FACOM 230-60 マクロコマンド会話型トランスレータ*

辻ケ堂　信**　白井　一暢**　伊藤　寿彦**

Abstract

The requirments of the introduction of the FACOM 230-60 Macro Command Conversational Translator, the grammatical specification and the structure are discussed. The introduction is required to make easy the modifications and standardization of the conversational job control language without any change of the internal job control programs. The structure reflects the design policy that many users can utilize the translator simultaneously and the techniques of read-only and roll-in/out are adopted.

1. コマンドの範囲

会話型処理方式で計算機を利用するとき，まず，目的のジョブステップを動作させる必要がある．具体的には，このジョブステップを行なうためのプログラムを定め，必要な主記憶や入出力装置を割り当てて，実行を開始させる．ジョブステップが実行を開始すると，データの授受を計算機利用者との間で行ない，会話形式で仕事が進む．仕事が終了すると，そのジョブステップについての会計情報の一部が端末に印刷され，つぎのジョブステップに移るか，または計算機の利用を終了する．

FACOM 230-60 の TSS コマンドとは，計算機利用者が計算機システムを利用するとき，目的のジョブステップを動作させるまでに必要な計算機に対する指令，計算機からの問合せに対する応答，ジョブステップの終了後，つぎのジョブステップを開始するまで，または，ジョブステップを完了し計算機の使用をやめるまで，または，ジョブステップを完了し，つぎのジョブステップに移行するまでに必要な計算機に対する指令と計算機からの問合せに対する応答，およびプロブレムプログラムにはできないサービスを依頼するシステムを制御するコマンドよりなる．さらに，正確には“3.2.1 コマンドの文法”に記述されているように，FACOM 230-60 TSS コマンドはマクロコマンド呼出し指令，システム制御コマンド，マクロコマンド会話型トランスレータ（MCCT）よりの問合せに対する応答および登録用マクロコマンドよりなる．このうちシステム制御コマンドは，プロブレムプログラムにはできないサービス，つまり，リモートバッチ処理を開始したり，システム出力ルーチンを制御したり，実行中のジョブステップを強制的に終了させることを行なうもの，あるいはプロブレムプログラムが行なうことができるが，わざわざジョブ制御プログラムにより動作を開始させるには，あまりにも手軽すぎるサービスを行なうもの（たとえば，日時の表示）の非常に少数に限られている．

現在，ほかの計算機システムでは，プロブレムプログラムを動作させればよいような，かなりの仕事に対する指令も，コマンド中に分類する例が多いが，FACOM 230-60 ではテキストエジタ（LINED）に対する制御指令，会話型言語（BACCUS）に対する制御指令，マクロコマンド登録ルーチンに対する制御指令というように区別し，これらのプロブレムプログラムが TSS コマンドにより動作を開始するという考え方をとっている．

2. マクロコマンド会話型トランスレータ導入の動機

2.1 コマンドの変更と標準化

コマンドの最も大きな役割は，バッチ型のジョブ制

---

* FACOM 230-60 Macro Command Conversational Translator, by Makoto Tsujigado, Kazunobu Shirai and Toshihiko Ito (Fujitsu Ltd.)

** 富士通株式会社ソフトウェア技術部

御と同じく目的とするジョブステップを動作させることであり，これに会話的要素，つまり計算機があるパラメータについて利用者にたずね，利用者からそのパラメータの値をもらうことが加わり，このことが計算機利用者を，ジョブ制御言語を覚えることから解放している．

このように会話によりジョブステップを動作させるとき，適用業務により会話の内容は非常に変わる．たとえば，高度なファイル処理を行なう場合には，ファイルを定義するためのパラメータは非常に複雑になる．ということは，利用者のレベルによって，ある場合には複雑なパラメータを問い合わせ値をもらい，ある場合には，システムで前もって定めてあるパラメータの値を押しつける必要が生じる．

