フルスクリーン表示によるFORTRANプログラムデバッグシステム

高橋和彦，麻生利紀，小林正和，山口誠　（富士通株式会社）

1. はじめに

近年，プログラムの開発量は著しく増大し，また多様化してきている。それと共に，プログラム開発における生産性と信頼性の向上は今日，最も重要なテーマの一つとなってきている。

本稿では，FORTRANプログラムを対象とし，プログラムの開発時のテスト・デバッグ段階における生産性と信頼性の向上を狙いとした会話型ツールについて紹介する。

本システムは，“プログラム実行のスロービデオ表示”という新概念のマンマシンインタフェースを提供することにより，従来ブラックボックスの中の出来事であったプログラム実行を視覚表示することを可能にした。

2. 機能概要

本システムの特長的機能は以下のとおりである。

- プログラム実行速度の制御
- プログラム実行の視覚表示
- メニュー方式による入力
- 実行モニタ
- スナップ文
- デバッグ情報の保存と再表示
- リソース保存による簡易実行機能
- プログラムの再実行

2.1 プログラム実行速度の制御

マシンとの対話形式によるプログラムのデバッグで，最も重要な点は，人間の思考のリズムに合せたプログラム実行である。本システムでは，利用者の指示によって，プログラムの実行速度を制御する手段を提供している。

（1）実行モードによる制御

本システムでは，“実行モード”をプログラム単位（主プログラム／副プログラム）ごとに指定することにより，最適な実行速度を選択できる。
実行モードは，次の4種類である。

- ステップモード
- 低速モード
- 中速モード
- 高速モード

上記の実行モードは，原始プログラム形式で入力されるプログラム単位に対して選択できる。

プログラムの実行中に実行時エラー（除算例外など）の発生など，一時的に実行を中断することがある。この状態を“静止モード”と呼ぶ。実行モードは，プログラムが静止モードとなった段階で任意に変更できる。

実行モードは，次項で述べる“プログラム実行の視覚表示”と密接に関連している。
実行モードごとの特長と相違を表－1に示す。

（2） 実行の一時的な高速化

実行モードは，プログラム単位ごとに規定される。実行モードは，プログラムが静止モードとなった時点で変更できるが，本システムでは更に次のような手段を提供し，より柔軟な実行速度の制御を可能にしている。

- DOループ内の高速実行
  一定のキー（PFキー）を押すと，ループ内をループ回数分だけ一時的に高速実行する。ループを抜け出した後は，実行モードに従う。

- 指定位置までの高速実行
  指定のプログラム名又は文（行番号指定）に到達するまで一時的に高速実行する。指定された副プログラムの入口又は文へ到達すると静止モードとなる。

表－1　実行モードごとの特長と相違

比較項目 ＼ 実行モード	表示形式	静止タイミング
ステップモード	文単位に輝度又は色を変えて表示	一文ごと （キーを押すと次の文へ進む）
低速モード	文単位に輝度又は色を変えて表示 （文単位の動きを追える）	文番号を持つ文， ブロックIF文， DO文など
中速モード	プログラム単位ごとの表示 （文単位の動きは追えない）	プログラム単位の先頭と最後
高速モード	表示なし	静止しない

2.2 プログラム実行の視覚表示

プログラムの実行の様子を視覚的に表示することはプログラムの実行論理を検証する上で大きなメリットがある。

本システムでは，文字ディスプレイ装置のフルスクリーン機能を利用して原始プログラムリストを表示し文ごとの実行の様子を，輝度（カラーディスプレイの場合は色）の変化で示す。表示形式は，実行モードにより異なる。

原始プログラムを表示する画面は通常，実行中のプログラム単位のプログラムリストを表示する。

プログラム実行の様子を表示する画面を，実行表示画面と呼ぶ。実行表示画面の例を図－1に示す。

2.3 メニュー方式による入力

入力するデータセット名などの指定は，いわゆるメニュー方式で行なえる。メニューは逐次表示され，利用者は，メニューに従って入力を完了させることができる（メニューの選択も可能である）。メニュー方式の利点は何をどのように指定すべきかの情報が画面に表示されているので初心者でも容易に使いこなすことができる点である。（指定方法がわからない場合は，ＨＥＬＰにより確認すればよい。）

