OUKA
The University of Osaka
Institutional Knowledge Archive

Title	スーパーコンピュータSX-1のタイム・シェアリング・システム　ATSS-AFの使い方(その2)
Author(s)	馬野, 元秀
Citation	大阪大学大型計算機センターニュース. 1986, 62, p. 31-52
Version Type	VoR
URL	https://hdl.handle.net/11094/65700
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

スーパーコンピュータ SX-1 の
タイム・シェアリング・システム ATSS-AF の使い方（その2）

大阪大学 大型計算機センター 研究開発部
馬野 元秀

はじめに

前回の「その1（大阪大学 大型計算機センター ニュース、Vol.16、No.1（1986 年 5 月号、 第61 号)、pp.51-68)」では、スーパーコンピュータ SX-1 の CP (control processor) 上の ATSS-AF (advanced time sharing system - advanced function)の基本的な事項について説明した。本稿では、前回の続きとして、まず統合ライブラリとファイルの管理用コマンドについてまとめ、次に行エディタについてまとめる。そして、コンパイル・ユニットとロード・モジュールの作成のしかたについて述べる。「その1」をまだお読みになっていない方は、まず「その1」をお読み下さるようお願いする。

なお、「その1」に誤りと説明不足の部分があった。まず、その部分の修正および追加をしておこう。

○統合ライブラリの大きさ(2.2 節) ： 統合ライブラリの領域の大きさは LIBALLOC コマンドの SIZE パラメータで指定するが、「この大きさは初期値であり、ある程度までは自動的に大きくなる」(p.59、上から 9-10 行目)としたが、統合ライブラリの領域を自動的に大きくすることはできない。領域がいっぱいになったときは、サイズの大きな別の統合ライブラリを作成して、すべてのサブファイルをそれにコピーしなければならない(コピーの方法は「その1」の p.60 を参照のこと)。

○数値計算用ライブラリの結合(2.3 節)： エディタの中から run でプログラムを実行させるときに、ASL や MATHLIB のような数値計算用ライブラリを結合したいときがある。 このときは、次のいずれかのようにすればよい。

```
E#RUN INLIB='LIB.ASLCP'                    ; ASL を使いたいとき
E#RUN INLIB='LIB.MLIBCP'                   ; MATHLIB を使いたいとき
E#RUN INLIB=('LIB.ASLCP','LIB.MLIBCP')     ; ASL と MATHLIB の両方を使いたいとき
```

TSS では、統合ライブラリ名やファイル名そのままを書くと、前に “ユーザ名.” を補うが、’ ’ でくくると “ユーザ名.” は補わない(完全修飾名)。これらの数値計算用ライブラリは LIB というユーザの所有であると考えればよい。

○データ・ファイルの大きさ(2.5 節)：データ・ファイルの領域の大きさは PREALLOC コマンドの SIZE パラメータで指定するが、「この大きさは初期値であり、ある程度までは自動的に大きく

なることは統合ライブラリと同じである」(p.65、上から 3-4 行目)としたが、統合ライブラリの領域を自動的に大きくすることはできないのだから、この記述は誤っている。 データ・ファイル(より詳しくは、順編成ファイル)は領域の大きさを自動的に大きくすることが可能であるが、 そのためには INCRSZ パラメータを指定しなければならない。例えば、

```
*PREALLOC DF1 RECFORM=VB BLOCKSZ=1000 SIZE=300 INCRSZ=100 FIXTRK NODELR
```

とすると、 まず、300 BLOCK の領域がとられ、これがいっぱいになると 100 BLOCK ずつ 15 回は自動的に増えていく。

○データ・ファイルの属性(2.5 節):Fortran で使用するデータ・ファイルの領域を確保するには PREALLOC コマンドで「FIXTRK と NODELR の２つのパラメータを指定する必要がある」(p.65、上から 4 行目)としたが、これは順編成ファイルに対してのみ有効である。そして、直編成ファイル(ACOS-1000 では、直接編成ファイル)の領域を確保するには、例えば、

```
*PREALLOC DIRF1 FILEORG=DIRECT SIZE=100 BLOCKSZ=80
```

とすればよい。 このとき、レコード長はブロック長(BLOCKSZ)と同じで 80 バイトになる。また、直編成ファイルの SIZE パラメータの単位の既定値は TRACK である (順編成ファイルでは BLOCK である)。なお、直編成ファイルの領域を自動的に大きくすることはできない。

それでは、本題に入ろう。

３．統合ライブラリとファイルの管理

FILLIST や DELETE 等の統合ライブラリやファイルを管理するためのコマンドについてまとめておこう。

3.1. 統合ライブラリ名とデータ・ファイル名

今までファイル名(統合ライブラリ名とデータ・ファイル名を含む)は英字で始まる長さ 16 文字以内の英数字とハイフン(-)と下線(_)からなる文字列であるとしてきたが、実はこれは単純名と呼ばれるもので、ファイル名は単純名を . で結合した 44 文字以内の文字列であってもよい。 このとき、 . もこの文字数のなかに入るし、必ず先頭につく利用者番号と .(計 7 文字)もこの文字数のなかに入ることに注意しよう。

したがって、次の名前はすべてファイル名として正しいことになる。

```
SF1.DATA、  FILE-A.FORTRAN、  PROGA.FOR.NO12
ABC1.DEF2.GH13.JKL4.MN05.PQR6.STU7.VWX8.Y
A123456789012345.B123456789012345.C123456
```

ただし、例えば PROGA.FOR.NO12 というファイルを作成してしまうと、 NO12 という名前はこの単純名の並びでは最後にしかこない名前(ファイル・エントリ)として登録され、 PROGA と FOR という名前は途中にしかこない名前(ディレクトリ)として登録されてしまう。 したがって、PROGA.FOR や PROGA.FOR.NO12.A というファイルを作れなくなる。しかし、異なる単純名の並びの中ではこの限りではないので、 P.FOR や A.PROGA.FOR や Q.FOR.NO12.A というファイルを作ることは可能である。このような機能はファイルのカタログ機能と呼ばれている。

統合ライブラリ中のサブファイルの指定は、 サブファイル名を ( ) でくくって統合ライブラリ名の後に並べて、統合ライブラリ名(サブファイル名) とすればよい。 統合ライブラリ名にはファイル名と同じ規則が適用され、サブファイル名は前に述べたように英字で始まる長さ 28 文字以内の英数字と下線(_)からなる文字列である。

