OUKA
The University of Osaka
Institutional Knowledge Archive

Title	スーパーコンピュータ　SX-1のジョブ制御言語入門
Author(s)	後藤, 米子
Citation	大阪大学大型計算機センターニュース. 1986, 62, p. 53-65
Version Type	VoR
URL	https://hdl.handle.net/11094/65701
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 のジョブ制御言語入門

大阪大学大型計算機センター研究開発部
後　藤　米　子

1. はじめに

本稿では、SX－1の基本形利用法[2,12]によるジョブ制御言語（Job Control Language; JCL と略す）の概要を、AP（Arithmetic Processor）で実行するジョブに限定して、説明する。基本形が必要となるのは次のような場合である。(i) コンパイル・ユニットをパーマネントファイルに保存したいとき、(ii) ロード・モジュールをパーマネントファイルに保存したいとき、(iii) OPTIMIZER[1] を使用したいとき、(iv) システムが使用する作業域が既定値で不足するために自分で指定する必要が生じたとき、などである。

本論に入る前に、2節でファイルに関する簡単な復習をする。3節では基本形のジョブの基本的な構成を紹介し、これをもとに、4節および5節で目的別のジョブ構成法を説明する。最後の6節で、本文中に述べなかった項目を補足および注意事項としてまとめる。ファイルの作成については既に解説されている[6]が、プログラムで扱う記録の長さとの関連で5節で説明を補足した。SX側にデータファイルを持っている場合には、簡易形[2,12]を使用している方にも関係があるので、読んでいただきたい。

2. SX のファイルの概要

2.1 ファイルの作成と属性

SX側のパーマネントファイルには、(i) 統合ライブラリ[6]、(ii) 順編成ファイル、(iii) 直編成ファイル、の3種類がある。(i) にはソースプログラム、コンパイル・ユニットおよびロード・モジュールの3種類の型が登録でき、作成にはLIBALLOCコマンド[6]を用いる。(ii) および (iii) にはプログラムで使用するデータを登録でき、作成にはPREALLOCコマンド[6]を用いる。ここでいう“ファイルの作成”とはファイルの名前付け（この名前を外部ファイル名という）とその領域を確保することを意味し、この時点ではファイルの内容は空の状態である。内容を書き込むことを本稿では“登録”という。なお、ファイル領域の大きさについては別稿[6,7]に解説されているので、それを参照していただきたい。

ファイルの属性には (a) ファイル編成、(b) ブロック長、(c) レコード長、(d) レコード形式、(e) 割り付け方（FIXTRK, DELR, NODELRで示されるパラメータをここではいうことにする)、

(f) データ形式[10]（2.3節参照）の6つがあり、(f)以外はファイルの作成時に決定され、そのファイル内の特別な領域（ファイルラベルと呼ぶ）に書きこまれる。この情報はファイルの作成後は変更できない[10,11]。統合ライブラリではLIBALLOCコマンドの既定値のままでよい。順編成および直編成ファイルでは、登録するデータが書式付きか書式なしかで作成時に指定するべきパラメータが異なるので、5節でまとめて述べる。

2.2 ファイルの内容の登録

前節のようにして作成したファイルに内容を書き込むには、次のようにする。ソースプログラムの場合には、(i) ACOS 1000側のファイルをSSTAGEコマンド[6,8,12]でSX側に転送する、(ii) SXのTSSのエディタ[7]で新規に登録する、の2通りがある。いずれの場合も、登録先の識別にメンバー名（サブファイル名ともいう）が必要となる。データの場合にはソースプログラムの場合と同様にして登録できるけれども、(i)の方法では書式付きのみが、(ii)では順編成書式付きのみが登録できる。データであるから(iii)プログラムの実行時に書き出すことによっても登録できる。なお、この場合はファイルの接続にOPEN文を用いる[3]ことで、簡易形による利用法でも可能である。コンパイル・ユニットおよびロード・モジュールの登録については4.2節および4.4節で述べる。

