FACOM 270–20 オペレーティングシステムの改造について

野崎剛一*・山口正道*
阪上直美*・山田英二**

The Improvement of the Operating System in the FACOM 270-20 System

by

Koichi NOZAKI
(Electronic Computer Center, Nagasaki University)

Masamichi YAMAGUCHI
(Electronic Computer Center, Nagasaki University)

Naomi SAKAUE
(Electronic Computer Center, Nagasaki University)

Eiji YAMADA
(Department of Electrical Engineering)

This is the report that presents a process of the improvement of the Operating System in the computer of FACOM 270-20.

When we used this computer on it's Batch System, We required some other function, so we analyzed this Operating System by machine language. And to add or to change this Operating System, we made some system programs in the assembly language (FASP) and some routines in a machine language.

Making a system program, we must use the assembly language and make an efficient program.

After analyzing this Operating System, we produced a new Operating System. On it's System,we can get some infomation of our program.

1. まえがき

電子計算機システムを運転する場合，そのシステムの機器構成，ハードウェアと共に，ソストウェア（オペレーティングシステム）によって，その使い易さが左右される．このオペレーティングシステムは，メーカーによってハードウェアと一緒にほとんど提供される．しかし，多岐にわたるサブシステムのどこまでを提供すべきかは，メーカーにとってもユーザにとっても大きな問題であり，メーカーはある程度の汎用性をもたせて，それらを作成し提供している．そのために，ユーザはある程度の改造を行ない独自のシステムを開発することもまれではない．

昭和53年5月13日受理
* 長崎大学電子計算機室
** 電気工学科

さて，FACOM 270-20 のオペレーティングシステムには，その一部である「モニター」と呼ばれる管理プログラムがある．このモニターは，その管理下に置くすべてのプログラムを管理し，ジョブの連続化を行ない，かつオペレータの介入を最小にしている．

ところが，この提供されているモニター及びシステムには，共同利用クローズド処理という立場からみて，様々な問題点をかかえていた．そこで，筆者らはモニター及びバッチ処理システムの解析を行ない，FORTRAN コンパイラの一部改造，システムプログラムの改良，改造，新規作成等を行ない，オペレーションの自動化，簡易省力化を図ってきた．

以下，本稿では，モニター及びシステム改造の概要について述べる．

2. FACOM 270-20 モニター（Edition 10）システムの問題点について

1) バッチ処理においてエラーメッセージ出力機能が充実していない．
2) プログラム格納アドレスが，システムタイプライタにしか出力されない．
3) R.B. (Relocatable Binary) 形式プログラムのマガジンファイル情報に無駄がある．
4) 実行プログラムのエラー発生回数を数えて自動的に打ち切る機能が無い．
5) タイマーが正確でない．
6) XY プロッタの仕分情報出力機能が無い．
7) その他

メーカー提供のモニターシステムには，上記の問題があり，これらは本システムの機器構成面からの制約もあるが，改良する余地は多分にあった．

3. 変更あるいは改造ならびに新規作成したシステムプログラムについて

1) FORT (130 records)：FORTRAN コンパイラ
2) MPRINT (13 records)：標準メッセージあるいは，ユーザ指定のメッセージをシステムタイプライタに印字するプログラム
3) END (4 records)：バッチ処理において，ジョブの終結処理を行なうプログラム．改造後 15 records
4) JOB (3 records)：1つのジョブの始まりを意味し，モニターの管理しているコントロールデータ（DCT 領域のデータ）の初期設定を行なうプログラム
5) KILL (1 record)：バッチ処理を中断することなく次のジョブに制御を移すプログラム
6) RBD (6 records)：ドラム上のリロケータブルバイナリ情報をマガジンファイル又は，紙テープに掃き出すプログラム
7) RBR (7 records)：リロケータブルバイナリ情報をドラムに読み込むプログラム．改造後 8 records
8) モニターの一部

上記プログラムのうち，提供されているプログラムには，R.B.（リロケータブルバイナリ）形式プログラムのないものがあり，またシステム改造にあたっては，これらのほか，BATCH (7 records)，PRBL (50 records)，モニター (25 records) 等の解析も必要となった．

