ソフトウェア工学 22－5
（1982. 2. 9）

あるプログラム実行モニターの変遷

牛島 和夫　　河村 豊実

（九州大学 工学部）

1. はじめに

大学の計算センターなどで行われる科学技術計算の大部分は数値計算で占められ，規模も比較的小さく，一人ですべての仕事が行える程度のものが多い．プログラムをする人も，計算機の専門家ではなく問題を持った当人である．このような環境におけるテスト／デバッグ実施上の問題点を列挙してみよう：

（1）テストやデバッグのための手当は，コーディングが一応完了した後，あるいは事故が発生した後など，後追いになりやすい．

（2）デバッグ用の出力は過大になりやすく，その中から必要な情報を抽出するのはかなり面倒である．

（3）デバッグ用出力とソーステキストとの対応付けが不便である．ソーステキスト中のどの出力文によって出力され，その経過がどうかを知るための解析作業は容易ではない．

（4）テスト／デバッグ終了後不要になったテスト／デバッグ用の文を除去する際にトラブルが発生することがしばしばある．

（5）大学計算センターなどの一般ユーザの使用に都合よく設計されたテスト指向型の道具がきわめて少ない．

これらの点に配慮し，文献［1，2，3］などを参考にして，九州大学情報工学科のFACOM 230-45S OSII/VSのバッチ処理環境のもとにテスト／デバッグ支援用の簡単なシステムFORPREX（FORTRAN program execution monitor）を，昭和50年，51年度の卒業研究で試作してもらい［4，5］，51年4月から学科内で使用してきた．使用しながら，しばしば改訂を行い現在では，九州大学大型計算機センターのFACOM M-200 OSIV/F4のTSS環境のもとで使用できるようになっている［6，7］．

以下，第2章でFORPREXの現況を使用例を通じて示す．第3章でFORPREXの発端から現在に至るまでの経緯を述べ，第4章でそれについて考察する．

2. FORPREXの現況

2．1 システムの機能

FORPREXは次のような機能を持っている．以下で，括弧内はそれぞれの機能を使うための文で，特別な注釈行（C&行と称する）に書き，C&文と称する．これをプログラム中の要所に置くことにより，プログラム中の計測点を指定し，さらに計測の対象となる変数や条件などを指定する．

（1）プログラム中の任意の点で指定された条件（拡張された論理式で示す）を判定する．条件に違反した場合には，指定された違反時処理を行う（ASSERT文）．

（2）プログラム中の任意の点で指定された変数または配列要素に入っている値を表示する（DISPLAY文）．

（3）プログラム中の任意の点を，指定された回数通過したところで実行を中断する（HALT文）．

（4）プログラム中の任意の点で指定された変数または配列要素が取る値の範囲を計測する（RANGE文）．

（5）各実行文の実行回数，さらには論理IF文の論理式の値が真になる回数を計数する．

（6）中断点において上記の結果を集計し，会話的選択的に出力する．

（7）異常終了時にもそれまでに得られた結果を（6）にならって出力する．

プログラムの実行が中断点に達するか終了（正常，異常）するかすると，システムは逆追跡情報などを出力してコマンド待ちになる．プログラム実行開始から中断点までに採取された実行に関する情報を，ユーザは，5種のコマンド（D，L，O，R，E，次節参照）を用いて，選択的会話的に出力させることができる．