2.3 データ形式（SSFとSARF）

データ形式にはSSF（システム標準形式）とSARF（標準アクセスレコード形式）の2種類がある。SSFとはプリンタ出力が可能な形式であり、一つのレコードはその先頭の8バイトのヘッダ（制御用）とデータとからなる。これに対して、SARFは磁気ディスクとの入出力などのシステム内部のみで入出力可能な形式である。

(1) ソースプログラムの場合

ソースプログラムはSSFであることが望ましい。その理由はSXのエディタではソースプログラムはSSFのみが処理できるからである。新規にエディタでソースプログラムを登録するときには、自動的にSSFとなるのでとくに何も指定しなくてよい。SSTAGEコマンドでは、ACOS 1000側のファイルが行番号付きであれば、SSFのファイルをSX側に登録する。固定形式の場合にも、コマンド（RESE#）で行番号を付加したのちこれを転送すると、SSFとなる。SSFでは行番号はヘッダ部に書かれるので、エディタ使用時においては文字列の検索の対象外となる。コンパイラはSSFおよびSARFのいずれの形式でも自動的に入力できる。

(2) データの場合

データでは表1に示すように定められており、内部ファイル名のSPRとSPU以外はSARFである。プリンタに出力したい場合には、SSFでなければならない。本稿ではSSFのパーマネントファイルは考慮しない。

表1　外部装置識別子と内部ファイル名との対応および既定値

（マニュアル[10]より転載）

外部装置識別子	内部ファイル名	ファイル編成	ブロック長	レコード長	レコード形式	データ形式
0	FF00	順編成	512	508	V	SARF
1	FF01	順編成	512	508	V	SARF
2	FF02	順編成	512	508	V	SARF
3	FF03	順編成	512	508	V	SARF
4	FF04	順編成	512	508	V	SARF
5	SIN	順編成	80	80	F	SARF
6	SPR	順編成	2016	140	VB	SSF
7	SPU	順編成	2016	88	VB	SSF
8	FF08	順編成	512	508	V	SARF
⋮	⋮	⋮	⋮	⋮	⋮	⋮
99	FF99	順編成	512	508	V	SARF

3. ジョブの構成

3.1　SXのJCLとACOS 1000のJCLとの比較

SXとACOS 1000ではそのJCLは互いに全く異なるが、機能的に対応するものを**表2**に示す[12]。

表2　SXとACOS 1000のJCLの比較

機能	ACOS 1000	SX
コンパイラの起動	$ FRT77	$FORT77SX
リンク，実行	$ GO	$LINK, $STEP
データの指定（パーマネントファイル）	$ PRMFL	$ASSIGN, $DEFINE
（テンポラリファイル）	$ FILE	$ASSIGN, $ALLOCATE, $DEFINE
共通　$ JOB文，$ENDJOB文　その他 $ SX文		

その他、下記のような主な差がある。

- SX固有のJCL[12,14,15]は桁にとらわれない。一つのJCLの最後は必ずセミコロンで終わる。
- $FORT77SX文ではコンパイラが入出力するためのファイル（ソースプログラム、コンパイル・ユニット）の指定も行う[10]。ACOS 1000ではこれらのファイルは、$ PRMFL文などで指定していた。
- SXでは$JOB文以外はすべてマニュアル[12,14,15]に記載されているJCLである。ACOS1000では$GO文がカタログであってマニュアルには記載がない。
- 装置番号の5、6、7は特別な内部ファイル名となる（**表1**参照）。ACOS 1000では差はなくJCLではファイル・コードとして指定できた。