（注） 1 record : 128 words

3-1 FORT の一部改造について

FORTRAN ソースプログラムのコンパイル時のエラーを，仕分情報と共に，ラインプリンタに出力させる機能を付加するために，このコンパイラの一部（エラーメッセージ処理ルーチン）を機械語により解析し変更を加えた．

FACOM 270-20 FORTRAN コンパイラは，システム中（内蔵ドラム）に A.B. (Absolute Binary) 形式で登録されていて，R.B. (Relocatable Binary) 形式のプログラムが提供されていない．R.B. 形式のプログラムが提供されている場合には，RBLIST というプログラムにより，アセンブリ言語のプログラムリストを作り出してそれを解析すればよい訳であるが，A.B. 形式のプログラムを解析する場合，直接，内部表現の16進又は8進数字等による解読をするのは，非常に困難で手間のかかることである．それ故，A.B. 形式（実行形式）の機械語 (Machine Language) で表わされたプログラムを解析するために，機械語のプログラムから，なるべく読みやすい，アセンブリ言語のプログラムリストを作り出すためのプログラム「逆アセンブラ」の作成が必要となった．

そこで，まずこの「逆アセンブラ」の作成を行ない，ドラム又は主記憶の指定番地より指定ワード数の領域のリストを作り出す機能を持たせた．そして，これにより FORTRAN コンパイラの解析を行なった．

ところで，このコンパイラはプログラム語数が約16キロワードで，メインプログラムと9個のサブプログラム（その内，6個はセグメントサブプログラム）から成っている膨大なプログラムであり，この全体的な解析は容易なことではない．そこで，この解析を始めた第一の目的は，コンパイル時のエラー情報を仕分情

報と共に，ラインプリンタに出力する機能を付加することであったので，コンパイル時のエラーメッセージ処理ルーチンを見つけ出すことを行なった．ここで対象としたエラーメッセージは，次の6個である．

F. EMERGENCY-ERROR INVALID CHARACTER (2B70)
F. EMERGENCY-ERROR DRUM TRANSFER ERROR (2B7D)
F. EMERGENCY-ERROR WORKING AREA OVER (2B89)
F. EMERGENCY-ERROR PROCESSOR TABLE OVER (2B94)
F. EMERGENCY-ERROR R.B. AREA OVER (2BA1)
F. EMERGENCY-ERROR ELDIC OVER (2BAB)

これらのエラー処理ルーチンは，FORTRAN コンパイラ（翻訳プログラム）の中のセグメントサブルーチンの2番目の中にあり，message print routine により，エラー検出時又は，発生時に，システムタイプライタにのみこれらのエラーメッセージを出力している．そこで，これらのエラーに関する情報を仕分情報と共にラインプリンタに出力する機能を付加するために，そのエラーメッセージテーブル先頭アドレスを，DCT (Drum Constant Table) の0番地に格納して，END プログラムに，この情報を伝える方法をとった．この DCT 定数領域は，内蔵ドラムの末尾の部分に設けられていて，一つのジョブの各ステップ間で情報の受け渡しをするために使用され，その0番地は現システムの機器構成上では使われていない．

ところで，FORTRAN コンパイラの変更を最小限にするために，エラーメッセージ先頭アドレスは，このコンパイラの実行（絶対）アドレス（エラーメッセージの後に示したカッコ内の数値）を使用している．そして，この情報を END プログラムで参照してエラーメッセージを出力する機能を付加した．

簡単な処理の流れを Fig. 1 に示す．

この FORTRAN コンパイラ解析中に，エラーメッセージテーブルの R.B. AREA OVER のメッセージワード数が #000B となっているが，これは #000F の間違いであることがわかった．

以上の機能を付加するために，36ワード分，内蔵ドラムに機械語で直接書込み追加変更を加えた．

3－2　MPRINT の一部改造について

このプログラムは，標準メッセージあるいは，ユーザ指定のメッセージをシステムタイプライタに印字するものである．

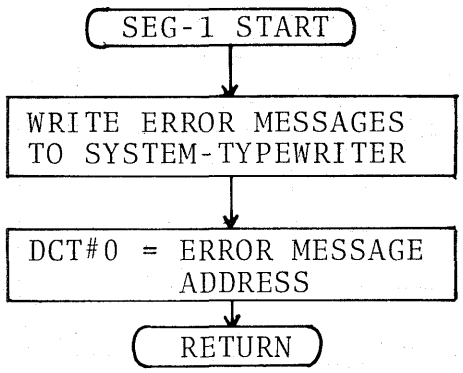

Fig. 1　Flowchart showing Segment-1 Subroutine of Fortran Compiler.