つぎに会話の型に各計算センタごとにかなりな変更が予想される．LOGIN を使うか LOGON を使うかは，好みの問題と片づけるにしても，会話を英文で行なうか，ローマ字で行なうか，あるいは端末装置が許せば，かなを用いて行なうかは，特定の計算センタにとっては重要な問題である．特に，公共利用計算センタ，いわゆる，Computer Utility のようなものに発展していく場合には，小学生にも使わせなければならない．このとき，計算機から "YOUR PASSWORD INVALID. ABORTED." というメッセージを出すのでは酷である．

さらに，会話型処理が広く使われだすと，JIS や ISO によるコマンドの標準化も，いずれは必要になると思われる．このことは，単に，人間と計算機という関係だけではなく，計算機と計算機が共同作業を行なうためにも必要である．

ところが，現在は，まだ，コマンドの開発段階であり，この標準化にはほど遠い．また，複雑な仕事を行なう利用者にも，簡単な仕事を行なう利用者にも，同一のコマンドを押しつけるのも好ましくない．そこで，基本的なジョブ制御文を決めておき，コマンドをこの基本制御文に変換することを考えてみる．これまでにもバッチジョブの範囲では，IBM 360 でのカタログドプロシジュア[1] およびそれを発展させた FACOM 230-60 でのジョブ制御マクロ[2] のように，基本となるジョブ制御文をマクロライブラリ中に登録しておき，計算機利用者により，与えられたパラメータに従って展開するという考え方があった．しかし，これらはバッチジョブのために開発されたものであり，利用者からシステムへの一方通行の指令である．会話型の場合には計算機から利用者への問合せの複雑さの程度ということが利用者にとってはるかに重要である．

以上より FACOM 230-60 会話型ジョブ制御では，マクロコマンド会話型トランスレータというものを用意し，計算機利用者とシステムの間に，かなり自由度のある会話が行なえるようにした．この手法は FACOM 230-60 に限らず，広く利用されてよいと思うので，ここに発表させていただく次第である．

2.2 バッチジョブ制御プログラムとの整合

制御プログラムを作成するとき，共通の機能単位，いわゆる，プログラムモジュールはできるだけ共用することが望ましい．バッチ型ジョブ制御と会話型ジョブ制御について，共通になりそうなところはシステム入力制御，ジョブステップイニシェータ，実行時のプログラム管理，ジョブステップターミネータおよびシステム出力制御（会話型ジョブでも，あとから見てもよい情報や大量の出力情報は，センタの印刷装置に出力するから）などである．ただし，バッチジョブはジョブというものを単位とし，このジョブが複数個のジョブステップに分れているのに対して，会話型のジョブでは，ジョブステップしかはいってこず，前もってジョブ全体を定義することは困難である．ここにマクロコマンド会話型トランスレータの存在意義が再び生じる．すなわち，会話型のジョブステップにジョブの着物を着せて（具体的には ¥NO 文[2]，¥JOB文[2] をジョブステップの前におき，¥JEND[2] 文をジョブステップのうしろにおく），バッチジョブと同じ型にしてしまう．これによりバッチジョブのために作成されたシステム入力制御，ジョブステップイニシェータおよびジョブステップターミネータが，そのまま利用されることになる．これが MCCT を導入した第二の理由である．

2.3 マクロコマンドと動詞表方式の比較

基本コマンド言語を拡張する方法としては，動詞表を追加する方式がある[4]．この場合，新しいコマンドを具体化するプログラムは，コマンド言語システムのインタフェース規定に整合するように書き，その上でコマンドの名前が動詞表中におかれる．つまり，動詞表によりコマンドを識別し，対応するコマンドプログラムが動作するよう，結合の手段となっている．

この長所は，コマンドに対するコマンドプログラムが，ジョブステップイニシェータを経由することなく，すぐ動作するので，システム内部に待行列ができにくいことである．短所はコマンドの追加削除のた

め，かなり大きな部分のリアセンブルが必要である．さらに，各計算センタごとに会話様式を任意に定めることは，コマンドプログラム全部を書き直すことと等価と思われる．

さらに，動詞表方式の場合，各計算センタで，たとえば，利用者のレベルに合わせて，LOGIN と KONNICHIWA の二つのコマンドを用意する気にはなれないと思う．

