JAERI-Tech
97-017

プログラム並列化支援解析ツールｋｐｘ

1997年3月

松山雄次・折居茂夫・大田敏郎・久米悦雄・相川裕史

日本原子力研究所
Japan Atomic Energy Research Institute

本レポートは，日本原子力研究所が不定期に公刊している研究報告書です。
入手の問合わせは，日本原子力研究所研究情報部研究情報課（〒319-11 茨城県那珂郡東海村）あて，お申し越しください。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。

This report is issued irregularly.
Inquiries about availability of the reports should be addressed to Research Information Division, Department of Intellectual Resources, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken 319-11, Japan.

編集兼発行　日本原子力研究所
印　　刷　日立高速印刷株式会社

JAERI-Tech 97-017

プログラム並列化支援解析ツールkpx

日本原子力研究所計算科学技術推進センター
松山　雄次・折居　茂夫・大田　敏郎
久米　悦雄・相川　裕史

（1997年2月10日受理）

kpxは，並列処理推進のための共通基盤として開発した，プログラム並列化支援解析ツールである。kpxは，プログラムのどの部分でどの位の実行時間が消費されているかを計測するためのktool，Paragonの並列化オーバヘッドを計測するためのptool，VPPの並列化オーバヘッドを計測するためのxtoolから構成されている。kpxのktoolは，全てのUNIX系コンピュータのFORTRANプログラムを対象としておりParagon，SP2，SR2201，VPP500，VPP300，Monte－4，SX－4，T90において動作確認されている。

日本原子力研究所：〒153　東京都目黒区中目黒2－2－54

JAERI-Tech 97-017

The kpx, a Program Analyzer for Parallelization

Yuji MATSUYAMA, Shigeo ORII, Toshiro OTA, Etsuo KUME and Hiroshi AIKAWA

Center for Promotion of Computational Science and Engineering
Japan Atomic Energy Research Institute
Nakameguro, Meguro-ku, Tokyo-to

(Received February 10, 1997)

The kpx is a program analyzer, developed as a common technological basis for promoting parallel processing. The kpx consists of three tools. The first is ktool, that shows how much execution time is spent in program segments. The second is ptool, that shows parallelization overhead on the Paragon system. The last is xtool, that shows parallelization overhead on the VPP system. The kpx, designed to work for any FORTRAN cord on any UNIX computer, is confirmed to work well after testing on Paragon, SP2, SR2201, VPP500, VPP300, Monte-4, SX-4 and T90.

Keywords: Parallelization, Analyzer, Overhead, Programing Tool

目　　次

Contents

1．はじめに

プログラムを並列化することにより、高速化を実現する試みがなされてから既に数十年の歳月が経つ。特に過去十年間においては、ベクトル型計算機のクロック周期が限界に近づき、シングルCPUでの高速化に限界が見えてきたため、超並列型計算機が新たなブレイク・スルーを実現する、基礎アーキテクチャとして脚光を浴びてきた。今日ではベクトル型計算機は、マルチCPUが常識となっているが、共有主記憶型では複数CPUで主記憶バンクを取り合うバンクコンフリクトが発生しやすくなり並列度にも限界がある。そこで、分散主記憶型のスカラ並列を実現した並列計算機が、ピーク時における価格性能比からも注目されている。ベクトル型計算機、いわゆるスーパーコンピュータでの並列化は、多くの場合コンパイラが自動並列化機能を持ち、ある程度の並列化を自動的に行った機械語を出力してくれるので、使い勝手もよい。しかしながら、超並列型計算機はその半性能を得るにも、アーキテクチャがプログラム・コードを選び、なおかつ多大な努力を強要する。また、線形な加速率を100PE以上の高並列まで維持することは、通信およびロード・バランスのオーバヘッド、もしくは入出力総量の関係から非常に困難である。

ここで、今、ユーザが所持しているコードが、どのアーキテクチャの計算機ではどの位の性能が得られるか、また、そのコードをある計算機にかけた時、ノード数がどの位の時が最も効率的か、などといったことを予測できるツールがあれば、並列化を行うにあたり良い指標となりうる。そこで、そのためのスケーラビリティ解析研究が企画され、ヘテロジニアスな計算機上でも、同一の性能でモデル・パラメータを測定できる共通のツールが必要となり、開発を行う運びとなった。kpxは、スケーラビリティ予測ツールに必要な、データを収集するプログラム部分を、出力インタフェースを取り、独立したプログラム並列化支援解析ツールとしたものである。

kpxは、各サブルーチン、ループ、入出力処理の実行負荷を測定するktoolと、スケーラビリティ予測に不可欠な並列化オーバヘッドを予測するptool（Paragon用）、xtool（VPP用）から構成される。分析対象のプログラムは、現時点の科学技術計算分野で最も広く使用されているFORTRANコードとし、オペレーティング・システムは、現在、超高速並列計算機で標準となっているUNIXとした。kpxのktoolは、Paragon、SP2、SR2201、VPP500、VPP300、Monte-4、SX-4、T90において動作確認を行い、ptoolはParagonで、xtoolはVPP500とVPP300で各々動作確認を行っている。リリース形式としては、tarファイルkpx.tarをユーザが各自ターゲットとする計算機に取り込み、ソースをコンパイルしkpxを構築する方法を想定している。kpxはフリーウェアとして広く世界に解放することとした。

本報告書では、kpxの仕様、設計、および利用法について記述し、最後に、作成した試用版について実施したアンケート結果と、それに基づいて検討した機能追加修正についても報告する。

2．仕様

kpxの仕様は大まかに設定され、詳細設計において個々の機能の実現方法が検討された。プログラム負荷解析の方法は、FORTRANプログラムの文構造を解析して時間計測ルーチンまたは回数計測ルーチンを挿入する形で実現する。また、できる限りのFORTRAN記述規則を受け入れることとした。

kpxはktool、ptool、xtoolから構成されており、以下のような機能を有する。

ktool　ユーザが指定した時間計測ルーチンにより、プログラム、サブルーチン、doループ、および入出力の実行に要した時間の計測を行う。また、実行回数について、プログラム、サブルーチン、doループ、および入出力回数の計測を行う。

ptool　Paragonのnxライブラリにより並列化されたプログラムの、ノード間通信等の並列化オーバヘッドを測定する。

xtool　VPPの並列化ディレクティブにより並列化されたプログラムの、ノード間通信等の並列化オーバヘッドを測定する。

2.1　概要

プログラム並列化支援解析ツールkpxの仕様を以下に示す。

・経過時間、CPU時間をサブルーチン毎、doループ毎、入出力毎に計測する機能を装備する
・実行回数をサブルーチン毎、doループ毎、入出力毎に計測する機能を装備する
・doループの平均ループ回数を出力する機能を装備する
・計測のオーバヘッドを削減する機能を装備する（オリジナル実行時間の1.5倍以内を目標）
・特定のサブルーチン、doループ、入出力を複数指定して計測ができる機能を装備する
・ツールのソース・コードは、並列計算機Paragon、SP2、SR2201、VPP500、VPP300、Monte-4、SX-4、T90に対し、同一のもので対応し、他の計算機に対しても拡張性を有する
・時間測定は、各並列計算機が用意する時間計測ルーチンを使用する
・時間計測ルーチンを挿入したソース・コードは編集可能とする
・計測したデータを分かりやすく出力する機能を装備する
・ツールが生成する変数、配列名を変更可能とする

2.2　時間計測

1）サブルーチン毎

各サブルーチンは主プログラムからの呼び出し階層（レベル）で管理し、各サブルーチンの開始時に計測を開始し、終了時にそのサブルーチンの計測を終了する。得られた所要時間をサ

ブルーチンの累積時間に加算し、１つ手前のレベルの呼び出し元サブルーチンの累積時間からその所要時間を減算する。

２）doループ毎

基本手法は１）のサブルーチン毎の計測方法と同じであるが、その計測カウンタはサブルーチンとは区別する。ループ内にサブルーチンの呼び出しが存在する場合でも、doの計測カウンタにより計測され、サブルーチンの計測とは独立に対応する。このサブルーチンの所要時間はその文の前後計測処理を追加して、その中にあたかも内側のループ（ダミーdoループ）が存在すると仮定して計測する。

３）入出力毎

入出力文の前後にその論理機器番号を対象に、開始・終了時刻（経過、CPU時間とも）を計測し加算する。ただし、入出力リストに関数の引用がある場合は、計測不能のため、その関数の処理時間もその入出力の処理時間に加算する。

2.3　回数計測

１）サブルーチン毎

サブルーチンの実行時にその実行回数として１を加算する。

２）doループ毎

doループの処理の直前にその回数として１を加算する。

３）入出力毎

論理機器番号毎、FORTRANの１文毎にその処理の直前にその回数として１を加算する。

3．設計

kpxはktool、ptool、xtoolの三種類のツールから構成されるが、時間計測ルーチンをソースに挿入して測定する等、共通の設計思想に基づき設計されているため、殆どのルーチンで共通の部分が多い。

3.1 概要

プログラム分析ツールkpxは、時間計測ルーチンを計測元ソースに挿入した、中間の計測用ソースを出力する。これにより特定目的を持つユーザが、この計測用ソースを編集することを可能としている。

時間はFig 3.1に示す各並列計算機で用意されている時間計測ルーチンにより、計測対象箇所を挟み込むことにより計測することとした。ktoolでは、時間計測ルーチンの選択方法により経過時間やCPU時間の計測が可能である。ptool、xtoolでは所定の時間計測ルーチンによる経過時間が報告される。

多重ループの場合や、do端末が一文のif文で記述され同時にそのラベルへのgoto文がある場合、またはマルチ・ステートメント「；」等でプログラムが記述されている場合は、時間計測ルーチンを挿入するため、事前にステップの分解等を行う。

また、プログラム解析データの集計および編集は、計測元ソースのstop文の直前に終了処理サブルーチンのcall文を挿入し、そのサブルーチンのソースを計測用ソースの最後に置くことによって実現している。

kpxにはツールとユーザ・プログラム間でのシンボル名重複を避けるため、ツールのシンボル名の再定義を可能としている。また、同様の目的により入出力に関してはFORTRAN論理機器番号の変更も可能である。さらに、効率的計測作業を実現するため、種々の計測制限も可能とした。他にインクルードの参照パスの指定、実行回数のみの計測指定等を含め、制御データおよびディレクティブによる、kpxの計測制御を実現している。

3.2 ktool

ktoolは、実行回数をサブルーチン毎、doループ毎、入出力毎に計測する機能と、消費時間をサブルーチン毎、doループ毎、入出力毎に計測する機能を装備する。また、計測のオーバヘッドを除去する機能も備えている。さらに、doループの平均ループ回数を出力する機能を有している。ktoolにおいて利用されるファイルおよび処理の流れをFig 3.2に示す。

3.2.1 計測法

1）実行回数

実行回数の測定はサブルーチン毎、doループ毎、入出力毎にカウンタを設定し、そのカウンタが呼び出される毎に１を加算する。

特にdoループの平均ループ回数について計測する機能に関しては処理のオーバヘッド削減を考慮し、doループの直前にそのdoループのループ回数を算出し、そのdoループの実行回数と総ループ回数により平均ループ回数を求める。do while、do until型のdoループの場合や、doループ内からdoループ外に分岐する処理がある場合には、doループ内にループ回数を計測するカウンタを設定する。

２）消費時間（経過時間またはCPU時間）

ktoolは、消費時間を計測する機能に付随した、計測のオーバヘッドを削減する機能を備える。したがって、プログラム、サブルーチン、doループを時間計測ルーチンで挟み込み、さらに時間計測ルーチン自体のオーバヘッドを算出してより正確な計測結果を得られるよう設計されている。具体的計測方法はFig. 3.3に示すとおり、測定前に挿入するサブルーチンと測定後に挿入するサブルーチンとで計測対象を挟み込む。ここで、時刻　tx　を次のように定義すると、

t1：測定対象前に挿入するサブルーチンの開始時刻
t2：測定対象前のサブルーチンでハードウェア・タイマを参照した時刻
t3：測定対象前に挿入するサブルーチンの終了時刻
t4：測定対象後に挿入するサブルーチンの開始時刻
t5：測定対象後のサブルーチンでハードウェア・タイマを参照した時刻
t6：測定対象後に挿入するサブルーチンの終了時刻

測定対象が消費した時間は　t4－t3　であるが、時間計測のためのオーバヘッドがあるため、サブルーチンが返す時刻から計算される値　t5－t2　に対してプログラム終了時に補正を行い、

$$(t5-t2)-\{(t3-t2)+(t5-t4)\} = t4-t3$$

とした。ここで、　(t3－t2)＋(t5－t4)　は計測対象に含まれるオーバヘッドである。また、測定対象が入れ子になっている場合は、　(t2－t1)＋(t6－t5)　は外側にかかるオーバヘッドとして計上されるため、上位ネスト・レベルの計測対象から除去する。

3.2.2　集計法

１）サブルーチンの計測

あるサブルーチンに呼び出しサブルーチンがある場合、そのサブルーチンの消費時間には呼び出されたサブルーチンの消費時間は含めない。ただし、ktoolは特定のサブルーチンを複数指

定して計測ができる機能を装備しているため、呼び出されたサブルーチンが測定対象となっていない場合、その消費時間は呼び出し側サブルーチンの消費時間として出力する。

doループ、入出力等での消費時間は、すべてそのサブルーチンの消費時間として出力する。

2）doループの計測

あるdoループに呼び出しサブルーチンがある場合、そのdoループの消費時間には呼び出されたサブルーチンの消費時間は含めない。ただし、ktoolは特定のサブルーチン、doループ、入出力を複数指定して計測ができる機能を装備しているため、呼び出されたサブルーチンが測定対象となっていない場合、その消費時間は呼び出し側doループの消費時間として出力する。

多重ループでは、そのdoループの消費時間には下位ネスト・レベルのdoループでの消費時間は含めない。ただし、ktoolは特定のdoループを複数指定して計測ができる機能を装備しているため、下位ネスト・レベルのdoループが測定対象となっていない場合、その消費時間は上位ネスト・レベルdoループの消費時間として出力する。

入出力等での消費時間は、すべてそのdoループの消費時間として出力する。

3）入出力の計測

ユーザ・プログラムの入出力コードが実行されると、処理はシステムに渡されるため、入出力時間に関しては単純に消費時間の集計が可能である。

3.2.3 ktool制御データ

以下に、ktool制御データで設定可能なシンボル名を挙げ、その意味とディフォルト値ついて記す。

1）ファイル名、入口名、コモン名等

計測用ソースに組み込まれるシンボル名が、計測元ソースのシンボル名と重複することを防止する目的で指定する。

シンボル名	意味	ディフォルト
ENT_ENT	プログラムの計測開始入口名	ENT_ENT
RET_ENT	プログラムの計測終了入口名	RET_ENT
ENT_DOL	doループの計測開始入口名	ENT_DOL
RET_DOL	doループの計測終了入口名	RET_DOL
CTR_DOL	doループのループ回数計測入口名	CTR_DOL
ENT_IOU	入出力の計測開始入口名	ENT_IOU
RET_IOU	入出力の計測終了入口名	RET_IOU
CPU_MSE	CPU／経過時間の計測開始入口名（要暗黙実数型）	CPU_MSE
CPU_MSR	CPU／経過時間の計測終了入口名（要暗黙実数型）	CPU_MSR
OUT_CTR	計測結果出力入口名	OUT_CTR

CPU_BLK	共通ブロック初期プログラム名	CPU_BLK
CPU_CTR	共通ブロックのインクルード・ファイル名	CPU_CTR
COM_CHR	文字型共通ブロック名	COM_CHR
COM_BIN	数値型共通ブロック名	COM_BIN
DUM_DUM	当該ツールのオーバヘッド計測入口名	DUM_DUM

2）ファイル論理機番等

ファイル情報を指定する。

シンボル名	意味	ディフォルト
INP_SOC	計測元ソース・ファイル名	@@test.fff
OUT_SOC	計測用ソース・ファイル名	@@f22.f
OUT_U66	計測結果ファイル論理機番	66（出力）
OUT_F66	計測結果ファイル名	@@k_tool.f66
OPN_U66	計測結果ファイルのオープン処理要求 Yes／Noで要／否を指定	Yes
OUT_U99	計測結果データ・ファイル論理機番	99（出力）
OUT_F99	計測結果データ・ファイル名	@@k_tool.f99
OPN_U99	計測結果データ・ファイルのオープン処理要求 Yes／Noで要／否を指定	Yes

3）時間計測ルーチンの試行回数

ツール自身のオーバヘッドを削減するために、別途行う、時間計測ルーチンの試行回数。0が指定された場合には、ツール自身のオーバヘッド時間を含んだ計測結果を出力する。

シンボル名	意味	ディフォルト
OHD_CTR	時間計測ルーチンの試行回数	1000

4）計測対象の特定

シンボル名	意味	ディフォルト
GLOBAL_ENT	計測対象プログラムのグローバル指定	Yes

Yes／Noの何れかで指定し、Yesの場合はユーザ・プログラムの総てが一旦その対象となり、ENTRYで除外するプログラムを指定する。Noの場合は、ENTRY指定されたプログラムのみが計測対象となる。

GLOBAL_DOL	計測対象doループのグローバル指定	Yes

計測対象から除外または追加するdoループの指定は、後述するDOSEQ、DOLABELで行う。GLOBAL_ENTと同様にYes／Noを指定して併用する。

GLOBAL_IOU	計測対象入出力処理のグローバル指定	Yes

計測対象から除外または追加する入出力処理の指定は、後述するIOSEQで行う。GLOBAL_ENTと同様にYes／Noを指定して併用する。計測対象の入出力処理文はOPEN、

CLOSE、READ、WRITE、PRINT、PUNCH文とする。

UNSAT_DO　doループの未確定ループ回数の計測指定　Yes

doループ処理前にループ回数が確定していないdoループ、例えば、do while、do untilや、ループ外への飛び出しのあるdoループ（goto文、return文、stop文）には、doループ内にそのループ回数を得る処理を挿入するが、計測処理のオーバヘッドが増加する。Yesが指定されると、未確定ループ回数のdoループの計測をし、Noが指定されると、未確定ループ回数のdoループの計測を抑止する。

ENTRY　プログラム名（入口名）　なし

計測対象から除外または追加するプログラム名を指定する。GLOBAL_ENTで、Yesが指定されている場合は計測除外するプログラム名を、Noが指定されている場合は追加計測するプログラム名を指定する。

DOLABEL　doループのラベルの指定　なし

計測対象から除外または追加するdoループを、ラベルにより指定する。プログラム名とラベルは［，］で区切って一緒に指定する。多重doループでラベルが重複されて利用された場合は、最内ループが指定されたとみなされる。最内ループ以外を指定したい時は、DOSEQデータにより指定する。DOLABELはGLOBAL_DOLと併用して利用する。

DOSEQ　doループ出現順番の指定　なし

計測対象から除外または追加するdoループを、プログラム内におけるループの出現順番により指定する。プログラム名とループ出現順番は［，］で区切って一緒に指定する。ラベルなしdoループの場合は、DOSEQでのみ指定が可能である。DOSEQはGLOBAL_DOLと併用して利用する。

IOSEQ　入出力処理出現順番の指定　なし

計測対象から除外または追加する入出力処理を、プログラム内における入出力処理の出現順番を指定して特定する。プログラム名と入出力処理出現順番は［，］で区切って一緒に指定する。IOSEQはGLOBAL_IOUと併用して利用する。

MAX_ENT　プログラムの計測対象最大値の指定　0

リスタート処理（実行回数計測結果を利用した実行時間計測）時に、測定対象のプログラムをその実行回数で制限できる。リスタート時にMAX_ENTが指定される（0以外）と、指定回数以上に実行されているプログラムは測定対象から除外される。また、リスタート処理を行うかどうかは、編集前の計測結果ファイルOUT_F99の有無により決定される。MAX_ENTが0またはOUT_F99が存在しない場合は実行回数計測、そうでない場合はリスタート処理を行う。

MAX_IOU　入出力処理の計測対象最大値の指定　0

MAX_IOUはMAX_ENTと同様な働きをする。リスタート処理時に、実行回数がMAX_IOUの値以上の入出力処理を計測対象から除外する。

MAX_DOL　　doループ処理の計測対象最大値の指定　　0

MAX_DOLはMAX_ENTと同様な働きをする。リスタート処理時に、実行回数がMAX_DOLの値以上のdoループ処理を計測対象から除外する。

MAX_MRT　　doループ処理の平均ループ回数の計測対象最大値の指定　0

MAX_MRTはMAX_ENT と同様な働きをする。リスタート処理時に、平均ループ回数がMAX_MRTの値以上のdoループ処理を計測対象から除外する。

5）その他の制御データ

シンボル名	意味	ディフォルト
MYSYSTEM	制御用ファイル名	mysystem.dat

制御用ファイル名を与える。MYSYSTEMが指定されない時は、カレント・ディレクトリにmysystem.datが存在するとして処理する。

TABCODE　　タブ・コードの内部コードの指定　　9

タブ・コードの内部コードを10進数で与える。TABCODEの値は、CHAR(n)のnとして利用される。TABCODEの指定は後述するTABNUMの前に行わければならない。

TABNUM　　タブ・コードによるカラム数の指定　　8

タブ・コードに対応するカラム数を与える。制御用データ入力時、TABCODE、TABNUMが指定されるまでは各々９と１が採用される。最終的にTABNUMがデータとして指定されない場合は、入力終了時に８の標準値が指定される。TABCODEの指定はTABNUMの前に行わなければならない。

JST_CTR　　計測の種別指定　　No

計測の種別は実行回数と時間計測、および実行回数計測のみの２通りがある。計測の種別をJST_CTRで指定する。Noの場合は両方の計測を、Yesの場合は実行回数のみを計測する。

INC_PATH　　インクルードの参照パスの指定　　なし

計測元ソースで、インクルード文の参照パスを特定したい時INC_PATHで指定する。INC_PATHは複数指定可能で、インクルード文が存在した場合、INC_PATHを出現順に検索して、必要なインクルード・ファイルを参照する。インクルード文が絶対パス指定の場合は、INC_PATHを参照しない。また、「./」でインクルード指定された場合は、無条件にカレント・ディレクトリを参照する。同じ名前のインクルード・ファイルが複数存在する時は、複数のINC_PATHで指定するが、指定順序に注意が必要である。インクルード文は次の形式のみ許される。

a.　include 'abc.inc'
b.　include "abc.inc"

c. *include abc

6）制御データの記述例

```
CPU_CTR = cpuinc.inc                    --> インクルード・ファイル名の変更
CPU_BLK = cpublk                        --> 共通ブロック初期化プログラム名の変更
COM_CHR = comxxx                        --> 共通ブロック名の変更
COM_BIN = comyyy                        --> 共通ブロック名の変更
! change file
out_u66 = 06                            --> 情報を標準出力と一緒に出力する
opn_u66 = 0          ! not open         --> open／closeを抑止
out_u99 = 12                            --> 論理機番の一致を防止
out_f99 = @@export.dat
! Global data
global_ent = yes       ! specified erase data
! Entry name
entry = e_sub                           --> エントリe_subを計測対象から除外
! Do loop (Label/Seq)
dolabel = d_sub , 100                   --> エントリd_subのラベル100doループを計測除外
doseq   = d_sub , 2                     --> エントリd_subの２番目のdoループを計測除外
.   ! end of data -------               --> 以下のデータは無視される。
```

3.2.4 ktoolディレクティブ

計測対象は一般に制御データで指定するが、計測元ソースに挿入するディレクティブでの指定も可能である。ディレクトリの仕様と種別を以下に示す。ディレクティブは制御データのGLOBAL_xxxと併用して、計測の除外対象か追加対象かを指定する。

1）ディレクティブの仕様

第１カラムから「COSTTIME@パラメータ」で指定し、第１カラム以外のブランクは無視する。大文字、小文字の区別はない。以下の例はいずれも同じ意味を持つ。

a. Cost Time@ DOL
b. C O S T T I M E @ D O L
c. COSTTIME@DOL
d. CostTime@ Dol

パラメータには、ENT、DOL、IOUの３つがあり、各々プログラムの特定、doループの特定、入出力処理の特定を指定する。

2）パラメータの詳細

ENT　　プログラムの計測の除外対象か追加対象かの指定

制御データで「GLOBAL_ENT=Yes」が指定されている場合には、そのプログラムは計測対象から除外され、「GLOBAL_ENT=No」の場合には計測対象となる。このディレクティブは、他のディレクティブに優先する。

(例)
```
Subroutine abc
implicit real*8 (a-h,o-z)
costtime@ ENT
:
:
end
```

DOL　　doループの計測の除外対象か追加対象かの指定

制御データで「GLOBAL_DOL=Yes」が指定されている場合には、そのdoループは計測対象から除外され、「GLOBAL_DOL=No」の場合には計測対象となる。このディレクトリが指定されて最初に出現したdoループがその対象となる。通常混乱を防止するため、doループの直前に指定した方が良い。

(例)
```
Subroutine abc
implicit real*8 (a-h,o-z)
:
costtime@ DOL                    対応doループがなく無効
:
costtime@ DOL
do 100 i=1,n
do 100 j=1,m
:
costtime@ DOL
do 300 k=1,n*m
:
end
```

IOU　　入出力処理の計測の除外対象か追加対象かの指定

制御データで「GLOBAL_IOU=Yes」が指定されている場合には、その入出力処理は計測対象から除外され、「GLOBAL_IOU=No」の場合には計測対象となる。このディレクティブが指定されて最初に出現した入出力処理がその対象となる。通常混乱を防止するため、入出力処理の直前に指定した方が良い。

(例)
```
Subroutine abc
implicit real*8 (a-h,o-z)
:
costtime@ IOU
if ( a(i) .gt. b ) write(...) .....
:
costtime@ IOU
write(...) .....
:
end
```

3.2.5　制限事項

goto文には単純goto文（go to n）と計算型goto文（go to (n,n,n,...n)e）、および割り当て型goto文（go to i (n,n,n,...n)）があり、ktoolではこの内、計算型goto文と割り当て型goto文が含まれる多重ループの計測は行わない。

3.2.6 表示出力

1）実行時の表示

ktoolを実行すると、計測元ソースの解析を行い、時間計測ルーチンを挿入し、集計および出力プログラム等を最後に挿入して計測用ソースを作成する。その際の処理ステップ等の情報や制御データ等、および計測制限等の情報を出力する。Fig. 3.4に出力例を示す。

2）出力結果

計測用ソースをコンパイル実行するとユーザ・プログラムが実行された後、以下のメッセージを出力する。

Tool result file : @@k_tool.f66	計測結果情報ファイル名
Tool export file : @@k_tool.f99	編集前計測結果情報ファイル名

計測結果情報ファイルの出力例をFig. 3.5に示す。計測結果情報はエントリの情報、入出力の情報、doループの情報、ソース・ファイルの情報に分かれる。

2.1）エントリの情報

a.RANK	消費時間の大きい順ランキング
b.PROGRAM NAME	プログラム名
c.TIMES	実行回数（呼び出し回数）
d.TIME(SEC)	消費時間（秒単位）
e.MEAN(SEC)	１回当たりの消費時間（秒単位）（=TIME／TIMES）
f.RATE(%)	計測対象全プログラムの中での消費時間の百分率
g.LINE NO.	プログラムのファイル内出現ライン番号
h.FILE	ソース・ファイルの情報で示されるファイル番号

2.2）入出力の情報

a.RANK	消費時間の大きい順ランキング
b.PROGRAM NAME	この入出力が存在するプログラム名
c.SEQ.	プログラム内の計測除外対象入出力も含む入出力出現順番
d.KIND	入出力の種別（read／write文等）
e.TIMES	実行回数
f.TIME(SEC)	消費時間（秒単位）
g.MEAN(SEC)	１回当たりの消費時間（秒単位）（=TIME／TIMES）

h.RATE(%)	計測対象全入出力の中での消費時間の百分率
i.LINE NO.	入出力のファイル内出現ライン番号
j.FILE	入出力が存在するファイルのファイル番号

2.3) doループの情報

a.RANK	消費時間の大きい順ランキング
b.PROGRAM NAME	このdoループが存在するプログラム名
c.SEQ.	プログラム内の計測除外対象doループも含むdoループ出現順番
d.LABEL	doループのラベル（ラベルなしは-1表示）
e.TIMES	実行回数
f.ROTATION	doループの総ループ回数
g.MEAN (ROT)	doループの１回当たりのループ回数（平均ループ回数）
h.TIME(SEC)	消費時間（秒単位）
i.MEAN(SEC)	１回当たりの消費時間（秒単位）（=TIME／TIMES）
j.RATE(%)	計測対象全doループ中での消費時間の百分率
k.T	doループ種別（１：ループ回数確定型、０：ループ回数未確定型）
l.LINE NO.	doループのファイル内出現ライン番号
m.FILE	doループが存在するファイルのファイル番号

多重ループでかつ共通のdo端末で同じラベルを利用しているdoループは、最内ループ以外には新たなラベルをktoolが設定する。したがって、「d.LABEL」に出力されるラベルは、計測元ソースのラベルではなく、追加設定されたラベルとなる。

2.4) ソース・ファイルの情報

a.FILE	ファイル番号
b.SOURCE FILE NAME	ソース・ファイル名

3.3 ptool

Paragonのnxライブラリにより並列化されたプログラムの、ノード間通信等の並列化オーバヘッドを測定するツールである。ライブラリはParagonのnxライブラリを想定して設計されたが、Paragonで稼働する他のライブラリ・セットも運用可能とする。時間計測ルーチンにはdclockを使用した。ptoolにおいて利用されるファイルおよび処理の流れをFig 3.6に示す。

3.3.1 計測法

具体的測定方法は、プログラムを分析し、登録されたライブラリに属するサブルーチンを検出した場合、それを時間計測ルーチンで挟み込み、そのサブルーチンで消費された経過時間を各ノード毎に計測する。事前に制御データで決められた試行回数だけdclockを実行し、時間計測ルーチンの平均値を利用してオーバヘッドを削減する。

3.3.2 集計法

集計はノード毎に行い、各ノード毎の経過時間と実行回数を出力する。実行回数についてはnxライブラリのmynode、numnode等は各ノードで処理が発生するため各々のノードで１回、csend等については送信側でのみ１回、crecv等については受信側でのみ１回とする。
また、システム・コールの種別毎の全経過時間および、ノード毎の全経過時間の集計も行う。

3.3.3 ptool制御データ

以下に、ptool制御データで設定可能なシンボル名を挙げ、その意味とディフォルト値ついて記す。

１）ファイル名、入口名、コモン名

計測用のために追加されるサブルーチン、インクルード・ファイル、変数の名前を指定する。

シンボル名	意味	ディフォルト
INP_SOC	計測元ソース・ファイル名	@@test.fff
MYSYSTEM	システム・データ・ファイル名	x_system.dat
OUT_SOC	計測用ソース・ファイル名	@@f22.f
CPU_CTR	計測インクルード・ファイル名	CPU_CTR
DUM_DUM0	ダミー・ルーチン名	DUM_DUM0
CPU_MSE	時刻取り出しルーチン名	CPU_MSE
DUM_DUM	自己オーバヘッド計測ルーチン名	DUM_DUM
ENT_ENT	計測開始処理ルーチン名	ENT_ENT
RET_ENT	計測終了、加算処理ルーチン名	RET_ENT
OUT_CTR	並列オーバヘッドの出力ルーチン名	OUT_CTR

（例）シンボル名の変更は次のように指定する。

```
INP_SOC = @@test.camp      ! 計測元ソース・ファイル名
MYSYSTEM = p_tool.sys      ! システム・データ・ファイル名
OUT_SOC = @@camp.f         ! 計測用ソース・ファイル名
CPU_CTR = @@test.inc       ! 計測インクルード・ファイル名
```

２）ファイル論理機番等

計測結果のファイル情報を指定する。

シンボル名	意味	ディフォルト
OUT_U66	計測結果の出力ユニット番号	66
OUT_F66	計測結果の出力ファイル名	@@p_tool.f66
OPN_U66	計測結果の出力ユニットのオープン処理要求 Yes／Noで要／否を指定	Yes
OUT_U99	計測結果のテーブル出力ユニット番号	99
OUT_F99	計測結果のテーブル出力ファイル名	@@x_tool.f99
OPN_U99	計測結果のテーブル出力ユニットのオープン処理要求 Yes／Noで要／否を指定	Yes

(例) ファイル出力情報の変更は次のように指定する。

```
OUT_F66 = @@test.f66        ! 計測結果の出力ファイル名
OUT_F66 = 66                ! 計測結果の出力ユニット番号
OUT_F99 = @@test.f99        ! 計測結果のテーブル出力ファイル名
OUT_F99 = 99                ! 計測結果のテーブル出力ユニット番号
```

3) 時間計測ルーチンの試行回数

ツール自身のオーバヘッドを削減するために別途行う、時間計測ルーチンの試行回数。0が指定された場合には、ツール自身のオーバヘッド時間を含んだ計測結果を出力する。

シンボル名	意味	ディフォルト
OHD_CTR	時間計測ルーチンの試行回数	0

4) 最大プロセッサ数

プログラムで使用するプロセッサ数を指定する。実際に使用するプロセッサ数以上の値を指定しておけばよい。

シンボル名	意味	ディフォルト
MAX_IDS	最大プロセッサ数	200

5) システム・ライブラリの指定

システム・データ内には、Paragonのシステム・ルーチン名、nxライブラリ・ルーチン名が記述されており、これらのルーチン名がディフォルトの計測対象となっている。これを無視して、計測するルーチン名をプライベートに指定することが可能である。

シンボル名	意味	ディフォルト
SYS_SUB	プライベートな計測対象ルーチン名	なし
SYS_FUNC	プライベートな計測対象関数名	なし

(例) プライベートな計測対象を一行毎に指定する。

```
SYS_SUB  = prvsub1          ! ルーチン名
SYS_SUB  = prvsub2          !
SYS_FUNC = prvfunc          ! 関数名
SYS_SUB  = gsum             ! nxライブラリ・ルーチンを指定してもよい
```

3.3.4 ptoolディレクティブ

計測用ソース上のすべてのライブラリcall文には、前後に計測ルーチンのcall文が挿入されるが、計測元ソース上にディレクティブを記述することにより、計測ルーチンが挿入されず計測の抑止を行うことができる。

第1カラムから「COSTTIME@パラメータ」で指定し、第1カラム以外のブランクは無視する。大文字、小文字の区別はない。

[形式]

CostTime@ ON | OFF

パラメータのonでディレクティブ以降の文を計測対象とすることを指示し、offで計測対象とせず、call文を挿入しないことを指示する。ディフォルトはonである。

例

```
      Subroutine
      ...
  CostTime@   Off      -----+
      iam  = mynode()     ¦この間は計測しない
      nodes = numnodes()  ¦
  CostTime@   On       -----+
      ...
      call csend(300,n,intsiz,-1,0) ！計測する
      ...
      call crecv(300,n,intsiz)      ！計測する
      ...
      end
```

パラメータのoffからonまでの範囲が計測対象外となる。対応するonが無い場合はend文までの範囲を計測対象外とし、次のプログラムから計測対象に戻す。

3.3.5 表示出力

1）実行時の表示

ptoolを実行すると、計測元ソースの解析を行い、時間計測ルーチンを挿入し、集計および出力プログラム等を最後に挿入して計測用ソースを作成する。その際の処理ステップ等の情報や制御データ等、および計測制限等の情報を出力する。Fig. 3.7に出力例を示す。

2）出力結果

計測用ソースをコンパイル実行するとユーザ・プログラムが実行された後、以下のメッセージを出力する。

```
Tool result file : @@p_tool.f66      計測結果情報ファイル名
Tool export file : @@p_tool.f99      編集前計測結果情報ファイル名
```

計測結果情報ファイルの出力例をFig. 3.8に示す。計測結果情報はシステム・コール種別の情報、時間計測ルーチン自身の情報、ソース・ファイルの情報、システム・コール毎かつノード毎の情報、総経過時間の情報、システム・コール毎のオーバヘッド合計時間の情報、ノード毎のオーバヘッド合計時間の情報に分かれる。

時間計測ルーチン自身の情報の「Inner」はdclockのオーバヘッド、「Outer」はそれを使用して作成した時間計測ルーチンのオーバヘッドである。「overhead counter」はオーバヘッドを算出するためのデータ採取回数である。

3.4 xtool

VPPの並列化ディレクティブにより並列化されたプログラムの、PE間通信等の並列化オーバヘッドを測定するツールである。VPPのxocl文で発生する並列化オーバヘッドを想定して設計がなされた。時間計測ルーチンにはgettodを使用した。xtoolにおいて利用されるファイルおよび処理の流れをFig 3.9に示す。

3.4.1 計測法

具体的測定方法は、プログラムを分析し、登録されたライブラリに属するxocl文を検出した場合、それを時間計測ルーチンで挟み込み、そのxocl文で消費された経過時間を各PE毎に計測する。事前に制御データで決められた試行回数だけgettodを実行し、時間計測ルーチンの平均値を利用してオーバヘッドを削減する。

3.4.2 集計法

集計はPE毎に行い、各PE毎の経過時間と実行回数を出力する。また、それぞれのPEでの実行回数についても集計する。
また、xocl文の種別毎の全経過時間およびPE毎の全経過時間の集計も行う。

3.4.3 xtool制御データ

以下に、ptool制御データで設定可能なシンボル名を挙げ、その意味とディフォルト値ついて記す。

1）ファイル名、入口名、コモン名

計測用のために追加されるサブルーチン、インクルード・ファイル、変数の名前を指定する。

シンボル名	意味	ディフォルト
INP_SOC	計測元ソース・ファイル名	@@test.fff
MYSYSTEM	システム・データ・ファイル名	x_system.dat
OUT_SOC	計測用ソース・ファイル名	@@test.f
XINC_GLO	計測インクルード・ファイル名（その１）	XINC_GLO
XINC_LOC	計測インクルード・ファイル名（その２）	XINC_LOC
X_INITIM	計測時間初期指定ルーチン名	X_INITIM
X_GLOSET	逐次実行時の計測ルーチン名	X_GLOSET
X_GLOSUM	逐次実行時の計測加算エントリ名	X_GLOSUM

X_LOCSET	冗長実行時の計測ルーチン名	X_LOCSET
X_LOCSUM	冗長実行時の計測加算エントリ名	X_LOCSUM
X_TIMSET	並列、冗長実行時の計測ルーチン名	X_TIMSET
X_TIMSUM	並列、冗長実行時の計測加算エントリ名	X_TIMSUM
X_PRINT	並列オーバヘッドの出力ルーチン名	X_PRINT
XDO_FRST	ＳＰＲＥＡＤ ＤＯ文計測用ローカル変数名	XDO_FRST
X_COMMON	計測インクルードのコモン・ブロック名（その１）	X_COMMON
X_LOCAL	計測インクルードのコモン・ブロック名（その２）	X_LOCAL
X_GETTOD	経過時間測定ルーチン名	X_GETTOD
	（＊）ディフォルト値は gettod である。	