メーカー提供のモニター（Edition 10）システムには実行プログラムのエラー発生回数を数えて自動的に打ち切る機能が全く無かった．このために，バッチ処理中にエラーの数多く発生するプログラムがはいり込むと，運転効率が著しく低下し，オペレーターが全く目を離せない状況が生じる．

そこで，関数演算エラーや FLOWTING OVERFLOW などが発生しエラー発生回数がある回数を越えた場合に，実行プログラムを自動的に打ち切る機能を付加するために，次の様な改造を行なった．

関数演算エラーや FLOWTING OVERFLOW 等のエラーが発生した場合に，モニター領域 #90番地に1が格納されて（但し，FLOWTING UNDERFLOW の場合には格納されていない。），メッセージプリントプログラムが起動されている．従って，このメッセージプリントプログラムにエラー発生回数を数えさせる機能を付加するように改造すれば良い訳である．

そこで，モニター領域に1が格納されてメッセージプリントプログラムが起動された場合に，そのエラー発生回数を数えて，10回を越えたら次のジョブに制御を移す KILL プログラムに制御を移す様に改造した．

この機能を付加するために，既存のプログラムの標準メッセージの文字コード MESSAGE PRINT ERROR が格納されている領域にこれらの処理ルーチンを入れ込み，31ワード分，内蔵ドラムに機械語で直接書き込み変更を加えた．それ故，この標準メッセージの出力はできないが，一般のバッチ処理には全く支障はない．

この処理プログラムの簡単なフローチャートを Fig. 2 に示す．

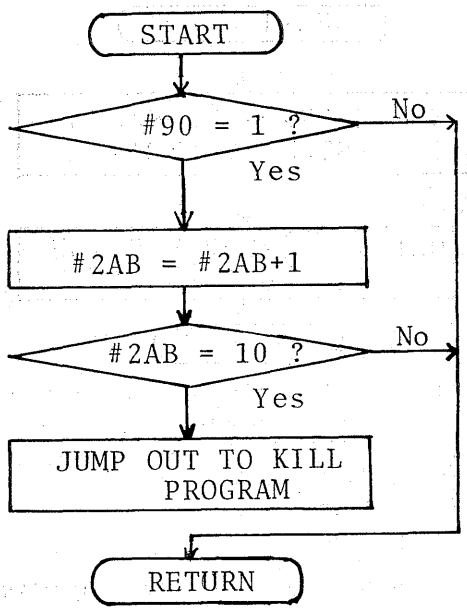

Fig. 2 Flowchart for MPRINT Program.

3-3 END の改造について

このプログラムは，一つのジョブの終結処理を行なうもので，計算処理時間，ラインプリンタ用紙出力枚数，課題番号，処理日時，プログラム格納アドレス及び，エラーメッセージ等を，システムスイッチ（モニター領域 #7F 番地）16ビットの各ビットのオン・オフにより指定された機器に出力する制御プログラムである．また，ラインプリンタのページコントロール及び，XY プロッタジョブ仕分情報自動出力機能も付加しプログラム全体をアセンブリ言語（FASP）により新規作成した．

メーカー提供のプログラムを改造した目的は，実行プログラム格納アドレス及びワード数やシステムプログラムで検出されたエラー情報をラインプリンタに出力し，ディバッグの資料とすると共に，XY プロッタジョブ仕分情報の自動出力等により，オペレーションの自動化を図ることにあった．そこで，次に変更及び新しく追加した機能についてその概要を示す．

(イ) システムタイマの調整

システムタイマの遅れを調整するために，各ジョブの処理時間に 0.0328125（定数）を掛けた数値を実時間タイマに加算することにより調整を行なっている．しかし，この方法ではアイドル時間があるとタイマが遅れることになるが，モニターのタイマールーチンを改造するよりも良策であると思われる．