3. マクロコマンドの概要

3.1 例題による説明

表で左端の欄は端末での会話の状況を示し，計算機から送出され印刷された部分には下線が引いてあり，計算機利用者が打鍵するものには下線がない．また，9は数字，Aは文字，Xは英字または数字を意味する場合があるが，これは問合せ文の意味から推察していただく．左から見て第2欄はマクロコマンドの内容であり，これは登録用マクロコマンドの形で書いてある．ここで第 1，2 桁の¥/は直接作用文の記号であり，MCCT はこの指示どおり動作する．そしてジョブ制御文（¥NO，¥JOB，¥EXEC，¥FD，¥＊，¥JEND）に出会うと，受動パラメータをその値でおきかえて出力していく．左から第3欄にはこの MCCT の出力情報が記述されている．右端には備考欄があり，対応する行の説明を行なっている．

第1表は会話型ジョブを開始する例題である．ここでは，この端末に対するシステムの状態が，コマンド待ち状態になったところから示されている．このためには端末に電源を入れ，呼出しボタンを押せば，端末に¥が印刷され，システムはこの端末に対して，コマンド待ち状態にあることが示される．

第2表はテキストエジタ LINED を動かし，原プログラムの編集を行ない，つぎに，これを FORTRAN コンパイラにより翻訳し，つぎに，そのエラーリストを端末に印刷するという三つのジョブステップを行なうマクロコマンドの例である．

3.2 コマンド一覧

3.2.1 コマンドの文法

FACOM 230-60 TSS コマンドの文法（第 18 ページ参照）を記述するため，つぎの規約をおく[3]．

（1） A→Bとい形は，AをBと書き換えよの意．

（2） [A] は，Aを書いても書かなくてもよいことの意．

（3） $\left\{\begin{matrix}B\\C\end{matrix}\right\}$ または {B|C} は，BまたはCの意．

（4） A→$\left\{\begin{matrix}B\\C\end{matrix}\right\}$ または A→{B|C} は $\left\{\begin{matrix}A\to B\\A\to C\end{matrix}\right\}$ の意．

註 以下，TSS コマンドの文法を見るため，二，三の用語の意味を記述する．なお，直接作用文がMCCT の特徴であるので，これは **3.2.2** に改めて記述する．

（1） MCCT はマクロコマンド会話型トランスレータの略称である．

（2） 能動パラメータは，そのパラメータの属するマクロコマンドが計算機利用者により呼ばれたとき，または，そのパラメータの属する直接作用文が実行されたとき，そのパラメータに値が与えられるものである．

（3） 受動パラメータは，能動パラメータにより定められた値を参照するパラメータである．

（4） 能動パラメータの後部のカッコをはぶいたものと，受動パラメータの頭の¥をはぶいたものとが等しいとき，この能動パラメータにより与えられた値が，受動パラメータにより参照される．

（5） 受動パラメータは，ジョブ制御定義文中にあるとき，その値でおきかえられた後，MCCT の出力となる．直接作用文中にあるとき，記号列の外にあれば，その値が使用され，記号列の中にあれば，受動パラメータの場所が，その値で置換される．

3.2.2 直接作用文の意味

（1） CALL

モジュール名で指定されているコマンドモジュールに制御を渡すもの．

（2） DELETE

SAVE 文でとられたパラメータの値を消去し，このためにとられていたスペースをシステムに返却する．

（3） IF

オペランドの値が真ならつぎの行を解釈する．偽なら対応する REND までとび，そのつぎの文を解釈する．IF と REND はカッコ構造により対とされる．

（4） JC (Job Closer)

ジョブクローザを動作させ，ジョブを終結する．

（5） JO (Job Opener)

ジョブオープナを動作させ，ジョブの開設を行なう．

（6） JSO (Job Step Opener)