(例) シンボル名の変更は次のように指定する。

```
INP_SOC = @@test.camp      ! 計測元ソース・ファイル名
MYSYSTEM = x_tool.sys      ! システム・データ・ファイル名
OUT_SOC = @@camp.f         ! 計測用ソース・ファイル名
XINC_GLO = @@test.glo      ! 計測インクルード・ファイル名（その１）
XINC_LOC = @@test.loc      ! 計測インクルード・ファイル名（その２）
```

2）ファイル論理機番等

計測結果のファイル情報を指定する。

シンボル名	意味	ディフォルト
OUT_U66	計測結果の出力ユニット番号	66
OUT_F66	計測結果の出力ファイル名	@@p_tool.f66
OPN_U66	計測結果の出力ユニットのオープン処理要求 Yes／Noで要／否を指定	Yes
OUT_U99	計測結果のテーブル出力ユニット番号	99
OUT_F99	計測結果のテーブル出力ファイル名	@@x_tool.f99
OPN_U99	計測結果のテーブル出力ユニットのオープン処理要求 Yes／Noで要／否を指定	Yes

(例) ファイル出力情報の変更は次のように指定する。

```
OUT_F66 = @@test.f66       ! 計測結果の出力ファイル名
OUT_F66 = 66               ! 計測結果の出力ユニット番号
OUT_F99 = @@test.f99       ! 計測結果のテーブル出力ファイル名
OUT_F99 = 99               ! 計測結果のテーブル出力ユニット番号
```

3）時間計測ルーチンの試行回数

ツール自身のオーバヘッドを削減するために別途行う、時間計測ルーチンの試行回数。０が指定された場合には、ツール自身のオーバヘッド時間を含んだ計測結果を出力する。

シンボル名	意味	ディフォルト
OHD_CTR	時間計測ルーチンの試行回数	1000

3.4.4 xtoolディレクティブ

計測用ソース上のすべてのxocl文には、前後に計測ルーチンのcall文が挿入されるが、計測元ソース上にディレクティブを記述することにより、計測ルーチンを挿入せず計測の抑止を行うことができる。

第１カラムから「COSTTIME@パラメータ」で指定し、第１カラム以外のブランクは無視する。大文字、小文字の区別はない。

[形式]

CostTime@ ON | OFF

パラメータのonでディレクティブ以降の文を計測対象とすることを指示し、offで計測対象とせず、call文を挿入しないことを指示する。ディフォルトはonである。

例

```
      Subroutine
      ...
!Xocl Parallel Region        ディフォルトは計測対象である
      ...
!Xocl Spread Region
      ...
CostTime@   Off      -----+
!Xocl Spread Do           ¦この間の Spread Do は計測対象外
      ...           ¦
!Xocl End Spread Do       ¦
CostTime@   On       -----+ 以降の文は計測対象
      ...
!Xocl Region
      ...
CostTime@   Off      -----+
!Xocl Spread Move         ¦この間の Spread Move は計測対象外
      ...           ¦
!Xocl End Spread Move     ¦
CostTime@   On       -----+ 以降の文は計測対象
      ...
!Xocl End Spread Region
      ...
!Xocl End Parallel
      ...
      end
```

パラメータのoffからonまでの範囲が計測対象外となる。対応するonが無い場合はend文までの範囲を計測対象外とし、次のプログラムから計測対象に戻す。

3.4.5 表示出力

1）実行時の表示

xtoolを実行すると、計測元ソースの解析を行い、時間計測ルーチンを挿入し、集計および出力プログラム等を最後に挿入して計測用ソースを作成する。その際の処理ステップ等の情報や制御データ等、および計測制限等の情報を出力する。Fig. 3.10に出力例を示す。

2）出力結果

計測用ソースをコンパイル実行するとユーザ・プログラムが実行された後、以下のメッセージを出力する。

Tool result file : @@x_tool.f66	計測結果情報ファイル名
Tool export file : @@x_tool.f99	編集前計測結果情報ファイル名

計測結果情報ファイルの出力例をFig. 3.11に示す。計測結果情報はxocl種別の情報、時間計測ルーチン自身の情報、ソース・ファイルの情報、xocl毎かつPE毎の情報、総経過時間の情報、xocl毎のオーバヘッド合計時間の情報、PE毎のオーバヘッド合計時間の情報に分かれる。
時間計測ルーチン自身の情報の「Inner」はgettodのオーバヘッド、「Outer」はそれを使用して作成した時間計測ルーチンのオーバヘッドである。「overhead counter」はオーバヘッドを算出したときの実行回数である。

```
*DECK CPU_MSE                                                    CPU_MSE        時刻計測開始処理
C
      FUNCTION CPU_MSE (CPUSEW)                                  CPU_MSE
C     RETURN MILI SECONDS (ENTRY  PROCESS)
      REAL*8 CPUSEW
C
C For Fujitsu VPP 'gettod' time routine
C      Sub.
C      A
C      +  --t1  (CPUSEW)        Sub. A Net Time
C      I  --t2  (CPUSEC)           = (t3-t2)+(t6-t5)
C      I      B                 Sub. B Net Time
C      +------+  --t3              = (t5-t4)
C             I  --t4           OverHead
C             I                    = (t2-t1)+(t4-t3)+alpha
C      +------+  --t5
C      I
C      +  --t6
C
      REAL*8 CPU_MSE                                             CPU_MSE
      REAL*8 CPUSEC
C     TIMEFACT : CONVERSION FACTOR TO MILI-SECOND
      REAL*8 TIMEFACT
      REAL*8 WORKTM
C     ==========================================
C     ------------------------------
C---- VPP300/VPP500 CPU TIME (CLOCK)
      DATA   TIMEFACT / 1000.0 /
C
      CALL CLOCK (CPUSEC,2,2)
      CPUSEW  = CPUSEC/TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------------
C---- VPP300/VPP500 ELAPSED TIME (GETTOD)
      DATA   TIMEFACT / 1000.0 /
C
      call gettod(CPUSEW)
      CPUSEW  = CPUSEW/TIMEFACT
      call gettod(CPUSEC)
      WORKTM  = CPUSEC/TIMEFACT
C     ------------------------------
C---- Monte-4 CPU TIME (CLOCK)
      DATA   TIMEFACT / 1000.0 /
C
      CALL CLOCK (CPUSEC)
      CPUSEW  = CPUSEC*TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------------
C---- Monte-4 ELAPSED TIME (ETIME)
      REAL   ETIME,TARRAY(2)
      DATA   TIMEFACT / 1000.0 /
C
      CPUSEC  = ETIME( TARRAY )
      CPUSEW  = CPUSEC*TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------------
C---- Paragon CPU TIME (ETIME)
      include 'fnx.h'
      REAL   ETIME,TARRAY(2)
      DATA   TIMEFACT / 1000.0 /
C
      CPUSEC  = ETIME( TARRAY )
      CPUSEW  = CPUSEC*TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------------
C---- Paragon ELAPSED TIME (DCLOCK)
      include 'fnx.h'
      DATA   TIMEFACT / 1000.0 /
C
      CPUSEC  = DCLOCK()
      CPUSEW  = CPUSEC*TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------------
C---- SX-4 CPU TIME (CLOCK)
      DATA   TIMEFACT / 1000.0 /
C
      CALL CLOCK (CPUSEC)
      CPUSEW  = CPUSEC*TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------------
C---- SX-4 ELAPSED TIME (ETIME)
      REAL   ETIME,TARRAY(2)
      DATA   TIMEFACT / 1000.0 /
C
      CPUSEC  = ETIME( TARRAY )
      CPUSEW  = CPUSEC*TIMEFACT
      WORKTM  = CPUSEW
```

Fig. 3.1 Source programs of timer routine in ktool (to be continued)

```
C     ------------------------
C---- SP2 CPU TIME (etime_)
      REAL    etime_,TARRAY(2)
      DATA    TIMEFACT / 1000.0 /
C
      CPUSEC  = etime_(TARRAY)
      CPUSEW  = CPUSEC*TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------
C---- SP2 ELAPSED TIME (TIMEF)
      REAL*8 TIMEF
C
      CPUSEC  = TIMEF()
      CPUSEW  = CPUSEC
      WORKTM  = CPUSEW
C     ------------------------
C---- SR2201 CPU TIME
      DATA    TIMEFACT / 1000.0 /
      DATA    IFIRST   / 1 /
C
      IF ( IFIRST .EQ. 1 ) THEN
        IFIRST = 0
        CALL XCLOCK(CPUSEC, 3)
      ENDIF
      CALL XCLOCK(CPUSEC, 5)
      CPUSEW  = CPUSEC*TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------
C---- SR2201 ELAPSED TIME
      DATA    TIMEFACT / 1000.0 /
      DATA    IFIRST   / 1 /
C
      IF ( IFIRST .EQ. 1 ) THEN
        IFIRST = 0
        CALL XCLOCK(CPUSEC, 7)
      ENDIF
      CALL XCLOCK(CPUSEC, 8)
      CPUSEW  = CPUSEC*TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------
C---- T94 CPU TIME (SECOND)
      DATA    TIMEFACT / 1000.0 /
C
      CALL SECOND (CPUSEC)
      CPUSEW  = CPUSEC*TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------
C---- T94 ELAPSED TIME (TIMEF)
C
      CALL TIMEF (CPUSEC)
      CPUSEW  = CPUSEC
      WORKTM  = CPUSEW
C     ========================================
      CPU_MSE = WORKTM                                  CPU_MSE
C
      END
                                                        CPU_MSR     時刻計測終了処理
*DECK CPU_MSR
C
      FUNCTION CPU_MSR ()                               CPU_MSR
C     RETURN MILI SECONDS (RETURN PROCESS)
      REAL*8 CPU_MSR                                    CPU_MSR
      REAL*8 CPUSEC
C     TIMEFACT : CONVERSION FACTOR TO MILI-SECOND
      REAL*8 TIMEFACT
      REAL*8 WORKTM
C     ========================================
C     ------------------------
C---- VPP300/VPP500 CPU TIME (CLOCK)
      DATA    TIMEFACT / 1000.0 /
C
      CALL CLOCK (CPUSEC,2,2)
      WORKTM  = CPUSEC/TIMEFACT
C     ------------------------
C---- VPP300/VPP500 ELAPSED TIME (GETTOD)
      DATA    TIMEFACT / 1000.0 /
C
      call gettod(CPUSEC)
      WORKTM  = CPUSEC/TIMEFACT
C     ------------------------
```

Fig. 3.1 Source programs of timer routine in ktool (to be continued)

```
C---- Monte-4 CPU TIME (CLOCK)
      DATA   TIMEFACT / 1000.0 /
C
      CALL CLOCK (CPUSEC)
      WORKTM  = CPUSEC*TIMEFACT
C     ------------------------------
C---- Monte-4 ELAPSED TIME (ETIME)
      REAL   ETIME,TARRAY(2)
      DATA   TIMEFACT / 1000.0 /
C
      CPUSEC  = ETIME( TARRAY )
      WORKTM  = CPUSEC*TIMEFACT
C     ------------------------------
C---- Paragon CPU TIME (ETIME)
      include 'fnx.h'
      REAL   ETIME,TARRAY(2)
      DATA   TIMEFACT / 1000.0 /
C
      CPUSEC  = ETIME( TARRAY )
      WORKTM  = CPUSEC*TIMEFACT
C     ------------------------------
C---- Paragon ELAPSED TIME (DCLOCK)
      include 'fnx.h'
      DATA   TIMEFACT / 1000.0 /
C
      CPUSEC  = DCLOCK()
      WORKTM  = CPUSEC*TIMEFACT
C     ------------------------------
C---- SX-4 CPU TIME (CLOCK)
      DATA   TIMEFACT / 1000.0 /
C
      CALL CLOCK (CPUSEC)
      WORKTM  = CPUSEC*TIMEFACT
C     ------------------------------
C---- SX-4 ELAPSED TIME (ETIME)
      REAL   ETIME,TARRAY(2)
      DATA   TIMEFACT / 1000.0 /
C
      CPUSEC  = ETIME( TARRAY )
      WORKTM  = CPUSEC*TIMEFACT
C     ------------------------------
C---- SP2 CPU TIME (etime_)
      REAL   etime_,TARRAY(2)
      DATA   TIMEFACT / 1000.0 /
C
      CPUSEC  = etime_(TARRAY)
      WORKTM  = CPUSEC*TIMEFACT
C     ------------------------------
C---- SP2 ELAPSED TIME (TIMEF)
      REAL*8 TIMEF
C
      CPUSEC  = TIMEF()
      WORKTM  = CPUSEC
C     ------------------------------
C---- SR2201 CPU TIME
      DATA   TIMEFACT / 1000.0 /
C
      CALL XCLOCK(CPUSEC,5)
      WORKTM  = CPUSEC*TIMEFACT
C     ------------------------------
C---- SR2201 ELAPSED TIME
      DATA   TIMEFACT / 1000.0 /
C
      CALL XCLOCK(CPUSEC,8)
      WORKTM  = CPUSEC*TIMEFACT
C     ------------------------------
C---- T94 CPU TIME (SECOND)
      DATA   TIMEFACT / 1000.0 /
C
      CALL SECOND (CPUSEC)
      WORKTM  = CPUSEC*TIMEFACT
C     ------------------------------
C---- T94 ELAPSED TIME (TIMEF)
C
      CALL TIMEF (CPUSEC)
      WORKTM  = CPUSEC
C     ==============================
      CPU_MSR = WORKTM                                    CPU_MSR
C
      END
```

Fig. 3.1 Source programs of timer routine in ktool (continued)

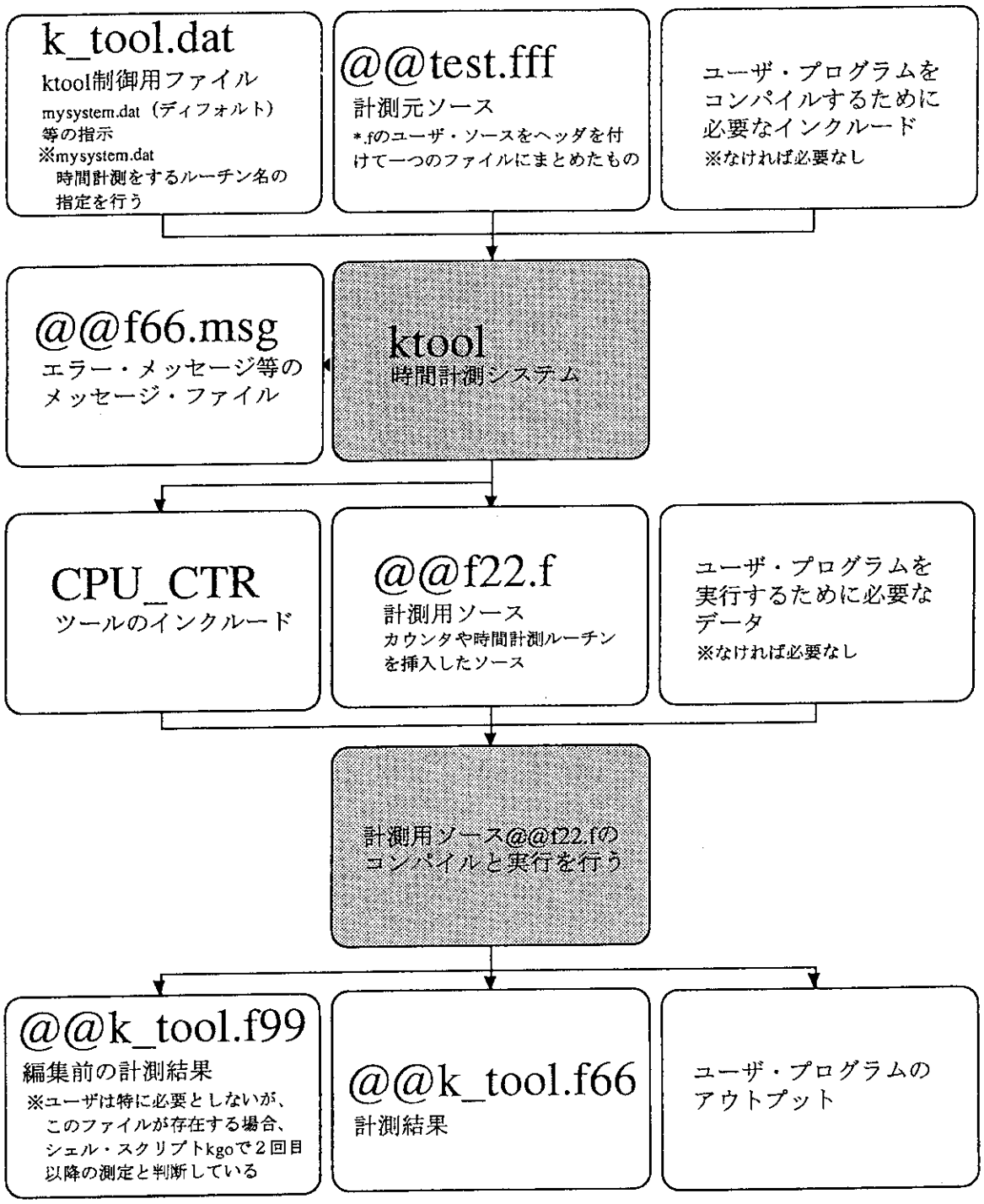


Fig. 3.2 Files to be used or created by ktool

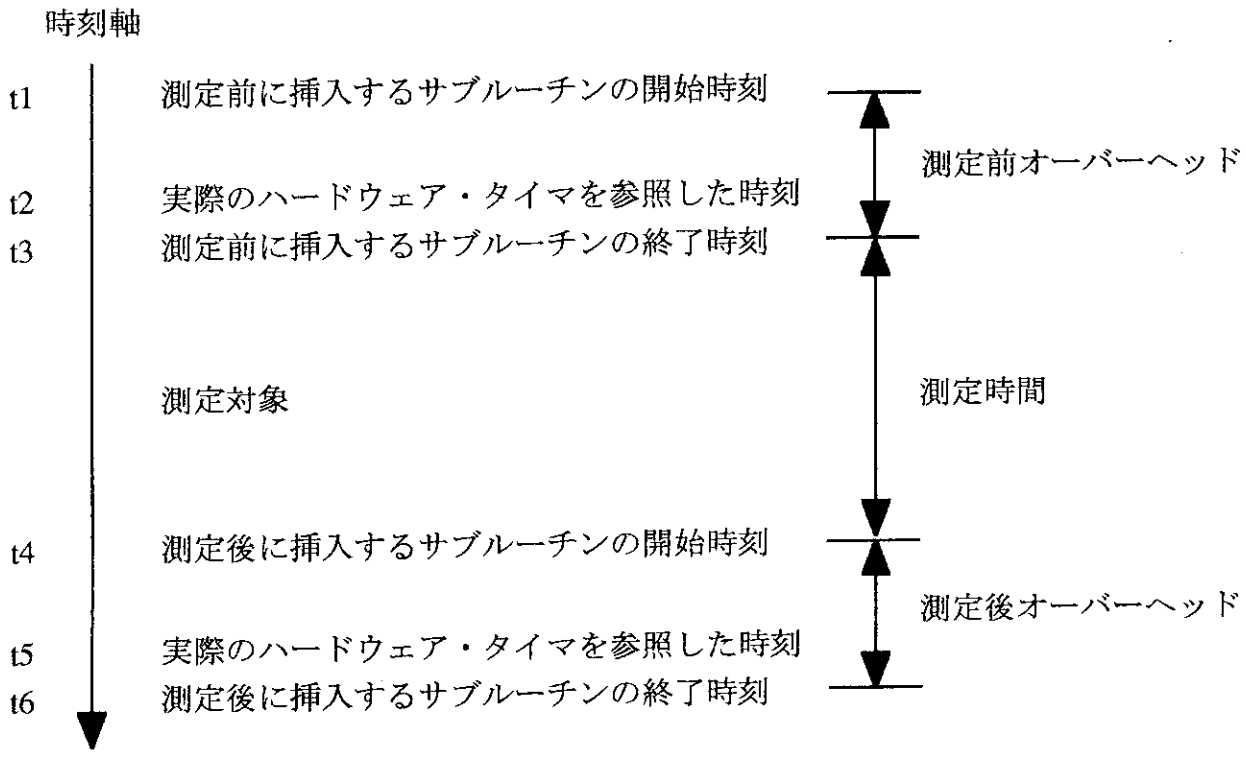


Fig. 3.3 Measurement method

```
- Tool -     (V00/L05) (1996/10/31)

##########################################
##  Input data listing if error exist.  ##
##    Kind = 1: No keyword              ##
##    Kind = 2: No data                 ##
##    Kind = 3: Keyword error           ##
##    Kind = 4: Illegal logical unit    ##
##    Kind = 5: Data error              ##
##    Kind = 6: Too long include path   ##
##########################################
Line Kind   D a t a
---- ---- ----*----1----*----2----*----3----*----4..
##### No error input data. #####

 Start function : Measure

start pass 0/4

e n d pass 0/4  cpu(sec):       0.000 total:       0.000
start pass 1/4
Pass1-1

Pass1-2

Pass1-3

e n d pass 1/4  cpu(sec):       0.000 total:       0.000
start pass 2/4
e n d pass 2/4  cpu(sec):       0.000 total:       0.000
start pass 3/4
Entry : ADDVAR                          str:       1 end:      18
Entry : CALCFN                          str:      19 end:     135
Entry : CONVOL                          str:     136 end:     168
Entry : DATAIN                          str:     169 end:     217
Entry : GETFLD                          str:     218 end:     261
Entry : GETKSP                          str:     262 end:     310
Entry : MKCNST                          str:     311 end:     341
Entry : NEXTJB                          str:     342 end:     361
Entry : OMG2PV                          str:     362 end:     424
Entry : PRINTL                          str:     425 end:     434
Entry : PUTFLD                          str:     435 end:     466
Entry : PUTKSP                          str:     467 end:     514
Entry : TRANS5                          str:     515 end:     670
Entry : VDFT3B                          str:     671 end:     880
Entry : VDFT3F                          str:     881 end:    1087
Entry : VDFT3I                          str:    1088 end:    1141
 e n d pass 3/4  cpu(sec):       0.000 total:       0.000
 start pass 4/4
 e n d pass 4/4  cpu(sec):       0.000 total:       0.000
```

Fig. 3.4 Output example in program analysis phase by ktool (to be continued)

```
pass  start(sec)    end(sec)  used (sec)
----  ----------  ----------  ----------
   0       0.000       0.000       0.000
   1       0.000       0.000       0.000
   2       0.000       0.000       0.000
   3       0.000       0.000       0.000
   4       0.000       0.000       0.000
                               =========
                               0.000

==============  Tool Information  ==============
  1) Global Entry                  : Yes
  2) Global Do loop                : Yes
  3) Global I/O                    : Yes
  4) Unsatisified Do loop          : Yes
  5) Just counter measure          : No
  6) Restart job?                  : Yes
  7) No. of Entry   specified data:          0
  8) No. of I/O     specified data:          0
  9) No. of Do loop specified data:          0
 10) Max. of Entry   execution     :      10000
 11) Max. of I/O     execution     :      10000
 12) Max. of Do loop execution     :      10000
 13) Max. of Do loop mean rotation:          0
```

制御データGLOBAL_ENT設定値
制御データGLOBAL_DOL設定値
GLOBAL_IOU設定値
UNSAT_DOL設定値
JST_CTR設定値
リスタート処理情報
ENTRY 設定データ数
IOSEQ 設定データ数
DOLABEL, DOSEQ設定データ数
MAX_ENT設定値
MAX_IOU設定値
MAX_DOL設定値
MAX_MRT設定値

Fig. 3.4 Output example in program analysis phase by ktool (continued)

(※ 報告時間の単位は秒。レートは消費時間百分率。)

エントリの情報

<<< PROGRAM UNIT RESULTS >>>

順位 RANK	プログラム名 PROGRAM NAME	実行回数 TIMES	実行時間 TIME(SEC)	平均実行時間 MEAN(SEC)	レート RATE(%)	行番号 LINE NO.	ファイル番号 FILE
1	VDFT3F	1200	15.824663	0.013187219	45.367	1	15
2	PUTFLD	5	7.385782	1.477156369	21.174	1	11
3	VDFT3B	600	7.165257	0.011942095	20.542	1	14
==========	省略	==========					
16	VDFT3I	1	0.000071	0.000071358	0.000	1	16
			34.881328				
			合計実行時間				

入出力の情報

<<< I/O PROCESS RESULTS >>>

順位 RANK	プログラム名 PROGRAM NAME	出現順位 SEQ.	種別 KIND	実行回数 TIMES	実行時間 TIME(SEC)	平均実行時間 MEAN(SEC)	レート RATE(%)	行番号 LINE NO.	ファイル番号 FILE
1	PUTFLD	7	close	5	3.897189	0.779437804	47.498	28	11
2	PUTFLD	6	write	5	3.163304	0.632660754	38.553	19	11
3	GETFLD	2	open	1	0.420309	0.420308792	5.123	12	5
==========	省略	==========							
22	GETFLD	4	write	0	0.000000	0.000000000	0.000	18	5
					8.204972				
					合計実行時間				

doループの情報

<<< DO LOOP RESULTS >>>

順位 RANK	プログラム名 PROGRAM NAME	出現順位 SEQ.	ラベル LABEL	実行回数 TIMES	総回転数 ROTATION	平均回転数 MEAN (ROT)	実行時間 TIME(SEC)	平均実行時間 MEAN(SEC)	レート RATE(%)	T	行番号 LINE NO.	ファイル番号 FILE
1	VDFT3F	32	920	76800	1228800.	16.0	2.316751	0.000030166	8.194	I	166	15
2	VDFT3F	29	918	76800	1228800.	16.0	2.301093	0.000029962	8.139	I	148	15
3	VDFT3F	16	910	76800	1228800.	16.0	1.948373	0.000025369	6.891	I	81	15
==========	省略	==========										
136	VDFT3I	5	210	1	5.	5.0	0.000000	0.000000000	0.000	I	40	16
							28.274056					
							合計実行時間					

ソース・ファイルの情報

<<< SOURCE FILE NAME(S) >>>

ファイル番号 FILE	ソース・ファイル名 SOURCE FILE NAME
1	addvar.f
2	calcfn.f
3	convol.f
==========	省略 ==========
16	vdft3i.f

Fig. 3.5 Output example in program execution phase by ktool

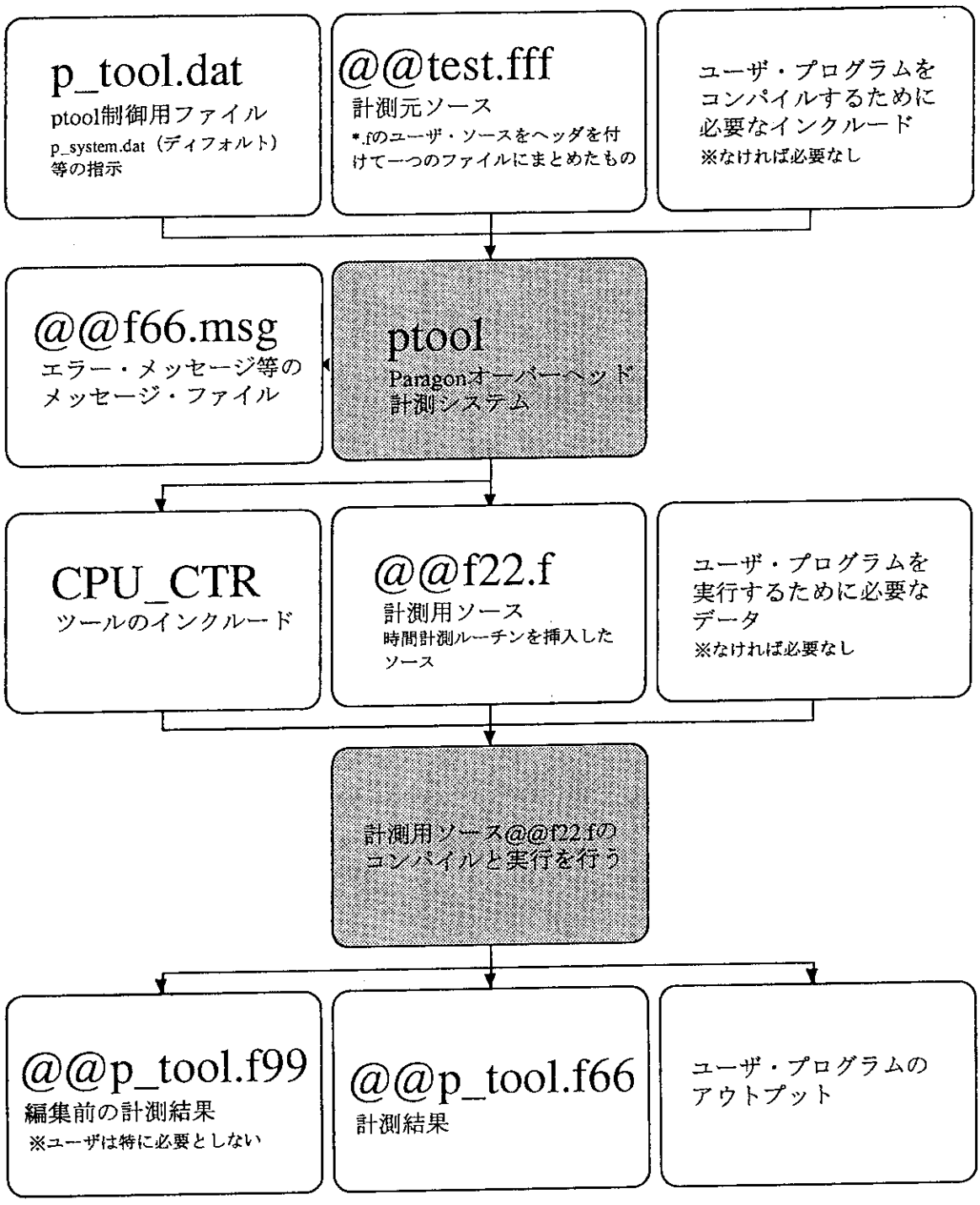


Fig. 3.6 Files to be used or created by ptool

```
 - Tool -      (V00/L05) (1996/10/31)

########################################
##  Input data listing if error exist.  ##
##    Kind = 1: No keyword              ##
##    Kind = 2: No data                 ##
##    Kind = 3: Keyword error           ##
##    Kind = 4: Illegal logical unit    ##
##    Kind = 5: Data error              ##
##    Kind = 6: Too long include path   ##
########################################
Line Kind   D a t a
---- ---- ----*----1----*----2----*----3----*----4..
##### No error input data. #####

 Start function : Measure

 start pass 0/4

 e n d pass 0/4  cpu(sec):       0.000 total:       0.000
 start pass 1/4
 Pass1-1

 Pass1-2

 Pass1-3

 e n d pass 1/4  cpu(sec):       0.000 total:       0.000
 start pass 2/4
 e n d pass 2/4  cpu(sec):       0.000 total:       0.000
 start pass 3/4
Entry : ADDVAR                          str:     1 end:     18
Entry : CALCFN                          str:    19 end:    135
Entry : CONVOL                          str:   136 end:    165
Entry : DATAIN                          str:   166 end:    247
Entry : GETFLD                          str:   248 end:    322
Entry : GETKSP                          str:   323 end:    371
Entry : MKCNST                          str:   372 end:    402
Entry : NEXTJB                          str:   403 end:    427
Entry : OMG2PV                          str:   428 end:    493
Entry : PRINTL                          str:   494 end:    507
Entry : PUTFLD                          str:   508 end:    543
Entry : PUTKSP                          str:   544 end:    608
Entry : TRANS5                          str:   609 end:    764
Entry : VDFT3B                          str:   765 end:   1040
Entry : VDFT3F                          str:  1041 end:   1301
Entry : VDFT3I                          str:  1302 end:   1355
 e n d pass 3/4  cpu(sec):       0.000 total:       0.000
 start pass 4/4
 e n d pass 4/4  cpu(sec):       0.000 total:       0.000

pass  start(sec)    end(sec)  used (sec)
----  ----------  ----------  ----------
   0       0.000       0.000       0.000
   1       0.000       0.000       0.000
   2       0.000       0.000       0.000
   3       0.000       0.000       0.000
   4       0.000       0.000       0.000
                               ==========
                               0.000

==========  Tool Information  ==========
  1) Output source      file name : ##f22.f    計測用ソース・ファイル名
  2) Output include     file name : CPU_CTR    インクルード・ファイル名
  3) No. of Entry measure prog. :       16    プログラム数
  4) No. of Measure libraries   :       71    ライブラリ登録システム・ルーチン数
```

Fig. 3.7 Output example in program analysis phase by ptool

```
(※ 報告時間の単位は秒。)
1 改ページ制御記号
<<< Paragon Fortran System Calls Cost List >>>

 Number of nodes   :     2  実行ノード数
 System call lines :    71  システム・コールの行数          日-月-年 時:分:秒
                                                          2-Sep-96 18:17:47
 順番号  ルーチン名                    連番号    システム・コール名    システム・コール種別  行番号 ファイル名
  seq.  subprogram name                 loc.    system call name
 -----  ----*----1----*----2----*----3--  -----  ----------------
     1  DATAIN                             1    MYNODE                Function        10  datain.f
     2  DATAIN                             2    NUMNODES              Function        11  datain.f
     3  DATAIN                             3    CSEND                 Subroutine      50  datain.f
 ========== 省略 ==========
    12  VDFT3F                            71    CRECV                 Subroutine     255  vdft3f.f

時間計測ルーチン自身のオーバーヘッド
 Self-Overhead Time per one-pair call. (    1000 overhead counter)
  ノード番号  作成タイマ使用時間  dclock使用時間
     node#        Outer          Inner    (sec)
        1      0.000007       0.000002
        2      0.000008       0.000002

ファイル番号 ソース・ファイル名
 File# filename
    1  addvar.f
    2  calcfn.f
    3  convol.f
 ========== 省略 ==========
   16  vdft3i.f                                          日-月-年 時:分:秒
                                                        2-Sep-96 18:25:05
 順番号  ルーチン名                    連番号 システム・コール名 ファイル番号 行番号
  seq.  subprogram name                 loc.    system call name  File#Line.
 ------  ----*----1----*----2----*----3--  ---  ----------------  ---- -----
                                        ノード番号          時間        実行回数
                                           node        time(sec)      counts
                                           ---#  ----------------  ----------
     1  DATAIN                             1     MYNODE              4    10
                                           1        0.000006  (       1)
                                           2        0.000006  (       1)
     2  DATAIN                             2     NUMNODES            4    11
                                           1        0.000005  (       1)
                                           2        0.000004  (       1)
     3  DATAIN                             3     CSEND               4    50
                                           1        0.000152  (       1)
                                           2        0.000000  (       0)
 ========== 省略 ==========
    12  VDFT3F                            71     CRECV              15   255
                                           1        2.556386  (   19200)
                                           2        2.540179  (   19200)
計測用プログラムの総経過時間                                  日-月-年 時:分:秒
 TOTAL  Elapsed  Time                                        2-Sep-96  18:25:05
    438.140892 second.
 順番号  システム・コール名      全ノードでのオーバーヘッド合計時間    出現回数
  seq.  system call name (summation all node)          time(sec)      counts
 -----  ----*----1----*----2----*----3--          ----------------  ----------
     1  CPROBE                                        0.000000  (       0)
     2  CPROBEX                                       0.000000  (       0)
     3  CREAD                                         0.000000  (       0)
 ========== 省略 ==========
   113  NX_WAITALL                                    0.000000  (       0)

ノード毎のオーバーヘッド総量          ノード番号 ノード毎総計時間
 TOTAL  OVERHEAD                          node  total time(sec)
                                          ---#  ----------------
                                             1       40.120147
                                             2       49.737358
```

Fig. 3.8 Output example in program execution phase by ptool

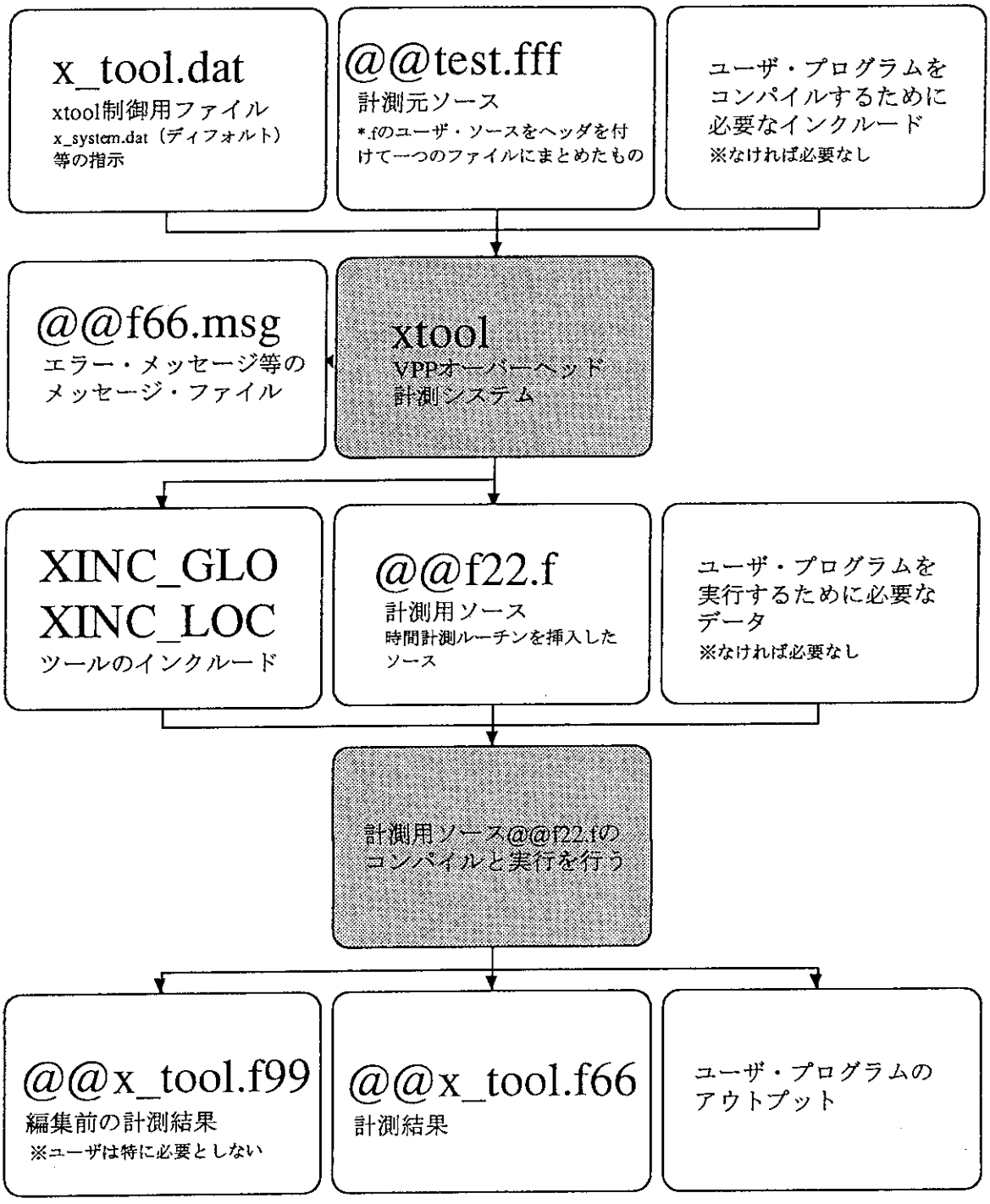


Fig. 3.9 Files to be used or created by xtool

```
 - Tool -     (V00/L05) (1996/10/31)

########################################
##  Input data listing if error exist.  ##
##    Kind = 1: No keyword              ##
##    Kind = 2: No data                 ##
##    Kind = 3: Keyword error           ##
##    Kind = 4: Illegal logical unit    ##
##    Kind = 5: Data error              ##
##    Kind = 6: Too long include path   ##
########################################
Line Kind   D a t a
---- ---- ----*----1----*----2----*----3----*----4..
##### No error input data. #####

 Start function : Measure

start pass 0/4

e n d pass 0/4  cpu(sec):       0.000 total:       0.000
start pass 1/4
Pass1-1

Pass1-2

Pass1-3

e n d pass 1/4  cpu(sec):       0.000 total:       0.000
start pass 2/4
 Processor PE(4 ) in GETKSP
 SubProcessor SPE in GETKSP
 Processor PE(4 ) in PUTKSP
 SubProcessor SPE in PUTKSP
 Processor PE(4 ) in TRANS5
 Processor PE(4 ) in VDFT3B
 SubProcessor SPE in VDFT3B
 Processor PE(4 ) in VDFT3F
 SubProcessor SPE in VDFT3F
 Processor PE(4 ) in VDFTZB
 SubProcessor SPE in VDFTZB
 Processor PE(4 ) in VDFTZF
 SubProcessor SPE in VDFTZF
e n d pass 2/4  cpu(sec):       0.000 total:       0.000
start pass 3/4
Entry : ADDVAR                        str:     1 end:    18
Entry : CALCFN                        str:    19 end:   155
Entry : DATAIN                        str:   156 end:   204
Entry : GETFLD                        str:   205 end:   248
Entry : GETKSP                        str:   249 end:   386
Entry : MKCNST                        str:   387 end:   417
Entry : NEXTJB                        str:   418 end:   437
Entry : OMG2PV                        str:   438 end:   496
Entry : PRINTL                        str:   497 end:   506
Entry : PUTFLD                        str:   507 end:   538
Entry : PUTKSP                        str:   539 end:   697
Entry : TRANS5                        str:   698 end:   867
Entry : VDFT3B                        str:   868 end:  1043
Entry : VDFT3F                        str:  1044 end:  1218
Entry : VDFT3I                        str:  1219 end:  1272
Entry : VDFTZB                        str:  1273 end:  1394
Entry : VDFTZF                        str:  1395 end:  1516
 e n d pass 3/4  cpu(sec):       0.000 total:       0.000
 start pass 4/4
 e n d pass 4/4  cpu(sec):       0.000 total:       0.000
```

