บทที่ ๓

การวิเคราะห์ขั้นตอนการทำงานของอินเตอร์พรีเตอร์ภาษาเบสิค

คำสั่งในภาษาเบสิค[๑] มีทั้งได้เรคโมดและอินไดเรคโมด ดังได้กล่าวมาแล้วในบทที่ ๒ ฉะนั้น ในบทนี้จะกล่าวคำสั่งที่ใช้ในแต่ละโมด ส่วนผังงานของโปรแกรมหลักแสดงไว้ในภาคผนวก ข.

๓.๑ คำสั่งที่ใช้ในไดเรคโมด

๓.๑.๑ CLOAD เป็นคำสั่งที่ใช้โหลดโปรแกรมจากเทปคาสเซท เข้ามาในหน่วยความจำ เมื่อคำสั่งนี้ทำงาน จะทำการค้นหาชื่อโปรแกรมในเทปคาสเซท เมื่อพบแล้วจะทำการโหลดโปรแกรมดังกล่าวเข้ามาในหน่วยความจำ ในการค้นหาชื่อโปรแกรมในเทปคาสเซทนั้น ถ้าพบชื่อโปรแกรมแต่ไม่ใช่ชื่อโปรแกรมที่ต้องการ จะแสดงคำว่า SKIP และชื่อโปรแกรมที่ไม่ต้องการนั้นขึ้นบนจอภาพ แต่ถ้าพบชื่อโปรแกรมที่ต้องการ จะปรากฏคำว่า FOUND ขึ้นบนจอภาพ รูปแบบของคำสั่งแสดงไว้ในรูปที่ ๓.๑ และขั้นตอนการทำงานของคำสั่งนี้ได้แสดงไว้ในผังงานที่ ๓.๑

Syntax

CLOAD "file name"

Examples

CLOAD "TEST1"	The program TEST1 is searched for on tape and loaded into memory.

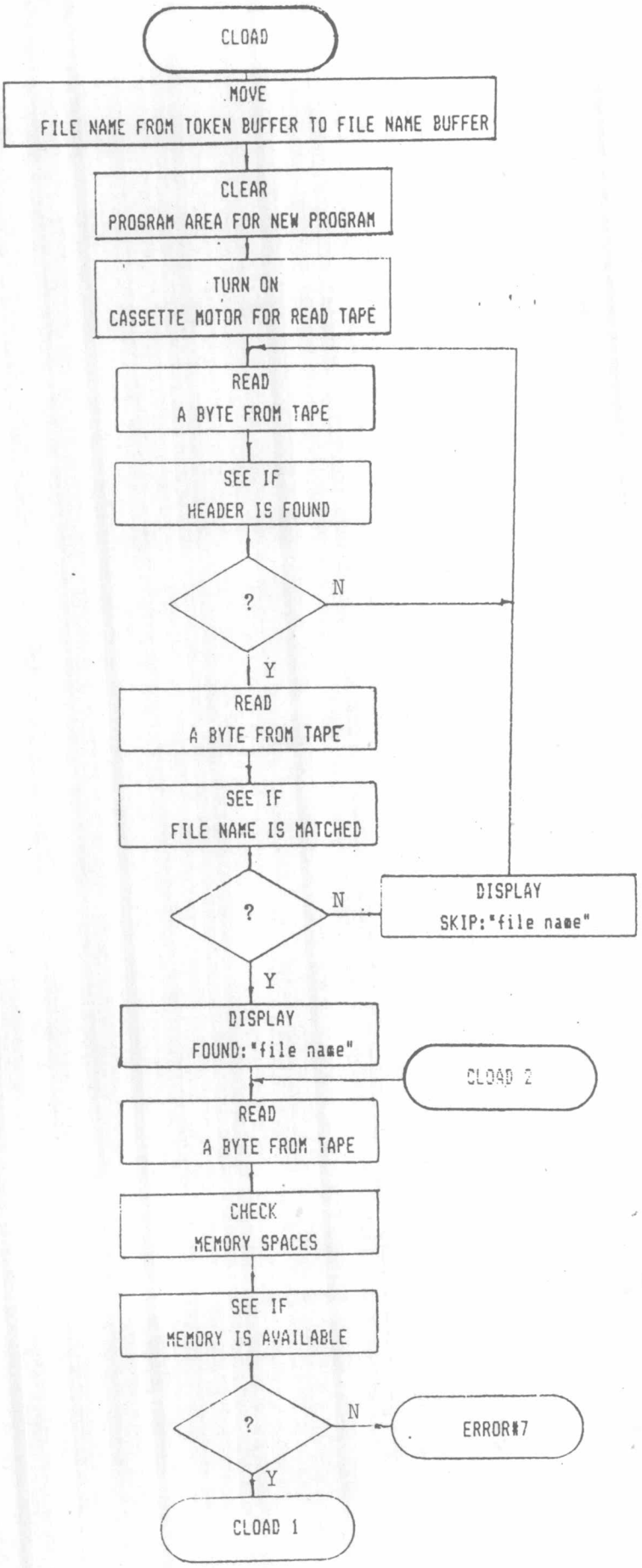
รูปที่ ๓.๑ แสดงรูปแบบของคำสั่ง CLOAD

---

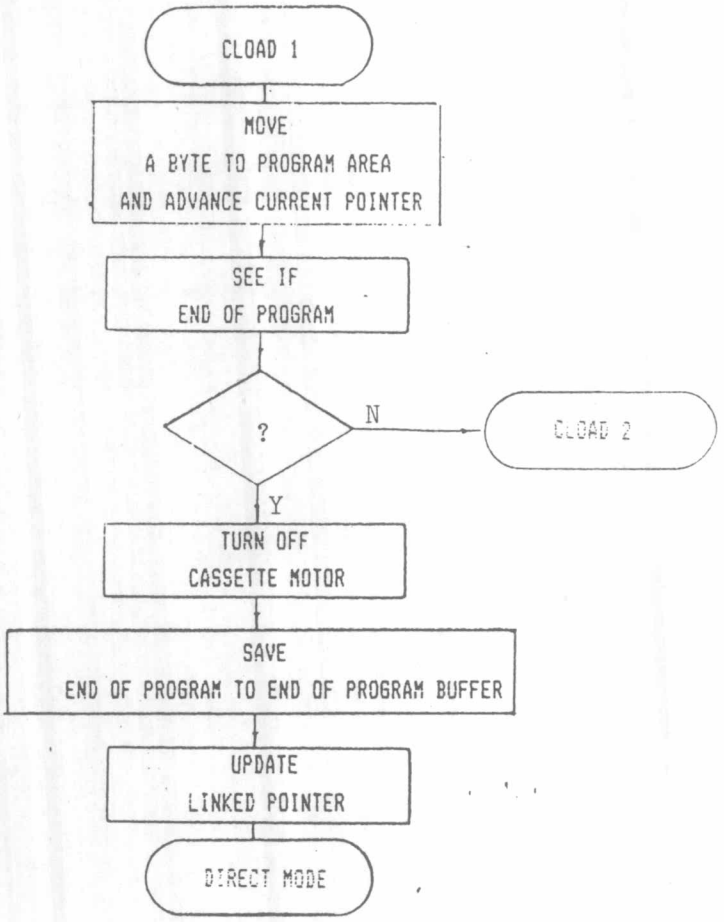
[๑] Nippon Electric Co., Ltd., PC-8001B N-Basic Reference Manual (Nippon Electric Co., Ltd., 1981), p. 2-1-2-64.

```
10 CLOAD"Prog00"
100 ' CLOAD Statement


8020 00 2F 80 0A 00 9B 22 50 72 6F 67 30 30 22 00 47 ./....."Prog00".G
8030 80 64 00 3A 8F E4 20 43 4C 4F 41 44 20 53 74 61 .d.:.. CLOAD Sta
8040 74 65 6D 65 6E 74 00 00 00 FF FF FF FF FF FF FF tement.........
8050 FF
```