3.2 ジョブ構成の基本

ジョブを構成するための JCL の順序は、SX とACOS 1000 とで基本的な差はない。本稿では、JCL を登録したファイルをACOS 1000 側にもつことにする。最も基本的なジョブ、すなわちファイル中のソースプログラムから実行するジョブの構成を**図1**に示す。ただし、このままではリンク時の不要なリストが出力されること、コンパイル時またはリンク時にエラーがあってもそのまま実行にはいること、などの不備があるので、実際にジョブを投入する場合には**6**節のように JCL を補っていただきたい。

```
   1カラム      8カラム      16カラム
①  $            JOB         A12345; A$pass-word, U
②  $            SX          TYPE=SX
③  $FORT77SX  INLIB=(A12345. ALIB), MEMBER=(X);
④  $LINK  FTMAIN;
⑤  $STEP  FTMAIN;
⑥              データ関係の JCL
⑦  $ENDSTEP;
⑧              データの記述
⑨  $            ENDJOB
```

図1 基本形ジョブの構成

以下は図の説明である。

① ジョブカードである。ジョブクラス[16]にはUまたはVを指定する。利用者番号は A12345、支払いコードはA（校費）、パスワードはpass-word と表記している。

② TYPE＝SXは基本形であることを宣言する。これ以外のパラメータも必要であるが、簡易形に同じであるから省略する[4,16]。

③ コンパイラを起動する。

④ リンカを起動する。FTMAIN はコード・モジュール名である。主プログラム名と同一にしておくこと。

⑤ プログラムの実行を開始させる。FTMAIN はロード・モジュール名である。

⑦ ⑤と組で使用する。

⑧ データを記述する。これを SYSIN に指定したデータという。

⑨ ジョブの終了を示す。

以下では**図1**の JCL を基本にして述べるので、置き換えるべき JCL のみを番号で示す。

登録済みのコンパイル・ユニットのみの場合には③が、ロード・モジュールを登録済みの場合に

は③および④が不要である。説明の都合上、ライブラリを使用しないものとしているので、使用する場合は6節を参照していただきたい。

3.3 コンパイル・ユニットとロード・モジュール

コンパイラおよびリンカは、それぞれコンパイル・ユニットおよびロード・モジュールを生成する。これらをファイルに保存しない場合には、システムは自動的にテンポラリファイルに登録し、ジョブの終了後、消去する。保存する場合には、登録先として統合ライブラリを用い、メンバー名は後述のような規則で命名される。なお、一つの統合ライブラリ内ではメンバー名は型ごとに一意であるから、登録の際には既にその型の同一の名前のメンバーがあると既存のものが置き換えられる。

(1) コンパイル・ユニットとその名前

コンパイラは、プログラム単位ごとに一つのコンパイル・ユニットを作成し、これを一つのメンバーとして登録する。メンバー名にはコンパイル・ユニットの名前[10]すなわちそのプログラム単位の名前を自動的に採用する。たとえば、PROGRAM SUN の指定されている主プログラムでは SUN, SUBROUTINE MOON(A,B)では MOON がそれぞれの名前となる。主プログラムと初期値設定副プログラムでは既定値をもち、それぞれ、FTMAINとFTBLKDである。

(2) ロード・モジュールとその名前

リンカは一つ以上のコンパイル・ユニットを結合して実行可能プログラムを作成する。これをロード・モジュールといい、ロード・モジュール名がメンバー名として登録される。この名前は自動的に命名されないので、ファイルに保存する場合はもちろんのこと、そうでない場合も$LINKの先頭のパラメータで利用者がかならず指定しなければならない。名前は28文字以内の単純識別名であればよいが、主プログラム名を採用しておくのがよい。主プログラム名と異なる場合には、他にENTRYパラメータ[13]が必要となるためである。なお、この名前は$STEP文の先頭のパラメータでも指定しなければならない。

4. ソースプログラムの使用、コンパイル・ユニット、ロード・モジュールの登録と使用

4.1 ソースプログラムの使用

統合ライブラリに登録されているソースプログラムを使用する場合は、$FORT77SX文のパラメータ INLIB= に統合ライブラリ名を、MEMBER= にメンバー名を指定する。異なる統合ライブラリに登録されている場合には、一つの統合ライブラリに対して一つの$FORT77SXが必要となる。

(a) 統合ライブラリ名XLIBのメンバー名Xに登録されているとき

```
③   $FORT77SX  INLIB=(A12345.XLIB), MEMBER=(X);
```

