FORTH ベースによる機器制御言語*

Machine Control Language Based on FORTH

小　林　信　雄[1)]
Nobuo KOBAYASHI

To control the micro-processor based machine properly the control oriented languages are desirable. In this papers the construction of such language is attempted by means of FORTH system. By its unique extensibility and portability the new language has its own vocaburaly and syntax, which makes easy to learn program. The statements are expressd in a rather natural way. And also it is well accomodated to meet the individual machine specifications.

1. まえがき

近年マイクロプロセッサーが急速な広がりを見せ，小さな測定器にすら登載されるようになった。我々の研究室においても種々の測定器からのデータを処理する為に，あるいは小規模な機器制御にマイクロプロセッサーを用いているが，ハードウェアの製作よりもソフトウェアの開発により時間を要するようになった。

これ迄は全てマシン語によるプログラムで動作させて来たが，機器のわずかな機能変更に対しても，殆んど全部最初からプログラムの再構成を行わざるを得ず，著しく効率が悪く，制御用の共用言語の必要性を痛感するに到った。

機器制御用言語として要求される性質は

1. 移殖性
2. 拡張性
3. 実行速度
4. 記述性

等である。マイクロコンピュータ用の言語としては多数のものがあるが上記の要求を満たすものは少ない。マイクロコンピュータ用の言語として標準となった感のあるのがＢＡＳＩＣであるが，組み込み用としては不向きである。通常ＢＡＳＩＣは各メーカが組み込んだ型で提供されるため，プログラムを他機種に移殖するのは難しい。又，ＢＡＳＩＣはインタープリタ型であるので演算速度の点でも難点がある。

そこで上記諸特性を満たすものとしてＦＯＲＴＨをベースとして採用する事にした。

2. FORTH

コンピュータ用言語ＦＯＲＴＨは1969年 C. Moore により作られたものである。元来が機器制御を目的としたものであったので，他言語にはない特徴を持っている。[1)]

ＦＯＲＴＨにおける実行単位は word と称されるものであり，内部構造はFig. 1に示されるような

＊昭和58年4月6日原稿受理

1)　大阪産業大学工学部機械工学科

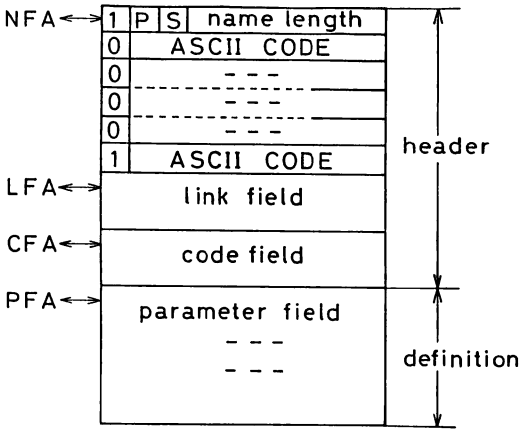

Fig. 1 Structure of word in FORTH

ものである。[2)]

ＮＦＡ (name field address) の内容はワードの識別コードである。ＬＦＡ (link field address) は直前のワードへのリンクポインタである。このようにＦＯＲＴＨにおけるワードは一方向の線型リスト構造になっている。ＣＦＡ (code field address) はワードにおいて最初に実行されるべきルーチンのアドレスである。ＰＦＡ(paramefer field address) は二種類あり，直接にマシン語が続くものと，他のワードのＣＦＡが続くものとがある。前者を一次語(primary)，後者を二次語(secordary) と称する。一次語は各ＣＰＵ毎に固有のコードで記述されなければならないが，二次語はＦＯＲＴＨによるＦＯＲＴＨ定義であり，ハードウェアの違いにはよらないものである。ＦＯＲＴＨの核は約８ＫＢであり，その中で一次語は約１ＫＢ（入力出ドライバーを除く）程度である。ＦＯＲＴＨの実行状態は

(1) ロード
(2) コンパイル
(3) 実行

に分けることができる。ロードはターミナルと外部記憶装置とから行う事ができ，ワードが入力される度は即時に実行する。入力されたワードがコンパイル指示命令であるならば直ちにコンパイルを行う。即ちセルフ・コンパイル方式になっており，インタープリティブにプログラム処理を行う事が可能となっている。