รูปที่ ๓.๒ แสดงการเก็บรูปแบบของคำสั่ง CLOAD ในหน่วยความจำ



ผังงานที่ ๓.๑ แสดงขั้นตอนการทำงานของคำสั่ง CLOAD



ผังงานที่ ๓.๑ แสดงขั้นตอนการทำงานของคำสั่ง CLOAD (ต่อ)

๓.๑.๒ CSAVE เป็นคำสั่งที่ใช้นำโปรแกรมจากหน่วยความจำไปเก็บไว้ในเทปคาสเซท มีรูปแบบดังรูปที่ ๓.๓

Syntax

CSAVE "file name"

Example

CSAVE "PROB1" Stores the current file "PROB1" on cassette tape.

รูปที่ ๓.๓ แสดงรูปแบบของคำสั่ง CSAVE

```
10 CSAVE"Prog00"
100 ' CSAVE Statement

8020 00 2F 80 0A 00 9A 22 50 72 6F 67 30 30 22 00 47 ./....."Prog00".G
8030 80 64 00 3A 8F E4 20 43 53 41 56 45 20 53 74 61 .d.:.. CSAVE Sta
8040 74 65 6D 65 6E 74 00 00 00 64 FF FF FF FF FF FF tement...d......
8050 FF
```

รูปที่ ๓.๔ แสดงการเก็บรูปแบบของคำสั่ง CSAVE ในหน่วยความจำ

การเก็บโปรแกรมไว้ในเทปคาสเซท มีรายละเอียดในการเก็บดังนี้ (ดูรูปที่ ๓.๕ และ ๓.๖)

๓.๑.๒.๑ หัวโปรแกรม (Header) ใช้รหัส OD3H จำนวน ๑๐ ไบท์

๓.๑.๒.๒ ชื่อโปรแกรม (Program name) ใช้เนื้อที่ ๖ ไบท์

๓.๑.๒.๓ ตัวโปรแกรม (Program statements) โปรแกรมจากหน่วยความจำที่มีแอดเดรสเริ่มต้นที่ 8020H

๓.๑.๒.๔ เครื่องหมายแสดงการหมดโปรแกรม (End Flag) ใช้รหัส ๐๐ จำนวน ๑๐ ไบท์

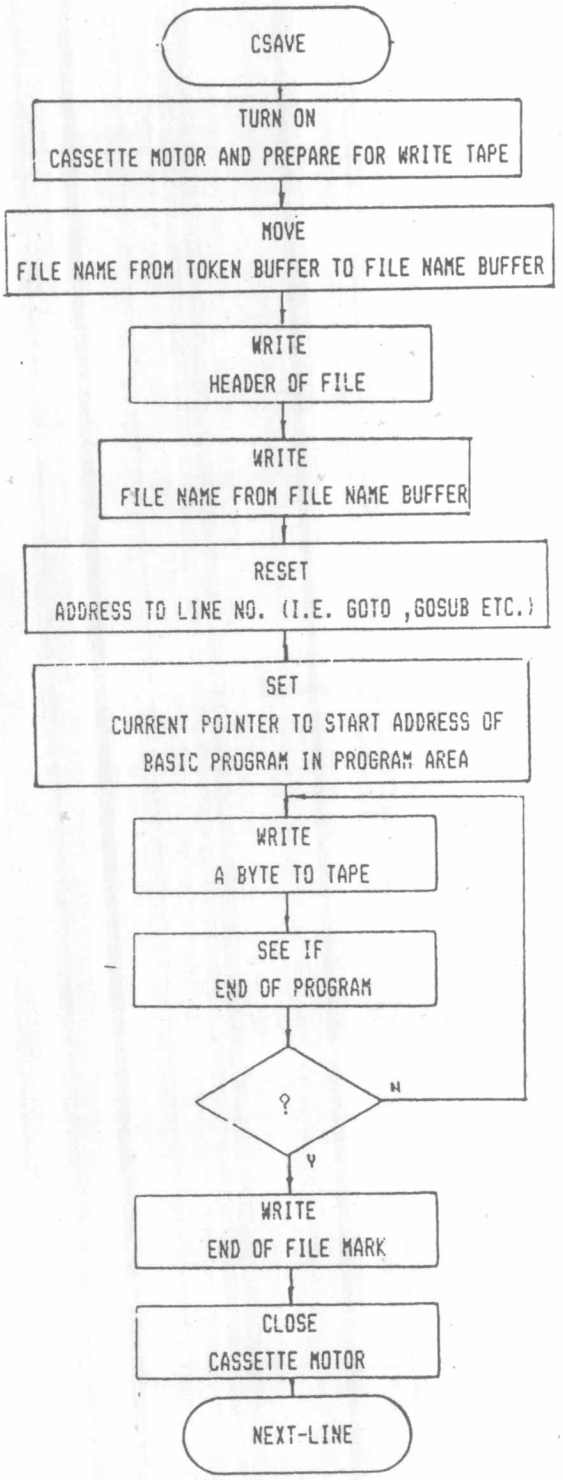
ก่อนหัวโปรแกรม จะมีตัวอักษรนำ (Leader) เพื่อรอให้เทปหมุนด้วยความเร็วคงที่ และซิงค์ไบท์ (Sync. byte) เพื่อเป็นเครื่องหมายแสดงว่าเริ่มเขียนหัวโปรแกรมลงไป นอกจากนี้ยังใช้เป็นเครื่องหมายแสดงการเริ่มอ่านหัวโปรแกรมเมื่อใช้คำสั่ง CLOAD ขั้นตอนการทำงานของ CSAVE แสดงไว้ในผังงานที่ ๓.๒

```
10 '   DATA REPRESENT IN TAPE
20 '
30 INPUT"DATA= ";A,B%,C#,D$
40 PRINT #-1,A,B%,C#,D$
50 END
```

