このデータセットが、エラー修正の際の基本となるソースである。このデータセットの編成はINSOCデータセットと同じでなければならない。

B.2.3.4 SYSPRINT データセット

実行の際の各種の印刷情報を出力するデータセットである。

SYSPRINT データセットは、通常システムの出カ装置を指定して (SYSOUT パラメータを用いる)、そのユニットレコード装置のクラスをラインプリンタ装置とする。

B.2.3.5 STEPLIB データセット

SEQNUM が登録されているロードモジュールデータセットである。

B.3 実行時の注意事項

シーケンス番号は、区分データセットの場合各メンバごとに1から順に付けられる。1メンバ中に複数のプログラム単位が入ってもそのまま連番が付けられる。これは、順データセットでも同様である。

また、処理はメンバ単位で行われるので、CDC 版データセットと FACOM 版データセットは同じ構造でなくてはならない。メンバ名が異なると処理の対象からはずされるので注意を要する。Fig.B.7, B.8にそれぞれ CDC 版ソースと対応するシーケンス番号付き FACOM 版ソースの例を示す。

```

SUBROUTINE ACCUM(I,K,L,J,JJ)
C
C ACCUMULATOR MASS, WALL HEAT TRANSFER AND MOMENTUM MODEL
C
C COGNIZANT ENGR: DMK
*INCLUDE CMPDAC
*INCLUDE CMPDAT
*INCLUDE COMCTL
*INCLUDE CONTRL
*INCLUDE FAST
*INCLUDE GENRL
*INCLUDE JUNDAT
*INCLUDE SCRATCH
*INCLUDE STATEC
*INCLUDE TRNHLP
*INCLUDE VOLDAT
C
C *** LOCAL VARIABLES
C
C INTEGER OUTPUT
DATA GRAV/9.80665/, PIE/3.141592654/, XN/0.3333333333/
DATA OUTPUT/L"OUTPUT"/
C
C *** CHECK FOR A SUCCESSFUL TIME STEP
C
C IF (SUCCES .EQ. 0) GO TO 11
C
C --- UNSUCCESSFUL...RESET NEW TIME TERMS TO OLD TIME TERMS
C
C IF (SHIFT(ACCON(I),2).GE.0 .OR. SHIFT(ACCON(I),5).LT.0) GO TO 10
QUALA(K) = 0.0
QUALAO(K) = 0.0
VOIDF(K) = 1.0
VOIDG(K) = 0.0
10 CONTINUE
ACCON(I) = (.NOT.MASK(3) .AND. ACCON(I)) .OR.
* SHIFT(SHIFT(MASK(3),57) .AND. ACCON(I), 3)
GO TO 13
11 CONTINUE
C
C --- SUCCESSFUL...RESET OLD TIME TERMS TO NEW TIME TERMS
C
VDMO(I) = VDMCI)
RHONO(I) = RHON(I)
TVAPO(I) = TEMPG(K)
VLIQO(I) = VLIQ(I)
TTANKO(I) = TTANK(I)
QTANKO(I) = QTANK(I)
BORONO(K) = BORON(K)
C
C ... CHECK IF ACCUMULATOR IS AN ACTIVE VOLUME
C
C IF (SHIFT(ACCON(I),2).LT.0 .OR. VLIQ(I).GT.0.0) GO TO 12
C
C ACCUMULATOR IS EMPTY OF LIQUID
C
ACCON(I) = SHIFT(MASK(2),59) .OR. ACCON(I)
QUALA(K) = 1.0
QUALAO(K) = 1.0
VOIDF(K) = 0.0
VOIDG(K) = 1.0
12 CONTINUE

```

Fig. B.7 An example of source program of CDC version.

```

SUBROUTINE ACCUM(I,K,L,J,JJ)
  IMPLICIT REAL*8 ( A-H,Q-Z )
  *INCLUDE %MASKD
  *INCLUDE %FUNC
  REAL*8          %B0001
  DATA          %B0001 / Z 0000 0000 0000 0000 /
C
C  ACCUMULATOR MASS, WALL HEAT TRANSFER AND MOMENTUM MODEL
C
C  COGNIZANT ENGR: DMK
  *INCLUDE CMPDAC
  *INCLUDE CMPDAT
  *INCLUDE COMCTL
  *INCLUDE CONTRL
  *INCLUDE FAST
  *INCLUDE %IO
  *INCLUDE GENRL
  *INCLUDE JUNDAT
  *INCLUDE SCRATCH
  *INCLUDE STATEC
  *INCLUDE TRNHLP
  *INCLUDE VOLDAT
C
C *** LOCAL VARIABLES
C
C!  INTEGER OUTPUT
CDC  DATA GRAV/9.80665/, PIE/3.141592654/, XN/0.3333333333/
     DATA GRAV/9.8066500/, PIE/3.14159265400/, XN/0.333333333300/
C!  DATA OUTPUT/L"OUTPUT"/
C
C *** CHECK FOR A SUCCESSFUL TIME STEP
C
C   IF (SUCCES .EQ. 0) GO TO 11
C
C --- UNSUCCESSFUL...RESET NEW TIME TERMS TO OLD TIME TERMS
C
CDC  IF (SHIFT(ACCON(I),2).GE.0 .OR. SHIFT(ACCON(I),5).LT.0) GO TO 10
     IF ( %GE( %SFT(ACCON(I),2), %B0001 ).OR. %LT( %SFT(ACCON(I),5),
  * %B0001)) GO TO 10
CDC  QUALA(K)   = 0.0
     QUALA(K)   = 0.000
CDC  QUALAO(K)  = 0.0
     QUALAO(K)  = 0.000
CDC  VOIDF(K)   = 1.0
     VOIDF(K)   = 1.000
CDC  VOIDG(K)   = 0.0
     VOIDG(K)   = 0.000
     10 CONTINUE
CDC  ACCON(I) = (.NOT.MASK(3) .AND. ACCON(I)) .OR.
CDC  * SHIFT(SHIFT(MASK(3),57) .AND. ACCON(I), 3)
     ACCON(I) = %OR( %AND( %MKN3,ACCON(I)) , %SFT( %A
  *ND( %SFT( %MKP3,57),ACCON(I)),3))
     GO TO 13
     11 CONTINUE
C
C --- SUCCESSFUL...RESET OLD TIME TERMS TO NEW TIME TERMS
C
  VDMO(I) = VOM(I)
  RHONO(I) = RHON(I)
  TVAPO(I) = TEMPG(K)
  VLIQO(I) = VLIQ(I)
  TTANKO(I) = TTANK(I)

```

Fig. B.8 An example of source program of FACOM version with the sequence number of source program of CDC version.

付録C FORTCOMP

C.1 機能

C.1.1 機能の概要

FORTCOMP プログラムは、修正前・修正後2つの CDC 版ソースを比較し FACOM 版エラー修正のためのアップデートカードを作成するプログラムである。

比較は、継続行も含めた文単位に行い、一文字でも異なっていればその文は変更されたとの判断を下す。変更部分について、チェックリスト及びアップデートカードの2系統に出力される。2つのソースの比較は、従来 SFCOMP が用いられていたが、出力がプリンタのみでありその形式もアップデートカードとして用いるためにはかなりの再編集が必要であった。また、データの扱いも順データセットのみであり、プログラム単位ごとに1メンバが作られている区分データセットでは、全メンバを比較するには大変な労力を必要としていた。

FORTCOMP プログラムではこれらは全て不要であり、時間と労力が軽減される。Fig.C. 1, Fig.C. 2 に新・旧2種類の CDC 版ソースの例, Fig.C. 3 にそれらによって作られるアップデートカードの例を示す。

また、区分データセットの全メンバについて一度で比較ができるので、SFCOMP の代用としても有効であると思われる。

C.1.2 アップデートカード

プログラムの修正・変更を行うための指示及びデータの集合をアップデートカードという。修正前のソースに対してアップデートカードの指示に従って、削除・挿入を行えば新しいソースが完成するわけである。Fig.C. 4 にアップデートカードの作成の考え方を示す。

アップデートカードのデータ部分を FACOM 版へコンバージョンし、CDC 版シーケンス番号の付いた FACOM 版ソースに対して修正を行えば、FACOM 版の新しいソースプログラムが完成する。

アップデートカードは、(1)*DELETE 制御文、(2)*INSERT 制御文、(3) 修正データの3種類のデータから構成されている。(1)は削除を指示するもの、(2)は挿入を指示するもの、(3)は挿入されるべき修正データである。

```

      SUBROUTINE SWAPLT(IFILE, ISIZE, LOLD, LNEW, NUMALL, NUMNDX, LENDX,
      *                 NAMNDX, NAMTYP, NUMNAM)
C
C *** SUBROUTINE TO SWAP PLOT AND PLOT COMPARISON DATA TABLE FILES FOR
C RESTART AND PLOT RUNS.
C
      INTEGER IFILE(1), NAMTYP(1), OUTPUT
      DATA OUTPUT/L"OUTPUT"/
C
C *** INITIAL TERMS
C
      IF (LNEW .GE. (LOLD + ISIZE)) GO TO 400
      NWORDS = NUMNAM / 10
      IF ((NWORDS * 10) .LT. NUMNAM) NWORDS = NWORDS + 1
      LTHIS = LOLD
      NEW = LNEW
C
C *** CHECK FOR REPLACEMENT, DELETION OR INSERTION
C
100 IF (IFILE(NEW+NUMNDX-1) .LE. IFILE(LTHIS+NUMNDX-1)) GO TO 200
      LTHIS = LTHIS + IFILE(LTHIS+LENDX-1)
      IF (LTHIS .LT. LNEW) GO TO 100
      GO TO 300
200 CONTINUE
      NSHFT = IFILE(NEW+LENDX-1)
      LSHFT = 0
      IF (IFILE(NEW+NUMNDX-1) .LT. IFILE(LTHIS+NUMNDX-1)) GO TO 210
      NSHFT = IFILE(NEW+LENDX-1) - IFILE(LTHIS+LENDX-1)
      LSHFT = IFILE(LTHIS+LENDX-1)
      IF (IFILE(NEW+NAMNDX-1) .NE. "DELETE") GO TO 220
      WRITE(OUTPUT, 2000) (NAMTYP(K), K=1,NWORDS),IFILE(NEW+NUMNDX-1),
      * "DELETED"
      NSHFT = -IFILE(LTHIS+LENDX-1)
      NUMALL = NUMALL - 1
      GO TO 221
220 WRITE(OUTPUT, 2000) (NAMTYP(K), K=1,NWORDS),IFILE(NEW+NUMNDX-1),
      * "REPLACED"
221 NUMALL = NUMALL - 1
      GO TO 212
210 WRITE(OUTPUT, 2000) (NAMTYP(K), K=1,NWORDS),IFILE(NEW+NUMNDX-1),
      * "INSERTED"
212 IF (NSHFT .EQ. 0) GO TO 230
C
C *** SHIFT OLD RECORD
C
      LFIRST = LTHIS + LSHFT
      LAST = LOLD - 1 + ISIZE
      LEND = LAST - LFIRST + 1
      IF (LEND .LE. 0) GO TO 230
      DO 211 I = 1, LEND
         IGET = MAX(0, (LAST + ISIGN(1, NSHFT)))
         * - MIN(0, (LFIRST + ISIGN(1, NSHFT)))
         * - (I - 1) * ISIGN(1, NSHFT)
         IPUT = IGET + NSHFT
         IFILE(IPUT) = IFILE(IGET)
211 CONTINUE
      ISIZE = ISIZE + NSHFT
      LNEW = LNEW + NSHFT
      NEW = NEW + NSHFT
230 IF (IFILE(NEW+NAMNDX-1) .EQ. "DELETE") GO TO 240
C
C *** REPLACE OR INSERT THE NEW RECORD
C
      LEND = IFILE(NEW+LENDX-1)
      DO 231 I = 1, LEND
         IFILE(LTHIS+I-1) = IFILE(NEW+I-1)
231 CONTINUE
240 NEW = NEW + IFILE(NEW+LENDX-1)
      IF (NEW .LT. (LOLD + ISIZE - 1)) GO TO 100
300 CONTINUE
C
C *** DELETE THE SWAPPED RECORDS
C
      NSHFT = LNEW - NEW
      IF (NSHFT .GE. 0) GO TO 400
      LAST = LOLD - 1 + ISIZE
      LEND = LAST - NEW + 1
      IF (LEND .LE. 0) GO TO 320
      DO 310 I = 1, LEND
         IFILE(NEW+I-1+NSHFT) = IFILE(NEW+I-1)
310 CONTINUE
320 ISIZE = ISIZE + NSHFT
400 RETURN
2000 FORMAT("O*****",3A10," NO ",I5," HAS BEEN ",A10)
      END

```

Fig. C.1 An example of old source program of CDC version.

```

      SUBROUTINE SWAPLT(IFILE, ISIZE, LOLD, LNEW, NUMALL, NUMNDX, LENDX,
      *                NAMNDX, NAMTYP)
C
C SWAP PLOT AND PLOT COMPARISON DATA TABLE FILES FOR RESTARTS AND PLOTS
C
C COGNIZANT ENGINEER: DMK
C
      CHARACTER *(*) NAMTYP
      INTEGER IFILE(1), OUTPUT
      LOGICAL DELETE
      DATA OUTPUT/L"OUTPUT"/
C
C *** INITIAL TERMS
C
      IF (LNEW .GE. (LOLD + ISIZE)) GO TO 400
      LTHIS = LOLD
      NEW = LNEW
C
C *** CHECK FOR REPLACEMENT, DELETION OR INSERTION
C
      100 DELETE = .FALSE.
      IF (IFILE(NEW+NUMNDX-1) .LE. IFILE(LTHIS+NUMNDX-1)) GO TO 200
      LTHIS = LTHIS + IFILE(LTHIS+LENDX-1)
      IF (LTHIS .LT. LNEW) GO TO 100
      GO TO 300
      200 CONTINUE
      NSHFT = IFILE(NEW+LENDX-1)
      LSHFT = 0
      IF (IFILE(NEW+NUMNDX-1) .LT. IFILE(LTHIS+NUMNDX-1)) GO TO 210
      NSHFT = IFILE(NEW+LENDX-1) - IFILE(LTHIS+LENDX-1)
      LSHFT = IFILE(LTHIS+LENDX-1)
      IF ((IFILE(NEW+NAMNDX-1) .NE. "DELETE") .AND.
      *   (IFILE(NEW+NAMNDX-1) .NE. "DISCARD")) GO TO 220
      WRITE(OUTPUT, 2000) NAMTYP, IFILE(NEW+NUMNDX-1), 'DELETED'
      NSHFT = -IFILE(LTHIS+LENDX-1)
      NUMALL = NUMALL - 2
      DELETE = .TRUE.
      GO TO 212
      220 WRITE(OUTPUT, 2000) NAMTYP, IFILE(NEW+NUMNDX-1), 'REPLACED'
      221 NUMALL = NUMALL - 1
      GO TO 212
      210 WRITE(OUTPUT, 2000) NAMTYP, IFILE(NEW+NUMNDX-1), 'INSERTED'
      212 IF (NSHFT .EQ. 0) GO TO 230
C
C *** SHIFT OLD RECORD
C
      LFIRST = LTHIS + LSHFT
      LAST = LOLD - 1 + ISIZE
      LEND = LAST - LFIRST + 1
      IF (LEND .LE. 0) GO TO 230
      DO 211 I = 1, LEND
      IGET = MAX(0, (LAST + ISIGN(1, NSHFT)))
      * - MIN(0, (LFIRST + ISIGN(1, NSHFT)))
      * - (I - 1) * ISIGN(1, NSHFT)
      IPUT = IGET + NSHFT
      IFILE(IPUT) = IFILE(IGET)
      211 CONTINUE
      ISIZE = ISIZE + NSHFT
      LNEW = LNEW + NSHFT
      NEW = NEW + NSHFT
      230 IF (DELETE) GO TO 240
C
C *** REPLACE OR INSERT THE NEW RECORD
C
      LEND = IFILE(NEW+LENDX-1)
      DO 231 I = 1, LEND
      IFILE(LTHIS+I-1) = IFILE(NEW+I-1)
      231 CONTINUE
      240 NEW = NEW + IFILE(NEW+LENDX-1)
      IF (NEW .LT. (LOLD + ISIZE - 1)) GO TO 100
      300 CONTINUE
C
C *** DELETE THE SWAPPED RECORDS
C
      NSHFT = LNEW - NEW
      IF (NSHFT .GE. 0) GO TO 400
      LAST = LOLD - 1 + ISIZE
      LEND = LAST - NEW + 1
      IF (LEND .LE. 0) GO TO 320
      DO 310 I = 1, LEND
      IFILE(NEW+I-1+NSHFT) = IFILE(NEW+I-1)
      310 CONTINUE
      320 ISIZE = ISIZE + NSHFT
      400 RETURN
C
C *** FORMATS
C
      2000 FORMAT('O***** ',A,' NO ',IS,' HAS BEEN ',A)
C
      END

