SPOT-6：高信頼性ソフトウェア開発のための言語システム

岩元莞二　紫合治　藤林信也
（日本電気（株）中央研究所）

1．はじめに

SPOT-6は、プログラミング方法論の言語化、表現の高水準化、及びプログラミング標準の言語化を通して、中大形ソフトウェアの信頼性向上と開発保守の効率化を狙って開発されたプログラミング言語システムである。SPOT-6は次の様な特長をもつ。

- 抽象化、情報の隠蔽、局所化のサポート
  一つの対象的オブジェクトに対する複数の抽象的オペレーション、あるいは互に関連する幾つかのオペレーションを一つのGROUPモジュールとして定義する機能をもつ。
- モジュール仕様の記述
  モジュール構造、パラメータと外部変数の使い方、モジュール内データの宣言等が明白に記述でき、文書性を高めるとともに広域なインタフェースの正当性のチェックを可能にしている。
- 文書性の高いプログラム記述
  簡単なユーザ定義のデータタイプ、式マクロの導入による高水準化と、goto less programmingのための制御構造により理解性の高い表現ができる。
- 論理構造と物理構造の分離
  抽象化の原則を現実のソフトウェア開発に適用するときの大きな問題であるオペレーション コールのオーバヘッドを解決するため、論理的意味を保ったままインラインコードに展開するサポート機能をもつ。
- 自動インデンテーション機能
  論理構造を視覚化したリストを提供するために、字下りを自動的に制御するリフォーマをもつ。

SPOT-6はインプリメンテーションの容易さのためPL/Iコンパイラのプリプロセッサとして、NEAC ACOS-6上で実現した。そのためPL/Iの記法との融和性を考慮した記法となっている。以下、SPOT-6の言語上の特長を中心に、SPOT-6の概要を述べる。

2．モジュール構造

SPOT-6プログラムは、従来の外部手続きの機能をもつprocedureと、データやリソースの抽象化のための機能をもつgroupから構成される。これらはコンパイル単位となる外部モジュールであり、それぞれ仕様節（spec_section）を含む。groupは複数のoperationをまとめたものであり、各operationはそのgroupによって表わされる一つの抽象データに対するアクセス手段を提供する。内部手続きに対応する

モジュールとしてroutineがある。operationとroutineは、内部データの宣言をまとめたデータ節(data_section)を含むことができる。実行文列はプログラム節(prog_section)にまとめられ、データ等の宣言(仕様節かデータ節に書かれる)と分離している。これらのモジュールの構文の概要は次の通りである。

```
procedure ::= p_head spec_section [routine ...] prog_section p_end
group     ::= g_head spec_section [routine ...] operation ... g_end
operation ::= o_head [data_section] [routine ...] prog_section o_end
routine   ::= r_head [data_section] [routine ...] prog_section r_end
```

ここでp_head, p_end等はそれぞれモジュールのヘッダ文、エンド文を示す。

3. 抽象化や情報の隠蔽のためのモジュール化機構

複雑な問題が与えられた場合、まずその問題領域の概念に最も適した抽象的な機能やデータを選び出し、それらを使って問題を解き、次にそれらの抽象的オブジェクトを具体化するというstepwise refinement[1]方式が有効である。この様な抽象を導入する場合、抽象的な機械やデータを使う側と、それを具体化する側を明確に分離し、使用者には使うために必要な情報(抽象オブジェクトの意味、アクセス方法)のみを与え、実現に関する情報は不要な詳細として隠す(情報隠蔽の原則[2])ことが重要となる。

従来のプログラミング言語に用意されている手続きやサブルーチンは機能の抽象化には有効であるが、データやリソースの抽象化には不充分である。データやリソースの抽象化では、実現のためのデータ構造とそれに対する複数の抽象的オペレーション(例えば、fileのopen, get, put, close; tableのclear, add, searchなど)の定義が必要となるが、オペレーションを従来の外部手続き等で実現すると、オペレーション間で共通に使われるデータが手続き間インタフェースデータとしてシステムに広く分散し、隠蔽の原則や局所化の原則に反した理解しにくい複雑な構造となる。

```
G1 : GROUP ;
   SPEC ;
      ----
      PARM P1--, P2-- ;  ------①
      OPER
         OP1( P1 IN, P2 OUT ),
         OP2(--) RETURNS(--),   }②
         --- ;
      GROUPVAR --- ;     ------③
      ----
   ENDSPEC ;
   OP1 : OPER( P1, P2 ) ;
         -------
   ENDOPER OP1 ;
   OP2 : OPER(--) RETURNS(--) ;   }④
         -------
   ENDOPER OP2 ;
  ----
ENDGROUP G1 ;
```