รูปที่ ๓.๔ โปรแกรมที่ต้องการเก็บไว้ในเทปคาสเซท

```
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 ................
80 D3 D3 D3 D3 D3 D3 D3 D3 D3 D3 50 47 4E 41 4D ...........PGNAM
45 42 80 0A 00 3A 8F E4 20 20 20 44 41 54 41 20 EB...:..   DATA
52 45 50 52 45 53 45 4E 54 20 49 4E 20 54 41 50 REPRESENT IN TAP
45 00 4A 80 14 00 3A 8F E4 00 63 80 1E 00 85 22 E.J...:...c....*
44 41 54 41 3D 20 22 3B 41 2C 42 25 2C 43 23 2C DATA= ";A,B%,C#,
44 24 00 78 80 28 00 91 20 23 F4 12 2C 41 2C 42 D$.x.(.. #..,A,B
25 2C 43 23 2C 44 24 00 7E 80 32 00 81 00 00 00 %,C#,D$.~.2.....
00 00 00 00 00 00 00 00 00 01 01 01 01 01 01 01 ................
```

รูปที่ ๓.๖ แสดงการเก็บโปรแกรมในเทปคาสเซท



ผังงานที่ ๓.๒ แสดงขั้นตอนการทำงานของคำสั่ง CSAVE

๓.๑.๓ DELETE เป็นคำสั่งที่ใช้ในการลบบรรทัดของโปรแกรมที่ไม่ต้องการออก มีรูปแบบดังรูปที่ ๓.๗ ขั้นตอนการทำงานแสดงไว้ในผังงานที่ ๓.๓

Syntax

DELETE line number [–to line number]

Examples

DELETE 40	Deletes line 40.
DELETE 40-70	Deletes lines 40 through 70.
DELETE-40	Deletes all lines up to and including 40.
DELETE	"ILLEGAL FUNCTION CALL". You must specify a line number. The same error occurs if you specify a line that does not exist.

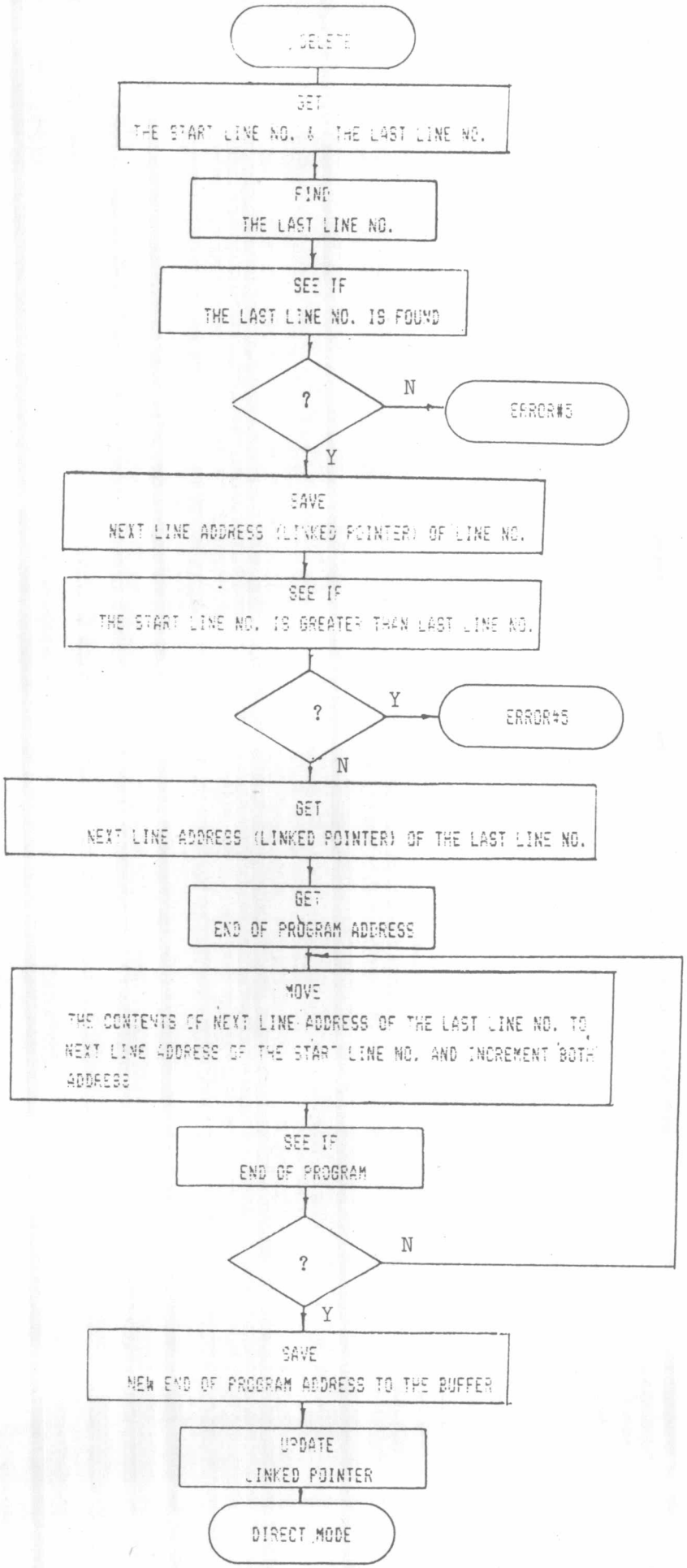
Note

You can use a comma (,) in place of the dash (-).