```

Fig. C.2 An example of new source program of CDC version.

```

*DELETE      1 -      7
*INSERT      7
      SUBROUTINE SWAPLT(IFILE, ISIZE, LOLD, LNEW, NUMALL, NUMNDX, LENDX,00000001
      *          NAMNDX, NAMTYP)          00000002
C          00000003
C SWAP PLOT AND PLOT COMPARISON DATA TABLE FILES FOR RESTARTS AND PLOTS00000004
C          00000005
C COGNIZANT ENGINEER: DMK          00000006
C          00000007
      CHARACTER *(*) NAMTYP          00000008
      INTEGER IFILE(1), OUTPUT      00000009
      LOGICAL DELETE                 00000010
*DELETE     13 -     14
*DELETE     19 -     20
*INSERT     20
C
      100 DELETE      = .FALSE.          00000020
      IF (IFILE(NEW+NUMNDX-1) .LE. IFILE(LTHIS+NUMNDX-1)) GO TO 200 00000021
*DELETE     30 -     32          00000022
*INSERT     32
      IF ((IFILE(NEW+NAMNDX-1) .NE. "DELETE") .AND.          00000032
      *      (IFILE(NEW+NAMNDX-1) .NE. "DISCARD")) GO TO 220 00000033
      WRITE(OUTPUT, 2000) NAMTYP, IFILE(NEW+NUMNDX-1), 'DELETED' 00000034
*DELETE     34 -     38
*INSERT     38
      NUMALL      = NUMALL - 2          00000036
      DELETE      = .TRUE.             00000037
*DELETE     40 -     41
*INSERT     41
      220 WRITE(OUTPUT, 2000) NAMTYP, IFILE(NEW+NUMNDX-1), 'REPLACED' 00000039
      221 NUMALL      = NUMALL - 1     00000040
      GO TO 212                        00000041
      210 WRITE(OUTPUT, 2000) NAMTYP, IFILE(NEW+NUMNDX-1), 'INSERTED' 00000042
*DELETE     60 -     61
*INSERT     61
      230 IF (DELETE) GO TO 240        00000061
C          00000062
*DELETE     84 -     84
*INSERT     84
C          00000085
C *** FORMATS          00000086
C          00000087
      2000 FORMAT('0***** ',A,' NO ',I5,' HAS BEEN ',A) 00000088
C          00000089

```

Fig. C.3 An example of update-card.

シテス番号

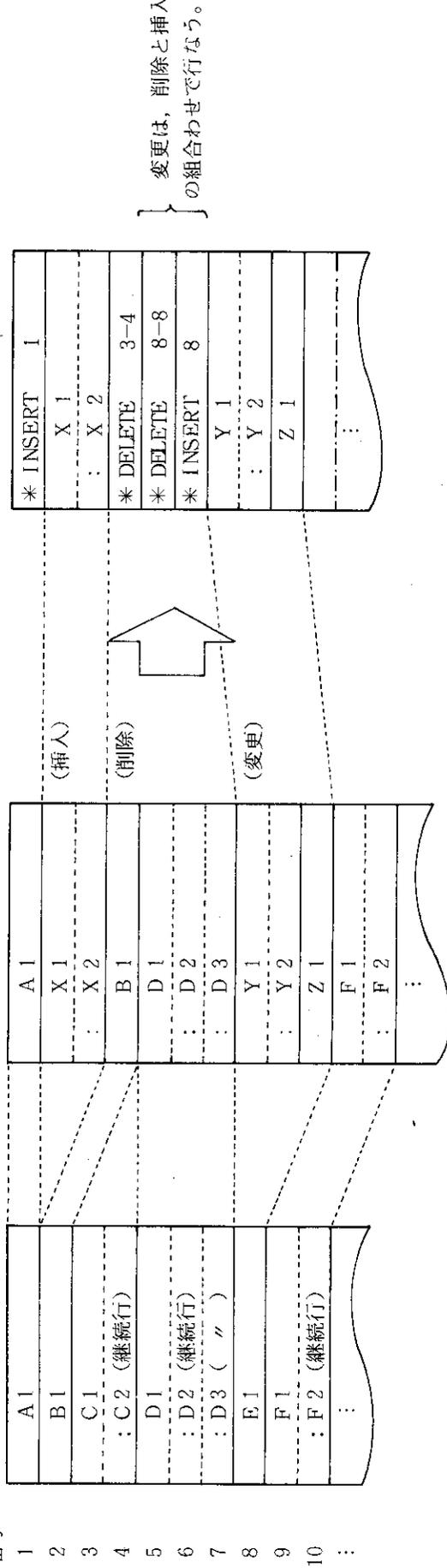


Fig. C.4 An example of making up update-card.

C.1.2.1 *DELETE 制御文

*DELETE 制御文の構成は次のとおりである。

1 2 3 4 5 6 7	8	13	1415	16	21
*DELETE		n1	-		n2

8～13カラム, 16～21カラムに削除範囲を示すシーケンス番号が入る。シーケンス番号がこの範囲にあるステートメントが削除される。1枚のみの削除は, n1 と n2 に同じシーケンス番号を入れる。

C.1.2.2 *INSERT 制御文

*INSERT 制御文の構成は次のとおりである。

1 2 3 4 5 6 7	8	13	
*INSERT		n1	

8～13カラムに示されたシーケンス番号のステートメントの直後に修正データが挿入される。

C.1.2.3 修正データ

これは, 通常の FORTRAN 文であり, *INSERT 制御文に続くものである。これらが, *INSERT 制御文の指示に従って FORTRAN ソースの中に挿入される。

FORTRAN 文の修正 (REPLACE) は, 削除と挿入を組合せて行う。つまり, *DELETE 制御文と, *INSERT 制御文が同じシーケンス番号を示すわけである。

このアップデートカードを用いて, 付録Dに示す UPDATE プログラムが, FACOM 版ソースを修正する。

C.1.3 FORTRAN ソース比較プログラムとしての利用

従来, 2種類の FORTRAN ソースを比較する際は, 前述したとおり, SFCOMP が用いられていた。FORTRAN プログラムの比較は, サブルーチンごとに行うのが好ましい。順データセットしか扱えない SFCOMP は, 複数のサブルーチンを比較するためには制御文を必要枚数そろえる必要があり, かなりの労力を必要とした。FORTCOMP は, 区分データセットの全メンバについて一度に比較ができるので, 実行に要する作業が大幅に軽減される。

なお, 区分データセットの場合の比較は, 同じ名前のメンバ同士で行い, サブルーチン名では行わない。

C.1.4 比較の方法

比較は, 2つのソースの先頭から順に行う。このとき, コメント文 (第1カラムが 'C' 又は '*' であるもの) は, 1行ずつ比較するが, 通常の FORTRAN 文については, 継続行も含めた文全体で比較を行う。継続行と継続行の間にコメント文が入っている場合, そのコメント文も含めたものを1文として比較を行う。

2つの文が一致したとは、文の行数、及び1～72カラムの内容全てが一致することを言う。つまり、変数名の位置が1カラムでもずれていれば、この2つの文は異なっていると判断する。

C.2 実行方法

FORTCOMP プログラムの主な入力は、2つの FORTRAN ソースである。その他の入力として実行時オプションがある。FORTRAN ソースは基本的には、旧 CDC 版ソースと新 CDC 版ソースの2つを指定する。旧 CDC 版ソースは OLDSOC データセットから、新 CDC 版ソースは NEWSOC データセットから入力される。また、実行時オプションは、EXEC 文の PARM パラメータから入力される。

FORTCOMP プログラムの主な出力は、アップデートカード、及び比較チェックリストである。アップデートカードは、UPDATECD データセットに、チェックリストは SYSPRINT データセットにそれぞれ出力される。

Fig.C.5 に入出力の概略を示す。

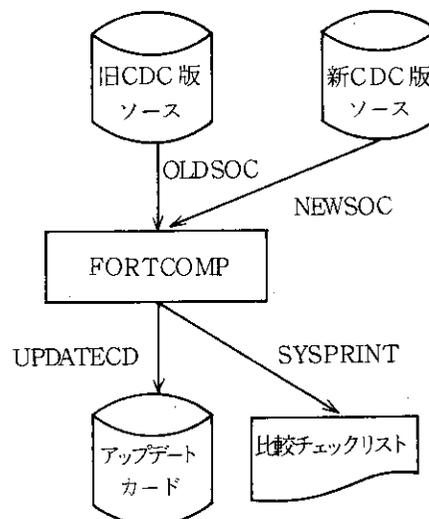


Fig. C.5 INPUT/OUTPUT configuration for FORTCOMP.

C.2.1 FORTCOMP の実行

FORTCOMP プログラムを実行するためには、1つの EXEC 文といくつかの DD 文を用意しなければならない。

EXEC 文には通常、TABLE.C.1 に示すパラメータを指定すればよい。PARM パラメータの指定内容の詳細は、C.2.2 実行時オプションを参照されたい。実行時に必要な DD 文は、TABLE.C.2 に示されるものである。この中には条件により用意しなくてもよいものがある。

実行時のジョブ制御文の例を Fig.C.6 に示す。

TABLE.C.1 Main parameters of EXEC statements of JCL for FORTCOMP.

PGM	プログラム名FORTCOMPを指定する。このパラメータは必須である。
PARM	実行時のオプションを指定する。

TABLE.C.2 DD statements of JCL for FORTCOMP.

DD名	必要の有無
OLDSOC	必ず用意する。
NEWSOC	必ず用意する。
UPDATECD	エラー修正を行う場合に必要
SYSPRINT	印刷情報を見たい場合に必要

```
//COMPARE EXEC PGM=FORTCOMP
//STEPLIB DD DSN=J████████.R5TOOL.LOAD,DISP=SHR
//OLDSOC DD DSN=J████████.R5CDC00.FORT77,DISP=SHR,LABEL=(,///,IN)
//NEWSOC DD DSN=J████████.R5CDC02.FORT77,DISP=SHR,LABEL=(,///,IN)
//UPDATECD DD DSN=J████████.@UPDTC.DATA,DISP=SHR
//SYSPRINT DD SYSOUT=*
// DCB=(LRECL=137,BLKSIZE=15344,RECFM=FBA,DSORG=PS)
```

Fig. C.6 An example of JCL for FORTCOMP.

C.2.2 実行時オプション

TABLE.C.3 に実行時オプションを示す。これらは、EXEC 文の PARM パラメータに指定できる。複数個の実行時オプションを指定する場合は、各々のオプションをコンマ又は空白で区切って指定する。相反する複数のオプションが指定された場合は最後のオプションが有効となる。指定されなかったオプションの値は、標準値が取られる。

TABLE.C.3 Options of JCL for FORTCOMP.

オプションの形式		
LINECOUNT ({60 n})		印刷情報の 1 ページあたりの行数の指定 (省略形)
LC ({60 n})		

注) 下線はオプション省略時に取られる標準値である。

次に実行時オプションの詳細を説明する。

LINECOUNT ({60 | n}) 又は LC ({60 | n})

印刷情報を出力するとき、1 ページ当りの行数を指定するものである。この行数には、FORTCOMP の見出し行も含まれる。特に、n=0 のときは、印刷情報は全てベタ打ちとなる。すなわち、見出し行は 1 回だけ出力され、その後は、編集のための改ページは行わない。省略時は、60 が指定されたものとみなす。

C.2.3 実行時のデータセット

FORTCOMP は TABLE.C. 4 に示されている入力装置上のデータセットに対してアクセスする。データセットを DD 文で定義する場合、基本的には定められた DD 名を使用する。DD 文では、DD 名、データセット名、装置情報、ボリューム情報、データセットの取り扱い及び DCB 情報等を記述する。データセットがカタログされている場合、DD 文のパラメータとして、データセット名 (DSNAME パラメータ) とデータセットの取り扱い (DISP パラメータ) だけを指定すればよい。

データセットの DCB 属性が与えられていない場合、FORTCOMP はある規約に従って DCB 属性を割り当てる。

TABLE.C.4 Attributes of dataset in FORTCOMP.

DD 名	入出力装置	D C B 属性			
		DSORG	RECFM	LRECL	BLKSIZE
OLDSOC	直接アクセス装置 磁気テープ装置 カード読み取り装置	P0 又は PS	F,FB	80	80の整数倍
NEWSOC	直接アクセス装置 磁気テープ装置 カード読み取り装置	P0 又は PS	F,FB	80	80の整数倍
UPDATECD	直接アクセス装置 磁気テープ装置 カード穿孔装置	P0 又は PS	F,FB	80	80の整数倍
SYSPRINT	直接アクセス装置 磁気テープ装置 ラインプリンタ	PS	FBA	137 / 81	レコード長の整数倍
			VBA	137 / 81	レコード長 + 4 バイト
STEPLIB	直接アクセス装置	P0	U	0	任意

備考 1) OLDSOC, NEWSOC は、入力ストリームから入力できる。SYSPRINT は、出力ストリームの SYSOUT クラスをラインプリンタとして使用できる。

備考 2) OLDSOC, NEWSOC と UPDATECD の DCB 属性は、DSORG は一致しなくてはならない。

備考 3) UPDATECD の DCB 属性が与えられていない場合、OLDSOC の DCB 属性が割り当てられる。

備考 4) SYSPRINT は、端末装置に割り付けてもよい。

備考 5) P0 データセットは直接アクセス装置にのみ割り付けられる。

C.2.3.1 OLDSOC データセット

旧 CDC 版ソースが格納されているデータセットである。このデータセットは基本的には区分データセット (SELECTX の出力データセット) であるが、順データセット (区分データセットのメンバ指定も含む) も使用可能である。

C.2.3.2 NEWSOC データセット

新 CDC 版ソースが格納されているデータセットである。このデータセットの編成は、OLDSOC データセットと同じでなければならない。

C.2.3.3 UPDATECD データセット

FORTCOMP が出力するアップデートカードを格納するデータセットである。このデータセットが、UPDATE プログラムによる FACOM 版ソースの修正に用いられる。このデータセットの編成は、OLDSOC データセットと同じでなければならない。DCB 属性が与えられていない場合、OLDSOC データセットの DCB 属性が割り当てられる。

FORTCOMP プログラムを、SFCOMP として使用する場合、この DD 文は省略できる。

C.2.3.4 SYSPRINT データセット

実行の際の各種の印刷情報を出力するデータセットである。詳細は C.4 印刷情報を参照されたい。

SYSPRINT データセットは、通常システムの出力装置を指定して (SYSOUT パラメータを用いる)、そのユニットレコード装置のクラスをラインプリンタ装置とする。

C.2.3.5 STEPLIB データセット

FORTCOMP が登録されているロードモジュールデータセットである。

C.3 アップデートカード・データセットの形式

入力データセットが区分データセットの場合、アップデートカードも区分データセットに作られる。その際アップデートカードのメンバ名は、比較を行ったソースのメンバ名が使われる。

OLDSOC データセットに存在し、NEWSOC データセットに存在しないメンバ、及び2つのソースが完全に一致するメンバに関するアップデートカードは出力されない。

OLDSOC データセットに存在せず、NEWSOC データセットに存在するメンバは、はじめに1枚シーケンス番号を0とした *INSERT 制御文が設定され、後はメンバそのものが続く形式のアップデートカードが作成される。Fig.C.7 にその例を示す。

このアップデートカードは、レコード長80バイトの固定長であり、修正データは FORTRAN ソースそのものである。また制御文は、第1カラムが、' * ' であるためコメント文として扱われるため、そのまま変換ツール (例えば STREAM 77 など) の入力とすることができる。変換対象となるデータは、修正データだけであり制御文には、一切影響がないため、変換後のアップデートカードも、そのまま、UPDATE プログラムの入力とすることができる。

C.4 印刷情報

FORTCOMP が出力する印刷情報には、データセット・チェックリスト、ソース比較チェックリスト、及びエラーメッセージの3種類がある。エラーメッセージの説明は、ここでは省略する。

C.4.1 データセット・チェックリスト

OLDSOC, NEWSOC, および UPDATECD 3 種類のデータセットのデータセット名, DCB 属性などを出力するものである。Fig.C.8 にその例を示す。

C.4.2 ソース比較チェックリスト

OLDSOC データセットと NEWSOC データセットを比較した際の不一致部分を出力するものである。形式は、アップデートカードの内容に、OLDSOC 側の内容も追加したものである。Fig.C.9 に通常の実出力例, Fig.C.10 に 2 つのソースが完全に一致した場合の実出力例, Fig.C.11 に NEWSOC データセットにメンバが存在しない場合の実出力例, Fig.C.12 に OLDSOC データセットにメンバが存在しない場合の実出力例を示す。

DATE 1986.05.06 TIME 15:32:47 PAGE 0001

=== SOURCE FILE COMPARE TOOL ===

L I S T I N G O F I N F O R M A T I O N

OLD SOURCE FILE DD NAME : OLDSOC
FILE NAME : J9303.R5CDC00.FORT77
LRECL : 80
BLKSIZE : 3120
RECFM : FB
DSORG : PO

NEW SOURCE FILE DD NAME : NEWSOC
FILE NAME : J9303.R5CDC02.FORT77
LRECL : 80
BLKSIZE : 3120
RECFM : FB
DSORG : PO

UPDATE CARD DECK DD NAME : UPDATECD
FILE NAME : J9303.0UPDTCD.DATA
LRECL : 80
BLKSIZE : 3120
RECFM : FB
DSORG : PO

Fig. C.8 An example of check list for dataset in FORTCOMP.