Fig. 3.10 Output example in program analysis phase by xtool (to be continued)

```
pass  start(sec)    end(sec)  used (sec)
----  ----------  ----------  ----------
   0       0.000       0.000       0.000
   1       0.000       0.000       0.000
   2       0.000       0.000       0.000
   3       0.000       0.000       0.000
   4       0.000       0.000       0.000
                               ==========
                               0.000

============  Tool Information  ============
  1) Output source      file name : @@f22.f
  2) Output include #1  file name : XINC_GLO
  3) Output include #2  file name : XINC_LOC
```

計測用ソース・ファイル名
インクルード・ファイル名
インクルード・ファイル名

Fig. 3.10 Output example in program analysis phase by xtool (continued)

```
(※ 報告時間の単位は秒。)
1 改ページ制御記号
<<< XOCL Executable Statements Cost List >>>

 Number of Processors :    4 実行PE数
 Executable XOCL Lines :   70 XOCLの行数
                                                    年/月/日 時:分:秒
XOCL Statement List                                  96/09/11 17:51:23
順番号 ルーチン名       連番号 XOCL文                ファイル番号 ファイル名
  Seq. SubProgram Name  Loc. XOCL Statement          File# Name
------ ----*----1----*- ---- ------------------------------ ----- ----
     1 GETKSP             1 SPREADMOVE                    33 getksp.f
     2 GETKSP             2 ENDSPREAD(R1)                 39 getksp.f
     3 GETKSP             3 MOVEWAIT(R1)                  40 getksp.f
========== 省略 ==========
    13 VDFTZF            70 MOVEWAIT(F2)                 117 vdftzf.f

時間計測ルーチン自身のオーバーヘッド
Self-Overhead Time per one-pair call. (    1000 overhead counter)
     PE番号 作成タイマ使用時間 gettod使用時間
     node#        Outer          Inner   (sec)
         1     0.000054       0.000010
         2     0.000054       0.000009
         3     0.000053       0.000009
         4     0.000053       0.000009
ファイル番号 ソース・ファイル名
File# filename
   1 addvar.f
   2 calcfn.f
   3 datain.f
========== 省略 ==========
  17 vdftzf.f
                                                    年/月/日 時:分:秒
XOCL Start Parallel Region 並列処理開始時刻          96/09/11 17:51:23
                                                    年/月/日 時:分:秒
XOCL End   Parallel Region 並列処理終了時刻          96/09/11 18:22:59
順番号 ルーチン名       連番号 XOCL文                ファイル番号 行番号
  Seq. SubProgram Name  Loc. XOCL Statement          File# Line#
------ ----*----1----*- ---- ------------------------------ ----- -----
                                        PE番号          時間        実行回数
                                        PE.No       Time(Sec)  (  Counts)
                                        ---#  ---------------  ----------
     1 GETKSP             1 SPREADMOVE                      5    33
                                           1      0.000000  (       0)
                                           2      0.000000  (       0)
                                           3      0.000000  (       0)
                                           4      0.000000  (       0)
     2 GETKSP             2 ENDSPREAD(R1)                   5    39
                                           1      0.545347  (    1200)
                                           2      0.547760  (    1200)
                                           3      0.549340  (    1200)
                                           4      0.546390  (    1200)
========== 省略 ==========
    13 VDFTZF            70 MOVEWAIT(F2)                  17   117
                                           1      0.026451  (    1200)
                                           2      0.027819  (    1200)
                                           3      0.027491  (    1200)
                                           4      0.028197  (    1200)

   並列処理の総経過時間                                  年/月/日 時:分:秒
TOTAL Elapsed Time(Sec) & OVERHEAD COST                 96/09/11 18:22:59
      1896.048205                       PE番号    PE毎総計時間
                                        PE.No     Time(Sec)
                                        ---#  ---------------
                                           1     13.082205
                                           2     13.120046
                                           3     12.994081
                                           4     13.128103
```

Fig. 3.11 Output example in program execution phase by xtool

4．利用法

kpxはソース・プログラムの形態で供給するため、ユーザがインストールして利用することになる。したがって、ユーザが簡単にインストールできるようParagon、SP2、SR2201、VPP500、VPP300、Monte-4、SX-4、T90でテストされたinstallerを用意した。また、実行方法もカスタマイズを含め繁雑になりうる。そこで、ユーザが簡単に利用できる基本的な実行シェル・スクリプトを用意した。この実行シェル・スクリプトは、これをユーザが見本とし、kpxの利用法を理解するのを補助することも目的としている。さらに、「付録C」の「kpx利用手引き」を作成した他、遠隔ユーザのためにアスキーの利用補助ファイルをツールに添付した。

4.1 設置

kpxを利用するにはまず、日本原子力研究所、計算科学技術推進センターのファイル・サーバccsemfaにアクセスし、tar形式のファイルを対象コンピュータのホーム・ディレクトリへ取り込む。Fig. 4.1にkpxの取り込み方法を示す。

ホーム・ディレクトリへの転送が終了したら、kpx.tarを解凍しinstallerを実行する。installerを実行すると、ホーム・ディレクトリにディレクトリkpxが作成され、内容を展開し、最後にホーム・ディレクトリのtool.tar、installerを消去すると共に、kpx.tar、READMEをディレクトリkpxに移動する。kpxはソースで供給されるので、installerは単にファイル・シルテムを構築し、ファイルを保存するだけでなく、対象コンピュータで現在使用されているバージョンのコンパイラ等を使用してロード・モジュールを作成する。installerは引き数として対象コンピュータ名（Paragon、SP2、SR2201、VPP500、VPP300、Monte-4、SX-4、T90）を与えて起動する。Fig. 4.2にinstallerの起動方法を示す。installerを起動すると、ktoolとそれに続き、VPP300およびVPP500ではxtool、Paragonではptoolのインストールが実行される。

インストール後のkpxのファイル・システムは、ツール名の基本ディレクトリと、ツール名の最初の１文字にdoを付けたユーザ・テスト用ディレクトリから構成される。installerを実行した後のkpxファイル・システムをFig. 4.3に示す。

4.2 実行

kpx利用の補助と、kpxの理解を助ける意味で、ユーザが簡単に利用できる基本的な実行シェル・スクリプトを用意した。実行シェル・スクリプトはktool用、ptool用、xtool用の三種類がある。

4.2.1 ktool

ディフォルトのシステムでは、Paragon、SP2、SR2201、VPP500、VPP300、Monte-4、SX-4、T90について経過時間が測定できるようにセッティングされている。以下にktoolの実行手順を

T90について経過時間が測定できるようにセッティングされている。以下にktoolの実行手順を示す。

1．ユーザ・ソースと、そのコンパイルと実行に必要な資源をkpx/kdoにコピーする。
　※メイン・ルーチンは一本で、ソース・ファイル名は*.fを想定している。
2．ソース*.fをヘッダを付けて１本にまとめ、計測元ソース@@test.fffとする。
3．ktoolを実行してソースに計測ルーチンを挿入し、新たな計測用ソース@@f22.fを作成する。
4．@@f22.fをコンパイル後、実行すると、計測結果のファイル@@k_tool.f66が出力される。

以上であるが、実行時間を短縮するために、まず、実行回数のみの計測を行い、次に実行回数による制限付きの時間計測を行うと、作業を効率的に終了させることができる場合がある。その場合、上記実行手順３から４までを２度繰り返すことになるが、このようなktoolの実行方法について、上記実行手順２以降の、標準的な会話型での実行を想定したシェル・スクリプトkgoを用意した。Fig. 4.4にkgoの内容と利用上の注意を示す。

4.2.2　ptool

ptoolは、Paragonのnxライブラリにより並列化されたプログラムの、ノード間通信等の並列化オーバヘッドを測定するツールである。以下にptoolの実行手順を示す。

1．ユーザ・ソースと、そのコンパイルと実行に必要な資源をkpx/pdoにコピーする。
　※メイン・ルーチンは一本で、ソース・ファイル名は*.fを想定している。
2．ソース*.fをヘッダを付けて１本にまとめ、計測元ソース@@test.fffとする。
3．ptoolを実行してソースに計測ルーチンを挿入し、新たな計測用ソース@@f22.fを作成する。
4．@@f22.fをコンパイル後、実行すると、計測結果のファイル@@p_tool.f66が出力される。

ptoolの実行方法について、上記実行手順２以降の、標準的な会話型での実行を想定したシェル・スクリプトpgoを用意した。Fig. 4.5にpgoの内容と利用上の注意を示す。

4.2.3　xtool

xtoolは、VPPの並列化ディレクティブにより並列化されたプログラムの、PE間通信等の並列化オーバヘッドを測定するツールである。以下にxtoolの実行手順を示す。

1．ユーザ・ソースと、そのコンパイルと実行に必要な資源をkpx/xdoにコピーする。
　※メイン・ルーチンは一本で、ソース・ファイル名は*.fを想定している。
2．ソース*.fをヘッダを付けて１本にまとめ、計測元ソース@@test.fffとする。
3．xtoolを実行してソースに計測ルーチンを挿入し、新たな計測用ソース@@f22.fを作成する。

xtoolの実行方法について、上記実行手順２以降の、標準的な実行を想定したシェル・スクリプトxgoを用意した。Fig. 4.6にxgoの内容と利用上の注意を示す。

4.3 利用補助ファイル

kpxはフリーウェアである。kpxがftp等により遠隔地に取り込まれた場合、本報告書あるいは作成した「kpx利用手引き」が行き届かない場合もありうる。確実にドキュメントを付けるにはドキュメントをファイルにしてtarファイルに含めることである。このような発想から、kpxでは、利用手引きとして「README」、設計思想として「DESIGN」、プログラム構造を示す「K_TREE」を用意した。さらに制御データのパラメーター一覧をヘルプ・ファイルとして添付した。Fig. 4.7、Fig. 4.8、Fig. 4.9にそれぞれ「README」「DESIGN」「K_TREE」の内容を示す。また、Fig. 4.10、Fig. 4.11、Fig. 4.12にそれぞれktool、ptool、xtoolのヘルプ・ファイルの内容を示す。

```
$ cd
$ ftp  ccsemfa
Connected to ccsemfa.
220 ccsemfa FTP server (UNIX(r) System V Release 4.0) ready.
Name (133.53.188.11:j007): j007                    idがj007の場合
Password: xxxxxx                                   パスワードがxxxxxxの場合
230 User j007 logged in.
ftp>cd /document/kpx
ftp>bi
200 Type set to I.
ftp>get  kpx.tar
local: kpx.tar remote: kpx.tar
200 PORT command successful.
150 ASCII data connection for kpx.tar (133.53.188.11,1324).
226 Transfer complete.
879246 bytes sent in 1.8 seconds (486.24 Kbytes/s)
ftp>bye
221 Goodbye.
```

Fig. 4.1 Example for getting kpx

```
$ tar  xvof  kpx.tar
x README, 3860 bytes, 8 tape blocks
x installer, 8976 bytes, 18 tape blocks
x tool.tar, 917504 bytes, 1792 tape blocks
$ csh  installer  VPP300                           対象システムがVPP300の場合
```

Fig. 4.2 Example for installing kpx

```
$ ls -l kpx/ktool                                          ktool基本ディレクトリ
total 4550
-rwx------   1 j9364    g0188         72 Sep  4 16:25 CatK              catシェル・スクリプト
-rw-r--r--   1 j9364    g0188       3255 Sep  4 16:25 k_tool.hlp        ktoolヘルプ
-rw-r--r--   1 j9364    g0188         39 Sep  4 16:25 k_tool1.dat       ktool制御データ実行回数計測用
-rw-r--r--   1 j9364    g0188         81 Sep  4 16:25 k_tool2.dat       ktool制御データ実行時間計測用
-rw-------   1 j9364    g0188       1150 Sep  4 14:24 kgo               ktool利用シェル・スクリプト
-rwxr-xr-x   1 j9364    g0188    2222100 Sep  4 16:26 ktool             ktoolロード・モジュール
-rw-r--r--   1 j9364    g0188      49738 Sep  4 16:26 myclock.dat       ktoolシステム用データCPU時間計測用
-rw-r--r--   1 j9364    g0188      49816 Sep  4 16:26 mysystem.dat      ktoolシステム用データ経過時間計測用
$ ls -l kpx/kdo                                            ktoolユーザ・テスト用ディレクトリ
total 199
-rw-------   1 j9364    g0188       1150 Sep  4 16:26 kgo               ktool利用シェル・スクリプト
-rw-r--r--   1 j9364    g0188      49738 Sep  4 16:26 myclock.dat       ktoolシステム用データCPU時間計測用
-rw-r--r--   1 j9364    g0188      49816 Sep  4 16:26 mysystem.dat      ktoolシステム用データ経過時間計測用
```

Paragonではptool関連ファイルが追加作成されます。

```
Paragon$ ls -l kpx/ptool                                   ptool基本ディレクトリ
total 1762
-rwx------   1 j9364    grp05         72 Sep  3 22:30 CatK              catシェル・スクリプト
-rw-------   1 j9364    grp05      19516 Sep  3 22:30 p_system.dat      ptoolシステム用データ
-rw-------   1 j9364    grp05        408 Sep  3 19:36 p_tool.dat        ptool制御データ
-rw-------   1 j9364    grp05       1956 Sep  3 22:30 p_tool.hlp        ptoolヘルプ
-rw-------   1 j9364    grp05        302 Sep  3 21:06 pgo               ptool利用シェル・スクリプト
-rwx------   1 j9364    grp05     862311 Sep  3 22:33 ptool             ptoolロード・モジュール
Paragon$ ls -l kpx/pdo                                     ptoolユーザ・テスト用ディレクトリ
total 42
-rw-------   1 j9364    grp05      19516 Sep  3 22:33 p_system.dat      ptoolシステム用データ
-rw-------   1 j9364    grp05        316 Sep  4 11:08 pgo               ptool利用シェル・スクリプト
```

VPPではxtool関連ファイルが追加作成されます。

```
VPP$ ls -l kpx/xtool                                       xtool基本ディレクトリ
total 6208
-rwx------   1 j9364    g0188         72 Sep  4 16:26 CatK              catシェル・スクリプト
-rw-r--r--   1 j9364    g0188      21163 Sep  4 16:26 x_system.dat      xtoolシステム用データ
-rw-------   1 j9364    g0188        253 Sep  3 19:36 x_tool.dat        xtool制御データ
-rw-r--r--   1 j9364    g0188       2048 Sep  4 16:26 x_tool.hlp        xtoolヘルプ
-rw-------   1 j9364    g0188        735 Sep  4 11:06 xgo               xtool利用シェル・スクリプト
-rwxr-xr-x   1 j9364    g0188    2957916 Sep  4 16:29 xtool             xtoolロード・モジュール
VPP$ ls -l kpx/xdo                                         xtoolユーザ・テスト用ディレクトリ
total 128
-rw-r--r--   1 j9364    g0188      21163 Sep  4 16:29 x_system.dat      xtoolシステム用データ
-rw-------   1 j9364    g0188        735 Sep  4 16:29 xgo               xtool利用シェル・スクリプト
```

Fig. 4.3 kpx file system

```
$ cat kpx/kdo/kgo                                                    特に必要なし
#!/bin/csh -f                                                        特に必要なし
cd $HOME/kpx/kdo
#--------+---------+---------+---------+---------+---------+---------+
rm -f @@* CPU_*       # 1st execution if @@k_tool.f99 is not exists.  @@k_tool.f99が無ければ一回目の実行
../ktool/CatK *.f > @@test.fff                                       計測元ソースを作成
##### 1st for counting execution times
cp ../ktool/k_tool1.dat k_tool.dat                                   k_tool.datで実行回数測定指示
../ktool/ktool                                                       計測用ソースを作成.....注1
rm @@f*.wrk                                                          不要なファイルの消去
######################################################################
##### Probably you have to change next two lines to fit.
##### Which compiler will you use? Dose your program need input?
fortran @@f22.f                                                      コンパイル.....注2
./a.out < input                                                      実行回数測定.....注3
######################################################################
rm @@f22.f @@f28.inf @@k_tool.f66                                    不要なファイルの消去
###### 2nd for measuring time
cp ../ktool/k_tool2.dat k_tool.dat                                   k_tool.datで実行時間測定指示
../ktool/ktool                                                       計測用ソースを作成.....注1
rm @@f*.wrk                                                          不要なファイルの消去
######################################################################
##### Probably you have to change next two lines to fit.
##### Which compiler will you use? Dose your program need input?
fortran @@f22.f                                                      コンパイル.....注2
./a.out < input                                                      実行時間測定.....注3
######################################################################
```

※ 注1　基本的に実行は全て対象システムで行いますが、VPP500の場合はGSPで実行します。
注2　ターゲット依存ですので書換が必要です。VPP500のコンパイルはGSPで行います。
注3　ユーザ・プログラム依存ですので書換が必要です。VPP500の場合はサブミットします。
実行時に機器番号66と99を使用します。

Fig. 4.4 Execution shell script for ktool

```
Paragon$ cat kpx/pdo/pgo
#!/bin/csh -f                                          特に必要なし
cd $HOME/kpx/pdo                                       特に必要なし
#--------+---------+---------+---------+---------+---------+
rm -f @@* CPU_*                                        不要なファイルの消去
../ptool/CatK *.f > @@test.fff                         計測元ソースを作成
cp ../ptool/p_tool.dat p_tool.dat                      p_tool.dat制御データ・ファイル設定
../ptool/ptool                                         計測用ソースを作成
rm @@f*.wrk                                            不要なファイルの消去
##### Change next two lines to fit.
if77 -nx -O3 @@f22.f                                   コンパイル.....注1
./a.out -plk -pn ps8 < input                           オーバーヘッド測定.....注2
```

※ 注1 オプションの変更等必要に応じて書換が必要です。
注2 ユーザ・プログラム依存ですので書換が必要です。
実行時に機器番号66と99を使用します。

Fig. 4.5 Execution shell script for ptool

```
VPP$ cat kpx/xdo/xgo
#!/bin/csh -f                                          特に必要なし
cd $HOME/kpx/xdo                                       特に必要なし
#--------+---------+---------+---------+---------+---------+
¥rm -f @@* XINC_*                                      不要なファイルの消去
../xtool/CatK *.f > @@test.fff                         計測元ソースを作成
¥cp ../xtool/x_tool.dat x_tool.dat                     p_tool.dat制御データ・ファイル設定
../xtool/xtool                                         計測用ソースを作成
¥rm @@f*.wrk                                           不要なファイルの消去
##### Change next line to fit.
frt -Wx @@f22.f                                        コンパイル.....注1
echo ''
echo    '==== Would you like to continue ? ===='
echo -n '==== Reply  y:continue  else:quit ===='       サブミット前の確認
if ($< == y) then
    ##### Change next line to fit.
    qsub -q fu004nlm -lP 4 go                          オーバーヘッド測定.....注2
endif
```

※ 注1 コンパイラや、オプションの変更等必要に応じて書換が必要です。
注2 必要に応じて書換が必要です。ユーザ・プログラム依存のgoを作成します。
実行時に機器番号66と99を使用します。

Fig. 4.6 Execution shell script for xtool

```
=== The kpx ===

    The kpx is a program analyzer, produced by Japan Atomic Energy Research Institute for promoting parallel
processing. This is a free software. The kpx consists of the following three tools.

        ktool : counting execution times and measuring execution cost,
        ptool : measuring overhead time only for Paragon,
    and xtool : measuring overhead time only for VPP.

    These tools are designed to work for any FORTRAN cord on any UNIX computer, is confirmed to work well after
testing on Paragon, SP2, SR2201, VPP500, VPP300, Monte-4, SX-4 and T90.

=== Release set ===

    kpx.tar <README:comment, installer:c-shell script, tool.tar:tool resource>

=== Install ===

        $ cd
        $ tar xvof kpx.tar
        $ csh installer "system"
        system : [Paragon, SP2, SR2201, VPP500, VPP300, Monte-4, SX-4 or T90]

    You can modify "installer" when applying in other systems.

=== Files and control flow ===

--- ktool ---
 Input:ktool    Execution   output:ktool  Comp. and exec.        Output
k_tool.dat---+           +---@@f66.msg                    +---@@k_tool.f66
@@test.fff---+---ktool---+---@@f22.f---+---FORTRAM & Run---+---@@k_tool.f99
Include files+           +---CPU_CTR---+                    +---Output of program
          Data file for user program---+

--- ptool ---
 Input:ktool    Execution   output:ktool  Comp. and exec.        Output
p_tool.dat---+           +---@@f66.msg                    +---@@p_tool.f66
@@test.fff---+---ptool---+---@@f22.f---+---FORTRAM & Run---+---@@p_tool.f99
Include files+           +---CPU_CTR---+                    +---Output of program
          Data file for user program---+
```

Fig. 4.7 Contents of file README (to be continued)

```
── xtool ──
 Input:ktool    Execution   output:ktool  Comp. and exec.        Output
x_tool.dat---+           +---@@f66.msg                     +---@@x_tool.f66
@@test.fff---+---xtool---+---@@f22.f---+---FORTRAM & Run---+---@@x_tool.f99
Include files+           +---XINC_GLO--+                   +---Output of program
                         +---XINC_LOC--+
          Data file for user program--+

k_tool.dat/p_tool.dat/x_tool.dat   : Control data file of ktool/ptool/xtool
@@test.fff                         : Original user source
@@f66.msg                          : message file
@@f22.f                            : Modified user source
CPU_CTR/XINC_GLO/XINC_LOC          : Include file of (ktool:ptool)/xtool/xtool
@@k_tool.f66                       : Formatted Result
@@k_tool.f99                       : Unformatted Result

== Execution ==

    Copy program "source files", "include files" and "input files" into /kpx/kdo before compiling and execution.
You have to modify c-shell script before execution according to FORTRAN command of your computer. After then,
input the following directive.

        $ csh kgo
        $ csh pgo : only for Paragon
        $ csh xgo : only for VPP

== Directories and files ==

$ ls -l kpx/ktool                    ktool directory
total 4550
-rwx------   1 j9364    g0188         72 Sep  4 16:25 CatK :cat shell
-rw-r--r--   1 j9364    g0188       3255 Sep  4 16:25 k_tool.hlp :ktool help
-rw-r--r--   1 j9364    g0188         39 Sep  4 16:25 k_tool1.dat :ktool control data for counting
-rw-r--r--   1 j9364    g0188         81 Sep  4 16:25 k_tool2.dat :ktool control data for measuring
-rw-------   1 j9364    g0188       1150 Sep  4 14:24 kgo :ktool execution shell
-rwxr-xr-x   1 j9364    g0188    2222100 Sep  4 16:26 ktool :ktool load module
-rw-r--r--   1 j9364    g0188      49738 Sep  4 16:26 myclock.dat :ktool system data for CPU time
-rw-r--r--   1 j9364    g0188      49816 Sep  4 16:26 mysystem.dat :ktool system data for elapse time
$ ls -l kpx/kdo                      ktool user directory
total 199
-rw-------   1 j9364    g0188       1150 Sep  4 16:26 kgo :ktool execution shell
-rw-r--r--   1 j9364    g0188      49738 Sep  4 16:26 myclock.dat :ktool system data for CPU time
-rw-r--r--   1 j9364    g0188      49816 Sep  4 16:26 mysystem.dat :ktool system data for elapse time
```

Fig. 4.7 Contents of file README (to be continued)

```
For the Paragon.
Paragon$ ls -l kpx/ptool                    ptool directory
total 1762
-rwx------   1 j9364    grp05         72 Sep  3 22:30 CatK :cat shell
-rw-------   1 j9364    grp05      19516 Sep  3 22:30 p_system.dat :ptool system data
-rw-------   1 j9364    grp05        408 Sep  3 19:36 p_tool.dat :ptool control data
-rw-------   1 j9364    grp05       1956 Sep  3 22:30 p_tool.hlp :ptool help
-rw-------   1 j9364    grp05        302 Sep  3 21:06 pgo :ptool execution shell
-rwx------   1 j9364    grp05     862311 Sep  3 22:33 ptool :ptool load module
Paragon$ ls -l kpx/pdo                      ptool user directory
total 42
-rw-------   1 j9364    grp05      19516 Sep  3 22:33 p_system.dat :ptool system data
-rw-------   1 j9364    grp05        316 Sep  4 11:08 pgo :ptool execution shell

For the VPP.
VPP$ ls -l kpx/xtool                        xtool directory
total 6208
-rwx------   1 j9364    g0188         72 Sep  4 16:26 CatK :cat shell
-rw-r--r--   1 j9364    g0188      21163 Sep  4 16:26 x_system.dat :xtool system data
-rw-------   1 j9364    g0188        253 Sep  3 19:36 x_tool.dat :xtool control data
-rw-r--r--   1 j9364    g0188       2048 Sep  4 16:26 x_tool.hlp :xtool help
-rw-------   1 j9364    g0188        735 Sep  4 11:06 xgo :xtool execution shell
-rwxr-xr-x   1 j9364    g0188    2957916 Sep  4 16:29 xtool :xtool load module
VPP$ ls -l kpx/xdo                          xtool user directory
total 128
-rw-r--r--   1 j9364    g0188      21163 Sep  4 16:29 x_system.dat :xtool system data
-rw-------   1 j9364    g0188        735 Sep  4 16:29 xgo :xtool execution shell
```

Fig. 4.7 Contents of file README (continued)

```
*COMDECK  README
c
c Program comment
c
c <<< A. Tree structure of the tool(K_TOOL, P_TOOL, X_TOOL) >>>
c
c  1. Tree Outline
c
c   1) K_TOOL
c
c          0         1         2         3         4
c       ------    -------   -------   -------   -------
c       k_tool  -*-input----*-inital
c                *-pass0
c                *-pass1----*-pass1sub-*-class
c                |                     *-incinc---*-class
c                *-pass2
c                *-pass3----*-onesub1--*-arystn
c                 |                     *-donest
c                 |                     *-newnest
c                 |                     *-insentry
c                 |                     *-wkmerge
c                 |                     *-makesoc
c                 |                     *-pass1sub-*-....
c                 |                     *-insnest -*-insdo
c                 |                     *-insiou
c                *-pass4
c
c   2) P_TOOL
c
c          0         1         2         3         4
c       ------    -------   -------   -------   -------
c
c       k_tool---*-input----*-inital
c                *-pass0
c                *-pass1----*-pass1sub-*-class
c                |                     *-incinc---*-class
c                *-pass2
c                *-pass3----*-onesub1--*-inslib
c                 |                     *-makesoc
c                *-pass4
c
```

Fig. 4.8 Contents of file DESIGN (to be continued)

```
c   3) X_TOOL
c
c          0          1          2          3          4
c       ---------  ---------  ---------  ---------  ---------
c       k_tool---*-input-----*-inital
c                *-pass0
c                *-pass1-----*-pass1sub-*-class
c                |                      *-xocl_cla
c                |                      *-incinc---*-class
c                *-pass2-----*-xocl_ps2
c                *-pass3-----*-xocl_one-*-lastdecl
c                 |                     *-xocl_STM
c                 |                     *-xocl_Get
c                 |                     *-wkmerge
c                 |                     *-makesoc
c                *-pass4
c
c  2. Full tree structure
c
c    Please see k_tree.inc/p_tree.inc/x_tree.inc
c
c <<< B. Program cross reference >>>
c
c    Please see k_tree.inc/p_tree.inc/x_tree.inc
c
c <<< C. Program brief comment >>>
c
c  apostr       : Searches a pair of apostrophes
c  arystn       : Get array name, statement no., and 1st executable st.
c  blkcom       : Block data
c  blkpack      : Erase blank, comment column(!), and convert low to up.
c  class        : Classify fortran statements (68 classes)
c  cmtdir       : Check comment line (vector directive)
c  cmtflg       : Check comment line.
c  cmtput       : Get comment line from working file, and put those to
c                 correction file.
c  cmtset       : Put comment line to working file.
c                 Move comment line(s) for vector directive
c  cpu_mse      : Get CPU/Elapsed time on tool processing
c  deadcopy     : Copy just 1 program to whole source (from u21 to u22)
c  decd         : Decode (convert character string to integer binary)
c  donest       : Get do loop block nest
c  doparam      : Get do parameters
c  eraseb       : Erase blank column, and get new string length
c  erasedo      : Check to erase a do loop?
c  eraseent     : Check to erase a entry?
c  eraseio      : Check to erase a I/O processing?
```

Fig. 4.8 Contents of file DESIGN (to be continued)

```
c  error      : Outputs error message, and terminates program execution
c               if fatal error.
c  ersblk     : Erase blank, and comment column(!)
c  fndstr     : Compares a sub string in string to the contents of sub string
c               without blank
c  fndxxx     : Compares a sub string in string to the contents of sub string
c  getlstm    : Get one statement from unit u21.
c  getf99     : Get restart data, if it is exist.
c  getnext    : Get next string
c  getu15     : Get one line from unit u15
c  getu21     : Get one line from unit u21
c  gi03       : Get one line from unit u03
c  gi04       : Get one line from unit u04
c  gi12       : Get one line from unit u12
c  gi13       : Get one line from unit u13
c  ifstm      : Get location of logical/arithmetic if statement
c  incinc     : Get include file content(s) (include nest)
c  inital     : Initialize program by default value
c  input      : Input tool control data
c  insdo      : Make correction data for do loop timer
c  insentry   : Make correction data for "entry" timer
c  inslib     : Make correction data for measure parallel system library
c  insnest    : Control one do loop block nest
c  insiou     : Make correction data for I/O processing timer
c  iouchk     : Check to analysis I/O st., then get I/O logical unit
c  keywrd     : Performs the keyword classification and determines the
c               type of fortran statements
c  k_tool     : Main control (p_tool/x_tool)
c  lastdecl   : Get last line no. for declaration in a program
c  lnblnk     : Count last non-blank character location at string
c  lookup     : Searches the variable name table whether it is already
c               tabulated within the table of the variable name.
c  low2up     : Convert lower character to upper character
c  makesoc    : Make new source program for timer
c  moveint    : Move data
c  newlabel   : Get new label(statement number)
c  newnest    : Make new do loop terminal for co-terminal do loop
c  nfblnk     : Count non first blank character location at string
c  number     : Get constant value from string, convert to binary
c               if it is numeric.
c  onesub1    : Just one program analyze and correct source program
c  openinc    : Open include file
c  operat     : Performs the matching test for the logical term,
c               relational  and logical operators. matching test begins
c               at string.
c  outinf     : Output source program information into the source program
c  outu31     : Output to unit u31.
```

Fig. 4.8 Contents of file DESIGN (to be continued)

```
c  pass0        : Get whole file lines (file i10—>i01).
c                 Set file sequence and line no. in the file.
c  pass1        : Get whole program statement(file i01-->i04)
c  pass1sub     : Analyze a program
c  pass2        : Get program name and entry statement entry name
c  pass3        : Analysis program statement control
c  pass4        : Add system library routines and include file
c  pi03         : Put line to i03 file
c  pi04         : Put line to i04 file
c  pi11         : Put line to i11 file
c  pi12         : Put line to i12 file
c  pi13         : Put line to i13 file
c  plshi        : Make polish notation
c  put1stm      : Put one statement to unit u31.
c  putu31       : Put line to u31 file
c  separate     : Separate file by file delimiter.
c                 whole source program to 1 program/1 file.
c  sepmake      : Get old running data for restart
c  setpar       : Evaluate parameter statement
c  smlparam     : Too small area to analyze, and execution deleted.
c                 Print out to change include parameter information.
c  socinc       : Make source/include line with change word
c  sortchr      : Descending sort character string (change just index)
c  sortint      : Ascending sort for integer data
c  subnam       : Get sub program name and main program
c  tab2blk      : Convert tab code to blanks
c  token        : Performs the token analysis and classifies the tokens
c                 into 22 classes.
c  up2low       : Convert upper character to lower character
c  whatio       : Check I/O statement to measure (for timer)
c  wkmerge      : Merge correction data new do loop and "entry" timer
c  xblksp       : Blank packing
c  xevic2       : Analyze expression
c  xidecd       : Decode (character to binary)
c  xindex       : Get location of character string
c  xocl_cla     : Classify XOCL statement
c  xocl_get     : Get original !xocl line
c  xocl_one     : Just one program analyze and correct source program
c                 for XOCL program
c  xocl_ps2     : Get program name and entry statement entry name
c                 for XOCL program
c  xocl_stm     : Analyze XOCL line
c
```

Fig. 4.8 Contents of file DESIGN (to be continued)

```
c <<< D. Files >>>
c
c  Number is FORTRAN unit number.
c  k_tool
c
c     10 ---> 01 -*-> 02 ---> 03 ---> 04 -*-> 16 ---> 13 ---------->*(2)
c                 *-> 21 --->*(1)          *-> 12 ---> 15 ---> 14 --->*(3)
c                                          -*-> 41
c
c  *(2) -+
c        +-> 15 -+
c  *(3) -+       |
c                +-> 31 ---> 02 ---> 03 ---> 34 -*-> 16
c      *(1) -21 -+                               *-> 12 ---> 15 --->*(4)
c
c  *(4) -+
c        +-> 22
c    34--+
c
c  1) pass0     : 10 ---> 01
c  2) pass1sub  : 01 ---> 02 ---> 03 ---> 04
c                 31 ---> 02 ---> 03 ---> 34
c  3) pass2     : 04 ---> 41
c  4) pass3     : 04 ---> 16
c  5) onesub1   : 12 ---> 15 ---> 14
c                 15+34 ---> 22
c  6) donest    : 16 ---> 12
c  7) insdo     : 16 ---> 13
c  8) insentry  : 16 ---> 13
c  9) wkmerge   : 13+14 ---> 15
c 10) pass4     :     ---> 22
c
c File contents : See iocom.inc
c
```

Fig. 4.8 Contents of file DESIGN (continued)

```
*COMDECK  K_TREE
c
c This is K_TOOL programming information.
c
c Fortran source  files : 81 (about 11000 lines)
c Fortran include files : 18 (about  1400 lines) without this comment
c
c     program reference    ( user program )
c
```

name	callers	callees
apostr (subr)	class token (subr) 1 (subr) 1	********
arystn	onesub1	decd getnxt gi04 lnblnk smlparam sortint token
blkcom (blkd)	********	********
blkpack	incinc pass1sub	error ichar index len mod
class	incinc insdo makesoc pass1sub	apostr keywrd
cmtdir	cmtset makesoc	********
cmtflg	getlstm incinc pass1sub	********
cmtput	makesoc	lnblnk
cmtset	makesoc	cmtdir getu21
cpu_mse	k_tool	********
deadcopy	onesub1 pass3	lnblnk low2up tab2blk
decd	arystn donest input onesub1 separate	index len
donest	onesub1	decd gi04 lnblnk nfblnk pi11 pi12 smlparam token whatio

Fig. 4.9 Contents of file K_TREE (to be continued)

doparam	insdo	********	
eraseb	input	len	
erasedo	insnest	********	
eraseent	pass3	********	
eraseio	insiou	********	
error	blkpack ersblk token	len	
ersblk	incinc pass1sub	error len mod	
fndstr	keywrd	********	
fndxxx	operat token	********	
getlstm	makesoc	cmtflg getu21 index outu31	
getf99	input	dble lnblnk smlparam	
getnxt	arystn	index len	
getu15	makesoc	gi13	
getu21	cmtset getlstm makesoc	lnblnk tab2blk	
gi03	incinc pass1sub	min	
gi04	arystn donest incinc insentry insiou outinf pass2 pass3	min	
gi12	insdo newnest onesub1	min	
gi13	getu15 wkmerge	min	
ifstm	makesoc	index	
incinc	pass1sub	blkpack class cmtflg ersblk gi03 gi04 len lnblnk low2up openinc pi03 pi04 tab2blk	