รูปที่ ๓.๗ แสดงรูปแบบของคำสั่ง DELETE

```
10 DELETE 10-100
100,'DELETE Statement

8020 00 2F 80 0A 00 A8 20 0E 0A 00 F4 0E 64 00 00 47 ./.... .....d..G
8030 80 64 00 3A 8F E4 44 45 4C 45 54 45 20 53 74 61 .d.:..DELETE Sta
8040 74 65 6D 65 6E 74 00 00 00 64 FF FF FF FF FF FF tement...d......
8050 FF
```

รูปที่ ๓.๘ แสดงการเก็บรูปแบบของคำสั่ง DELETE ในหน่วยความจำ



ผังงานที่ ๓.๓ แสดงขั้นตอนการทำงานของคำสั่ง DELETE

๓.๑.๔ END เป็นคำสั่งให้โปรแกรมหยุดการปฏิบัติงาน สามารถใส่คำสั่งนี้ไว้ในส่วนไหนของโปรแกรมก็ได้ มีรูปแบบดังรูปที่ ๓.๙ ขั้นตอนการทำงานแสดงไว้ในผังงานที่ ๓.๔

Syntax

END

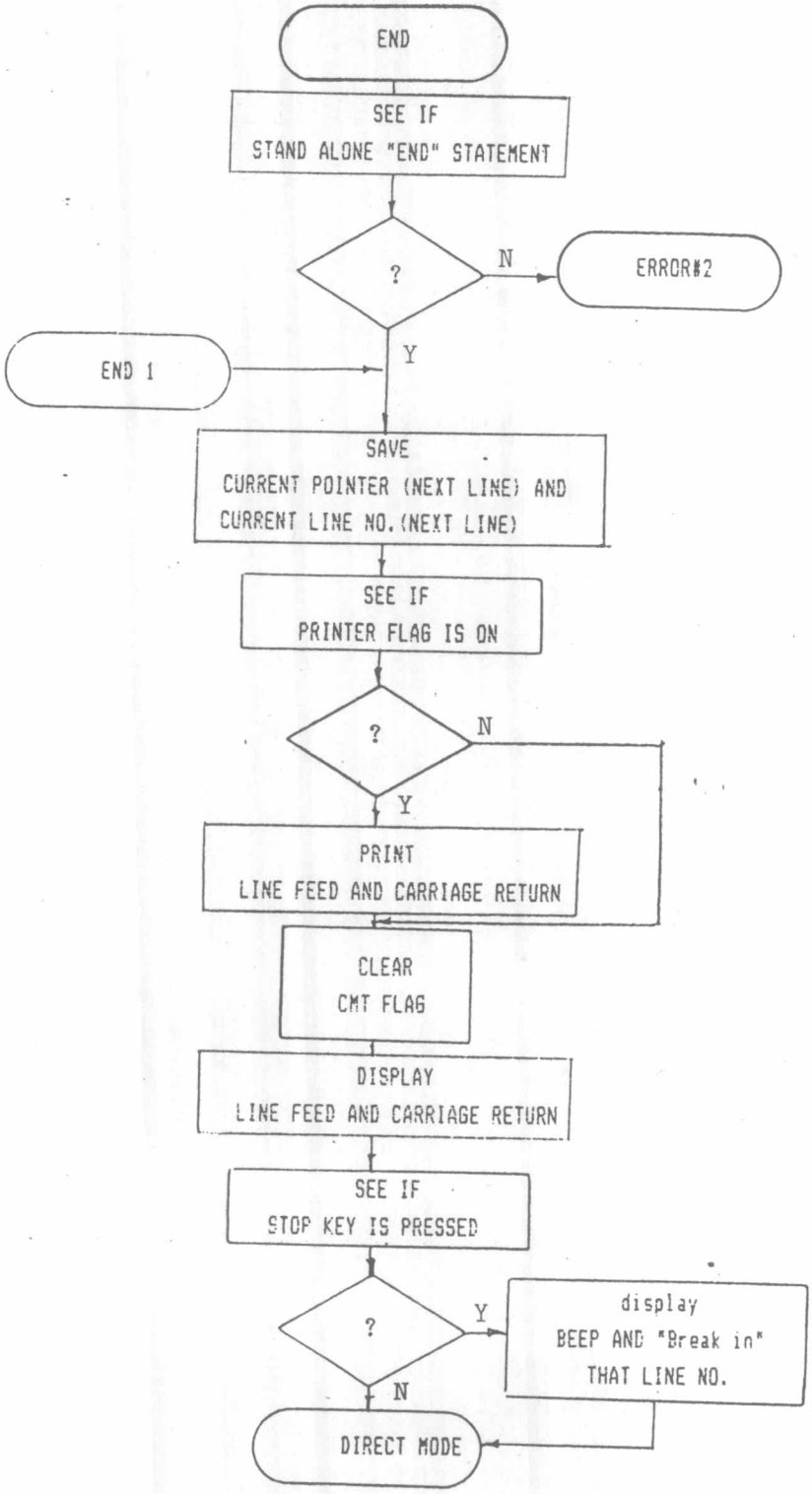
Examples

```
10 X=X+1
20 PRINT X
30 IF X>3 THEN END ELSE GOTO 10
run
 1
 2
 3
 4
ok
```

รูปที่ ๓.๙ แสดงรูปแบบของคำสั่ง END

```
10 END
100 ' END Statement

8020 00 27 80 0A 00 81 00 3D 88 64 00 3A 8F E4 20 45 .'.....=.d.:.. E
8030 4E 44 20 53 74 61 74 65 6D 65 6E 74 00 00 00 64 ND Statement...d
8040 FF
```

รูปที่ ๓.๑๐ แสดงการเก็บรูปแบบของคำสั่ง END ในหน่วยความจำ



ผังงานที่ ๓.๔ แสดงขั้นตอนการทำงานของคำสั่ง END

๓.๑.๕ LIST เป็นคำสั่งที่ใช้แสดงบางส่วนหรือทั้งหมดของโปรแกรมที่อยู่ในหน่วยความจำ มีรูปแบบดังรูปที่ ๓.๑๑ การแสดงผลจะแสดง ในจอภาพ ถ้าต้องการแสดงผลทางเครื่องพิมพ์ (Printer) ให้ใช้คำสั่ง LLIST แทนคำสั่ง LIST รูปแบบของคำสั่ง LLIST จะเหมือนกับคำสั่ง LIST ทุกประการ ขั้นตอนการทำงานของคำสั่ง LIST และ LLIST แสดงไว้ในผังงานที่ ๓.๕

Syntax

LIST [line number] [{-,} line number]

Examples

LIST	Lists the entire program currently in memory.
LIST 500-	Lists line 500.
LIST 150-	Lists all lines from line 150 to the end of the program.
LIST -150	Lists all lines from the beginning of the program to line 150.
LIST 50-100	Lists all lines from 50 to 100 inclusive.
LIST.	Lists a line that caused an error and halted program execution.