ジョブステップオープナを動作させる．ジョブステップオープナは，ここまでに MCCT の出力として作成されたジョブデックをバッチ用システム入力制御に

第 1 表

端　末	(登録用) マクロコマンド	MCCTの出力	備　考
¥HELLO	DEFINE HELLO		マクロコマンド呼出し指令HELLOにより、マクロコマンドが呼出される。
TOHROKU BANGO=999AA999	¥/WTUR 'TOHROKU BANGO=',BANGO(8)		登録番号と計算機利用者に要求する。WTURはWRITE TO USER with REPLYの意である。
YOUR NAME=AXXXXXXX	¥/WTUR 'YOUR NAME=',NAME(8)		計算機利用者の名前を要求。
	¥/CALL REGISTRY(¥BANGO,¥NAME,RESULT(1))		登録番号検査モジュールを呼び出す。このモジュールREGISTRYは各計算センタごとに作成される。そこでは、もしRESULTの値が'0'なら未登録のゆえ、計算機を利用させない。¥/IFはオペランドが真値をもてば直後の行をMCCTが解釈する。偽なら最初の¥/RENDまでとばす。
	¥/IF ¥RESULT.EQ.'0'		
[UNREGISTERED. THEREFORE ABORTED.]	¥/WTU 'UNREGISTERED. THEREFORE ABORTED.CR'		WTUはWRITE TO USERの意
	¥/TRANSIT OFF		この端末に対するシステムの状態をオフライン状態とする
	¥/REND		
YOUR PASSWORD=XXXXXXXX	¥/WTUR 'YOUR PASSWORD=',PASSWORD(8)		合言葉を計算機利用者に要求する。
	¥/CALL CKPSWD (¥NAME, ¥PASSWORD, ACCESS(1))		パスワードチェックモジュールを呼び出す。このモジュールは各計算センタごとに作成される。
	¥/IF ¥ACCESS.EQ.'0'		¥ACCESSの値が'0'ならパスワードが合わない。'1'なら最初の¥RENDまでとばす。
[YOUR PASSWORD INVALID. ABORTED.]	¥/WTU 'YOUR PASSWORD INVALID. ABORTED.CR'		
	¥/TRANSIT OFF		この端末に対するシステムの状態をオフライン状態にする。
	¥/REND		
	¥/CALL GNRTJN (JOBNUMBER(12))		この例では各計算センタのルーチンがジョブ番号を作り出し、計算機利用者に知らせている。戻りパラメータの値が4桁なら、その値は続4桁にはいり、以後ブランクがはいる。実際に与えられた値の桁数が用いられる。
YOUR JOBNUMBER=A999.	¥/WTU 'YOUR JOBNUMBER=¥JOBNUMBER CR'		
	¥/SAVE(1) ¥BANGO, ¥NAME, ¥JOBNUMBER		登録番号、計算機利用者名およびジョブ番号を後の利用のため保存する。
	¥NC ¥JOBNUMBER	¥NO A999	バッチ用ジョブ制御文と同じものがMCCTの出力となる。
	¥JOB ¥BANGO, ¥NAME	¥JOB 999AA999, AXXXXXXX	
	¥/JO		ジョブオープナ「JO」が動作し、デマンドジョブの開設となる。
	¥/CALL DATETIME (DATETIME(16))		このときの日付および時刻を与える。
JOB A999 OPENED AT 1968.06.25 16.25	¥/WTU 'JOB ¥JOBNUMBER OPENED AT ¥DATETC IME CR'		マクロコマンド定義本体文が1行にはいらないときは第72桁目にブランク以外のコードを入れて、次行につづくことを示す。 ⓒはCONTINUATIONを表わし(第72欄)空白以外の文字を記入したことを示す。
CALL NEXT MACRO	¥/WTU 'CALL NEXT MACRO CR'		
¥	¥/TRANSIT TM		この端末に対するシステムの状態をコマンド待ち状態とする。
	DEND		

第 2 表 (1)