== SOURCE FILE COMPARE TOOL == CONDENSED DATE 1986.05.06 TIME 15:32:47 PAGE 0008

```

*****
** CONDENSED
*****
*DELETE 40 - 42
(( HTURB=0.065*THCONF(IV)*SQRT(RHF)/VISCF(IV)*
(( 1 SQRT(0.0395*VISCF(IV)*CSUBPF(IV)*RHG*VELG(IV)*VELG(IV)/
(( 2 (THCONF(IV)*REYG)**0.25)
*INSERT 42
(( HTURB=0.065*THCONF(IV)*SQRT(RHF)/VISCF(IV)*
(( 1 SQRT(0.0395*VISCF(IV)*CSUBPF(IV)*RHG*VELG(IV)*VELG(IV)/
(( 2 (THCONF(IV)*REYG**0.25))
*INSERT 57
(( GR=9.8*BETAFF(IV)*ABS(TW-TEMPF(IV))*(RHOF(IV)/VISCF(IV))**2
(( *HTHDMO(LS)**3
(( HTCF=AMAX1(HTCF,0.021*(THCONF(IV)/HTHDMO(LS)))*
(( * (VISCF(IV)*CSUBPF(IV)/THCONF(IV)*GR)**0.4*VOIDF(IV))
*DELETE 70 - 71
(( GAMW = QFGO*HTSRFO(LS)/(V(IV))*(HFG+0.375*CSUBPG(IV))*
(( 1 AMAX1(0.0, TEMPG(IV)-SATI(IV)))
*INSERT 71
(( GAMW=QFGO*HTSRFO(LS)*RECIPV(IV)/(HFG+0.375*CSUBPG(IV))*
(( * AMAX1(0.0, TEMPG(IV)-SATI(IV)))
*****
DD NAME = NEWSOC 87 (TOTAL) 82 (SAME) 5 (DIF)
DD NAME = NEWSOC 91 (TOTAL) 82 (SAME) 9 (DIF)

```

Fig. C.9 An example of check list for comparison between old source program and new source program when old one is different from new one.

=== SOURCE FILE COMPARE TOOL === ACCUM DATE 1986.05.06 TIME 15:32:47 PAGE 0002

 ** ACCUM **

```
<< ALL SAME STATEMENT >>
*****
NUMBER OF CARD *****
DD NAME = OLDSOC 364 (TOTAL) 364 (SAME) 0 (DIF)
DD NAME = NEWSOC 364 (TOTAL) 364 (SAME) 0 (DIF)
```

Fig. C.10 An example of check list for comparison between old source program and new source program when old one is exactly the same as new one.

=== SOURCE FILE COMPARE TOOL === CHRINT DATE 1986.05.06 TIME 15:32:47 PAGE 0007

 ** CHRINT **

CREATED MEMBER

```
*****
NUMBER OF CARD *****
DD NAME = OLDSOC 0 (TOTAL) 0 (SAME) 0 (DIF)
DD NAME = NEWSOC 146 (TOTAL) 0 (SAME) 146 (DIF)
```

Fig. C.11 An example of check list for comparison between old source program and new source program when there is no member in NEWSOC dataset.

 ** PLOT3D **

DELETED MEMBER

 ** PLTCOM **

DELETED MEMBER

Fig. C.12 An example of check list for comparison between old source program and new source program when there is no member in OLDSOC dataset.

付録D UPDATE

D.1 機能

D.1.1 機能の概要

UPDATE プログラムは、FORTCOMP により作成されたアップデートカード (FACOM 版へのコンバージョン済みのものも含む) と、CDC シーケンス番号付き FACOM 版ソースを入力として、アップデート後 FACOM 版ソースを作成するプログラムである。

ソースの73~80カラムに付けられているシーケンス番号をチェックし、アップデートカードの *DELETE 制御文が示すシーケンス番号に一致するソースは出力しない。*INSERT 制御文が示すシーケンス番号のついたソースの後に修正データを挿入する。Fig.D. 1 ~ Fig.D. 3 にそれぞれ、アップデートカード、旧 FACOM 版ソース及びアップデート後 FACOM 版ソースの例を示す。さらにこのソースを修正し、新 CDC 版ソースに対応した新 FACOM 版ソースができるわけである。Fig.D. 4 にその例を示す。アップデートカードの内容については D.1.2 アップデート作業の考え方を参照されたい。

D.1.2 アップデート作業の考え方

FACOM 版ソース・プログラムに付けられているシーケンス番号及びアップデートカード内の制御文が示すシーケンス番号は、それぞれ昇順に並んでいなければならない。アップデート作業はこのシーケンス番号によるシーケンシャルマッチングによって行われる。Fig.D. 5 に考え方の例を示す。アップデートカードのコンバージョンは、それだけで正しく行われるとは限らない。前後関係を調べた上で手作業で行う必要がある。そのため、アップデート後 FACOM 版ソースをエディタを用いて正しく修正する。これで新 FACOM 版ソースが完成する。

```

*DELETE      1 -      7
*INSERT      7
      SUBROUTINE SWAPLT(IFILE, ISIZE, LOLD, LNEW, NUMALL, NUMNDX, LENDX,00000001
      *          NAMNDX, NAMTYP)                                00000002
      IMPLICIT REAL*8 ( A-H,O-Z )                               00000002
C                                                    00000003
C SWAP PLOT AND PLOT COMPARISON DATA TABLE FILES FOR RESTARTS AND PLOTS00000004
C                                                    00000005
C COGNIZANT ENGINEER: DMK                                     00000006
C                                                    00000007
*INCLUDE %IO                                                00000006
      CHARACTER *(*) NAMTYP                                     00000003
C!! INTEGER IFILE(1), OUTPUT                                00000009
CDC INTEGER IFILE(1)                                        00000009
C!! REAL*8 IFILE(1)                                         00000009
      INTEGER IFILE(2,1)                                       00000009
      LOGICAL DELETE                                           00000010
*DELETE      13 -     14
*DELETE      19 -     20
*INSERT      20
C                                                    00000020
      100 DELETE = .FALSE.                                       00000021
CDC IF (IFILE(NEW+NUMNDX-1) .LE. IFILE(LTHIS+NUMNDX-1)) GO TO 200 00000022
      IF (IFILE(2,NEW+NUMNDX-1) .LE. IFILE(2,LTHIS+NUMNDX-1)) GO TO 200 00000022
*DELETE      30 -     32
*INSERT      32
CDC IF ((IFILE(NEW+NAMNDX-1) .NE. "DELETE") .AND.              00000032
CDC * (IFILE(NEW+NAMNDX-1) .NE. "DISCARD")) GO TO 220          00000033
C!! IF ((IFILE(NEW+NAMNDX-1) .NE. 'DELETE') .AND.             00000032
C!! * (IFILE(NEW+NAMNDX-1) .NE. 'DISCARD')) GO TO 220         00000033
      IF ((IFILE(2,NEW+NAMNDX-1) .NE. 'DELETE') .AND.         00000032
      * (IFILE(2,NEW+NAMNDX-1) .NE. 'DISCARD')) GO TO 220     00000033
CDC WRITE(OUTPUT, 2000) NAMTYP, IFILE(NEW+NUMNDX-1), 'DELETED' 00000034
      WRITE(OUTPUT, 2000) NAMTYP, IFILE(2,NEW+NUMNDX-1), 'DELETED' 00000034
*DELETE      34 -     38
*INSERT      38
      NUMALL = NUMALL - 2                                       00000036
      DELETE = .TRUE.                                           00000037
*DELETE      40 -     41
*INSERT      41
CDC20 WRITE(OUTPUT, 2000) NAMTYP, IFILE(NEW+NUMNDX-1), 'REPLACED' 00000039
      220 WRITE(OUTPUT, 2000) NAMTYP, IFILE(2,NEW+NUMNDX-1), 'REPLACED' 00000039
      221 NUMALL = NUMALL - 1                                     00000040
      GO TO 212                                                  00000041
CDC10 WRITE(OUTPUT, 2000) NAMTYP, IFILE(NEW+NUMNDX-1), 'INSERTED' 00000042
      210 WRITE(OUTPUT, 2000) NAMTYP, IFILE(2,NEW+NUMNDX-1), 'INSERTED' 00000042
*DELETE      60 -     61
*INSERT      61
      230 IF (DELETE) GO TO 240                                  00000061
C                                                    00000062
*DELETE      84 -     84
*INSERT      84
C                                                    00000085
C *** FORMATS                                                  00000086
C                                                    00000087
C 2000 FORMAT('O*****',A,' NO ',I5,' HAS BEEN ',A)           00000088
C                                                    00000089

```

Fig. D.1 An example of update-card.

```

SUBROUTINE SWAPLT(IFILE, ISIZE, LOLD, LNEW, NUMALL, NUMNDX, LENDX,00000001
*      NAMNDX, NAMTYP, NUMNAM) 00000002
      IMPLICIT REAL*8 ( A-H,O-Z ) 00000002
C 00000003
C *** SUBROUTINE TO SWAP PLOT AND PLOT COMPARISON DATA TABLE FILES FOR 00000004
C RESTART AND PLOT RUNS. 00000005
C 00000006
*INCLUDE *IO 00000006
C! INTEGER IFILE(1), NAMTYP(1), OUTPUT 00000007
CDC INTEGER IFILE(1) 00000008
      REAL*8 IFILE(1) 00000008
CDC REAL NAMTYP(1) 00000008
      REAL*8NAMTYP(1) 00000008
C! DATA OUTPUT/6LOUTPUT/ 00000008
C 00000009
C *** INITIAL TERMS 00000010
C 00000011
      IF (LNEW .GE. (LOLD + ISIZE)) GO TO 400 00000012
      NWORDS = NUMNAM / 10 00000013
      IF ((NWORDS + 1) .LT. NUMNAM) NWORDS = NWORDS + 1 00000014
      LTHIS = LOLD 00000015
      NEW = LNEW 00000016
C 00000017
C *** CHECK FOR REPLACEMENT, DELETION OR INSERTION 00000018
C 00000019
100 IF (IFILE(NEW+NUMNDX-1) .LE. IFILE(LTHIS+NUMNDX-1)) GO TO 200 00000020
      LTHIS = LTHIS + IFILE(LTHIS+LENDX-1) 00000021
      IF (LTHIS .LT. LNEW) GO TO 100 00000022
      GO TO 300 00000023
200 CONTINUE 00000024
      NSHFT = IFILE(NEW+LENDX-1) 00000025
      LSHFT = 0 00000026
      IF (IFILE(NEW+NUMNDX-1) .LT. IFILE(LTHIS+NUMNDX-1)) GO TO 210 00000027
      NSHFT = IFILE(NEW+LENDX-1) - IFILE(LTHIS+LENDX-1) 00000028
      LSHFT = IFILE(LTHIS+LENDX-1) 00000029
CDC IF (IFILE(NEW+NAMNDX-1) .NE. "DELETE") GO TO 220 00000030
      IF (IFILE(NEW+NAMNDX-1) .NE. 'DELETE') GO TO 220 00000030
CDC WRITE(OUTPUT, 2000) (NAMTYP(K), K=1,NWORDS), IFILE(NEW+NUMNDX-1), 00000031
CDC = "DELETED" 00000032
C! WRITE(OUTPUT, 2000) (NAMTYP(K), K=1,NWORDS), IFILE(NEW+NUMNDX-1), 00000032
C! = 'DELETED' 00000032
      WRITE(OUTPUT, 2000) (NAMTYP(IWK8), IWK8=1, NWORDS), IFILE(NEW+NUMNDX-1) 00000032
      =, 'DELETED' 00000032
      NSHFT = -IFILE(LTHIS+LENDX-1) 00000033
      NUMALL = NUMALL - 1 00000034
      GO TO 221 00000035
CDC20 WRITE(OUTPUT, 2000) (NAMTYP(K), K=1,NWORDS), IFILE(NEW+NUMNDX-1), 00000036
CDC = "REPLACED" 00000037
C!220 WRITE(OUTPUT, 2000) (NAMTYP(K), K=1,NWORDS), IFILE(NEW+NUMNDX-1), 00000037
C! = 'REPLACED' 00000037
      220 WRITE(OUTPUT, 2000) (NAMTYP(IWK8), IWK8=1, NWORDS), IFILE(NEW+NUMNDX-1) 00000037
      =, 'REPLACED' 00000037
      221 NUMALL = NUMALL + 1 00000038
      GO TO 212 00000039
CDC10 WRITE(OUTPUT, 2000) (NAMTYP(K), K=1,NWORDS), IFILE(NEW+NUMNDX-1), 00000040
CDC = "INSERTED" 00000041
C!210 WRITE(OUTPUT, 2000) (NAMTYP(K), K=1,NWORDS), IFILE(NEW+NUMNDX-1), 00000041
C! = 'INSERTED' 00000041
      210 WRITE(OUTPUT, 2000) (NAMTYP(IWK8), IWK8=1, NWORDS), IFILE(NEW+NUMNDX-1) 00000041
      =, 'INSERTED' 00000041
      212 IF (NSHFT .EQ. 0) GO TO 230 00000042
C 00000043
C *** SHIFT OLD RECORD 00000044
C 00000045
      LFIRST = LTHIS + LSHFT 00000046
      LAST = LOLD - 1 + ISIZE 00000047
      LEND = LAST - LFIRST + 1 00000048
      IF (LEND .LE. 0) GO TO 230 00000049
      DO 211 I = 1, LEND 00000050
          IGET = MAX(0, (LAST + ISIGN(1, NSHFT))) 00000051
          = MIN(0, (LFIRST + ISIGN(1, NSHFT))) 00000052
          = (I - 1) * ISIGN(1, NSHFT) 00000053
          IPUT = IGET + NSHFT 00000054
          IFILE(IPUT) = IFILE(IGET) 00000055
      211 CONTINUE 00000056
      ISIZE = ISIZE + NSHFT 00000057
      LNEW = LNEW + NSHFT 00000058
      NEW = NEW + NSHFT 00000059
CDC30 IF (IFILE(NEW+NAMNDX-1) .EQ. "DELETE") GO TO 240 00000060
      230 IF (IFILE(NEW+NAMNDX-1) .EQ. 'DELETE') GO TO 240 00000060
C 00000061
C *** REPLACE OR INSERT THE NEW RECORD 00000062
C 00000063
      LEND = IFILE(NEW+LENDX-1) 00000064
      DO 231 I = 1, LEND 00000065
          IFILE(LTHIS+I-1) = IFILE(NEW+I-1) 00000066