2．2 使用例

図1に使用例を示す．以下図中の番号を通じて説明する．

（1）プログラムの実行があらかじめ設定したC& HALT文に達すると，実行を中断して逆追跡情報（ここでは，手続きIMPRUV

```
FACOM  OSIV/F4   FORPREX      V03L04                  DATE 82-01-18  TIME 16:23

   **HALT**          AT  IMPRUV
   IMPRUV    CALLED  BY  £MAIN          (1)

            (1)    (2)   (3)    (4)      (5)         (6)          (9)
   ROUTINE CALLED  EXECU-        UNEXE-   EXECUTIONS(  I/O  )      COST     %
                   TABLE(I/O)    CUTED

4& IMPRUV       1      34   0       5        74701        0        74702
1  £MAIN        1      13   2       3           10        2          109

COMMAND ?
D                (2)

            (1)    (2)   (3)    (4)      (5)         (6)          (9)
   ROUTINE CALLED  EXECU-        UNEXE-   EXECUTIONS(  I/O  )      COST     %
                   TABLE(I/O)    CUTED

1  £MAIN        1      13   2       3           10        2          109    0.1
2& DECOMP       1      11   0       0         4095        0         4096    4.2
3  SOLVE        5      11   0       0        15360        0        15365   15.6
4& IMPRUV       1      34   0       5        74701        0        74702   75.9
5  OUTPUT       0       2   1       2            0        0            0    0.0
6& PROBLM       1      12   0       0         4099        0         4100    4.2

** TOTAL                83   3      10        98265        2        98372

COMMAND ?
L IMPRUV         (3)

EXECUTIONS      FORTRAN  STATEMENT     (IMPRUV)
     TRUE*
        1       SUBROUTINE IMPRUV(NN,M,A,UL,B,X,R,DX)
                DIMENSION A(M,3),UL(M,3),B(M),X(M)
                DIMENSION R(M),DX(M)
                DATA EPS,ITMAX/1.E-6,20/
          C
        1       N=NN
        1       XNORM=0.
        1       DO 1 I=1,N
     1023         XNORM=AMAX1(XNORM,ABS(X(I)))
     1023     1 CONTINUE
        1       ITER=0
        1       IF(XNORM .LE. 0.0) GO TO 10
       0*
        1       DO 9 ITER=1,ITMAX
        4         DO 5 I=1,N
     4092           SUM=0.DO
     4092           DO 4 J=1,3
    12276             JJ=I+J-2
    12276             IF(JJ .EQ. 0)  GO TO 4
        4*
    12272             IF(JJ .EQ. N+1)GO TO 4
        4*
    12268 C&                              ASSERT (1.LE.JJ .AND. JJ.LE.N)
          C&  +                             ELSE DISPLAY (I,J,JJ)  HALT 1

                (4)             * VIOLATION COUNT          0

    12268               SUM=SUM+DBLE(A(I,J))*X(JJ)
    12276       4     CONTINUE
     4092             R(I)=B(I)-SUM
     4092       5   CONTINUE
          C
        4       CALL SOLVE(N,M,UL,R,DX)
          C
        4       DXNORM=0.0
        4       DO 6 I=1,N
     4092         T=X(I)
     4092         X(I)=X(I)+DX(I)
     4092         DXNORM=AMAX1(DXNORM,ABS(X(I)-T))
     4092     6 CONTINUE
        4 C&                                DISPLAY 10 (DXNORM) HALT 4

                (5)             * PASS     (          1)
                                   DXNORM=  0.3153539E-01;

                                * PASS     (          2)
                                   DXNORM=  0.1546085E-02;

                                * PASS     (          3)
                                   DXNORM=  0.8249283E-04;

                                * PASS     (          4)
                                   DXNORM=  0.4768372E-05;

                                *    HALT  POINT

        4       IF(DXNORM .LE. EPS*XNORM) GO TO 10
       0*
        3     9 CONTINUE
        0       ITER=ITMAX+1
        0    10 CONTINUE
        0       RETURN
                END

COMMAND ?
L DECOMP&        (6)

EXECUTIONS      FORTRAN  STATEMENT     (DECOMP)
     TRUE*
     1022 C&                              RANGE  ( EM )

                                  (EM    ) FIRST=  0.5000002
                                           LAST =  0.9995500
                                           MAX  =  0.9995500
                                                              (PASS=      1022)
                                           MIN  =  0.5000002
                                                              (PASS=         1)

COMMAND ?
R                (7)

FACOM  OSIV/F4   FORPREX      V03L04                  DATE 82-01-18  TIME 16:28

   **HALT**          AT  IMPRUV
   IMPRUV    CALLED  BY  £MAIN
```

以 下 略

図 1. FORPREX 使用例

と#MAIN）とこれらの手続きに関連する統計情報（後述）を出力してコマンド待ちとなる．

（2）D（isplay）コマンドにより，プログラム中の全手続きに関する統計の一覧表を出力する．この表はユーザにとって，プログラムに関する手続きレベルの大略を知り次の行動に移る指針となる．この一覧表は左欄から，手続き名とその参照番号，呼出回数（第1欄），実行文の数（第2欄），入出力文の内数（第3欄），未実行文の数（第4欄），実行回数の和（第5欄），入出力文の実行回数の和（第6欄），実行コストの推定値とその百分率（第9欄）から成っている．なお，最も左側の参照番号の右側についた&記号は，その手続き中にC&文が含まれていることを示す．

