UDC 681.326.3.07

# FACOM 270-30 ジョブスタックシステムについて

On the Job Stack System for FACOM 270-30

柏本昌美*・渡辺　勝**
Masami KASHIMOTO and Masaru WATANABE

**一連のジョブを連続的に処理する場合，それらのジョブを大容量記憶装置へ格納し，計算機システムの管理の下に処理する方式が考えられた．ここではこれらのジョブを磁気テープに格納し処理する方法を記述する．また実際に FACOM 270-30 モニタシステムにおいてどのように実現したかその方法を示す．**

## 1. はじめに

FACOM 270-30 において各ジョブのプログラムを連続的に処理する方法として BATCH 処理がある．これはカードリーダ（または紙テープリーダ）より制御カードを読みその情報に従ってプログラムを起動する．しかしながらいくつかのまたは一連のジョブを一括して処理する場合，それらのジョブをカードリーダより入力する時オペレータはジョブの開始，終了時期に関連して常にカードリーダに対し注意を払わねばならない．そこでこのようなまとまったジョブを大容量記憶装置へ格納し，計算機の管理の下にそれらのジョブを自動的に処理するシステムが考えられて来た．大容量記憶装置の代表的なものに磁気ディスク装置があるが現在のシステムには備っていない為，各ジョブをシーケンシャルに処理することを前提として磁気テープへそれらのジョブをスタックするシステムを作成したので以下に報告する．

本システムはカードリーダ上にセットされている情報(制御カード，ソースプログラム，データカード等)を磁気テープへジョブ単位でスタックする「STACK プログラム」とそれらのスタックされたジョブを制御テーブル(後述) に従って自動的に処理する「START プログラム」より成っている．またこのシステムは FACOM 270-30 モニタ $III_3$ $E_4$ を基に作成したものであり，現時点では FORTRAN を使用したジョブを対象に処理を行なう．

## 2. JOB STACK システム

### (1) 磁気テープの格納形式

モニタ $III_3$ において磁気テープへの情報の格納形式には以下の2通りの方法がある．

(a) STANDARD 形式

(b) OMITTED 形式

(a) の形式は1つのファイルに対しファイル名をつけて情報を格納するのに対し，(b) の形式ではテープマーク (TM) を情報の後に書くことによりそこまでの情報を1つのファイルとして扱っている．(すなわち，TM から TM までを1つのファイルとみる．)

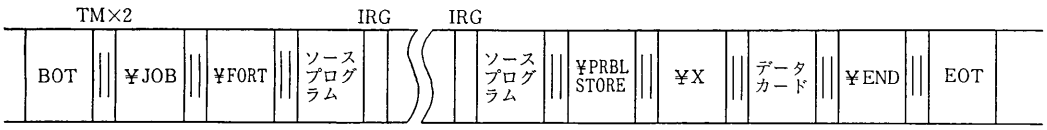
本システムで，FORTRAN のプログラムが磁気テープに格納されているデータを READ する時は (b) の形式で格納されていなければならないこと，および磁気テープにスタックされたジョブを管理する制御テーブルをあまり大きくとれない（ドラムに空の領域があまりない為）等の事情により (b) の形式で磁気テープに格納した．

各ジョブについては図1に示すような形式で磁気テープに格納する．(¥ JOB のファイルから ¥ END のファイルまでをジョブの単位としている．)制御カードは1文字/語の形式で1つのファイルに格納され，FORTRAN のソースプログラムおよびデータカードは2文字/語にパックしブロッキングして磁気テープに格納する．また ¥JOB のカードが格納されているファイルにはスタック番号を一緒に格納し，後でそのジョブを処理する時制御テーブルに格納されているスタック番号と照合する．

### (2) 制御テーブルの構成

制御テーブルは図2に示されているように各ジョブの情報，および STACK プログラムや START プログラムで使用するパラメータ等が格納されており，ドラムの固定領域に2レコード (256 語) の大きさで登録されている．制御テーブルの相対0番地から6番地までがこれらのパラメタ等に使用され，残り 249 語に 83 ジョブ分 (1ジョブ当り3語使用する．)の情報が格納されている．各ジョブについての情報は磁気テープへスタックされるジョブのシーケンス番号であるスタック番号(STKNO)，ジョブ内で使用するファイルの数 (FNO) および，BOT からのファイルの数 (STKFC) の3語より構成されている．また，スタック番号を格納する WORD の上2ビットはそのジョブの優先順位を表わし，通常スタックする時それらのビットは '00' としてスタックされる．優先

* 東京大学生産技術研究所　電子計算機室
** 東京大学生産技術研究所　第3部




cf. BOT…Begin of tape　　TM…Tape mark
EOT…End of tape　　IRG…Inter-record gap

図1　磁気テープの格納形式

順位を表わすビットは以下のことを意味する.