ＦＯＲＴＨにおける実行制御は，２つに分けることができる。一つは Fig. 1 に示したような，実行単位であるワードを作ることであり，コンパイル指示がワードがこれに当る。もう一つはプログラムを実行するものであり，各ワードのＣＦＡに従い実行すべきアドレスを決定していくものであり，アドレス・インタープリターがこれに当る。

機器制御用言語として要求される性質をＦＯＲＴＨにおいて見ると，次のようになる。

1. 移殖性

代表的なＣＰＵに対して標準となるソースが公表されており，今回もそれに基づいて作成した。標準のＦＯＲＴＨに要求されるハードウェア上の制約はメモリー容量のみであり10ＫＢ程度を要する。プログラムの実行に際してはアドレス・インタープリタはワードのＣＦＡのみを参照するのでターゲットに移殖する際にはＮＡＦを取り除く事ができる。又，使用しないワードを除去する事もできる。このようにすれば所要メモリ容量を大巾に減少される事が可能となる。機器構成においても，柔軟性に富んでおり，外部記憶装置，ＣＲＴ，又はキーボードすら必要とせず，しかもプログラムも殆んど変更する事なく動作させる事が可能である。

2. 拡張性

ＦＯＲＴＨにおけるワードは線型リスト構造をなしており，'辞書'と呼ばれている。従ってこのリストにFig. 1に示されるようなモジュールを追加すれば，ＦＯＲＴＨ体系自体がそれだけ拡張された事になる。まさにコンパイル・ワードの機能がそれである。このようにして新しく '辞書' に登録されたワードは他のワードと何らの区別もなくＦＯＲＴＨの構成要素となる。この点

がＦＯＲＴＨと他のコンピュータ言語と異なる点である。

3. 実行速度

実行速度を左右するのはアドレス・インタープリターである。ＦＯＲＴＨにおけるコードはFig. 1に示したように，実行可能なアドレスをポイントするのではなく，実行可能なアドレスをポイントするアドレスを指しており，この間に介入するアドレス・インタープリタのオーバヘッドが大きく，それだけ実行速度が低下する。しかしながら元来マシン語と共存できるコード体系になっているので必要な所ではマシン語で記述できる。コンパイラ型の言語実行速度には及ぶべくもないが，同じインタープリタ型のＢＡＳＩＣと比較すれば各段の差がある。

4. 記述性

ＦＯＲＴＨは全ての演算をスタック上で行う。従って記述は逆ポーランド記法になり，通常の代数式的な記述から見れば，一見解かりにくい。この点がＦＯＲＴＨ系言語の普及の障害となっている最大の理由であろう。

3. 適　　用

3-1 媒体変換装置

コンピュタの外部記憶装置の種類が多様化し，異機種でデータを共用する事が困難になっている。そこで異なる媒体間のデータ転送を目的とした装置の開発を計画した。メインには当研究室で制御用に開発した汎用システム（ＤＣＯＭ）を用い，制御言語としてＦＯＲＴＨを用いている。まだ製作途上であるが基本的な機能は持っている。転送は直列と並列とがあるがそれぞれＬＳＩ8251，8255を用いる。例として直，並列での１バイト転送ルーチンのプログラムをFig. 2に示す。１，９行はＬＳＩの定義である。２行は直列転送の出力用ワードであり，５行は入出用ワードである。２行を例にとるとその機能は次の様に定義される。

：	ワード定義開始を指示するワード
ＳＯＵＴ	ワード名
ＢＥＧＩＮ	ＵＮＴＩＬ迄のループの起点
ＣＮＴＬ−ＰＯＲＴ	ポートアドレスＯＤＤＨがスタックに読まれる
Ｐ@	スタック上のデータをポートアドレスとしてそのポートからのデータを入力し，それをスタックに読むワード
２	数値２をスタックに読む
ＡＮＤ	スタック上位の２つのデーのＡＮＤをとる。この場合にはポートデータのビット１を抽出する。
ＵＮＴＩＬ	その結果が真（０以外の数値）であればループから抜けるが，偽（数値０）であればＢＥＧＩＮに戻る。
ＤＡＴＡ−ＰＯＲＴ	送信可能になったのでデータポートに
Ｐ！	送る
；	ワード定義終了