端　末	(登録用) マクロコマンド	MCCTの出力	備　考
¥LFD	DEFINE LFD ¥/RESTORE(1) BANGO(8), NAME(8), JOBNUMBER(12)		マクロコマンドLFDを呼び出す。L=LINED, F=FORTRAN, D=DISPLAYの意。前のマクロコマンドHELLOで保存しておいた登録番号，利用者名およびジョブ番号をこのマクロでも利用するため，その値を設定する。
	¥NO ¥JOBNUMBER	¥NO A999	
	¥JOB ¥BANGO, ¥NAME, MSGLVL=ERROR, TIME=(3.00,12.00), CORE=16	¥JOB 999AA999, AXXXXXXX, MSGLVL=ERROR, TIME=(3.00,12.00), CORE=16	ジョブ制御定義文は，その中の受動パラメータをその値で置換された後，MCCTの出力情報となる。
LINED INFILE=OKADA.DATA1(M1)	¥/WTUR 'LINED INFILE=', INFILE(54)		LINEDの入力ファイルの名前をたずねる。
LINED OUTFILE=OKADA.DATA2(M2)	¥/WTUR 'LINED OUTFILE=', OUTFILE(54)		LINEDの出力ファイルの名前をたずねる。
	¥EXEC LINED, PRTY=(27,7)	¥EXEC LINED, PRTY=(27,7)	ジョブ制御のEXEC文
	¥FD OLDFILE, FILE=(OLD, ¥INFILE)	¥FD OLDFILE, FILE=(OLD, OKADA.DATA1(M1))	ジョブ制御のFD文
	¥FD OUTFILE, FILE=(NEW, ¥OUTFILE), DISP=CATLG	¥FD OUTFILE, FILE=(NEW, OKADA.DATA2(M2)), DISP=CATLG	
	¥FD WORKFILE1	¥FD WORKFILE1	
	¥JEND	¥JEND	ジョブ制御のJEND文
LINED WILL BE EXECUTED ここでLINEDとの会話が行われる。	¥/WTU 'LINED WILL BE EXECUTED'		
	¥/JSO		ここでジョブステップオープナが動作する。つまり上のNO文からJEND文までがシステム入力制御により製表化され，ジョブステップオープナに渡たされ，つぎのジョブはイニシェータにより実行開始を行なう実行の終了待ちタイミング経由ジョブステップクローザに渡り，MCCTはつぎの¥/CALLの解釈から再開する。
LINED FINISHED AT 1968.06.25 17.00	¥/CALL DATETIME(DATETIME(16))		
	¥/WTU 'LINED FINISHED AT ¥DATETIME'		
	¥NO ¥JOBNUMBER	¥NO A999	ジョブ制御のNO文，ここからFORTRANコンパイラの実行である。
	¥JOB ¥BANGO, ¥NAME, MSGLVL=ERROR, TIME=(1.00,2.00), CORE=40, SYSOUT=(10,300)	¥JOB 999AA999, AXXXXXXX, MSG_VL=ERROR, TIME=(1.00,2.00), CORE=40, SYSOUT=(10,300)	ジョブ制御のJOB文
	¥EXEC FORTRAN, PRTY=(17,17)	¥EXEC FORTRAN, PRTY=(17,17)	ジョブ制御のEXEC文，ロールアウトを行なわないようプライオリティをつけてある。
	¥FD SOURCE, FILE=(OLD, ¥OUTFILE)	¥FD SOURCE, FILE=(OLD, OKADA.DATA2(M2))	ジョブ制御のFD文
	¥FD RELBIN, FILE=(NEW, TEMP, ¥NAME), UNIT=F461K, VOL=WORK, SPACE=(1080, 128)	¥FD RELBIN, FILE=(NEW, TEMP, AXXXXXXX), UNIT=F461K, VOL=WORK, SPACE=(1080, 128)	オブジェクトプログラムの出力されるファイルが定められる。
	¥FD SYSPRT, FILE=(TEMP, T6), SPACE=(136, 128), UNIT=F461K, VOL=WORK, SYSOUT=PRINT	¥FD SYSPRT, FILE=(TEMP, T6), SPACE=(136, 128), UNIT=F461K, VOL=WORK, SYSOUT=PRINT	ソースプログラムのリストはシステム出力経由で印刷する。
	¥FD ERROR, FILE=(NEW, ERROR, ¥NAME), DISP=CATLG	¥FD ERROR, FILE=(NEW, ERROR, AXXXXXXX), DISP=CATLG	エラー表示用ファイルを定義。このファイルはつぎのジョブステップで使われる。
	¥FD SYSWK1, FILE=(TEMP, T1), SPACE=(1080, 128), UNIT=F461K, DISP=DELETE	¥FD SYSWK1, FILE=(TEMP, T1), SPACE=(1080, 128), UNIT=F461K, DISP=DELETE	
	¥FD SYSWK2, FILE=(TEMP, T2), SPACE=(1080, 128), UNIT=F461K, DISP=DELETE	¥FD SYSWK2, FILE=(TEMP, T2), SPACE=(1080, 128), UNIT=F461K, DISP=DELETE	
	¥FD SYSWK3, FILE=(TEMP, T3), SPACE=(1080, 128), UNIT=F461K, DISP=DELETE	¥FD SYSWK3, FILE=(TEMP, T3), SPACE=(1080, 128), UNIT=F461K, DISP=DELETE	
	¥FD SYSWK4, FILE=(TEMP, T4), SPACE=(1080, 128), UNIT=F461K, DISP=DELETE	¥FD SYSWK4, FILE=(TEMP, T4), SPACE=(1080, 128), UNIT=F461K, DISP=DELETE	
	¥JEND	¥JEND	
FORTRAN WILL BE EXECUTED	¥/WTU 'FORTRAN WILL BE EXECUTED' ¥/JSO EXECUTE(1)		ここでジョブステップオープナが動作する。前のジョブステップの実行の結果このジョブステップの実行が行なわれなければEXECNTE(1)='0'となる。その値が'1'なら最初の¥/RENDのつぎへとぶ。
NO FORTRAN EXECUTED, BECAUSE OF THE LINED EXECUTION	¥/IF ¥EXECUTE.EQ.'0' ¥/WTU 'NO FORTRAN EXECUTED, BECAUSE OF THE LINED EXECUTION.'		
¥	¥/TRANSIT TM ¥/REND ¥/CALL DATETIME(DATETIME(16))		FORTRANの実行ができないときは，つぎのマクロコマンドを呼ぶため，コマンド待ち状態とする。