Fig. 4.9 Contents of file K_TREE (to be continued)

inital	input	lnblnk	
input	k_tool	char	decd eraseb
		getf99	index inital
		lnblnk	low2up smlparam
		tab2blk	
insdo	insnest	class	doparam gi12
		lnblnk	pi13 token
insentry	onesub1	gi04	lnblnk pi13
insiou	onesub1	eraseio	gi04 iouchk
		lnblnk	pi13
insnest	onesub1	erasedo	insdo
iouchk	insiou	whatio	
k_tool	********	cpu_mse	input lnblnk
		max	pass0 pass1
		pass2	pass3 pass4
		separate	
keywrd	class	fndstr	
lnblnk	arystn cmtput deadcopy	len	
	donest getf99 getu21		
	incinc inital input		
	insdo insentry insiou		
	k_tool makesoc onesub1		
	outinf pass0 pass1		
	pass1sub pass2 pass3		
	pass4 put1stm putu31		
	separate sepmake socinc		
low2up	deadcopy incinc input	index	len min
	makesoc pass1sub separate		
makesoc	onesub1	class	cmtdir cmtput
		cmtset	get1stm getu15
		getu21	ifstm index
		lnblnk	low2up outu31
		put1stm	putu31
moveint	newlabel	abs	

Fig. 4.9 Contents of file K_TREE (to be continued)

c| newlabel	newnest	moveint	smlparam
c| newnest	onesub1	gi12	newlabel pi13
c| nfblnk	donest pass2	len	
c| number	token	operat	
c| onesub1	pass3	arystn	deadcopy decd
c|		donest	gi12 insentry
c|		insiou	insnest lnblnk
c|		makesoc	newnest pass1sub
c|		pi12	smlparam wkmerge
c| openinc	incinc	len	
c| operat	number	fndxxx	
c| outinf	pass3	gi04	lnblnk tab2blk
c|		whatio	
c| outu31	get1stm makesoc put1stm	********	
c|	putu31		
c| pass0	k_tool	lnblnk	max mod
c|		smlparam	
c| pass1	k_tool	lnblnk	pass1sub
c| pass1sub	onesub1 pass1	blkpack	class cmtflg
c|		ersblk	gi03 incinc
c|		lnblnk	low2up mod
c|		pi03	pi04 tab2blk
c| pass2	k_tool	gi04	index lnblnk
c|		nfblnk	subnam
c| pass3	k_tool	deadcopy	eraseent gi04
c|		lnblnk	onesub1 outinf
c|		pi04	
c| pass4	k_tool	lnblnk	max min
c|		socinc	tab2blk
c| pi03	incinc pass1sub	min	
c| pi04	incinc pass1sub pass3	min	

Fig. 4.9 Contents of file K_TREE (to be continued)

```
c+---------+-------------------------------+-------------------------------+
c| pil1    | donest                        | len      min                  |
c+---------+-------------------------------+-------------------------------+
c| pil2    | donest   onesub1              | min                           |
c+---------+-------------------------------+-------------------------------+
c| pil3    | insdo    insentry insiou      | min                           |
c|         | newnest  wkmerge              |                               |
c+---------+-------------------------------+-------------------------------+
c| putlstm | makesoc                       | lnblnk   min      outu31      |
c+---------+-------------------------------+-------------------------------+
c| putu31  | makesoc                       | lnblnk   min      outu31      |
c+---------+-------------------------------+-------------------------------+
c| separate| k_tool                        | decd     lnblnk   low2up      |
c|         |                               | sepmake  tab2blk  up2low      |
c+---------+-------------------------------+-------------------------------+
c| sepmake | separate                      | lnblnk   smlparam sortchr     |
c+---------+-------------------------------+-------------------------------+
c| smlparam| arystn   donest   getf99      | ********                      |
c|         | input    newlabel onesub1     |                               |
c|         | pass0    sepmake  token       |                               |
c+---------+-------------------------------+-------------------------------+
c| socinc  | pass4                         | index    lnblnk   min         |
c+---------+-------------------------------+-------------------------------+
c| sortchr | sepmake                       | ********                      |
c+---------+-------------------------------+-------------------------------+
c| sortint | arystn                        | ********                      |
c+---------+-------------------------------+-------------------------------+
c| subnam  | pass2                         | index                         |
c+---------+-------------------------------+-------------------------------+
c| tab2blk | deadcopy getu21   incinc      | len                           |
c|         | input    outinf   passlsub    |                               |
c|         | pass4    separate             |                               |
c+---------+-------------------------------+-------------------------------+
c| token   | arystn   donest   insdo       | apostr   error    fndxxx      |
c|         |                               | index    number   smlparam    |
c+---------+-------------------------------+-------------------------------+
c| up2low  | separate                      | index    len      min         |
c+---------+-------------------------------+-------------------------------+
c| whatio  | donest   iouchk   outinf      | ********                      |
c+---------+-------------------------------+-------------------------------+
c| wkmerge | onesub1                       | gi13     pi13                 |
c+---------+-------------------------------+-------------------------------+
c
```

Fig. 4.9 Contents of file K_TREE (to be continued)

```
c     program reference    ( intrinsic function )
c
c+-----------+-------------------------------------------------------+
c| name      |                    callers                            |
c+-----------+-------------------------------------------------------+
c| abs       | moveint                                               |
c+-----------+-------------------------------------------------------+
c| char      | input                                                 |
c+-----------+-------------------------------------------------------+
c| dble      | getf99                                                |
c+-----------+-------------------------------------------------------+
c| ichar     | blkpack                                               |
c+-----------+-------------------------------------------------------+
c| index     | blkpack  decd     get1stm  getnxt   ifstm    input     |
c|           | low2up   makesoc  pass2    socinc   subnam   token     |
c|           | up2low                                                |
c+-----------+-------------------------------------------------------+
c| len       | blkpack  decd     eraseb   error    ersblk   getnxt    |
c|           | incinc   lnblnk   low2up   nfblnk   openinc  pil1     |
c|           | tab2blk  up2low                                       |
c+-----------+-------------------------------------------------------+
c| max       | k_tool   pass0    pass4                               |
c+-----------+-------------------------------------------------------+
c| min       | gi03     gi04     gi12     gi13     low2up   pass4     |
c|           | pi03     pi04     pi11     pi12     pi13     put1stm   |
c|           | putu31   socinc   up2low                              |
c+-----------+-------------------------------------------------------+
c| mod       | blkpack  ersblk   pass0    pass1sub                   |
c+-----------+-------------------------------------------------------+
c
```

Fig. 4.9 Contents of file K_TREE (to be continued)

```
c     tree structure
c
c
c0001: k_tool
c      |
c0002: |-input
c      | |
c0003: | |-lnblnk
c      | | |
c0004: | | +-len*
c      | |
c0005: | |-char*
c      | |
c0006: | |-inital
c      | | |
c0007: | | +-lnblnk --> 0003
c      | |
c0008: | |-tab2blk
c      | | |
c0009: | | +-len*
c      | |
c0010: | |-index*
c      | |
c0011: | |-eraseb
c      | | |
c0012: | | +-len*
c      | |
c0013: | |-low2up
c      | | |
c0014: | | |-len*
c      | | |
c0015: | | |-min*
c      | | |
c0016: | | +-index*
c      | |
c0017: | |-decd
c      | | |
c0018: | | |-len*
c      | | |
c0019: | | +-index*
c      | |
c0020: | |-smlparam
c      | |
c0021: | +-getf99
c      |   |
c0022: |   |-lnblnk --> 0003
c      |   |
```

Fig. 4.9 Contents of file K_TREE (to be continued)

```
c0023: |   |-smlparam
c      |   |
c0024: |   +-dble*
c      |
c0025: |-cpu_mse
c      |
c0026: |-separate
c      | |
c0027: | |-sepmake
c      | | |
c0028: | | |-lnblnk --> 0003
c      | | |
c0029: | | |-smlparam
c      | | |
c0030: | | +-sortchr
c      | |
c0031: | |-lnblnk --> 0003
c      | |
c0032: | |-low2up --> 0013
c      | |
c0033: | |-tab2blk --> 0008
c      | |
c0034: | |-up2low
c      | | |
c0035: | | |-len*
c      | | |
c0036: | | |-min*
c      | | |
c0037: | | +-index*
c      | |
c0038: | +-decd --> 0017
c      |
c0039: |-pass0
c      | |
c0040: | |-lnblnk --> 0003
c      | |
c0041: | |-smlparam
c      | |
c0042: | |-max*
c      | |
c0043: | +-mod*
c      |
c0044: |-pass1
c      | |
c0045: | |-lnblnk --> 0003
c      | |
c0046: | +-pass1sub
```

Fig. 4.9 Contents of file K_TREE (to be continued)

```
c       |   |
c0047:  |   |-mod*
c       |   |
c0048:  |   |-lnblnk --> 0003
c       |   |
c0049:  |   |-tab2blk --> 0008
c       |   |
c0050:  |   |-low2up --> 0013
c       |   |
c0051:  |   |-blkpack
c       |   | |
c0052:  |   | |-len*
c       |   | |
c0053:  |   | |-ichar*
c       |   | |
c0054:  |   | |-error
c       |   | | |
c0055:  |   | | +-len*
c       |   | |
c0056:  |   | |-mod*
c       |   | |
c0057:  |   | +-index*
c       |   |
c0058:  |   |-cmtflg
c       |   |
c0059:  |   |-ersblk
c       |   | |
c0060:  |   | |-len*
c       |   | |
c0061:  |   | |-mod*
c       |   | |
c0062:  |   | +-error --> 0054
c       |   |
c0063:  |   |-pi03
c       |   | |
c0064:  |   | +-min*
c       |   |
c0065:  |   |-gi03
c       |   | |
c0066:  |   | +-min*
c       |   |
c0067:  |   |-class
c       |   | |
c0068:  |   | |-keywrd
c       |   | | |
c0069:  |   | | +-fndstr
c       |   | |
```

Fig. 4.9 Contents of file K_TREE (to be continued)

```
c0070: |   | +-apostr
c      |   |
c0071: |   |-incinc
c      |   | |
c0072: |   | |-len*
c      |   | |
c0073: |   | |-openinc
c      |   | | |
c0074: |   | | +-len*
c      |   | |
c0075: |   | |-lnblnk --> 0003
c      |   | |
c0076: |   | |-tab2blk --> 0008
c      |   | |
c0077: |   | |-low2up --> 0013
c      |   | |
c0078: |   | |-blkpack --> 0051
c      |   | |
c0079: |   | |-cmtflg
c      |   | |
c0080: |   | |-ersblk --> 0059
c      |   | |
c0081: |   | |-pi03 --> 0063
c      |   | |
c0082: |   | |-gi03 --> 0065
c      |   | |
c0083: |   | |-class --> 0067
c      |   | |
c0084: |   | |-pi04
c      |   | | |
c0085: |   | | +-min*
c      |   | |
c0086: |   | +-gi04
c      |   |   |
c0087: |   |   +-min*
c      |   |
c0088: |   +-pi04 --> 0084
c      |
c0089: |-pass2
c      | |
c0090: | |-gi04 --> 0086
c      | |
c0091: | |-lnblnk --> 0003
c      | |
c0092: | |-subnam
c      | | |
c0093: | | +-index*
```

Fig. 4.9 Contents of file K_TREE (to be continued)

```
c       | |
c0094:  | |-index*
c       | |
c0095:  | +-nfblnk
c       |   |
c0096:  |   +-len*
c       |
c0097:  |-pass3
c       | |
c0098:  | |-gi04 --> 0086
c       | |
c0099:  | |-pi04 --> 0084
c       | |
c0100:  | |-lnblnk --> 0003
c       | |
c0101:  | |-outinf
c       | | |
c0102:  | | |-lnblnk --> 0003
c       | | |
c0103:  | | |-tab2blk --> 0008
c       | | |
c0104:  | | |-gi04 --> 0086
c       | | |
c0105:  | | +-whatio
c       | |
c0106:  | |-eraseent
c       | |
c0107:  | |-onesub1
c       | | |
c0108:  | | |-arystn
c       | | | |
c0109:  | | | |-gi04 --> 0086
c       | | | |
c0110:  | | | |-decd --> 0017
c       | | | |
c0111:  | | | |-smlparam
c       | | | |
c0112:  | | | |-lnblnk --> 0003
c       | | | |
c0113:  | | | |-getnxt
c       | | | | |
c0114:  | | | | |-len*
c       | | | | |
c0115:  | | | | +-index*
c       | | | |
c0116:  | | | |-token
c       | | | | |
```

Fig. 4.9 Contents of file K_TREE (to be continued)

```
c0117: | | | | |-index*
c      | | | | |
c0118: | | | | |-number
c      | | | | | |
c0119: | | | | | +-operat
c      | | | | |   |
c0120: | | | | |   +-fndxxx
c      | | | | |
c0121: | | | | |-apostr
c      | | | | |
c0122: | | | | |-fndxxx
c      | | | | |
c0123: | | | | |-error --> 0054
c      | | | | |
c0124: | | | | +-smlparam
c      | | | |
c0125: | | | +-sortint
c      | | |
c0126: | | |-donest
c      | | | |
c0127: | | | |-gi04 --> 0086
c      | | | |
c0128: | | | |-lnblnk --> 0003
c      | | | |
c0129: | | | |-pi12
c      | | | | |
c0130: | | | | +-min*
c      | | | |
c0131: | | | |-token --> 0116
c      | | | |
c0132: | | | |-decd --> 0017
c      | | | |
c0133: | | | |-smlparam
c      | | | |
c0134: | | | |-pi11
c      | | | | |
c0135: | | | | |-len*
c      | | | | |
c0136: | | | | +-min*
c      | | | |
c0137: | | | |-nfblnk --> 0095
c      | | | |
c0138: | | | +-whatio
c      | | |
c0139: | | |-gi12
c      | | | |
c0140: | | | +-min*
```

Fig. 4.9 Contents of file K_TREE (to be continued)

```
c       | | |
c0141: | | |-pi12 --> 0129
c       | | |
c0142: | | |-smlparam
c       | | |
c0143: | | |-decd --> 0017
c       | | |
c0144: | | |-newnest
c       | | | |
c0145: | | | |-newlabel
c       | | | | |
c0146: | | | | |-smlparam
c       | | | | |
c0147: | | | | +-moveint
c       | | | |   |
c0148: | | | |   +-abs*
c       | | | |
c0149: | | | |-gi12 --> 0139
c       | | | |
c0150: | | | +-pi13
c       | | |   |
c0151: | | |   +-min*
c       | | |
c0152: | | |-insentry
c       | | | |
c0153: | | | |-gi04 --> 0086
c       | | | |
c0154: | | | |-lnblnk --> 0003
c       | | | |
c0155: | | | +-pi13 --> 0150
c       | | |
c0156: | | |-wkmerge
c       | | | |
c0157: | | | |-gi13
c       | | | | |
c0158: | | | | +-min*
c       | | | |
c0159: | | | +-pi13 --> 0150
c       | | |
c0160: | | |-makesoc
c       | | | |
c0161: | | | |-getu21
c       | | | | |
c0162: | | | | |-lnblnk --> 0003
c       | | | | |
c0163: | | | | +-tab2blk --> 0008
c       | | | |
```

Fig. 4.9 Contents of file K_TREE (to be continued)

```
c0164: | | | |-low2up --> 0013
c      | | | |
c0165: | | | |-lnblnk --> 0003
c      | | | |
c0166: | | | |-outu31
c      | | | |
c0167: | | | |-cmtdir
c      | | | |
c0168: | | | |-cmtset
c      | | | | |
c0169: | | | | |-getu21 --> 0161
c      | | | | |
c0170: | | | | +-cmtdir
c      | | | |
c0171: | | | |-getu15
c      | | | | |
c0172: | | | | +-gi13 --> 0157
c      | | | |
c0173: | | | |-cmtput
c      | | | | |
c0174: | | | | +-lnblnk --> 0003
c      | | | |
c0175: | | | |-putu31
c      | | | | |
c0176: | | | | |-min*
c      | | | | |
c0177: | | | | |-lnblnk --> 0003
c      | | | | |
c0178: | | | | +-outu31
c      | | | |
c0179: | | | |-getlstm
c      | | | | |
c0180: | | | | |-index*
c      | | | | |
c0181: | | | | |-outu31
c      | | | | |
c0182: | | | | |-getu21 --> 0161
c      | | | | |
c0183: | | | | +-cmtflg
c      | | | |
c0184: | | | |-index*
c      | | | |
c0185: | | | |-ifstm
c      | | | | |
c0186: | | | | +-index*
c      | | | |
c0187: | | | |-putlstm
```

Fig. 4.9 Contents of file K_TREE (to be continued)

```
c       | | | | |
c0188: | | | | |-lnblnk --> 0003
c       | | | | |
c0189: | | | | |-outu31
c       | | | | |
c0190: | | | | +-min*
c       | | | |
c0191: | | | +-class --> 0067
c       | | |
c0192: | | |-lnblnk --> 0003
c       | | |
c0193: | | |-passlsub --> 0046
c       | | |
c0194: | | |-insnest
c       | | | |
c0195: | | | |-erasedo
c       | | | |
c0196: | | | +-insdo
c       | | |   |
c0197: | | |   |-gi12 --> 0139
c       | | |   |
c0198: | | |   |-lnblnk --> 0003
c       | | |   |
c0199: | | |   |-token --> 0116
c       | | |   |
c       | | |   |
c0201: | | |   |-pi13 --> 0150
c       | | |   |
c0202: | | |   +-class --> 0067
c       | | |
c0203: | | |-insiou
c       | | | |
c0204: | | | |-gi04 --> 0086
c       | | | |
c0205: | | | |-iouchk
c       | | | | |
c0206: | | | | +-whatio
c       | | | |
c0207: | | | |-eraseio
c       | | | |
c0208: | | | |-lnblnk --> 0003
c       | | | |
c0209: | | | +-pi13 --> 0150
c       | | |
c0210: | | +-deadcopy
c       | |   |
c0211: | |   |-lnblnk --> 0003
```

Fig. 4.9 Contents of file K_TREE (to be continued)

```
c       | |   |
c0212:  | |   |-tab2blk --> 0008
c       | |   |
c0213:  | |   +-low2up --> 0013
c       | |
c0214:  | +-deadcopy --> 0210
c       |
c0215:  |-pass4
c       | |
c0216:  | |-lnblnk --> 0003
c       | |
c0217:  | |-tab2blk --> 0008
c       | |
c0218:  | |-socinc
c       | | |
c0219:  | | |-index*
c       | | |
c0220:  | | |-min*
c       | | |
c0221:  | | +-lnblnk --> 0003
c       | |
c0222:  | |-max*
c       | |
c0223:  | +-min*
c       |
c0224:  |-max*
c       |
c0225:  +-lnblnk --> 0003
c
c                          << alphabetical index >>
c
```

Fig. 4.9 Contents of file K_TREE (to be continued)

cname	type	line	name	type	line
cabs	(intr)	0148	k_tool	(prog)	0001
capostr	(subr)	0070	keywrd	(subr)	0068
carystn	(subr)	0108	len	(intr)	0004
cblkpack	(subr)	0051	lnblnk	(func)	0003
cchar	(intr)	0005	low2up	(subr)	0013
cclass	(subr)	0067	makesoc	(subr)	0160
ccmtdir	(func)	0167	max	(intr)	0042
ccmtflg	(func)	0058	min	(intr)	0015
ccmtput	(subr)	0173	mod	(intr)	0043
ccmtset	(subr)	0168	moveint	(subr)	0147
ccpu_mse	(func)	0025	newlabel	(func)	0145
cdble	(intr)	0024	newnest	(subr)	0144
cdeadcopy	(subr)	0210	nfblnk	(func)	0095
cdecd	(func)	0017	number	(subr)	0118
cdonest	(subr)	0126	onesub1	(subr)	0107
cdoparam	(subr)	0200	openinc	(subr)	0073
ceraseb	(func)	0011	operat	(subr)	0119
cerasedo	(subr)	0195	outinf	(subr)	0101
ceraseent	(subr)	0106	outu31	(subr)	0166
ceraseio	(subr)	0207	pass0	(subr)	0039
cerror	(subr)	0054	pass1	(subr)	0044
cersblk	(subr)	0059	pass1sub	(subr)	0046
cfndstr	(subr)	0069	pass2	(subr)	0089
cfndxxx	(subr)	0120	pass3	(subr)	0097
cget1stm	(subr)	0179	pass4	(subr)	0215
cgetf99	(subr)	0021	pi03	(subr)	0063
cgetnxt	(subr)	0113	pi04	(subr)	0084
cgetu15	(subr)	0171	pi11	(subr)	0134
cgetu21	(subr)	0161	pi12	(subr)	0129
cgi03	(subr)	0065	pi13	(subr)	0150
cgi04	(subr)	0086	put1stm	(subr)	0187
cgi12	(subr)	0139	putu31	(subr)	0175
cgi13	(subr)	0157	separate	(subr)	0026
cichar	(intr)	0053	sepmake	(subr)	0027
cifstm	(subr)	0185	smlparam	(subr)	0020
cincinc	(subr)	0071	socinc	(subr)	0218
cindex	(intr)	0010	sortchr	(subr)	0030
cinital	(subr)	0006	sortint	(subr)	0125
cinput	(subr)	0002	subnam	(subr)	0092
cinsdo	(subr)	0196	tab2blk	(subr)	0008
cinsentry	(subr)	0152	token	(subr)	0116
cinsiou	(subr)	0203	up2low	(subr)	0034
cinsnest	(subr)	0194	whatio	(subr)	0105
ciouchk	(subr)	0205	wkmerge	(subr)	0156

Fig. 4.9 Contents of file K_TREE (continued)

Key word	Value	Comment
ENT_ENT	ent_ent	Symbol of enter subprogram in PROGRAM unit
RET_ENT	ret_ent	Symbol of return subprogram in PROGRAM unit
ENT_DOL	ent_dol	Symbol of enter subprogram in DO LOOP unit
RET_DOL	ret_dol	Symbol of return subprogram in DO LOOP unit
CTR_DOL	ctr_dol	Symbol of count subprogram in DO LOOP unit
ENT_IOU	ent_iou	Symbol of enter subprogram in I/O unit
RET_IOU	ret_iou	Symbol of return subprogram in I/O unit
CPU_MSE	cpu_mse	Symbol of entry CPU/elapsed time subprogram(real)
CPU_MSR	cpu_msr	Symbol of return CPU/elapsed time subprogram(real)
OUT_CTR	out_ctr	Symbol of measure result output subprogram
CPU_BLK	cpu_blk	Symbol of block data subprogram
CPU_CTR	CPU_CTR	Include file name of common block
COM_CHR	com_chr	Common block name for character type data
COM_BIN	com_bin	Common block name for numerical type data
DUM_DUM	dum_dum	Symbol of proper tool overhead measure subprogram
INP_SOC	@@test.fff	Input program file name
OUT_SOC	@@f22.f	Output program file name
MYSYSTEM	mysystem.dat	Tool system data file name
OUT_U66	66	Logical unit to print out measure result
OUT_F66	@@k_tool.f66	File name to print out measure result
OPN_U66	Yes	Require file open of measure result print out
OUT_U99	99	Logical unit of measure result for restart
OUT_F99	@@k_tool.f99	File name of measure result for restart
OPN_U99	Yes	Require file open of measure result for restart
OHD_CTR	1000	No. of loop to decrease overhead of proper tool
GLOBAL_ENT	Yes	Global assignment of measure object PROGRAM unit
GLOBAL_DOL	Yes	Global assignment of measure object DO LOOP unit
GLOBAL_IOU	Yes	Global assignment of measure object I/O unit
JST_CTR	No	Assignment of measuring classification (just count)
UNSAT_DO	Yes	Measure option of unsatisfied rotation DO LOOP
ENRTY	-none-	Specified PROGRAM unit
DOLABEL	-none-	Specified DO LOOP label
DOSEQ	-none-	Specified DO LOOP appearance order in a subprogram
IOSEQ	-none-	Specified I/O appearance order in a subprogram
TABCODE	chr(9)	Assignment inner code of tab code
TABNUM	8	Assignment of column number by tab code
INC_PATH	-none-	Set of include reference path
MAX_ENT	0	Maximum times of PROGRAM when restart

Fig. 4.10 Contents of file k_tool.hlp (to be continued)

```
MAX_IOU     0               Maximum times of I/O when restart
MAX_DOL     0               Maximum times of DO LOOP when restart
MAX_MRT     0               Maximum mean times of DO LOOP rotation when restart

costtime@ ent               Specified PROGRAM unit directive in source program
costtime@ iou               Specified I/O unit directive in source program
costtime@ dol               Specified DO LOOP unit directive in source program

          k_tool.dat        Tool control data file name
```

Fig. 4.10 k_tool.hlp (continued)

Key word	Value	Comment
ENT_ENT	ent_ent	Symbol of enter subprogram in measure library
RET_ENT	ret_ent	Symbol of return subprogram in measure library
CPU_MSE	cpu_mse	Symbol of entry CPU/elapsed time subprogram(real)
INI_CTR	ini_ctr	Symbol of measure initial subprogram
OUT_CTR	out_ctr	Symbol of measure result output subprogram
CPU_CTR	CPU_CTR	Include file name of common block
COM_BIN	com_bin	Common block name for numerical type data
DUM_DUM	dum_dum	Symbol of proper tool overhead measure subprogram
INP_SOC	@@test.fff	Input program file name
OUT_SOC	@@f22.f	Output program file name
MYSYSTEM	p_system.dat	Tool system data file name
OUT_U66	66	Logical unit to print out measure result
OUT_F66	@@p_tool.f66	File name to print out measure result
OPN_U66	Yes	Require file open of measure result print out
OUT_U99	99	Logical unit of measure result for restart
OUT_F99	@@p_tool.f99	File name of measure result for restart
OPN_U99	Yes	Require file open of measure result for restart
OHD_CTR	1000	No. of loop to decrease overhead of proper tool
MAX_IDS	8192	Maximum no. of processor
SYS_SUB	-none-	Specified to measure subroutine subprogram name
SYS_FUNC	-none-	Specified to measure function subprogram name
TABCODE	chr(9)	Assignment inner code of tab code
TABNUM	8	Assignment of column number by tab code
INC_PATH	-none-	Set of include reference path
costtime@ on		Measure on directive in source program
costtime@ off		Measure off directive in source program
	p_tool.dat	Tool control data file name

Fig. 4.11 Contents of file p_tool.hlp

Key word	Value	Comment
X_INITIM	x_initm	Symbol of measure initialize subprogram
X_PRINT	x_print	Symbol of measure result output subprogram
XINC_GLO	XINC_GLO	Include file name of common block for global
XINC_LOC	XINC_LOC	Include file name for local data declaration
X_COMMON	x_common	Common block name for global data
X_LOCAL	x_local	Common block name for local data
X_GLOSET	x_gloset	Symbol of measure sub. before Parallel region
X_LOCSUM	x_locsum	Symbol of measure sub. after Parallel region
X_LOCSET	x_locset	Symbol of measure sub. before End parallel region
X_GLOSUM	x_glosum	Symbol of measure sub. after End parallel region
X_TIMSET	x_timset	Symbol of measure sub. before other XOCL statements
X_TIMSUM	x_timsum	Symbol of measure sub. after other XOCL statements
INP_SOC	@@test.fff	Input program file name
OUT_SOC	@@f22.f	Output program file name
MYSYSTEM	x_system.dat	Tool system data file name
OUT_U66	66	Logical unit to print out measure result
OUT_F66	@@x_tool.f66	File name to print out measure result
OPN_U66	Yes	Require file open of measure result print out
OUT_U99	99	Logical unit of measure result for restart
OUT_F99	@@x_tool.f99	File name of measure result for restart
OPN_U99	Yes	Require file open of measure result for restart
OHD_CTR	1000	No. of loop to decrease overhead of proper tool
TABCODE	chr(9)	Assignment inner code of tab code
TABNUM	8	Assignment of column number by tab code
INC_PATH	-none-	Set of include reference path
costtime@ on		Measure on directive in source program
costtime@ off		Measure off directive in source program
	x_tool.dat	Tool control data file name

Fig. 4.12 Contents of file x_tool.hlp

5．試用版評価

汎用目的のユーティリティ・ソフトウェアを開発した場合、様々な利用形態が想定され、開発関係者のみによるデバッグによりバグを取り切ることは困難である。また、ユーザの意見を広く聞くためにも試用版を作成して、試用アンケート調査を実施した。アンケートの依頼は「付録C」に添付した「kpx利用手引き」を配布して２回に分けて行った。モニタは全部で12名、他機種による試用を依頼したケースもあり、計16通の回答を得た。

5.1 調査項目

アンケートは以下の８項目に分けて実施した。

１．kpxで解析したユーザ・プログラムについて質問します。
２．並列化作業において、どのように既存のアナライザを使用していますか。
３．アナライザの解析機能や項目等の可不可について質問します。
４．kpx利用環境について質問します。
５．kpxのディフォルト構成および設定値についてお答えください。
６．アナライザ出力結果について質問します。
７．既存のアナライザと比較した場合について質問します。
８．その他、kpxを使用してお気づきになったバグ等や、ご意見があれば述べて下さい。

「１．」はユーザ・プログラムの特徴を知るための問である。インクルードがあったり、起動時や実行時に指定が必要な入力ファイルがあった場合、ツール実行のための手続きが１ステップ増える。また、プログラムの実行時間等の要因も、ツール使用時の快適度にかかわってくるため問うことにした。また、適度な並列化を行う場合と、高度な並列化を行う場合では、アナライザの使用方法も自ずと異なるため、チェックすることにした。

「２．」はモニタが普段からアナライザを使用しているかどうか、また、最初に一度だけ実行して並列化作業を行うかどうかについての問いである。

「３．」はアナライザの解析機能や項目等について、ユーザの一般的要望を問うための設問である。

「４．」は配布した「kpx利用手引き」について情報量の過不足と、kpxを利用することによるユーザの作業量負荷を問うための設問である。

「５．」はkpxが装備する数々の機能について、ディフォルト設定値が妥当かどうかの質問である。

「６．」はkpxの出力レイアウトや出力結果の精度についての質問である。

「７．」は既存のアナライザと比較したときのkpxの有効性についての質問である。

「８．」はバグの通知や率直な意見を伺うための項目である。

5.2 集計結果

項目毎の集計結果を以下に示す。

1．kpxで解析したユーザ・プログラムについて質問します。

本調査では、計算科学技術推進センターで並列処理について研究し、既に正常作動する並列プログラムを有する研究者にモニタを依頼した。アンケートは基本的に一人一通としたが、複数の提出をお願いしたケースもある。Table 5.1に集計用モニタ番号と、プログラムの実行時間、ktoolでの実行時間、ロードモジュール・サイズを示す。Table 5.2にプログラムにinclude文があるかどうか、また、起動時に入力ファイルがあるかどうかと、実行時に入力ファイルがあるかどうか、およびプログラム・ステップ数の一覧を示す。並列化に要した日数と最大加速率についても調査したが、ここでは割愛した。

2．並列化作業において、どのように既存のアナライザを使用していますか。

普段アナライザを使用する場合、どのような場合にどのようにアナライザを使用するか具体的に記述してください。

常に使用する　81%　13通　　殆どまたは全く使用しない　19%　3通

- 逐次プログラムを並列化する時、並列化したプログラムの最適化を行う時。
- Paragonの場合、Paragraphを使用して通信パターンを解析、profを使用して各ルーチンのCPU使用率を調べる。
- 計算コストの集中しているサブルーチン群やループを知るために使用する。各サブルーチン、ループの計算コストを測定。
- 逐次でVPPサンプラを実行し、コスト分布とその割合から並列化ルーチンを選定後、1つ並列化する毎にVPPサンプラを実行、並列化効率とその効果疎外要因を調査。
- 殆ど使用していない。
- 並列化設計時にプログラム分析。
- 高コスト・ルーチン抽出のためprofコマンドはよく使用する。
- CRAYで並列化する場合は、チューニング前処理として、モジュール関連図を出力してプログラムの概要を把握し、プロファイラを用いて実行時間のかかるサブルーチンを抽出する。チューニング後処理として、プロファイラを用いてルーチン単位、ループ単位のベクトル実行、並列実行による加速率を測定する。
- 並列化開始前。
- 取り敢えず、アナライザを起動して分析する。
- 並列化開始前。
- 全く使用せず。今回は2次元用の並列プログラムを3次元に拡張したのでアナライザの必要はなかった。また、普段もアナライザは使用しない。
- どのサブルーチンで最も実行時間がかかるかを調べる場合。普段は滅多に使わない。
- 並列化を行うために、主にサブルーチン・レベルでのホット・スポットを検出する。
- ベクトル化のためのdoループのループ長、ベクトル化率、実行時間を調べる。

3．アナライザの解析機能や項目等の可不可について質問します。

A) プログラム実行時間の計測。

ア）有効である　100%　16通　イ）なくてもよい　0%　0通
B）サブルーチン実行時間の計測。
ア）有効である　100%　16通　イ）なくてもよい　0%　0通
C）サブルーチン実行回数の計測。
ア）有効である　100%　16通　イ）なくてもよい　0%　0通
D）Doループ実行時間の計測。
ア）有効である　100%　16通　イ）なくてもよい　0%　0通
E）Doループ実行回数の計測。
ア）有効である　100%　16通　イ）なくてもよい　0%　0通
F）ステップ毎の実行時間の計測。
ア）有効である　56%　9通　イ）なくてもよい　44%　7通
G）ステップ毎の実行回数の計測。
ア）有効である　63%　10通　イ）なくてもよい　37%　6通
H）入出力実行時間の計測。
ア）有効である　81%　13通　イ）なくてもよい　19%　3通
I）入出力実行回数の計測。
ア）有効である　63%　10通　イ）なくてもよい　37%　6通
J）ノード間通信等の並列化オーバヘッド測定。
ア）有効である　100%　16通　イ）なくてもよい　0%　0通

4．kpx利用環境について質問します。

A）「kpx利用手引き」でkpxを利用できましたか。
ア）一人でできた　63%　10通　イ）不明な点を聞いてできた　37%　6通
ウ）できなかった　0%　0通
B）「kpx利用手引き」は十分ですか。不十分な場合コメントを下さい。
ア）はい　50%　8通　イ）いいえ　50%　8通
不十分な点・ptoolの出力結果の意味がよくわからない。
・付録1の指定に順番があるかどうかわからない。
・複数のdoループ等の指定方法がわからない。
・計測元ソース作成に必要なシェル・スクリプトCatKの内容の記述がない。
・計測元ソースを作成するときの「ヘッダを付けて」が不明。
・ktoolを2回実行する意味が不明。
・kdo/k_tool.datは、kgoでは../ktoolからコピーされることを明記されたい。
・単一版のためのものか並列版のためのものかわからない。
C）インストールは問題ありませんでしたか。問題があった場合コメントを下さい。
ア）はい　9通　イ）いいえ　7通
問題点　・Paragonでインストールするつもりで、別のコンピュータでインストーラを実行したが、その時にインストール失敗のメッセージが欲しい。
・インストールが正常終了したかどうか不明。
・makeの方が簡単で分かりやすい。
・インストールのパスが固定されているのは問題。
・ホーム・ディレクトリのkpx.tarや解凍したtool.tarとinstallerを、installer自身がkpxデレクトリに移すのが問題。
・エラー・メッセージが出たが正常に終了した。
・引き数のシステム名が誤っている場合、早めに打ち切れないか。
D）kpx利用に関する実質的人的負荷合計時間について質問します。
（kpxを配布資料で理解するのに要した時間）＋（kpxアナライザのtarファイルを転送後、

最初の解析結果を得るまでに要した時間）－（計測のためのプログラムの実行時間）を回答して下さい。

ただし、Paragon、VPP500、VPP300を使用された方は、インストールを除いたptoolまたはxtoolのみの実質的人的負荷合計時間も報告して下さい。

インストールを含んだktoolの人的負荷合計時間

1時間以内	44%	7通
1時間～2時間	19%	3通
2時間～3時間	31%	5通
3時間より上	6%	1通

インストールを含まないptoolまたはxtoolのみの人的負荷合計時間

1時間以内	100%	6通

E）kpx実行方法に関してインタラクティブ／バッチ等、その他お気づきの点があればコメントを下さい。

- VPP500はバッチシステムのため、kgoがGSPに対応していないので利用手順が難しい。
- エラー・メッセージが意味不明。
- 個人的にrmを-iオプション付きに変更しており、バック・スラッシュが必要となった。
- VPP300のバッチ・クラスはマルチ・ジョブ環境にないため、バッチの方が測定に適当。
- インストール先および実行ディレクトリが固定されていて使いづらい。
- シェル・スクリプトの完成度が低い。
- プログラムktoolの最後で不要なワーク・ファイルを削除しておいてほしい。
- 通常Makefileを使っているので、Makefileを利用できるようにしてほしい。
- バッチでジョブを流す場合、kgoを分割する必要があった。
- ディレクトリkdoでしか実行できない。
- Expressの実行環境では、メイン・ルーチンをオブジェクトとしてリンクするが、kpxではメイン・ルーチンのソースが必要。

F）kpx実行によるツール自体のオーバヘッド等に関してお気づきの点があればコメントを下さい。

- 通常の実行時間6,505秒が、kgoの2回目では7,260秒で終了し、許せる範囲。
- 並列実行時間102秒が、ptoolでは約120秒で終了し、許せる範囲。
- オーバヘッドが異常に少ない、勘違いか。
- 測定時間はktoolでは1.5倍、xtoolでは2.2倍。call文の挿入でベクトル・ループがスカラ化。
- 測定時間はkgoの回数測定で1.5倍、時間測定で1倍。
- オーバヘッドは比較的少ないと思う。
- オーバヘッド大、しかたないか。
- 遅いが許容範囲。
- オーバヘッドは無視できる程度。

5．kpxのディフォルト構成および設定値についてお答えください。

ktoolに関して。

A）計測対象プログラムのグローバル指定。（GLOBAL_ENT「kpx利用手引き」P16）

イ）妥当である　75%　12通　ロ）妥当でない　0%　0通

ハ）何とも言えない、不明　25%　4通

B）計測対象doループのグローバル指定。（GLOBAL_DOL「kpx利用手引き」P17）

イ）妥当である　81%　13通　ロ）妥当でない　0%　0通

ハ）何とも言えない、不明　19%　3通

C）計測対象入出力処理のグローバル指定。（GLOBAL_IOU「kpx利用手引き」P17）

イ）妥当である　81%　13通　ロ）妥当でない　0%　0通

ハ）何とも言えない、不明　19%　3通
D）doループの未確定ループ回数の計測指定。（UNSAT_DO「kpx利用手引き」P17）
イ）妥当である　56%　9通　ロ）妥当でない　6%　1通
ハ）何とも言えない、不明　38%　6通
E）計測の種別指定。（JST_CTR「kpx利用手引き」P6、P18）
イ）妥当である　63%　10通　ロ）妥当でない　6%　1通
ハ）何とも言えない、不明　31%　5通
F）使用した時間計測ルーチンについての妥当性。（mysystem.dat「kpx利用手引き」P22）
イ）妥当である　94%　15通　ロ）妥当でない　0%0通（妥当な時間計測ルーチン名）
ハ）何とも言えない、不明　6%　1通

ptoolに関して。
G）時間計測ルーチンの試行回数。（OHD_CTR「kpx利用手引き」P10、P25）
イ）妥当である　100%　16通　ロ）妥当でない　0%　0通（　　が妥当である）
ハ）何とも言えない、不明　0%　0通
H）最大プロセッサ数。（MAX_IDS「kpx利用手引き」P10、P26）
イ）妥当である　100%　16通　ロ）妥当でない　0%　0通（　　が妥当である）
ハ）何とも言えない、不明　0%　0通

xtoolに関して。
I）時間計測ルーチンの試行回数。（OHD_CTR「kpx利用手引き」P14、P29）
イ）妥当である　50%　8通　ロ）妥当でない　0%　0通（　　が妥当である）
ハ）何とも言えない、不明　50%　8通