(b) 統合ライブラリ名がXLIBのメンバー名XとYに登録されているとき

③　$FORT77SX INLIB=(A12345.XLIB), MEMBER=(X, Y);

括弧内に指定した順序でコンパイル・リストが出力される。指定の順序は任意である。

(c) 二つの統合ライブラリ内にそれぞれ一つのメンバーとして登録されているとき

その統合ライブラリ名をALIB, BLIB, メンバー名をX, Yとする。

③-1 $FORT77SX INLIB=(A12345.ALIB), MEMBER=(X);
③-2 $FORT77SX INLIB=(A12345.BLIB), MEMBER=(Y);

この順序は任意である。

4.2 コンパイル・ユニットの保存

コンパイル・ユニットを保存するには$FORT77SXのCULIB=のパラメータで統合ライブラリ名を指定すればよい。ソースプログラムの登録先とコンパイル・ユニットの登録先とは互いに無関係であるから異なるものでもよいが、通常は両者を統一的に管理する意味で同じものを使用するとよい。

(a) 一つの統合ライブラリに登録する。

4.1節の(a)の場合を示す。**4.1**節の(b)の場合でも、同様である。

③　$FORT77SX INLIB=(A12345.XLIB), MEMBER=(X), CULIB=(A12345.XLIB);

4.1節の(c)の場合は次のとおりである。

③-1 $FORT77SX INLIB=(A12345.ALIB), MEMBER=(X), CULIB=(A12345.ZLIB);
③-2 $FORT77SX INLIB=(A12345.BLIB), MEMBER=(Y), CULIB=(A12345.ZLIB);

同一の統合ライブラリ名の指定があればその内容は自動的に追加される。

(b) 複数の統合ライブラリに登録する(**4.1**節の(b), (c)などの場合)

このときは$FORT77SX文が、少なくとも登録用の統合ライブラリの数だけ必要となる。**4.1**節の(b)の場合で、XとYのコンパイル・ユニットを異なる統合ライブラリXLIBとYLIBに登録したい場合には、次のようになる。(c)の場合も同様である。

③-1 $FORT77SX INLIB=(A12345.ALIB), MEMBER=(X), CULIB=(A12345.XLIB);
③-2 $FORT77SX INLIB=(A12345.ALIB), MEMBER=(Y), CULIB=(A12345.YLIB);

4.3 コンパイル・ユニットの使用

保存しているコンパイル・ユニットを使用するには、$LINKのパラメータのINLIB$i$=でそれが登録されている統合ライブラリ名だけを指定する。メンバー名は指定する必要がない。INLIBi=のiに1から9までを使用することで、統合ライブラリなどを9個まで指定できる。ASL/SXなどのライブラリの指定もこのパラメータでおこなう(**6**節参照)。リンカはiの値の小さい順序で検索し、コンパイル・ユニット名が見つかるとそれを結合する。複数の統合ライブラリにコンパ

イル・ユニットが登録されていて、かつ、同一の名前がある場合には、最初に見つかったものを結合するので指定順序に注意しなければならない。$LINKのパラメータについての詳細はマニュアル[13]を参照のこと。

(1) 実行可能プログラムの構成にソースプログラムを必要としない場合

(a) 一つの統合ライブラリに登録されている場合

```
④  $LINK FTMAIN, INLIB1=(A12345.ALIB);
```

(b) 複数の統合ライブラリに登録されている場合(4.2節の(b)の場合)

```
④  $LINK FTMAIN, INLIB1=(A12345.XLIB), INLIB2=(A12345.YLIB);
```