```

Fig. D.2 An example of old source program of FACOM version.

```

SUBROUTINE SWAPLT(IFILE, ISIZE, LOLD, LNEW, NUMALL, NUMNDX, LENDX, *0000001
*      NAMNDX, NAMTYP) *0000002
      IMPLICIT REAL*8 (A-H,O-Z) *0000003
C *0000004
C SWAP PLOT AND PLOT COMPARISON DATA TABLE FILES FOR RESTARTS AND PLOTS *0000005
C *0000006
C COGNIZANT ENGINEER: DMK *0000007
C *0000008
*INCLUDE *IQ *0000009
      CHARACTER *(*) NAMTYP *0000010
C!! INTEGER IFILE(1), OUTPUT *0000011
CDC INTEGER IFILE(1) *0000012
C!! REAL*8 IFILE(1) *0000013
      INTEGER IFILE(2,1) *0000014
      LOGICAL DELETE *0000015
CDC INTEGER IFILE(1) *0000016
      REAL*8 IFILE(1) *0000017
CDC REAL NAMTYP(1) *0000018
      REAL*8NAMTYP(1) *0000019
C! DATA OUTPUT/6OUTPUT/ *0000020
C *0000021
C *** INITIAL TERMS *0000022
C *0000023
      IF (LNEW .GE. (LOLD + ISIZE)) GO TO 400 *0000024
      LTHIS = LOLD *0000025
      NEW = LNEW *0000026
C *0000027
C *** CHECK FOR REPLACEMENT, DELETION OR INSERTION *0000028
C *0000029
      100 DELETE = .FALSE. *0000030
CDC IF (IFILE(NEW+NUMNDX-1) .LE. IFILE(LTHIS+NUMNDX-1)) GO TO 200 *0000031
      IF (IFILE(2,NEW+NUMNDX-1) .LE. IFILE(2,LTHIS+NUMNDX-1)) GO TO 200 *0000032
      LTHIS = LTHIS + IFILE(LTHIS+LENDX-1) *0000033
      IF (LTHIS .LT. LNEW) GO TO 100 *0000034
      GO TO 300 *0000035
      200 CONTINUE *0000036
      NSHFT = IFILE(NEW+LENDX-1) *0000037
      LSHFT = 0 *0000038
      IF (IFILE(NEW+NUMNDX-1) .LT. IFILE(LTHIS+NUMNDX-1)) GO TO 210 *0000039
      NSHFT = IFILE(NEW+LENDX-1) - IFILE(LTHIS+LENDX-1) *0000040
      LSHFT = IFILE(LTHIS+LENDX-1) *0000041
CDC IF ((IFILE(NEW+NAMNDX-1) .NE. 'DELETE') .AND. *0000042
* (IFILE(NEW+NAMNDX-1) .NE. 'DISCARD')) GO TO 220 *0000043
C!! IF ((IFILE(NEW+NAMNDX-1) .NE. 'DELETE') .AND. *0000044
* (IFILE(NEW+NAMNDX-1) .NE. 'DISCARD')) GO TO 220 *0000045
C!! IF ((IFILE(2,NEW+NAMNDX-1) .NE. 'DELETE') .AND. *0000046
* (IFILE(2,NEW+NAMNDX-1) .NE. 'DISCARD')) GO TO 220 *0000047
CDC WRITE(OUTPUT, 2000) NAMTYP, IFILE(NEW+NUMNDX-1), 'DELETED' *0000048
      WRITE(OUTPUT, 2000) NAMTYP, IFILE(2,NEW+NUMNDX-1), 'DELETED' *0000049
      NSHFT = -IFILE(LTHIS+LENDX-1) *0000050
      NUMALL = NUMALL - 2 *0000051
      DELETE = .TRUE. *0000052
      GO TO 212 *0000053
CDC20 WRITE(OUTPUT, 2000) NAMTYP, IFILE(NEW+NUMNDX-1), 'REPLACED' *0000054
      220 WRITE(OUTPUT, 2000) NAMTYP, IFILE(2,NEW+NUMNDX-1), 'REPLACED' *0000055
      221 NUMALL = NUMALL - 1 *0000056
      GO TO 212 *0000057
CDC10 WRITE(OUTPUT, 2000) NAMTYP, IFILE(NEW+NUMNDX-1), 'INSERTED' *0000058
      210 WRITE(OUTPUT, 2000) NAMTYP, IFILE(2,NEW+NUMNDX-1), 'INSERTED' *0000059
      212 IF (NSHFT .EQ. 0) GO TO 230 *0000060
C *0000061
C *** SHIFT OLD RECORD *0000062
C *0000063
      LFIRST = LTHIS + LSHFT *0000064
      LAST = LOLD - 1 + ISIZE *0000065
      LEND = LAST - LFIRST + 1 *0000066
      IF (LEND .LE. 0) GO TO 230 *0000067
      DO 211 I = 1, LEND *0000068
        IGET = MAX(0, (LAST + ISIGN(1, NSHFT))) *0000069
        * MIN(0, (LFIRST + ISIGN(1, NSHFT))) *0000070
        * (I - 1) * ISIGN(1, NSHFT) *0000071
        IPUT = IGET + NSHFT *0000072
        IFILE(IPUT) = IFILE(IGET) *0000073
      211 CONTINUE *0000074
      ISIZE = ISIZE + NSHFT *0000075
      LNEW = LNEW + NSHFT *0000076
      NEW = NEW + NSHFT *0000077
      230 IF (DELETE) GO TO 240 *0000078
C *0000079
C *** REPLACE OR INSERT THE NEW RECORD *0000080
C *0000081
      LEND = IFILE(NEW+LENDX-1) *0000082
      DO 231 I = 1, LEND *0000083
        IFILE(LTHIS+I-1) = IFILE(NEW+I-1) *0000084
      231 CONTINUE *0000085
      240 NEW = NEW + IFILE(NEW+LENDX-1) *0000086

```

Fig. D.3 An example of source program of FACOM version processed by UPDATE.

```

SUBROUTINE SWAPLT(IFILE, ISIZE, LOLD, LNEW, NUMALL, NUMNOX, LENDX,00000001
*      NAMNDX, NAMTYP)                                00000002
      IMPLICIT REAL*8 ( A-H,O-Z )                    00000002
C                                                    00000003
C SWAP PLOT AND PLOT COMPARISON DATA TABLE FILES FOR RESTARTS AND PLOTS00000004
C                                                    00000005
C COGNIZANT ENGINEER: DMK                            00000006
C                                                    00000007
*INCLUDE XIQ                                          00000006
      CHARACTER *(*) NAMTYP                          00000008
C!! INTEGER IFILE(1), OUTPUT                        00000009
CDC INTEGER IFILE(1)                                00000008
C!! REAL*8 IFILE(1)                                  00000008
      INTEGER IFILE(2,1)                              00000008
      INTEGER IWK1(2)                                  00000008
      REAL*8 DWK1
      EQUIVALENCE (IWK1(1),DWK1)
      LOGICAL DELETE                                  00000010
C: DATA OUTPUT/'OUTPUT'/                            00000008
C                                                    00000009
C *** INITIAL TERMS                                  00000010
C                                                    00000011
      IF (LNEW .GE. (LOLD + ISIZE)) GO TO 400         00000012
      LTHIS = LOLD                                    00000015
      NEW = LNEW                                       00000016
C                                                    00000017
C *** CHECK FOR REPLACEMENT, DELETION OR INSERTION 00000018
C                                                    00000020
      100 DELETE = .FALSE.                            00000021
CDC IF (IFILE(NEW+NUMNOX-1) .LE. IFILE(LTHIS+NUMNOX-1)) GO TO 200 00000022
      IF (IFILE(2,NEW+NUMNOX-1) .LE. IFILE(2,LTHIS+NUMNOX-1)) GO TO 200 00000022
CDC LTHIS = LTHIS + IFILE(LTHIS+LENDX-1)            00000021
      LTHIS = LTHIS + IFILE(2,LTHIS+LENDX-1)          00000021
      IF (LTHIS .LT. LNEW) GO TO 100                  00000022
      GO TO 300                                        00000023
      200 CONTINUE                                    00000024
CDC NSHFT = IFILE(NEW+LENDX-1)                       00000025
      NSHFT = IFILE(2,NEW+LENDX-1)                   00000025
      LSHFT = 0                                       00000026
CDC IF (IFILE(NEW+NUMNOX-1) .LT. IFILE(LTHIS+NUMNOX-1)) GO TO 210 00000027
      IF (IFILE(2,NEW+NUMNOX-1) .LT. IFILE(2,LTHIS+NUMNOX-1)) GO TO 210 00000027
CDC NSHFT = IFILE(NEW+LENDX-1) - IFILE(LTHIS+LENDX-1) 00000028
      NSHFT = IFILE(2,NEW+LENDX-1) - IFILE(2,LTHIS+LENDX-1) 00000028
CDC LSHFT = IFILE(LTHIS+LENDX-1)                    00000029
      LSHFT = IFILE(2,LTHIS+LENDX-1)                 00000029
CDC IF ((IFILE(NEW+NAMNDX-1) .NE. 'DELETE') .AND. 00000032
CDC = (IFILE(NEW+NAMNDX-1) .NE. 'DISCARD')) GO TO 220 00000033
C!! IF (IFILE(NEW+NAMNDX-1) .NE. 'DELETE') .AND. 00000032
C!! = (IFILE(NEW+NAMNDX-1) .NE. 'DISCARD')) GO TO 220 00000033
      IWK1(1) = IFILE(1,NEW+NAMNDX-1)
      IWK1(2) = IFILE(2,NEW+NAMNDX-1)
      IF (DWK1 .NE. 'DELETE') .AND. 00000032
      = (DWK1 .NE. 'DISCARD')) GO TO 220 00000033
CDC WRITE(OUTPUT, 2000) NAMTYP, IFILE(NEW+NUMNOX-1), 'DELETED' 00000034
      WRITE(OUTPUT, 2000) NAMTYP, IFILE(2,NEW+NUMNOX-1), 'DELETED' 00000034
CDC NSHFT = -IFILE(LTHIS+LENDX-1)                    00000033
      NSHFT = -IFILE(2,LTHIS+LENDX-1)                00000033
      NUMALL = NUMALL - 2                             00000036
      DELETE = .TRUE.                                 00000037
      GO TO 212                                       00000039
CDC20 WRITE(OUTPUT, 2000) NAMTYP, IFILE(NEW+NUMNOX-1), 'REPLACED' 00000039
      220 WRITE(OUTPUT, 2000) NAMTYP, IFILE(2,NEW+NUMNOX-1), 'REPLACED' 00000039
      221 NUMALL = NUMALL - 1                          00000040
      GO TO 212                                       00000041
CDC10 WRITE(OUTPUT, 2000) NAMTYP, IFILE(NEW+NUMNOX-1), 'INSERTED' 00000042
      210 WRITE(OUTPUT, 2000) NAMTYP, IFILE(2,NEW+NUMNOX-1), 'INSERTED' 00000042
      212 IF (NSHFT .EQ. 0) GO TO 230                  00000042
C                                                    00000043
C *** SHIFT OLD RECORD                              00000044
C                                                    00000045
      LFIRST = LTHIS + LSHFT                          00000046
      LAST = LOLD - 1 + ISIZE                          00000047
      LEND = LAST - LFIRST + 1                        00000048
      IF (LEND .LE. 0) GO TO 230                      00000049
      DO 211 I = 1, LEND                               00000050
          IGET = MAX(0, (LAST + ISIGN(1, NSHFT)))        00000051
          = MIN(0, (LFIRST + ISIGN(1, NSHFT)))          00000052
          = (I - 1) * ISIGN(1, NSHFT)                 00000053
          IPUT = IGET + NSHFT                          00000054
CDC IFILE(IPUT) = IFILE(IGET)                          00000055
      IFILE(1,IPUT) = IFILE(1,IGET)                  00000055
      IFILE(2,IPUT) = IFILE(2,IGET)                  00000055
      211 CONTINUE                                    00000056
      ISIZE = ISIZE + NSHFT                            00000057
      LNEW = LNEW + NSHFT                              00000058
      NEW = NEW + NSHFT                                00000059
      230 IF (DELETE) GO TO 240                        00000061

```

Fig. D.4 An example of new source program of FACOM version.

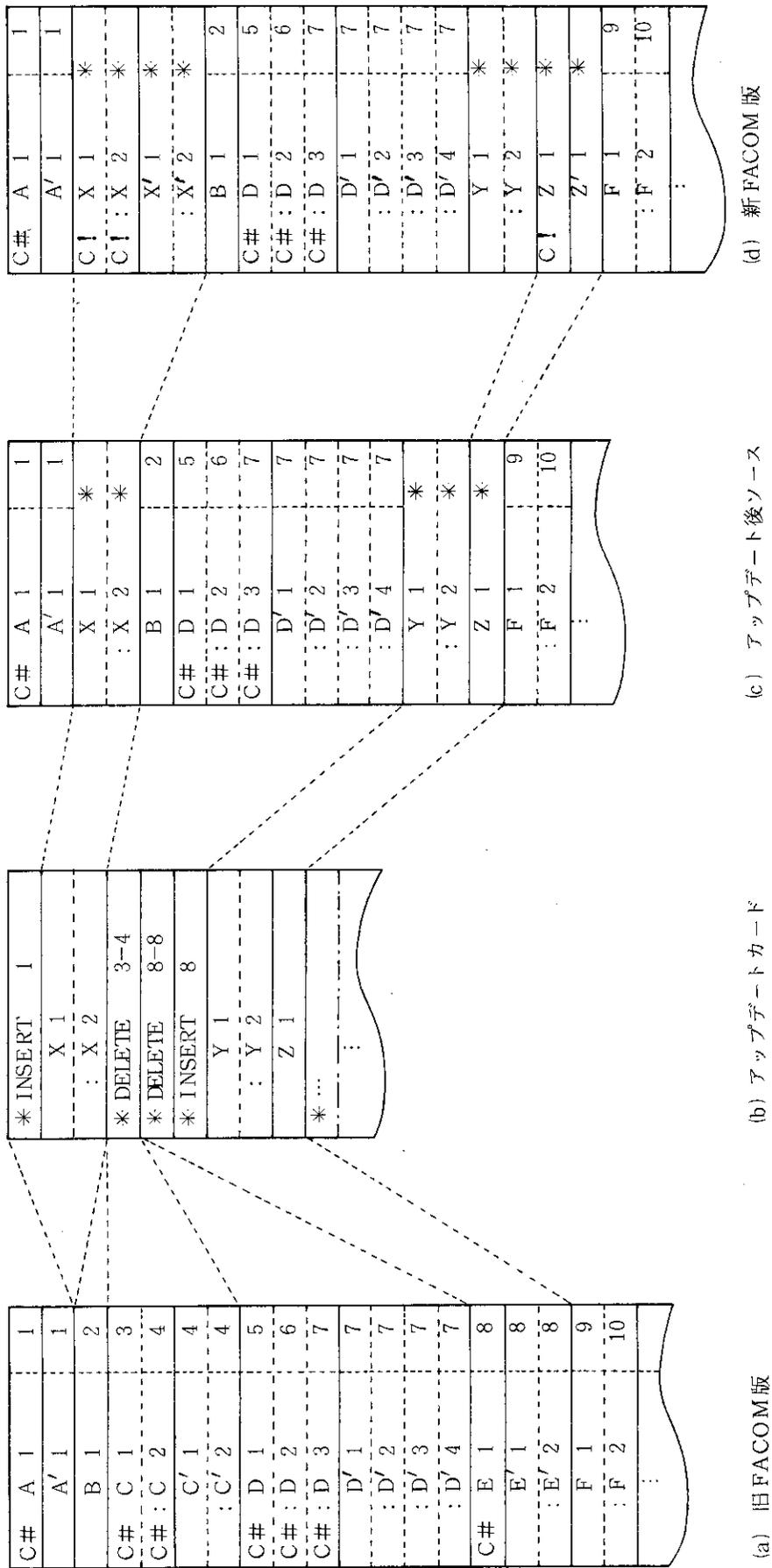


Fig. D.5 An example of process by UPDATE.