3.2. 統合ライブラリとファイル情報の表示

ファイルの情報は FILLIST で見ることができる。 この項でファイルと言えば、ファイルと統合ライブラリの両方を意味することにする。

○FILLIST　　すべてのファイルの情報を表示する。

○FILLIST ファイル名　　指定されたファイルの情報のみを表示する。

○FILLIST 統合ライブラリ名 SUBFILE　　指定された統合ライブラリとその中のすべてのサブファイルの情報を表示する。

○FILLIST ファイル名 DETAIL　　指定されたファイルの詳しい情報を表示する。

○FILLIST ファイル名 SUMMARY　　指定されたファイルの DETAIL よりは簡単な情報を表示する。

○FILLIST 統合ライブラリ名 SUBFILE DETAIL
　　指定された統合ライブラリとその中のすべてのサブファイルの詳しい情報を表示する。

3.3. 統合ライブラリとファイルの内容の表示

ファイルの内容は LIST で見ることができる。この項でファイルと言えば、ファイルと統合ライブラリ中のサブファイルの両方を意味することにする。

○LIST ファイル名　　指定されたデータ・ファイルの内容を表示する。

○LIST ファイル名 SL　　指定されたソース・ユニットの内容を表示する。

○LIST ファイル名 PR　　指定された印刷イメージのファイルの内容を表示する。

○LIST ファイル名 HEX　　指定されたファイルの内容を 16 進数で表示する。

○LIST ファイル名 MIX　　指定されたファイルの内容を文字と 16 進数で表示する。

○LIST ファイル名 RANGE=(a,b,c)　　指定されたファイルの行番号 a から行番号 b までを c

番号おきに表示する。データ・ファイルの行番号は、1、2、…とする。

3.4. ファイルの消去

ファイルの消去は DELETE で行なう。この項でファイルと言えば、ファイルと統合ライブラリの両方を意味することにする。

○DELETE ファイル名　　指定されたファイルを消去する。

○DELETE 統合ライブラリ名(サブファイル名) 型
　　指定された統合ライブラリ中の指定された型のサブファイルを消去する。ソース・ユニットの型は SL を指定する。

○DELETE 統合ライブラリ名(サブファイル名,サブファイル名,…) 型
　　指定された統合ライブラリ中の指定された型の指定されたすべてのサブファイルを消去する。ソース・ユニットの型は SL を指定する。

3.5. 統合ライブラリ名とファイル名の星印規則

統合ライブラリやファイルの数が多くなってくると、名前がよく似たものをまとめて操作する機能が欲しくなる。ATSS-AF では、次の星印規則で統合ライブラリやファイルを指定することができる。以下では、～は単純名を表わすものとする。

(1) ファイル名の最後の単純名が＊のときは、対応する位置の任意の単純名を表わす。

○FILLIST F1.＊　名前が F1.～ というファイルの情報を表示する。F1.～.～ や F1.～.～.～ などは表示されない。

○FILLIST F1.DATA.＊　名前が F1.DATA.～ というファイルの情報を表示する。F1.DATA.～.～ や F1.DATA.～.～.～ などは表示されない。

○FILLIST ＊ SUBFILE　すべてのファイルとサブファイルの情報を表示する

○DELETE F1.＊　名前が F1.～というすべてのファイルを消去する。F1.～.～や F1.～.～.～ などは消去されない。該当するファイルに対して消去するかどうかを順に問い返してくる。

○DELETE F1.DATA.＊ NOCONFIRM
　　名前が F1.DATA.～ というすべてのファイルを消去する。いまは NOCONFIRM パラメータを指定しているので、消去するかどうかを問い返さない。

この ＊ 指定は LIST では使えない。

(2) ファイル名の最後の単純名が＊＊のときは、対応する位置から後の　．で区切られた任意の単純名の並びを表わす。

○FILLIST F1.**　名前が F1.～、F1.～.～、F1.～.～.～ などのファイルの情報を表示する。

○FILLIST F1.DATA.**
名前が　F1.DATA.～、F1.DATA.～.～、F1.DATA.～.～.～ などのファイルの情報を表示する。

○DELETE F1.**　名前が F1.～、F1.～.～、F1.～.～.～ などのファイルすべてを消去する。該当するファイルに対して消去するかどうかを問い返してくる。NOCONFIRM パラメータをつければ、問い返さずに該当するファイルをすべて消去する。

○DELETE **　すべてのファイルを消去する。すべてのファイルに対して消去するかどうかを問い返してくる。NOCONFIRM パラメータをつければ、問い返さずにすべてのファイルを消去する。

この ** 指定も LIST では使えない。

3.6. サブファイル名の星印規則

統合ライブラリの中のサブファイルに対しても、星印規則が使用できる。こちらの方がファイルに対するものより柔軟な指定が可能である。

(1) サブファイル名の一部が？のときは、対応する位置の任意の１文字を表わす。

○FILLIST ILIB(A?) SUBFILE
統合ライブラリ ILIB 中で名前が A で始まり長さが 2 文字のサブファイルの情報を表示する。

○LIST ILIB(A??)　統合ライブラリ ILIB 中で名前が A で始まり長さが 3 文字のサブファイルの内容を表示する。

○DELETE ILIB(???) SL　統合ライブラリ ILIB 中で名前の長さが 3 文字のサブファイルで型が SLのものをすべて消去する。 このとき、該当するサブファイルに対して順に消去するかどうかを問い返してくる。NOCONFIRM パラメータをつければ、問い返さずに該当するサブファイルをすべて消去する。

(2) サブファイル名の一部が＊のときは、対応する位置の任意の文字列(空列を含む)を表わす。

○FILLIST ILIB(A*) SUBFILE
統合ライブラリ ILIB 中で名前が　A で始まるサブファイルの情報を表示

する。このとき、サブファイル A の情報も表示される。

○FILLIST ILIB(A*Z) SUBFILE

統合ライブラリ ILIB 中で名前が A で始まり、 Z で終わるサブファイルだけの情報を表示する。このとき、サブファイル AZ の情報も表示される。

○LIST ILIB(*FORT*)　統合ライブラリ ILIB 中で名前のなかに FORT を含むサブファイルの内容を表示する。このとき、サブファイル FORT の内容も表示される。

○DELETE ILIB(*F*A*) SL

統合ライブラリ ILIB 中で名前のなかに F と A をこの順序で含むサブファイルで型が SL のものを消去する。このとき、該当するサブファイルに対して消去するかどうかを順に問い返してくる。NOCONFIRM パラメータをつければ、問い返さずに該当するサブファイルをすべて消去する。

○DELETE ILIB(*) SL　統合ライブラリ ILIB 中のすべてのサブファイルを消去する。順に消去するかどうかを問い返してくる。NOCONFIRM パラメータをつければ、問い返さずにすべてのサブファイルを消去する。