出力データはスタックにある事が前提となっているので例えば

41 ＯＵＴ

とすればＡＳＣＩＩコード41（文字Ａ）が転送される。入力に関しても全く同様である。１バイト入出力用ワードが定義されたので，これを用いて，複数データの転送，あるいはファイル転送のワードを構築する事ができる。

```
 0 (I/O : SERIAL/PARARELL) HEX
 1 ODD CONSTANT CNTL8251 ODC CONSTANT
   DATA8251
 2 : SOUT
 3 BEGIN CNTL8251 P@ 2 AND UNTIL
 4 DATA8251 P! ;
 5 : SIN
 6 BEGIN CNTL8251 P∂ 1 AND UNTIL
 7 DATA8251 P@ ;
 8
 9 OFE CONSTANT CNTL8255 OFC CONSTANT
   CNTL8255
10 : POUT
11 BEGIN CNTL8255 P@ 1 AND UNTIL
12 DATA8255 P! ;
13 : PIN
14 BEGIN CNTL8255 P@ 4 AND UNTIL
15 DATA8255 P@ ;
```

Fig.2 Program for data transfer

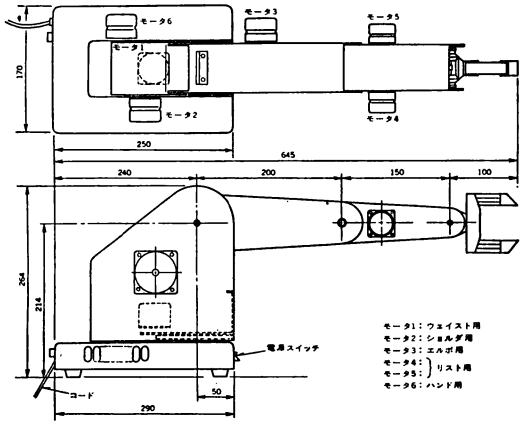

Fig.3 Diagram of micro robot

```
SCR # 40
 0 (ROBOT [PCLPOUT], LPOUT
   82-12-9/83-3-11)
 1 FORTH DEFINITIONS HEX
 2 : ROBOT TASK ; (START ROBOT)
 3 CREATE [PCLPOUT] ( 1 CHR OUTPUT TO
   LP)
 4 E1 C, 40DB, 01E6, FA20, 7D C, 10D3, 3A C,
   EA67, FEE6,
 5 40D3, 01F6, 40D3, E9FD, SMUDGE DECIMAL
 6
 7 : LPOUT (ADR, CNT-> ; FROM ADR BY
   CNT PC-LPOUT)
 8 OVER + SWAP DO I @ [PCLPOUT] LOOP ;
 9
10 50 USER CRP 52 USER STEPS 54 USER TEMP
11 : !CRP RP@ CRP ! ;
12 : ?CRP RP@ CRP @ - 39 ?ERROR ;
13
```

3-2 マイクロ・ロボットの制御

RM—101 型マイクロ・ロボットは5つの自由度を持つ腕型ロボットである。このマイクロ・ロボットはマイクロコンピュータを内蔵したインテリジェントタイプであるが，ここでは出来るだけ直接制御に近づけるため，相対移動コマンドのみを使用する。このコマンドは現在位置から指定されたステップだけステッピングモータを回転させるものである。構成をFig. 3に示す。[3)]

制御言語としては，対象装置に特徴的な表現のできる事が望ましい。例えば

「ウェストを30度回転せよ」

という命令に対して所期の動作を行うようにする。ここではこれと同等な命令として，

WAIST 30 DEGREE ROTATE

と入力する事が達成される。Fig. 4にそのプラグラムを示す。

WAISTというのは関節を特定するためのワードであり，DEGREEはスタック上の数値をステップ数に変換するワードである。ROTATEというワードは，関節の種類と移動ステップ数から，マイクロ・ロボットの制御コマンドに合ったコマンド文字列を作り，それを送信する機能を持つ。

5つの関節をそれぞれWAIST，SHOULDER，ELBOW，WRIST，HANDと名づけており，上記と同様の表現が可能である。

なお上記表現に対しては

30DEGREE WAIST ROTATE

というのは可能だが，

ROTATE WAIST 30DAGREE

という表現では所期の動作を行わない。これは，謂わばROTATEという動詞が特定のデータが以前に与えられている事を前提としているからである。丁度通常の日本語表現のように動詞が最後に置かれる事が必要である。