なお，現システムのタイマの精度は0.7秒である．

(ロ) ラインプリンタ制御命令の変更

必ず行なわれていたラインプリンタのオープン命令とページコントロールを END プログラム制御パラメータの9文字目により制御を行なうように変更した．

(ハ) 仕分情報の花文字出力

仕分けミスを防ぎ，見やすくするために，ジョブ番号下2桁及び，課題番号下3桁の花文字出力カルーチンを新しく作成した．

(ニ) コンパイルエラー情報の出力

FORTRAN コンパイラの一部変更に伴うコンパイル時のエラー情報を，DCT 領域の0番地の値を参照することにより出力する．

(ホ) 結合登録（PRBL）処理エラー情報の出力

DCT 領域の7番地で，1つのジョブで結合登録（PRBL）処理が2回以上行なわれないようにチェックしている（¥JOB で0を，¥PRBL で1を格納している）ので，この番地が1ならば，ソースプログラムには，FORTRAN コンパイラが検出できる文法エラーは無かったことになる．そこで，次に結合登録（PRBL）処理で検出したエラーの有無を調べる．この処理は，PRBL 処理プログラムが主記憶上に残した情報を探すことにより行なっている．そのために，END プログラムを主記憶の後部のアドレスで実行させるという手法をとった．

(ヘ) 実行プログラムのロケーション及びワード数出力

この処理は，DCT 領域に残されたメインプログラム及びサブプログラムの各エレメントの R.B. 先頭レコードアドレスから，内蔵ドラムの作業用領域に残されている各エレメントの R.B. 情報を参照することにより，プログラム格納アドレス及び，ワード数の計算を行ない，ラインプリンタに出力する方法をとった．

ところで，結合登録（PRBL）処理をしないジョブやドラムを使用するジョブの R.B. 情報は，無かったり，破壊されたりしているので，これらの場合は，この処理ルーチンを通過しないように，この前の時点でチェックを行なっている．

(ニ)，(ホ)，(ヘ)の処理のフローチャートを Fig. 3 に示す．

(ト) XY プロッタジョブ仕分情報の自動出力

XY プロッタジョブの仕分情報が，XY プロッタ用紙に出力されないと，仕分作業が非常に面倒である．そして，XY プロッタジョブが連続して処理された場合，オペレータが目を離すと，前のジョブで作成した図面上に作図するということがよく発生する．この様なことに対するオペレータの仕分及び，監視作業の軽減化のために，XY プロッタジョブ仕分情報の自動出力及び，ペンの自動移動（仕分情報出力後17cm X 軸正方向にペンの移動）を行なわせるルーチンを作成した．