（3）L（ist） IMPRUVコマンドにより IMPRUV手続きのリストを実行回数情報および採取情報（C&情報と称する）付きで出力する．出力リストの左側にある数字の列が右側の各実行文の中断点に至るまでの実行回数である．右側に＊印を付された数字は論理IF文が真となった回数を示す．ここで実行回数0の文の個数が，Dコマンドによる一覧表中第4欄にまとめられている．また，各実行文の実行回数の手続き毎の総和が第5欄にまとめられている．C&情報は，C&行の直後に出力され，プログラムテキストとの対応づけを容易にしている．図1で（4）および（5）がC&情報である．

（4）特別な注釈行

```
C&     ASSERT(1.LE.JJ.AND.JJ.LE.N)
C&   +   ELSE DISPLAY(I,J,JJ) HALT 1
```

により，この点の直前で，1≦JJ≦Nが成立していることを表明している．ELSE以下は，この表明に違反したときの処置を指示するもの．

```
DISPLAY (I,J,JJ)
```

により，違反発生時にI，J，JJの値を表示することを指定し，

```
HALT 1
```

により，違反が1回生じたら実行を中断することを指定している．

ここでは12268回ASSERT文が検査されたが違反は1回もなかったことを，

```
*  VIOLATION COUNT        0
```

によって示している．

（5）前記（4）ASSERT文のELSE以下を独立させたもの．

```
C&     DISPLAY 10 (DXNORM)  HALT 4
```

この点を通過するときに，DXNORMの値を表示することを指示している．10は最初の10回通過するときだけ出力し，11回目以降は出力を抑制する．

次のHALT 4は，この点を4回通過したところで実行を中断することを指示している．この手続きは，DXNORMの値の収束を意図している．DISPLAY文の以下に出力された値によって，第1回から第4回までの，DXNORMの値が，0.31E-01，0.15E-02，0.82E-04，...，と収束してゆく様子を確認できる．

（6）手続き名（またはその参照番号）の次に&を付けて，C&情報のみ出力させることができる．ここでは，L DECOMP&（L 2&としても同じ），SUBROUTINE DECOMP中の特別な注釈行

```
C&     RANGE (EM)
```

により，この点を通過する直前に，変数EMに得られている値を計測して初期値，最終値，最大値，最小値を記憶しておくことを指示する．EMの最大値は0．9995500で，1022回目の通過の際に実現したことがわかる．EXECUTIONS欄の値から，このDOループの本体は1022回繰り返されており，最大値は最終値でもある．

なお，C&情報が不要な場合には L DECOMP+ のように指令すれば実行回数情報付きのリストだけが得られる．

（7）R（esume）コマンドにより実行の再開を指令する．

このほか，実行の結果得られた中断点までの出力結果をまとめて出力させるO（Output）コマンドと，モニターの終了を指示するE（End）コマンドが用意されている．

手続きに関する一覧表の出力を指令するDコマンドについて説明を加える．この表はプログラムを手続きレベルで大局的に把握するための情報を提供する．ユーザは着目する欄の数字をみて，必要な手続きのリスト（L コマンド）などを行えばよい．しかし，手続きの数が多くなるとその判断も大変になるから，Dコマンドにパラメータをつけて，各欄ごとに数字の大小順に上位n個の手続きの情報のみを選択的に出力することなどを可能にしている．

ASSERT文の仕様を以下にまとめる．以下で［ ］内は省略可能，｛ ｝内は選択を示し，｛ ｝内の下線は，標準値を示す．

```
C&     ASSERT <拡張された論理式>
          [ELSE] [<DISPLAY指定>] [<HALT指定>]
```

<拡張された論理式>は｛＋｜＊｝（<添数付き論理式>）とする．

ただし<添数付き論理式>は，＜FORTRAN論理式>か（<添数付き論理式>，＜DO型仕様>）とする．なお＜DO型仕様>とは，i＝m1，m2，m3か，i＝m1，m2でiは制御変数で3重まで付け加えることを許す．式の前につけた＋（プラス）は，各DO型仕様の制御変数に対するFORTRAN論理式の論理和，＊（スター）は論理積を表わす．

プログラムの実行の制御がC&文の直前まで到達すると，その点での<拡張された論理式>を評価し，その値が真であればそのまま次の文に制御が移る．偽ならば違反が検出されて，ELSE以下にある＜DISPLAY指定>および（または）＜HALT指定>の処理を行って，次の文に実行の制御を移す．