“INLIB1”は“INLIB”でもよいが、本稿ではINLIB1と書くことにする。

(2) 実行可能プログラムの構成にソースプログラムを必要とする場合

ソースプログラムとコンパイル・ユニットで実行可能プログラムを構成するときには、ソースプログラムの指定は4.1節のようにおこない、既存のコンパイル・ユニットの指定は上の(1)のようにする。ただし、ソースプログラムから入力するものはコンパイル・ユニットがテンポラリファイル(ファイル名はTEMP)に登録されているため、これをINLIBi= のパラメータで指定する必要がある。

```
③  $FORT77SX INLIB=(A12345.ALIB), MEMBER=(X);
④  $LINK FTMAIN, INLIB1=(TEMP), INLIB2=(A12345.ALIB);
```

4.4 ロード・モジュールの保存

ロード・モジュールをファイルに保存するには、$LINKのパラメータのOUTLIB=で登録先の統合ライブラリ名を指定すればよい。簡単のために、コンパイル・ユニットを保存せずソースプログラムから実行し登録する場合を考えれば、リンク時に必要なパラメータの指定は、ロード・モジュール名と登録先の統合ライブラリ名との2つでよい。

```
④  $LINK FTMAIN, OUTLIB=(A12345.ALIB);
```

コンパイル・ユニットが必要である場合には、この他に4.3節のようなINLIBi= のパラメータが必要である。

4.5 ロード・モジュールの使用

保存しているロード・モジュールを使用するには、$STEPにロード・モジュール名を、FILE=でそれが登録されている統合ライブラリ名を指定する。4.4節で作成したものを使用するには、次のようにする。

```
⑤  $STEP FTMAIN, FILE=(A12345.ALIB);
```

この際には③、④は不要であることは明らかであろう。

5. データの使用

5.1 パーマネントファイルの作成（PREALLOCコマンド）

この節では、PREALLOCコマンドで指定するパラメータの値の根拠となる事項について、効率を無視して使用の際にエラーが起きないという範囲で説明する。パーマネントファイルが一度作成されると、ファイルの属性を変更できないことは**2.1**節で述べた。ただし、この属性の種類は下記および**表3**に示す項目である。順編成で使用するか直編成で使用するかを決めれば、レコード形式および割り付け方は決まるが、レコード長およびブロック長は利用者のプログラムに依存して決まる場合がある。

(1) ファイル編成

ファイル編成が順編成か直編成かを決め、直編成ファイルの場合にはFILEORG＝DIRECTを指定する。順編成では既定値のため不要である。

(2) レコード形式、割り付け方

順編成ファイルの場合、RECFORM＝VB（可変長ブロック化レコード）、FIXTRK、NODELRを指定する[6,10]。VBではブロックの先頭に4バイトの制御語がとられる。割り付け方を指定しなくても作成時にはエラーとはならないが、Fortranの入出力ルーチンではこの値を前提としているので、データの登録時にエラーとなる。

直編成ファイルの場合、RECFORM＝FB（固定長ブロック化レコード）の指定だけでよく、割り付け方は既定値でよい[7,10]。FBではブロック制御語は使用しない。

レコード形式はVBおよびFB以外にもあるが、利用者の使い勝手がよいので、本稿ではこの二つに限定して述べる。

(3) レコード長

レコード長はRECSIZE＝rlでバイト単位で指定する。rlはプログラムで扱うFortranの記録（記録と略す）の長さから決まる。この記録とは、利用者にとってそのプログラムで見えている論理的なものであって、その長さはREAD文またはWRITE文の入出力並びと、書式付きである場合にはさらに用いる書式（FORMAT文）とによって決まる。

書式付きでは一つの記録は一つのレコードに格納されなければならず[10]、レコード長が不足するとデータの登録時にエラーとなる。このため、利用者は記録の最大長（flと書く）からrlを決めなければならない。SARFでは、この場合はレコード制御語を必要としないから、$rl \geqq fl$とすればよい。SSFのパーマネントファイルは本稿では考慮しない。

書式なしでは一つの記録が一つのレコードに入らないときには、Fortranの入出力ルーチンが自動的に複数のレコードに分割して格納する[10]ため、利用者は記録の長さをとくに意識する必要はない。必要に応じてレコード制御語がとられる。表には内部ファイル名の既定値を示しておいた。