Fig.1 Group definition

```
P : PROC ;
  SPEC ;
    ---
    SUBGROUP
      G1(
          OP1(---)
          OP2(--) RETURNS(--) ) ;   }①
    ----
  ENDSPEC ;
  PROG ;
    ---
    G1.OP1(X,Y) ;  ------②
    ---
  ENDPROG ;
ENDPROC P ;
```

Fig.2 Usage of group

Fig. 1 に group 記述例の枠組を示す。仕様節（SPEC; と ENDSPEC; で囲まれた部分）には、この group G1 が表わす抽象データの使用法の記述（パラメータ(④)やオペレーション(②)の宣言）やオペレーション間共通な内部データの宣言（group 変数宣言(③)）が含まれる。オペレーションの実現は⑤に書かれる。group を使う側にとっては、このうち使用法の記述のみが必要な情報である。

group を使ったプログラムの記述例を Fig. 2 に示す。group を参照する場合、そのモジュールの仕様節の subgroup 宣言文(①)で、group名とそこで参照するオペレーション名を宣言し、実行文中では group 名を修飾した形のオペレーションの呼び出しを行う(②)。但し、オペレーション名がそこでは一意的な名前なら group 名による修飾を省略できる。

4. モジュール仕様の記述

外部モジュールの仕様は内部データの宣言とともに仕様節に書かれる。文書性向上のため、同じ性質をもつ名前を類別して宣言し、さらにモジュール仕様書としてどのプロジェクトでも記述すべき内容を鍵語付き注釈文で記述できるようにした。宣言文の見出しになる鍵語として下記のものがある。

FUNCTION, AUTHOR, DATE, REMOTEPROCEDURE, TYPE, EMACRO, SUBGROUP, SUBPROCEDURE, EXTERNALVAR, INTERNALVAR, PARAMETER, GROUPVAR, CONSTANT, BUILTIN, CONDITION, EXTERNALFILE, INTERNALFILE, DEFAULT

これらには省略形が許されている。これらの鍵語は SPOT-6 プロセッサに対する指示を与えているが、良いプログラミング標準をも提供している。以下、特徴的な機能を説明する。

- 鍵語付き注釈文
  モジュールの機能概要、作成者、作成・修正日は、それぞれ、FUNCTION、AUTHOR、DATE で始まりセミコロンで終る自由形式で表現できる。これによりソース プログラムからこの様な項目だけを自動的に抽出することが可能である。
- インタフェースの記述
  外部モジュールとの間でどのようにデータが受け渡されるかを明記できる。これらのインタフェースは、下記の対応モジュールとの間でパラメータと外部変数の受け渡し方で記述する。
  - 当該モジュールを呼び出すモジュール
  - 当該モジュールが呼び出す下位モジュール
  - 呼び出しによる上下位の関連はないが外部変数を通して関連する横のモジュール

データの受け渡し方を示すために属性の一つとしてアクセス属性を導入した。上位モジュールとの関係は parameter 宣言文、external 宣言文の中でアクセス属性として宣言する。下位モジュールとの関係は subprocedure 宣言文や subgroup 宣言文の中で宣言する。パラメータの使われ方についてはパラメータ属性の一部として

アクセス属性を宣言する．外部変数の使われ方に関しては、パラメータ属性宣言に続く下記の形式の *extref* オプションを宣言する．

extref-option ::= EXTERNALREF (*variable access-attribute* [, *variable access-attribute*]...)