00……シーケンス順に実行する(スタックされた順序)

01……優先順位＝+1 (00, 11 のジョブより優先して実行する)

10……終了したジョブ

11……優先順位＝−1 (00, 01 のジョブがすべて終了後実行する)

START プログラムでは制御テーブル内の各ジョブの上記ビットを参照することによって次に行なうべきジョブを決定する. またスタック番号とそのジョブの対応をとるために STACK プログラムによってスタック番号と JOB カードのイメージをファコムライタへ印字するようにしてある。

## 3. STACK プログラム

STACK プログラムは図1に示すような形式でカードリーダにセットされている情報を磁気テープへ格納すると共に, 各ジョブの情報 (STKNO, FNO, STKFC 等) を制御テーブルへ格納する。STACK プログラムは以下の制御命令をファコムライタからタイプインすることにより起動される.

¥ STACK␣MT $n$ { −INITIAL (CR) / ␣READ (CR) / (CR) }

ただし (CR)……CARRIAGE RETURN KEY

$n$……磁気テープの機番

新しくジョブをスタックする場合は, パラメタとして INITIAL を指定する. このパラメタを指定すると制御テーブルをクリアすると共に, 以前に磁気テープにスタックされているジョブを削除し, 磁気テープの最初(BOT) からこれからスタックするジョブを格納する.

以前に磁気テープにスタックされたジョブに続けてスタックする時にはパラメタとして READ, または(CR) のみを指定する. パラメタ READ の指定は, ドラムに格納されているはずの制御テーブルがこわれている時に使用し, 紙テープより制御テーブルを READ する. (CR) のみの時は, 制御テーブルはドラムから読み取られる.

STACK プログラムが起動されると, カードリーダより制御カードやソースプログラムカード等の情報を磁気テープへスタックする. ¥ FIN のカードを読むと制御テーブルをドラムへ格納し, 磁気テープを巻きもどして STACK プログラムを終りにする. 以上 STACK プログラムのフローチャートを図3に示す.

| | | |
|---|---|---|
| MTNO | スタックジョブの磁気テープの機番 | STACK, START プログラムで使用するパラメタおよびスイッチ |
| FCOUNT | 磁気テープのヘッドの位置を示すカウンタ | |
| ESTKNO | End stack number | |
| ESTKFC | End stack file counter | |
| EXESNO | Execution end stack number | |
| ENDSW | Execution end switch | |
| JOBSW | Job switch | |
| PRIO STKNO.1 | PRIO;優先順位 STKNO;スタック番号 | 各ジョブについての情報 |
| FNO | ¥JOBから¥ENDまでのファイル数 | |
| STKFC | 磁気テープのはじめからのファイル数 | |
| PRIO STKNO.2 | | |
| FNO | | |
| STKFC | | |
| ⋮ | | |
| PRIO STKNO.83 | | |
| FNO | | |
| STKFC | | |

図2　制御テーブルの構成

## 4. START プログラム

START プログラムは磁気テープへスタックされたジョブを制御テーブルに従って自動的に処理するプログラムであり, 次の点を考慮して作成されなければならない.

(1) FORTRAN コンパイラの入力パラメタの解釈

(2) 入力データ (カードリーダからの) の処理

(1) 通常のバッチ処理では, 制御カード ¥ FORT␣CD をソースプログラムの前に入れて, ソースプログラムがカードリーダより入力されることを指定する. しかしジョブスタック方式では, ソースプログラムは磁気テープに格納ずみであり, カードリーダではなく磁気テープから入力するように変更しなければならない.