第 2 表 (2)

端末	(登録用) マクロコマンド	MCCT の出力	備考
FORTRAN FINISHED AT 1968. 06. 25 17.01	¥/WTU 'FORTRAN FINISHED AT ¥DATETIME Ⓕ		
	¥NO ¥JOBNUMBER	¥NO A999	以下はエラー表示ルーチンの実行である．FORTRANコンパイラが見つけ出したシンタクス上のエラーを表示する．
	¥JOB ¥BANGO, ¥NAME, TIME=(1.00, 4.00), CORE Ⓒ	¥JOB 999AA999, AXXXXXXX, TIME=(1.00, 4.00)Ⓒ	
	=8,	CORE=8	
	¥EXEC DISPLAY, PRTY=(27,7)	¥EXEC DISPLAY, PRTY=(27,7)	
	¥FD INFILE, FILE=(OLD, ERROR.¥NAME), Ⓒ	¥FD INFILE, FILE=(OLD, ERROR.AXXXXXXX), Ⓒ	入力ファイルはFORTRANで作成されたエラーファイルである．
	DISP=DELETE	DISP=DELETE	
	¥JEND	¥JEND	
DISPLAY OF ERRORS.	¥/WTU 'DISPLAY OF ERRORS. Ⓕ'		
	¥/JSO		ここでジョブステップオーブナが動作する．
ここでエラーリストのダンプが行なわれる．			
DISPLAY OF ERRORS FINISHED.	¥/WTU 'DISPLAY OF ERRORS FINISHED. Ⓕ'		
*	¥/TRANSIT TM		この様におけるシステムの状態はコマンド待ち状態とする．
	DEND		

依頼し製表化してもらう．つぎに，この製表化されたジョブデックをジョブステップイニシェータに依頼し，実行を開始させる．

（7） REND

IF 文と組み合わせて，条件が成立しないとき無視するジョブ制御文の範囲の終わりを示す．

（8） RESTORE

SAVE 文で保存されたパラメータの値をとりよせる．これは二つ以上のマクロコマンドの間で，パラメータの値を利用するとき使用する．

（9） SAVE

パラメータの値を保存し，これよりあとに利用されるマクロコマンドに渡すための直接作用文である．保存されたデータはジョブクローザ JC が動作するか，または，同じ保存番号を持つ DELETE 文が解釈されるまで残る．