横のモジュールとの関係を記述するために *remote-procedure* 宣言文がある．この宣言文の本体の形式はパラメータ属性宣言がない点を除いて *subprocedure* 宣言文と同じである．但し、不必要な複雑さを避けるため手続き名の代りに任意の手続きという意味の ANY を記述できる．

アクセス属性は呼び出される側の立場から定義され、次の意味をもつ．

IN ：値を参照するだけで変更しない
OUT ：入口では値が未定義で内部で値をセットする
INOUT ：入口で値が定義され内部で値を変更することがある
INX ：入口では値が定義されているが出口では未定義
PASS ：参照、変更をせず下位モジュールへの受け渡しだけを行う

PASS はその他と組合せて使用できる．上位モジュールとの関係では一般にこの様な使い方をするという宣言であり、その他との関係では、このモジュールの文脈ではその様な働きをしているとみなすことを宣言する．Fig. 4 は Fig. 3 におけるモジュール構造での手続き P におけるインタフェースの記述である．

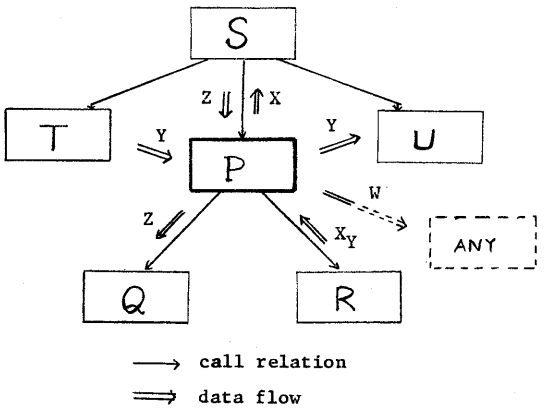


Fig. 3 Illustration of intermodule interface

```
P : PROC ;
  SPEC ;
    ---
    EXTVAR
      W BIN,
      X BIN OUT,
      Y CHAR(10),
      Z BIT(6) IN PASS ;
    SUBPROC
       Q(--)
         EXTREF( Z IN ),
       R(--)
         EXTREF( X OUT, Y OUT ) ;
    REMPROC
       T EXTREF( Y OUT ),
       U EXTREF( Y IN ),
       ANY EXTREF( W IN ) ;
       ---
ENDSPEC ;
---
ENDPROC P ;
```

Fig.4 Interface description for the module P

5. モジュール内記述

プログラム節には PL/I の実行文が全て許されるが、SPOT-6では group のオペレーションが使用できるほかに、ユーザ定義のデータタイプと式マクロによる表現の高水準化、制御構造の標準化のための goto-less 制御文が用意されている．

5.1 データタイプと式マクロ

マクロ機能はある記号列を他の記号列に置き換える便宜として多くのプログラミングシステムでサポートされている。しかし、機構が簡便で強力なことから、無規律に利用するとプログラムを理解する際に結局はマクロ展開した文書に頼ることになる。このためSPOT-6では、マクロ機能を言語要素として理解できるよう次の機構を導入した。

(1) ユーザ定義のデータタイプ

問題領域の概念に近い言葉をデータで扱う道具として、ユーザ定義のデータ属性を許す。機能はPascal[3]のスカラタイプと同程度である。タイプの定義は仕様節で例えば

```
TYPE    COLOR = ( WHITE, BLUE, YELLOW, RED );
INTVAR CAR_BODY_C   COLOR ;
```

で行い、プログラム節の中では、CAR_BODY_Cに値WHITE, BLUE, YELLOW, REDを代入したり、値をそれらと比較することができる。

(2) 式マクロ

パラメータ付加可能なマクロで、文書性の低い式や参照を意味のある名標で表現するためのものである。構文上は式の許される位置で使用できる。宣言は仕様節／データ節で例えば下記の様に行う。

```
EMACRO    NEPHEW(PERSON, I) =
          PERSON.SON(I) →PERSON.
          NAME ;
```

5.2 goto-less 制御構造

goto-less programmingを支援するため、PL/Iの制御文に加えて下記の文を導入した。

(1) DO FOREVER文とUNTIL文

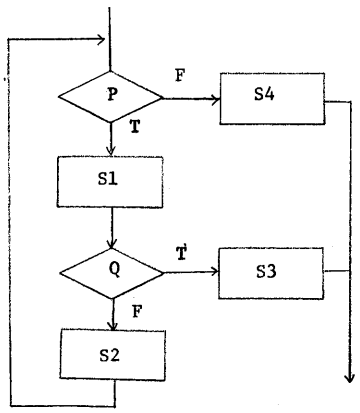
これにより無限ループとループの出口における脱出判定ができる。

(2) EXIT文とEPILOGUEオプション

ループの途中からの脱出を指示するEXIT文と、脱出後の後処理を指示するEPILOGUEオプションが使える。Fig.5(a)はFig.5(b)の構造を表現している。

```
DO WHILE(P);
   S1;
   EXIT [ABT] WHEN(Q);
   S2;
END EPILOGUE(
   [ABT]: S3;
   ENDED: S4; );
```