実行時に使用するデータセット，端末装置などの割当て，割当てられたデータセットの参照及び割当ての解除もメニュー上で行なえる。メニューの例を図－２に示す。

```
EXECUTION SCREEN  ARGCHK => N  SUBCHK => N  FREQ => N  (Y/N)  MODE =>   S(L)M H
==>
   SCROLLS  :  ROW( P  )  COL( 20 ) ,  ROW( P  )  COL( 20 )   VLINE  :  ( 42 )
       ----*----1----*----2----*----3-  |  MAIN                              (NEST)
  00600     READ(UNIT=5,FMT=*) VS,COEF,U |AT 600 PLEASE, INPUT DATA. EXECUTION M
  00700     G = 9.8                      |16.0,0.8,8.0,0.25
  00800     FTIMEO = 0.                  |T (SEC)   X (M)  0---2---4---6---8---A
  00900     FSPD = VS                    | 0.0       0.0   *
  01000     FTIME1 = 2.*FSPD/G           | 0.25      3.69        *
  01100     WRITE(UNIT=6,FMT=100) SCALE  | 0.50      6.78                *
  01200     DO 10 T = 0.,UBND,NOTCH      | 0.75      9.24                    *
  01300       IF(T.GT.FTIME1) THEN       |
  01400         FTIMEO = FTIME1          |
  01500         FSPD = COEF*FSPD         |
  01600         FTIME1 = FTIMEO+2.*FSPD/ |
  01700       ENDIF                      |
  01800       X = FSPD*(T-FTIMEO)-G*(T-F |
  01900       K = INT(2.*X+0.5)          |
  02000       ASTER(K+1:K+1) = '*'       |
  02100       WRITE(UNIT=6,FMT=200) T,X, |
  02200       ASTER(K+1:K+1) = ' '       |
  02300 10  CONTINUE                     |
  02400     STOP                         |
  02500 100 FORMAT(1X,'T (SEC)',3X,'X ( |
```

□内が実行中の文であることを示す。

図－１　実行表示画面の例（低速モードの例）

```
-----------------<  DOCK/FORT77 SOURCE MODULE DATA SET MENU  >-----------------
ENTER/VERIFY PARAMETERS BELOW:

DOCK LIBRARY DATA SET:
  PROJECT ===> USER1
  LIBRARY ===> SRC
  TYPE    ===> FORT77
  MEMBER  ===> PROG1      ===>            (BLANK FOR MEMBER SELECTION MENU)

OTHER PARTITIONED OR SEQUENTIAL DATA SET:
  DATA SET NAME ===>
  VOLUME SERIAL ===>            (IF NOT IN CATALOG)

DATA SET PASSWORD ===>          (IF PASSWORD PROTECTED)




PRESS ENTER TO REPEAT THIS MENU
PRESS PF3   TO GO FORWARD TO THE NEXT MENU
PRESS PF4   TO TERMINATE THE DOCK SESSION
```

図－２　メニューの例（データセット名の指定）

2.4 実行モニタ

実行モニタには，データの振舞とプログラムの振舞に着目した機能がある。

（1） データモニタ

本システムでは，次のデータモニタ機能を備えている。

a． 実引数と仮引数の数・型及び類の対応を検査する。
b． 配列要素の添字式及び文字部分列の部分列式の範囲について正当性を検査する。
c． 利用者の指定する変数などの値の変化を監視する。

ここでは，特に，c．について述べる。利用者は，図－３に示すメニューによって，次の二通りの方法でプログラム実行中のデータの監視条件を定義することができる。

- 指定した変数・配列要素及び文字部分列の値が変化する時点を監視する。
- 指定した論理式の評価結果が真となる時点を監視する。

これらを，“モニタ条件”と呼ぶ。また，図－３のメニューで上記の指定をするとき，モニタ条件の成立時に静止モードとするか，通知（値の表示など）のみで実行を継続させるかを選択できる。その他，一時的にモニタ条件の検査をやめることもできる。

（2） プログラムモニタ

プログラムの実行ごとに得られる情報として，文ごとの実行回数表示がある。この機能は，特に実行・未実行パスを容易に検出できるので，プログラムのテスト時あるいは，プログラムの論理的な検証を行うときに役立つ。図－４にその例を示す。