その他のディフォルト値（「kpx利用手引き」の付録P16以降）で意見があれば述べて下さい。
・Paragonの時間計測ルーチンdclockをptoolではディフォルトで測定対象外にした方が良い。
・適切。
・call文の挿入でベクトル・ループがスカラ化し、時間がかかるため対象のループの測定を抑止したいが、制御データにはその様な機能がない。（kpxのディレクティブ機能でユーザ・ソースを変更すれば可能。）
・例えば、「ENT_ENT」を「KPX_ENT_ENT」にした方が計測元ソースのシンボル名と重複しないと思う。
・基本的にとんでもない値でなければ良いのではないでしょうか。

6．アナライザ出力結果について質問します。

A）出力レイアウト（出力文字数、出力文字間隔等）は適切ですか。不適切であると思われる方はその理由を述べて下さい。
ア）はい　37%　6通　イ）いいえ　63%　10通
理由　・見やすいサイズのフォントでA4プリント出力した時、一行で完結して欲しい。
・ktoolの出力はサブルーチン毎にdoループを集計してほしい。
・ktoolの出力は横に長すぎる。
・コラム値は通常80を採用しているので、80コラムの方が見やすい。
・一行であるべき行が、画面の右で折り返して２行になる。
・出力にある「MEAN」が何の平均かすぐには理解できなかった。
・一行130文字程度以内にした方がよいと思う。
・doループ表示等をサブルーチン毎に表示してほしい。
・出力の文字数は１行80文字にしてほしい。

B）アナライザ出力結果の精度は十分ですか。不十分であればコメントを下さい。
ア）はい　63%　10通　イ）いいえ　37%　6通
コメント・ptoolの結果は19秒であったが、独自に測定した値では25秒であった。
・xtoolの結果は以前、独自に測定した値より大きな値であった。
・xtoolで、PE間通信等が正しく評価できているか疑問を感じた。
・xtoolでは純粋な通信時間を正しく評価できていない。
・ktoolのkgoを使用するとktool2.datのディフォルトで測定制限されているので、既存ツールで１位のルーチンが測定出力されなかった。
・既存ツールと順位が異なる。

C）その他、解析結果についてお気づきになった点があればお書き下さい。
ptoolで全ノードのオーバヘッドを積算しているが、何に有効なのか不明。
ptoolで全オーバヘッドのノード数による平均が欲しい。
ktoolの出力結果に論理機器番号の情報がほしい。
サブルーチンの消費時間の報告では、あるルーチン内からコールされているルーチンの時間をコール側に含めたリストも別に出力してほしい。

7．既存のアナライザと比較した場合について質問します。

A）比較した既存のアナライザ名を記述して下さい。
モニタ１　gprof。
モニタ２　Paragraph。
モニタ３　ANALYZER-P/SX。
モニタ４　VPPアナライザ。
モニタ５　VPPアナライザ。
モニタ６　N/A。（普段より既存ツールを使用していない）
モニタ７　afrt。
モニタ８　fanp、prof。
モニタ９　flowview、jumpview、procview、atexpert、HPM、perfview。
モニタ10　procview、jumpview。
モニタ11　atexpert、jumpview、HPM。
モニタ12　prof。
モニタ13　N/A。（普段より既存ツールを使用していない）
モニタ14　N/A。（普段より既存ツールを使用していない）
モニタ15　XTOOL（Express）。
モニタ16　N/A。

B）あなたにとって、kpxの有効な点や不必要な点、あるいは不十分な点があればコメントを下さい。
モニタ１　ptoolの有効な点：各ノード毎のオーバヘッドがでる。
ptoolの不十分な点：測定精度が不明。
kpxの有効な点：サンプリングでなく、時間計測ルーチンを使用している点。
モニタ２　長所：Paragraphは非常に遅くていらいらする。また、出力するまでの手順が繁雑なのに対し、kpxは手軽に結果が得られる。
短所：Paragraphは各種GUIを使って見やすく情報提供してくれるのに対し、kpxはファイル出力しかサポートしていない。
モニタ３　既存ツールに対する優位性が不明確。
モニタ４　有効な点：サブルーチンのdoループ毎の実行回数がわかる。時間測定ができる。
無効な点：kpx実行中の出力情報の殆どが意味不明。
モニタ５　N/A。（モニタ４と重複）

モニタ６　N/A。（普段より既存ツールを使用していない）
モニタ７　doループ毎の実行回数、実行時間が有効である。
モニタ８　ソース・ファイルを一つのディレクトリに集め、しかも一つのファイルにしなければならず、ユーザにとって非常に負担が大きい。
モニタ９　kpxのデータは既存ツールを利用して採取可能で、しかもGUIが付随して解析しやすい。また、既存ツールの方が高速である。特に有効性を感じなかった。
モニタ10　kpxは既存ツールより遅い。コンパイル・オプションを指定して起動ができないので使いづらい。
モニタ11　既存ツールと比較すると、１．起動法が単純でない　２．GUIがほしい　３．並列化オーバヘッドの測定だけでは不足、そこから各PE数での並列化効果をシミュレートしてほしい。
モニタ12　doループ毎の解析は有効。だが、精度を信用していいのか不安。
モニタ13　N/A。（普段より既存ツールを使用していない）
モニタ14　N/A。（普段より既存ツールを使用していない）
モニタ15　kpxはMPI上ではOKだが、Express上では駄目。ExpressのXTOOLはExpress上ではOKだが、MPI上では駄目。補間の関係がある。ただし、ExpressのXTOOLは各ノード毎にアナライズできる。
モニタ16　N/A。

C）kpxアナライザを必要としますか。また、その理由を述べてください。
ア）必要である　31%　４通　イ）あればよい　38%　５通
ウ）既存ので十分、または必要としない　31%　４通　エ）その他　０%　０通
（ただし、普段からアナライザを使用していない３名は集計から除外した）
選択した理由。

モニタ１　Paragonでは、各ノードが異なった動きをするプログラミングが多い。この時各ノード毎の測定が容易にできる。
モニタ２　Paragonで手軽に使えるツールとして便利。
モニタ３　調査項目７．B）に同じ。
モニタ４　既存アナライザと比較しながら測定制度を検討するのに必要。自分で計測ルーチンを挿入する代りに有効。
モニタ５　N/A。（モニタ４と重複）
モニタ６　N/A。（普段より既存ツールを使用していない）
モニタ７　doループ毎の実行回数、実行時間測定のため一度使って見たい。
モニタ８　使いやすさではprofに及ばないし、機能ではfanpに劣ると思われる。したがって、kpxのメリットを見い出せない。
モニタ９　調査項目７．B）に同じ。
モニタ10　既存の方が信用できる。
モニタ11　CRAYには優れたアナライザが豊富にあるため。
モニタ12　doループ毎の解析は有効。
モニタ13　N/A。（普段より既存ツールを使用していない）
モニタ14　N/A。（普段より既存ツールを使用していない）
モニタ15　調査項目７．B）に同じ。
モニタ16　N/A。

D）既存アナライザとの比較を踏まえ、その他全般に渡った感想をお聞かせ下さい。
モニタ１　gprofはシステムのルーチンについても同様に報告するため、報告リストが理解しづらく役に立たない。kpxはParagonでは必須。
モニタ２　調査項目７．B）に同じ。
モニタ３　調査項目７．B）に同じ。
モニタ４　ベータ版のためツールのバグが多い。すんなり結果がでることが重要。

モニタ５　（モニタ４と重複）
モニタ６　N/A。（普段より既存ツールを使用していない）
モニタ７　N/A。
モニタ８　N/A。
モニタ９　結果をファイルに出すという方法は考え直すべきではないか。計測したデータをどのように加工して解り易い形で提供するかと言う点を考えた方がよい。
モニタ10　現状ではメリットを感じない。
モニタ11　感覚的に古いツールのような気がする。
モニタ12　N/A。
モニタ13　N/A。（普段より既存ツールを使用していない）
モニタ14　N/A。（普段より既存ツールを使用していない）
モニタ15　GUIがあればよいと思う。
モニタ16　N/A。

８．その他、kpxを使用してお気づきになったバグ等や、ご意見があれば述べて下さい。

様々な意見やアドバイスを頂いたが、バグ以外について列挙する。

・ptoolは、ノード数の出力を縦一列に出す。表示レイアウトのデザインが必要。
・ptoolではSYS_SUB、SYS_FUNCが使用できるが、xtoolではできないので使いにくい。ptool、xtoolでは指定した箇所を計測するオプションも必要と考える。
・ユーザ・プログラムのコンパイル時にインフォメーションが多いが、kpxが未使用の変数をインクルードしているためで、なんとかならないか。コンパイル・オプションでインフォメーションを消すとプログラムのデバッグに問題がある。
・kgoの「fortran @@f22.f」の「fortran」の部分にシェル・スクリプトの最初で適当なコンパイル・コマンドを代入する。
・kpxのインストール先および実行ディレクトリが固定されているのは良くない。
・*.fを一つのファイルにするとmakeが使えない。
・サブルーチン毎のコンパイルができない。
・プロセッサの利用率（CPU時間と経過時間の比を見ればいいか）が見れたら嬉しい。
・オブジェクトのMAINに対応をするのは難しいでしょうか。
・出力をポストスクリプトにするフィルタがあると便利。
・時間（オーバヘッド）との兼ね合いで計測対象範囲を決めるのが難しそう。試行錯誤が必要だが時間のコストが高すぎる。
・対象範囲の操作にビジュアル化が必要。（静的（動的がベスト）な手続き構造を見せてこの下は対象とするしない等）
・SR2201のf90のOPT(4),EXPANDでABENDした。疑似ベクトル指定で動かないのは問題。

5.3　検討

１．kpxで解析したユーザ・プログラムについて質問します。

モニタは12名で、16通のアンケートを回収した。モニタおよびプログラムは無作為に抽出したが、計算科学技術推進センターでは、並列化の研究をしており、既に動いていたプログラムを選択したため、プログラム構造は大規模科学技術計算の並列化向き特性を少なからず備えている。プログラムは1000ステップ以下のMDから1万ステップ以上のRTMCまで、実行時間も数10秒のTrans5から数時間のMDまで千差万別であり、プログラム・サイズもmyflow、mdllcの数

10KBからxlapwcp、RTMCの数100GBまでと幅広い。

２．並列化作業において、どのように既存のアナライザを使用していますか。

普段からアナライザを殆ど使っていない人が19%おり、これらの人はどうしても計測が必要な場合には自分自身で時間計測ルーチンを挿入して時間計測を行っている。kpxは時間計測ルーチンを挿入した形の計測用ソースをユーザに解放しているので、その意味では、従来のアナライザに無関心なこれらのユーザにも受け入られる余地があるのではないだろうか。

３．アナライザの解析機能や項目等の可不可について質問します。

ステップ毎の実行時間、実行回数の計測については、「イ）なくてもよい」が各々44%、38%で、入出力の実行回数の計測についても「イ）なくてもよい」が38%あった。ちなみに、kpxにはステップ毎の計測を直接行う機能はない。

４．kpx利用環境について質問します。

本アンケート調査は「kpx利用手引き」のみを配布して行われた。アンケート16通の内、ktoolでは約半数の人が、インストールを含めて１時間以内でプログラム解析結果を得ている。３時間以上の人的負荷を強いられたのは１名だけであった。ktoolと、ptoolまたはxtoolを併用した人は、利用法がktoolと類似しており、インストール時間も含まないので、全員１時間以内にプログラム解析結果を得ている。また、ツール自体のオーバヘッドについては「オーバヘッド大」とする１名を除き、「許せる範囲」「異常に少ない」「比較的少ない」「許容範囲」「無視できる程度」との回答が多かった。プログラムの特性にも依るが、kpxには計測制限機能があり、うまくこの機能が利用できれば、オーバヘッドのかなりの部分を削減することが可能である。

５．kpxのディフォルト構成および設定値についてお答えください。

ディフォルト構成および設定値については「ロ）妥当でない」を選択した人は殆どいなかった。ディフォルトは変更が可能なので「基本的にとんでもない値でなければ良いのではないでしょうか。」と言う意見もあった。

６．アナライザ出力結果について質問します。

特に一行の出力文字数に対する不満が多く、これについては、「5.4　機能追加修正」で後述する様に修正変更をする。

7．既存のアナライザと比較した場合について質問します。

普段からアナライザを使用しない３名を除くと、「ア）必要である」「イ）あればよい」を合わせて69%のモニタがkpxの有効性を主張している。「ア）必要である」は計４名であった。内訳はParagonモニタ２名中１名、VPPモニタ４名中３名で、特定機種のユーザに集中していた。「ウ）既存ので十分、または必要としない」も４名で、内訳はSX-4モニタ１名中１名のユーザが「使いやすさではprofに及ばないし、機能ではfanpに劣ると思われる。したがって、kpxのメリットを見い出せない。」とし、CRAYモニタ３名は３名共、各々有効性、信頼性、優位性においてCRAYアナライザが勝っているとし、kpxには有用性がないとしている。kpxの有効性、無効性はそれぞれのコンピュータで使用可能な既存アナライザに強く依存している。しかしながら、kpxはすべてのUNIXコンピュータで動作するように設計されており、どのコンピュータでも使えるツールとして汎用性があるという利点がある。

8．その他、kpxを使用してお気づきになったバグ等や、ご意見があれば述べて下さい。

省略。

5.4　機能追加修正

kpxの試用版は、計算科学技術推進センターの12名によってモニタされ、数々の試験と貴重な回答を得るに至った。BMTワーキング・グループではユーザの意見を検討し、kpxに対し「付録B」に添付する機能追加修正、および下記項目「７）」の修正を行うことを決定した。以下に機能追加修正項目の原案抜粋を示す。

1）ユーザのプログラム・ソース・ファイルを１つにまとめずにkpxの実行を可能とする。
2）kpx独自のサブルーチン群をライブラリ化する。ただし、テスト版として開発する。
3）ktool出力フォーマットを80文字以内に制限し、内容も変更する。
4）ptoolおよびxtoolで全オーバヘッドのノード数による平均値を出力する。
5）ptoolおよびxtoolでデータ採取対象の制限を、対応する制御データ（p_tool.datまたはx_tool.dat）で指定可能にする。
6）ktoolの出力結果をグラフ化する。
7）インストーラの修正。

Table 5.1 Programs used for evaluating tool kpx (to be continued)

T1: Execution time in second without useing analyzer
T2: Execution time in second with useing analyzer
LM Size: Load Module Size(KB)

	Paragon	Monte-4	VPP500	VPP300	SX-4	T94	SP2	SR2201
Trans5						Monitor 9		
T1／T2						13／63		
LM Size						1,595		
MD	Monitor1		Monitor 4	Monitor 5				
T1／T2	6,505/7,260		701／1,057	703／1,663				
LM Size	13,000		13,000	14,000				
gyro3c						Monitor10	Monitor12	
T1／T2						530／1,250	360／420	
LM Size						1,800	200	
LB2				Monitor 6				
T1／T2				95／95				
LM Size				2,032				
Library								Monitor15
T1／T2								150／945
LM Size								—
xlapwcp					Monitor 8			
T1／T2					77／82			
LM Size					100,000			
RTMC		Monitor 3						
T1／T2		36／41						
LM Size		340,000						
Pstc-2d							Monitor13	
T1／T2							394／397	
LM Size							57,000	

Table 5.1 Programs used for evaluating tool kpx (continued)

T1: Execution time in second without useing analyzer
T2: Execution time in second with useing analyzer
LM Size: Load Module Size(KB)

	Paragon	Monte-4	VPP500	VPP300	SX-4	T94	SP2	SR2201
myflow							Monitor14	
T1／T2							200／1,200	
LM Size							53	
mdllc				Monitor 7				
T1／T2				990／987				
LM Size				76				
SPE						Monitor11		
T1／T2						1,279/4,959		
LM Size						1,800		
EGS4	Monitor 2							Monitor16
T1／T2	56／56							344／6,944
LM Size	614							1,200

Table 5.2 Programs Characteristics

Include : Needed Include files to compiling
Input(Invo.) : Needed data files to execution at invoking program
Input(Exec.) : Needed data files to execution at executing time

	Include	Input(Invo.)	Input(Exec.)	Step size
Trans5	○	○	○	1,174
MD	○	—	—	938
gyro3c	—	—	—	7,000
LB2	○	○	○	2,889
Library	○	—	—	—
xlapwcp	—	—	—	(Part)300
RTMC	○	○	○	11,176
Pstc-2d	○	○	○	3,000
myflow	○	—	—	1,577
mdllc	○	○	—	3,600
SPE	○	○	○	7,000
EGS4	○	—	○	5,739

6．おわりに

kpxは、スケーラビリティ予測ツールに必要なデータを入力するためのプログラム分析ツールで、ヘテロジニアスな計算機上で均一にモデル・パラメータを測定するツールである。スケーラビリティ予測の研究途上で独立したプログラム並列化支援解析ツールとしたが、フリーウェアとして科学技術計算を行う多くの研究者に利用していただければ幸いである。

謝辞

計算科学技術推進センターの研究員の方々にモニタをお願いし、多数の大変貴重な意見を賜わり感謝いたします。

参考文献

（1）T94：Guide to Parallel Vector Applications SG-2182 2.0
（2）SP2：IBM Parallel Environment for AIX : Operation and Use Version 2.1.0
（3）SP2：AIX Version 4.1 Commands Reference
（4）SP2：User's Guide Parallelware -FORTRAN- 8X20-3-541(E)
（5）SR2201：HI-UX/WE2 リアルタイムパフォーマンスモニタfor SR2201
（6）SR2201：HI-UX/WE2 アプリケーションプログラムチューニングツールfor SR2201
（7）SR2201：The FORGE Explorer TOOLBOX The Distributed Memory Fortran Parallelizer FORGEX/DMP User's Guide Version 2.0
（8）SR2201：FORGE Explorer The complete Fortran Program Browser On UNIX/Motif Workstations
（9）FORGEX Version 2.0 User's Guide
（10）SR2201：FORGE High Performance Fortran HPF Parallelizing Pre-Compiler for Distributed Memory Multi-Processor Systems xhpf Version 2.0 User's Guide
（11）Yuji MATSUYAMA：Monte-4,VPP500：JAERI-Data/Code 96-006

6．おわりに

kpxは、スケーラビリティ予測ツールに必要なデータを入力するためのプログラム分析ツールで、ヘテロジニアスな計算機上で均一にモデル・パラメータを測定するツールである。スケーラビリティ予測の研究途上で独立したプログラム並列化支援解析ツールとしたが、フリーウェアとして科学技術計算を行う多くの研究者に利用していただければ幸いである。

謝辞

計算科学技術推進センターの研究員の方々にモニタをお願いし、多数の大変貴重な意見を賜わり感謝いたします。

参考文献

（1）T94：Guide to Parallel Vector Applications SG-2182 2.0
（2）SP2：IBM Parallel Environment for AIX : Operation and Use Version 2.1.0
（3）SP2：AIX Version 4.1 Commands Reference
（4）SP2：User's Guide Parallelware -FORTRAN- 8X20-3-541(E)
（5）SR2201：HI-UX/WE2 リアルタイムパフォーマンスモニタfor SR2201
（6）SR2201：HI-UX/WE2 アプリケーションプログラムチューニングツールfor SR2201
（7）SR2201：The FORGE Explorer TOOLBOX The Distributed Memory Fortran Parallelizer FORGEX/DMP User's Guide Version 2.0
（8）SR2201：FORGE Explorer The complete Fortran Program Browser On UNIX/Motif Workstations
（9）FORGEX Version 2.0 User's Guide
（10）SR2201：FORGE High Performance Fortran HPF Parallelizing Pre-Compiler for Distributed Memory Multi-Processor Systems xhpf Version 2.0 User's Guide
（11）Yuji MATSUYAMA：Monte-4,VPP500：JAERI-Data/Code 96-006

6．おわりに

kpxは、スケーラビリティ予測ツールに必要なデータを入力するためのプログラム分析ツールで、ヘテロジニアスな計算機上で均一にモデル・パラメータを測定するツールである。スケーラビリティ予測の研究途上で独立したプログラム並列化支援解析ツールとしたが、フリーウェアとして科学技術計算を行う多くの研究者に利用していただければ幸いである。

謝辞

計算科学技術推進センターの研究員の方々にモニタをお願いし、多数の大変貴重な意見を賜わり感謝いたします。

参考文献

（1）T94：Guide to Parallel Vector Applications SG-2182 2.0
（2）SP2：IBM Parallel Environment for AIX : Operation and Use Version 2.1.0
（3）SP2：AIX Version 4.1 Commands Reference
（4）SP2：User's Guide Parallelware -FORTRAN- 8X20-3-541(E)
（5）SR2201：HI-UX/WE2 リアルタイムパフォーマンスモニタfor SR2201
（6）SR2201：HI-UX/WE2 アプリケーションプログラムチューニングツールfor SR2201
（7）SR2201：The FORGE Explorer TOOLBOX The Distributed Memory Fortran Parallelizer FORGEX/DMP User's Guide Version 2.0
（8）SR2201：FORGE Explorer The complete Fortran Program Browser On UNIX/Motif Workstations
（9）FORGEX Version 2.0 User's Guide
（10）SR2201：FORGE High Performance Fortran HPF Parallelizing Pre-Compiler for Distributed Memory Multi-Processor Systems xhpf Version 2.0 User's Guide
（11）Yuji MATSUYAMA：Monte-4,VPP500：JAERI-Data/Code 96-006

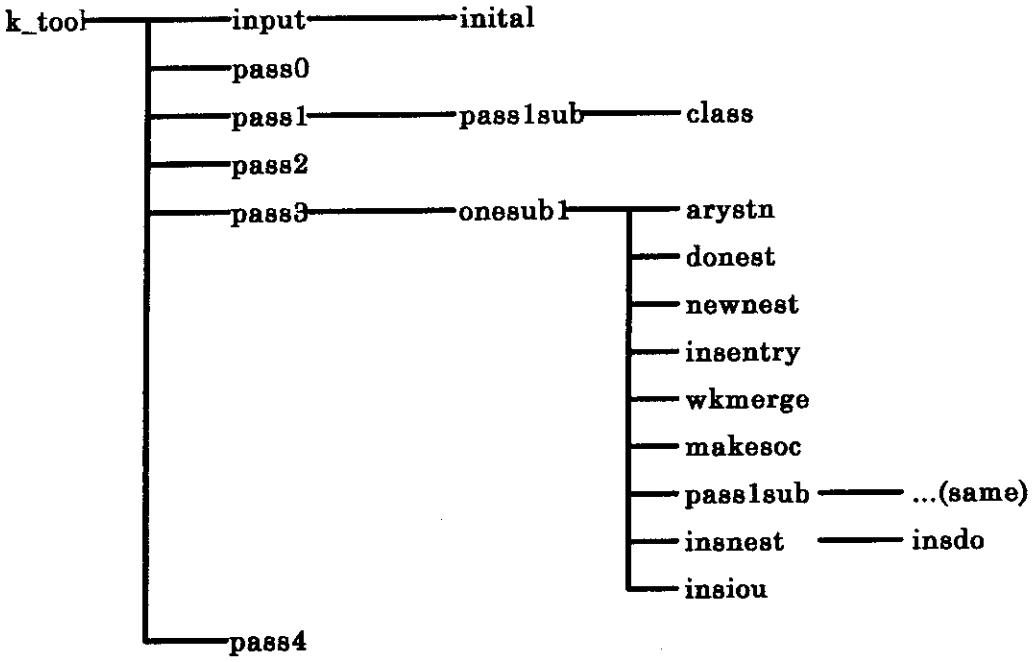
付録 A

ktoolプログラム概要

目　　次

1．システムの概要図

1)概略ツリー構造



2)主な処理の概略説明

(1)k_tool :主プログラム。
(2)input :制御データの入力。
(3)inital :初期値設定（標準値）。
(4)pass0 :オリジナル・ファイル、ライン情報の取得、マルチ文の分割。
(5)pass1 :FORTRAN 文解釈の制御。
(6)pass1sub :FORTRAN 文が 1 レコードとなるファイルを作成。
(7)class :FORTRAN 文のクラス分類。
(8)incinc :インクルード・ファイルのソース・プログラムへの展開。
(9)pass2 :プログラムの入口名の取得。
(10)pass3 :プログラム毎にを作成する制御。
(11)onesub1 :プログラム毎にプログラムを解析し、修正データを作成。
(12)arystn :関数名を求めるため配列名の取得、第一実行文の位置の取得。
(13)donest :多重 DO ループのネストの取得。
(14)newnest :共通 DO 端末の分割、DO 端末処理のための新文番号の取得。
(15)insenrty :プログラム計測のための（修正データ）を作成。

(16)wkmerge :計測用ソースの修正データの併合。
(17)makesoc :修正データによりを作成。
(18)insnest :DO ループの修正データを作成。
(19)insdo :DO ループのための（修正データ）を作成。
(20)insiou :入出力処理のための（修正データ）を作成。
(21)pass4 :計測用ソースから利用する計測用ライブラリを作成。

3)全体処理の概略説明

二段階のフェーズで計測用ソースを作成している。第一フェーズでは、プログラムの計測ルーチンの挿入、共通 DO ループ端末の分割、DO 端末における入出力文や論理 IF 文が存在する場合に DO ラベルの新設を行う。第二フェーズでは、入出力の計測文の挿入、DO ループの計測ルーチンの挿入を行う。

(1)第一フェーズ

プログラムの計測ルーチンは、処理の開始前に計測開始ルーチンの挿入、副プログラムの場合にはリターン処理の前に計測終了ルーチンの挿入を行う。主プログラムの場合や STOP 文がある場合は、計測終了ルーチンの挿入と共に、計測結果の出力処理ルーチンの挿入を行う。終了処理は、STOP 文のほか、CALL EXIT も対象としている。

共通の DO ループ端末を持つ DO ループは、最内ループを除いて DO ループ・ラベルを新設し、多重ループを分割する。また、DO 端末に入出力がある場合や論理 IF 文がある場合にも、同様に DO ループ・ラベルを新設する。

以上の条件で計測元ソースから作業用の計測ソースを作成する。

(2)第二フェーズ

第一フェーズで作成された計測ソースを再び解釈して、DO ループ、入出力を解析する。

DO ループは、その回転数が DO ループを開始する前に判明している場合と、判明していない場合がある。 DO ループの回転数があらかじめ判明していない場合は、DO WHILE 型、DO UNTIL 型や、GO TO 文で DO ループ外へ飛び出しのある場合や、RETURN 文、STOP 文を内包する場合である。いずれの場合も、DO ループの前に計測開始ルーチンを挿入し、後に DO ループの計測終了ルーチンを挿入する。ただし、回転数があらかじめ判明していな

い場合は、DO ループ外への飛び出しの前に計測終了ルーチンを挿入する。

入出力に関しては、前に計測開始ルーチンを挿入し、後に計測終了ルーチンを挿入する。本文の論理機番の説明で示される変数名は、入出力される変数名や配列名を表している。

4)ファイルの入出力

実際に入出力を行うルーチンは、共通した部分があるためサブルーチンになっている。

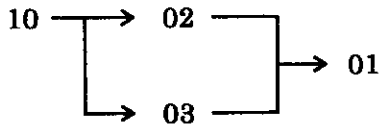
以下の説明で、ｎｎ ⟶ｍｍは論理機番ｎｎから論理機番ｍｍが作成される事を示しており、ｎｎだけの場合はファイルの引用を表し、ｍｍだけの場合はファイルの作成を表している。

(0)ファイルに共通する変数の説明

sav,wrk,w 等の接頭語のついたものは同じ意味を持つため説明を省略する。

f_seq :ファイルの存在位置データ。
o_lin :オリジナル・ライン位置。
line :ライン位置（ソース修正の指標になる）。
lens :レコードの長さ（バイト数）。
nc :レコードの長さ（バイト数）。
buf :FORTRAN ラインのデータ。
lstr :FORTRAN 文の開始ライン位置。
lend :FORTRAN 文の終了ライン位置。
str :FORTRAN 文のデータ。
type :FORTRAN 文のタイプ。
adall :修正情報。
lcv2 :IF 文等の第２文の開始位置。

(1)pass0



a)10 :計測元ソースで、ファイル分割指示子があるもの。

buf

b)02 :レコードの長さ（バイト数）を知りその情報を付加。

f_seq,o_lin,nc,wrk_buf(1:nc)

c)03 :マルチ FORTRAN 文の情報。

cf_seq,co_lin,c_pos

cf_seq :**f_seq** に同じ。

co_lin :**o_lin** に同じ。

c_pos :マルチ文の（；）の位置。

d)01 :計測元ソースに、ファイル位置、オリジナル・ライン位置情報が付加されたもの。
マルチ文があれば分解したものを作成。

f_seq,o_lin,buf

(2)pass1

01 ⟶ 02 ⟶ 03 ⟶ 04

実際には pass1sub でファイルの入出力を実行。

(3)pass1sub

01 ⟶ 02 ⟶ 03 ⟶ 04
31 ⟶ 02 ⟶ 03 ⟶ 16

引数で論理機番が指定されており、呼び出し元により論理機番が異なるが、ファイルの内容は等しいので 01 から 04 で代表して説明する。

a)01 :pass0 の 01 から引き継ぐもの。

b)02 :一時的利用で、ライン位置、レコードの長さの情報を追加。

f_seq,o_lin,line,lens,buf(:lens)

c)03 :FORTRAN 文を 1 レコードで作成（大文字化、ブランク詰め）。

sav_f_seq,sav_o_lin,lstr,lend,lens,str

d)04 :FORTRAN 文の分類、IF 文等の第 2 文が存在するデータの追加。

e) IF(....)WRITE(...)....の場合は第 2 文が存在し、その文のタイプ等が必要となるため、その文を第 2 レコードとして追加する。この時、第 1 文目の lcv2 が 0 以外の値になる。

f_seq, o_lin, type, lcv2, lstr, lend, adall, lens, str

(4)pass2

04 ⟶ 41

a)04 :pass1sub の 04 から引き継ぐもの。

b)41 :プログラム情報（デバッグ用）。

prg_inf(j, i), j=1, 3), prg_nam(i), prg_ent(i)

prg_inf :プログラム種別、レコード番号（開始、終了）。

prg_nam :プログラム名（主入口名）。

prg_ent :副入口名。

(5)pass3

01 ⟶ 21

04 ⟶ 16

⟶ 24

a)01 :pass0 の 01 から引き継ぐもの。

b)21 :01 と同じ内容で、プログラム毎に抽出したもの。

c)04 :pass1sub の 04 から引き継ぐもの。

d)16 :04 と同じ内容で、プログラム毎に抽出したもの。

e)24 :計測対象となったプログラム情報。

w_seq,w_lin,seq_ent,g_entry

seq_ent :プログラム順番。

g_entry :プログラム入口名。

(6)deadcopy

21 ⟶ 22　　フェーズ１で全く修正の必要のない場合。
31 ⟶ 22　　フェーズ２で修正のない場合。

a)21　:pass3 の 21 から引き継ぐもの（31）。
b)22　:作成されるプログラム。
buf(:lensw)

(7)onesub1

12 ⟶ 15

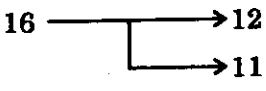
a)12　:donest の 1 から引き継ぐもの。
b)15　:DO の１ブロックの抽出（入れ子の DO ネストを含む）。

(8)arystn

16 ⟶

a)16　:pass3 の 16 から引き継ぐもの。

(9)donest



a)16　:pass3 の 16 から引き継ぐもの。
b)12　:DO ループの解析のためのデータ。
f_seq,o_lin,lbl_nest,lbl_work,lbl_goto,do_kind,st_kind,dir_knd,lstr,
lend,adflg,doseq,lens,str
lbl_nest:DO ループのネスト・レベル。
lbl_work:DO ループ・ラベル。
lbl_goto:GOTO 文のラベル。
do_kind:DO ループ種別。
=1;DO 文（WHILE,UNTIL 型を除く）

(6)deadcopy

21 ⟶ 22　　フェーズ１で全く修正の必要のない場合。
31 ⟶ 22　　フェーズ２で修正のない場合。

a)21　:pass3 の 21 から引き継ぐもの（31）。
b)22　:作成されるプログラム。
buf(:lensw)

(7)onesub1

12 ⟶ 15

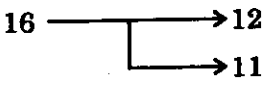
a)12　:donest の１から引き継ぐもの。
b)15　:DO の１ブロックの抽出（入れ子の DO ネストを含む）。

(8)arystn

16 ⟶

a)16　:pass3 の 16 から引き継ぐもの。

(9)donest



a)16　:pass3 の 16 から引き継ぐもの。
b)12　:DO ループの解析のためのデータ。
f_seq,o_lin,lbl_nest,lbl_work,lbl_goto,do_kind,st_kind,dir_knd,lstr,
lend,adflg,doseq,lens,str
lbl_nest:DO ループのネスト・レベル。
lbl_work:DO ループ・ラベル。
lbl_goto:GOTO 文のラベル。
do_kind:DO ループ種別。
=1;DO 文（WHILE,UNTIL 型を除く）

=**2**;DO 文（ WHILE,UNTIL 型）

=**3**;DO 端末文

=**0**;DO 関連文以外

st_kind :DO ネスト内の文の種別。

=**0**;GOTO 文

=**1**;DO 文

=**2**;GOTO 文を伴う論理 IF 文

=**3**;入出力文を伴う論理 IF 文

=**4**;RETURN 文（論理 IF 文内のものを含む）

=**5**;STOP/CALL EXIT 文（同）

=**6**;算術 IF 文

=**7**;DO 端末の入出力文

=**8**;計算型、割当型 GOTO 文

=**9**;その他のラベル付き文

dir_kind:ツール・ディレクティブの存在フラグ。

adflg :修正データ種別（詳細は makesoc で行う）。

doseq :DO ループの出現順番。

c)11 :副プログラムの呼び出し情報。

stmtype,call_name,lstr,lend,lens,str

stmtype:文の種別。

(10)newnest

15 ⟶ 14

a)15 :onesub1 の 15 から引き継ぐもの。

b)14 :DO ループで新ラベルの DO ループへ書き換えが必要な修正データ。

f_seq,o_lin,kind,ksub,do_lbl,lstr,lend,adflg,lens,str

kind :文の種別（1;DO ループ、2;ルーチン出口）。

ksub :計測処理のライブラリ種別。

do_lbl :DO ループの新しいラベル・データ。

(11)insentry

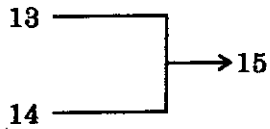
16 ⟶ 13

a)16 :pass3 の 16 から引き継ぐもの。

b)13 :プログラムの計測用の書き換えが必要な修正データ。

f_seq,o_lin,kind,ksub,dolbl,w_lstr,w_lend,adflg,sav_lens,sav_str

(12)wkmerge

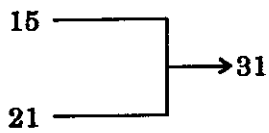


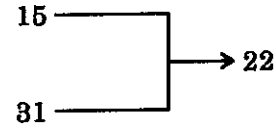
a)13 :insentry の 13 から引き継ぐもの。

b)14 :newnest の 14 から引き継ぐもの。

c)15 :ソースの修正データで、13 と 14 のソース修正ライン番号で併合。

(13)makesoc





左側は 1 回目の作業用ソースの作成時の論理機番で、右側は 2 回目のソースを作成する時の論理機番である。内容が同じであるため左側の論理機番で説明する。

a)15 :wkmerge の 15 から引き継ぐもの。

f_seq_15,o_lin_15,kind,ksub,dolbl,lstr,lend,adflg,lens,str

adflg :ソースの修正フラグ。

=1;オリジナル・ライン番号の後に単純追加

=2;オリジナル・ライン番号の単純削除

=3;DO ループ・ラベルの変更に伴う DO 文の置き換え

=4;オリジナル・ライン番号の前に単純追加

=5;論理 IF 文の入出力文のブロック IF 文への置き換え

=6;ラベル付き入出力文の置き換え（CONTINUE 文の追加）

=7;DO ループ内算術 IF 文のブロック IF 文への置き換え

=8;現在未利用

=9;DO ループ内の DO 端末への GOTO 文のラベル変更

=10;論理 IF の GOTO 文の変更

=**11**;DO 端末の入出力文の変更

=**12**;論理 IF における RETURN/STOP/CALL EXIT の変更

=**13**;論理 IF の真条件の文を追加（第２文）

内容の詳細は、**adflagc.inc** に示されている。

b)21 :pass3 の 21 から引き継ぐもの。

c)31 :修正の加えられたソース。

f_seq,o_lin,i_buf(:lb) :中間ソースの場合。

i_buf(:lb) :最終ソースの場合。

(14)cmtset

21 ⟶17 (31 ⟶ 17)

a)21 :pass3 の 21 から引き継ぐもの（31 の場合は２回目の処理）。

b)17 :コメント文を抽出したもの。

(15)cmtput

17 ⟶31 (17 ⟶ 22)

a)17 :cmtset の 17 から引き継ぐもの。

b)31 :修正の加えられたソース（22 の場合は２回目の処理）。

(16)insnest

⟶26

a)26 :計測対象となった DO ループ情報。

do_lbl(16,i),do_lbl(17,i),seq_dol,g_entry,seq_ent,do_lbl(2,i),
do_lbl(6,i),do_lbl(9,i)

do_lbl(16,i) :ソースのファイル情報。

do_lbl(17,i) :ソースのライン情報。

do_lbl(2,i) :DO ループ・ラベル。

do_lbl(6,i) :DO ループのプログラム内出現順番。

do_lbl(9,i) :DO ループの種別（回転数の確定か否）。

seq_dol :DO ループの出現順番。

g_entry :プログラム名。
seq_ent :プログラムの出現順番。

(17)insdo

15 ⟶14

a)15 :onesub1 の 15 から引き継ぐもの。
b)14 :DO ループの計測ルーチンのための修正データ。

(18)insiou

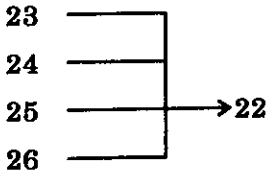
16 ⟶13
⟶25

a)16 :pass3 の 16 から引き継ぐもの。
b)13 :入出力の計測ルーチンのための修正データ。
c)25 :計測対象となった入出力情報。
f_seq,o_lin,seq_iou,g_entry,seq_ent,seq_sio,cioustm(ioustm)
seq_iou :入出力の出現順番。
seq_sio :入出力のプログラム内出現順番。
cioustm :入出力の種別。