(a) Example program



(b) Flowchart for (a)

Fig. 5 Illustration of EXIT statement and EPILOGUE option

(3) CASE文

多岐選択構造として CASE構造文が使える．選択条件の種類により3つの変形がある（Fig.6）．RANGE指定は効率の良いオブジェクト（FORTRANの計算型GOTO文的な実現）を指示するために使われる．

```
CASE e   RANGE[ lo : up ]
  [ C1, C2 ] : --- ;
  [ C3 ] :     --- ;
  ----
  ELSECASE --- ;
ENDCASE ;
```

eはスカラ式、c_iとeは比較可能な定数．RANGE指示はオプションで、lo、up、c_iが10進整定数かスカラタイプ値の時に指定できる．

```
CASE ;
  [e1, e2 ] : --- ;
  [e3] : ---      ;
  ----
  ELSECASE --- ;
ENDCASE ;
```

e_iは論理値．

Fig.6 Case structure

6. モジュール展開機能

SPOT-6 プログラムでは、従来外部変数を直接参照していた部分がgroupのオペレーション呼び出しとなることが多く、パラメータの受け渡し等の呼び出しオーバヘッドを無視できない場合が生じる．プログラムの作成のし易さや理解性の高さから、プログラムの作成時には group や procedure で論理を表現し、プログラムの正しさの確認後必要に応じてモジュールを展開する方式をとる．この展開は論理的意味は同じた procedure や groupのoperation と同じ意味を保存するように行う．展開処理はソースプログラムのレベルで行うので展開後さらに言語プロセッサの最適化処理により効率の改善が行える．また、この展開機能は下記の点で通常のマクロ展開機能と異なる．

- operation の展開では group 変数をシステム全体で一意的な名前の外部変数に書き変え、呼び出している側の宣言も書き変える．
- 埋め込まれた本体の RETURN文を GOTO文に修正する．これに伴う一意的なラベル名の生成も必要である．
- 式を返す RETURN 文はその呼び出しの位置によって（例えば、代入文の右辺か DO WHILE文の条件式か）処理方法を変える．
- IF文の THEN節 や ELSE節なら DO; と END; で本体を囲む等．

モジュール展開機能はこれらの処理を行うもので、展開処理がなされたプログラムは、一般のプログラマが直接見ることの出来ない一時ファイルに入り、言語プロセッサによってオブジェクトに変換される．ユーザはもとのモジュールの記述をもとにプログラムの保守管理をしていく．

7. 自動インデンテーション機能

プログラムの構造に従って字下りや改行を行うリフォーマという自動インデンテーションの機能をもつ。このインデンテーションは仕様節とプログラム節の両方にわたって行い、詳細フローチャートなしにプログラムの構造が解るようになっている。リフォーマには次の2種類がある。

- S-リフォーマ　ソースファイルそのものをINCLUDEマクロの展開なしに整形する。結果は新しいソースマスターファイルに入る。
- I-リフォーマ　ファイルはそのままでINCLUDEマクロ展開後のソースプログラムイメージを整形し印字出力する。この場合はプログラムのネスト構造の対応を分り易くするため、対応するレベルの始めと終りを縦線で結ぶ。

付録のプログラムリストはI-リフォーマの出力である。出力媒体に合せて出力イメージの行の最大長と字下りの単位長をリフォーマに対するパラメータとして指定できる。改行、改頁の指示やINCLUDEマクロをリスト上で展開させないなどの指示がソースプログラム中に書け、体裁の良いプログラムドキュメントを得ることができる。