** や *? などの意味のない指定は、エラーとなる。

(3) サブファイル名の先頭が ^ のときは、その後ろ全体の名前と一致しないサブファイルを表わす。

○FILLIST ILIB(^A*) SUBFILE

統合ライブラリ ILIB 中で 名前が A で始まらないサブファイルの情報を表示する。

○LIST ILIB(^??)　統合ライブラリ ILIB 中で 名前の長さが 2 文字でないファイルの内容を表示する。

○DELETE ILIB(^*F*A*) SL

統合ライブラリ ILIB 中で名前のなかに F と A がこの順序で含まないサブファイルで型が SL のものを消去する。

指定 ^* は一致するものが常に存在しないので、エラーとなる。

3.7. ファイルのコピー

ファイルのコピーには、COPY コマンドを用いる。 このとき、新たにデータ・ファイルやサブファイルを作成する場合でも、もとのファイルの属性をそのまま使用するので、前もって領域を確保しておく必要はない(統合ライブラリは確保する必要がある)。

○COPY ファイル名1　ファイル名2

指定されたファイル1をファイル2にコピーする。

○COPY 統合ライブラリ名(サブファイル名)　ファイル名

指定された統合ライブラリのサブファイルをファイルにコピーする。

○COPY ファイル名　統合ライブラリ名(サブファイル名)

指定されたファイルを統合ライブラリのサブファイルにコピーする。このとき、指定されたサブファイルがすでに存在しているときはコピーしない。

○COPY ファイル名　統合ライブラリ名(サブファイル名)　REPLACE

指定されたファイルを統合ライブラリのサブファイルにコピーする。このとき、指定されたサブファイルがすでに存在していてもコピーする。

○COPY 統合ライブラリ名1(サブファイル名1)　統合ライブラリ名2(サブファイル名2)　型

指定された統合ライブラリ1の指定された型のサブファイル1を統合ライブラリ2のサブファイル2にコピーする。 統合ライブラリ名2が省略されていれば、統合ライブラリ名1と考え、(サブファイル名2)が省略されていれば、(サブファイル名1)と考える。ソース・ユニットの型は SL を指定する。また、REPLACE パラメータをつければ、サブファイル2 がすでに存在していてもコピーする。

○COPY 統合ライブラリ名1(サブファイル名11,サブファイル名12,…)

統合ライブラリ名2(サブファイル名21,サブファイル名22,…)　型

指定された統合ライブラリ1の指定された型のサブファイル11を 統合ライブラリ2のサブファイル21に、統合ライブラリ1のサブファイル12を統合ライブラリ2のサブファイル22に、……にコピーする。統合ライブラリ名2が省略されていれば、統合ライブラリ名1と考える。 ソース・ユニットの型は SL を指定する。さらに、REPLACE パラメータをつければ、サブファイル名21、サブファイル名22、…がすでに存在していてもコピーする。

○COPY 統合ライブラリ名1　統合ライブラリ名2　ALL

指定された統合ライブラリ1のすべてのサブファイルを統合ライブラリ2にコピーする。 このとき、統合ライブラリ2に同じ名前のサブファイル存在していれば、そのサブファイルはコピーしない。REPLACE パラメータをつければ、統合ライブラリ2に存在しているサブファイルもコピーする。

COPY コマンドのファイルやサブファイルの指定においても、星印規則を使うことができる。 ただし、ファイルに対して ** は使えないし、サブファイルに対しては * は 1 個しか使えず、* と ? の順序が第 1 パラメータと第 2 パラメータで一致しなくてはならない。また、^ は使えない。

○COPY F1.* F2.*　F1.* の * に対応する単純名を F2.* の * に置き換えてコピーする。例えば、F1.A を F2.A に、F1.DATA を F2.DATA にコピーする。

○COPY ILIB1(A*) ILIB2(B*)

例えば、サブファイル ABCD を BBCD に、A_PROG1 を B_PROG1 にコピーする。

○COPY ILIB1(????A_*) ILIB2(????*)

例えば、サブファイル PROGA_A1 を PROGA1 に、PROGA_X12 を PROGX12 にコピーする。

3.8. ファイル名の付け替え

ファイル名の変更には、RENAME コマンドを用いる。

○RENAME ファイル名1　ファイル名2

指定されたファイル1をファイル名2に変更する。

○RENAME 統合ライブラリ名(サブファイル名1)（サブファイル名2)

指定された統合ライブラリのサブファイル1をサブファイル名2に変更する。

○RENAME 統合ライブラリ名(サブファイル名11,サブファイル名12,…)

（サブファイル名21,サブファイル名22,…)

指定された統合ライブラリのサブファイル11をサブファイル名21に、サブファイル12をサブファイル名22に、……に変更する。

RENAME コマンドのファイルやサブファイルの指定においても、星印規則を使うことができる。ただし、ファイルに対して、** は使えないし、サブファイルに対しては * は 1 個しか使えず、* と ? の順序が第 1 パラメータと第 2 パラメータで一致しなくてはならない。また、^ は使えない。

○RENAME F1.* F2.*　F1.* の * に対応する単純名を F2.* の * に置き換えたファイル名に変更する。例えば、F1.A を F2.A に、F1.PROG を F2.PROG に変更する。

○RENAME ILIB1(A*) ILIB2(b*)

例えば、サブファイル ABCD を BBCD に、A_PROG1 を B_PROG1 に変更する。

○RENAME ILIB1(????A_*) ILIB2(????*)

例えば、サブファイル PROGA_A1 を PROGA1 に、PROGA_X12 を PROGX12 に変更する。

4．行エディタ

2.3 節では、サブコマンドを使わない修正のしかたについて述べたが、行エディタには、もちろん、多くのサブコマンドがある。行エディタのサブコマンドについてまとめておこう。

4.1. 行エディタのサブコマンド

行エディタのサブコマンドには、指定されたサブコマンドに対応する処理を行ないすぐに E# のプロンプトを返す編集サブコマンドと、行の入力を必要とするために行番号のプロンプトを返し、その行の入力を待つ入力サブコマンドとがある。入力サブコマンドでは、NEW でエディタに入ったときと同じように [F を入力すると E# のプロンプトにもどる。

以下の項目(1)、(2)で、入力サブコマンドと編集サブコマンドについてまとめる。サブコマンドの見出しで、英大文字と特殊文字はその文字自身の入力を表わし、英小文字については説明中で述べてある意味を表わす(パターンについては 4.2 節を、行については 4.3 節を参照のこと)。また、説明の最後の ( ) の中はパラメータを省略したときの既定値を表わしており、 ここで見出しと同じ部分は省略できないパラメータを意味する。また、[ ] の中はサブコマンドを実行した後の現在行を表わす。