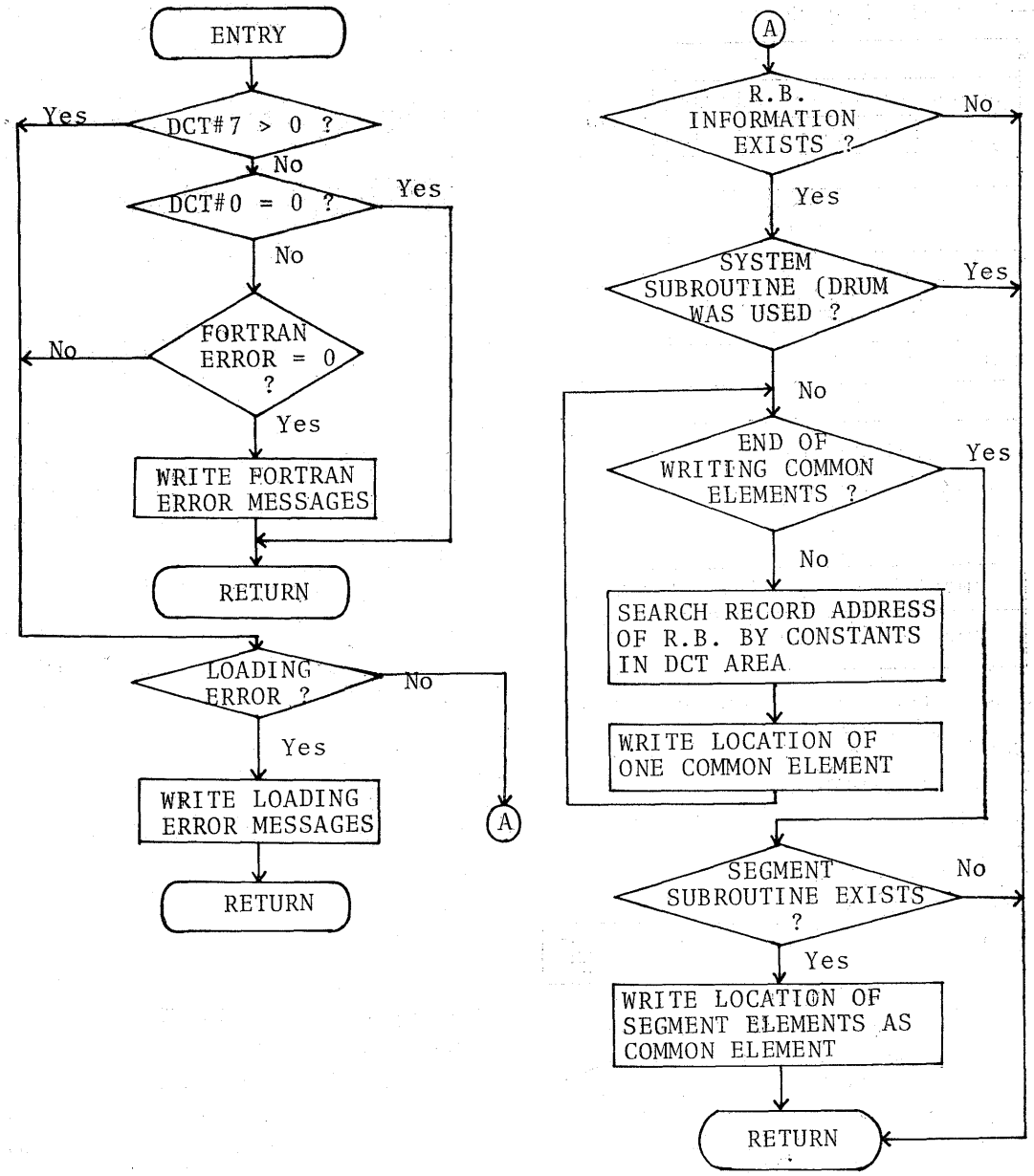

Fig. 3 Flowchart for the routine of writing error messages.

このルーチンは，ユーザのプログラム情報 (PLL : Program Location List 又は，SET : Segment Table) より，XY プロッタ使用ジョブであるか否かを判定し，プロッタジョブの場合は，ペンをジョブの最終位置より Y 軸負方向へ中心点より135㎜ 移動させ，ジョブ番号及び，課題番号下6桁を出力させるものである．

実行プログラムは，Fig. 4 の様にプログラムに関する種々の情報を集めたリストをその先頭に持っている．コモン・エレメントのそのリストを，PLL (Program Location List) といい，セグメント・エレメントのそれを，SGT (Segment Table) という．SGTの構成は，PLL の #13番地からと同様になっている．XY プロッタジョブでは，必ずシステムサブルーチン「OPNPL」が使用され，これはドラム上のアドレスが #0EB レコードで，大きさが5レコード故，PLL 又はSGT 情報の「サブルーチンのドラム上の位置」の情報が #0EB5 である．従って，PLL 又は，SGT 情報を検索した場合，「サブルーチンのドラム上の位置」の情報に「#0EB5」を持つジョブは，XY プロ