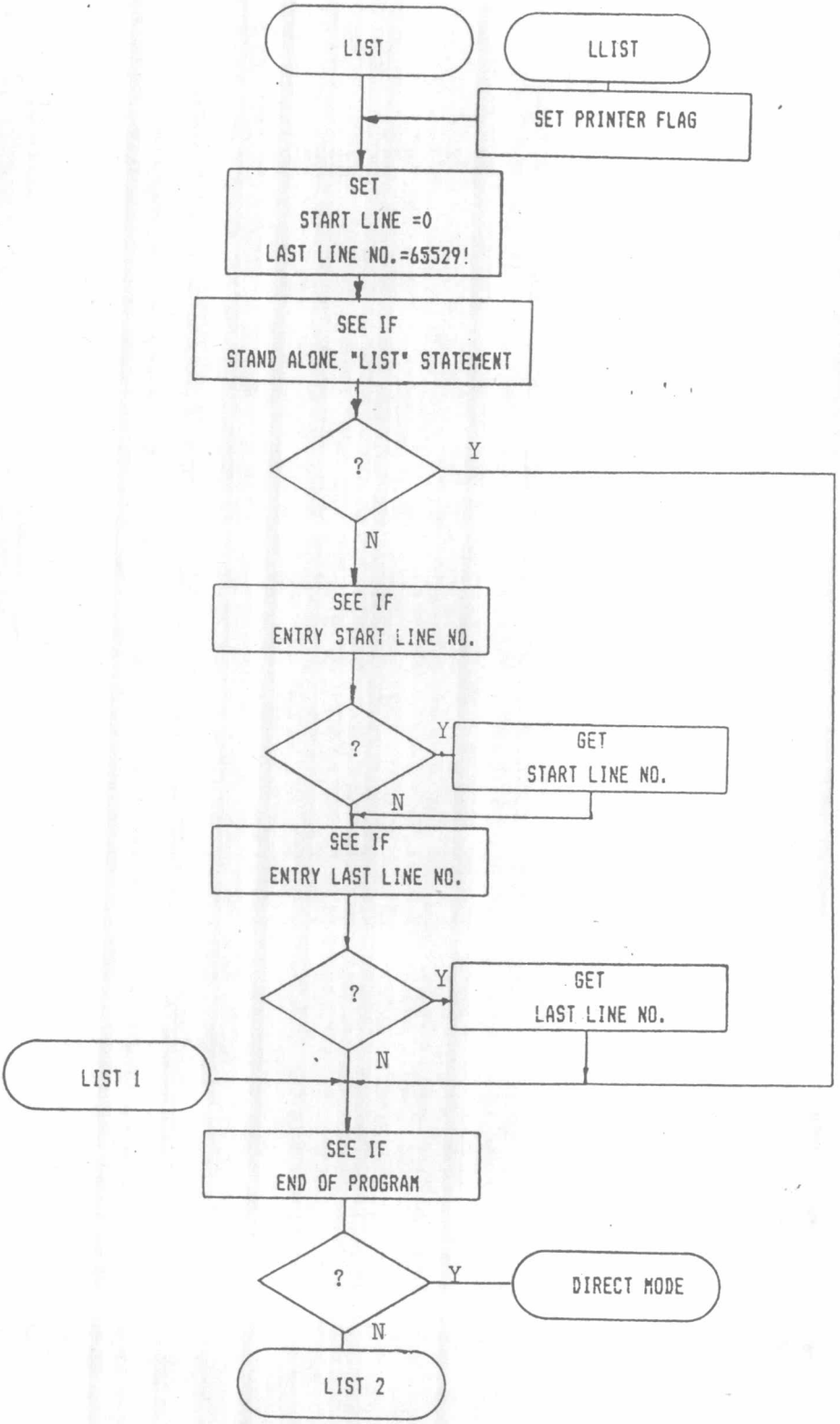
Note

A comma (,) can be substituted for a hyphen (-).

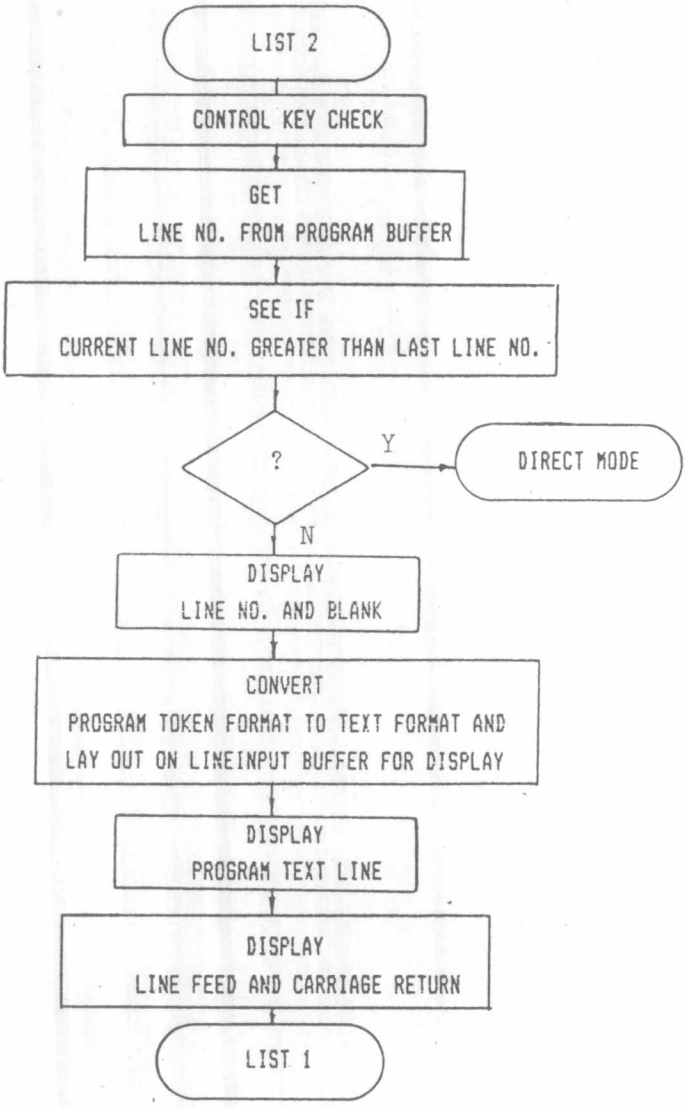
รูปที่ ๓.๑๑ แสดงรูปแบบของคำสั่ง LIST

```
10 LIST 100
100 ' LIST Statement

8020 00 2B 80 0A 00 93 20 0E 64 00 00 42 80 64 00 3A .+.... .d..B.d.:
8030 8F E4 20 4C 49 53 54 20 53 74 61 74 65 6D 65 6E .. LIST Statemen
8040 74 00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF t...............
8050 FF
```

รูปที่ ๓.๑๒ แสดงการเก็บรูปแบบของคำสั่ง LIST ในหน่วยความจำ



ฝังงานที่ ๓.๕ แสดงขั้นตอนการทำงานของคำสั่ง LIST



ผังงานที่ ๓.๕ แสดงขั้นตอนการทำงานของคำสั่ง LIST (ต่อ)