(1) 入力サブコマンド

○A n,a　　行 n の後に増分 a で行を追加する。ただし、n>¥。入力の終わりは [F。(¥+a,10)、[追加した最後の行]。

○I n,a　　行 n とその次の行との間に増分 a で行を追加する。入力の終わりは [F。(.,10)、[最後に挿入した行]。

○C n1:n2,a　　行 n1 から n2 までを増分 a の行で置き換える。入力の終わりは [F。(.:.,10)、[入力した最後の行]。

(2) 編集サブコマンド

○L n1:n2　　行 n1 から n2 までの行番号と内容を表示する。(.:.)、[n2]。

○GL @str@,n1:n2　　行 n1 から n2 のうち文字列 str を含む行の行番号と内容を表示する。(@str@,^:¥)、[n2]。

GL /pat/,n1:n2　　行 n1 から n2 のうちパターン pat を含む行の行番号と内容を表示する。(/pat/,^:¥)、[n2]。

○XL @str@,n1:n2　　行 n1 から n2 のうち文字列 str を含まない行の行番号と内容を表示する。(@str@,^:¥)、[n2]。

XL /pat/,n1:n2　　行 n1 から n2 のうちパターン pat を含まない行の行番号と内容を表示す

る。(/pat/,^:¥)、[n2]。

○D n1:n2　　行 n1 から n2 までを削除する。(.:.)、[n2 の次の行]。

○S @str1@str2@,n1:n2

行 n1 から n2 までの行に対してすべての文字列 str1 を文字列 str2 で置き換える。(@str1@str2@,.:.)、[n2]。

S /pat/str/,n1:n2

行 n1 から n2 までの行に対してすべてのパターン pat を文字列 str で置き換える。(/str1/str2/,.:.)、[n2]。

○CB c1:c2　　文字列の探索範囲を桁 c1 から c2 に変更する。(1:72)、[.]。

○N n　　現在行を行 n にする他は何もしない。(.)、[n]。

○M n1:n2,n3,a　行 n1 から n2 までの行を行 n3 の後に増分 a で移動させる。
(.:.,n3,10)、[最後に移動された行(n3 の後の側)]。

○K n1:n2,n3,a　行 n1 から n2 までの行を行 n3 の後に増分 a でコピーする。
(.:.,n3,10)、[最後にコピーされた行(n3 の後の側)]。

○NB n1,a,n2:n3　行番号 n1 から増分 a で行 n2 から n3 までの行番号を付け直す。
(10,10,^:¥)、[変化せず]。

○FILE ?　　現在、編集している 統合ライブラリ名(サブファイル名) を表示する。例えば、ILIB1(F1) が表示される。この統合ライブラリを「編集統合ライブラリ」と呼ぶことにする。初期値は EDIT コマンドで指定した統合ライブラリ名である。

○FILE lib　　編集統合ライブラリ名を lib に変更する。

○R sf　　現在の編集用バッファをクリアし、編集統合ライブラリのサブファイル sf の内容を読み込む。(sf)、[最後に読み込んだ行]。

○RA sf,fn1:fn2,n,a

編集統合ライブラリ中のサブファイル sf の行 fn1 から fn2 までを読み込み、バッファ中の行 n の後に増分 a で追加する。(sf,sf全体,¥,10)、[最後に読み込んだ行]。

○W sf,n1:n2　　行 n1 から n2 までを編集統合ライブラリのサブファイル sf に書き出す。ファイルの上書きを確認する。(編集中サブファイル,^:¥)、[変化せず]。

○Z sf,n1:n2　　行 n1 から n2 までを編集統合ライブラリのサブファイル sf に書き出す。上書きの確認はしない。(編集中サブファイル,^:¥)、[変化せず]。

○E または Q　　エディタを終了する。

○E W または Q W　バッファ全体をファイルに書き出し、エディタを終了する。

以上、エディタのサブコマンドについてまとめた。ここでは、サブコマンド名としては省略形を用いた。 省略しないサブコマンド名には、例えば、LIST や DELETE や SUBSTITUTE などがある。

すると、DELETE などは普通のコマンドにも同じものがあるので、

```
E#*DELETE ILIB1(F1)
```

として、統合ライブラリ ILIB1 のサブファイル F1 を削除するつもりで、* を忘れて、

```
E#DELETE ILIB1(F1)
```

とすると、エディタのサブコマンドとして解釈され、

```
ED410 INVALID SUBCOMMAND NAME: ILIB1(F1)
```

のような正体不明のメッセージが出てしまうので、注意をする必要がある。普通のコマンドと同じ綴りのエディタのサブコマンドは、現在のところ、

D - DELETE、 L - LIST、 K - COPY、

の３つだけである。

4.2. パターン

GL、XL、S サブコマンドではパターンは / / ではさむことにより指定した。しかし、パターンをはさむ記号は / でなくてもスペースと @ 以外の文字ならば何でもよく、例えば、 ?パターン? や :パターン: としてもよい。そして、パターンの中で特別の意味を持つ文字がある。

- ^　パターンの最初にあれば、行の先頭の文字の左(行の始め)を表わす。
  例　/^SUB/　行の先頭にある文字列 SUB を表わす。
  /X^2/　^ が行の先頭にないので、単に文字列 X^2 を表わす。
- ¥　パターンの最後にあれば、行の最後の文字の右(行の終わり)を表わす。
  例　/+Y)¥/　行の最後にある文字列 +Y) を表わす。
- .　任意の 1 文字を表わす。
  例　/A../　文字 A で始まる 3 文字の文字列を表わす。
- *　直前の文字が任意個続いた文字列(空文字列でもよい)を表わす。
  例　/AB*C/　文字列 AC、ABC、ABBC、ABBBC、… のいずれかを表わす。
  /A.*C/　文字 A で始まり、文字 C で終わる文字列を表わす。文字列 AC でもよい。
- [C　直後の特別な意味を持つ文字の意味を無効にし、単にその文字を表わす。いわゆる、エスケープ文字である。すべての特別な意味を持つ文字の意味を無効にするには、@ @ ではさめばよい。
  例　/3[C.14159/　文字列 3.14159 を表わす。 . の特別な意味を打ち消している。