(19)pass4

23 ⟶27

a)23 :計測用ソースのインクルードのための基本データ。
b)27 :計測用ソースの加工されたインクルード。



c)23 :計測用ソースのライブラリ・ソースのための基本データ。

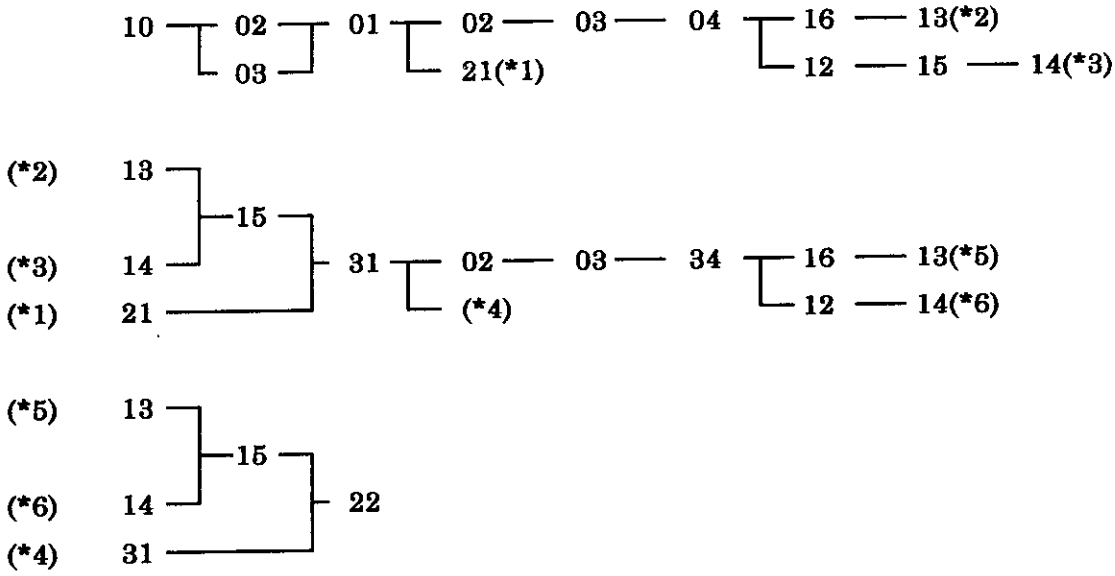
d)24 :計測対象となったプログラム情報 (pass3 から)。

e)25 :計測対象となった入出力情報 (insiou から)。

f)26 :計測対象となった DO ループ情報 (insnest から)。

g)22 :最終的に作成される計測用ソース。

5)基本ファイルの流れ



2．出力例の意味

＜出力例＞

```
 start pass 0/4
```

ライン数が５０００ラインを超える時、処理中のライン番号の情報が出力される。pass 0 から pass 2 の処理で出力される。

```
 e n d pass 0/4  cpu(sec):       0.226 total:       0.226
 start pass 1/4
 Pass1-1

 Pass1-2

 Pass1-3

 e n d pass 1/4  cpu(sec):       4.621 total:       4.847
 start pass 2/4
 e n d pass 2/4  cpu(sec):       0.353 total:       5.199
 start pass 3/4
Entry : ADDVAR                          str:       1 end:      21
Entry : CALCFN                          str:      22 end:     144
 e n d pass 3/4  cpu(sec):      17.685 total:      22.884
 start pass 4/4
 e n d pass 4/4  cpu(sec):       0.475 total:      23.360
```

＜出力例の内容の説明と機能概略＞

(1) pass 0/4

計測元ソースを読み込んで、ソース・ファイル情報と、ライン位置情報を得る。この時、マルチ FORTRAN 文（；で分割される文）を分割する処理も行う。論理機番１０から１を作成する。

(2) pass 1/4

pass 1 は pass 0 で作成されたファイルから、FORTRAN の１文が１レコードになるフ

ファイルを作成する処理である。論理機番１から４を作成している。pass 1 には、pass 1-1 から pass 1-3 の３つの処理がある。各々論理機番１から４へと作成される。

a) pass 1-1

タブ・コードが存在したら、所定の数だけブランクを追加する。インクルード行をインクルード文に変更、コメント・ラインの削除、１ラインのカラム数を取得する。

b) pass 1-2

小文字の大文字化、不要なブランクの削除、継続行を考慮した FORTRAN の１文を作成する。

c) pass 1-3

得られた FORTRAN の１文から、文の種類を得る。この文がインクルード文の場合は、所定のファイルを読み込んで、独立に pass 1-1 から pass 1-3 の処理を施し、得られた FORTRAN の１文として同様の処理を施す。インクルードの入れ子の処理も合わせて行う。

(3) pass 2/4

プログラムの入口名（主入口と副入口）、ライン位置、種別を取得する。

(4) pass 3/4

プログラム毎に計測用ソース（処理時間を計測するプログラム）を作成する。処理中の入口名が出力され、計測対象外のプログラムは skip の情報が出力される。

(5) pass 4/4

pass 3/4 で作成されたソースに、計測のためのライブラリを追加する。また、計測用ソースのためのインクルードが作成される。

3．各サブルーチンの説明

(1)input

(A)処理概要

ツール・プログラムの標準値の設定、ツールの制御データの標準値の設定および、制御データを入力する。

(B)処理の詳細の列記

a)読み込まれた制御データが、コメント行の場合は除去し、コメント・カラムの場合はブランクに置き換える。
b)タブ・コードが表われた場合は、所定の数のブランクに置き換える。
タブ・コードは、タブ・コードとそのカラム数が制御データで設定される前は、各々**char(9)**と**1**が設定されているとみなされ、計測元ソースを読み込む時、制御データが設定されていない場合は、各々**char(9)**と**8**に設定される。
c)不要なブランクを取り除く。
d)制御データは「キーワード」＝「内容」で定義されるため、キーワードと内容を分割する。
e)制御データのキーワードと内容を大文字に変更する。
f)キーワード毎に内容を解釈して、制御データを設定する。
g)制御データのEOFか制御データの終了コードが表われた時、制御データの入力を終了する。
h)制御データのファイルが存在しない場合は、すべて標準値が設定される。
i)制御データのコーディング例。
OUT_U66の計測結果のプリント・ファイルの論理機番の設定（２から７行目）、OUT_F66のファイル名の設定（８から９行目）、OPN_U66のファイルのオープン処理の設定（１０行目から）。

```
....*....1....*....2....*....3....*....4....*....5....*....6
c       - result print file
      elseif ( keyp .eq. qOUT_U66 ) then
        out_u66 = decd(buf(leq+1:nc))
        if ( out_u66 .lt. 0 .or. out_u66 .gt. 99 ) then
          nerror = nerror + 1
          write(*,6020) ndata,4,sav_buf(1:sav_nc)
        endif
```

```
      elseif ( keyp .eq. qOUT_F66 ) then
        out_f66 = buf(leq+1:nc)
      elseif ( keyp .eq. qOPN_U66 ) then
        if ( nc-leq .gt. 3 ) then
          nerror = nerror + 1
          write(*,6020) ndata,5,sav_buf(1:sav_nc)
          go to 2500
        endif
        ns = leq + 1
        ne = nc
        call low2up (keyword,buf(ns:ne))
        YesNo = 2
        if ( keyword(1:3).eq.'YES' ) YesNo = 1
        if ( keyword(1:3).eq.'NO ' ) YesNo = 0
        if ( YesNo .eq. 2 ) then
          nerror = nerror + 1
          write(*,6020) ndata,5,sav_buf(1:sav_nc)
          go to 2500
        endif
        opn_u66 = YesNo
....*....1....*....2....*....3....*....4....*....5....*....6
```

decd で数字をバイナリ変換し、low2up で小文字を大文字化する。

(C)特に説明の要する変数等

a)keyword :制御データのキーワードの設定領域。
b)inpkey(*) :制御データのキーワードで、計測元ソースの入口名等の定義データ。
inpdecl.inc でパラメータ定数とデータ文により定義している。
c)othkey(*) :制御データのキーワードで、計測元ソースの入口名等以外の定義データ
inpdecl.inc でパラメータ定数とデータ文により定義している。
d)qXXXXXX :キーワードの位置を表わすパラメータ定数。

(2)cpu_mse

(A)処理概要

ツールの処理時間を得る。 ＣＰＵ時間や経過時刻の戻り値はミリ秒単位である。インストール先の計算機に依存するため、戻り値に常に０秒を設定しているが、インストール計算機毎に修正すれば、各処理での処理時間を得ることができる。

(3)separate

(A)処理概要

統合されたソース・プログラム・ファイル、インクルード・ファイルや、計測結果のファイルの分割を行う。分割は決められた分割子が表われた場所で行う。現在、計測結果の報告方法が異なるため、この機能は使われていない。また、入力データの説明でもこの機能を省略してある。このため、この機能についての説明を省略する。

(4)pass0

(A)処理概要

計測元ソースを読み込んで、ソース・ファイルの情報と、ライン位置の情報を得る。この時、マルチ FORTRAN 文（；で分割される文）を分割する処理も行う。論理機番１０から１を作成する。

(B)処理の詳細の列記

a) 計測元ソースは、オリジナルの*.ｆ毎に第一カラムに FILENAME（*）を挿入後、*.ｆを結合したファイルである。この FILENAME よりソース・プログラムのファイル名情報を得る。
b)タブ・コードを所定のブランクに置き換える。
c)１ライン毎にファイルの順序番号、ライン位置、ライン文字数を得る。
d)a)から c)の処理でファイル機番１０から２を作成する。
e)マルチ文の処理のため、ライン毎にマルチ文の記述子（；）の位置を得る。マルチ文を判定するため、コメント・ラインを除き継続行を処理し、大文字化して FORTRAN 文を解釈する。マルチ文が表われた場合、ファイル情報、ライン位置情報、および、マルチ文の記述子のカラム位置情報をファイル機番３へ出力する。複数のマルチ文の記述子が存在する可能

性も考慮する。

f)ファイル機番２（オリジナル・ライン情報）、ファイル機番３（マルチ文情報）の比較からマルチ文を分割したソース・プログラムをファイル機番１へ出力する。

g)マルチ文の変更例

＜オリジナル・ライン＞

```
....*....1....*....2....*....3....*....4....*....5....*....6
      save=a;a=b;b=save
```

＜変更後のライン＞

```
....*....1....*....2....*....3....*....4....*....5....*....6
      save=a
           a=b
                b=save
```

マルチ文の記述子（;）をブランクに置き換えて、その記述子以降のデータを新しいラインとして出力する。１ラインを読み込み、記述子の左側のみをファイルに出力し、出力が終了したカラムとマルチ文の記述子をブランクで置き換える。これをマルチ文の記述子が終了するか、そのラインが終了するまで繰り返す。継続行を含むマルチ文の場合で、分割されたラインが開始行である場合には、継続行の第６カラムをブランクに置き換える。マルチ文で分割されたラインの位置情報は、同じライン位置情報を持つ。

h)pass0 でのファイルの処理例。

(original)

```
....*....1....*....2....*....3...
   FILENAME abcefg.f
 1       subroutine abc
 2       :
15       :
16       end
17       subroutine efg
18       :
52       :
53       end
   FILENAME funcall.f
 1       function funca(a)
 2       b=a;a=c*d
35       :
36       end
37       function funcb(a)
```

```
 38        :
124        :
125        end
(modified)
                ....*....1....*....2....*....3...
                FILENAME abcefg.f
1          1        subroutine abc
1          2        :
1         15        :
1         16        end
1         17        subroutine efg
1         18        :
1         52        :
1         53        end
                FILENAME funcall.f
2          1        function funca(a)
2          2        b=a
2          2            a=c*d
2         35        :
2         36        end
2         37        function funcb(a)
2         38        :
2        124        :
2        125        end
```

変更後のファイルの１つ目の数字はファイルの情報を表わし、２つ目の数字はライン位置を表わす。

(C)特に説明の要する変数等

a)nfil :ファイル情報の数。
b)f_seq :ファイル情報の順番。
c)o_lin :ライン位置。
e)cf_seq :修正データのファイル情報の順番。
f)co_lin :修正データのライン位置。
g)c_pos :修正データのマルチ文記述子のカラム位置。

h)complin :オリジナルのライン位置と修正後のライン位置を比較した結果にファイル順序番号を加味したもので、complin<0 以外の場合はマルチ文の修正が行われたことを表わす。

(5)pass1

(A)処理概要

pass1 は pass0 で作成されたファイルから、FORTRAN の１文が１レコードになるファイルを作成する処理である。論理機番１から４を作成している。pass1 には、pass1-1 から pass1-3 の３つの処理がある。

pass1 は単なるドライバ・ルーチンで、複数の処理（副プログラム）から呼び出される。実際の処理は pass1sub で行っている。 ここでは pass1sub について説明する。

(B)処理の詳細の列記

a)タブ・コードが存在したら、所定の数だけブランクを追加する。
b)＊INCLUDE ABC のインクルード行を include'ABC'のインクルード文へ変更し、ダブル引用符のインクルード文(include "ABC")をシングル引用符のインクルード文(include 'ABC')へ変更する。さらに、コメント・ラインの削除、１ラインのカラム数を取得する。
c)ツールのディレクティブを通常の文と同様に７カラムから始まる文に変更する。
d)ここまでについて、ファイル機番１からファイル機番２を作成する。
e)文字定数を除いた小文字の大文字化、不要なブランクの削除、継続行を考慮した FORTRAN の１文を完成する。ファイル機番２からファイル機番３を作成する。
f)得られた FORTRAN の１文から、文の種類を得る。ツールのディレクティブについては種類の分類は行わない。
g)インクルード文は、副プログラム incinc で処理を行う。副プログラム incinc では、インクルード文から所定のファイルを読み込み、それぞれのラインについて独立に pass1-1 から pass1-3 の処理を施す。この時、インクルードの入れ子の処理も合わせて行う。
h)FORTRAN の１文を１つのレコードとしてファイル機番４へ出力する。この時、論理 if 文などは、第２の FORTRAN 文が存在するため、これを考慮し文の種類を求め、第２の文を改めてファイル機番４へ出力する。

(6)pass2

(A)処理概要

プログラムの入口名（主入口と副入口）、ライン位置、種別を取得する。

(B)処理の詳細の列記

a)プログラムの主入口名と種別を取得する。
b)プログラムの副入口名を取得する。
c)プログラム毎のライン位置を取得する。

(C)特に説明の要する変数等

a)nprg	:プログラムの入口名（主入口と副入口）の数。
b)prg_nam	:プログラムの主入口名。
c)prg_ent	:プログラムの副入口名。
d)prg_inf(1,n)	:プログラムの種別。
e)prg_inf(2,n)	:ファイル機番４の開始レコード番号。
f)prg_inf(3,n)	:ファイル機番４の終了レコード番号。
g)prg_inf(4,n)	:ソース・ファイルの情報。
h)prg_inf(5,n)	:オリジナル・ライン位置情報。
i)chksub	:.true.の間は、入口名がまだ得られていない状態。

(7)pass3

(A)処理概要

プログラム毎に（処理時間を計測するプログラム）を作成する。ブロック・データ副プログラム、制御データで計測を除外した副プログラムや、ツールのディレクティブで除外指定されものをここで除外する。

(B)処理の詳細の列記

a)ファイル機番４からプログラム毎に同じ内容をファイル機番１６へ出力する。

b)ファイル機番１からプログラム毎に同じ内容をファイル機番２１へ出力する。
c)ブロック・データ副プログラムは解析しない。
d)eraseent 副プログラムにより、プログラム解析の選択を行う。
e)onesub1 副プログラムにより、プログラム毎に計測用ソースを作成する。
f)計測対象外の場合は deadcopy 副プログラムにより、ファイル２１番から最終ファイルへ複写する。

(C)特に説明の要する変数等

a)no_main　　:主プログラムの個数が最終的に１以外の時、警告メッセージを出力する。

(8)pass4

(A)処理概要

pass3 で作成されたソースに、計測のためのライブラリを追加する。この時、計測対象の個数、計測の情報などがライブラリのデータ文や、パラメータ文で宣言される。同時に計測用ソースのためのインクルードが作成される。

(B)処理の詳細の列記

a)mysystem.dat からインクルード・ファイルのデータを抽出して、計測用ソースのためのインクルード・ファイルを作成する。
b)パラメータ定数を設定する。計測対象が決定したことで、全体の計測ループ数、計測副プログラム数、計測入出力数を特定し、パラメータ定数で宣言する。これにより、計測用ソースの配列のサイズを決定する。
c)mysystem.dat からライブラリのソース・プログラム・データを抽出して、計測用ソースのためのライブラリ・ソース・プログラムを作成する。
d)回数のみを計測する計測用ソースの場合は、時間計測する部分を削除する。
e)計測用ソースのライブラリの入口名変更要求があった場合、新しい名前を設定する。
f)インクルードと同様にパラメータ定数を設定する。
g)計測対象の副プログラム名を定義するデータ文を設定する。
h)ループの順序番号、ループのラベルなど、計測結果に表われる計測対象の種々のデータを設定する。
i)ファイル機番２４から計測対象の副プログラム名情報を得る。

j)ファイル機番２６から計測対象のループ情報を得る。
k)ファイル機番２５から情報を得る。
l)計測結果を出力するファイル名や機番等を設定する。

(C)特に説明の要する変数等

a)inckey	:インクルードのためのキーワードの宣言。パラメータ定数名等を宣言。
b)keypos	:inckey の文字列と比較して一致した文字列の inckey 内の位置。 sockey の文字列と比較して一致した文字列の sockey 内の位置。
c)sockey	:ライブラリ・ソースのためのキーワードの宣言。パラメータ定数名、データ文の置換用変数名等を宣言。
d)seq_dol	:計測対象ループ数。
e)seq_iou	:計測対象入出力数。
f)seq_ent	:計測対象副プログラム数。

(9)max

(A)処理概要

FORTRAN組み込み関数で、引数の中から最大の値を求める。

(10)lnblnk

(A)処理概要

与えられた文字列の後方にある、ブランクを除く文字列の長さを求める。

(B)処理の詳細の列記

a)chr の文字列の後方からブランク以外の文字を探し、ブランク以外の文字までをその文字列の長さとする。

(C)特に説明の要する変数等

a)chr　　:与えられた文字列の文字変数

(11)その他のルーチンの概略説明

・apostr　　:引用符で定義される文字定数を得る。
・arystn　　:配列名、文番号、および第一実行文の位置を求める。
・blkcom　　:コモン・ブロック初期値の宣言。
・blkpack　　:不要なブランクとコメント・カラムの削除、小文字の大文字化。
・class　　:FORTRAN 文のクラス分け（６８種のクラス）。
・cmtdir　　:コメント行か、ベクトル並列化ディレクティブかの判定。
・cmtflg　　:コメント行かの判定。
・cmtput　　:ディレクティブを含むコメント行を一時ファイルから入力。
・cmptset　　:ディレクティブを含むコメント行を一時ファイルへ出力。
・deadcopy　　:ソース・プログラムをそのまま複写する。
・decd　　:数字列をバイナリにする（decode を行う）。
・donest　　:DO ループのネスト情報を得る。
・doparam　　:DO ループの制御変数の始値、終値、増分値の内容を得る。
・eraseb　　:不要なブランクを削除して、文字列の文字数を得る。
・erasedo　　:DO ループが選択データで計測対象指定されているかどうかを得る。
・eraseent　　:プログラムが選択データで計測対象指定されているかどうかを得る。
・eraseio　　:入出力文が選択データで計測対象指定されているかどうかを得る。
・error　　:エラー・メッセージを出力し、致命的エラーの場合には実行を終了する。
・esblk　　:不要なブランクと、コメント・カラム（！）を削除。
・fndstr　　:ブランクを除外して、文字列の中に特定の文字列が存在するかを検索。
・fndxxx　　:ブランクを含み、文字列の中に特定の文字列が存在するかを検索。
・getlstm　　:FORTRAN 文の論理機番 u21 からの入力。
・getf99　　:リスタート・データの入力（回数計測の結果入力）。
・getnxt　　:（）で囲まれた文字列や、数字列の検索。
・getu15　　:１レコードのデータの入力（論理機番 u15）。
・gi03　　:１レコードのデータの入力（論理機番 u03）。
・gi04　　:１レコードのデータの入力（論理機番 u04）。
・gi12　　:１レコードのデータの入力（論理機番 u12）。
・gi13　　:１レコードのデータの入力（論理機番 u13）。
・ifstm　　:論理および算術 IF 文の括弧の次の位置を得る。
・incinc　　:インクルード・ファイルの内容を得る。入れ子のインクルードも処置。

- inital　：初期値および、初期条件を設定する。
- insdo　：DO ループの計測ルーチンの挿入指示データ（修正データ）を作成する。
- insentry　：プログラムの計測ルーチンの挿入指示データ（修正データ）を作成する。
- inslib　：ライブラリの計測ルーチンの挿入指示データ（修正データ）を作成する。
- insnest　：DO ループ・ブロック１つの抽出と、指示データの作成を制御する。
- insiou　：入出力計測ルーチンの挿入指示データ（修正データ）を作成する。
- iouchk　：入出力文の解析と論理機番の取得。
- keywrd　：FORTRAN 文のキーワードの取得。
- lasdecl　：プログラム内の宣言文の最終的位置を取得。
- lookup　：変数・配列名の存在をチェックし、存在しなければテーブルに追加する。
- low2up　：小文字を大文字にする。
- makesoc　：ソース・プログラムを修正データにより修正する。
- moveint　：整数配列データの上方移動と下方移動。
- newlabel　：プログラムで使用されていない新しいラベル（文番号）を作成する。
- newnest　：共通 DO ループ端末の DO ループ分解を行う。
- nfblnk　：ブランク以外の文字の最初の出現位置を取得する。
- number　：数字列をバイナリに変換する。整数と実数の両方をデコード。
- onesubl　：プログラムの解析と計測のための修正データを作成。
- openinc　：インクルード・ファイルをオープンする。
- operat　：論理演算、関係演算の解析と評価を行う。
- outinf　：メッセージ・ファイルへの情報の出力。
- outu31　：１レコードのデータの出力（論理機番 u31）。
- passlsub　：副プログラム passl の制御用。
- pi02　：１レコードのデータの出力（論理機番 u02）。
- pi03　：１レコードのデータの出力（論理機番 u03）。
- pi04　：１レコードのデータの出力（論理機番 u04）。
- pi11　：１レコードのデータの出力（論理機番 u11）。
- pi12　：１レコードのデータの出力（論理機番 u12）。
- pi13　：１レコードのデータの出力（論理機番 u13）。
- plshi　：ポーランド記法の解析と解釈。
- putlstm　：１ FORTRAN 文を 72 カラムに収納してファイルに出力。
- putu31　：１レコードのデータの出力（論理機番 u13）。
- setpar　：パラメータ定数の解析と解釈。
- smlparam　：ツールの解析領域が不足した時のパラメータ定数の変更方法の出力。
- socinc　：ソースとインクルード・ファイルのツール・キーワードの変更。
- sortchr　：文字型データのソート。

- **sortint** : 整数型データのソート。
- **subnam** : プログラム名の取得。
- **tab2blk** : タブ・コードを対応する数のブランクに置き換える。
- **token** : FORTRAN のトークン分割。
- **up2low** : 大文字を小文字にする。
- **whatio** : 入出力文の種別を取得。
- **wkmerge** : 計測ルーチン作成のための修正データのレコードを併合。
- **xblksp** : x_tool のためのブランク削除。
- **xevic2** : x_tool のための演算式の解析。
- **xidecd** : x_tool のための数字列のバイナリ化。
- **xindex** : x_tool のための文字列の検索。
- **xocl_cla** : x_tool のための xocl ディレクティブのクラス分類。
- **xocl_get** : x_tool のための xocl 文の入力。
- **xocl_one** : x_tool のための 1 プログラムの処理を行う制御プログラム。
- **xocl_ps2** : x_tool のための 1 プログラム名の取得。
- **xocl_stm** : x_tool のための xocl 文の解析。

付録 B

kpxの機能追加修正

1）ユーザのプログラム・ソース・ファイルを１つにまとめずにkpxの実行を可能とする。

モニタの意見として、「通常Makefileを使っているので、Makefileを利用できるようにしてほしい。」、「*.fを一つのファイルにするとmakeが使えない。」、「サブルーチン毎のコンパイルができない。」との意見が多数あり、kpxの仕様変更が必要である。

具体的方策
※現在ユーザが分けている任意のファイル単位でkpxの実行、およびそれに続くコンパイル、リンケージを可能とする。

2）kpx独自のサブルーチン群をライブラリ化する。ただし、テスト版として開発する。

モニタの意見として、「VPPで計測用ソースのコンパイル時にインフォメーションが多出するが、kpxが未使用の変数をインクルードしているためで、なんとかならないか。」という意見があった。また、「SR2201のf90のOPT(4),EXPANDでABENDした。疑似ベクトル指定で動かないのは問題。」という意見もあり、これに対しては本項目で対応できないかも知れない。本項目はコンパイル時間の短縮にもつながる。ただし、実行時間が長くなる事態が予想され、本項目についてはテスト版とする。

具体的方策
※インストール時にディレクトリkpxの下にlibというライブラリを設け、そこにkpxで使用する全ての独自サブルーチンを登録する。だたし、ユーザはプログラム・リンケージ時にここを参照しなければならない。

3）ktool出力フォーマットを80文字以内に制限し、内容も変更する。

モニタの意見として、「見やすいサイズのフォントでA4プリント出力した時、一行で完結して欲しい。」、「ktoolの出力は横に長すぎる。」、「コラム値は通常80を採用しているので、80コラムの方が見やすい。」、「一行であるべき行が、画面の右で折り返して２行になる。」、「出力の文字数は１行80文字にしてほしい。」等があった。また、「ktoolの出力はサブルーチン毎にdoループを集計してほしい。」、「doループ表示等をサブルーチン毎に表示してほしい。」という意見がありこれにも対応する。さらに、「ktoolの出力結果に論理機器番号の情報がほしい。」という意見もあり内容を変更する。

具体的方策
※ktool出力フォーマットを表１に示す。「-----*...-----8」は空行とする。

```
----*----1----*----2----*----3----*----4----*----5----*----6----*----7----*----8
<<< PROGRAM UNIT RESULTS >>>
PROGRAM     TIMES  TIME:SEC    % LINE FILE/*.f
------- --------- --------- ---- ---- --------
VDFT3F       1200    15.824 45.3    1 vdft3f
PUTFLD          5     7.385 21.1    1 putfld
VDFT3B        600     7.165 20.5    1 vdft3b
 ========== 省略  ==========
VDFT3I          1     0.000 0.00    1 vdft3i
              ------------
                    34.881
-----*----1----*----2----*----3----*----4----*----5----*----6----*----7----*----8
<<< I/O PROCESS RESULTS >>>
PROGRAM SEQ. UNIT NO. KIND      TIMES TIME:SEC    % LINE FILE/*.f
------- ---- -------- ----- --------- -------- ---- ---- --------
PUTFLD     7       11 close         5    3.897 47.4   28 putfld
PUTFLD     6       12 write         5    3.163 38.5   19 putfld
GETFLD     2       13 open          1    0.420  5.1   12 getfld
 ========== 省略  ==========
GETFLD     4       14 write         0    0.000  0.0   18 getfld
                                 ------------
                                       8.204
----*----1----*----2----*----3----*----4----*----5----*----6----*----7----*----8
<<< DO LOOP RESULTS >>>
PROGRAM  SEQ. LABEL T   ROTATION MEAN:ROT      TIMES TIME:SEC    % LINE FILE/*.f
-------- ---- ----- - ---------- -------- --------- -------- ---- ---- --------
VDFT3F     32   920 1    1228800     16.0     76800    2.316  8.1  166 vdft3f
VDFT3F     29   918 1    1228800     16.0     76800    2.301  8.1  148 vdft3f
VDFT3F     16   910 1    1228800     16.0     76800    1.948  6.8   81 vdft3f
========== 省略  ==========
VDFT3I      5   210 1          5      5.0         1    0.000  0.0   40 vdft3i
                                              ------------
                                                    28.274
----*----1----*----2----*----3----*----4----*----5----*----6----*----7----*----8
<<< DO LOOPS IN SUBROUTINES >>>
PROGRAM  SEQ. LABEL T   ROTATION MEAN:ROT      TIMES TIME:SEC    % LINE FILE/*.f
-------- ---- ----- - ---------- -------- --------- -------- ---- ---- --------
VDFT3F     32   920 1    1228800     16.0     76800    2.316  8.1  166 vdft3f
VDFT3F     29   918 1    1228800     16.0     76800    2.301  8.1  148 vdft3f
VDFT3F     16   910 1    1228800     16.0     76800    1.948  6.8   81 vdft3f
========== 省略  ==========
VDFT3I      5   210 1          5      5.0         1    0.000  0.0   40 vdft3i
                                              ------------
                                                    28.274
```

表1　ktool出力フォーマット

4）ptoolおよびxtoolで全オーバーヘッドのノード数による平均値を出力する。

モニタの意見として、「ptoolで全オーバーヘッドのノード数による平均が欲しい。」との意見があり、ptoolだけでなくxtoolについても対応する。

具体的方策
※全オーバーヘッドを合計し、使用ノード数で割った値を出力する。

5）ptoolおよびxtoolでデータ採取対象の制限を、対応する制御データ（p_tool.datまたはx_tool.dat）で指定可能にする。

モニタの意見として、「Paragonの時間計測ルーチンdclockをptoolではディフォルトで測定対象外にした方が良い。」との意見があった。よって、ptoolおよびxtoolの制御データによりサブルーチン、doループのデータ採取制限を可能とする。

具体的方策
※ktoolは制御データ（k_tool.dat）によりサブルーチン、doループおよび入出力の制限が可能であるが、この内、サブルーチン、doループについてのみ同様の機能をptoolおよびxtoolに拡張する。

6）ktoolの出力結果をグラフ化する。

モニタの意見として、「Paragraphは各種GUIを使って見やすく情報提供してくれるのに対し、kpxはファイル出力しかサポートしていない。」、「GUIがほしい。」、「CRAYではGUIが付随して解析しやすい。」、「GUIがあればよいと思う。」、「対象範囲の操作にビジュアル化が必要（静的（動的がベスト）な手続き構造を見せてこの下は対象とするしない等）。」との意見があった。GUIインタフェースでの操作環境の実現は理想であるが、今回は特にktoolの出力結果のグラフ化を図る。

具体的方策
※コマンドkviewを入力することにより、X Window Systemsのグラフィカル画面を出力する。表示およびメニューは全て英語を用いること。出力例を示すが、より程度の良いものであればよい。また、出力項目数等はリサイズに対応することとする。

A）ウインドウの画面仕様
Xのグラフィカル画面は「Analyzer kpx main viewer」を含め、全部で８画面とする。

「Analyzer kpx main viewer」
データ・ファイルを読み込む機能を有し、サブルーチンの時間コスト情報を出力する。（サブルーチン名を項目とし、TIME:SECを棒グラフ化）図１に出力例を示す。
起動法：コマンドkviewで立ち上げる。
ウインドウ１
サブルーチンの実行回数コスト画面（サブルーチン名を項目とし、TIMESを棒グラフ化）図２に出力例を示す。
起動法：「Analyzer kpx main viewer」の「Execution Times」をクリックする。
ウインドウ２
対応するサブルーチンのみのDoループ時間コスト画面（FILE/*.f：LINEを項目とし、TIME:SECを棒グラフ化）
起動法：「Analyzer kpx main viewer」のグラフの棒をクリックする。

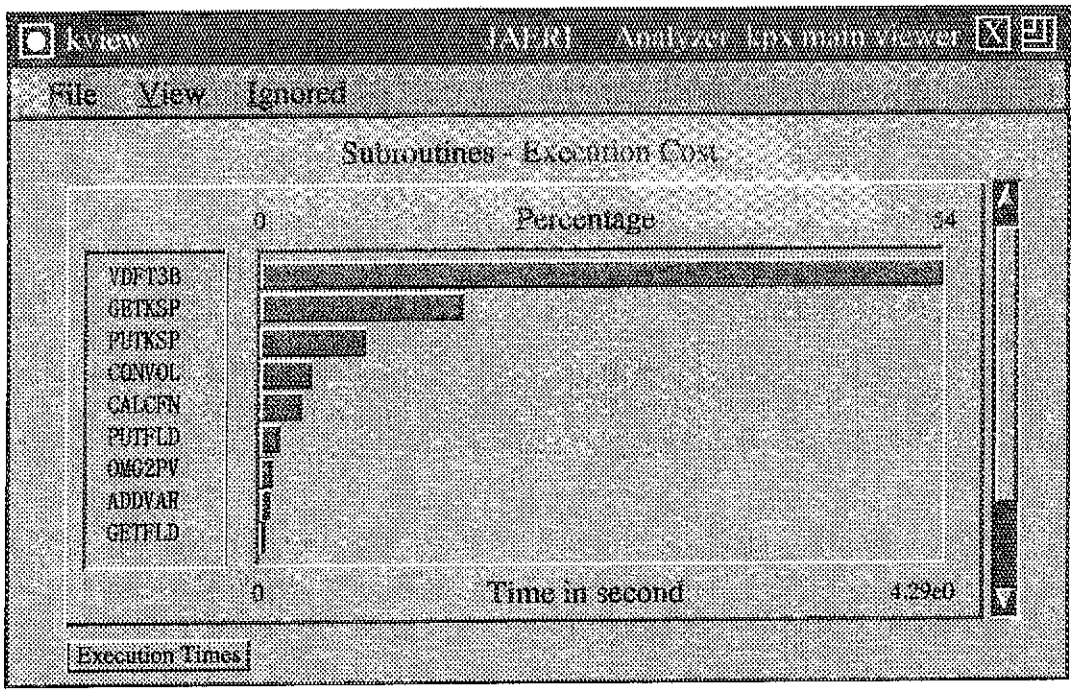


図1　Analyzer kpx main viewer

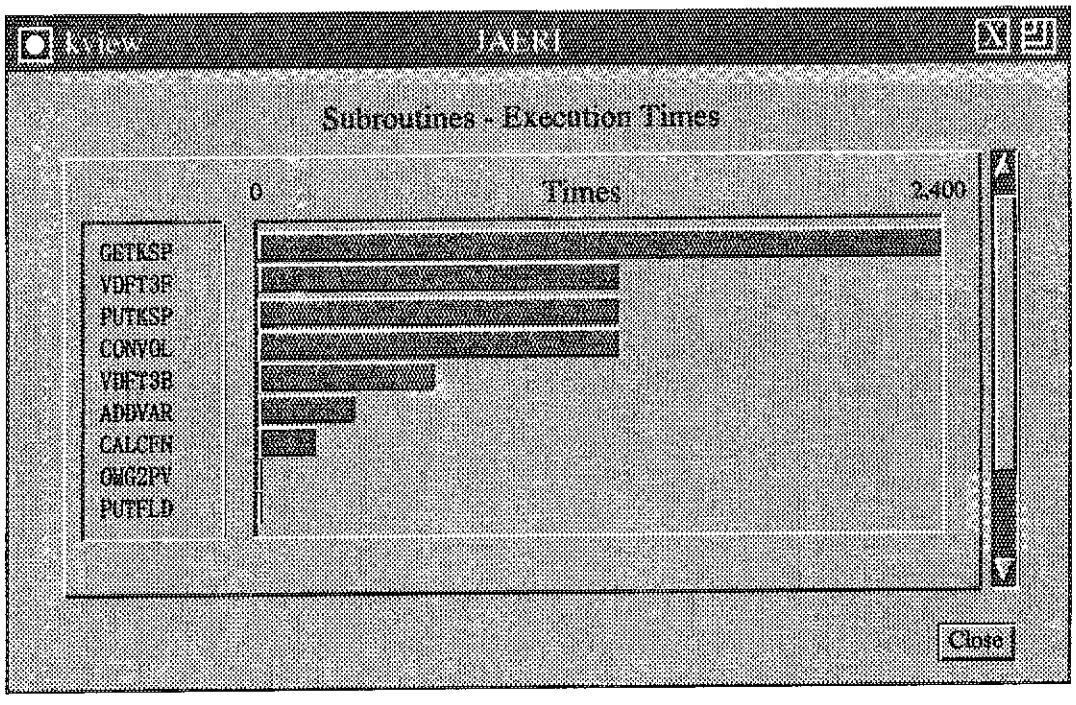


図2　Subroutines - Execution Times

ウインドウ３
対応するサブルーチンのみのDoループ実行回数コスト画面（FILE/*.f：LINEを項目とし、ROTATION、TIMESを棒グラフ化）
起動法：ウインドウ２の「Execution Times」をクリックする。
ウインドウ４
Doループ時間コスト画面（FILE/*.f：LINEを項目とし、TIME:SECを棒グラフ化）
起動法：「Analyzer kpx main viewer」の「View」の「Do Loops」を選択する。
ウインドウ５
Doループ実行回数コスト画面（FILE/*.f：LINEを項目とし、ROTATION、TIMESを棒グラフ化）
起動法：ウインドウ４の「Execution Times」をクリックする。
ウインドウ６
入出力時間コスト画面（FILE/*.f：LINE：KIND：UNIT.NOを項目とし、TIME:SECを棒グラフ化）
起動法：「Analyzer kpx main viewer」の「View」の「Input/Output」を選択する。
ウインドウ７
入出力実行回数コスト画面（FILE/*.f：LINE：KIND：UNIT.NOを項目とし、TIMESを棒グラフ化）
起動法：ウインドウ６の「Execution Times」をクリックする。
ウインドウ８
測定対象外となったサブルーチンの一覧表画面
起動法：「Analyzer kpx main viewer」の「Ignored」の「Subroutines」を選択する。
ウインドウ９
測定対象外となったdoループの一覧表画面
起動法：「Analyzer kpx main viewer」の「Ignored」の「Do Loops」を選択する。
ウインドウ10
測定対象外となった入出力の一覧表画面
起動法：「Analyzer kpx main viewer」の「Ignored」の「Input/Output」を選択する。

B）「Analyzer kpx main viewer」のメニュー仕様
「Analyzer kpx main viewer」で定義されるメニュー仕様について記述する。

「File」
「Open」
データはkviewの「File」メニューで指定できること。その時、kviewを起動したディレクトリ名とディフォルトのファイル名を出力し、データ・ファイル名の入力を補助すること。データ・ファイルが正常であれば、「Analyzer kpx main viewer」にサブルーチンの時間コスト情報を出力する。その時のスケールの最大値は最大コストのサブルーチンのパーセンテージを使用する。
「Quit」
「Quit」をセレクトした場合はkviewを終了する。他のウインドウが開かれていればそれも閉じる。
「View」
「Do Loops」を選択することにより、ウインドウ４を立ち上げる。
「Input/Output」を選択することにより、ウインドウ６を立ち上げる。
「Ignored」
「Subroutines」を選択することにより、ウインドウ８を立ち上げる。
「Do Loops」を選択することにより、ウインドウ９を立ち上げる。
「Input/Output」を選択することにより、ウインドウ１０を立ち上げる。

C) その他

「Execution Times」
「Analyzer kpx main viewer」およびウインドウ２、４、６の「Execution Times」をクリックすことにより、対応する実行回数コスト画面を立ち上げる。