(2) FORTRAN のソースプログラムの中でカードリーダよりデータを READ する場合 (READ 文で機番

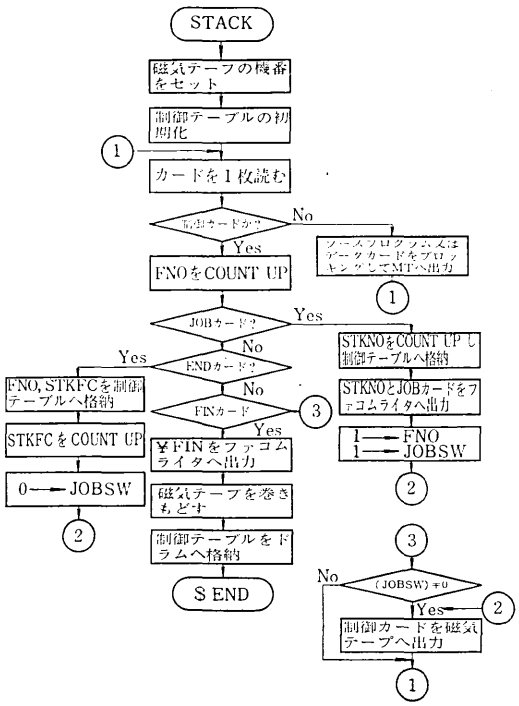



図3　スタックプログラムのフローチャート

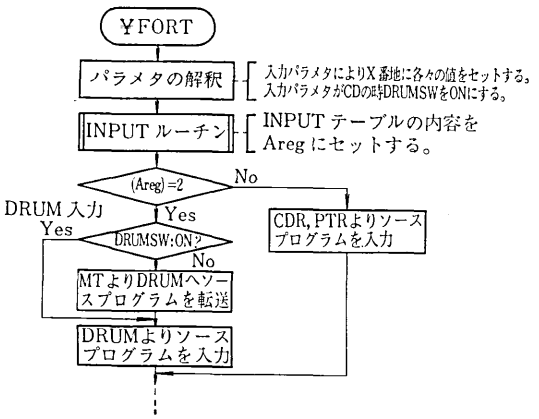



図4　FORTRAN コンパイラのフローチャートの一部

5又は 40 を指定した時）も，同様にカードリーダにかえて，磁気テープ上にスタックされているデータを読むような変更が必要である．それらの方法をつぎにのべる．

**（1） FORTRAN コンパイラの入力パラメタの解釈**

FORTRAN のコンパイラのフローチャートの一部を図4に示す．ここでコンパイラは制御カードの入力パラメタを解釈し，ある番地（仮に X 番地とする．）に次のような値をセットする．

紙テープ入力のとき……0

カードリーダのとき……1

磁気テープ（またはドラム）のとき……2

しかし，START プログラムで処理する場合ソースプログラムはすでに磁気テープにスタックされているので X番地の内容を参照する時は2がロードされねばならない．そこで本システムでは INPUT テーブルおよびそのテーブルを参照する INPUT ルーチンを磁心固定領域にセットしておき，X番地の内容を参照する時は INPUT ルーチンを CALL するようにコンパイラを変更する．そして START プログラムの実行中はそのテーブルの CDR に相当する番地の内容を2に修正する．たとえば，START プログラム実行中に ¥ FORT␣CD のファイルが磁気テープより READ されると FORTRAN コンパイラでは入力パラメタの CD を解釈し X 番地に1をセットする．そしてその後，ソースプログラムを入力するために X 番地の内容を参照する時には INPUT ルーチンが CALL され INPUT テーブル中の CDR に相当する番地が2になっているので磁気テープまたはドラムよりソースプログラムを入力する．（図5参照）

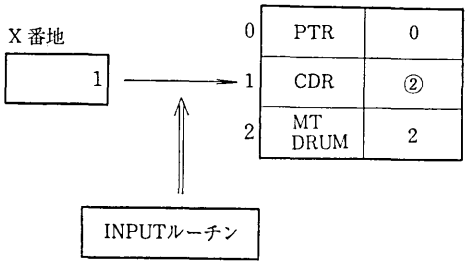



図5　INPUT テーブル

ところが磁気テープからソースプログラムを入力するには STANDARD 形式で格納されていなければならない．そこで START プログラム中では FORTRAN のソースプログラムはコンパイラを起動する前に磁気テープからドラムに転送しドラム入力として処理する．そしてコンパイラは入力パラメタが CD で INPUT テーブルの内容が2になっている時は，ドラム入力として処理し，入力パラメタが磁気テープで INPUT テーブルの内容が2のときは磁気テープ入力として処理するようにした．