8. システム構成

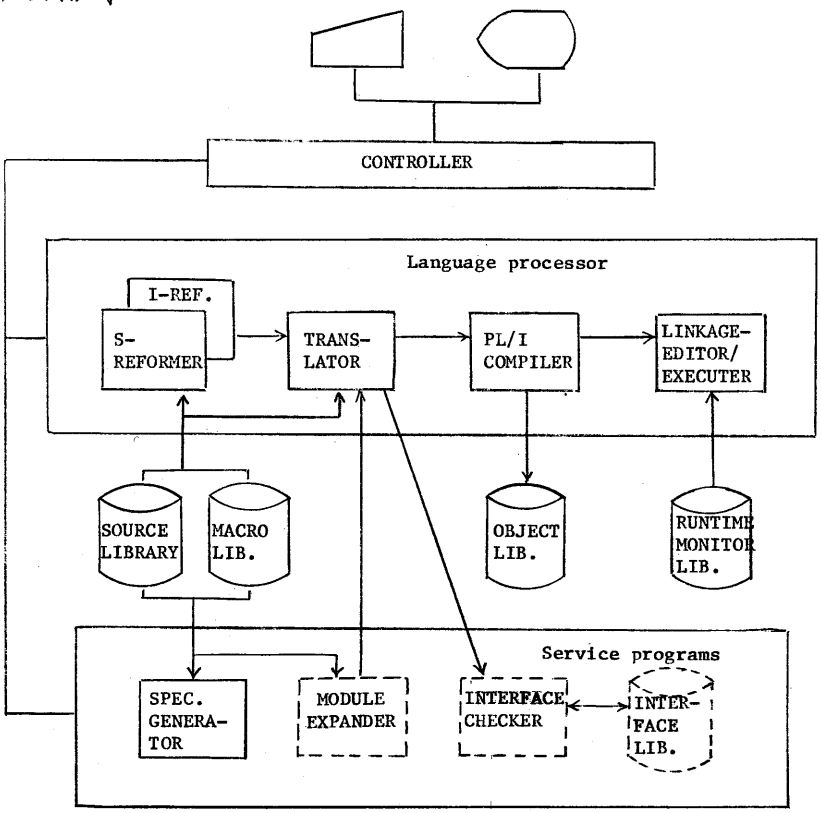


Fig. 7 SPOT-6 processor organization

SPOT-6言語で書かれたソースプログラムは（必要ならリフォーマをへて）トランスレータによってPL/Iプログラムに変換され、連続したアクティビティとしてPL/Iコンパイラを起動することによりオブジェクトプログラムに変換される（Fig. 7）。I-リフォーマとS-リフォーマは一部の出力関係を除いて、同じモジュールで構成されている。プロセッサの使用にあたっては国の機能を選択できるようになっている。プログラムの診断はSPOT-6で導入された構文以外はPL/Iコンパイラにまかせているので、コンパイラ出力のエラーメッセージとSPOT-6ソースプログラムとの対応はライン番号でつけられる様にし、保守はSPOT-6ソースプログラムで行えるようにしてある。実行モニタライブラリは、外部モジュールの呼び出し回数、CPU時間（絶対値と百分率）を測定するライブラリである。モジュール仕様生成はプログラムから仕様部分を抽出編集して内部仕様書を自動生成する機能である。点線のボックスは現在開発中のものでその他は稼動中である。

SPOT-6プロセッサはSPOT-6で書かれている。トランスレータ、リフォーマ、コントローラの各サブシステム間のインタフェースはパラメータとgroupでとられている。Fig. 8はプロセッサとシステムファイルの関連の一部を示したものである。図でSS、RS、RIはそれぞれS-リフォーマ、I-リフォーマ、トランスレータの入力となる抽象ファイルでありgroupで実現されている。その他の2文字のファイルコードで示されるファイルはSPOT-6プロセッサで用いられるシステムファイルである。.Lはユーザ指定のマクロライブラリである。ソースプログラムの入力ファイルとして通常は*Sファイルを用いるが、FILEDITジョブ中でプロセッサを起動するときは*Gを使用する。プロセッサはNEAC ACOS-6オペレーティングシステムの環境に適合して構成されており、ACOS-6上の標準ライブラリ管理、テキストエディタ（TSS）、リンケージエディタ等がそのまま使える。