D.2 実行方法

UPDATE プログラムの主な入力は、CDC シーケンス番号付き旧 FACOM 版ソースとアップデートカードである。その他の入力として、実行時オプションがある。旧 FACOM 版ソースは OLDSOC データセットから、アップデートカードは UPDATECD データセットからそれぞれ入力される。また、実行時オプションは、EXEC 文の PARM パラメータから入力される。

UPDATE プログラムの主な出力は、アップデートカード後 FACOM 版ソースである。

Fig.D. 6 に入出力の概略を示す。

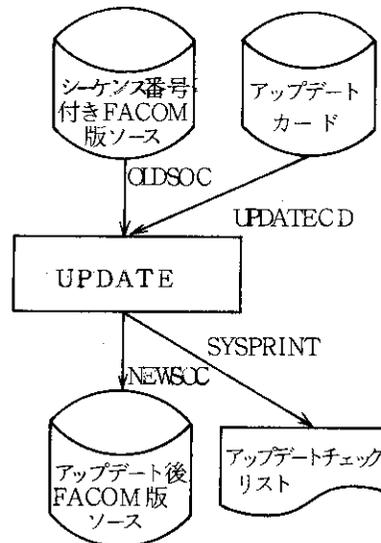


Fig. D.6 INPUT/OUTPUT configuration for UPDATE.

D.2.1 UPDATE の実行

UPDATE プログラムを実行するためには、1つの EXEC 文といくつかの DD 文を用意しなければならない。EXEC 文には通常、TABLE.D. 1 に示すパラメータを指定すればよい。PARM パラメータの指定内容の詳細は、D.2.2 実行時オプションを参照されたい。実行時に必要な DD 文は、TABLE.D. 2 に示されるものである。この中には条件により用意しなくてもよいものがある。

実行時のジョブ制御文の例を Fig.D. 7 に示す。

TABLE.D.1 Main parameters of EXEC statements of JCL for UPDATE.

PGM	プログラム名UPDATEを指定する。このパラメータは必須である。
PARM	実行時のオプションを指定する。

TABLE.D.2 DD statements of JCL for UPDATE.

DD名	必要の有無
UPDATECD	必ず用意する。
OLDSOC	必ず用意する。
NEWSOC	必ず用意する。
SYSPRINT	印刷情報を見たい場合に必要
STEPLIB	必ず用意する。

```
//UPDATE EXEC PGM=UPDATE
//STEPLIB DD DSN=J████████.R5TOOL.LOAD,DISP=SHR
//UPDATECD DD DSN=J████████.@UPDTC.DATA,DISP=SHR,LABEL=(///,IN)
//OLDSOC DD DSN=J████████.R5FCM00.FORT77,DISP=SHR,LABEL=(///,IN)
//NEWSOC DD DSN=J████████.R5FCM02W.FORT77,DISP=SHR
//SYSPRINT DD SYSOUT=*
// DCB=(LRECL=137,BLKSIZE=15344,RECFM=FBA,DSORG=PS)
```

Fig. D.7 An example of JCL for UPDATE.

D.2.2 実行時オプション

TABLE.D.3 に実行時オプションを示す。これらは、EXEC 文の PARM パラメータに指定できる。複数個の実行時オプションを指定する場合は、各々のオプションをコンマ又は空白で区切って指定する。相反する複数のオプションが指定された場合は最後のオプションが有効となる。指定されなかったオプションの値は、標準値が取られる。

TABLE.D.3 Options of JCL for UPDATE.

オプションの形式		
LINECOUNT ({60 n})		印刷情報の 1 ページあたりの行数の指定 (省略形)
LC ({60 n})		

注) 下線はオプション省略時に取られる標準値である。

次に実行時オプションの詳細を説明する。

LINECOUNT ({60 | n}) 又は LC ({60 | n})

印刷情報を出力するとき、1 ページ当りの行数を指定するものである。この行数には、UPDATE が出力する見出し行も含まれる。特に、n=0 のときは、印刷情報は全てベタ打ちとなる。すなわち、見出し行は 1 回だけ出力され、その後は、編集のための改ページは行わない。省略時は、60 が指定されたものとみなす。

D.2.3 実行時のデータセット

UPDATE は TABLE.D.4 に示されている入出力装置上のデータセットに対してアクセス

する。データセットを DD 文で定義する場合、基本的には定められた DD 名を使用する。DD 文では、DD 名、データセット名、装置情報、ボリューム情報、データセットの取り扱い及び DCB 情報等を記述する。データセットがカタログされている場合、DD 文のパラメータとして、データセット名 (DSNAME パラメータ) とデータセットの取り扱い (DISP パラメータ) だけを指定すればよい。

データセットの DCB 属性が与えられていない場合、UPDATE はある規約に従って DCB 属性を割り当てる。

TABLE.D.4 Attributes of dataset in UPDATE.

DD 名	入出力装置	D C B 属性			
		DSORG	RECFM	LRECL	BLKSIZE
UPDATECD	直接アクセス装置 磁気テープ装置 カード読み取り装置	PS 又は PO	F,FB	80	80の整数倍
OLDSOC	直接アクセス装置 磁気テープ装置 カード読み取り装置	PS	F,FB	80	80の整数倍
NEWSOC	直接アクセス装置 磁気テープ装置 カード読み取り装置	PS 又は PO	F,FB	80	80の整数倍
SYSPRINT	直接アクセス装置 磁気テープ装置 カード読み取り装置	PS	FBA	137 / 81	レコード長の整数倍
			VBA	137 / 81	レコード長 + 4 バイト
STEPLIB	直接アクセス装置	PO	U	0	任意

備考 1) UPDATECD, OLDSOC は、入力ストリームから入力できる。SYSPRINT は、出力ストリームの SYSOUT クラスをラインプリンタとして使用できる。

備考 2) UPDATECD, OLDSOC と NEWSOC の DCB 属性は、DSORG は一致しなくてはならない。

備考 3) NEWSOC の DCB 属性が与えられていない場合、OLDSOC の DCB 属性が割り当てられる。

備考 4) SYSPRINT は、端末装置に割り付けてもよい。

備考 5) PO データセットは直接アクセス装置にのみ割り付けられる。

D.2.3.1 UPDATE データセット

FACOM 版ソースを修正するための制御データが格納されているデータセットである。このデータセットは基本的には区分データセット (FORTCOMP の出力データセット) であるが、順データセット (区分データセットのメンバ指定も含む) も使用可能である。

D.2.3.2 OLDSOC データセット

CDC シーケンス番号付き旧 FACOM 版ソースが格納されているデータセットである。このデータセットの編成は、UPDATECD データセットと同じでなければならない。

D.2.3.3 NEWSOC データセット

UPDATE が出力するアップデート後 FACOM 版ソースを格納するデータセットである。このデータセットのソースを手作業で修正することで新 FACOM 版ソースが完成する。このデータセットの編成は、OLDSOC データセットと同じでなければならない。DCB 属性が与えられていない場合、OLDSOC データセットの DCB 属性が割り当てられる。

D.2.3.4 SYSPRINT データセット

実行の際の各種の印刷情報出力するデータセットである。詳細は D.4 印刷情報を参照されたい。SYSPRINT データセットは、通常システムの出力装置を指定して (SYSOUT パラメータを用いる)、そのユニットレコード装置のクラスをラインプリンタ装置とする。

D.2.3.5 STEPLIB データセット

UPDATE が登録されているロードモジュールデータセットである。

D.3 アップデート後 FACOM 版ソースの形式

NEWSOC データセットの編成は、入力データセットの編成に依存する。すなわち、OLDSOC, UPDATECD データセットが区分データセットならば、NEWSOC データセットも区分データセットとなる。その際、メンバ名は、OLDSOC データセットと同じメンバ名となる。

UPDATECD データセットに依存しないメンバは、NEWSOC データセットには、出力されない。OLDSOC データセットに存在せず、UPDATECD データセットに存在するメンバは、新たに NEWSOC データセットに創成される。

D.4 印刷情報

UPDATE が出力する印刷情報には、データセット・チェックリスト、アップデートカード作業チェックリスト、及びエラーメッセージの3種類がある。エラーメッセージの説明は、ここでは省略する。

D.4.1 データセット・チェックリスト

UPDATECD, OLDSOC, NEWSOC の3種類のデータセットのデータセット名、DCB 属性などを出力するものである。Fig.D.8 にその例を示す。

D.4.2 アップデートカード作業チェックリスト

アップデートカード作業の結果を出力するもので、削除したソースの内容及び挿入されたソースの内容を出力する。さらに、旧 FACOM 版ソースのカード枚数、削除された枚数及びアップデート後 FACOM 版ソースの総枚数が表示される。Fig.D.9 にチェックリストの例を示す。

また、新たに創成されるメンバについては、内容は出力されず、総カード枚数のみ出力される。Fig.D.10 にその例を示す。

PAGE 0001

TIME 19:14:08

DATE 1986.05.08

=== SOURCE FILE UPDATE TOOL ===

L I S T I N G O F I N F O R M A T I O N

```

UPDATE CARD DECK      DD NAME : UPDATECD
FILE NAME : J9303.UPDATECD.DATA
LRECL : 80
BLKSIZE : 3120
RECFM : FB
DSORG : PO

OLD SOURCE FILE      DD NAME : OLDSOC
FILE NAME : J9303.R5FCM00W.FORT77
LRECL : 80
BLKSIZE : 3120
RECFM : FB
DSORG : PO

NEW SOURCE FILE      DD NAME : NEWSOC
FILE NAME : J9303.R5FCM02W.FORT77
LRECL : 80
BLKSIZE : 3120
RECFM : FB
DSORG : PO
    
```

Fig. D.8 An example of check list for dataset in UPDATE.

=== SOURCE FILE UPDATE TOOL ===

```

*****
** CONDEN **
*****

*DELETE 40 -
((CDC HTURB=0.065*THCONF(IV)*SQRT(RHF)/VISCF(IV)*
((CDC 1 SQRT(0.0395*VISCF(IV)*CSUBPF(IV)*RHG*VELG(IV)*VELG(IV)/
((CDC 2 (THCONF(IV)*REYG)**0.25)
(( HTURB=0.065DO*THCONF(IV)*SQRT(RHF)/VISCF(IV)*
(( 1 SQRT(0.0395DO*VISCF(IV)*CSUBPF(IV)*RHG*VELG(IV)*VELG(IV)/
(( 2 (THCONF(IV)*REYG)**0.25DO)
*INSERT 42
(( HTURB=0.065*THCONF(IV)*SQRT(RHF)/VISCF(IV)*
(( 1 SQRT(0.0395*VISCF(IV)*CSUBPF(IV)*RHG*VELG(IV)*VELG(IV)/
(( 2 (THCONF(IV)*REYG**0.25))
*INSERT 57
(( GR=9.8*BETAFF(IV)*ABS(TM-TEMPF(IV))*(RHO(IV)/VISCF(IV))**2
(( *HTHDMO(LS)**3
(( HTCF=AMAX1(HTCF,0.021*(THCONF(IV)/HTHDMO(LS))*)
(( * (VISCF(IV)*CSUBPF(IV)/THCONF(IV)*GR)**0.4*VDIDF(IV))
*DELETE 70 -
((CDC GAMW = QFGO*HTSRFO(LS)/CV(IV)*(HFG+0.375*CSUBPG(IV)*
((CDC 1 AMAX1(0.0, TEMPG(IV)-SATT(IV)))
(( GAMW = QFGO*HTSRFO(LS)/CV(IV)*(HFG+0.375DO*CSUBPG(IV)*
(( 1 AMAX1(0.0DO, TEMPG(IV)-SATT(IV)))
*INSERT 71
(( GAMW=QFGO*HTSRFO(LS)*RECIPV(IV)/(HFG+0.375*CSUBPG(IV)*
(( * AMAX1(0.0, TEMPG(IV)-SATT(IV)))

***** N U M B E R O F C A R D *****
DD NAME = OLDSOC TOTAL = 118
DELETED = 10
INSERTED = 9
DD NAME = NEWSOC TOTAL = 117

```

Fig. D.9 An example of check list of statement deleted and inserted by UPDATE when old source program is different from new source program.

付録E FREDUCT

E.1 機能

E.1.1 機能の概要

FREDUCT プログラムは、RELAP 5 の FACOM 版へのコンバージョンの際に作られるビット操作ファンクションの個数を減らすためのプログラムである。

CDC 計算機は、その FORTRAN コンパイラにビット操作ファンクション、論理演算子を持っており、領域節約のためのパック語の取り扱いの際にそれらを用いている。富士通計算機では、1 語中にビット数の違いなどから完全にそれらに対応するファンクションはなく、またコンバージョンの手間などから、アセンブラによるビット操作ファンクションを作成し、すべてをそれらに置き換えている。ファンクションの変更作業は、ツールを用いて機械的に行われるため、無駄なファンクションの組合せが生じる場合がある。Fig.E.1にそのような例を示す。

ビット操作ファンクションは、外部関数であるため、FORTRAN コンパイラの最適化の対象にはならない。そこで、できるだけ人間の手で最適化をしてやる必要がある。Fig.E.2に最適化を行った例を示す。このような最適化をできるだけ機械的に行おうというのがこのプログラムを作成した理由である。実際、RELAP 5 でこのプログラムを利用し、ファンクションの数を削減したところ、処理性能が1.4倍以上向上した。

このプログラムは、RELAP 5 のベクトル化の際に作成されたものであり、エラーコレクションには直接関係ないが、今後もこのような問題は必ず現れてくるものであるから、ここにその使用法を記す。

ファンクションの変更方法は、4種類あり、それぞれ変更1、変更2、変更3、変更4と呼ぶ。次項から各変更について説明する。

```
KP = (.NOT.MASK(48) .AND. SHIFT(IJ1(I),30)) + IXSOP - 1
```



```
KP = %ID( %SUB( %ADD( %AND( %NOT( %MASK(48)) ,  
%SFT( IJ1(I),30)) , %DI(IXSOP)),%B0001))
```

注) パック語による 60 ビット整数演算のため、このような複雑なファンクションの組合せとなる。

Fig. E.1 An example of bit operation functions made by automatic conversion tool which waste time.

```

KP = ¥ID( ¥SUB( ¥ADD( ¥AND( ¥NOT( ¥MASK(48)) ,
      ¥SFT( IJ1(I),30)) , ¥DI(IXSOP)),¥B0001))

```



```

KP = ¥IDAND( ¥MKN48 , ¥SFT( IJ1(I),30)) + IXSOP - 1

```

*) ¥ IDAND ... ¥IDと¥ANDを合わせた新しいファンクションである。

**) ¥ MKNxx ... マスクデータについては、DATA文であらかじめ定義したものを使用する。

Fig. E.2 An example of optimization of bit operation functions.