INPUT テーブルの内容は START プログラムで，CDR の内容を修正する以外は標準値として PTR のときは0，CDR は1，MT と DRUM は2がセットされているので START プログラム以外 (BATCH 中) で FORTRAN コンパイラを使用しても修正前と何ら変らない．

**（2） 入力データカードの処理**

（a） 入力装置の変更

FORTRAN の READ ステートメントでカードリーダよりデータを Read する場合は機番として5，または 40 を指定する．この場合データカードはすでに磁気テープにスタックされているので磁気テープよりデータを Read するように変更しなければならない．FORTRAN プログラムでは入出力文 (READ 文，WRITE 文) に

対して図6のような形で実際の入出力装置にアクセスする．FORTRAN コンパイラでは(READ または(WRITE のサブルーチンを CALL するオブジェクトコードを出す．(DIAL サブルーチンは図6のようにフィジカルユニットアサインテーブル (PUAT) を参照してそれぞれの IOCS を呼び出す．PUAT は図7に示すような物理機番と論理機番の対応表である．READ 文で機番5を指定すると図7の CDR の番地の内容に対応する IOCS を CALL する．(この場合は5なのでカードリーダの IOCS を CALL する．) 従って START プログラム中において図7の CDR の番地を2に変更すれば磁気テープの IOCS を CALL するようになる．

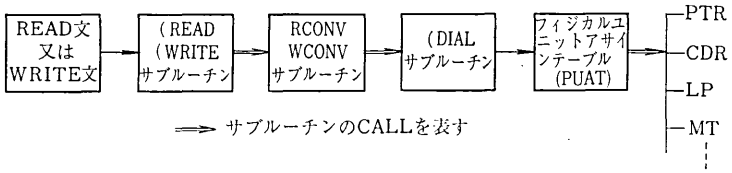


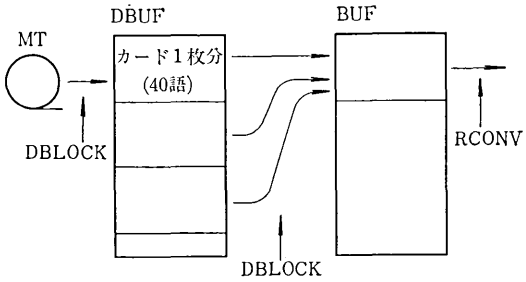

図6　READ, WRITE に対する I/O 装置へのアクセス方法

| | 物理機器 | 論理機番 |
|---|---|---|
| 1 | ―――― | 1 |
| 2 | 磁気テープ(MT) | 2 |
| 3 | キーボード(KB) | 3 |
| 4 | 紙テープリーダ(PTR) | 4 |
| 5 | カードリーダ(CDR) | 5 |
| 6 | ラインプリンタ(LP) | 6 |
| 7 | タイプライタ(TYP) | 7 |
| 8 | 紙テープパンチ(PTP) | 8 |
| 9 | カードパンチ(CDP) | 9 |
| 10 | XY プロッタ(PLT) | 10 |

図7　PUTA の構成

(b)　デブロックルーチン

磁気テープに格納されているデータはブロック単位で読み書きされる．これに対し，カードリーダのデータは1枚分 (80 文字) 単位で Read される．データカードを磁気テープへスタックする時はブロッキングしているので磁気テープへスタックされたデータを Read するにはデブロックしなければならない．すなわち1ブロックには通常約3枚分のデータカードが格納されているがそれらを1枚分に分解してバッファに格納しなければならない．そこで本システムにおいては RCONV サブルーチン中 (図6参照) にデブロックルーチン (DBLOCK) を追加し，図7の CDR の内容が2に変更された時は磁気テープより Read したデータはデブロックして処理するように修正した．図8に示されているようにデブロックルーチンは磁気テープより1ブロック Read したデータを一度 DBUF に格納し，そこでカード1枚分の大きさに分解して別のバッファ (BUF) へ格納する．




DBUF デブロックのために必要なバッファ(128語)
BUF　RCONV で使用するバッファ(128語)