これは @3.14159@ と書いても同じである。

/[C^[C/[C.[C*[C¥/　文字列 ^/.*¥ を表わす。@^/.*¥@ と同じである。

よく使用する S サブコマンドの例をいくつか与えよう。

○S サブコマンド

例　S /INTEGER/REAL/　現在行のすべての文字列 INTEGER を REAL に置き換える。

S @1.8@1,8@　現在行のすべての文字列 1.8 を 1,8 に置き換える。

S /A.../　現在行のすべての A で始まる 3 文字の文字列を削除する。

S ?/.*/?//?　現在行のすべての / で始まり / で終わる文字列を // に置き換える(/ と / の間を削除)。

また、S /pat/str/ の str の中の & は pat でマッチした文字列全体を表わす。

例　S /A+B/(&)/　& はパターンが一致した文字列を表わすので、現在行のすべての A+B を (A+B) に置き換えることになる。

S /F.../&&/　現在行のすべての F で始まる 4 文字の文字列を それを 2 回繰り返した 8 文字の文字列に置き換える。

また、サブコマンド GL と XL のパターンの中では、次のような表記も可能である。

○GL /パターン1 ! パターン2 ! … ! パターン10/

XL /パターン1 ! パターン2 ! … ! パターン10/

パターン1、パターン2、…、パターン10 の いずれかのパターンを含む(GL)または含まない(XL)行を行番号とともに表示する。! は「または」を表わす。

例　GL /ABC!XYZ/　文字列 ABC または XYZ を含むすべての行を行番号と共に表示する。

○GL /パターン1 & パターン2 & … & パターン10/

XL /パターン1 & パターン2 & … & パターン10/

パターン1、パターン2、…、パターン10 のすべてのパターンを含む(GL) または含まない(XL)行を行番号とともに表示する。& は「かつ」を表わす。

例　XL /ABC&XYZ/　文字列 ABC も XYZ も含まないすべての行を行番号と共に表示する。

4.3. 行の指定

サブコマンドの中で行を指定する方法には、次のものがある。

○行番号　指定した行番号の行を表わす。例えば、10、00020 など。

○ ^　バッファの最初の行を表わす。

○ ¥　バッファの最後の行を表わす。

○　　現在行を表わす。 原則として、直前に操作を行なった行をさしている(4.1 節の各サブコマンドの [ ] の中を参照)。既存のファイルや統合ライブラリ中のサブファイルを指定してエディタに入ったときには、まずバッファへの読み込みを行なう(R サブコマンドに当たる)ので、現在行は ¥ となる。

○ @文字列@ または /パターン/

現在行の次の行から調べて、最初に文字列(またはパターン)を含む行を表わす。このとき、バッファの最後まで調べたら、最初の行から現在行までを調べる。行指定のパターンは / / でしかはさめない。

例　/FUNCTION/、 @/@、 @3.14159@、 @X**3@、 /A../、 /S.*E/ など。

○行 + 行数　指定された行の指定された行数分だけ後の行を表わす。

行 - 行数　指定された行の指定された行数分だけ前の行を表わす。

例　^+3、 ¥-10、 .+1、 @FUNCTION@+1、 /S.*E/-1 など。

以上のパターンと行の指定のしかたを頭において、もう一度、4.1 節のサブコマンドを見て頂きたい。前には気が付かなかったいろいろなことを表現できるのに気が付くだろう(例えば、n @str@ とすれば、次に文字列 str を含む行に現在行を移すことができる)。

5．コンパイル・ユニットとロード・モジュールの作成

コンパイル・ユニットやロード・モジュールを作成しておくと、実行するたびごとに翻訳や結合を行なわなくてすむので、特に、ソース・プログラムのサイズが大きいときには有効である。CP で実行させるロード・モジュールを作成する場合と AP で実行させるロード・モジュールを作成する場合とでは、異なるコンパイラを使用する必要があるので、以下、5.1 節では、 CP 用のコンパイル・ユニットとロード・モジュールを作成し、実行させる方法を、5.2 節では、 AP 用のコンパイル・ユニットとロード・モジュールを作成する方法について述べる。

5.1. ＣＰ用のコンパイル・ユニットとロード・モジュールの作成

(1) コンパイル・ユニットの作成(CP 用)

統合ライブラリ ILIB1 のサブファイル F1 の CP 用のコンパイル・ユニットが必要なときは、

```
*EDIT ILIB1(F1)
CREATED  UPDATED  REV    LINE NO-OF-BYTE LANTYPE ORIGINAL-ENTRY-NAME
86-04-20 86-04-20 000 0000000 0000000000 FTFREE2 F1
E*FORT77 CULIB=ILIB1                          ; CP 用の Fortran 77 コンパイラ
FORTRAN 77 ENTERED
PROGRAM : FACT
NO ERROR
OBJECT PROGRAM PRODUCED
```

```
PROGRAM : F
NO ERROR
OBJECT PROGRAM PRODUCED
E#
```

とすると、CULIB で指定した統合ライブラリ ILIB1 の中にコンパイル・ユニットが作成される。

```
E#*FILLIST ILIB1 SUBFILE
                                /***  NAME  LIST  ***/            FILLIST REV.071
  NO. FILE / SUBFILE NAME                FILEORG  TYPE      PWORD  RUAF STATUS
 0001 A60000.ILIB1                       LINKQD   FILE      NO     YES
    1 A1                                          SL
    2 F1                                          SL
    3 S1                                          SL
    4 F                                           CU(MAIN)
    5 FACT                                        CU(MAIN)
E#
```

このとき、コンパイル・ユニット(CU)は、サブプログラムごとに1つのサブファイルとして作成される。主プログラムは、名前が付いているときには、その名前のサブファイルが作られ(Fortran では、program 文で主プログラムに名前を付けることができる)、 名前が付いていないときには、FTMAIN というサブファイルが作られる。 したがって、1つの統合ライブラリにプログラムをいくつか入れていると、コンパイルしたときにコンパイル・ユニットの名前に重複が起きることがあるので注意を要する。1つの統合ライブラリには1つ(1種類)のプログラムを入れておくとよい。

また、コンパイル・リストが必要な場合には、PRFILE パラメータを指定して、

```
E#FORT77 CULIB=ILIB1 PRFILE=* SOURCE                ; コンパイル・リストは端末に
FORTRAN 77 ENTERED
SOURCE LIST
      ELN      ILN     FORTRAN STATEMENT
   000010        1           PROGRAM FACT
   000020        2           INTEGER N,F
   000030        3         1 READ(5,*) N
   000040        4           IF (N.LT.0) STOP
   000050        5           WRITE(6,*) F(N)
   000060        6           GOTO 1
   000070        7           END
SOURCE LIST
      ELN      ILN     FORTRAN STATEMENT
   000080        1     C
   000090        2           INTEGER FUNCTION F(N)
   000100        3           INTEGER N,J
   000110        4           F=1
   000120        5           DO 10 J=1,N
   000130        6             F=J*F
   000140        7        10 CONTINUE
   000150        8           END
```

```
*************************************************************************
*                                                                       *
*   FACT            NO ERROR                                    CU      *
*                                                                       *
*   F               NO ERROR                                    CU      *
*                                                                       *
*   ERROR TOTAL :  F : 0    S : 0    W : 0    0 : 0                     *
*                                                                       *
*************************************************************************
E#
```