「Close」
ウインドウ１～７の「Close」をクリックすることにより、対応するウインドウを閉じること。「Analyzer kpx main viewer」にはこのボタンは設けない。

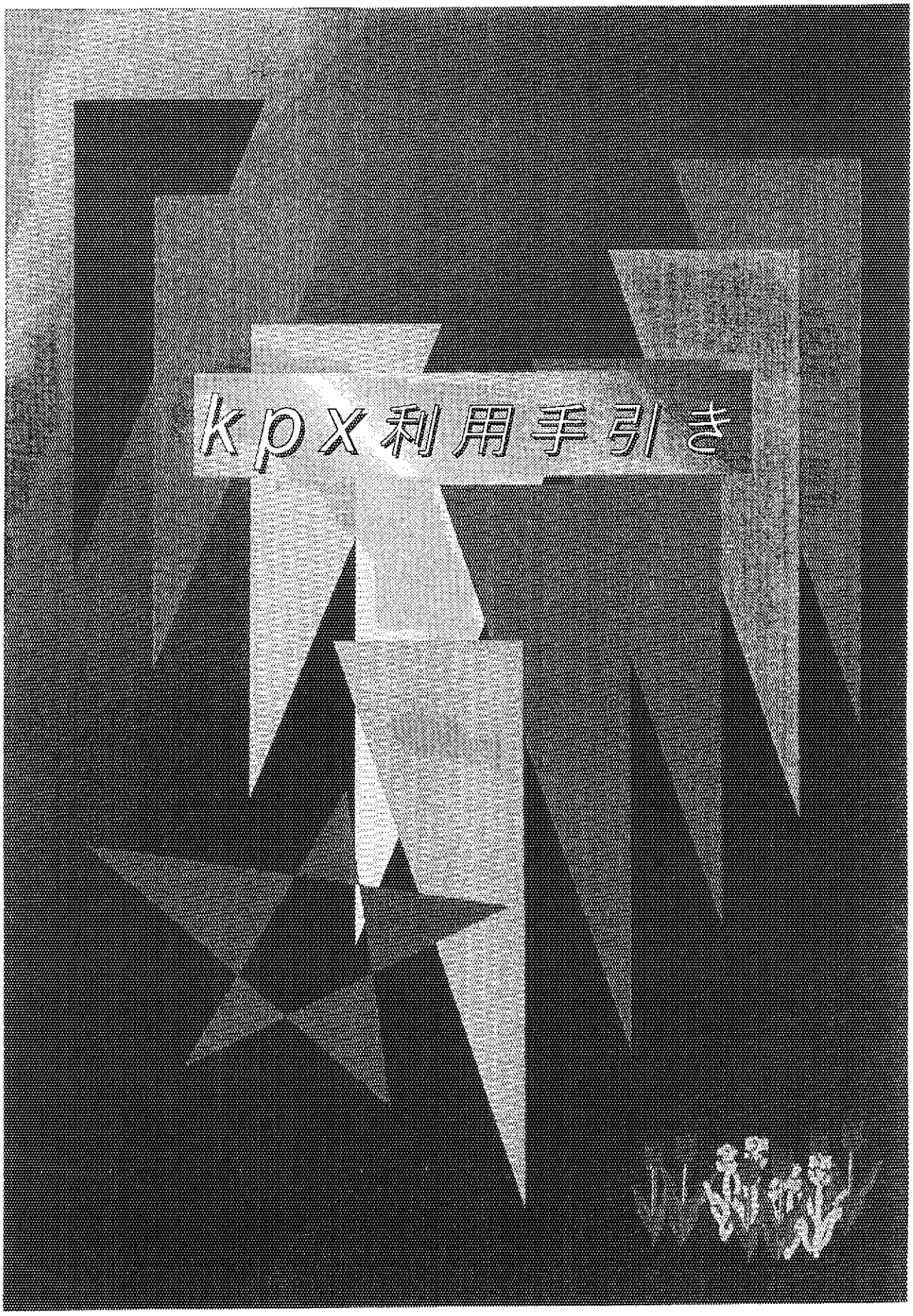
スクロール・バー
スクロール・バーでスクロールできること。

スケール
スクロール等により画面に変化がある場合、常に一番上の項目が持つ値をグラフ・メモリの最大値とし、グラフとグラフのスケールを動的にアレンジすること。
ただし、この動的変更方式に利用上または感覚上の不具合が生じた場合は、ボタンを設けこのボタンをクリックしたときのみスケールをアレンジすること。

項目数
初期画面のディフォルトで出力する項目数は見やすい範囲でできるだけ多くの情報を出力すること。

以上

付録C


kpx利用手引き

プログラム並列化支援解析ツール

kpx

１９９６年１０月１日

日本原子力研究所
計算科学技術推進センター

目次

kpxとは

kpxは、プログラムを解析するツールで、日本原子力研究所の計算科学技術推進センターが、並列処理推進のための共通基盤として開発したフリーウェアです。kpxはktool、ptool、xtoolから構成されており、以下のような機能を持っています。

ktool　ユーザが指定した時間計測ルーチンにより、プログラム、サブルーチン、doループ、および入出力の実行に要した時間の計測を行う。また、実行回数について、プログラム、サブルーチン、doループ、および入出力回数の計測を行う。

ptool　Paragonのnxライブラリにより並列化されたプログラムの、ノード間通信等の並列化オーバヘッドを測定する。

xtool　VPPの並列化ディレクティブにより並列化されたプログラムの、PE間通信等の並列化オーバヘッドを測定する。

kpxは、ユーザ・プログラムにカウンタと時間計測ルーチンを挿入することにより計測を行います。したがって、ユーザ・プログラムにカウンタや時間計測ルーチンを挿入した計測用ソースを作成した後、計測用ソースをコンパイル、実行することにより測定結果を出力します。

kpxのktoolは、全てのUNIX系コンピュータのFORTRANプログラムを対象としており、Paragon、SP2、SR2201、VPP500、VPP300、Monte-4、SX-4、T90において動作確認されています。kpxは、ソースのtarファイルで供給され、ユーザがホーム・ディレクトリにシステムを構築して利用する方法を想定しています。

本書ではkpxのインストール方法とktool、ptool、xtoolの利用法について解説します。

2 インストール

日本原子力研究所、計算科学技術推進センターのccsemfa（ファイル・サーバ）より、tarファイルを対象コンピュータのホーム・ディレクトリへ取り込みます。

```
$ cd
$ ftp ccsemfa
Connected to ccsemfa.
220 ccsemfa FTP server  (UNIX (r)  System V Release 4.0)  ready.
Name  (133.53.188.11:j007):  j007                          idがj007の場合
Password: xxxxxx                                           パスワードがxxxxxxの場合
230 User j007 logged in.
ftp>cd /document/kpx
ftp>bi
200 Type set to I.
ftp>get kpx.tar
local: kpx.tar remote: kpx.tar
200 PORT command successful.
150 ASCII data connection for kpx.tar  (133.53.188.11,1324) .
226 Transfer complete.
879246 bytes sent in 1.8 seconds  (486.24 Kbytes/s)
ftp>bye
221 Goodbye.
```

ホーム・ディレクトリへの転送が終了したら、kpx.tarを解凍しinstallerを実行します。installerを実行すると、ホーム・ディレクトリにディレクトリkpxが作成され、内容を展開し、最後にホーム・ディレクトリのtool.tar、installerを消去すると共に、kpx.tar、READMEをディレクトリkpxに移動します。kpxはソースで供給されますので、installerは単にファイル・シルテムを構築し、ファイルを保存するだけでなく、ターゲットで現在使用されているバージョンのコンパイラ等を使用してロード・モジュールを作成します。installerは引き数として対象コンピュータ名（Paragon、SP2、SR2201、VPP500、VPP300、Monte-4、SX-4、T90）を与えて起動します。

```
$ tar xvof kpx.tar
x README, 3860 bytes, 8 tape blocks
x installer, 8976 bytes, 18 tape blocks
x tool.tar, 917504 bytes, 1792 tape blocks
$ csh installer VPP300                                     対象システムがVPP300の場合
```

以上の作業でktoolと、VPP300およびVPP500ではxtool、Paragonではptoolがインストールされました。

※ VPP500ではGSPで作業を行います。

3 ファイル構成

kpxのファイル・システムは、ツール名の基本ディレクトリと、ツール名の最初の１文字にdoを付けたユーザ・テスト用ディレクトリから構成されます。installerを実行した後のkpxファイル・システムを示します。

```
$ ls -l kpx/ktool                                                    ktool 基本ディレクトリ
total 4550
-rwx------   1 j9364    g0188         72 Sep  4 16:25 CatK            cat シェル・スクリプト
-rw-r--r--   1 j9364    g0188       3255 Sep  4 16:25 k_tool.hlp      ktool ヘルプ
-rw-r--r--   1 j9364    g0188         39 Sep  4 16:25 k_tool1.dat     ktool 制御データ実行回数計測用
-rw-r--r--   1 j9364    g0188         81 Sep  4 16:25 k_tool2.dat     ktool 制御データ実行時間計測用
-rw-------   1 j9364    g0188       1150 Sep  4 14:24 kgo            ktool 利用シェル・スクリプト
-rwxr-xr-x   1 j9364    g0188    2222100 Sep  4 16:26 ktool          ktool ロード・モジュール
-rw-r--r--   1 j9364    g0188      49738 Sep  4 16:26 myclock.dat    ktool システム用データ CPU 時間計測用
-rw-r--r--   1 j9364    g0188      49816 Sep  4 16:26 mysystem.dat   ktool システム用データ経過時間計測用
$ ls -l kpx/kdo                                                    ktool ユーザ・テスト用ディレクトリ
total 199
-rw-------   1 j9364    g0188       1150 Sep  4 16:26 kgo            ktool 利用シェル・スクリプト
-rw-r--r--   1 j9364    g0188      49738 Sep  4 16:26 myclock.dat    ktool システム用データ CPU 時間計測用
-rw-r--r--   1 j9364    g0188      49816 Sep  4 16:26 mysystem.dat   ktool システム用データ経過時間計測用
```

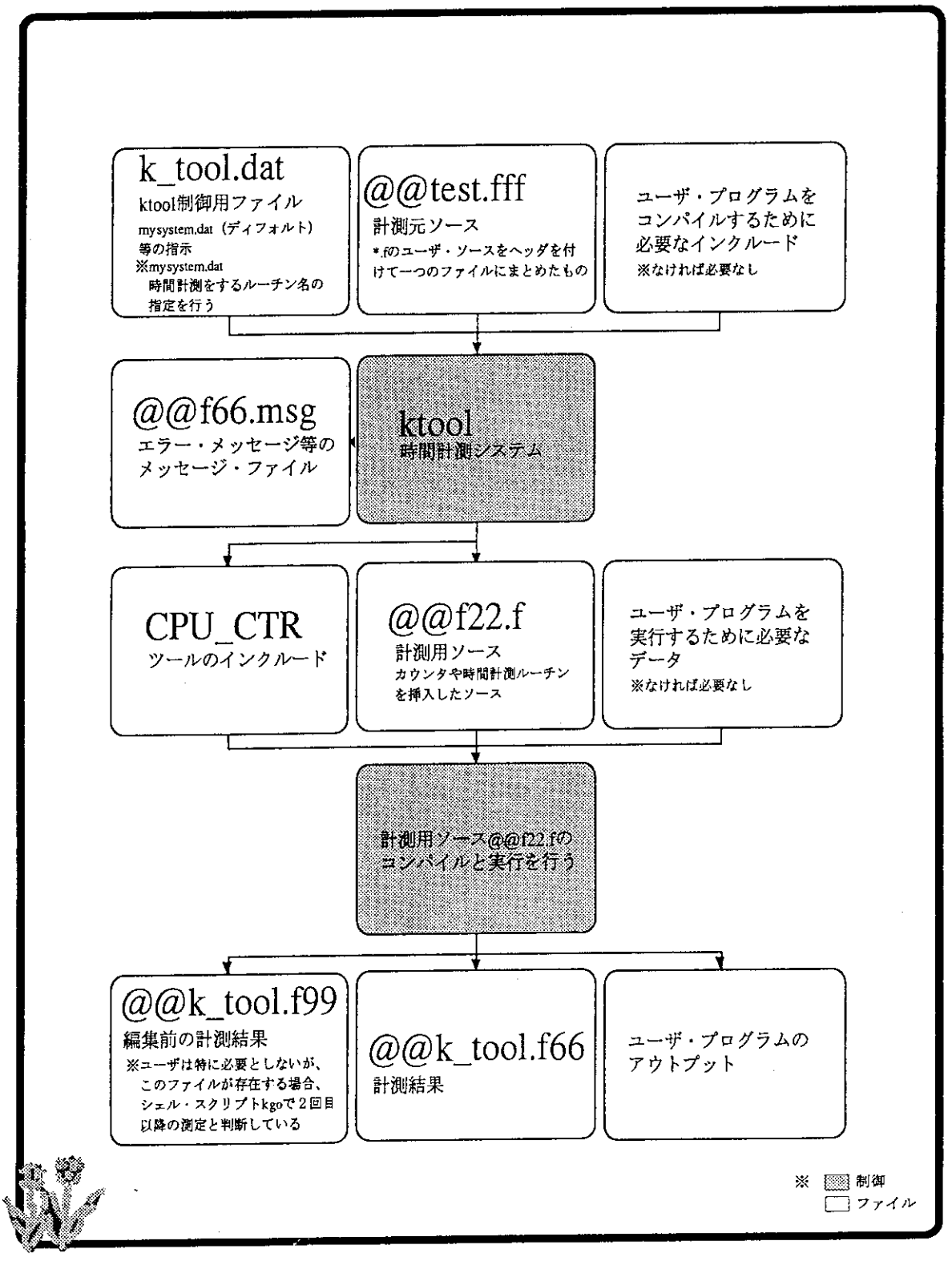
Paragonではptool関連ファイルが追加作成されます。

```
Paragon$ ls -l kpx/ptool                                           ptool 基本ディレクトリ
total 1762
-rwx------   1 j9364    grp05         72 Sep  3 22:30 CatK            cat シェル・スクリプト
-rw-------   1 j9364    grp05      19516 Sep  3 22:30 p_system.dat    ptool システム用データ
-rw-------   1 j9364    grp05        408 Sep  3 19:36 p_tool.dat      ptool 制御データ
-rw-------   1 j9364    grp05       1956 Sep  3 22:30 p_tool.hlp      ptool ヘルプ
-rw-------   1 j9364    grp05        302 Sep  3 21:06 pgo             ptool 利用シェル・スクリプト
-rwx------   1 j9364    grp05     862311 Sep  3 22:33 ptool           ptool ロード・モジュール
Paragon$ ls -l kpx/pdo                                             ptool ユーザ・テスト用ディレクトリ
total 42
-rw-------   1 j9364    grp05      19516 Sep  3 22:33 p_system.dat    ptool システム用データ
-rw-------   1 j9364    grp05        316 Sep  4 11:08 pgo             ptool 利用シェル・スクリプト
```

VPPではxtool関連ファイルが追加作成されます。

```
VPP$ ls -l kpx/xtool                                               xtool 基本ディレクトリ
total 6208
-rwx------   1 j9364    g0188         72 Sep  4 16:26 CatK            cat シェル・スクリプト
-rw-r--r--   1 j9364    g0188      21163 Sep  4 16:26 x_system.dat    xtool システム用データ
-rw-------   1 j9364    g0188        253 Sep  3 19:36 x_tool.dat      xtool 制御データ
-rw-r--r--   1 j9364    g0188       2048 Sep  4 16:26 x_tool.hlp      xtool ヘルプ
-rw-------   1 j9364    g0188        735 Sep  4 11:06 xgo             xtool 利用シェル・スクリプト
-rwxr-xr-x   1 j9364    g0188    2957916 Sep  4 16:29 xtool           xtool ロード・モジュール
VPP$ ls -l kpx/xdo                                                 xtool ユーザ・テスト用ディレクトリ
total 128
-rw-r--r--   1 j9364    g0188      21163 Sep  4 16:29 x_system.dat    xtool システム用データ
-rw-------   1 j9364    g0188        735 Sep  4 16:29 xgo             xtool 利用シェル・スクリプト
```

4.1 ktool システム概要



4.2 ktool実行方法

ディフォルトのシステムでは、Paragon、SP2、SR2201、VPP500、VPP300、Monte-4、SX-4、T90について経過時間が測定できるようにセッティングされています。ktoolの実行は、以下のようにして行います。

1．ユーザ・ソースと、そのコンパイルと実行に必要な資源をkpx/kdoにコピーします。
※メイン・ルーチンは一本で、ソース・ファイル名は*.fを想定しています。
2．ソース*.fをヘッダを付けて1本にまとめ、計測元ソース@@test.fffとします。
3．ktoolを実行してソースに計測ルーチンを挿入し、新たな計測用ソース@@f22.fを作成します。
4．@@f22.fをコンパイル後、実行することにより、計測結果のファイル@@k_tool.f66が出力されます。

以上ですが、実行時間を短縮するために、まず、実行回数のみの計測を行い、次に実行回数による制限付きの時間計測を行うと、作業が効率的に終了することがあります。その場合、上記実行手順3から4までを2度繰り返すことになります。このようなktoolの実行方法について、上記実行手順2以降の、標準的な会話型での実行を想定したシェル・スクリプトkgoを用意しました。以下にkgoの内容を示します。

```
$ cat kpx/kdo/kgo
#!/bin/csh -f                                                    特に必要なし
cd $HOME/kpx/kdo                                                 特に必要なし
#-------+---------+---------+---------+---------+---------+---------+
rm -f @@* CPU_*       # 1st execution if @@k_tool.f99 is not exists.   @@k_tool.f99が無ければ一回目の実行
../ktool/CatK *.f > @@test.fff                                   計測元ソースを作成
##### 1st for counting execution times
cp ../ktool/k_tool1.dat k_tool.dat                               k_tool.datで実行回数測定指示
../ktool/ktool                                                   計測用ソースを作成..... 注1
rm @@f*.wrk                                                      不要なファイルの消去
#####################################################################
##### Probably you have to change next two lines to fit.
##### Which compiler will you use? Dose your program need input?
fortran @@f22.f                                                  コンパイル..... 注2
./a.out < input                                                  実行回数測定..... 注3
#####################################################################
rm @@f22.f @@f28.inf @@k_tool.f66                                不要なファイルの消去
###### 2nd for measuring time
cp ../ktool/k_tool2.dat k_tool.dat                               k_tool.datで実行時間測定指示
../ktool/ktool                                                   計測用ソースを作成..... 注1
rm @@f*.wrk                                                      不要なファイルの消去
#####################################################################
##### Probably you have to change next two lines to fit.
##### Which compiler will you use? Dose your program need input?
fortran @@f22.f                                                  コンパイル..... 注2
./a.out < input                                                  実行時間測定..... 注3
#####################################################################
```

kgoの実行は、上記実行手順1の終了後、必要に応じてkgoを変更したあと、対象システムの$HOME/kpx/kdoで以下を入力します。

```
$ csh kgo
```

※ 注1 基本的に実行は全て対象システムで行いますが、VPP500の場合はGSPで実行します。
注2 ターゲット依存ですので書換が必要です。VPP500のコンパイルはGSPで行います。
注3 ユーザ・プログラム依存ですので書換が必要です。VPP500の場合はサブミットします。
実行時に機器番号66と99を使用します。

4.3 ktool カスタマイズ

ktoolをカスタマイズして使用するには、実行シェルkgoの中でディレクトリktoolからコピーしている、ktoolの入力ファイルk_tool.datを変更するか、ソース・ファイルを変更します。k_tool.datの変更については「付録１．ktool制御データ」を、ソース・ファイルの変更については「付録２．ktoolディレクティブ」を参照してください。ここでは、機能の一部をご紹介いたします。k_tool.datのサンプルとして、実行回数計測用k_tool1.datと実行時間計測用k_tool2.datがあります。

```
$ cat kpx/ktool/k_tool1.dat          実行回数計測用データ・サンプル
mysystem = mysystem.dat                mysystem.datは経過時間計測用、myclock.datはCPU時間計測用
jst_ctr  = yes                         実行回数のみを計測する

$ cat kpx/ktool/k_tool2.dat          実行時間計測用データ・サンプル
mysystem = mysystem.dat                mysystem.datは経過時間計測用、myclock.datはCPU時間計測用
max_ent  = 10000                       10,000回以内で呼び出されるエントリを時間計測対象とする
max_iou  = 10000                       10,000回以内の入出力命令を時間計測対象とする
max_dol  = 10000                       10,000回以内で実行されるdoループを時間計測対象とする
```

１．時間計測ルーチンの変更

新種のコンピュータ上でktoolを使用したい場合や、特別な時間計測ルーチンを使用したい場合はmysystem.datをエディタで編集します。「付録３．時間計測ルーチン」を参照してください。ただし、CPU時間を計測するにはmyclock.datが用意されていますので、ktoolの入力ファイルk_tool.datで、経過時間計測用のmysystem = mysystem.datを以下のようにCPU時間計測用に変更します。

```
mysystem = myclock.dat
```

２．Fortran論理機器番号の変更

ktoolでは結果出力のため、Fortran論理機器番号66番と99番を使用しています。ユーザ・プログラムとの競合を避るには、ktoolの入力ファイルk_tool.datで機器番号の指定変更を行います。66番を43番へ、99番を47番へそれぞれ変更したい場合は以下のように指定します。

```
out_u66 = 43
out_u99 = 47
```

３．実行回数による時間計測制限の変更

doループの実行回数による時間計測制限は、ディフォルトではmax_dol = 0で無制限です。時間計測の対象を100,000回以内で実行されるdoループのみに制限したい場合は、ktoolの入力ファイルk_tool.datで以下のように指定します。

```
max_dol = 100000
```

4.4 ktool出力結果

エントリの情報

```
<<< PROGRAM UNIT RESULTS >>>
順位  プログラム名                    実行回数      実行時間  平均実行時間  レート  行番号  ファイル番号
RANK  PROGRAM NAME                    TIMES      TIME (SEC)    MEAN (SEC)  RATE (%) LINE NO. FILE
---- ----*----1----*----2----*----3-- --------- ------------ ------------ ------- -------- ----
   1 VDFT3F                             1200     15.824663   0.013187219  45.367        1   15
   2 PUTFLD                                5      7.385782   1.477156369  21.174        1   11
   3 VDFT3B                              600      7.165257   0.011942095  20.542        1   14
========== 省略 ==========
  16 VDFT3I                                1      0.000071   0.000071358   0.000        1   16
                                                ----------
                                                34.881328
                                              合計実行時間
```

入出力の情報

```
<<< I/O PROCESS RESULTS >>>
順位  プログラム名                    出現順位  種別  実行回数      実行時間  平均実行時間  レート  行番号  ファイル番号
RANK  PROGRAM NAME                     SEQ.    KIND  TIMES      TIME (SEC)    MEAN (SEC)  RATE (%) LINE NO. FILE
---- ----*----1----*----2----*----3-- -------- ----- --------- ------------ ------------ ------- -------- ----
   1 PUTFLD                                7  close         5      3.897189   0.779437804  47.498       28   11
   2 PUTFLD                                6  write         5      3.163304   0.632660754  38.553       19   11
   3 GETFLD                                2  open          1      0.420309   0.420308792   5.123       12    5
========== 省略 ==========
  22 GETFLD                                4  write         0      0.000000   0.000000000   0.000       18    5
                                                                  ----------
                                                                   8.204972
                                                                 合計実行時間
```

doループの情報

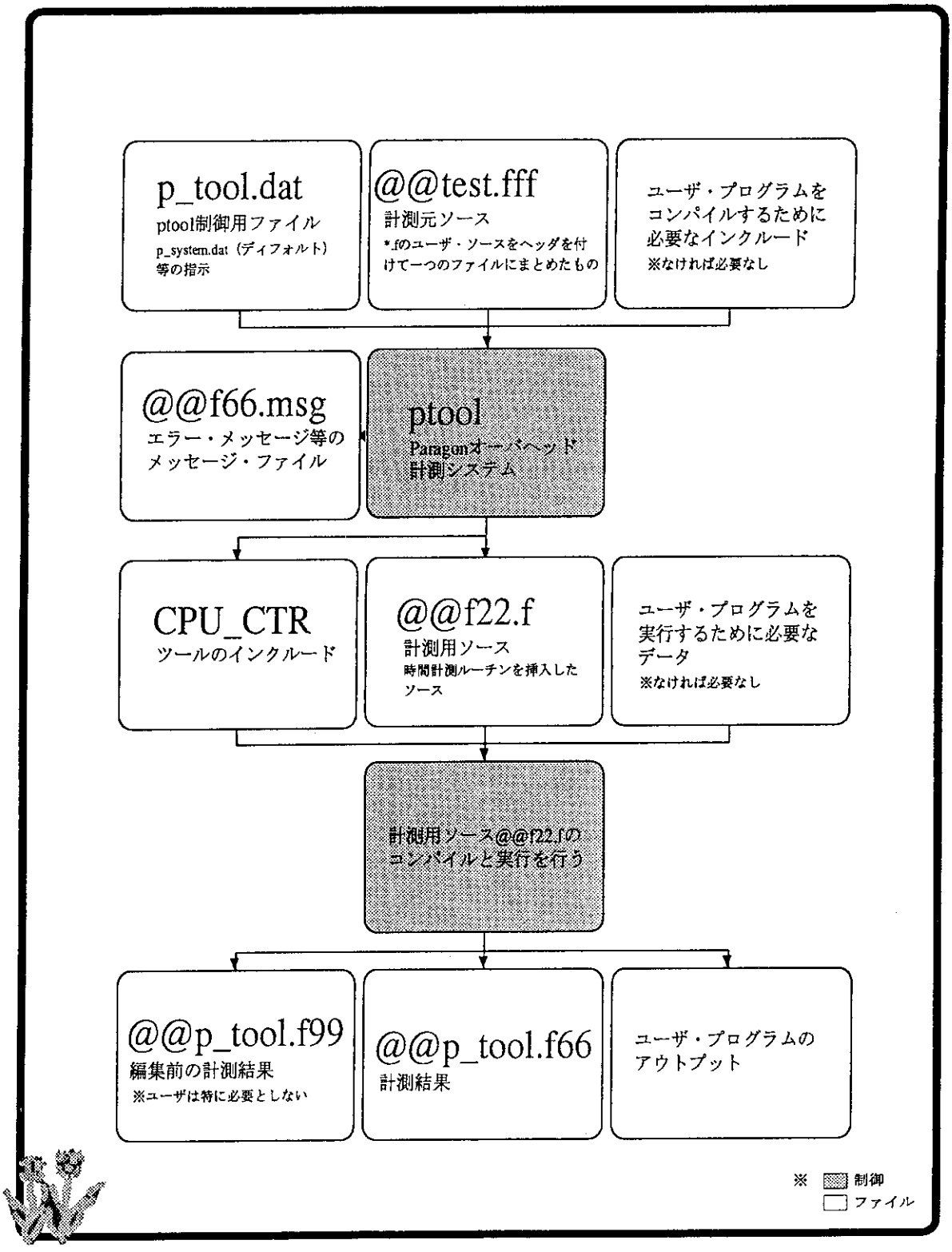
```
<<< DO LOOP RESULTS >>>
順位  プログラム名                    出現順位  ラベル  実行回数      純回転数  平均回転数      実行時間  平均実行時間  レート  T  行番号  ファイル番号
RANK  PROGRAM NAME                     SEQ.    LABEL   TIMES      ROTATION    MEAN  (ROT)  TIME (SEC)    MEAN (SEC)  RATE (%)    LINE NO. FILE
---- ----*----1----*----2----*----3-- -------- ------ --------- ------------ ---------- ------------ ------------ ------- - -------- ----
   1 VDFT3F                               32    920      76800     1228800.       16.0     2.316751   0.000030166   8.194 1      166   15
   2 VDFT3F                               29    918      76800     1228800.       16.0     2.301093   0.000029962   8.139 1      148   15
   3 VDFT3F                               16    910      76800     1228800.       16.0     1.948373   0.000025369   6.891 1       81   15
========== 省略 ==========
 136 VDFT3I                                5    210          1           5.        5.0     0.000000   0.000000000   0.000 1       40   16
                                                                                          ----------
                                                                                          28.274056
                                                                                        合計実行時間

                                                                                  doループの回転数
                                                                                  1 : 確定
                                                                                  0 : 不確定
```

ソース・ファイルの情報

```
<<< SOURCE FILE NAME (S)  >>>
ファイル番号  ソース・ファイル名
FILE                SOURCE FILE NAME
---- ----*----1----*----2----*----3----*----4----*----5.
   1 addvar.f
   2 calcfn.f
   3 convol.f
========== 省略 ==========
  16 vdft3i.f
```

※ 報告時間の単位は秒。
※ レートは消費時間百分率。
※ 特定のエントリやdoループで時間測定が抑制された場合、基本的に上位のエントリまたはdoループにその消費時間を加算する。

5.1 ptoolシステム概要



5.2 ptool実行方法

ptoolは、Paragonのnxライブラリにより並列化されたプログラムの、ノード間通信等の並列化オーバヘッドを測定するツールです。ptoolの実行は、以下のようにして行います。

1．ユーザ・ソースと、そのコンパイルと実行に必要な資源をkpx/pdoにコピーします。
　※メイン・ルーチンは一本で、ソース・ファイル名は*.fを想定しています。
2．ソース*.fをヘッダを付けて1本にまとめ、計測元ソース@@test.fffとします。
3．ptoolを実行してソースに計測ルーチンを挿入し、新たな計測用ソース@@f22.fを作成します。
4．@@f22.fをコンパイル後、実行することにより、オーバヘッドの計測結果のファイル@@p_tool.f66が出力されます。

ptoolの実行方法について、上記実行手順2以降の、標準的な会話型での実行を想定したシェル・スクリプトpgoを用意しました。以下にpgoの内容を示します。

```
Paragon$ cat kpx/pdo/pgo
#! /bin/csh -f                                          特に必要なし
cd $HOME/kpx/pdo                                       特に必要なし
#--------+---------+---------+---------+---------+---------+
rm -f @@* CPU_*                                        不要なファイルの消去
../ptool/CatK *.f > @@test.fff                         計測元ソースを作成
cp ../ptool/p_tool.dat p_tool.dat                      p_tool.dat 制御データ・ファイルを設定
../ptool/ptool                                         計測用ソースを作成
rm @@f*.wrk                                            不要なファイルの消去
##### Change next two lines to fit.
if77 -nx -O3 @@f22.f                                   コンパイル..... 注1
./a.out -plk -pn ps8 < input                           オーバヘッド測定..... 注2
```

pgoの実行は、上記実行手順1の終了後、必要に応じてpgoを変更したあと、Paragon上の$HOME/kpx/pdoで以下を入力します。

```
Paragon$ csh pgo
```

※　注1　オプションの変更等必要に応じて書換が必要です。
　　注2　ユーザ・プログラム依存ですので書換が必要です。
　　実行時に機器番号66と99を使用します。

5.3 ptoolカスタマイズ

ptoolをカスタマイズして使用するには、ptoolの入力ファイルp_tool.datを変更するか、ソース・ファイルを変更します。p_tool.datの変更については「付録４．ptool制御データ」を、ソース・ファイルの変更については「付録５．ptoolディレクティブ」を参照してください。ここでは、機能の一部をご紹介いたします。サンプルとして用意したp_tool.datを以下に示します。

```
Paragon$ cat kpx/ptool/p_tool.dat          オーバヘッド計測用データ・サンプル
mysystem = p_system.dat                      システム・データ・ファイル
! system library name                        「!」以下はコメント
! SYS_SUB  = GSYNC                          計測対象ルーチン名を指定する
! SYS_FUNC = MYNODE                         計測対象関数名を指定する

! inp_soc = usou.f                          計測元ソース名をusou.fにする
! out_soc = nsou.f                          計測用ソース名をnsou.fにする
! cpu_ctr = ninc                            ツールのインクルード名をninc にする

max_ids = 256                               使用する可能性のある最大ノード数を指定する

! out_u66 = 43                              計測結果ファイルをFortran論理機器番号43に結合する
! out_f66 = oh.lst                          計測結果ファイル名をoh.lstにする
! opn_u66 = yes            ! open           計測結果ファイルのオープン処理を行う
! out_u99 = 47                              編集前の計測結果ファイルをFortran論理機器番号47に結合する
! out_f99 = oh.tbl                          編集前の計測結果ファイル名をoh.tblにする
! opn_u99 = yes            ! open           編集前の計測結果ファイルのオープン処理を行う

ohd_ctr = 1000            ! overhead counter  時間計測ルーチンの試行回数を1,000回とする
.    ! end of data ----------------------------------------
```

記述規則として「!」以下はコメントとなるので、サンプルとして用意したp_tool.datでは、mysystem = p_system.dat、max_ids = 256、ohd_ctr = 1000の３ヵ所だけが有効です。最後の「.」の行はデータの終了を宣言していますが、後にデータを記述した行がない場合、省略できます。

１．Fortran論理機器番号の変更

ptoolでは結果出力のため、Fortran論理機器番号66番と99番を使用しています。ユーザ・プログラムとの競合を避るには、ptoolの入力ファイルp_tool.datで機器番号の指定変更を行います。

２．使用する可能性のある最大ノード数

対象システムの最大ノード数を指定しておけば安全です。1024ノードのParagonを使用していたら、特に理由がない限り、ptoolの入力ファイルp_tool.datで以下のように指定します。

```
max_ids = 1024
```

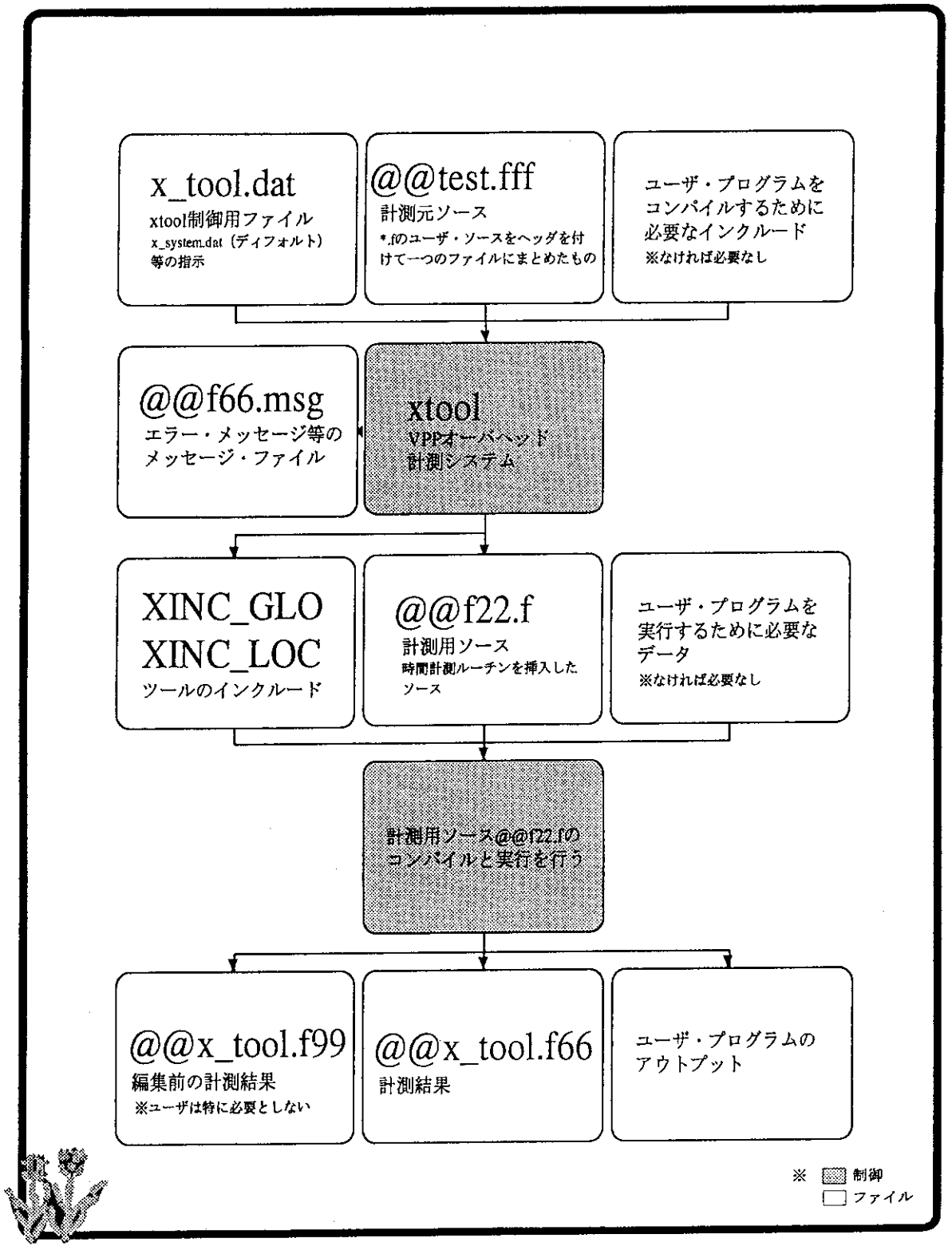
5.4 ptool 出力結果

Paragon オーバヘッド情報リスト

```
1  改ページ制御記号
 <<< Paragon Fortran System Calls Cost List >>>

 Number of nodes   :     2  実行ノード数
 System call lines :    71  システム・コールの行数          日-月-年 時:分:秒
                                                          2-Sep-96 18:17:47
 順番号  ルーチン名                        連番号    システム・コール名     システム・コール種別  行番号 ファイル名
  seq.  subprogram name                     loc.    system call name
------  ----+----1----+----2----+----3---  ------  ----------------
     1  DATAIN                                 1    MYNODE                 Function        10  datain.f
     2  DATAIN                                 2    NUMNODES               Function        11  datain.f
     3  DATAIN                                 3    CSEND                  Subroutine      50  datain.f
==========  省略  ==========
    12  VDFT3F                                71    CRECV                  Subroutine     255  vdft3f.f

時間計測ルーチン自身のオーバヘッド
Self-Overhead Time per one-pair call.  (    1000 overhead counter)
 ノード番号  作成タイマ使用時間  dclock使用時間
    node#        Outer          Inner    (sec)
       1      0.000007       0.000002
       2      0.000008       0.000002

ファイル番号 ソース・ファイル名
File# filename
   1  addvar.f
   2  calcfn.f
   3  convol.f
==========  省略  ==========
  16  vdft3i.f                                          日-月-年 時:分:秒
                                                       2-Sep-96 18:25:05
 順番号  ルーチン名                        連番号 システム・コール名 ファイル番号 行番号
  seq.  subprogram name                     loc.    system call name  File#Line.
------  ----+----1----+----2----+----3---  ------  ----------------  ---------
                                          ノード番号          時間          実行回数
                                             node          time (sec)        counts
                                              --#  ----------------  ----------
     1  DATAIN                                 1    MYNODE                4    10
                                               1      0.000006   (         1)
                                               2      0.000006   (         1)
     2  DATAIN                                 2    NUMNODES              4    11
                                               1      0.000005   (         1)
                                               2      0.000004   (         1)
     3  DATAIN                                 3    CSEND                 4    50
                                               1      0.000152   (         1)
                                               2      0.000000   (         0)
==========  省略  ==========
    12  VDFT3F                                71    CRECV                15   255
                                               1      2.556386   (     19200)
                                               2      2.540179   (     19200)
計測用プログラムの総経過時間                                  日-月-年 時:分:秒
TOTAL  Elapsed  Time                                        2-Sep-96 18:25:05
   438.140892 second
 順番号  システム・コール名      全ノードでのオーバヘッド合計時間          出現回数
  seq.  system call name  (summation all node)          time (sec)        counts
------  ----+----1----+----2----+----3---  ----------------  ----------
     1  CPROBE                                 0.000000   (         0)
     2  CPROBEX                                0.000000   (         0)
     3  CREAD                                  0.000000   (         0)
==========  省略  ==========
   113  NX_WAITALL                             0.000000   (         0)

ノード毎のオーバヘッド総量                    ノード番号 ノード毎総計時間
TOTAL  OVERHEAD                                  node  total time (sec)
                                                  --#  ----------------
                                                   1      40.120147
                                                   2      49.737358
```