図8　デブロッキング

START プログラムは以下の制御命令をファコムライタよりタイプインすることによって起動される．

¥ START␣MT $n$ {−INITIAL (CR) / (CR)}

$n$ : 磁気テープの機番

(CR) : CARRIAGE RETURN KEY

INITIAL を指定する時は磁気テープのヘッドがロードポイント上に位置している時に行なう．一方 (CR) を指定した時は磁気テープのヘッドが次のジョブの情報をアクセス出来る位置にある時に行なう．

START プログラムが起動されると制御テーブルをドラムより Read し INPUT テーブル，PUAT の変更等を行なう．次に制御テーブルをサーチし，次に実行するジョブを決定しそのジョブが格納されている位置まで磁気テープを移動しそこに格納されているジョブの処理を行なう．制御テーブルの中に実行すべきジョブが無くなった時，および ¥ FINISH を行なった時 START プログラムは終了し，JOB EXECUTION END をファコムライタへタイプする．図9に START プログラムのフローチャートを示す．

## 5.　その他のシステムプログラム

ジョブスタックシステムは STACK プログラムと，START プログラムから成ることは以前に述べた．ここではそれらのプログラムを援助するシステムプログラムについてそれらの機能を説明し，最後にそれらの制御命令をまとめて記述する．これらの制御命令をファコムライタよりタイプインすることにより各々のシステムプログラムを起動することが出来る．

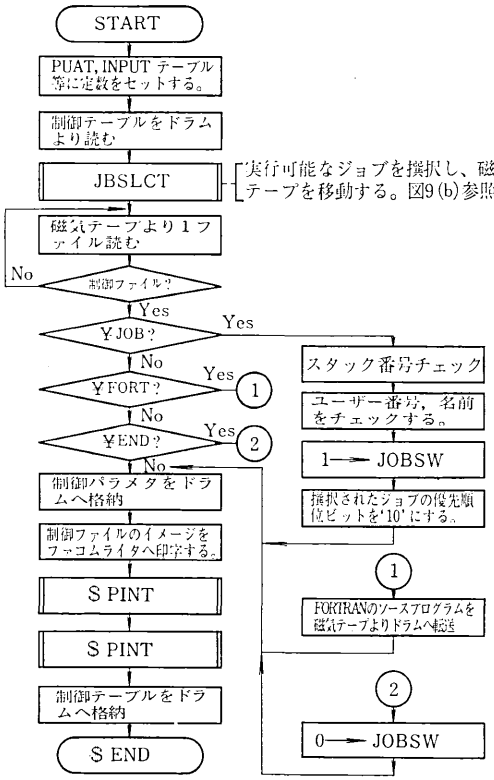



図9（a）START プログラムのフローチャート

**（1） FINISH**

START プログラムは制御テーブルに格納されているジョブがすべて終了した時に終る．しかし未だ制御テーブルに未処理のジョブがある時点で START プログラムを終了させたい場合にこのプログラムを使用する．

このプログラムが起動されると制御テーブル内の ENDSW（相対5番地）を ON にし，現在実行しているジョブで START プログラムを終了させる．パラメタとしてスタック番号を指定した時はそのスタック番号を制御テーブル内の EXESNO（相対4番地）に格納し，そのスタック番号のジョブを実行後 START プログラムを終了させる．

**（2） ALTER**

制御テーブル内に格納されているジョブを削除したり他のジョブより優先して実行したい場合にこのプログラムを使用する．パラメタとして指定したスタック番号の優先順位ビットを変更する．

**（3） JSTOP**

現在実行中のジョブを終了させ，磁気テープを ¥END の格納されているファイルまで移動する．パラメタ X を指定した場合は次に ¥X のファイルがあればそれを実行する．

**（4） CTDUMP**

制御テーブルの情報すべてをラインプリンタへ出力する．パラメタ P を指定した時は紙テープパンチへもバイナリコードで出力する．制御テーブルはモニタをロードした場合にこわされるのでそのような場合にはこのプログラムにより紙テープに制御テーブルを出力し，モニタをロードした後で紙テープを次の CTREAD プログラムで読み再び START プログラムを続けることが出来る．