(10) TRANSIT

端末から見たシステムの状態をオフライン (TRANSIT OFF のとき)，またはコマンド待ち（TRANSIT TM のとき）状態とする．

(11) WTU (Write To User)

オペランドに記入されている記号列を端末に送り印刷する．

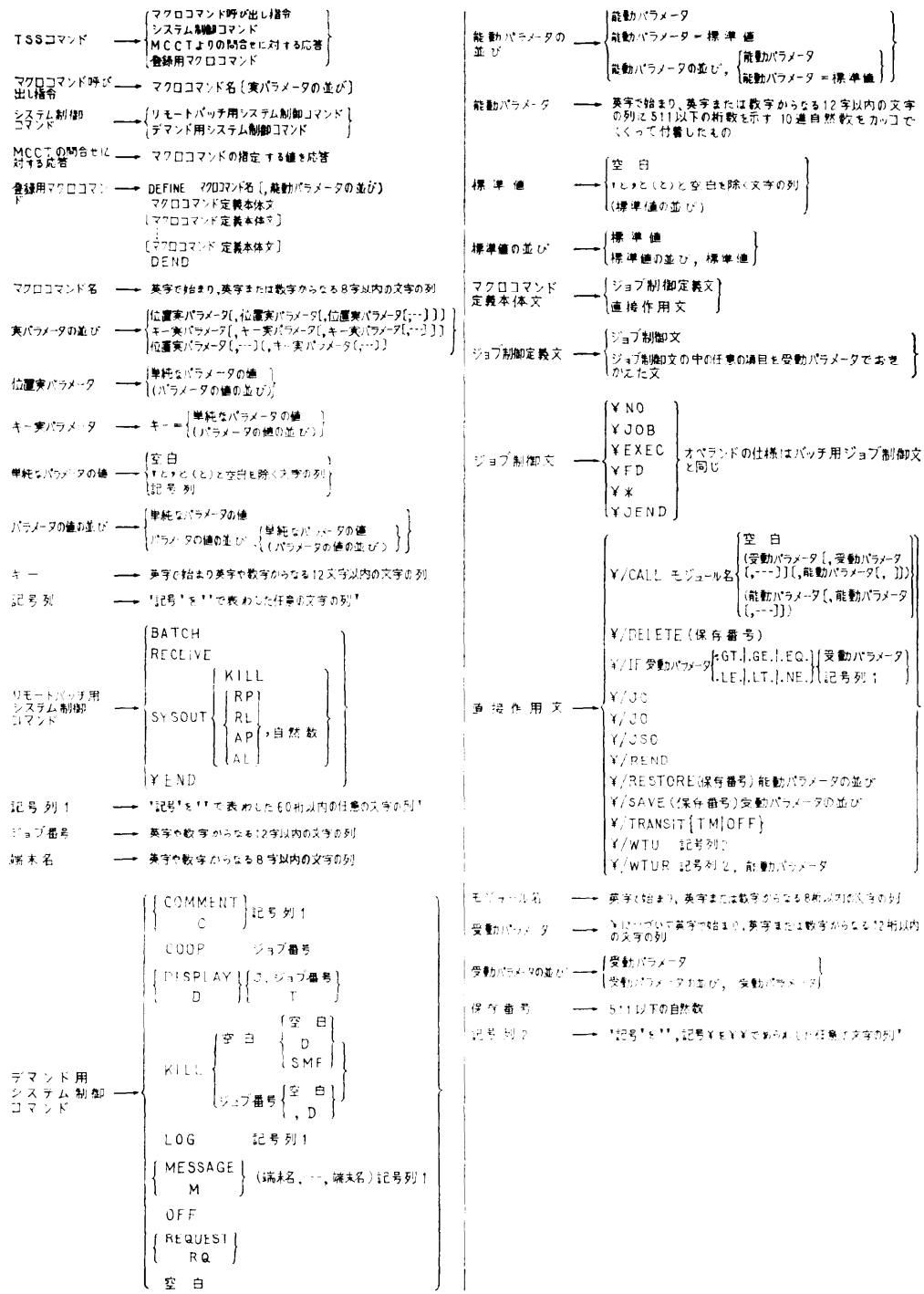
(12) WTUR (Write To User with Reply)