又，上記表現と同等なものとして

30 DEGREE TWIST

```
 14
 15 -->
SCR # 41
  0 ( ROBOT   ROBOT DEFINITION
    82-12-9/83-3-11)
  1
  2 VOCABULARY ROBOT IMMEDIATE
  3 ROBOT DEFINITIONS
  4
  5
  6 : [GA] ; : [YORI] ; : [O] ; : [KARA] ;
  7 : [E] ; : [DAKE] ;
  8
  9 0 VARIABLE ROTATE-MODE ( 0=REL :
    OTHER=ASBO)
 10 0 VARIABLE JOINT ( 0-5)
 11 0 VARIABLE CMDHLD
 12
 13 0 VARIABLE #CMD ( #OF CMD STRING)
 14 -->
 15
SCR # 42
  0 (ROBOT   ARRAY, TABLE   82-12-9)
  1 ( ----DEFINE ARRAY----- : [DIM] ARRAY
    [NAME])
  2 : ARRAY ( N-> ADR-2, N-1 IN EXEC)
  3 <BUILDS 1+ 2* ALLOT DOES> SWAP 2* +
    DUP @ ;
  4 ( ----DEFINE TABLE--- : TABLE [NAME]
    [N1], [N2], ...)
  5 : TABLE (INDEX->VALUE)
  6 <BUILDS DOES> SWAP 2* +@ ;
  7 DECIMAL
  8 6 ARRAY MAG ( CLEAR IT) 0 MAG DROP
    12 ERASE
  9 6 ARRAY POSITION ( CLEAR IT) 0 POSI-
    TION DROP 12 ERASE
 10
 11 TABLE UPLIMIT 3000 , 1875 , 562 , 7200 ,
    1800 , 0 ,
 12 TABLE LOWLIMI -3000 , -1875 , -937 ,
    -7200 , -1800 , -2000 ,
 13 TABLE PITCH 4 , 4 , 8 , 5 , 5 , -5 ,
 14 -->
 15
SCR # 43
  0 ( ROBOT   ARROT 2   82-12-9)
  1
  2 : STRING-ARRAY ( DEFINE STRING ARR
    AY : MAX LENGTH=12)
  3 (N -> ADDR : IN EXEC MODE)
  4 <BUILOT 12* ALLAY DOES> SWAP 12* +;
  5
  6 20 CONSTANT MAX. #CMD (MAX # OF
    CMD STRING)
  7 MAX. #CMDSTRING-ARRAY CMD-STRI
    NG
  8
  9 -->
 10
 11
 12
 13
 14
 15
SCR # 44
  0 (ROBOT   MAKE, SEND   82-12-9)
  1
  2 : MAKE (N -> : MAKE CMD STRING AFT
    ER PAD)
  3 S->D SWAP OVER DABS
  4 <# 44 ( ,) HOLD #S SIGN #> ( -> ADR,
    CNT)
  5 DUP CMDHLD @ + >R
  6 CMDHLD @ SWAP CMOVE
  7 R> [O] CMDHLD [NI] ! ;
  8 : SEND ( -> : SEND CMD)
  9 CMDHLD @ PAD - (CHR#) PAD 1 + SW
    AP ( ->ADDR, BYTES) LPOUT ;
 10 DECIMAL -->
 11
 12
 13
 14
 15
SCR #45
  0 (ROBOT   I-CMD   82-12-9)
  1 : I-CMD ( -> : CREATE I CMD)
  2 PAD 1+ CMDHLD ! (INIZ. CMDHLD)