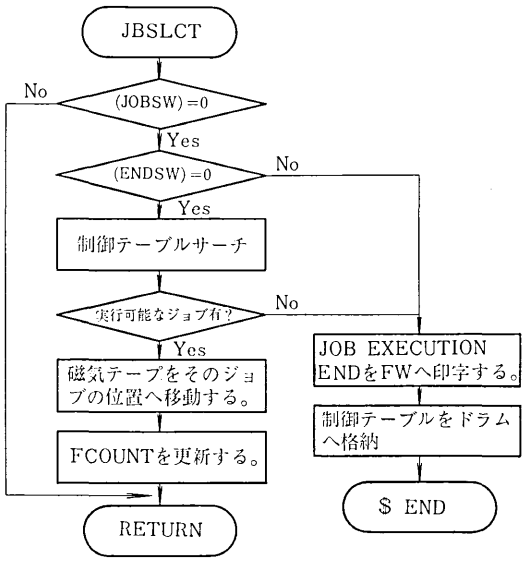



図9（b）JBSLCT サブルーチン

**（5） CTREAD**

CTDUMP プログラムで紙テープに出力された制御テーブルを読みドラムの固定領域に格納する．

**（6） CTBPRT**

START プログラム実行中に各ジョブの情報（優先順位，FNO, STKFC 等）を知りたい場合にこのプログラムを使用する．このプログラムを起動するとパラメタとして与えたスタック番号に対応する各ジョブの情報をファコムライタへ出力する．

**（7） SCSTNO**

START プログラムではジョブの実行を開始する時磁気テープのヘッドが ¥JOB のファイルをすぐ読める位置になければならない．しかし磁気テープが何等かの原因によりエラーを生じそのジョブが終了した場合には次のジョブを実行する場合に ¥JOB のファイルまで磁気テープを移動する必要がある．このプログラムは上記のような時に使用し，現在の位置から FORWARD へ ¥JOB のファイルまで磁気テープを移動し，そのジョブの STKFC を FCOUNT へ格納する．そしてファコムライタへスタック番号と STKFC の値を出力する．スタック番号をパラメタとして与えた場合はそのジョブが格納されているファイルまで磁気テープを移動する．

パラメタとして DUMP を指定した時はそのスタック番号のジョブのファイル全部（¥ JOB のファイルから ¥ END までのファイル）を 16 進4桁の形式でライン

プリンタへ出力する.

以下に上述のプログラムの制御命令を列記する.

¥ FINISH{␣×××××(CR) / (CR)}

×××××：スタック番号（5桁以内の数字）

(CR) CARRIAGE RETURN KEY

¥ ALTER␣×××××␣PRIO±{0 / 1 / 2}(CR)

×××××：スタック番号

PRIO␣0のとき，　シーケンス順に実行

＋1　〃　　優先順位＝＋1

－1　〃　　優先順位＝－1

－2　〃　　ジョブの削除

¥ JSTOP{␣X(CR) / (CR)}

¥ CTDUMP␣××××××{␣P(CR) / (CR)}

××××××：参照番号（6桁以内の数字）

¥ CTREAD (CR)

¥ CTBPRT␣×××××(CR)

×××××：スタック番号

¥ SCSTNO{␣×××××␣〔DUMP〕(CR) / (CR)}

×××××：スタック番号

## 6. おわりに

以上，FACOM 270-30においてカードリーダより入力されるデータを磁気テープにスタックし，それらを処理するシステムについて述べて来た．現在，計算機室においてはクローズドにより1日 30 前後のジョブを処理している．これらのジョブを磁気テープにスタックした場合約 20 分前後磁気テープへの転送に時間を要する．しかし一度スタックされた後は計算機の管理の下に自動的に処理されるのでオペレータの負担が減少する．

今後，上述の転送時間の減少およびFORTRAN以外のジョブたとえば TRANSFER 等についても，同様なスタック処理ができるように改善してゆきたい．最後に日頃指導下さる藤田計算機室長および室員の方々に深く感謝致します．(1972年6月23日受理)

### 参考文献

1) FACOM 270-30 MONITOR III, 仕様書
2) FACOM 270-30 MONITOR III, 操作仕様書
3) FACOM 270-30 FASP 仕様書
4) FACOM 270-30 FORTRAN 仕様書
5) FACOM 230-45 解説