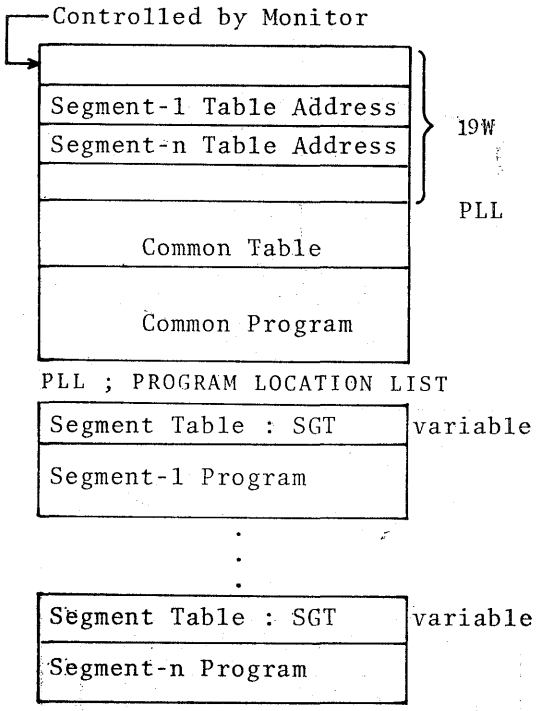

Fig. 4 Relation between PLL and Program.

ッタ出力ジョブであることになる．ジョブ番号，課題番号は，ドラムの DCT 領域に格納されているので，その情報をプロッタサブルーチン「SYMBOL」の引数に格納し，XY プロッタに出力するという方法をとった．このルーチンの概略を Fig. 5 に示す．

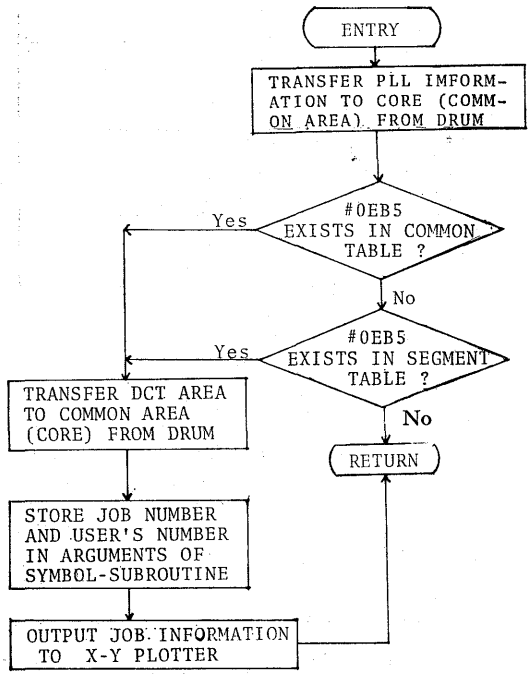

Fig. 5 Flowchart for output of JOB information to X-Y Plotter.

3-4 JOB の一部改造について

このプログラムは，1つのジョブの始まりを意味し，モニターが管理しているコントロールデータ（DCT 領域のデータ）の初期設定を行なうものである．

¥JOB 制御カードのパラメータで，計算処理打ち切り時間と，ラインプリンタ出力打ち切り枚数の指定ができるが，パラメータの指定の無い場合 T = 60分，LP = 100枚を限度とするために改造を行なった．

この機能を付加するために，このプログラムのアセンブリ言語形式のプログラムリストを作成し，解析した．そして，モニタ領域である主記憶の #27A，#27B 番地を，T の指定が無い場合の JOB TIME LIMIT 定数とし，#279番地を，LP の指定が無い場合の LP LIMIT 定数とした．

この機能を付加するために，17ワード分，内蔵ドラムに機械語で直接書き込み改造した．

この処理プログラムのフローチャートを Fig. 6 に示す．

3-5 KILL の改造について

このプログラムは，バッチ処理を中断することなく次のジョブに制御を移すための制御プログラムである．オペレータの介入やジョブの制限から自動的に，打ち切られたメッセージを出力するために，アセンブリ言語（FASP）により新規作成した．

3-6 RBD, RBR の改造について

RBD, RBR プログラムは，この2つで対になって使用される．RBD プログラムは，ドラム上のリロケータブルバイナリ情報を，マガジンファイル又は，紙テープに掃き出すもので，RBR プログラムは，RBD プログラムによって掃き出されたマガジンファイル又は，紙テープのリロケータブルバイナリ情報を，ドラムに読み込むものである．

この2つのプログラムは，ほとんどマガジンファイル用に使われているが，メーカー提供のプログラムには，マガジンファイル用の R.B. 情報に無駄があり，実際は½の情報で済む．SSL をマガジンファイルに入れて使用しているので，読取り時間短縮のためにも，この R.B. 情報は短かい方が良い．