レコード長の指定は省略でき、そのときの既定値については(4)で述べる。

(4) ブロック長

ブロック長 bl はBLOCKSZ＝bl でバイト単位で指定する。bl はレコード長 rl から決まり、VBでは $bl \geqq rl+4$ であればよい。FB では bl は rl の整数倍であればよい。

BLOCKSZ＝bl を指定して、かつ、RECSIZE＝rl を省略した場合は、VBではBLOCKSZ＝$bl+4$、RECSIZE＝bl を、FBではBLOCKSZ＝bl、RECSIZE＝bl を指定したものとしてファイルラベルを作成する。

表3 PREALLOC で指定するべき値

	編成 FILEORG	ブロック長 BLOCKSZ	レコード長 RECSIZE	レコード形式 RECFORM	割り付け方
順編成書式付き	SEQ	$bl \geqq rl+4$	$rl-\alpha \geqq fl$	VB	FIXTRK, NODELR
順編成書式なし	SEQ	512	508	VB	FIXTRK, NODELR
直編成書式付き	DIRECT	$bl=n \times rl$	$rl \geqq fl$	FB	NOFIXTRK, DELR
直編成書式なし	DIRECT	512	508	FB	NOFIXTRK, DELR
PREALLOC の既定値	SEQ	なし	本文参照	FB	NOFIXTRK, DELR

備考 1. 長さの単位はバイト。

備考 2. α=8(SSFの場合)、α=0(SARFの場合)。

5.2 入出力の制御情報

入出力が実行される時点では、内部ファイル名と外部ファイル名との対応付けがなされていることの他に、ファイルの属性などの制御情報が確定していなければならない。パーマネントファイルの場合には、ファイルラベル(**2.2** 節参照)と内部ファイル名の値との2つから定まるが、そのとき、**5.1** 節で述べた項目についてはファイルラベルが優先する[10,11]。残るデータ形式だけが内部ファイル名の値のみで決まる。内部ファイル名の値は既定値をもち(**表1** 参照)、変更するためには $DEFINE文を使用する。テンポラリファイルの場合にはファイルラベルが存在しないため、内部ファイル名の値で決まる。

直編成ファイルでは事前の接続を変更しない場合でも、OPEN文を実行しなければならない[3,10]。

SYSINファイルおよびSYSOUTファイルでは、**表1** のSINおよびSPRの属性と同一の値が書かれたファイルラベルがあるものとみなせばよい。

5.3 パーマネントファイルの使用

パーマネントファイルはファイルラベルをもち、これが内部ファイル名の値よりも優先して入出力の制御情報が決定されることは既に述べた(**5.2** 節参照)。したがって、ファイルの作成を正しくおこなっておけば、ファイルの編成および書式の有無にかかわらず、用いるJCLは $ASSIGN文だけでよく、外部ファイル名と内部ファイル名を指定するだけでよい。既定値ではパーマネント

ファイルを接続するものとしている。

以下ではファイルは順編成書式付きの場合を示しているが、そうでない場合も同様である。なお、改めて断るまでもないが、ファイルとの操作が入力および出力の双方を行う場合でも、$ASSIGN文は一つでよい。

(a) 外部ファイル名がDATA1のファイルをプログラムでREAD(5,10)で入力する場合

```
⑥  $ASSIGN SIN, A12345.DATA1;
```

内部ファイル名はSINであり FF05 ではないことに注意すること。

(b) 同一のファイルをREAD(9, 10)で入力する場合

```
⑥  $ASSIGN FF09, A12345.DATA1;
```

内部ファイル名はFF09である。

パラメータにSHARE=READを指定しておくと、このファイルを同時に異なるジョブで入力だけができる。ただし、そのジョブでもこのパラメータの指定が必要であることはいうまでもない。

```
⑥  $ASSIGN FF09, A12345.DATA1, SHARE=READ;
```