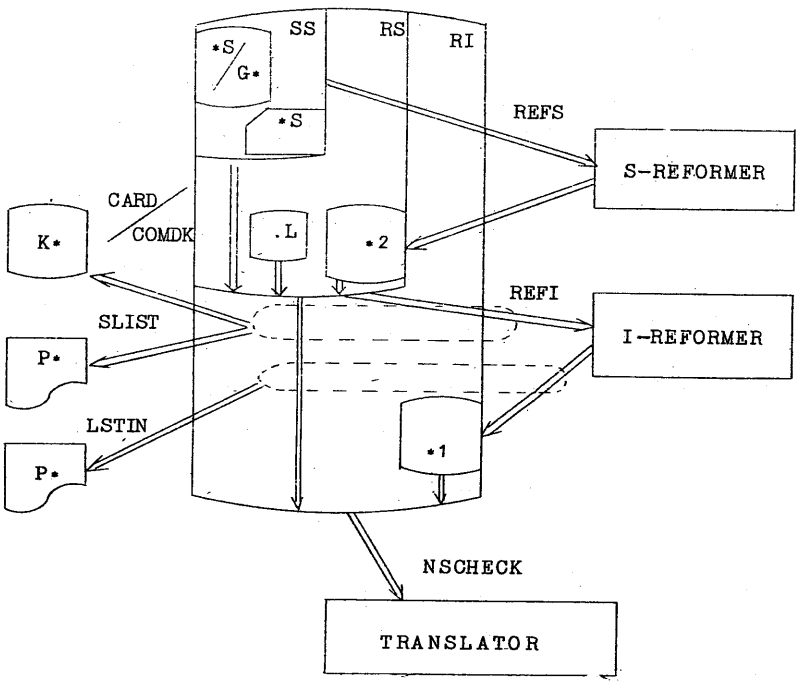


Fig. 8 File structure in SPOT-6 processor

9. むすび

以上、SPOT-6の主な特徴を述べてきた。現在、SPOT-6はSPOT-6プロセッサ自身の他は2，3のプログラム開発に用いられている。この使用経験を通してSPOT-6言語の有効性が確認された。抽象化のための機構groupは、CLU[4]のclusterやAlphard[5]のformのような抽象データタイプではなく、一つの抽象オブジェクトを表現するものであるが、タイプに比べインプリメントが容易であるという利点をもち、現実のソフトウェア開発では一つの抽象タイプに対して一つの抽象オブジェクトで間に合う場合が多いので有効に利用できる。SPOT-6プロセッサの開発では、トランスレータ、リフォーマ、コントローラ等のサブシステム間のインタフェースはパラメータとgroupによって取られ、groupの仕様節がインタフェース仕様書の役割を果した。groupの使用によって抽象のレベルが与えられ、いわゆる単体テストの段階では、ダミーのgroupを用いて上位のモジュールをテストできgroupを使用しているモジュールとgroupの開発を独立して行えた。groupの使用とユーザ定義のスカラ値のデータタイプの導入により、通常設計言語と称される疑似言語で表現される設計がそのままSPOTプログラムとして書け、記述水準が大きく向上した。記述水準を高めるためにはこのようなデータ記述の高水準化が必須であろう。groupのもう一つの効果として外部変数の極端な減少があげられる。これによりシステムのインタフェースが単純化され保守し易いシステムの開発が支援できる。

この論文では主として言語機能を述べたが、SPOT-6は開発を支援するために、インタフェースチェッカー・アナライザを始めとしてモジュール仕様生成機能、モジュール展開機能、実行モニタライブラリのユーティリティを含んでいる。これらの詳細については別の機会に報告したいが、このようなユーティリティの中にはSPOT-6の言語機能に負う所が大きいものがあることだけを述べておく。

謝辞 SPOT-6プロセッサのインプリメンテーションを手伝っていただいた当社コンピュータシステム研究部の伊東君、福田さん、長谷川さんに謝意を表する。また、同研究部藤野部長には終始励ましと有益な助言をいただいた。ここに深く感謝する次第である。

参考文献

[1] N. Wirth: "Program development by stepwise refinement", CACM, Vol.14, No.4(1971).

[2] D.L. Parnas: "On the criteria to be used in decomposing systems into modules", CACM, Vol.15, No.12 (1972).