  3 73 ( I CMDHLD @ C ! 1 CMDHLD +! ( INC
    CMDHLD)
  4 6 0 DO I MAG SWAP DROP MAKE LOOP
  5 0 MAG DROP 12 ERASE (CLEAR MAG
  6 13 CMDHLD @ 1- C ! (ADD CR)
  7 10 CMDHLD @ C ! (ADD LF) SEND !
  8
  9 HEX
 10 : 3LPOUT 0A0E (CR/LF) PAD 2+ ! PAD 1+
    ADDR.) 3 (BYTES)
 11 LPOUT !
 12 : SET-HOME (H COMMAND : SET HOSITI
    ON)
 13 48 (H PAD 1+ C ! 3 LPOUT ;
 14 : HOME (N COMMAND : RETURN TO HO
    ME POSITION)
 15 4E (N) PAD 1+ C ! 3LPOUT ; DECIMAL-->
SCR # 46
  0 (ROBOT [RELROT], RELROT, RELROT-IM
    82-12-9)
  1 : (RELROT) (-> : )
  2 JOINT @ >R
  3 R POSITION SWAP DROP [NI] STEPS ∂
    [O]+[O] TEMP [NI] !
  4 TEMP @ [GA] R UPLIMIT [YORI]>
```

```
 5 IF R UPLIMIT [KARA] STEPS @ [O] - [O]
   STEPS!
 6 ELSE TEMP @ [GA R LOWLIMIT [YORI] <
 7 IF R LOWLIMIT [KARA] STEPS @ [O] -
   [O] STEPS!
 8 THEN THEN
 9 STEPS @ [O] R> POSITION DROP [NI] + !;
10 : RELROT (-> : -----RELATIVE ROTATION
   -----)
11 JOINT @ MAG IF I-CMD THEN (RELROT)
12 STEPS @ [O] JOINT @ MAG DROP [NI]!;
13 : RELROT-IM (-> : ----- RELATIVE ROTA
   TE IMMEDIATELY-----)
14 (RELROT) STEPS @ [O] JOINT @ MAG
   D ROP [NI]! I-CMD;
15 DECIMAL -->
SCR # 48
 0 (ROBOT   DEGREE, WAIST,..., HAND 82-
   12-9)
 1 : DEGREE (N->N : CONVERT DEGREE UN
   IT TO ABSOLUTE STEPS)
 2 100 [O] JOINT @ PITCH */ ;
 3
 4 : WAIST          0 JOINT! ;
 5 : SHOULDER       1 JOINT! ;
 6 : ELBOW          2 JOINT! ;
 7 : WRIST          3 JOINT! ;
 8 : HAND           5 JOINT! ;
 9
10
11 -->
12
13
14
15
SCR # 49
 0 (ROBOT   ROTAE, BEND   82 12-9)
 1
 2 : ROTATE (N-> : )
 3 STEPS!
 4 ROTATE-MODE @ IF ABSROT ELSE REL
   ROT THEN!
 5
 6 : BEND (WRIST BEND)
 7 JOINT @ 3=IF 4 JOINT! ROTATE THEN!
 8 : TWIST WAIST ROTATE;
 9 : UP/DOWN SHOULDER ROTATE;
10 : SWING ELBOW ROTATE;
11 : GRASP -2000 HAND ROTATE;
12 : RELEASE 2000 HAND ROTATE;
13 -->
14
15
```