＜DISPLAY指定>は次のとおり．

DISPLAY［n1［,n2［,n3］］］(v1,...,vm)

ここで，n1，n2，n3は正整数定数．v1，．．．，vmは変数名または配列要素名の並びである．

この指定により，n1回目の違反からn2回目の違反まで，n3回ごとに，v1，．．．，vmの値を表示することを指示する．n3を省略するとn3＝1とみなす．n2を省略すると，1，n1，1と解釈する．全部省略した場合は1，5，1と解釈する．

変数または配列要素の値は各型に対応した標準の書式で出力される．なお　'変数名と書くと，指定された変数の値を文字とみなして，文字出力を行う．また，$変数名と書くと，データを16進書式で出力する．なお，'変数名($変数名)は　'配列要素名($配列要素名)としてもよい．

＜HALT指定>は次のとおり．

HALT　［n］

nは正整数定数．省略されるとn＝1と解釈する．この指定により，違反がn回起きたら，プログラムの実行を中断することを指示する．

＜DISPLAY指定>と＜HALT指定>の記述の順序・有無は任意である．

3．FORPREXの変遷

前章に述べた姿にたどりつくまでにFORPREXは四つの段階を経由した．

3．1　第I段階　ASSERT文の実現

このようなシステム作成の動機はいろいろある．第1章に挙げたようなテスト／デバッグにおける問題点の解決を助けるツールを実現できないか，ソーステキスト中の実行文と注釈との不一致がしばしばみられることから注釈を何らかの形で解釈できるツールが実現できないか，そこでALGOL Wのためのデバッギングシステム［1］の中にあるａｓｓｅｒｔ文をFORTRANの中に埋め込むことを考え，昭和50年度の卒業研究でFACOM 230 OSII/VSのもとにFORPREXの原型を設計し作成してもらった［4］．最初のFORPREXに含まれていたのはASSERT文のみである．"拡張された論理式"（いわゆる表明言語）の文法がまず問題になったが，使用目的と実現上の兼ねあいから第2章に示したものにとどめた．

FORPREXは，ASSERT文を含むソーステキストを前処理－コンパイル－結合処理－実行－事後編集を行ってFORPREXリストを出力する．前処理は，C&行を識別して，ASSERT文に含まれる拡張された論理式を実行文に展開し，論理式の値が偽の場合に選択的な処理を行う文を埋め込む．

プロトタイプの作成，試用を通じて次のようなことを経験した．

（1）この程度の道具でも，それがあるだけでプログラム設計時にASSERT文を用意させる動機づけになる．

（2）形式的検証を行っているわけではないから，C&論理式の真偽に拘わらず適切なテストデータを何組か用意する必要がある．

（3）C&行は用済後もそのまま注釈として残すことができる．プログラムを変更したときは注釈行（C&行）の変更が必須である．

（4）拡張された論理式の表現がFORTRAN文法の制約を受ける．この論理式に関数呼び出しを含むと副作用の恐れがある．

（5）ループ中にASSERT文を含むと非常に時間がかかる場合がある．

（6）出力結果とソーステキストとの対応づけのよさは予定通り．

3．2　第II段階　RANGE文の追加

第I期プロトタイプの使用経験に基づきいくつかの機能拡張を昭和51年度の卒業研究として試みた［5］．

（1）DISPLAY文の独立．　DISPLAY文は，従来，拡張された論理式が偽の時に限って効力を発揮した．通過値を見たいだけの目的で，論理式の値を故意に偽にするような使用法があったため，この文を独立させた．

（2）16進出力，文字出力．　互換性の観点から従来採用していなかった．システム記述用にFORTRANを使用する側からのの要求．

（3）RANGE文の新設．これはPET［2］を参考にした．PETには，プログラム中のすべての代入文の左辺の変数について初期値，最終値，最大値，最小値を自動的に計測する機能がある．これを実現するには莫大な記憶容量が必要となるので，注目する変数の数と計測点とをRANGE文により限ることにした．

このシステムは設計当初どちらかといえばテスト支援の方に重点が置かれていた．ASSERT文を適切に挿入するのはかなりのプログラミング能力が要求されるので，実際使用に供してみると，手軽に中間結果の出力が得られること，ソーステキストとモニター出力との対応が容易なことが評価されており，設計意図とは逆に，むしろデバッグ向きの道具として使用されている．第II段階の機能拡張はこの方向にある．