4. あとがき

以上のシステム改造は，電子計算機を共同利用とい

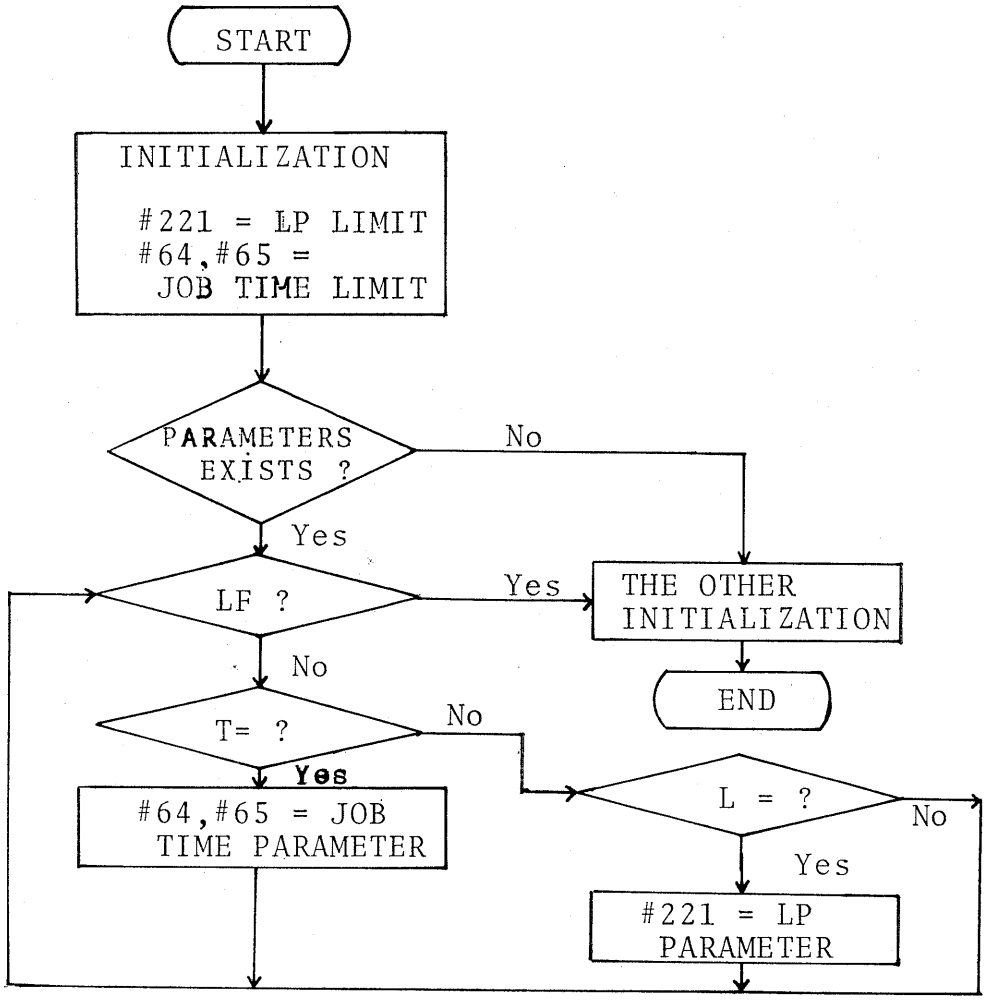

Fig. 6 Flowchart for JOB Program.

う立場から運用する場合に要求される様々な問題点を解決するために行なったが，オペレーションの自動化，省力化にもつながったと思われる．しかし，ハードウェアの制約からくる様々の制限により，思うようなシステムの改良，改造にまでは至らなかった．

参考文献

1) FACOM 270-20/30 MONITOR Ⅲ$_2$/Ⅲ$_3$ 解説編
2) FACOM 270-20/30 MONITOR Ⅲ$_2$/Ⅲ$_3$ 仕様書
3) FACOM 270-20/30 MONITOR Ⅲ$_2$/Ⅲ$_3$ 操作仕様書
4) FACOM 270-20/30 FASP 仕様書