[3] N. Wirth: "The programming language Pascal", Acta Informatica, Vol.1, No.1 (1971).

[4] B.H. Liskov and S. Zilles: "Programming with abstract data types", SIGPLAN Notices, Vol.9, No.4 (1974).

[5] W.A. Wulf, R.L. London and M. Shaw: "An introduction to the construction and verification of Alphard programs", IEEE Trans. Software Eng., Vol.SE-2, No.4 (1976).

[付録] SPOT-6 ソースプログラムリストの例（リフォーム結果）

```
27037 01  03-03-52   13.919     ***  S P O T - 6   P R O C E S S O R  ***  770301 520303MCHC            PAGE      2
ALT#  LINE#      +----------+--  MSCHECK_AND_COPY_SPECIAL_OPTIONS_AND_SETUP  --+----------+---------+---------+

   6       1     MSCHECK_AND_COPY_SPECIAL_OPTIONS_AND_SETUP_OPTION_TABLE : PROCEDURE ;
   7       2
   8       3     SPECIFICATION ;
   9       4
  10       5        /*******************   EXTERNAL SPECIFICATION   **********************/
  11       6        FUNCTION
  12       7              PRINT STAR CARD IMAGES ON OPTION LISTING FILE.
  13       8              SCAN STAR CARDS IN SPOT_SRC_FILE AND CHECK SPECIAL OPTIONS,
  14       9           THEN SETUP OPTION TABLE. ALSO COPY STAR CARDS IMAGE INTO REF_SRC_
  15      10           FILE IF REFS IS SPECIFIED, AND PUT MODIFIED STAR CARD IMAGE INTO
  16      11           PLI_SRC_FILE IF COMP IS SPECIFIED.
  17      12           POST CONDITION:
  18      13              SPOT_SRC_FILE IS POSITIONED AT THE BEGINING OF SPOT-6 PROGRAM
  19      14              TEXT ( NON STAR CARD ) IMAGES.
  20      15           SPECIAL REMARKS:
  21      16              THE PURPOSE TO COPY STAR CARD IMAGES INTO REF_SRC_FILE IS TO
  22      17              MERGE NEW REFORMED SPOT SOURCE PROGRAM TEXT WITH THEM AND
  23      18              PUT THEM INTO K* FILE AT LATER I-REFORMER OR TRANSLATOR READ-
  24      19              ING TIME.
  25      20                                              ***   END OF FUNCTION   *** ;
  26      21        AUTHOR  S. FUJIBAYASHI ;
  27      22        DATE WRITTEN  76/11/08 ;
  28x      *

27037 01  03-03-52   13.919     ***  S P O T - 6   P R O C E S S O R  ***  770301 520303MCHC            PAGE      3
ALT#  LINE#      +----------+--  MSCHECK_AND_COPY_SPECIAL_OPTIONS_AND_SETUP  --+----------+---------+---------+

  29      23     /*******************   INTERNAL SPECIFICATION   **********************/
  30      24
  31      25        SUBGROUP
  32      26           US$SPOT_SRC_FILE_S5
  33      27              (READ_CARD (IMAGE   CHAR(80) ALIGNED OUT) ,
  34      28              READ_ALTNO (ALTNO  FIXED BIN OUT)          ,
  35      29              SEND_BACK_CARD                             ),
  36      30           US$PLI_SRC_FILE_PS
  37      31              (WRITE_CARD (IMAGE  CHAR(80) ALIGNED IN, ALTNO  FIXED BIN IN) ),
  38      32           US$REF_SRC_FILE_RS
  39      33              (WRITE_CARD (IMAGE  CHAR(80) ALIGNED IN) ),
  40      34           WSOPTION_TABLE_OT
  41      35              (IS_COMPILE                                RETURNS(BIT(1)),
  42      36              IS_REFS                                    RETURNS(BIT(1)),
  43      37              SCAN_STAR_CARD (IMAGE   CHAR(80) ALIGNED INOUT, ANUM   FIXED BIN IN, PLIOPT  BIT(1)