๓.๑.๖ NEW เป็นคำสั่งใช้ลบโปรแกรมที่อยู่ในหน่วยความจำทั้งหมด ใช้คำสั่งนี้เพื่อลบโปรแกรมเดิมในหน่วยความจำก่อนจะนำโปรแกรมใหม่ เข้ามาในหน่วยความจำ รูปแบบของคำสั่งนี้แสดงไว้ในรูปที่ ๓.๑๓ ขั้นตอนการทำงานแสดงไว้ในผังงานที่ ๓.๖

Syntax

NEW

Example

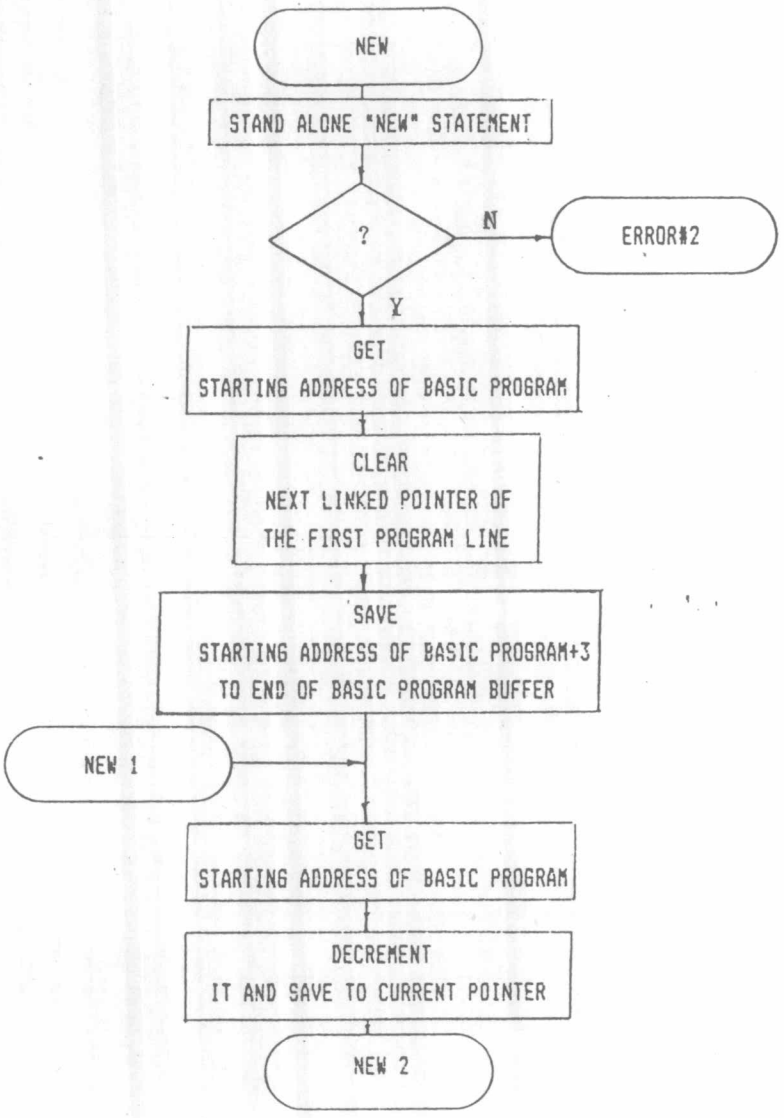
NEW

NEW deletes the program currently in memory and clears all variables. Use NEW to clear memory before entering a new program.

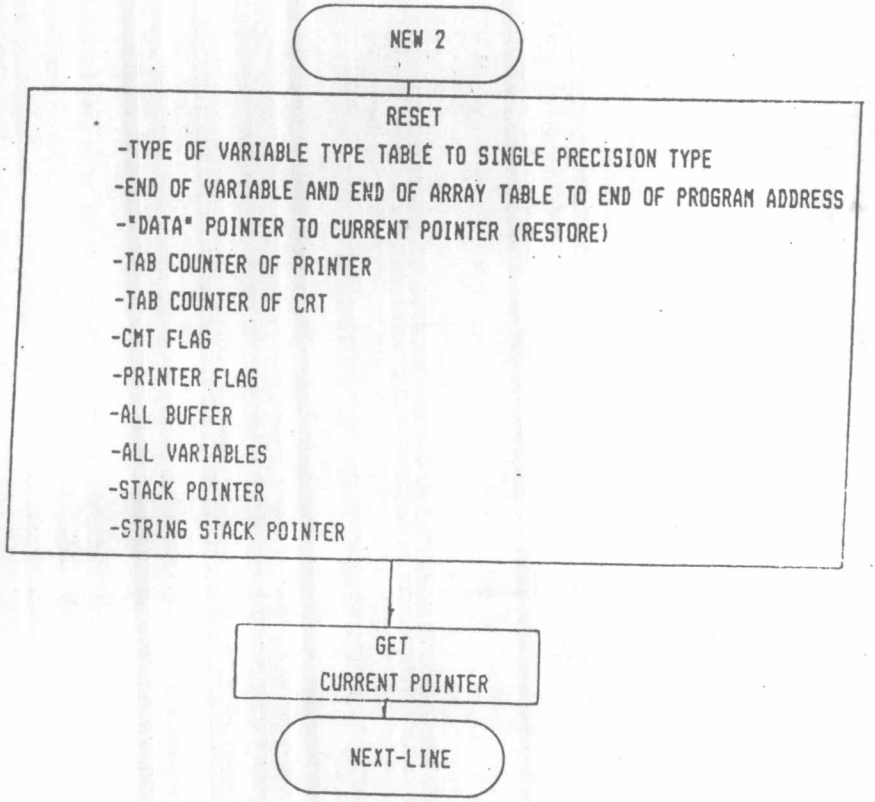
รูปที่ ๓.๑๓ แสดงรูปแบบของคำสั่ง NEW

```
10 NEW
100 ' NEW Statement

8020 00 27 80 0A 00 94 00 3D 80 64 00 3A 8F E4 20 4E .'.....=.d.:.. N
8030 45 57 20 53 74 61 74 65 6D 65 6E 74 00 00 00 FF EW Statement....
8040 FF FF
```

รูปที่ ๓.๑๔ แสดงการเก็บรูปแบบของคำสั่ง NEW ในหน่วยความจำ



ผังงานที่ ๓.๖ แสดงขั้นตอนการทำงานของคำสั่ง NEW



ผังงานที่ ๓.๖ แสดงขั้นตอนการทำงานของคำสั่ง NEW (ต่อ)