※ 報告時間の単位は秒。

6.1 xtool システム概要



6.2 xtool実行方法

xtoolは、VPPの並列化ディレクティブにより並列化されたプログラムの、PE間通信等の並列化オーバヘッドを測定するツールです。xtoolの実行は、以下のようにして行います。

1．ユーザ・ソースと、そのコンパイルと実行に必要な資源をkpx/xdoにコピーします。
　※メイン・ルーチンは一本で、ソース・ファイル名は*.fを想定しています。
2．ソース*.fをヘッダを付けて1本にまとめ、計測元ソース@@test.fffとします。
3．xtoolを実行してソースに計測ルーチンを挿入し、新たな計測用ソース@@f22.fを作成します。
4．@@f22.fをコンパイル後、実行することにより、オーバヘッドの計測結果のファイル@@x_tool.f66が出力されます。

xtoolの実行方法について、上記実行手順2以降の、標準的な実行を想定したシェル・スクリプトxgoを用意しました。以下にxgoの内容を示します。

```
VPP$ cat kpx/xdo/xgo
#! /bin/csh -f                                          特に必要なし
cd $HOME/kpx/xdo                                       特に必要なし
#--------+---------+---------+---------+---------+---------+
¥rm -f @@* XINC_*                                      不要なファイルの消去
../xtool/CatK *.f > @@test.fff                         計測元ソースを作成
¥cp ../xtool/x_tool.dat x_tool.dat                     p_tool.dat制御データ・ファイルを設定
../xtool/xtool                                         計測用ソースを作成
¥rm @@f*.wrk                                           不要なファイルの消去
##### Change next line to fit.
frt -Wx @@f22.f                                        コンパイル..... 注1
echo ''
echo    '==== Would you like to continue ? ===='
echo -n '==== Reply  y : continue  else : quit ===='   サブミット前の確認
if  ($< == y)  then
    ##### Change next line to fit.
    qsub -q fu004nlm -lP 4 go                          オーバヘッド測定..... 注2
endif
```

xgoの実行は、上記実行手順1の終了後、必要に応じてxgoを変更したあと、VPP300またはGSP上の$HOME/kpx/xdoで以下を入力します。

```
VPP$ csh xgo
```

※ 注1　コンパイラや、オプションの変更等必要に応じて書換が必要です。
　 注2　必要に応じて書換が必要です。ユーザ・プログラム依存のgoを作成します。
　　実行時に機器番号66と99を使用します。

6.3 xtool カスタマイズ

xtool をカスタマイズして使用するには、xtool の入力ファイル x_tool.dat を変更するか、ソース・ファイルを変更します。x_tool.dat の変更については「付録６．xtool 制御データ」を、ソース・ファイルの変更については「付録７．xtool ディレクティブ」を参照してください。ここでは、機能の一部をご紹介いたします。サンプルとして用意した x_tool.dat を以下に示します。

```
vpp$ cat kpx/xtool/x_tool.dat              オーバヘッド計測用データ・サンプル
mysystem  = x_system.dat                     システム・データ・ファイル

! inp_soc  = orig.f                         計測元ソース名を orig.f にする
! out_soc  = news.f                         計測用ソース名を news.f にする
! out_f66  = oh.lst                         計測結果ファイル名を oh.lst にする
! out_f99  = oh.tbl                         編集前の計測結果ファイル名を oh.tbl にする
! out_u66  = 63                             計測結果ファイルを Fortran 論理機器番号 63 に結合する
! out_u99  = 67                             編集前の計測結果ファイルを Fortran 論理機器番号 67 に結合する

ohd_ctr = 10000          ! overhead counter    時間計測ルーチンの試行回数を 10,000 回とする
.    ! end of data ------------------------------------------------
```

記述規則として「!」以下はコメントとなるので、サンプルとして用意した x_tool.dat では、mysystem = x_system.dat、ohd_ctr = 10000 の２ヵ所だけが有効です。最後の「.」の行はデータの終了を宣言していますが、後にデータを記述した行がない場合、省略できます。

１．Fortran 論理機器番号の変更

xtool では結果出力のため、Fortran 論理機器番号 66 番と 99 番を使用しています。ユーザ・プログラムとの競合を避るには、xtool の入力ファイル x_tool.dat で機器番号の指定変更を行います。66 番を 63 番へ、99 番を 67 番へそれぞれ変更したい場合は以下のように指定します。

```
out_u66  = 63
out_u99  = 67
```

6.4 xtool 出力結果

VPPオーバヘッド情報リスト

```
1  改ページ制御記号
<<< XOCL Executable Statements Cost List >>>

 Number of Processors  :     4  実行PE数
 Executable XOCL Lines :    70  XOCLの行数
                                                        年/月/日  時:分:秒
XOCL Statement List                                     96/09/11  17:51:23
順番号  ルーチン名        連番号 XOCL文                 ファイル番号 ファイル名
  Seq.  SubProgram Name   Loc.  XOCL Statement              File#  Name
------  ----+----1----+-  ----  ------------------------------  -----  ----
     1  GETKSP             1  SPREADMOVE                      33  getksp.f
     2  GETKSP             2  ENDSPREAD (R1)                  39  getksp.f
     3  GETKSP             3  MOVEWAIT (R1)                   40  getksp.f
==========  省略  ==========
    13  VDFTZF            70  MOVEWAIT (F2)                  117  vdftzf.f

時間計測ルーチン自身のオーバヘッド
Self-Overhead Time per one-pair call.  (    1000 overhead counter)
     PE番号  作成タイマ使用時間  gettod使用時間
     node#        Outer           Inner     (sec)
        1      0.000054        0.000010
        2      0.000054        0.000009
        3      0.000053        0.000009
        4      0.000053        0.000009
ファイル番号 ソース・ファイル名
File# filename
   1  addvar.f
   2  calcfn.f
   3  datain.f
==========  省略  ==========
  17  vdftzf.f
                                                        年/月/日  時:分:秒
XOCL Start Parallel Region  並列処理開始時刻            96/09/11  17:51:23
                                                        年/月/日  時:分:秒
XOCL End   Parallel Region  並列処理終了時刻            96/09/11  18:22:59
順番号  ルーチン名        連番号 XOCL文                 ファイル番号 行番号
  Seq.  SubProgram Name   Loc.  XOCL Statement              File#  Line#
------  ----+----1----+-  ----  ------------------------------  -----  -----
                                          PE番号          時間          実行回数
                                          PE.No       Time (Sec)     ( Counts)
                                          ----#  ----------------  ----------
     1  GETKSP             1  SPREADMOVE                        5    33
                                             1   0.000000  (      0)
                                             2   0.000000  (      0)
                                             3   0.000000  (      0)
                                             4   0.000000  (      0)
     2  GETKSP             2  ENDSPREAD (R1)                    5    39
                                             1   0.545347  (   1200)
                                             2   0.547760  (   1200)
                                             3   0.549340  (   1200)
                                             4   0.546390  (   1200)
==========  省略  ==========
    13  VDFTZF            70  MOVEWAIT (F2)                    17   117
                                             1   0.026451  (   1200)
                                             2   0.027819  (   1200)
                                             3   0.027491  (   1200)
                                             4   0.028197  (   1200)

   並列処理の総経過時間                                 年/月/日  時:分:秒
TOTAL Elapsed Time (Sec)  & OVERHEAD COST               96/09/11  18:22:59
      1896.048205
                                          PE番号    PE毎総計時間
                                          PE.No     Time (Sec)
                                          ----#  ----------------
                                             1   13.082205
                                             2   13.120046
                                             3   12.994081
                                             4   13.128103
```

※ 報告時間の単位は秒。

付録１．ktool制御データ

以下に、ktool制御データで設定可能なシンボル名を挙げ、その意味とディフォルト値ついて記す。

１）ファイル名、入口名、コモン名等

計測用ソースに組み込まれるシンボル名が、計測元ソースのシンボル名と重複することを防止する目的で指定する。

シンボル名	意味	ディフォルト
ENT_ENT	プログラムの計測開始入口名	ENT_ENT
RET_ENT	プログラムの計測終了入口名	RET_ENT
ENT_DOL	doループの計測開始入口名	ENT_DOL
RET_DOL	doループの計測終了入口名	RET_DOL
CTR_DOL	doループのループ回数計測入口名	CTR_DOL
ENT_IOU	入出力の計測開始入口名	ENT_IOU
RET_IOU	入出力の計測終了入口名	RET_IOU
CPU_MSE	CPU／経過時間の計測開始入口名（要暗黙実数型）	CPU_MSE
CPU_MSR	CPU／経過時間の計測終了入口名（要暗黙実数型）	CPU_MSR
OUT_CTR	計測結果出力入口名	OUT_CTR
CPU_BLK	共通ブロック初期プログラム名	CPU_BLK
CPU_CTR	共通ブロックのインクルード・ファイル名	CPU_CTR
COM_CHR	文字型共通ブロック名	COM_CHR
COM_BIN	数値型共通ブロック名	COM_BIN
DUM_DUM	当該ツールのオーバヘッド計測入口名	DUM_DUM

２）ファイル論理機番等

ファイル情報を指定する。

シンボル名	意味	ディフォルト
INP_SOC	計測元ソース・ファイル名	@@test.fff
OUT_SOC	計測用ソース・ファイル名	@@f22.f
OUT_U66	計測結果ファイル論理機番	66 (出力)
OUT_F66	計測結果ファイル名	@@k_tool.f66
OPN_U66	計測結果ファイルのオープン処理要求 Yes／Noで要／否を指定	Yes
OUT_U99	計測結果データ・ファイル論理機番	99 (出力)
OUT_F99	計測結果データ・ファイル名	@@k_tool.f99
OPN_U99	計測結果データ・ファイルのオープン処理要求 Yes／Noで要／否を指定	Yes

３）時間計測ルーチンの試行回数

ツール自身のオーバヘッドを削減するために、別途行う、時間計測ルーチンの試行回数。０が指定された場合には、ツール自身のオーバヘッド時間を含んだ計測結果を出力する。

シンボル名	意味	ディフォルト
OHD_CTR	時間計測ルーチンの試行回数	1000

４）計測対象の特定

シンボル名	意味	ディフォルト
GLOBAL_ENT	計測対象プログラムのグローバル指定	Yes

Yes／Noの何れかで指定し、Yesの場合はユーザ・プログラムの総てが一旦その対象となり、ENTRYで除外するプログラムを指定する。Noの場合は、ENTRY指定されたプログラムのみが計測対象となる。

GLOBAL_DOL　　計測対象doループのグローバル指定　　Yes

計測対象から除外または追加するdoループの指定は、後述するDOSEQ、DOLABELで行う。GLOBAL_ENTと同様にYes／Noを指定して併用する。

GLOBAL_IOU　　計測対象入出力処理のグローバル指定　　Yes

計測対象から除外または追加する入出力処理の指定は、後述するIOSEQで行う。GLOBAL_ENTと同様にYes／Noを指定して併用する。計測対象の入出力処理文はOPEN、CLOSE、READ、WRITE、PRINT、PUNCH文とする。

UNSAT_DO　　doループの未確定ループ回数の計測指定　　Yes

doループ処理前にループ回数が確定していないdoループ、例えば、do while、do untilや、ループ外への飛び出しのあるdoループ（goto文、return文、stop文）には、doループ内にそのループ回数を得る処理を挿入するが、計測処理のオーバヘッドが増加する。Yesが指定されると、未確定ループ回数のdoループの計測をし、Noが指定されると、未確定ループ回数のdoループの計測を抑止する。

ENTRY　　プログラム名（入口名）　　なし

計測対象から除外または追加するプログラム名を指定する。GLOBAL_ENTで、Yesが指定されている場合は計測除外するプログラム名を、Noが指定されている場合は追加計測するプログラム名を指定する。

DOLABEL　　doループのラベルの指定　　なし

計測対象から除外または追加するdoループを、ラベルにより指定する。プログラム名とラベルは［，］で区切って一緒に指定する。多重doループでラベルが重複されて利用された場合は、最内ループが指定されたとみなされる。最内ループ以外を指定したい時は、DOSEQデータにより指定する。DOLABELはGLOBAL_DOLと併用して利用する。

DOSEQ　　doループ出現順番の指定　　なし

計測対象から除外または追加するdoループを、プログラム内におけるループの出現順番により指定する。プログラム名とループ出現順番は［，］で区切って一緒に指定する。ラベルなしdoループの場合は、DOSEQでのみ指定が可能である。DOSEQはGLOBAL_DOLと併用して利用する。

IOSEQ　　入出力処理出現順番の指定　　なし

計測対象から除外または追加する入出力処理を、プログラム内における入出力処理の出現順番を指定して特定する。プログラム名と入出力処理出現順番は［，］で区切って一緒に指定する。IOSEQはGLOBAL_IOUと併用して利用する。

MAX_ENT　　プログラムの計測対象最大値の指定　　0

リスタート処理（実行回数計測結果を利用した実行時間計測）時に、測定対象のプログラムをその実行回数で制限できる。リスタート時にMAX_ENTが指定される（0以外）と、指定回数以上に実行されているプログラムは測定対象から除外される。また、リスタート処理を行うかどうかは、編集前の計測結果ファイルOUT_F99の有無により決定される。MAX_ENTが0またはOUT_F99が存在しない場合は実行回数計測、そうでない場合はリスタート処理を行う。

MAX_IOU　　入出力処理の計測対象最大値の指定　　0

MAX_IOUはMAX_ENTと同様な働きをする。リスタート処理時に、実行回数がMAX_IOUの値以上の入出力処理

を計測対象から除外する。

MAX_DOL	doループ処理の計測対象最大値の指定	0

MAX_DOLはMAX_ENTと同様な働きをする。リスタート処理時に、実行回数がMAX_DOLの値以上のdoループ処理を計測対象から除外する。

MAX_MRT	doループ処理の平均ループ回数の計測対象最大値の指定	0

MAX_MRTはMAX_ENT と同様な働きをする。リスタート処理時に、平均ループ回数がMAX_MRTの値以上のdoループ処理を計測対象から除外する。

5）その他の制御データ

シンボル名	意味	ディフォルト
MYSYSTEM	制御用ファイル名	mysystem.dat

制御用ファイル名を与える。MYSYSTEMが指定されない時は、カレント・ディレクトリにmysystem.datが存在するとして処理する。

TABCODE	タブ・コードの内部コードの指定	9

タブ・コードの内部コードを10進数で与える。TABCODEの値は、CHAR(n)のnとして利用される。TABCODEの指定は後述するTABNUMの前に行わければならない。

TABNUM	タブ・コードによるカラム数の指定	8

タブ・コードに対応するカラム数を与える。制御用データ入力時、TABCODE、TABNUMが指定されるまでは各々９と１が採用される。最終的にTABNUMがデータとして指定されない場合は、入力終了時に８の標準値が指定される。TABCODEの指定はTABNUMの前に行わなければならない。

JST_CTR	計測の種別指定	No

計測の種別は実行回数と時間計測、および実行回数計測のみの２通りがある。計測の種別をJST_CTRで指定する。Noの場合は両方の計測を、Yesの場合は実行回数のみを計測する。

INC_PATH	インクルードの参照パスの指定	なし

計測元ソースで、インクルード文の参照パスを特定したい時INC_PATHで指定する。INC_PATHは複数指定可能で、インクルード文が存在した場合、INC_PATHを出現順に検索して、必要なインクルード・ファイルを参照する。インクルード文が絶対パス指定の場合は、INC_PATHを参照しない。また、「./」でインクルード指定された場合は、無条件にカレント・ディレクトリを参照する。同じ名前のインクルード・ファイルが複数存在する時は、複数のINC_PATHで指定するが、指定順序に注意が必要である。インクルード文は次の形式のみ許される。

```
a.    include 'abc.inc'
b.    include "abc.inc"
c. *include abc
```

6）制御データの記述例

```
CPU_CTR = cpuinc.inc      —> インクルード・ファイル名の変更
CPU_BLK = cpublk          —> 共通ブロック初期化プログラム名の変更
COM_CHR = comxxx          —> 共通ブロック名の変更
COM_BIN = comyyy          —> 共通ブロック名の変更
! change file
```

```
out_u66 = 06                                  —> 情報を標準出力と一緒に出力する
opn_u66 = 0          ! not open               —> open／closeを抑止
out_u99 = 12                                  —> 論理機番の一致を防止
out_f99 = @@export.dat
! Global data
global_ent = yes       ! specified erase data
! Entry name
entry = e_sub                                 —> エントリe_subを計測対象から除外
! Do loop (Label/Seq)
dolabel = d_sub , 100                         —> エントリd_subのラベル100doループを計測除外
doseq   = d_sub , 2                           —> エントリd_subの２番目のdoループを計測除外
.  ! end of data ————                         —> 以下のデータは無視される。
```

付録２．ktoolディレクティブ

計測対象は一般に制御データで指定するが、計測元ソースに挿入するディレクティブでの指定も可能である。ディレクトリの仕様と種別を以下に示す。ディレクティブは制御データのGLOBAL_xxxと併用して、計測の除外対象か追加対象かを指定する。

1）ディレクティブの仕様

第１カラムから「COSTTIME@パラメータ」で指定し、第１カラム以外のブランクは無視する。大文字、小文字の区別はない。以下の例はいずれも同じ意味を持つ。

```
a. Cost Time@ DOL
b. C O S T   T I M E @   D O L
c. COSTTIME@DOL
d. CostTime@ Dol
```

パラメータには、ENT、DOL、IOUの３つがあり、各々プログラムの特定、doループの特定、入出力処理の特定を指定する。

2）パラメータの詳細

ENT　　プログラムの計測の除外対象か追加対象かの指定

制御データで「GLOBAL_ENT=Yes」が指定されている場合には、そのプログラムは計測対象から除外され、「GLOBAL_ENT=No」の場合には計測対象となる。このディレクティブは、他のディレクティブに優先する。

(例)
```
        Subroutine abc
        implicit real*8 (a-h,o-z)
costtime@ ENT
        :
        :
        end
```

DOL　　doループの計測の除外対象か追加対象かの指定

制御データで「GLOBAL_DOL=Yes」が指定されている場合には、そのdoループは計測対象から除外され、「GLOBAL_DOL=No」の場合には計測対象となる。このディレクトリが指定されて最初に出現したdoループがその対象となる。通常混乱を防止するため、doループの直前に指定した方が良い。

(例)
```
        Subroutine abc
        implicit real*8 (a-h,o-z)
        :
costtime@ DOL                          対応doループがなく無効
        :
costtime@ DOL
        do 100 i=1,n
        do 100 j=1,m
        :
costtime@ DOL
        do 300 k=1,n*m
        :
        end
```

IOU　　入出力処理の計測の除外対象か追加対象かの指定

制御データで「GLOBAL_IOU=Yes」が指定されている場合には、その入出力処理は計測対象から除外され、「GLOBAL_IOU=No」の場合には計測対象となる。このディレクティブが指定されて最初に出現した入出力処理がその対象となる。通常混乱を防止するため、入出力処理の直前に指定した方が良い。

(例)

```
        Subroutine abc
        implicit real*8 (a-h,o-z)
        :
costtime@ IOU
        if ( a(i) .gt. b ) write(...) .....
        :
costtime@ IOU
        write(...) .....
        :
        end
```

付録３．時間計測ルーチン

時間計測ルーチンは、mysystem.datの一部である次の２つの副プログラムで定義している。WORKTMがミリ秒単位の値を持つようにTIMEFACTをセットする。時間計測ルーチンを以下に示す。

１．CPU_MSE　　：時刻計測開始処理

```
*DECK CPU_MSE                                                    CPU_MSE
C
      FUNCTION CPU_MSE (CPUSEW)                                  CPU_MSE
C     RETURN MILI SECONDS (ENTRY  PROCESS)
      REAL*8 CPUSEW
C
C For Fujitsu VPP 'gettod' time routine
C      Sub.
C      A
C      +  --t1  (CPUSEW)        Sub. A Net Time
C      I  --t2  (CPUSEC)           = (t3-t2)+(t6-t5)
C      I      B                 Sub. B Net Time
C      +------+  --t3              = (t5-t4)
C             I  --t4           OverHead
C             I                    = (t2-t1)+(t4-t3)+alpha
C      +------+  --t5
C      I
C      +  --t6
C
      REAL*8 CPU_MSE                                             CPU_MSE
      REAL*8 CPUSEC
C     TIMEFACT : CONVERSION FACTOR TO MILI-SECOND
      REAL*8 TIMEFACT
      REAL*8 WORKTM
C     ================================================
C     ------------------------
C---- VPP300/VPP500 CPU TIME (CLOCK)
      DATA   TIMEFACT / 1000.0 /
C
      CALL CLOCK (CPUSEC,2,2)
      CPUSEW  = CPUSEC/TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------
C---- VPP300/VPP500 ELAPSED TIME (GETTOD)
      DATA   TIMEFACT / 1000.0 /
C
      call gettod(CPUSEW)
      CPUSEW  = CPUSEW/TIMEFACT
      call gettod(CPUSEC)
      WORKTM  = CPUSEC/TIMEFACT
C     ------------------------
C---- Monte-4 CPU TIME (CLOCK)
      DATA   TIMEFACT / 1000.0 /
C
      CALL CLOCK (CPUSEC)
      CPUSEW  = CPUSEC*TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------
C---- Monte-4 ELAPSED TIME (ETIME)
      REAL   ETIME,TARRAY(2)
      DATA   TIMEFACT / 1000.0 /
C
      CPUSEC  = ETIME( TARRAY )
      CPUSEW  = CPUSEC*TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------
C---- Paragon CPU TIME (ETIME)
      include 'fnx.h'
      REAL   ETIME,TARRAY(2)
      DATA   TIMEFACT / 1000.0 /
C
      CPUSEC  = ETIME( TARRAY )
      CPUSEW  = CPUSEC*TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------
C---- Paragon ELAPSED TIME (DCLOCK)
      include 'fnx.h'
      DATA   TIMEFACT / 1000.0 /
C
      CPUSEC  = DCLOCK()
      CPUSEW  = CPUSEC*TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------
```

```
C---- SX-4 CPU TIME (CLOCK)
      DATA   TIMEFACT / 1000.0 /
C
      CALL CLOCK (CPUSEC)
      CPUSEW  = CPUSEC*TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------
C---- SX-4 ELAPSED TIME (ETIME)
      REAL   ETIME, TARRAY(2)
      DATA   TIMEFACT / 1000.0 /
C
      CPUSEC  = ETIME( TARRAY )
      CPUSEW  = CPUSEC*TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------
C---- SP2 CPU TIME (etime_)
      REAL   etime_, TARRAY(2)
      DATA   TIMEFACT / 1000.0 /
C
      CPUSEC  = etime_(TARRAY)
      CPUSEW  = CPUSEC*TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------
C---- SP2 ELAPSED TIME (TIMEF)
      REAL*8 TIMEF
C
      CPUSEC  = TIMEF()
      CPUSEW  = CPUSEC
      WORKTM  = CPUSEW
C     ------------------------
C---- SR2201 CPU TIME
      DATA   TIMEFACT / 1000.0 /
      DATA   IFIRST   / 1 /
C
      IF ( IFIRST .EQ. 1 ) THEN
        IFIRST = 0
        CALL XCLOCK(CPUSEC, 3)
      ENDIF
      CALL XCLOCK(CPUSEC, 5)
      CPUSEW  = CPUSEC*TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------
C---- SR2201 ELAPSED TIME
      DATA   TIMEFACT / 1000.0 /
      DATA   IFIRST   / 1 /
C
      IF ( IFIRST .EQ. 1 ) THEN
        IFIRST = 0
        CALL XCLOCK(CPUSEC, 7)
      ENDIF
      CALL XCLOCK(CPUSEC, 8)
      CPUSEW  = CPUSEC*TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------
C---- T94 CPU TIME (SECOND)
      DATA   TIMEFACT / 1000.0 /
C
      CALL SECOND (CPUSEC)
      CPUSEW  = CPUSEC*TIMEFACT
      WORKTM  = CPUSEW
C     ------------------------
C---- T94 ELAPSED TIME (TIMEF)
C
      CALL TIMEF (CPUSEC)
      CPUSEW  = CPUSEC
      WORKTM  = CPUSEW
C     ========================                              CPU_MSE
      CPU_MSE = WORKTM
C
      END
```

2. CPU_MSR : 時刻計測終了処理

```
*DECK CPU_MSR                                                 CPU_MSR
C
      FUNCTION CPU_MSR ()                                     CPU_MSR
C     RETURN MILI SECONDS (RETURN PROCESS)
      REAL*8 CPU_MSR                                          CPU_MSR
      REAL*8 CPUSEC
C     TIMEFACT : CONVERSION FACTOR TO MILI-SECOND
      REAL*8 TIMEFACT
      REAL*8 WORKTM
C     ========================
C     ------------------------
C---- VPP300/VPP500 CPU TIME (CLOCK)
      DATA   TIMEFACT / 1000.0 /
C
      CALL CLOCK (CPUSEC, 2, 2)
```

```
      WORKTM  = CPUSEC/TIMEFACT
C     ------------------------------
C---- VPP300/VPP500 ELAPSED TIME (GETTOD)
      DATA    TIMEFACT / 1000.0 /
C
      call gettod(CPUSEC)
      WORKTM  = CPUSEC/TIMEFACT
C     ------------------------------
C---- Monte-4 CPU TIME (CLOCK)
      DATA    TIMEFACT / 1000.0 /
C
      CALL CLOCK (CPUSEC)
      WORKTM  = CPUSEC*TIMEFACT
C     ------------------------------
C---- Monte-4 ELAPSED TIME (ETIME)
      REAL    ETIME, TARRAY(2)
      DATA    TIMEFACT / 1000.0 /
C
      CPUSEC  = ETIME( TARRAY )
      WORKTM  = CPUSEC*TIMEFACT
C     ------------------------------
C---- Paragon CPU TIME (ETIME)
      include 'fnx.h'
      REAL    ETIME, TARRAY(2)
      DATA    TIMEFACT / 1000.0 /
C
      CPUSEC  = ETIME( TARRAY )
      WORKTM  = CPUSEC*TIMEFACT
C     ------------------------------
C---- Paragon ELAPSED TIME (DCLOCK)
      include 'fnx.h'
      DATA    TIMEFACT / 1000.0 /
C
      CPUSEC  = DCLOCK()
      WORKTM  = CPUSEC*TIMEFACT
C     ------------------------------
C---- SX-4 CPU TIME (CLOCK)
      DATA    TIMEFACT / 1000.0 /
C
      CALL CLOCK (CPUSEC)
      WORKTM  = CPUSEC*TIMEFACT
C     ------------------------------
C---- SX-4 ELAPSED TIME (ETIME)
      REAL    ETIME, TARRAY(2)
      DATA    TIMEFACT / 1000.0 /
C
      CPUSEC  = ETIME( TARRAY )
      WORKTM  = CPUSEC*TIMEFACT
C     ------------------------------
C---- SP2 CPU TIME (etime_)
      REAL    etime_, TARRAY(2)
      DATA    TIMEFACT / 1000.0 /
C
      CPUSEC  = etime_(TARRAY)
      WORKTM  = CPUSEC*TIMEFACT
C     ------------------------------
C---- SP2 ELAPSED TIME (TIMEF)
      REAL*8 TIMEF
C
      CPUSEC  = TIMEF()
      WORKTM  = CPUSEC
C     ------------------------------
C---- SR2201 CPU TIME
      DATA    TIMEFACT / 1000.0 /
C
      CALL XCLOCK(CPUSEC, 5)
      WORKTM  = CPUSEC*TIMEFACT
C     ------------------------------
C---- SR2201 ELAPSED TIME
      DATA    TIMEFACT / 1000.0 /
C
      CALL XCLOCK(CPUSEC, 8)
      WORKTM  = CPUSEC*TIMEFACT
C     ------------------------------
C---- T94 CPU TIME (SECOND)
      DATA    TIMEFACT / 1000.0 /
C
      CALL SECOND (CPUSEC)
      WORKTM  = CPUSEC*TIMEFACT
C     ------------------------------
C---- T94 ELAPSED TIME (TIMEF)
C
      CALL TIMEF (CPUSEC)
      WORKTM  = CPUSEC
C     ==============================
      CPU_MSR = WORKTM                                  CPU_MSR
C
      END
```

付録４．ptool制御データ

以下に、ktool制御データで設定可能なシンボル名を挙げ、その意味とディフォルト値ついて記す。

1）ファイル名、入口名、コモン名

計測用のために追加されるサブルーチン、インクルード・ファイル、変数の名前を指定する。

シンボル名	意味	ディフォルト
INP_SOC	計測元ソース・ファイル名	@@test.fff
MYSYSTEM	システム・データ・ファイル名	x_system.dat
OUT_SOC	計測用ソース・ファイル名	@@f22.f
CPU_CTR	計測インクルード・ファイル名	CPU_CTR
DUM_DUMO	ダミー・ルーチン名	DUM_DUMO
CPU_MSE	時刻取り出しルーチン名	CPU_MSE
DUM_DUM	自己オーバヘッド計測ルーチン名	DUM_DUM
ENT_ENT	計測開始処理ルーチン名	ENT_ENT
RET_ENT	計測終了、加算処理ルーチン名	RET_ENT
OUT_CTR	並列オーバヘッドの出力ルーチン名	OUT_CTR

(例) シンボル名の変更は次のように指定する。

```
INP_SOC   = @@test.camp      ! 計測元ソース・ファイル名
MYSYSTEM  = p_tool.sys       ! システム・データ・ファイル名
OUT_SOC   = @@camp.f         ! 計測用ソース・ファイル名
CPU_CTR   = @@test.inc       ! 計測インクルード・ファイル名
```

2）ファイル論理機番等

計測結果のファイル情報を指定する。

シンボル名	意味	ディフォルト
OUT_U66	計測結果の出力ユニット番号	66
OUT_F66	計測結果の出力ファイル名	@@p_tool.f66
OPN_U66	計測結果の出力ユニットのオープン処理要求 Yes／Noで要／否を指定	Yes
OUT_U99	計測結果のテーブル出力ユニット番号	99
OUT_F99	計測結果のテーブル出力ファイル名	@@x_tool.f99
OPN_U99	計測結果のテーブル出力ユニットのオープン処理要求 Yes／Noで要／否を指定	Yes

(例) ファイル出力情報の変更は次のように指定する。

```
OUT_F66   = @@test.f66       ! 計測結果の出力ファイル名
OUT_F66   = 66               ! 計測結果の出力ユニット番号
OUT_F99   = @@test.f99       ! 計測結果のテーブル出力ファイル名
OUT_F99   = 99               ! 計測結果のテーブル出力ユニット番号
```

3）時間計測ルーチンの試行回数

ツール自身のオーバヘッドを削減するために別途行う、時間計測ルーチンの試行回数。０が指定された場合には、ツール自身のオーバヘッド時間を含んだ計測結果を出力する。

シンボル名	意味	ディフォルト
OHD_CTR	時間計測ルーチンの試行回数	0

4）最大プロセッサ数

プログラムで使用するプロセッサ数を指定する。実際に使用するプロセッサ数以上の値を指定しておけばよい。

シンボル名	意味	ディフォルト
MAX_IDS	最大プロセッサ数	200

5）システム・ライブラリの指定

システム・データ内には、Paragonのシステム・ルーチン名、nxライブラリ・ルーチン名が記述されており、これらのルーチン名がディフォルトの計測対象となっている。これを無視して、計測するルーチン名をプライベートに指定することが可能である。

シンボル名	意味	ディフォルト
SYS_SUB	プライベートな計測対象ルーチン名	なし
SYS_FUNC	プライベートな計測対象関数名	なし

(例) プライベートな計測対象を一行毎に指定する。

```
SYS_SUB  = prvsub1          ! ルーチン名
SYS_SUB  = prvsub2          !
SYS_FUNC = prvfunc          ! 関数名
SYS_SUB  = gsum             ! nxライブラリ・ルーチンを指定してもよい
```

付録５．ptoolディレクティブ

計測用ソース上のすべてのライブラリcall文には、前後に計測ルーチンのcall文が挿入されるが、計測元ソース上にディレクティブを記述することにより、計測ルーチンが挿入されず計測の抑止を行うことができる。

第１カラムから「COSTTIME@パラメータ」で指定し、第１カラム以外のブランクは無視する。大文字、小文字の区別はない。

[形式]

```
CostTime@ ON | OFF
```

パラメータのonでディレクティブ以降の文を計測対象とすることを指示し、offで計測対象とせず、call文を挿入しないことを指示する。ディフォルトはonである。

例

```
      Subroutine
      ...
CostTime@    Off        -----+
      iam   = mynode()       | この間は計測しない
      nodes = numnodes()     |
CostTime@    On         -----+
      ...
      call csend(300,n,intsiz,-1,0)  ! 計測する
      ...
      call crecv(300,n,intsiz)       ! 計測する
      ...
      end
```

パラメータのoffからonまでの範囲が計測対象外となる。対応するonが無い場合はend文までの範囲を計測対象外とし、次のプログラムから計測対象に戻す。

付録６．xtool制御データ

以下に、ktool制御データで設定可能なシンボル名を挙げ、その意味とディフォルト値ついて記す。

1）ファイル名、入口名、コモン名

計測用のために追加されるサブルーチン、インクルード・ファイル、変数の名前を指定する。

シンボル名	意味	ディフォルト
INP_SOC	計測元ソース・ファイル名	@@test.fff
MYSYSTEM	システム・データ・ファイル名	x_system.dat
OUT_SOC	計測用ソース・ファイル名	@@test.f
XINC_GLO	計測インクルード・ファイル名（その１）	XINC_GLO
XINC_LOC	計測インクルード・ファイル名（その２）	XINC_LOC
X_INITIM	計測時間初期指定ルーチン名	X_INITIM
X_GLOSET	逐次実行時の計測ルーチン名	X_GLOSET
X_GLOSUM	逐次実行時の計測加算エントリ名	X_GLOSUM
X_LOCSET	冗長実行時の計測ルーチン名	X_LOCSET
X_LOCSUM	冗長実行時の計測加算エントリ名	X_LOCSUM
X_TIMSET	並列、冗長実行時の計測ルーチン名	X_TIMSET
X_TIMSUM	並列、冗長実行時の計測加算エントリ名	X_TIMSUM
X_PRINT	並列オーバヘッドの出力ルーチン名	X_PRINT
XDO_FRST	ＳＰＲＥＡＤ　ＤＯ文計測用ローカル変数名	XDO_FRST
X_COMMON	計測インクルードのコモン・ブロック名（その１）	X_COMMON
X_LOCAL	計測インクルードのコモン・ブロック名（その２）	X_LOCAL
X_GETTOD	経過時間測定ルーチン名 （＊）ディフォルト値は gettod である。	X_GETTOD

(例) シンボル名の変更は次のように指定する。

```
INP_SOC  = @@test.camp      ! 計測元ソース・ファイル名
MYSYSTEM = x_tool.sys       ! システム・データ・ファイル名
OUT_SOC  = @@camp.f         ! 計測用ソース・ファイル名
XINC_GLO = @@test.glo       ! 計測インクルード・ファイル名（その１）
XINC_LOC = @@test.loc       ! 計測インクルード・ファイル名（その２）
```

2）ファイル論理機番等

計測結果のファイル情報を指定する。

シンボル名	意味	ディフォルト
OUT_U66	計測結果の出力ユニット番号	66
OUT_F66	計測結果の出力ファイル名	@@p_tool.f66
OPN_U66	計測結果の出力ユニットのオープン処理要求 Yes／Noで要／否を指定	Yes
OUT_U99	計測結果のテーブル出力ユニット番号	99
OUT_F99	計測結果のテーブル出力ファイル名	@@x_tool.f99
OPN_U99	計測結果のテーブル出力ユニットのオープン処理要求 Yes／Noで要／否を指定	Yes

(例) ファイル出力情報の変更は次のように指定する。

```
OUT_F66  = @@test.f66       ! 計測結果の出力ファイル名
OUT_F66  = 66               ! 計測結果の出力ユニット番号
OUT_F99  = @@test.f99       ! 計測結果のテーブル出力ファイル名
OUT_F99  = 99               ! 計測結果のテーブル出力ユニット番号
```

3）時間計測ルーチンの試行回数

ツール自身のオーバヘッドを削減するために別途行う、時間計測ルーチンの試行回数。0が指定された場合には、ツール自身のオーバヘッド時間を含んだ計測結果を出力する。

シンボル名	意味	ディフォルト
OHD_CTR	時間計測ルーチンの試行回数	1000

付録７．xtoolディレクティブ

計測用ソース上のすべてのxocl文には、前後に計測ルーチンのcall文が挿入されるが、計測元ソース上にディレクティブを記述することにより、計測ルーチンを挿入せず計測の抑止を行うことができる。

第１カラムから「COSTTIME@パラメータ」で指定し、第１カラム以外のブランクは無視する。大文字、小文字の区別はない。

[形式]

CostTime@ ON | OFF

パラメータのonでディレクティブ以降の文を計測対象とすることを指示し、offで計測対象とせず、call文を挿入しないことを指示する。ディフォルトはonである。

例

```
      Subroutine
      ...
!Xocl Parallel Region          ディフォルトは計測対象である
      ...
!Xocl Spread Region
      ...
CostTime@    Off       ------+
!Xocl Spread Do               | この間の Spread Do は計測対象外
      ...                     |
!Xocl End Spread Do           |
CostTime@    On        ------+ 以降の文は計測対象
      ...
!Xocl Region
      ...
CostTime@    Off       ------+
!Xocl Spread Move             | この間の Spread Move は計測対象外
      ...                     |
!Xocl End Spread Move         |
CostTime@    On        ------+ 以降の文は計測対象
      ...
!Xocl End Spread Region
      ...
!Xocl End Parallel
      ...
      end
```

パラメータのoffからonまでの範囲が計測対象外となる。対応するonが無い場合はend文までの範囲を計測対象外とし、次のプログラムから計測対象に戻す。