5.4 テンポラリファイル(順編成書式なしファイル)の使用

テンポラリファイルの使用法は順編成書式なしファイルの場合に限って述べ、そうでない場合については省略する。テンポラリファイルはプログラムの実行時にのみ存在するファイルであり、当然のことながらこれを前もって PREALLOC コマンドで確保しておけない。外部ファイル名としては、一意なファイル名を与える。接続には $ASSIGN文を、内部ファイル名の既定値(**表1**参照)の変更には $DEFINE文を、領域の確保には $ALLOCATE文を用いる。

ファイルの属性は、レコード形式をVBとすること以外は内部ファイル名の既定値でよい。レコード長およびブロック長については、パーマネントファイルの項で述べたことがテンポラリファイルにも同様に適用できるからである。割り付け方は領域を確保するとき自動的にFIXTRK, NODELR にされる。

WRITE(9)およびREAD(9)で5シリンダ使用するとすれば、次のようになる。外部ファイル名として、TEMP09 を使用する。以下の JCL の順序は任意である。

```
⑥-1 $ASSIGN FF09, TEMP09, FILESTAT=TEMP, PUBLIC;
⑥-2 $DEFINE FF09, RECFORM=VB;
⑥-3 $ALLOCATE FF09, SIZE=5;
```

テンポラリファイルであることを示すために$ASSIGN文にFILESTAT=TEMPの指定が必要である。ASSIGN文の既定値では1シリンダの領域が確保されるので、これで十分な場合は $ALLOCATE文は不要である。

5.5 SYSIN ファイルからの入力

SYSINファイルを使用するときには、$ASSIGN文で FILESTAT＝SYSIN を指定しなければならない。入力に用いる装置番号は、標準では5である。

(1) 装置番号5を使用するとき

データは$INPUT文と$ENDINPUT文で囲み、$ENDSTEP文の次におく。$INPUT 文には外部ファイル名に相当するデータ記述域名を付け、これを$ASSIGN文に指定する。

例えば、READ(5, 10)で2行のデータを入力する場合は次のとおりである。入力記述域名[10]をABXYZとする。

```
⑥    $ASSIGN SIN, ABXYZ, FILESTAT=SYSIN;
⑧-1  $INPUT ABXYZ;
⑧-2  5, 10, 15, 20 …… データ
⑧-3  1, -2, -3 …… データ
⑧-4  $ENDINPUT;
```

(2) 装置番号5以外を使用するとき

(1)と同じデータを例えばREAD(9,10)で SYSINから入力する場合は、内部ファイル名が異なるだけである。

```
⑥    $ASSIGN FF09, ABXYZ, FILESTAT=SYSIN;
```

⑧－1から⑧－4までは同じであるため、省略する。

5.6 SYSOUT ファイルへの出力

SYSOUT ファイルへの出力に用いる装置番号は標準では6である。6以外を非標準ということにする。標準では JCL は一切不要である。非標準で SYSOUT ファイルに出力する場合の注意事項は次のとおりである。

- $ASSIGN文にFILESTAT＝SYSOUTを指定する。
- ブロック長、レコード長およびレコード形式は変更できない。
- 非標準では外部ファイル名に相当する名前 *PR（*はS以外のAからZまでの任意の英字1文字）を用い、$DEFINE文で DATAFORM＝SSFを指定する。

例えば WRITE(20,10)でプリンタに出力する場合は次のとおりである。

```
⑥-1  $ASSIGN FF20, DPR, FILESTAT=SYSOUT;
⑥-2  $DEFINE FF20, DATAFORM=SSF;
```