2.5 スナップ文 — 動的な値の変更と参照及び制御の移行など

プログラムの実行中に静止モードとなった段階で，プログラム中の変数などの値の参照や変更あるいは，制御の移行を行うことができる。値の変更及び参照の指示は，“スナップ文”と呼ぶＦＯＲＴＲＡＮ文の形式で行う。

スナップ文は，原始プログラムを変更することなく，入力された時点で直ちに実行されるので強力なデバッグ手段となる。

スナップ文として，以下のものが用意されている。

- 代入文
- ＷＲＩＴＥ文（端末への出力）
- 論理ＩＦ文
- ＧＯＴＯ文
- ＳＴＯＰ文

ＧＯＴＯスナップ文では，文番号を持つ実行文へ制御を移行させることができる。文番号を持たない実行文への制御の移行を可能とするため，ＧＯ行サブコマンドが用意されている。ＧＯ行サブコマンドは，実行表示画面の行番号欄へ指定する。

スナップ文の例を，図－５に示す。

2.6 デバッグ情報の保存と再表示

実行表示画面では，プログラムの実行結果，本システムからのメッセージなどのデバッグ情報が表示される。（原始プログラム表示域の右側）
これらの情報は，自動的に大記憶上に蓄えられる。利用者は，スクロール機能を用いて，以前のデータを画面上に表示させたり，リスト出力させることができる。

```
------------------< DOCK/FORT77 DATA MONITOR DEFINITION MENU >------------------
===>

         ----*----1----*----2----*----3----*----4----*----5----*----6----*----7-
   00040       NUM = 0
   00050       READ(5,*) N
   00060       DO 10 I=1,N
   00070         IF(ARY(I).NE.1) GOTO 10
   00080         DO 20 J=I,N,I
   00090           ARY(J)=I
   00100    20   CONTINUE
   00110         NUM = NUM + 1
   00120         ARY(NUM) = I
   00130    10 CONTINUE
-------+----------+-------------------------------------------------------------
STATUS |   NAME   | CONDITIONS ( STATUS : M - MONITOR, T - TRACE, S - SUSPEND )
-------+----------+-------------------------------------------------------------
 M     | TRACE    |  NUM
       |          |
       |          |
       |          |
       |          |
       |          |
PRESS PF3 TO TERMINATE THIS MENU, PRESS PF4 TO TERMINATE THE DOCK SESSION
```

図－３ モニタ条件定義メニュー

```
EXECUTION SCREEN  ARGCHK => N  SUBCHK => N  FREQ => Y  (Y/N)  MODE =>   S L M H
==>
   SCROLLS  :  ROW( P  )  COL( 20 ) ,  ROW( P  )  COL( 20 )   VLINE  :  ( 42 )
COUNT      ----*----1----*----2----*---
     00020       INTEGER*4  ARY(100)
    1 00030       ARY(1) = 1
    1 00040       NUM = 0
    1 00050       READ(5,*) N
    1 00060       DO 10 I=1,N
   50 00070        IF(ARY(I).NE.1) GOTO 1
   16 00080        DO 20 J=I,N,I
  127 00090         ARY(J)=I
  127 00100    20  CONTINUE
   16 00110        NUM = NUM + 1
   16 00120        ARY(NUM) = I
   50 00130    10 CONTINUE
    1 00140       WRITE(6,100) N
    1 00150       WRITE(6,200) NUM
    1 00160       WRITE(6,300) (ARY(I),I=
    0 00170       STOP
      00180   100 FORMAT('  PRIME NUMBERS
      00190   200 FORMAT('  NUMBER IS',I3
      00200   300 FORMAT(4I4)
    0 00210       END
```

```
AT 50 PLEASE, INPUT DATA. EXECUTION MO
50
 PRIME NUMBERS, FROM 1 TO 50
 NUMBER IS 16
  1   2   3   5
  7  11  13  17
 19  23  29  31
 37  41  43  47
```