とすればよい。さらに、コンパイル・リストをファイルに入れたいときは、

```
E#FORT77 CULIB=ILIB1 PRFILE=ILIB1(PF1) SOURCE  ; コンパイル・リストをILIB1(PF1)に
FORTRAN 77 ENTERED
PROGRAM : FACT
NO ERROR
OBJECT PROGRAM PRODUCED
PROGRAM : F
NO ERROR
OBJECT PROGRAM PRODUCED
E#
```

とすればよい。 ファイルの確認は FILLIST コマンドや LIST コマンドにより可能である。このとき、FILLIST コマンドでみると、サブファイル ILIB1(PF1) の型は SL となっているが、 LIST コマンドで表示するときには型 PR を指定する。そして、このサブファイルをセンターのライン・プリンタや日本語プリンタに印刷したいときには、ACOS-1000 に逆転送[5]して、 $FPRI コマンドや $FPRIJ コマンドを用いればよい。なお、ここで述べた以外のパラメータについては、文献[4]を参照されたい。

また、エディタの中ではなく、* のプロンプトで同じことを行なうには、ソース・ユニットを指定して、

```
E#E
*FORT77 ILIB1(F1) CULIB=ILIB1
FORTRAN 77 ENTERED
PROGRAM : FACT
NO ERROR
OBJECT PROGRAM PRODUCED
PROGRAM : F
NO ERROR
OBJECT PROGRAM PRODUCED
*
```

のようにすればよい。

(2) コンパイル・ユニットからの実行(CP 用)

CP 用のコンパイル・ユニットから実行を行なうには LOADGO コマンドを使う。 統合ライブラリ ILIB1 の中のコンパイル・ユニット FACT を実行させるには、

```
*LOADGO FACT INLIB=ILIB1
EXECUTION STARTED
?5
120
?8
40320
?-1
*
```

とすればよい。 このとき、FACT から呼ばれているすべてのサブプログラムのコンパイル・ユニットが ILIB1 の中になければならない。 必要なコンパイル・ユニットが複数の統合ライブラリに渡って存在しているときは、 INLIB パラメータを INLIB=(ILIB1,ILIB2,ILIB3,ILIB4) のようにすれば、最高 4 個の統合ライブラリ中のコンパイル・ユニットを結合できる(数値計算用のライブラリを結合したいときは、ILIB2 などの代わりに 'LIB.ASLCP' や 'LIB.MLIBCP' を指定すればよい)。このとき, RUN コマンドと同じようにロード・モジュールのサブファイルは作成されない。

実行時の CPU 時間を指定するには、CPTIME パラメータ(CPUTIME ではない)を指定して、

```
*LOADGO FACT INLIB=ILIB1 CPTIME=60                    ; CPU 時間を 60 秒までに
EXECUTION STARTED
?10
3628800
?5
120
?-1
*
```

とすればよい。単位は秒である。

(3) ロード・モジュールの作成

コンパイル・ユニットからロード・モジュールを作成するには、LINK コマンドを使う。 統合ライブラリ ILIB1 の中のコンパイル・ユニット FACT からロード・モジュールを作成するには、

```
*LINK ILIB1(FACT) INLIB=ILIB1
VER. R1.21  STLNK REV. 211  USER LM : FACT REV.000
AN EXECUTABLE LOAD MODULE HAS BEEN PRODUCED
NO ERRORS DETECTED
*FILLIST ILIB1 SUBFILE
                        /***  NAME  LIST  ***/            FILLIST REV.071
  NO. FILE / SUBFILE NAME            FILEORG  TYPE      PWORD  RUAF STATUS
 0001 A60000.ILIB1                   LINKQD   FILE      NO     YES
```

```
    1 A1                                    SL
    2 F1                                    SL
    3 PF1                                   SL
    4 S1                                    SL
    5 F                                     CU(MAIN)
    6 FACT                                  CU(MAIN)
    7 FACT                                  LM
*
```

とすればよい。 このとき、1 番目のパラメータである ILIB1(FACT) から呼ばれているすべてのサブプログラムのコンパイル・ユニットが ILIB1 の中になければならない。 必要なコンパイル・ユニットが複数の統合ライブラリに渡って存在しているときは、INLIB パラメータを INLIB=(ILIB1, ILIB2,…,ILIB9) のようにすれば、最高 9 個の統合ライブラリ中のコンパイル・ユニットを結合できる(数値計算用ライブラリを結合したいときは、ILIB2 などの代わりに 'LIB.ASLCP' や 'LIB.MLIBCP' を指定すればよい)。

また、コンパイル・ユニットと異なる名前のロード・モジュールを作成するには、次のようにすればよい。

```
*LINK ILIB1(PROG1) INLIB=ILIB1 ENTRY=FACT
VER. R1.21  STLNK REV. 211  USER LM : PROG1 REV.000
AN EXECUTABLE LOAD MODULE HAS BEEN PRODUCED
NO ERRORS DETECTED
*
```

ここでは、統合ライブラリ ILIB1 の中のコンパイル・ユニットに対して FACT を主プログラムとして、FACT から呼ばれているすべてのサブプログラムを結合し、ILIB1(PROG1) というロード・モジュールを作成した。

```
*FILLIST ILIB1 SUBFILE
                        /***  NAME  LIST  ***/          FILLIST REV.071
  NO. FILE / SUBFILE NAME          FILEORG  TYPE     PWORD  RUAF STATUS
 0001 A60000.ILIB1                 LINKQD   FILE     NO     YES
    1 A1                                    SL
    2 F1                                    SL
    3 PF1                                   SL
    4 S1                                    SL
    5 F                                     CU(MAIN)
    6 FACT                                  CU(MAIN)
    7 FACT                                  LM
    8 PROG1                                 LM
*
```

(4) ロード・モジュールからの実行(CP 用)

CP 用のロード・モジュールを実行させるには、CALL コマンドを用いる。

```
*CALL ILIB1(FACT)
?5
120
?8
40320
?-1
*CALL ILIB1(PROG1)
?10
3628800
?5
120
?-1
*CALL ILIB1(PROG1) CPTIME=30                    ; CPU 時間の指定
?8
40320
?10
3628800
?-1
*
```