3．3　第III段階　バッチ処理環境での統合

若干の機能拡張と整備を行い，これをさらに九大大型計算機センターのFACOM OSIV/F4のもとに移し換え，一般利用者の使用に供することにした［6］．

（1）C& SUPPRESS文の追加．FORPREXを使用するたびに全てのソーステキストを出力するのが煩わしいので，手続きごとにC& SUPPRESS文をおくことを可能にしてその手続きのソーステキストのリスト出力を抑制することができるようにした．

（2）全ての実行文の実行回数を計数することにした．第I段階の前年の卒業研究によるプロトタイプをもとにFORTRANプログラム実行回数計数ツールFORDAPを作成し使用していた［3］ので，第I期の試作時には，実行回数の計数はFORDAPで，ASSERT文はFORPREXで，というツールの分業を考えていた．しかし，両者が一体となっている方がはるかに便利なので，すべての文の実行回数を計数することにした．

（3）手続きに関する一覧表の出力．FORDAPを使用していて，プログラム全体をまず手続きごとに把握する情報がほしいと感じていたために一覧表を作成し，これをFORPREX出力の最後に付加することにした．

3．4　第IV段階　TSS環境への適応

プログラムの実行が進むにつれて実行回数情報が電光ニュースのように変化しながら表示されてゆくシステムをTSS環境のもとに実現できないかというのが発端であった．FORPREXの中にあるHALT文の機能を変更して利用することにした．電光ニュースというわけにはゆかないが，バッチ処理環境で使用していて不便な点をいくつか解消することになった［7］．最大の改善点は，手続きに関する一覧表を経由して，必要な計測情報をトップダウンに選択して出力できるようになったことであろう．デバッグ用ツールにおける過大になりがちな出力情報を抑制することがTSS環境に適応させることによってさらに容易になった．システムとの会話を容易にするために，手続きに関する一覧表に，手続き参照番号を付け加えたこと，C&文を含む手続き名を明示したこと，各欄に参照番号を付け加えたこと，そして，テストの網羅性を知るために未実行文の数欄を増設したことなどである．SUPPRESS文は不要になった．このほか，端末機器に応じて，80桁用と132桁用と二通りの出力様式を可能にした．

4．考察

4．1　ソフトウェアのライフサイクル

第3章に述べた様に現在のFORPREXは最初から現在の姿を想定した要求仕様を掲げてつくったシステムではない．ASSERT機能をとにかく実現してみる．それを試用して問題点を認識する．それに基いて改訂を行う．各機能の効用や問題点を認識した上で第1章に述べたテスト／デバッグ上の問題点に鑑みて計算センターユーザの使用を前提としたシステムに組上げる．計算機の使用環境の変化に適応させるべく改訂を行う．というライフサイクルを辿ってきた．振り返って卒業研究によるプロトタイプの作成を高く評価することができる．機能を実現する上での技術的問題は，ソフトウェア作成上からは二義的である．勿論，ものを一つ作ってドキュメントを的確に用意するという経験を積ませることは，教育的立場から重要である．

最初バッチ処理環境のもとでスタートしたものを四つの段階を経て現状までとにかく適応させ得た大きな要因は，第一に，プロトタイプ作成に関与した学生があったものの，作成担当者が変わらなかったこと．第二に，他システムへの移し換えを考慮して，システムの記述をFORTRAN（のちにPortable FORTRAN）で行っていたこと，が挙げられる．

4．2　実行モニターとしての位置づけ

FORPREXは，情報採取点の指定，変数の値の表示，実行の中断などの機能を有しているが，変数の値を設定する機能，プログラムの一部を書き変える（パッチを当てる）機能を有していない．最近のFORTRAN処理系には上のような設定や変更を可能にする機能が

用意されている［8］．よく計画を立てないまま思いつきで変数の値を設定したり，プログラムにつぎを当てたりすることは，プログラム作成にとって危険なことが指摘されている．ＦＯＲＰＲＥＸではこれらを可能にすることを敢えて行わず，上のような危険を避けてみた．

プログラムの誤りの発生個所を予見することは難しい．プログラム全体，あるいは指定した手続き全体にわたって，添字の範囲外，変数の値の異常など，網をはっておく機能があれば，誤り発見の手段として有用である．システムをＦＯＲＴＲＡＮで記述しているため，変数の物理的な記憶場所を知ることが不可能であるなどの理由で，これらの機能をＦＯＲＰＲＥＸに実現することはシステムをいたずらに肥大させるだけであり，被測定プログラムの実行速度も著しく低下させる．全体にわたる計測は実行回数の計測にとどめた．