E.1.2 変更1の機能と目的

変更1は、次に示す形式のものである。

Func1	(x, y)	⇒	(x) symbol (y)
Func1	(Func2 (x, y))	⇒	Func1 (x) symbol Func2 (y)

この機能は、あるファンクションを別の命令（例えば加算などの演算）に置き換えるものである。

CDC 計算機は、整数データを1語60ビットで扱っており、パック語も整数演算を行っている。FACOM 版プログラムは、パック語について倍精度実数型として定義しているため、整数演算を行うことができない。そのため、60ビット整数演算のためのファンクション¥ADD, ¥SUB などを用意しているわけであるが、実際の実行文の中には結果として整数値を求めているものもある。FACOM 版では、そのような場合次のように表現される。

$$I = ¥ID (¥ADD (X, Y))$$

このX, Yが実際には、32ビット以下しか使われていないことが、あらかじめわかっているような場合には

$$I = ¥ID (X) + ¥ID (Y)$$

のようにファンクションによる加算を通常の演算に変更することが可能である。この例ではファンクションの総数に変化はないが、複雑な演算式において他の変更と組み合わせて用いることによって、ファンクションの数をかなり減らすことができる。

E.1.3 変更2の機能と目的

変更2は、次に示す形式のものである。

Func1 (x)	⇒	Func3 (x)
Func1 (Func2 (x))	⇒	Func3 (x)
Func1 (x)	⇒	x
Func1 (Func2 (x))	⇒	x

この機能は、あるファンクション（1つ又は2つ）を別のファンクションに置き換えたり、ファンクションを取り去ったりするものである。

演算が複雑に入り組んでいる場合、データの型を単精度整数型、倍精度整数型（パック語）間で何回か行う場合がある。このとき、機械的なコンバージョンの結果、

... ¥ID (¥DI (X)) ...

というファンクションの組合せができてしまう場合がある。この場合2つのファンクションは削除することができ、

... X ...

とすることができる。また、

... ¥ID (¥AND (X, Y)) ...

のようにある一定の組合せでファンクションが用いられているような場合、この2つのファンクションをあわせた新しいファンクションを作成し、それに変更すればファンクションの呼出し回数は $\frac{1}{2}$ になる。

... ¥IDAND (X, Y) ...

この新しいファンクションは、前の2つのファンクションの内容を調べた上で同じ動作をするものを作成する。

E.1.4 変更3の機能と目的

変更3は、次に示す形式のものである。

Func1 (n)	⇒	symbol n
Func1 (Func2 (n))	⇒	symbol n

この機能は、マスク設定のためのファンクションを変数名に変更するものである。ファンクションの引数は、数字でなければならない。

RELAP 5では、パック語からのデータの取り出し、パック語へのデータの組み込みにマスクデータを用いている。各実行文に用いられているマスクデータはそれぞれ一定の値であり、データ文で定義された変数を用いても全く問題のないものである。

マスクデータは、ファンクションを用いて次のように定義される。

¥MASK (48), ¥NOT (¥MASK (12))

これらを変数に変更する際、変数名が一意になるようにするためには中の数字を変数名に利用して、

¥MKP48, ¥MKN12

のようにすればよい。あとは、BLOCKDATA を用いて各変数にマスクデータを定義する。

変更3においては、ファンクションの引数は数字であることを前提としており、たとえファンクション名が該当するものであったとしても、引数が整数以外のものであった場合は、変更の対象とはしない。このチェックによって、もし引数が変数であったときに変な変数名を作成してしまう危険が除かれる。

E.1.5 変更4の機能と目的

変更4は、次に示す形式のものである。

Func1 (x)	⇒ symbol x
Func1 (Func2 (x))	⇒ symbol x

この機能は、変更3と同じくファンクションを変数に置き換えるものである。変更4において引数の制限はない。

変更4は、ベクトル化において実際に使われた機能でない。機能は変更3と全く同じであり、ただ引数が変数だけでなく、計算式であってもファンクションを取り去り文字を付けるので、使用の際には注意を要する。

E.1.6 実行時における変更の順序

各変更の実行順序は、データの入力順序に関係なく、変更1、変更2、変更3、変更4の順序に行われる。指定されなかった変更については、行われない。各変更の中では、入力データの順に検索・変更が行われる。Fig.E.3 に示すデータの場合、各ステートメントごとに(2) → (7) → (8) → (10) → (11) → (4) → (5)の順に検索・変更が行われる。

1ステートメント中に変更対象ファンクションが複数存在するとき、その全てを変更する。Fig.E.3 の例において(2)、(8)は同じ変更を指示しているが、これは(7)に関する変更の実行後新たにこの形式のファンクションが表面に現われてくることがあるためである。

```

.....'.....1.....'.....2.....'.....3.....'.....4
(1)  -MODIFY1
(2)  ¥ID(¥ADD,+
(3)  -MODIFY3
(4)  ¥NOT(¥MASK,¥MKN
(5)  ¥MASK,¥MKP
(6)  -MODIFY1
(7)  ¥ID(¥SUB,-
(8)  ¥ID(¥ADD,+
(9)  -MODIFY2
(10) ¥ID(¥AND,¥IDAND
(11) ¥ID(¥DI
(12) -END

```

Fig. E.3 An example of control data to show the order of process by FREDUCT.

E.2 実行方法

FREDUCT プログラムの主な入力、FACOM 版 FORTRAN ソースと変更を指示する制御データである。その他の入力として実行時オプションがある。FORTRAN ソースは INSOC データセットから、制御データは SYSIN データセットから入力する。また、実行時オプションは EXEC 文の PARM パラメータから入力される。

FREDUCT プログラムの主な出力は、ファンクション変更後の FACOM 版ソースとチェックリストである。ソースは OUTSOC データセットにチェックリストは SYSPRINT データセットにそれぞれ出力される。

Fig.E.4 に入出力の概略を示す。

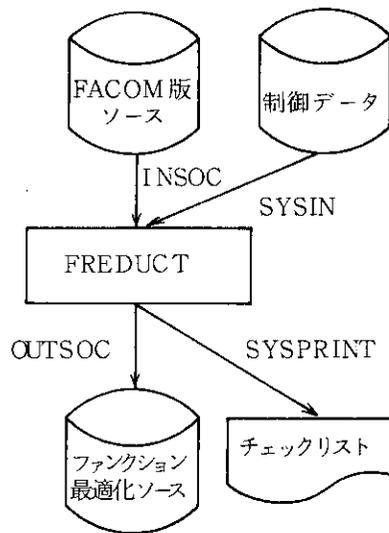


Fig. E.4 INPUT/OUTPUT configuration for FREDUCT.

E.2.1 FREDUCT の実行

FREDUCT プログラムを実行するためには、1つの EXEC 文といくつかの DD 文を用意しなければならない。EXEC 文には通常、TABLE.E.1 に示すパラメータを指定すればよい。PARM パラメータの指定内容の詳細は、E.2.2 実行時オプションを参照されたい。実行時に必要な DD 文は、TABLE.E.2 に示されるものである。この中には条件により用意しなくてもよいものがある。

実行時のジョブ制御文の例を Fig.E.5 に示す。

TABLE.E.1 Main parameters of EXEC statements of JCL for FREDUCT.

PGM	プログラム名FREDUCT を指定する。このパラメータは必須である。
PARM	実行時のオプションを指定する。

TABLE.E.2 DD statements of JCL for FREDUCT.

DD名	必要の有無
INSOC	必ず用意する。
SYSIN	制御データを入力する際に必要
OUTSOC	必ず用意する。
SYSPRINT	印刷情報を見たい場合に必要
STEPLIB	必ず用意する。

```

//          EXEC PGM=FREDUCT,
//          PARM='ELM(*),CHKLIST<INOUT>,REPLACE'
//STEPLIB  DD DSN=J████████.RSTOOL.LOAD,DISP=SHR
//INSOC    DD DSN=J████████.R5FCM02.FORT77,DISP=SHR,LABEL=(,,,IN)
//OUTSOC   DD DSN=J████████.R5FCM02W.FORT77,DISP=SHR
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=137,BLKSIZE=6850)
//SYSIN    DD *
-MODIFY1
  %ID(%ADD,+
  %ID(%SUB,-
  %ID(%ADD,+
-MODIFY2
  %ID(%AND,%IDAND
  %ID(%DI
-MODIFY3
  %NOT(%MASK,%MKN
  %MASK,%MKP
-END
/*

```

Fig. E.5 An example of JCL for FREDUCT.

E.2.2 実行時オプション

TABLE.E.3 に実行時オプションを示す。これらは、EXEC 文の PARM パラメータ又は SYSIN データセットの先頭（制御データの前）に指定できる。複数個の実行時オプションを指定する場合は、各々のオプションをコンマ又は空白で区切って指定する。相反する複数のオプションが指定された場合は最後のオプションが有効となる。指定されなかったオプションの値は、標準値が取られる。

次に実行時オプションの詳細を説明する。

CHKLIST ({IN | OUT | INOUT | NO})

印刷情報の出力レベルを指定する。

CHKLIST (IN) は、変更対象となったステートメントについて変更前のステートメントについて出力することを意味する。

CHKLIST (OUT) は、変更対象となったステートメントについて変更後のステートメントについて出力することを意味する。

CHKLIST (INOUT) は、変更対象となったステートメントについて変更前・変更後の両方のステートメントについて出力することを意味する。

TABLE.E.3 Options of JCL for FREDUCT.

オプションの形式		
CHKLIST ({IN <u>OUT</u> INOUT NO})		印刷情報の出力レベル
ELM (mem1 [-mem2] [,mem1 [-mem2]] ...)		区分データセット内の対象とするメンバ名の指定
<u>ELM (*)</u>		区分データセット内のすべてのメンバを対象とする。
LINECOUNT ({60 n})		印刷情報の1ページあたりの行数の指定
LC ({60 n})		(省略形)
RENUMBER ((({100 n1}) [, {100 n2}]))	<u>NORENUMBER</u>	シーケンス番号付けの指定
RENUM ((({100 n1}) [, {100 n2}]))	<u>NORENUM</u>	(省略形)
REN ((({100 n1}) [, {100 n2}]))	<u>NOREN</u>	(省略形)
REPLACE	<u>NOREPLACE</u>	既に存在するメンバを置き換えるか否かの指定
REP	<u>NOREP</u>	(省略形)

注) 下線はオプション省略時に取られる標準値である。

CHKLIST (NO) は、印刷情報は出力しないことを意味する。

このオプションを省略した場合は、CHKLIST (OUT) が指定されたものとみなす。また、このオプションとは関係なく SYSIN データの内容が最初に出力される。印刷情報の詳細は、F.5 印刷情報を参照されたい。

ELM (mem1 [-mem2] [,mem3 [-mem4]] ...)

ELM (*)

ソース・プログラムの入力を区分データセットから行う際のメンバ名を指示するものである。mem1, mem2, ... はソース・プログラムが存在するメンバの名前であり、最大100組指定できる。メンバの指定方法には、直接メンバ名を指定する方法と、範囲で指定する方法がある。直接指定する方法は対象とするメンバ名をコンマで区切って並べる。範囲で指定する方法は、'-'の前後にメンバ名を書く。始まりと終りを示す名前は、必ずしも実在の名前でなくてもよい。範囲指定の場合、EBCDIC コードによる大小判断で行う。このとき始まりを示すメンバ名は後ろに low-value を補い、終りを示すメンバ名には、high-value を補う。また、ELM(*) を指定した場合は、全メンバが対象となる。

(例)

ELM (A-C, Z) ... 先頭1文字がAまたはBまたはCではじまるメンバと、名前がZで始まるメンバが対象となる。

順データセットの場合この指定は、意味を持たない。区分データセットのとき、このオプションを省略した場合は、ELM(*) が指定されたものとみなされる。メンバの処理は、EBCDIC 順に行われる。

LINECOUNT ({60 | n}) または、LC ({60 | n})

印刷情報を出力するとき、1ページ当りの行数を指定するものである。この行数には、UPDATE

が出力する見出し行も含まれる。特に、 $n = 0$ のとき、印刷情報は全てベタ打ちとなる。すなわち、見出し行は1回だけ出力され、その後は、編集のための改ページは行わない。省略時は、60が指定されたものとみなす。

RENUMBER [([{100 | n1}] [, {100 | n2}])] または

REN [([{100 | n1}] [, {100 | n2}])]

NORENUMBER または NORENUM または NORE

ファンクション変更後のソースを出力する際にシーケンス番号付けを行うかどうかを指定する。シーケンス番号は、出力メンバが代わるたびに初期値から付け直す。

RENUMBER (n1, n2) は初期値 n1, 増分値 n2 とすることを意味する。

RENUMBER (n1) は初期値 n1, 増分値100とすることを意味する。

RENUMBER (, n2) は初期値100, 増分値 n2 とすることを意味する。

RENUMBER は、初期値100, 増分値100とすることを意味する。

NORENUMBER は、シーケンス番号付けを行わないことを意味する。

標準値は NORENUMBER である。

REPLACE または REP

NOREPLACE または NOREP

出力データセットに既に同じ名前のメンバが存在している場合に、それと置き換えるかどうかを指定する。REPLACE は、置き換えることを意味する。古いメンバ名は消去される。NOREPLACE は、置き換えないことを意味する。新しいメンバの処理は行われない。

標準値は、NOREPLACE である。

E.2.3 実行時のデータセット

FREDUCT は TABLE.E. 4 に示されている入出力装置上のデータセットに対してアクセスする。データセットを DD 文で定義する場合、基本的には定められた DD 名を使用する。DD 文では、DD 名、データセット名、装置情報、ボリューム情報、データセットの取り扱い及び DCB 情報等を記述する。データセットがカタログされている場合、DD 文のパラメータとして、データセット名 (DSNAME パラメータ) とデータセットの取り扱い (DISP パラメータ) だけを指定すればよい。

データセットの DCB 属性が与えられていない場合、FREDUCT はある規約に従って DCB 属性を割り当てる。

E.2.3.1 INSOC データセット

CDC 版からコンバージョンしただけの FACOM 版ソースが格納されているデータセットである。このデータセットは順データセット又は区分データセットでなければならない。

DD 文の指定より連結したデータセットを定義してもよい。この場合は、各データセットの DCB 属性は同じでなければならない。また、1つのデータセットの処理が終わってから次のデータセットの処理に移る。

INSOC データセットは入力ストリームの中でデータを定義しても良い。この場合は、データの最後に区切りとして区切り文を置く。

TABLE.E.4 Attributes of dataset in FREDUCT.