๓.๑.๗ RUN เป็นคำสั่งที่ใช้ปฏิบัติการทำงาน โปรแกรมที่อยู่ในหน่วยความจำ มีรูปแบบดังรูปที่ ๓.๑๕ ขั้นตอนการทำงานแสดงไว้ในผังงานที่ ๓.๗

Syntax

RUN

Examples

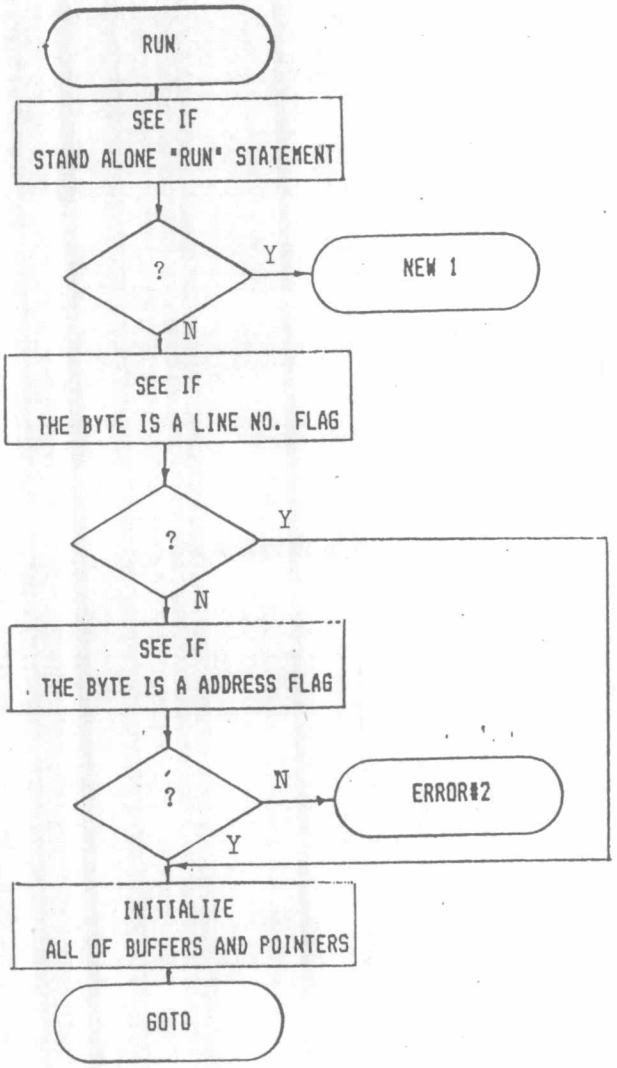
RUN 100	Begins execution of the program currently in memory at line 100.

Use RUN to execute a program currently in memory, to start program execution at a specific line, or to load a program from disk into memory and execute it.

รูปที่ ๓.๑๕ แสดงรูปแบบของคำสั่ง RUN

```
10 RUN 100
100 ' RUN Statement

8020 00 2B 80 0A 00 8A 20 0E 64 00 00 41 80 64 00 3A .+.... .d..A.d.:
8030 8F E4 20 52 55 4E 20 53 74 61 74 65 6D 65 6E 74 .. RUN Statement
8040 00
```

รูปที่ ๓.๑๖ แสดงการเก็บรูปแบบของคำสั่ง RUN ในหน่วยความจำ



ผังงานที่ ๓.๗ แสดงขั้นตอนการทำงานของคำสั่ง RUN

๓.๒ คำสั่งที่ใช้ในอินไดเรคโมด

๓.๒.๑ DATA เป็นคำสั่งที่ใช้ในการจัดข้อมูลให้กับคำสั่ง READ มีรูปแบบดังแสดงไว้ในรูปที่ ๓.๑๗ ขั้นตอนการทำงานเมื่อพบคำสั่งนี้ อินเตอร์พรีเตอร์จะข้ามไปคำสั่งต่อไปโดยไม่มีการปฏิบัติการ คำสั่งนี้จะมีบทบาทเมื่อคำสั่ง READ ถูกปฏิบัติงาน

Syntax

DATA constant,constant . . .

Example

```
10 REM PROGRAM TO READ LIST
20 REM OF VARIABLES AND PRINT
30 REM THEM OUT.
40 FOR I=1 TO 3
50 READ A,B,C
60 PRINT A;B;C
70 NEXT I
80 DATA 1,2,3
90 DATA 4,5,6
100 DATA 7,8,9
110 END
RUN
1  2  3
4  5  6
7  8  9
```

รูปที่ ๓.๑๗ แสดงรูปแบบของคำสั่งDATA

```
10 DATA 12,145,Hello
20 'DATA Statement
```

```
8020 00 34 80 0A 00 84 20 31 32 2C 31 34 35 2C 48 65 .4.... 12,145,He
8030 6C 6C 6F 00 4A 80 14 00 3A 8F E4 44 41 54 41 20 llo.J...:..DATA
8040 53 74 61 74 65 6D 65 6E 74 00 00 00 00 00 00 00 Statement.......
```

รูปที่ ๓.๑๘ แสดงการเก็บรูปแบบของคำสั่ง DATA ในหน่วยความจำ

๓.๒.๒ DIM เป็นคำสั่งที่ใช้จองเนื้อที่ในหน่วยความจำของตัวแปรหมวด รายละเอียดของตัวแปรหมวด ดูหัวข้อ ๒.๒.๕ รูปแบบของคำสั่งนี้แสดงไว้รูปที่ ๓.๑๙ ขั้นตอนการทำงานแสดงไว้ในผังงานที่ ๓.๘

Syntax

DIM variable ([minimum value,] maximum value)

Examples

a)
```
10 DIM A(5)
20 FOR I=1 TO 5
30 A(I)= I
40 PRINT A;
50 NEXT I
run
1  2  3  4  5
```

b)
```
10 DIM A(5)
20 FOR I=1 TO 5
30 A(I)= I
40 PRINT A
50 NEXT I
run
1
2
3
4
5
```

c)
```
10 DIM A(5,5)
20 FOR I=1 TO 5
30 FOR J=1 TO 5
40 PRINT A(I,J);
60 NEXT J
60 PRINT
70 NEXT I
run
0  0  0  0  0
0  0  0  0  0
0  0  0  0  0
0  0  0  0  0
0  0  0  0  0
```

d)
```
20 FOR I=1 TO 15
30 A(I)= I
40 PRINT A(I);
run
1 2 3 4 5 6 7 8 9 10
Subscript out of range in 30
ok
```
The system default of 10 is not large enough to handle the entire loop. As a result, the error is displayed.