オペランドにある記号列を端末に送り，その答を指定されたところに受け取る．これによりパラメータの値が与えられる．

4. FACOM 230-60 マクロコマンド会話型トランスレータの構造

4.1 マクロコマンドライブラリ

登録用マクロコマンドは，分割型順編成ファイル中にマクロコマンド登録ユーティリティを用いて登録される．MCCT が解釈する時間を短くするため，登録用マクロコマンドは，登録時にマクロコマンド本体，パラメータ値表，およびパラメータ名前表に分割される．そしてマクロコマンド本体中のパラメータは，すべてパラメータ値表中の特定番地をさすようにされ，もし，パラメータに標準値



FACOM 230-60 TSS コマンド文法一覧

がついているときには，標準値はこのパラメータ値表中の対応箇所におかれる．パラメータ値表中には，パラメータ名前表中の対応箇所をさす情報が用意され，マクロコマンドの内容を表示する要求にこたえている．

4.2 トランスレータの構造の概要

FACOM 230-60 マクロコマンド会話型トランスレータの設計に際しては，つぎのような構造をとることにした．

（1） マクロコマンド会話型トランスレータのプログラム部分は，全端末に共通に一つだけおかれる．ただし，全端末がトランスレータを必要としないときには，その占有している主記憶を制御部分を除き，システムにあけ渡す．

（2） 各端末ごとにマクロコマンド本体，およびパラメータ値表を読み込むための領域を用意する．ただし，マクロコマンド本体は，プロブレムプログラムが始まるときに捨ててしまい，プロブレムプログラムの終了とともに再び読み込む．パラメータ値表はプロブレムプログラムの開始にさきだちロールアウトし，終了後ロールインする．このため各端末ごとに 16 語の常駐部分を用意し，マクロコマンド会話型トランスレータの制御表とする．

（3） SAVE, RESTORE を制御するデータ保存制御表も，パラメータ値表と同じくプロブレムプログラムの実行にさきだちロールアウトされ，終了後ロールインされる．

（4） 各端末ごとに MCCT の出力用ファイルを用意する．このファイルはバッチ用システム入力制御が読みとれる構造とする．

5. マクロコマンドの組立てとその利用

このシステムでは，マクロコマンドの組立ては，各計算センタの責任で行なわれることを目標にしている．つまり，登録用マクロコマンドの作成に誤りがあると，登録時，使用時およびシステム入力制御の 3 段階で検査されるにせよ，この検査により発見されない誤りが沢山存在しうる．したがって，マクロコマンドは他の制御プログラムと同様に，誤りのないことが必要である．また，JOB 文，EXEC 文中にある優先度についてのパラメータの値も，各計算センタの運営方針にしたがって決められるもので，一般利用者が任意に値を設定する必要はないと思われる．

したがって，計算センタの運営管理者は，登録用マクロコマンドを作成し，登録するとともに，応答規約を作り，一般利用者に配布することが要請される．

謝　辞

京都大学および九州大学の各位は，FACOM 230-60 導入の立場から，本仕様を検討され有益な意見を多数寄せられた．また，会話型ジョブ制御は，すでに設計の完了していたバッチ用ジョブ制御を利用する関係から，バッチジョブ制御作成担当者の考えたことを多数利用させてもらった．

以上の担当者は富士通丸山武および国沢春雄の両君である．上記両氏とともに京大萩原教授，富士通山本卓真，井関幸男および山地克郎の各位のご援助ご協力に謝辞を捉する．

参考文献

1) IBM System Reference Library, IBM System /360 Operating System Job Control Language (1965)
2) FACOM 230-60 MONITOR Vシステム，ジョブ制御言語文法編（富士通，1968）
3) Emmon Bach: An Introduction to Transformational Grammers. 1964 Holt, Rinehart and Winston, Inc. New York.
4) IBM System Reference Library System/360 Model 67 Time Sharing System Preliminary Technical Summary (1966)

（昭和 43 年 8 月 6 日受付）