ユーザが予めプログラム中の要所を選定して計測点を設定する作業は，対象プログラムに対する確かな理解がなければ難しい．ＡＳＳＥＲＴ文の適切な使用はことのほか難しい．筆者らは，ＡＳＳＥＲＴ文抜きのＤＩＳＰＬＡＹ文をよく用いる．プログラムの改訂に当たって改訂前と改訂後の値のチェックを要所ごとに行う場合などに，計測結果とソーステキストとの対応付けのよさを利用する．また，他人のつくったプログラムの理解（保守作業の一環として）のため，要所にＤＩＳＰＬＡＹ文を挿入することもよく行う．

4．3　今後の問題

4．2節に述べたこと以外にもＦＯＲＰＲＥＸにはまだ，さまざまな問題があり，それらを列挙する．

（1）　ＦＯＲＴＲＡＮ７７への対応，

（2）　他の計算機システムへの移し換え，

（3）　採取情報のデータベース化，

（4）　計測点の会話的変更．

ＦＯＲＰＲＥＸは，九大大型計算機センターと情報工学科と，いわば作成者の目の届くところでのみ動いており依然としてプロトタイプの域を脱していない．このシステムをさらに改良するよりはこの実験システムの作成および使用経験に基づき，最初からＴＳＳ環境（あるいは分散処理環境）を想定した別のシステムの設計・構築を考えた方がよい．手続き間情報を解析してデータベース化し実行回数情報とともに整理して提示するツールＡＵＤＩＥ［9］の設計・作成にも，上の経験が反映している．

なお，この研究の一部は文部省科学研究費補助金（昭和５５－５６年度一般研究）の援助を受けた．

参考文献

［1］Ｓａｔｔｅｒｔｈｗａｉｔｅ，Ｅ．：Ｄｅｂｕｇｇｉｎｇ Ｔｏｏｌｓ ｆｏｒ Ｈｉｇｈ Ｌｅｖｅｌ Ｌａｎｇｕａｇｅｓ，Ｓｏｆｔｗａｒｅ－Ｐｒａｃｔ．＆ Ｅｘｐｅｒ．，Ｖｏｌ．２，ｐｐ．１９７－２１７，１９７２．

［2］Ｓｔｕｋｉ，Ｌ．Ｇ．ａｎｄ Ｆｏｓｈｅｅ，Ｃ．Ｌ．：Ｎｅｗ Ａｓｓｅｒｔｉｏｎ Ｃｏｎｃｅｐｔｓ ｆｏｒ Ｓｅｌｆ－Ｍｅｔｒｉｃ Ｓｏｆｔｗａｒｅ Ｖａｌｉｄａｔｉｏｎ，２ｎｄ Ｉｎｔｎｌ．Ｃｏｎｆ．ｏｎ Ｒｅｌｉａｂｌｅ Ｓｏｆｔｗａｒｅ，１９７５．

［3］藤村，牛島：プログラムの実行解析システムの作成と使用について，九大工学集報，Ｖｏｌ．４８，ｐｐ．９５－１０１，１９７５．

［4］牛島，河村，見戸：デバッグ／テスト支援システムの作成，Ｓ５１電気四学会九州支部連大，１９７６．

［5］牛島，河村，武富：デバッグ／テスト支援システムＦＯＲＰＲＥＸの機能拡張について，Ｓ５２電気四学会九州支部連大，１９７７．

［6］牛島，河村：ＦＯＲＴＲＡＮプログラムテスト／デバッグ支援のためのシステムＦＯＲＰＲＥＸの使用について，九大大型計算機センター広報，Ｖｏ．１１，ｐｐ．２７０－２８３，１９７８．

［7］牛島，河村：ＦＯＲＴＲＡＮプログラム実行モニターＦＯＲＰＲＥＸの会話的使用について，Ｓ５５電気四学会九州支部連大，１９８０．

［8］富士通：ＦＡＣＯＭ ＯＳＩＶ／Ｆ４ ＦＯＲＴＲＡＮインタラクティブデバッグ使用手引書，６４ＳＰ－３２００－１．

［9］牛島，田町：手続き間情報の解析と整理のツールＡＵＤＩＥについて，情報処理学会論文誌，Ｖｏｌ．２３，Ｎｏ．１，ｐｐ．８１－８７，１９８２．