図－４　実行回数の表示

```
EXECUTION SCREEN  ARGCHK => N  SUBCHK => N  FREQ => N  (Y/N)  MODE =>   S L M H
==> N=100
   SCROLLS  :  ROW( P  )  COL( 20 ) ,  ROW( P  )  COL( 20 )   VLINE  :  ( 42 )
       ----*----1----*----2----*----3-
  00020       INTEGER*4  ARY(100)
  00030       ARY(1) = 1
  00040       NUM = 0
  00050       READ(5,*) N
  GO 60       DO 10 I=1,N
  00070        IF(ARY(I).NE.1) GOTO 10
  00080        DO 20 J=I,N,I
  00090         ARY(J)=I
  00100    20  CONTINUE
  00110        NUM = NUM + 1
  00120        ARY(NUM) = I
  00130    10 CONTINUE
  00140       WRITE(6,100) N
  00150       WRITE(6,200) NUM
  00160       WRITE(6,300) (ARY(I),I=1,N
  00170       STOP
  00180   100 FORMAT('  PRIME NUMBERS, F
  00190   200 FORMAT('  NUMBER IS',I3)
  00200   300 FORMAT(4I4)
  00210       END
```

```
 MAIN                                (NEST)
AT 50 PLEASE, INPUT DATA. EXECUTION MO
50
 PRIME NUMBERS, FROM 1 TO  50
 NUMBER IS 16
  1   2   3   5
  7  11  13  17
 19  23  29  31
 37  41  43  47
```

図－５　スナップ文の例

2.7　リソース保存による簡易実行機能

本システムでは，プログラムの実行開始に必要な実行環境（リソース）を大記憶上に保存しておき，セション再開後，メニュー上での簡単な指示により，プログラムの先頭から実行させる機能を備えている。この機能は，次のような利用方法がある。

（1）ＴＳＳサービス時間の終了間際に，実行環境を保存するジョブを＂ＳＵＢＭＩＴ＂しておき，次のＴＳＳサービス時間に，プログラムの実行を開始させる。

（2）環境保存を行った後，プログラムの実行を一度中止し，後刻改めて実行をやり直す。

保存するリソースは，次の三つである。

- 利用者プログラムのオブジェクトモジュール（指定の原始プログラムを翻訳したもの）
- 画面表示に必要なソースモジュール
- 各種の内部テーブル類

セションの再開後は，一つのメニュー上で上記の情報が格納されているデータセット名を指定し，実行を開始させる。

2.8　プログラムの再実行

本システムでは，テスト・デバック対象のプログラムを再実行させる方法として，以下の手段を用意している。

（1）　分岐による再実行
（2）　プログラムの先頭からの（単純）再実行
（3）　コンパイラオプション等の条件を変更した後の再実行
（4）　原始プログラムを修正した後の再実行

分岐による再実行とは，ＧＯＴＯスナップ文又は，ＧＯ行サブコマンドにより，任意の文まで制御を戻して再実行させる方法である。

プログラムを最初から単に再実行させるには，サブコマンドの＂ＲＥＲＵＮ＂を指定する。この場合，既に定義されているモニタ条件などは，再実行後もそのまま保持される。

（3）及び（4）は，いずれも一度プログラムの実行を終了させ，図－６のメニュー（動作選択メニュー）により，実行の条件もしくは原始プログラムを変更した後，再実行させる方法である。

```
--------------------< DOCK/FORT77 ACTION SELECTION MENU >----------------------
OPTION ===>


   1 - PERFORM THE SOURCE MODULE DATA SET SPECIFICATION
   2 - PERFORM THE INCLUDED SOURCE MODULE DATA SET SPECIFICATION
   3 - PERFORM THE OBJECT/LOAD MODULE DATA SET SPECIFICATION
   4 - PERFORM THE AUTOMATIC CALL LIBRARY DATA SET SPECIFICATION
   5 - PERFORM THE COMPILER OPTIONS SPECIFICATION
   6 - PERFORM THE EXECUTION OPTIONS AND ENTRY NAME SPECIFICATION
   7 - PERFORM A DATA SET EDITTING









PRESS PF3 TO RESTART THE PROGRAM EXECUTION PROCESS
PRESS PF4 TO TERMINATE THE DOCK SESSION
```