6. 補足と注意事項

以上で説明したことの補足と注意事項を述べる。重複している項目もあるがまとめの意味で記述

した。

- 一つの JCL を1行に書ききれないときは、その行をあるパラメータの指定の終了したところで終わり、次の行を任意の桁から開始する。継続用の特別な文字はない。
- SYSIN ファイルに行番号があってはならないので[4]、CARDINサブシステムでジョブを投入するときには、予め行番号を除去しておくか、行番号を除去する応答をしなければならない。SYSIN ファイルの有効範囲は第1桁から第80桁までである。
- 初期値設定副プログラムを使用している場合は、その名前を EXTERNAL 文で宣言する必要がある[10]。
- $LINK の INLIB$i$（$1 \leqq i \leqq 9$）のパラメータが複数ある場合には、それらの指定の順序は i の値に依存せず、i が連続する必要はない。検索の順序は i の小さい順におこなわれ、異なる統合ライブラリ内に同一のコンパイル・ユニット名があると最初に見つかったものを結合する。
- リンカの出力するリストが不要の場合には、次のようにすれば出力を抑制できる。

```
④   $LINK FTMAIN,COMMAND='LIST=N,;';
```

- ライブラリを使用する場合には、次のようにする。SXで使用できるライブラリにはASL/SXとMATHLIB/SXがあるが、いずれも$LINKの INLIB$i$ = のパラメータでライブラリの登録されているファイル記述を指定する。

```
ASL/SX          INLIBi=(LIB.ASLAP)
MATHLIB/SX      INLIBj=(LIB.MLIBAP)
```

- 致命的なエラーが検出されたとき、そのジョブ処理の段階で終了させるための JCL は次のとおりである。

```
③の直後   $WHEN STATUS,>=,SERIOUS,JUMP,ENDJOB;
④の直後   $WHEN STATUS,>=,FATAL,JUMP,ENDJOB;
```

- 簡易形では、$JOB文のジョブクラスがUまたはVの場合 $FORT77SX文を生成し、SまたはTの場合 $FORTRAN77 文を生成する。基本形ではジョブクラスはこの働きをしない。

7. おわりに

本稿で述べた内容は、$FORTRAN77 文でコンパイラの指定をCP(Control Processor)用に変更すれば[10]、CP で実行するジョブにも適用できる。しかし、SX の演算性能の高速性は AP で発揮されるのであって、CP はあくまでも AP を支援するためのものであるから[1,2,5]、バッチ処理では極力 AP を使用するべきである。なお、ジョブの結果を早く入手するために、小さい方のUクラスで十分な場合は、これを指定するのがよい。Uクラスではサービス時間が最長で15分である

待ち行列に並ぶことになるが、一方、大きい方のVクラスではそれが2時間の待ち行列に並ぶことになるからである。

本稿がAPジョブを使用する上で役立てば幸いである。

参　考　文　献

1) 渡辺，近藤，端山，大中，藤井：スーパーコンピュータ SX－1の概要(1)，大阪大学大型計算機センターニュース，Vol.15, No.4 (1986).

2) 藤井：スーパーコンピュータ SX－1の概要(2)，大阪大学大型計算機センターニュース，Vol.15, No.4 (1986).

3) 大中，後藤：SX FORTRAN77 概要(1)，大阪大学大型計算機センターニュース，Vol.16, No.1 (1986).

4) 後藤，大中：SX FORTRAN77 概要(2)，大阪大学大型計算機センターニュース，Vol.16, No.1 (1986).

5) 片山，河原，大中：FORTRAN77/SXにおける高速化技法，大阪大学大型計算機センターニュース，Vol.16, No.1 (1986).

6) 馬野：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1），大阪大学大型計算機センターニュース，Vol.16, No.1 (1986).

7) 馬野：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ニュース，Vol.16, No.2 (1986).

8) 多喜：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).

9) GGB11－1 FORTRAN77言語説明書，日本電気(1985).

10) GGB12－2 FORTRAN77, 77/SX プログラミング手引書，日本電気(1986).

11) GDA12－2 SXCP 解説書，日本電気(1986).

12) GJF11－1 MSF－6利用説明書，日本電気(1986).

13) GGK16－1 プログラム管理サービスプログラム説明書，日本電気(1986).

14) GCB12－1 ジョブ制御言語ハンドブック〈上巻〉，日本電気(1986).

15) GCB13－1 ジョブ制御言語ハンドブック〈下巻〉，日本電気(1986).

16) 大阪大学大型計算機センター速報，No.134 (1986).