(5) 管理用コマンド

統合ライブラリのサブファイルとして作成されたコンパイル・ユニットやロード・モジュールの情報をみるために、次のような FILLIST コマンドの使い方がある。

```
*FILLIST ILIB1 SL
                          /***  NAME  LIST  ***/          FILLIST REV.071
  NO. FILE / SUBFILE NAME          FILEORG  TYPE      PWORD  RUAF STATUS
 0001 A60000.ILIB1                 LINKQD   FILE      NO     YES
    1 A1                                    SL
    2 F1                                    SL
    3 PF1                                   SL
    4 S1                                    SL
*FILLIST ILIB1 CU
                          /***  NAME  LIST  ***/          FILLIST REV.071
  NO. FILE / SUBFILE NAME          FILEORG  TYPE      PWORD  RUAF STATUS
 0001 A60000.ILIB1                 LINKQD   FILE      NO     YES
    1 F                                     CU(MAIN)
    2 FACT                                  CU(MAIN)
*FILLIST ILIB1 LM
                          /***  NAME  LIST  ***/          FILLIST REV.071
  NO. FILE / SUBFILE NAME          FILEORG  TYPE      PWORD  RUAF STATUS
 0001 A60000.ILIB1                 LINKQD   FILE      NO     YES
    1 FACT                                  LM
    2 PROG1                                 LM
*
```

5.2. AP用のコンパイル・ユニットとロード・モジュールの作成

AP 用のコンパイル・ユニットとロード・モジュールの作成も SX-1 の TSS で可能であるが、AP

による実行はバッチ処理でのみ可能であり、ACOS-1000 の CARDIN サブシステムを使わなければならない(コンパイル・ユニットやロード・モジュールを ACOS-1000 に逆転送する必要はない)。

(1) コンパイル・ユニットの作成(AP 用)

AP 用のコンパイル・ユニットを作成するには、FORT77SX というコンパイラを使う点が異なるだけで、CP 用の場合とほとんど同じである。

```
*EDIT ILIB1(F1)
CREATED  UPDATED  REV    LINE NO-OF-BYTE LANTYPE ORIGINAL-ENTRY-NAME
86-04-20 86-04-20 000 0000000 0000000000 FTFREE2 F1
E*FORT77SX CULIB=ILIB1 PRFILE=* SOURCE
FORTRAN 77/SX ENTERED
SOURCE LIST
       ELN      ILN     FORTRAN STATEMENT
    000010        1           PROGRAM FACT
    000020        2           INTEGER N,F
    000030        3         1 READ(5,*) N
    000040        4           IF (N.LT.0) STOP
    000050        5           WRITE(6,*) F(N)
    000060        6           GOTO 1
    000070        7           END
SOURCE LIST
       ELN      ILN     FORTRAN STATEMENT
    000080        1     C
    000090        2           INTEGER FUNCTION F(N)
    000100        3           INTEGER N,J
    000110        4           F=1
    000120        5           DO 10 J=1,N
    000130        6             F=J*F
    000140        7        10 CONTINUE
    000150        8           END
DIAGNOSTIC LIST
       ELN      LEVEL    NO.   DIAGNOSTIC MESSAGE
    000120        VEC      1 : VECTORIZED BY DO INDEX J
    ********************************************************************************
    *                                                                              *
    *   FACT           NO ERROR                                          CU      *
    *                                                                              *
    *   F              NO ERROR                                          CU      *
    *                                                                              *
    *   ERROR TOTAL :  F : 0    S : 0    W : 0    O : 0                            *
    *                                                                              *
    ********************************************************************************
E*E
*
```

ここでは、DIAGNOSTIC LIST として、ベクトル化情報が出ている。コンパイル・リストを作成する場合や、* のプロンプトの場合も FORT77 を FORT77SX に変更するだけでよい。しかし、AP 用の

コンパイル・ユニットの型も CU であり、CP 用の同じ名前のコンパイル・ユニットがあると、その内容は壊れてしまうことになる。さらに、コンパイル・ユニットが CP 用であるか、AP 用であるかの情報は FILLIST コマンド等でみることができないので、CP 用と AP 用のコンパイル・ユニットは別の統合ライブラリに作成するようにした方がよいだろう。

(2) ロード・モジュールの作成

LINK コマンドは CP 用と AP 用の両方のコンパイル・ユニットに対して使えるので、AP 用のロード・モジュールの作成のしかたは、CP 用のロード・モジュールの作成のしかたとまったく同じである。ただし、数値計算用ライブラリは、'LIB.ASLAP' と 'LIB.MLIBAP' を使う。このときも、ロード・モジュールの名前は CP 用と区別しないので、同じ名前の CP 用のロード・モジュールが存在しているときは注意を要する。

(3) ロード・モジュールからの実行(AP 用)

AP ではバッチ・ジョブの実行のみが可能であり、バッチで SX-1 を使用するには ACOS-1000 の CARDIN サブシステムを使用する必要がある。SX-1 上に AP 用のコンパイル・ユニットやロード・モジュールがある場合のジョブ制御文については、文献[9]をされたい。なお、コンパイル・ユニットから実行を行なうための LOADGO コマンドは使えない (LOADGO コマンドは CP 上で実行させる)