図－６　動作選択メニュー

3.　実現方式

本システムの基本的な構成要素は，以下の三つである。

- 主制御部
- 実行制御部
- 画面制御部

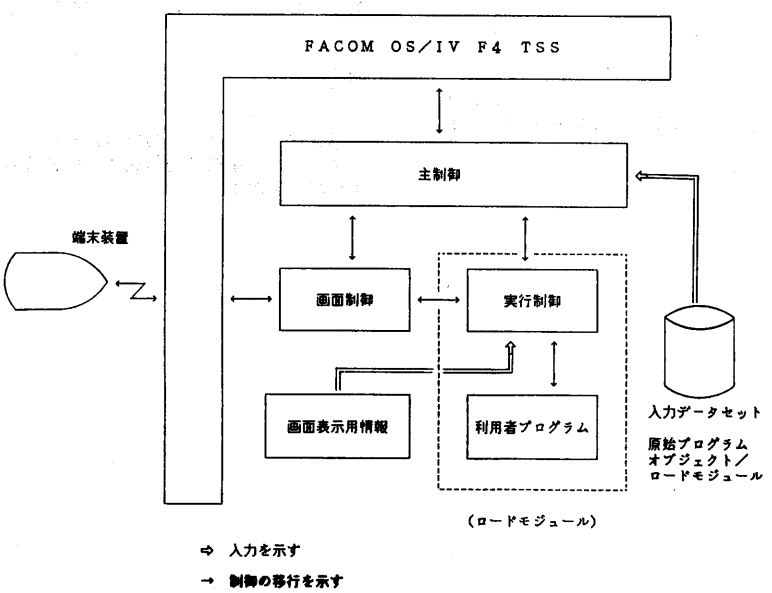
主制御部は，各種データセット名や利用者からの指示の受取り，プログラム実行に必要な環境の初期設定など本システム全体の制御を行う。

実行制御部は，利用者プログラムの実行に伴う，文ごとの表示タイミングの把握，各種デバッグ機能の制御などを行う。

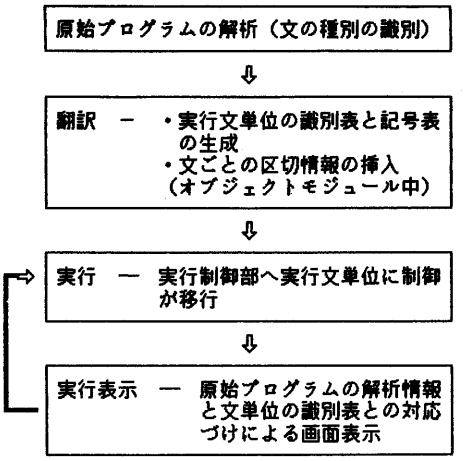
画面制御部は，フルスクリーン画面での入出力全体の制御を行なう。

本システムの構成の概略を，図－７に示す。

文ごとの実行表示は，図－８の概念図に示す方式により，実現している。



図－７　システム構成



図－８　実行表示の方式

4.　利用効果

本システムは，ＴＳＳの下でＦＯＲＴＲＡＮプログラムのテスト・デバッグを支援する会話型システムであり，以下のような利用効果が考えられる。

（1）　デバッグ

- プログラムの論理的な動きの検証
  → 低速／ステップモードによる実行表示
- 特定のデータ値に着目したデバッグ
  → データモニタ
- エラー発生時（一時的に静止モードとなる）における値の参照・変更あるいは制御の移行による実行の継続
  → スナップ文

（2）　テスト

- 実行／未実行パスの検証
  → 実行回数表示
- データ変更などを利用した再実行
  → スナップ文，再実行指示（ＲＥＲＵＮサブコマンドなど）

（3）　ＦＯＲＴＲＡＮ教育

プログラム実行の視覚表示という本システムの特長を活かし，プログラムの動き，各文の機能説明など（静止モード時）を行いながら，どのような場合にどのような原因でエラーが発生するかなどをディスプレイを見ながら教育する。

5.　おわりに

本システムは，テスト・デバッグ対象のプログラムを原始プログラム形式で入力させ，ロードモジュールを実行する方式である。

プログラムのデバッグを主目的として，本システムを使用する場合，次のような点の考慮が必要となってくる。すなわち，原始プログラムを一部変更し，即座に変更した部分を実行させる。この場合，実行表示画面で，原始プログラムの変更が行なえることが望ましい。

この問題を解決するには，原始プログラムの編集機能（エディタ）と逐次実行機能（インタプリタ）を兼備えることが必要となる。

更に，巨大プログラムのデバッグをいかに効率的に進められるかという点も大きな課題である。

謝辞

最後に，本システムの開発にあたり，貴重な助言をいただいた日本原子力研究所　計算室長（代）浅井清殿に改めて感謝します。