DD名	入出力装置	D C B 属性			
		D S O R G	RECFM	LRECL	BLKSIZE
INSOC	直接アクセス装置 磁気テープ装置 カード読み取り装置	PS 又は PO	F,FB	80	80の整数倍
SYSIN	直接アクセス装置 磁気テープ装置 カード読み取り装置	PS	F,FB	80	80の整数倍
OUTSOC	直接アクセス装置 磁気テープ装置 カード読み取り装置	PS 又は PO	F,FB	80	80の整数倍
SYSPRINT	直接アクセス装置 磁気テープ装置 カード読み取り装置	PS	FBA	137 / 81	レコード長の整数倍
			VBA	137 / 81	レコード長+4バイト
STEPLIB	直接アクセス装置	PO	U	0	任意

備考1) INSOC, SYSIN は、入力ストリームから入力できる。OUTSOCは、出力ストリームのSYSOUTクラスをカード穿孔装置として使用できる。SYSPRINTは、出力ストリームのSYSOUTクラスをラインプリンタとして使用できる。

備考2) OUTSOCのDCB属性が与えられていない場合、INSOCのDCB属性が割当てられる。

備考3) SYSIN, SYSPRINTは、端末装置に割り付けてもよい。

備考4) POデータセットは、直接アクセスデータ装置のみ割り付けられる。

E.2.3.2 SYSIN データセット

ファンクション変更のための制御データが格納されているデータセットである。制御データについての詳細は、E.3 制御データを参照されたい。

データセットの先頭、制御データより前に E.2.2 で示した実行時オプションを置くことができる。PARM パラメータのオプションより、これらのオプションのほうが優先される。

SYSIN データセットは入力ストリームの中でデータを定義しても良い。この場合は、データの最後に区切りとして区切り文を置く。

E.2.3.3 OUTSOC データセット

FREDUCT が出力するファンクション変更後のソースを格納するデータセットである。このソースを基本としてベクトル化が行われる。

このデータセットの DCB 属性は、その編成がINSOC データセットと同じでなければならない。DCB 属性が与えられていない場合は、INSOC の DCB 属性が割り当てられる。

E.2.3.4 SYSPRINT データセット

実行の際の各種の印刷情報を出力するデータセットである。詳細は E.4 印刷情報を参照されたい。SYSPRINT データセットは、通常システムの出力装置を指定して (SYSOUT パラメータを用いる)、そのユニットレコード装置のクラスをラインプリンタ装置とする。

E.2.3.5 STEPLIB データセット

FREDUCT が登録されているロードモジュールデータセットである。

E.3 制御データ

制御データは、ファンクションの変更方法、変更するファンクション名などを指定するもので、一制御文と変更データから成り立っている。制御データの例を Fig.E.6 に示す。1枚の一制御文に対して複数の変更データが付く。変更データの内容・意味は、一制御文ごとに変わる。以下に各制御データの書き方、内容について説明する。

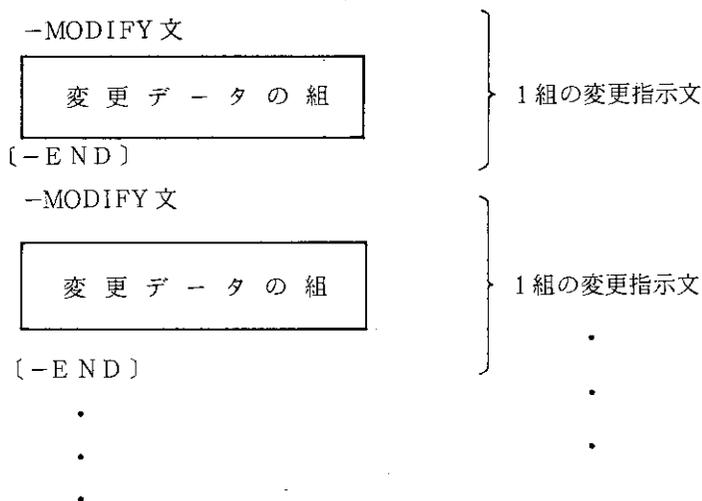
```

-MODIFY1
  ¥ID(¥ADD,+
  ¥ID(¥SUB,-
  ¥ID(¥ADD,+
-MODIFY2
  ¥ID(¥AND,¥IDAND
  ¥ID(¥DI
-MODIFY3
  ¥NOT(¥MASK,¥MKN
  ¥MASK,¥MKP
-END
    
```

Fig. E.6 An example of control data for FREDUCT.

E.3.1 制御データの構成

制御データの構成は次の通りである。1枚の -MODIFY 文と複数の変更データ及び1枚の -END 文によって1組のファンクションの変更指示が構成され、これが数組集まって制御データが作られている。Fig.E.7 に制御データの構成を示す。ここで、-END 文は省略可能である。



*) -END 文は省略可能である。

Fig. E.7 Structure of control data.

E.3.2 -MODIFY 1 制御文

これは、変更1を行うことを指示するもので、書き方は次の通りである。

1 2 3 4 5 6 7 8	9	72	73	80
*MODIFY 1				

9～72カラムは空白でなければならない。この制御文と -END 文または、次の制御文の間にある変更データはすべて変更1のためのデータである。

E.3.3 -MODIFY 2 制御文

これは、変更2を行うことを指示するもので、書き方は次の通りである。

1 2 3 5 5 6 7 8	9	72	73	80
*MODIFY 2				

9～72カラムは空白でなければならない。この制御文と -END 文または、次の制御文の間にある変更データはすべて変更2のためのデータである。

E.3.4 -MODIFY 3 制御文

これは、変更3を行うことを指示するもので、書き方は次の通りである。

1 2 3 4 5 6 7 8	9	72	73	80
*MODIFY 3				

9～72カラムは空白でなければならない。この制御文と -END 文または、次の制御文の間にある変更データはすべて変更3のためのデータである。

E.3.5 -MODIFY 4 制御文

これは、変更4を行うことを指示するもので、書き方は次の通りである。

1 2 3 4 5 6 7 8	9	72	73	80
*MODIFY 4				

9～72カラムは空白でなければならない。この制御文と -END 文または、次の制御文の間にある変更データはすべて変更4のためのデータである。

E.3.6 -END 制御文

1 2 3 4	5	72	73	80
-END				

-END 文は、各制御文に続く変更データの終りを意味する。書き方は上に示す通りである。ここで、5～72カラムは空白でなければならない。また、この文は省略してもよい。

E.3.7 変更データ

変更データの形式は、以下のいずれかである。

```
alpha
alpha (beta
alpha, gamma
alpha (beta, gamma
```

ここで、alpha, beta は変更対象とするファンクション名であり、gamma は、変更後のファンクション名または記号名である。gamma の意味は、変更の種類によって変わる。

変更データは、1～72カラムのどの位置に書いてもよい。ただし、1枚の中に複数の変更データをいれたり、2枚以上に継続して書くことはできない。複数のファンクションの変更を行う場合は、1つの変更について1枚使い、必要枚数そろえる必要がある。

TABLE.E.5 に変更種別の変更データの書き方、ソースの変更方法及び gamma の意味するもの一覧表を示す。ただし、この条件さえ満たせば中身は、何であろうとすべて変更してしまうので、FREDUCT 実行の際には十分な調査を行う必要がある。

Table E.5 Typical patterns of modification by FREDUCT.

種類	データの形式	変更前の文	変更後の文	gamma の説明
変更1	alpha, gamma alpha(beta,gamma	alpha (X,Y) alpha (beta(X,Y))	(X) gamma (Y) alpha(X) gamma alpha(Y)	演算記号, 論理記号 他 (+, - など)
変更2	alpha alpha (beta alpha, gamma alpha(beta,gamma	alpha (X) alpha (beta(X)) alpha (X) alphh (beta(X))	X X gamma (X) gamma (X)	ファンクション名
変更3	alpha, gamma alpha(beta,gamma	alpha (n) alpha (beta (n))	gamma n gamma n	変数名の一部
変更4	alpha, gamma alpha(beta,gamma	alpha (X) alpha (beta(X))	gamma X gamma X	変数名の一部

注) X, Y は、変数名又は演算式である。
n は、数字である。

E.4 出力ソースの形式

出力ソースのデータセットの編成は、入力データセットと同じでなければならない。すなわち、入力データセットが順データセットならば、出力データセットも順データセットであり、入力データセットが区分データセットならば、出力データセットも区分データセットでなければならない。区分データセットの場合、出力メンバ名は入力の際のメンバ名と同じとなる。Fig.E. 8にFREDUCT実行前のソースの例、Fig.E. 9にFREDUCT実行後ソースの例を示す。

```

C
C MASS, ENERGY AND QUALITY CONTRIBUTIONS TO CELL K FROM JUNCTION I.
J = IXIPX
DO 11 I = IJ,IJE,IJSKP
CDC KXV(J) = .NOT.MASK(43) .AND. SHIFT(IJ1(I),18)
KXV(2,J)=%ID(%AND(%NOT(%MASK(43)), %SFT(IJ1(I),18)))
SCV2(J) = QUALAJ(I)*CONMG(J)
CDC SCV3(J) = (UGJ(I) + PO(KXV(J))/RHOGJ(I))*CONMG(J)
SCV3(J) = (UGJ(I) + PO(KXV(2,J))/RHOGJ(I))*CONMG(J)
CDC SCV4(J) = (UFJ(I) + PO(KXV(J))/RHOFJ(I))*CONMF(J)
SCV4(J) = (UFJ(I) + PO(KXV(2,J))/RHOFJ(I))*CONMF(J)
J = J + 26
11 CONTINUE
J = IXIPX
DO 12 I = IJ,IJE,IJSKP
CDC K = .NOT.MASK(43) .AND. SHIFT(JC(I),24)
K = %ID(%AND(%NOT(%MASK(43)), %SFT(JC(I),24)))
CDC KP = (.NOT.MASK(48) .AND. SHIFT(IJ1(I),30)) + IXSOP - 1
KP = %ID(%SUB(%ADD(%AND(%NOT(%MASK(48)), %SFT(IJ1(I),30))
*,%DI(IXSOP)),%B0001))
SOURCA(K) = SOURCA(K) - SCV2(J)
SOURCG(K) = SOURCG(K) - SCV3(J)
SOURCF(K) = SOURCF(K) - SCV4(J)
SOURCM(K) = SOURCM(K) - COND(J)
SOURCP(KP) = SOURCP(KP) - CONM(J)
J = J + 26
12 CONTINUE
C
C MASS, ENERGY AND QUALITY CONTRIBUTIONS TO CELL L FROM JUNCTION I.
J = IXIPX
DO 13 I = IJ,IJE,IJSKP
CDC LXV(J) = .NOT.MASK(43) .AND. SHIFT(IJ2(I),18)
LXV(2,J)=%ID(%AND(%NOT(%MASK(43)), %SFT(IJ2(I),18)))
CDC SCV5(J) = (UGJ(I) + PO(LXV(J))/RHOGJ(I))*CONMG(J)
SCV5(J) = (UGJ(I) + PO(LXV(2,J))/RHOGJ(I))*CONMG(J)
CDC SCV6(J) = (UFJ(I) + PO(LXV(J))/RHOFJ(I))*CONMF(J)
SCV6(J) = (UFJ(I) + PO(LXV(2,J))/RHOFJ(I))*CONMF(J)
J = J + 26
13 CONTINUE
J = IXIPX
DO 14 I = IJ,IJE,IJSKP
CDC L = .NOT.MASK(43) .AND. SHIFT(JC(I+1),24)
L = %ID(%AND(%NOT(%MASK(43)), %SFT(JC(I+1),24)))
CDC LP = (.NOT.MASK(48) .AND. SHIFT(IJ2(I),30)) + IXSOP - 1
LP = %ID(%SUB(%ADD(%AND(%NOT(%MASK(48)), %SFT(IJ2(I),30))
*,%DI(IXSOP)),%B0001))
SOURCA(L) = SOURCA(L) + SCV2(J)
SOURCG(L) = SOURCG(L) + SCV5(J)
SOURCF(L) = SOURCF(L) + SCV6(J)
SOURCM(L) = SOURCM(L) + COND(J)
SOURCP(LP) = SOURCP(LP) + CONM(J)
J = J + 26
14 CONTINUE
C
C CONVECT BORON.
J = IXIPX
DO 15 I=IJ,IJE,IJSKP
CDC KK = .NOT.SHIFT(VELFJ(I),-59).AND.KXV(J) .OR.
* SHIFT(VELFJ(I),-59).AND.LXV(J)
C! KK = %ID(%OR(%AND(%NOT(%SFT(VELFJ(I),-59)), %DI(KXV(1,J))),
C! * %AND(%SFT(VELFJ(I),-59), %DI(LXV(1,J))))
IF(VELFJ(I).LT.0.000) THEN
KK=LXV(2,J)
ELSE
KK=KXV(2,J)
ENDIF
CONBOR = BORONO(KK)*VELFJ(I)*SCV1(J)
CDC BORON(KXV(J)) = BORON(KXV(J)) - CONBOR*RECIPV(KXV(J))
BORON(KXV(2,J))=BORON(KXV(2,J))-CONBOR*RECIPV(KXV(2,J))
CDC BORON(LXV(J)) = BORON(LXV(J)) + CONBOR*RECIPV(LXV(J))
BORON(LXV(2,J))=BORON(LXV(2,J))+CONBOR*RECIPV(LXV(2,J))
J = J + 26
15 CONTINUE

```

Fig. E.8 An example of source program before optimization by FREDUCT.

```

C
C MASS, ENERGY AND QUALITY CONTRIBUTIONS TO CELL K FROM JUNCTION I.
J = IXIPX
DO 11 I = IJ,IJE,IJSKP
CDC KXV(J) = .NOT.MASK(43) .AND. SHIFT(IJ1(I),18)
KXV(2,J)=%IDAND( %MKN43, %SFT(IJ1(I),18))
SCV2(J) = QUALAJ(I)*CONMG(J)
CDC SCV3(J) = (UGJ(I) + PO(KXV(J)))/RHOGJ(I)*CONMG(J)
SCV3(J) = (UGJ(I) + PO(KXV(2,J)))/RHOGJ(I)*CONMG(J)
CDC SCV4(J) = (UFJ(I) + PO(KXV(J)))/RHOFJ(I)*CONMF(J)
SCV4(J) = (UFJ(I) + PO(KXV(2,J)))/RHOFJ(I)*CONMF(J)
J = J + 26
11 CONTINUE
J = IXIPX
DO 12 I = IJ,IJE,IJSKP
CDC K = .NOT.MASK(43) .AND. SHIFT(JC(I),24)
K = %IDAND( %MKN43, %SFT(JC(I),24))
CDC KP = (.NOT.MASK(48) .AND. SHIFT(IJ1(I),30)) + IXSOP - 1
KP = %IDAND( %MKN48, %SFT(IJ1(I),30)) + IXSOP - %ID(%B0001)
SOURCA(K) = SOURCA(K) - SCV2(J)
SOURCG(K) = SOURCG(K) - SCV3(J)
SOURCF(K) = SOURCF(K) - SCV4(J)
SOURCM(K) = SOURCM(K) - COND(J)
SOURCP(KP) = SOURCP(KP) - CONM(J)
J = J + 26
12 CONTINUE
C
C MASS, ENERGY AND QUALITY CONTRIBUTIONS TO CELL L FROM JUNCTION I.
J = IXIPX
DO 13 I = IJ,IJE,IJSKP
CDC LXV(J) = .NOT.MASK(43) .AND. SHIFT(IJ2(I),18)
LXV(2,J)=%IDAND( %MKN43, %SFT(IJ2(I),18))
CDC SCV5(J) = (UGJ(I) + PO(LXV(J)))/RHOGJ(I)*CONMG(J)
SCV5(J) = (UGJ(I) + PO(LXV(2,J)))/RHOGJ(I)*CONMG(J)
CDC SCV6(J) = (UFJ(I) + PO(LXV(J)))/RHOFJ(I)*CONMF(J)
SCV6(J) = (UFJ(I) + PO(LXV(2,J)))/RHOFJ(I)*CONMF(J)
J = J + 26
13 CONTINUE
J = IXIPX
DO 14 I = IJ,IJE,IJSKP
CDC L = .NOT.MASK(43) .AND. SHIFT(JC(I+1),24)
L = %IDAND( %MKN43, %SFT(JC(I+1),24))
CDC LP = (.NOT.MASK(48) .AND. SHIFT(IJ2(I),30)) + IXSOP - 1
LP = %IDAND( %MKN48, %SFT(IJ2(I),30)) + IXSOP - %ID(%B0001)
SOURCA(L) = SOURCA(L) + SCV2(J)
SOURCG(L) = SOURCG(L) + SCV5(J)
SOURCF(L) = SOURCF(L) + SCV6(J)
SOURCM(L) = SOURCM(L) + COND(J)
SOURCP(LP) = SOURCP(LP) + CONM(J)
J = J + 26
14 CONTINUE
C
C CONVECT BORON.
J = IXIPX
DO 15 I=IJ,IJE,IJSKP
CDC KK = .NOT.SHIFT(VELFJ(I),-59).AND.KXV(J) .OR.
CDC * SHIFT(VELFJ(I),-59).AND.LXV(J)
C! KK = %ID( %OR( %AND( %NOT( %SFT(VELFJ(I),-59)), %DI(KXV(1,J))),
C! * %AND( %SFT(VELFJ(I),-59), %DI(LXV(1,J))))
IF(VELFJ(I).LT.0.000) THEN
KK=LXV(2,J)
ELSE
KK=KXV(2,J)
ENDIF
CONBOR = BORONO(KK)=VELFJ(I)*SCV1(J)
CDC BORON(KXV(J)) = BORON(KXV(J)) - CONBOR*RECIPV(KXV(J))
BORON(KXV(2,J))=BORON(KXV(2,J))-CONBOR*RECIPV(KXV(2,J))
CDC BORON(LXV(J)) = BORON(LXV(J)) + CONBOR*RECIPV(LXV(J))
BORON(LXV(2,J))=BORON(LXV(2,J))+CONBOR*RECIPV(LXV(2,J))
J = J + 26
15 CONTINUE

```

Fig. E.9 An example of source program after optimization by FREDUCT.