  44      38                 ALIGNED OUT)                            ,
  45      39              CHECK_STAR_CARDS_USAGE (IMAGE   CHAR(80) ALIGNED IN, ALTNO   FIXED BIN IN) ) ;
  46      40        SUBGROUP
  47      41           MSINPUT_LINE_IMAGE_LIST_IL
  48      42              (PUT_STAR_CARD (IMAGE   CHAR(80) ALIGNED IN, ALTNO   FIXED BIN IN) ) ;
  49      43        EMACRO
  50      44           TRUE = '1'B,
  51      45           FALSE = '0'B,
  52      46           STAR_CARD_IS_FOUND = SUBSTR(IMAGE,1,1)='*',
  53      47           STAR_CARD_TYPE = SUBSTR(IMAGE,2,4) ;
  54      48        INTVAR
  55      49           IMAGE                                      CHAR(80) ALIGNED,
  56      50           ANUM                                       FIXED BIN,
  57      51           REFORMED_SOURCE_IS_REQUESTED               BIT(1) ALIGNED,
  58      52           OBJECT_PROGRAM_IS_REQUESTED                BIT(1) ALIGNED,
  59      53           PLI_OPTION                                 BIT(1) ALIGNED,
  60      54           CARD_IS_NOT_NOTE                           BIT(1) ALIGNED INIT(TRUE) ;
  61      55
  62      56     ENDSPECIFICATION ;
  63x      *

27037 01  03-03-52   13.919     ***  S P O T - 6   P R O C E S S O R  ***  770301 520303MCHC            PAGE      4
ALT#  LINE#      +----------+--  MSCHECK_AND_COPY_SPECIAL_OPTIONS_AND_SETUP  --+----------+---------+---------+

  64      57     PROGRAM ;
  65      58     !
  66      59     !  REFORMED_SOURCE_IS_REQUESTED = IS_REFS() ;
  67      60     !  OBJECT_PROGRAM_IS_REQUESTED = IS_COMPILE() ;
  68      61     !  LOOP : DO FOREVER ;
  69      62     !  !  READ_CARD (IMAGE) ;
  70      63     !  !  READ_ALTNO (ANUM) ;
  71      64     !  !  IF STAR_CARD_IS_FOUND THEN
  72      65     !  !  !  CASE STAR_CARD_TYPE ;
  73      66     !  !  !  !  ( 'NOTE' ) : CARD_IS_NOT_NOTE = FALSE ;
  74      67     !  !  !  !  ( 'ENDN' ) : CARD_IS_NOT_NOTE = TRUE ;
  75      68     !  !  !  ENDCASE ;
  76      69     !  !  ELSE EXIT LOOP WHEN (CARD_IS_NOT_NOTE) ;
  77      70     !  !  PUT_STAR_CARD (IMAGE, ANUM) ;
  78      71     !  !  IF REFORMED_SOURCE_IS_REQUESTED THEN USREF_SRC_FILE_RS.WRITE_CARD (IMAGE) ;
  79      72     !  !  IF CARD_IS_NOT_NOTE THEN
  80      73     !  !  !  DO ;
  81      74     !  !  !  !  WSOPTION_TABLE_OT.SCAN_STAR_CARD (IMAGE,ANUM,PLI_OPTION) ;
  82      75     !  !  /* POST CONDITION:  IMAGE MAY BE MODIFIED TO DELETE */
  83      76     !  !  /* INTRINSIC INFORMATION.                           */
  84      77     !  !  !  !  IF OBJECT_PROGRAM_IS_REQUESTED & PLI_OPTION THEN
  85      78     !  !  !  !  !  US$PLI_SRC_FILE_PS.WRITE_CARD (IMAGE,ANUM) ;
  86      79     !  !  !  END ;
  87      80     !  END LOOP ;
  88      81     !  SEND_BACK_CARD ;
  89      82     !  CHECK_STAR_CARDS_USAGE (IMAGE, ANUM) ;
  90      83     !     /* THE 1ST CARD IMAGE OF NON STAR CARDS WILL BE USED FOR */
  91      84     !     /* DEFAULT LABEL ID AND SUBTITLE STRING.                 */
  92      85     !
  93      86     ENDPROGRAM ;
  94      87     ENDPROCEDURE MSCHECK_AND_COPY_SPECIAL_OPTIONS_AND_SETUP_OPTION_TABLE ;
```