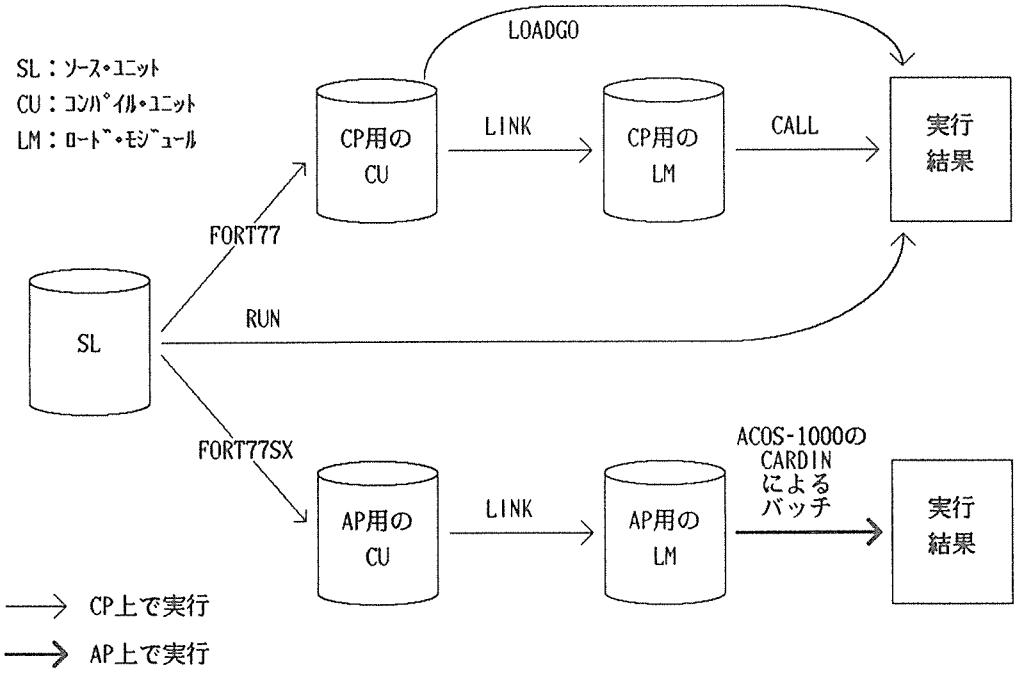

図1．ロード・モジュールとコンパイル・ユニットの作成

し、対応するコマンドもない。

5.3. コンパイル・ユニットとロード・モジュールのまとめ

以上の処理をまとめると図1のようになる。この図で、→ は CP 上で TSS により実行され、⇒ は AP 上でバッチにより実行される。このうち、RUN、FORT77、FORT77SX、LINK、CALL に対応するものは CP 上のバッチによっても実行できる。CP 上では TSS とバッチを処理することができるが、AP 上ではバッチしか処理できないことを思い出そう[1]、[2]。 そして、SX-1 のバッチへの投入はすべて ACOS-1000 の CARDIN サブシステムを使用して行なうことに注意しよう。

6. その他

(1) コマンドの一般形式

いままで、特に説明しなかったが、コマンドの一般的な形は次のようである。

*コマンド名　パラメータ1　パラメータ2 … パラメータn

ここで、コマンド名とパラメータ並びの間はスペースでなければならないが、パラメータ間の区切りは ，でもよい。 そして、パラメータには、書く位置の決まったもの(定位置パラメータ)と書く位置は任意でキィーワードで指定するもの(キィーワード・パラメータ)がある。定位置パラメータは主に省略できないパラメータ(例えば、ファイルや統合ライブラリなど)を指定するのに使われる。一方、キィーワード・パラメータはオプション的なものに使われる。なお、本稿で述べた以外のコマンドについては、文献[10]、[11]を参照されたい。

(2) コマンドの省略名

コマンド名のうち、長い綴り(5 文字以上)のものには省略形がある。今までは、省略形を用いなかったが、これは長い名前の方がもとの意味をとりやすく、最初のうちは分かりやすいと考えたからである。しかし、慣れてくるとわずらわしくなってくる。そこで、本稿で使用したコマンドとその省略形を示す。省略形が - のものは、省略形がないことを表わす。

ASSIGN	ASS	FORT77SX	FSX
ASSLIST	ASSL	FORT77SX	FSX
CALL	-	LIBALLOC	LIBA
CANCEL	CAN	LINK	-
COPY	-	LIST	-
DELETE	DEL	LOADGO	LG
EDIT	-	PREALLOC	PREA
FILLIST	FILL	RENAME	REN
FORT77	F77	RUN	-

また、パラメータのキィーワードは、他のパラメータのキィーワードと区別がつくところまで書

けば残りは省略可能である。

(3) ファイルのバックアップ

SX-1 上のファイルは種々の事情によりバックアップを行なわない。 ACOS-1000 上のファイルはバックアップを行なっているので、重要な SX-1 上のファイルは ACOS-1000 に 逆転送しておけばよい。しかし、ユーザがファイルを１つずつ逆転送するのは面倒なので、バックアップ用のツールがある。このツールについては、文献[12]を参照されたい。

7．おわりに

以上、スーパーコンピュータ SX-1 の TSS である ATSS-AF について、基本的な使い方、統合ライブラリとファイルの管理、行エディタ、コンパイル・ユニットとロード・モジュールの作成について述べた。SX-1 を使用する際の参考になれば幸いである。

[参考文献] (「その１」の掲載分も含む)

1. 渡辺、近藤、端山、大中、藤井 (1986)：「スーパーコンピュータ SX-1 の概要 (1) － アーキテクチャを中心に －」、 大阪大学 大型計算機センター ニュース、Vol.15、No.4 (1986 年 2 月号、第 60 号)、pp.109-129。
2. 藤井 (1986) ：「スーパーコンピュータ SX-1 の概要 (2) － 運用を中心に －」、大阪大学大型計算機センター ニュース、Vol.15、No.4 (1986 年 2 月号、第 60 号)、pp.131-136。
3. 大中、後藤 (1986) ：「SX Fortran77 概説 (1) － 文法を中心として －」、大阪大学 大型計算機センター ニュース、Vol.16、No.1 (1986 年 5 月号、第 61 号)、pp.17-35。
4. 後藤、大中 (1986) ：「SX Fortran77 概説 (2) － 利用法を中心として －」、大阪大学 大型計算機センター ニュース、Vol.16、No.1 (1986 年 5 月号、第 61 号)、pp.37-53。
5. 多喜 (1986)：「ACOS-1000 と SX-1 とのファイル転送について」、 大阪大学 大型計算機センター ニュース、Vol.16、No.1 (1986 年 5 月号、第 61 号)、pp.69-82。
6. MSF-6 利用説明書、SX ソフトウェア GJF11-1、日本電気 (1986)。
7. 三原、山本、後藤 (1986;掲載予定):「スーパーコンピュータ SX-1 の性能向上支援ツールの利用法」、大阪大学 大型計算機センター ニュース、Vol.16、No.2 (1986 年 11 月号、第 63 号)。
8. ATSS-AF 画面エディタ説明書、SX ソフトウェア GED12-2、日本電気 (1986)。
9. 後藤 (1986)：「スーパーコンピュータ SX-1 のジョブ制御言語入門」、大阪大学 大型計算機センター ニュース、Vol.16、No.2 (1986 年 8 月号、第 62号)。
10. ATSS-AF コマンド説明書 <メニュー編>、SX ソフトウェア GED14-1、日本電気 (1986)。
11. ATSS-AF コマンド説明書 <非メニュー編>、SX ソフトウェア GED11-2、日本電気 (1986)。
12. 「SX 利用者ファイルのバックアップ方法について」、大阪大学 大型計算機センター 速報、No.135 (1986 年 5 月 28 日発行)、pp.6-11。