E.5 印刷情報

FREDUCT が出力する印刷情報には、SYSIN データ・チェックリスト、ファンクション変更チェックリスト、及びエラーメッセージの三種類がある。エラーメッセージの説明は、ここでは省略する。

E.5.1 SYSIN データ・チェックリスト

SYSIN データセットから入力したデータの内容をそのまま出力するものである。Fig.E.10 にその例を示す。このリストは、実行時パラメータの CHKLIST の値に関係なく出力される。

E.5.2 ファンクション変更チェックリスト

制御データに従って変更されたステートメントの内容を出力するものである。出力情報は、実行時パラメータ CHKLIST の値によって変化する。

CHKLIST (IN) の場合、変更の行われたステートメントについて、変更前の内容を出力する。Fig.E.11にその例を示す。

CHKLIST (OUT) の場合、変更の行われたステートメントについて、変更後の内容を出力する。Fig.E.12にその例を示す。

CHKLIST (INOUT) の場合、変更の行われたステートメントについて、変更前・変更後の両方の内容を出力する。Fig.E.13にその例を示す。

CHKLIST (NO) の場合、チェックリストは出力されない。

LINECOUNT 値が 0 より大きい場合、通常の改ページの他にもメンバの先頭にアスタリスクで囲まれたメンバが出力される。

<<< SYSIN DATA CHECK LIST >>>

```

.....1.....".....3.....".....4.....".....5.....".....6.....".....7.....".....8
-MODIFY1
%ID(%ADD,+
%ID(%SUB,-
%ID(%ADD,+
-MODIFY2
%ID(%AND,%IDAND
%ID(%OI
-MODIFY3
%NOT(%MASK,%MKN
%MASK,%MKP
-END
00130000
00140000
00150000
00160000
00170000
00180000
00190000
00200000
00210000
00220000
00230000

```

Fig. E.10 An example of check list of SYSIN dataset.

```

FUNCTION REDUCTION PROGRAM V10L10 DATE 1986.05.15 TIME 18:29:42 PAGE.0008
*****
** EQFINL **
*****
(INPUT) : I KXV(2,J)=#ID( #AND( #NOT( #MASK(43)), #SFT(IJ1(I),18))) 000009401
CODE=0004

(INPUT) : I K = #ID( #AND( #NOT( #MASK(43)), #SFT(JC(I),24))) 000010501
CODE=0004

(INPUT) : I KP = #ID( #SUB( #ADD( #AND( #NOT( #MASK(48)), #SFT(IJ1(I),30)),000010701
I *,#DI(IXSOP)),#B0001)) 000010801
CODE=0004

(INPUT) : I LXV(2,J)=#ID( #AND( #NOT( #MASK(43)), #SFT(IJ2(I),18))) 000012101
CODE=0004

(INPUT) : I L = #ID( #AND( #NOT( #MASK(43)), #SFT(JC(I+1),24))) 000013101
CODE=0004

(INPUT) : I LP = #ID( #SUB( #ADD( #AND( #NOT( #MASK(48)), #SFT(IJ2(I),30)),000013301
I *,#DI(IXSOP)),#B0001)) 000013401
CODE=0004

(INPUT) : I K = #ID( #AND( #NOT( #MASK(48)), #SFT(IJ1(I),30))) 000018701
CODE=0004

(INPUT) : I L = #ID( #AND( #NOT( #MASK(48)), #SFT(IJ2(I),30))) 000018901
CODE=0004

(INPUT) : I 39 VCTRL(I) = #OR(VCTRL(I), #SFT( #MASK(1),12)) 000039801
CODE=0004

```

Fig. E.11 An example of check list of the object statements before optimization by FREDUCT.

```

*****
** EPIFL **
*****
      KXV(2,J)=%IDAND( %MKN43, %SFT(IJ1(I),18))
      (OUTPUT) : I      000009401
      CODE=0004

      K = %IDAND( %MKN43, %SFT(JC(I),24))
      (OUTPUT) : I      000010501
      CODE=0004

      KP = %IDAND( %MKN48, %SFT(IJ1(I),30)) + IXSOP - %ID(%B0001)
      (OUTPUT) : I      000010701
      CODE=0004

      LXV(2,J)=%IDAND( %MKN43, %SFT(IJ2(I),18))
      (OUTPUT) : I      000012101
      CODE=0004

      L = %IDAND( %MKN43, %SFT(JC(I+1),24))
      (OUTPUT) : I      000013101
      CODE=0004

      LP = %IDAND( %MKN48, %SFT(IJ2(I),30)) + IXSOP - %ID(%B0001)
      (OUTPUT) : I      000013301
      CODE=0004

      K = %IDAND( %MKN48, %SFT(IJ1(I),30))
      (OUTPUT) : I      000018701
      CODE=0004

      L = %IDAND( %MKN48, %SFT(IJ2(I),30))
      (OUTPUT) : I      000018901
      CODE=0004

      VCTRL(I) = %OR(VCTRL(I), %SFT( %MKP1,12))
      (OUTPUT) : I  39      000039801
      CODE=0004
    
```

Fig. E.12 An example of check list of the object statements after optimization by FREDUCT.

```

*****
**  EQFINL  **
*****
(INPUT) : I      KXV(2,J)=VID( AND( NOT( MASK(43)), SFT(IJ1(I),18)))      000009401
(OUTPUT) : I      KXV(2,J)=VIDAND( MKN43, SFT(IJ1(I),18))                    000009401
CODE=0004

(INPUT) : I      K = VID( AND( NOT( MASK(43)), SFT(JC(I),24)))              000010501
(OUTPUT) : I      K = VIDAND( MKN43, SFT(JC(I),24))                          000010501
CODE=0004

(INPUT) : I      KP = VID( SUB( ADD( AND( NOT( MASK(48)), SFT(IJ1(I),30))000010701
          I      *,DI(IXSOP)),B0001))
(OUTPUT) : I      KP = VIDAND( MKN48, SFT(IJ1(I),30)) + IXSOP - VID(B0001)    000010701
CODE=0004

(INPUT) : I      LXV(2,J)=VID( AND( NOT( MASK(43)), SFT(IJ2(I),18)))          000012101
(OUTPUT) : I      LXV(2,J)=VIDAND( MKN43, SFT(IJ2(I),18))                    000012101
CODE=0004

(INPUT) : I      L = VID( AND( NOT( MASK(43)), SFT(JC(I+1),24)))            000013101
(OUTPUT) : I      L = VIDAND( MKN43, SFT(JC(I+1),24))                        000013101
CODE=0004

(INPUT) : I      LP = VID( SUB( ADD( AND( NOT( MASK(48)), SFT(IJ2(I),30))000013301
          I      *,DI(IXSOP)),B0001))
(OUTPUT) : I      LP = VIDAND( MKN48, SFT(IJ2(I),30)) + IXSOP - VID(B0001)    000013301
CODE=0004

(INPUT) : I      K = VID( AND( NOT( MASK(48)), SFT(IJ1(I),30)))              000018701
(OUTPUT) : I      K = VIDAND( MKN48, SFT(IJ1(I),30))                          000018701
CODE=0004

(INPUT) : I      L = VID( AND( NOT( MASK(48)), SFT(IJ2(I),30)))              000018901
(OUTPUT) : I      L = VIDAND( MKN48, SFT(IJ2(I),30))                          000018901
CODE=0004

(INPUT) : I      39 VCTRL(I) = FOR(VCTRL(I), SFT( MASK(1),12))                000039801
(OUTPUT) : I      39 VCTRL(I) = FOR(VCTRL(I), SFT( MKNP1,12))                000039801
CODE=0004

```

Fig. E.13 An example of check list of the object statements before and after optimization by FREDUCT.

付録 F STREAM77

F.1 機能

F.1.1 機能の概要

STREAM77は、CDC 又はCRAY 計算機用に開発された FORTRAN プログラムを FACOM 計算機用に変換するための支援パッケージである。STREAM77は、富士通提供ソフトウェアパッケージであり、詳細はマニュアルを参照されたい。ここでは、機能・使用方法の概要及び RELAP 5 / MOD 2 のエラー修正に実際に用いた例を示すにとどめる。

FORTRAN は、国際的な規格のある高級計算機言語である。しかし、各計算機メーカーは、各自の計算機ハードウェアの特徴を活かすために、規格に含まれていない文法を許している。また、ハードウェアの違い、特に 1 語の長さ（ビット数）の違いによる演算精度の差などのため他の計算機システムにおいて開発されたプログラムを導入する際には、何らかの変換作業が必要になる場合が多い。

RELAP 5 も CDC 計算機の特徴を利用しており、60ビット整数演算、ビット演算などが用いられている。これらは、パック語演算に用いられており、FACOM 計算機への変換の大きな障害の 1 つとなっている。

STREAM77は、この変換作業を効率よく行うための支援パッケージであり、以下に示す部品で構成されている。

- (1) C-エンジン（ソフトウェア・ツール）
 - FORTRAN プログラムの静的構造解析
 - FORTRAN プログラムの変換
- (2) P-エンジン（ソフトウェア・ツール）
 - C-エンジンのための前後処理
 - 補助機能
- (3) 基本ライブラリ（CRAY・CDC の機能の模擬）
 - ビット操作（ブーリアン演算）の機能
 - 非同期入出力機能
 - アドレス通知機能

次に各機能について説明する。

F.1.2 C-エンジンの機能

C-エンジンには、プログラムの変換機能と、解析機能の 2 つの機能がある。C-エンジンの機能一覧を、TABLE.F.1 に示す。解析機能は、現在原研で使用されているツール ANALYSIS-77 とほぼ同じ機能である。ただし、その内容は ANALYSIS-77 に比べてかなり詳しくなっ

いるのでプログラムのデバッグには有効であると思われる。各機能の詳細は、マニュアルを参照されたい。

TABLE.F.1 Functions of STREAM77 (C-ENGINE)

	機 能
変換機能	倍精度化 倍精度化に伴う領域合わせ 長い名前の短縮 ビット操作（ブーリアン演算）の変換 INCLUDE 文の追加 インクルード・データセットの変換 変換記録の表示
解析機能	共通ブロックの宣言状況 プログラム単位の相互参照関係 FORTRAN 関数などの使用状況 未解決外部参照 プログラムの木構造 FORTRAN 文の分類 実引数の検査 共通ブロックの検査 インクルード・データセットのメンバ使用状況

F.1.3 P-エンジンの機能

P-エンジンには、C-エンジン実行前に利用する前処理機能と、実行後のプログラムに対して行う後処理機能、及び補助機能がある。P-エンジンの機能一覧を、TABLE.F.2に示す。各機能の詳細は、マニュアルを参照されたい。

TABLE.F.2 Functions of STREAM77 (P-ENGINE)

	機 能
前処理機能	原始プログラムの分割 (CDC のUPDATEユーティリティのソース・データセット形式の場合) 通貨記号行及び通貨記号による多重文の変換 英小文字を英大文字に変換 CRAY FORTRANベクトル化ディレクティブの変換
後処理機能	C-エンジンでコメント化された文の消去 C-エンジンで短縮化された名前の変更
補助機能	指定された名前の検出 原始プログラムの複写

F.1.4 基本ライブラリ

基本ライブラリは、CRAY, CDC 計算機を持つ特別な機能（ビット演算など）を模擬するファンクションを集めたものであり、TABLE.F.3 に示すものがある。

TABLE.F.3 Basic libraries within STREAM77.

分類	関数名	機能
ビット操作	¥BADD	64ビットの整数加算
	¥BAND	64ビットの論理積
	¥BCSMG	64ビットのselective merge
	¥BDIV	64ビットの整数除算
	¥BEQV	64ビット同値
	¥BI	32ビットの整数を64ビット整数に変換
	¥BMASK	マスクデータの設定
	¥BMULT	64ビットの整数乗算
	¥BNOT	64ビットの論理否定
	¥BOR	64ビット論理和
	¥BSFT	ビットシフト
	¥BSFTB	ビットシフト
	¥BSFTL	左方論理シフト
	¥BSFTR	右方論理シフト
	¥BSUB	64ビット整数減算
	¥BXOR	64ビット排他論理和
	¥IB	64ビットの整数を32ビット整数に変換
	¥IBLZ	左側から先行する0の数をかぞえる
	¥IBPC	1のビット数をかぞえる
	¥IBSGN	64ビット整数の符号判定
¥LEQB	64ビット整数比較 ($a = b$)	
¥LGEB	64ビット整数比較 ($a \geq b$)	
¥LGTB	64ビット整数比較 ($a > b$)	
¥LLEB	64ビット整数比較 ($a \leq b$)	
¥LLTB	64ビット整数比較 ($a < b$)	
¥LNEB	64ビット整数比較 ($a \neq b$)	
非同期入力	¥BUFFI	非同期入力
	¥BUFFO	非同期出力
	¥LENGT	非同期入力で読み込んだデータの長さ
	¥UNIT	非同期入出力での同期を行う
通知ア	¥LOCFB	引数の相対番地を知らせる

F.2 実際の使用例

RELAP 5 / MOD 2 のエラー修正用アップデートカードに対して STREAM77を使用する際には、メンバごと別の実行する必要がある。

C-エンジン実行の際指定したオプションは、次の6個である。各オプションの示す意味についてはマニュアルを参照されたい。

C-エンジン実行時オプション ; NOSUB, NOTAB, NOT, INCLUDE, NOGEN, CONV
ところが、STREAM77が作り出すビット操作ファンクションの名前は、実際に REALP 5 が用いているビット操作ファンクションの名前とは異なっている。そこでP-エンジンの CHANGE オプションを用いて関数名の変更を行う。P-エンジン実行の際指定したオプションは、以下の

3個である。

P-エンジン実行時オプション; CHANGE (), INCLUDE, READ
 ここで READ オプションは, CHANGE オプションを補助入力機番から入力するために指定したものである。CHANGE オプションは, 実際には, Fig.F.1 に示されるものが指定されている。

```

CHANGE ( %BADD = %ADD )
CHANGE ( %BAND = %AND )
CHANGE ( %BI = %DI )
CHANGE ( %BMASK = %MASK )
CHANGE ( %BNOT = %NOT )
CHANGE ( %BOR = %OR )
CHANGE ( %BSFT = %SFT )
CHANGE ( %BSFT8 = %SFT8 )
CHANGE ( %BSUB = %SUB )
CHANGE ( %BXOR = %XOR )
CHANGE ( %IB = %ID )
CHANGE ( %IBSGN = %SIGN3 )
CHANGE ( %LEQB = %EQ )
CHANGE ( %LGEB = %GE )
CHANGE ( %LGTB = %GT )
CHANGE ( %LLEB = %LE )
CHANGE ( %LLTB = %LT )
CHANGE ( %LNEB = %NE )

```

Fig. F.1 An example of CHANGE option
 in STREAM77 (P-ENGINE)

6. お わ り に

スリーマイル島2号原子炉（TMI - II原子炉）の事故を契機にして開始された緊急時関連研究は、5カ年の予定で開始された研究が6カ年に延長された点を除き、当初の目的を満足する成果を得て終了することとなった。

「気象・風洞実験専門部会」の活動成果を用いた研究成果には、原研環境第一研究室における3次元風速場、拡散計算モデルの開発、気象研究所応用気象研究部に於ける力学的気象予測モデルの開発等に反映されている。これらの成果の一部は「緊急時モニタリング・予測専門部会」活動成果報告書⁽¹⁾に述べられている。

本専門部会の活動の中で、6カ年に渡る野外拡散実験は我が国では他に例をみない大規模な実験である。原研では、6年間の実験結果をデータ集にまとめ公開した（2章の参考文献参照）。今後各方面での活用を期待するものである。

本専門部会の活動は、昭和56年度から58年度まで専門部会長を勤められた（故）坂上治郎博士の指導に因るところが大きい。博士の御指導に感謝するとともにご冥福をお祈り申し上げます。

参考文献

- (1) 環境放射能研究委員会、緊急時モニタリング、予測専門部会：緊急時モニタリングと予測計算のための手法並びにシステムの開発、JAERI-M 86-111（1986）