Fig. 4 Program to control micro robot

とする事もできる。これはTWISTというワードはWAIST ROTATEと同じであると定義しているに過ぎないが，より自然で理解しやすくなる。同様に，各関節の動作を適切に表現するワードとして，UP，DOWN，SWING，RELEASE等を定義することができる。

このようにしてマイクロ・ロボットはわずか10数語ではあるが，独自の語彙と文法を解する事が可能となった。これらを基礎として語彙を増やして，複雑な動作を制御することができる。

3-3 Teaching Playback

ロボットに一連の動作をさせるのに最も基本的な方法は，先ず手動でロボットを所定の場所に移動し，その位置を記憶させ (teaching)，その後記憶させた通り動かせる (play baclk) ものである。その為のワードとして2つを追加した。(Fig.5) 先ずTEACHは手動操作で各関節を移動させ，必要な位置を記憶させる事を可能にする。PLAY—BACKはTEACHによりCMD—STRINGというワード内に畜えられたデータを順次取り出して実行する。

各ワードの動作内容に比して，そのプログラムのコンパクトな点が注目される。これはFORTH系の言語においては，以前に定義されたワードの組み合せで，新しいワードを定義するという方式になっているからである。

```
SCR # 52
 0 (ROBOT  PLAY BACK 1  MANUAL-MOVE
   83-3-22)
 1 : TEACH
 2 CR CP "START MANUAL MOVE !" CR
 3 0  CMD-STRING  12 ERASE  SET-HOME
   (CLEAR 0-TH CMD& SET HOME POS)
 4 1 #CMD ! (INIZ.)
 5 0 POSITION DROP 12 ERASE
 6 0 MAG DROP 12 ERASE
 7 BEGIN @GET >R R 69 (E)-WHILE
 8 R 13(CR)=IF  #CMD @ CR. CMD-STORE
   ELSE
 9 R 49 (1)=IF WAIST -1 DEGREE STEPS !
   RELROT-IM ELSE
10 R 50 (2) =IF WAIST 1 DEGREE STEPS !
```

```
    RELROT-IM ELSE
 11 THEN THEN THEN
 12 R> DROP REPEAT
 13 R> DROP HOME CR, "END OF CMD" ;
    DECIMAL
 14 -->
 15
SCR # 53
  0 (ROBOT    PLAY  BACK 2 PLAY  BACK
    83-3-22)
  1
  2 DECIMAL
  3 : PLAY-BACK (->)
  4 CR CR. " ** START PLAY BACK **" CR
  5 #CMD @  0 DO I  CMD-STRING 0 MAG
    DROP 12  CMOVE I-CMD LOOP ;
  6 DECIMAL
  7 ;S
  8
  9
 10
 11
 12
 13
 14
 15
OK
```

Fig. 5 Program for teaching-playback

3-4 位置制御

ロボットの適切な制御のためには，ハンド先端位置の制御が不可欠である。[4] Fig. 6に示すように，各リンクの重心に固定した座標をとり，これらの座標系の単位ベクトルを

$$\vec{e}_{ji} \qquad \begin{matrix} i=1,\ 2,\ 3 \\ j=1,\ \cdots,\ 5\ (\text{リンク番号}) \end{matrix}$$

とする。隣接するリンク間の座標変換行列をAiとすると

$$\vec{e}_{ji} = Ai \quad \vec{e}_{j-1,i}$$

ハンド先端位置$\vec{r}$は

$$\vec{r} = \sum_{k=1}^{4} \ell_k \vec{e}_{k3} + \sum_{p=1}^{3} (\ell_{5p} + x_{5p}) \vec{e}_{5p}$$

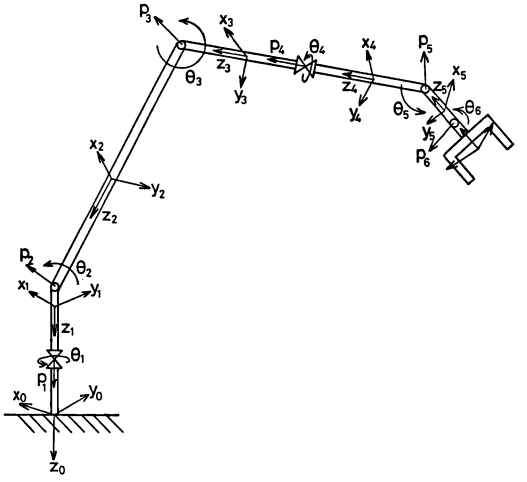

Fig. 6 Schematic diagram of joints

と支えられる。ℓ_k はリンクkの長さ，ℓ_{5p} はリンク5の座標系からみた関節の位置，x_{5p} はハンドの位置である。座標変換行列は

$$A_1 = Az\ (\theta_1)$$
$$A_2 = Ag\ (\theta_2)$$
$$A_3 = Ag\ (\theta_3)$$
$$A_4 = Ag\ (\theta_4)$$
$$A_5 = Az\ (\theta_5)$$
$$Ag(\theta) = \begin{pmatrix} \cos\theta & o & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$
$$Az(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$$

である。ここではベクトル及びマトリックスというデータ型式が必要となるが，元来ＦＯＲＴＨには

スカラ型式しかないので,新しくベクトル及びマトリックスを定義する為のワードをFig. 7に示す。[5] FORTHにおいては新しいワードを定義出来るのみならず，ワード群を定義する為のワードという一段階高いレベルのワードを持つ事ができる。

```
3 1   ARRAY    X
3 3   ARRAY    Y
```

とする事によりXは3成分ベクトルをYは3×3の行列を定義する。

```
SCR # 61
 0 (ARRAY DEFINITIONS   ARRAY)
 1
 2 : ARRAY
 3 <BUILDS OVER , * 2* ALLOTDOES>
 4 DUP @ ROT * 2* + SWAP 2* + 2+;
 5
 6 ; S
 7
 8 IN DEFINITION : [ROW#] [COOL#]
   ARRAY [NAME]
 9 IN REFEERENCE : [ROW#] [COL#] [NA
   ME] (ADDRESS ON STACK)
10 INDEX STARTS FROM O TO [N] -1
11
12
13
14
15
```

Fig.7 Definition of definig word : ARRAY

又，三角関数については，浮動小数点型式を持たないため，その値の10000倍にスケールアップした整数値で相対誤差0.1％以内の精度で計算させる事が可能である。[6] (Fig. 8)

位置制御の為の計算にはかなりのステップを要するので，リアルタイム処理には速度的に制限が厳しくなる。

```
SCR # 62
 0 (SCALED INTEGER SINE  FUNCTION)
 1 10000 CONSTANT 1OK (SCALING CONST
   ANT)
 2 VARIABLE XS (SQUARE OF SCALED AN
   GLE)
 3
 4 (AB---M : M=10000-AX*X/B... COMMON
   TERM IN SERIES)
 5 : L XS @ SWAP / MINUS 1OK */ 1OK + ;
 6
 7 (THETA --- 10000*SIN : -15708<THETA
   <15708 RAD*10000)
 8 : (SIN) DUP (SAVE X) DUP 1OK */ XS !
   (SAVE)
 9 1OK (PUT 1*10000 ON STACK TO START
   SERIES)
10 72 L 42 L 20 L 6 L (SUM SERIES)
11 10K */ (FINALY MULTIPLY BY SAVED
   X);
12
13 -->
14
15
SCR # 63
 0 (SIN, COS, TAN FOR WHOLE ANGLE
   [DEGREE UNIT] )
 1
 2 (THETA --- THETA : REDUCE Y-AXIS
   SYMMETRY ; 0-180 -> 0-90-0)
 3 : ?MIRROR DUP 90 > IF 180 SWAP-THEN;
 4
 5 (THETA [ANY] --- THETA [-90 TO 90]
   : ANGLE RANGE REDUCTION)
 6 : REDUCE 360 MOD DUP 0< IF 360+THEN
   DUP 180 <
 7 IF (0-180) ?MIRROR ELSE 180-?MIRROR
   MINUS THEN ;
 8
 9 (THETA --- 10000*SIN ETC. ANGLE IN
   DEGREE)
10 : SIN REDUCE 17453 100 */ (DEG. TO RAD
   *10000) (SIN) ;
11 : COS 360 MOD (PREVENT POSSIBLE OV
   ERFLOW) 90 SWAP-SIN;
12
13 (THETA --- 100*TAN : TAN SET +-30000
   FOR +-90 DEG)
14 : TAN DUP SIN SWAP COS ?DUP IF 100
   SWAP */ ELSE 3* THEN;
15 ; S
```

Fig.8 Words to calcurate trigonometric functions

4. まとめ

機器制御分野のマイクロセッサーの今後の広がりを考えると，それらを制御するプログラムは，できるだけ自然言語に近くかつ，きめ細い表現が必要となる。ここではＦＯＲＴＨ系言語の豊富な造語機能に着目し，それを基として機器制御言語の構築を試みた。

現在はまだ開発途上ではあるが，その基本的な特徴は見る事ができる。試みに全く予備知識のない者に，マイクロ・ロボット制御用のワードとその文法を示して実際に操作させた所，容易に操作を実行できた。これは従来の汎用言語を用いては得る事のできない特徴であり，この事からも制御対象に合致した言語系の必要性を示すものである。

今後は更にこの体系を整備，充実し，実用に耐え得るものに仕上げる予定である。

なおこの研究は「昭和57年度産業研究所特別研究費」の助成を受けたものであることを附記し，感謝の意を表します。

参考文献

1) Moore, C.H., BYTE, No. 80/4, (1980)
2) 原，マイコンピューク，V3,(1981)
3) 三菱電気，RM-101操作説明書
4) 井上博允他，情報処理学会マン・マシンシステム研究会資料 75-22,(1975)
5) Brodie, L., Staring FORTH, (1981), Prentice-Hall
6) Bumgarner, J., FORTH DIMENSIONS, V4, N1, (1982)