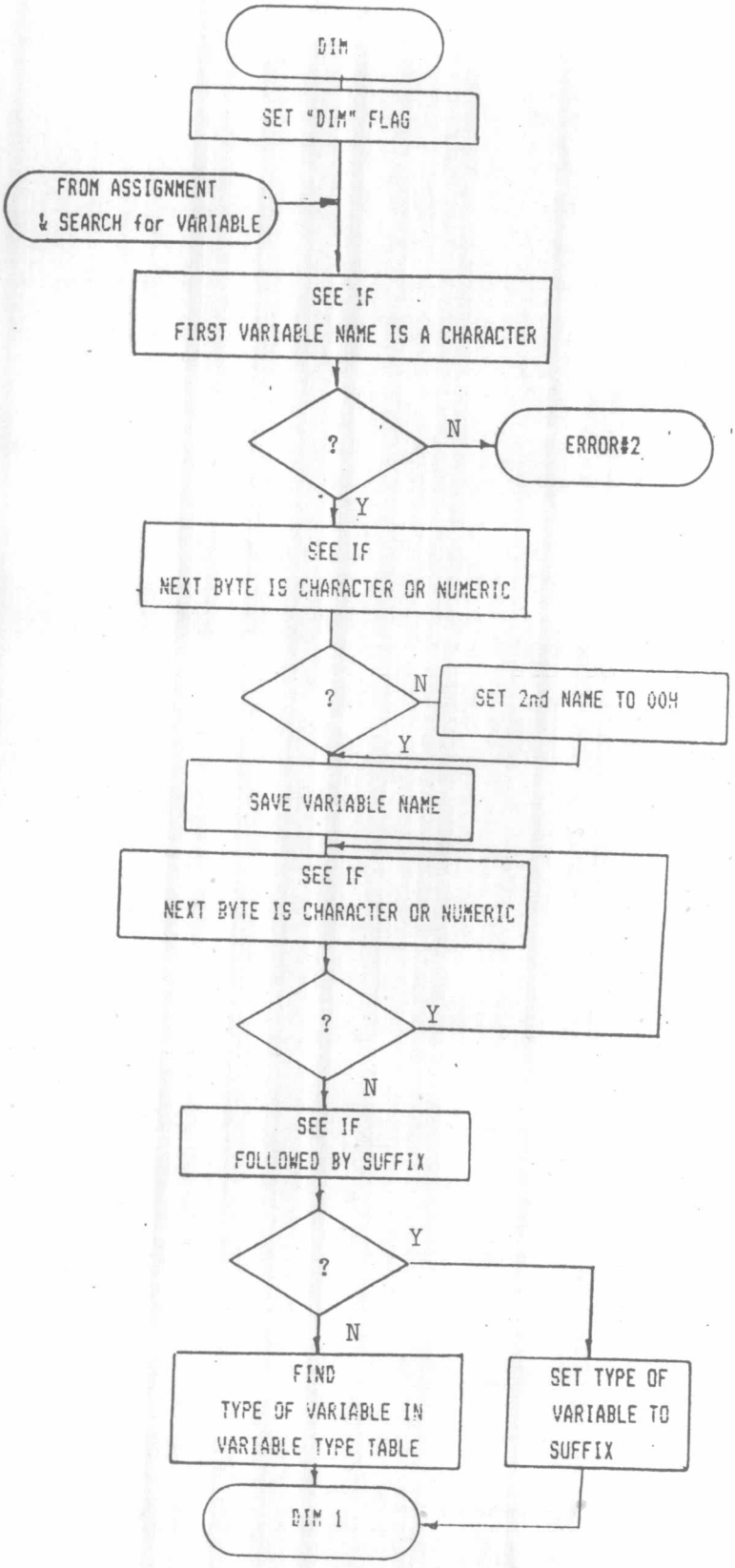
e)
```
10 DIM A$(15)
20 A$="GOOD AFTERNOON EVERYONE"
30 PRINT A$
run
GOOD AFTERNOON EVERYONE
```

รูปที่ ๓.๑๙ แสดงรูปแบบของคำสั่ง DIM

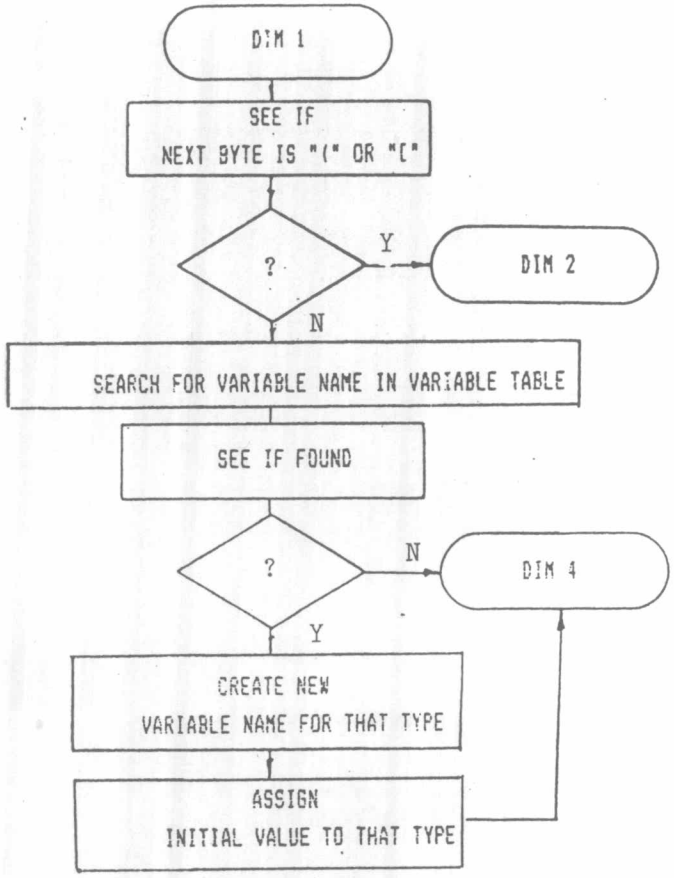
```
10 DIM D$(1,2,3)
20 D$(1,1,2)="String array"
```

```
8020 00 31 80 0A 00 86 20 44 24 28 12 2C 13 2C 14 29 .1.... D$(.,.,.)
8030 00 4E 80 14 00 44 24 28 12 2C 12 2C 13 29 F1 22 .N...D$(.,.,.)."
8040 53 74 72 69 6E 67 20 61 72 72 61 79 22 00 00 00 String array"...
8050 03 00 44 4F 00 03 04 00 03 00 02 00 00 00 00 00 ..DO............
8060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
8070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
8080 00 00 00 00 00 00 00 00 00 0C 40 80 00 00 00 00 ..........@.....
8090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
80A0 00 00 00 00 FF FF FF FF FF FF FF FF FF FF FF FF ................
```

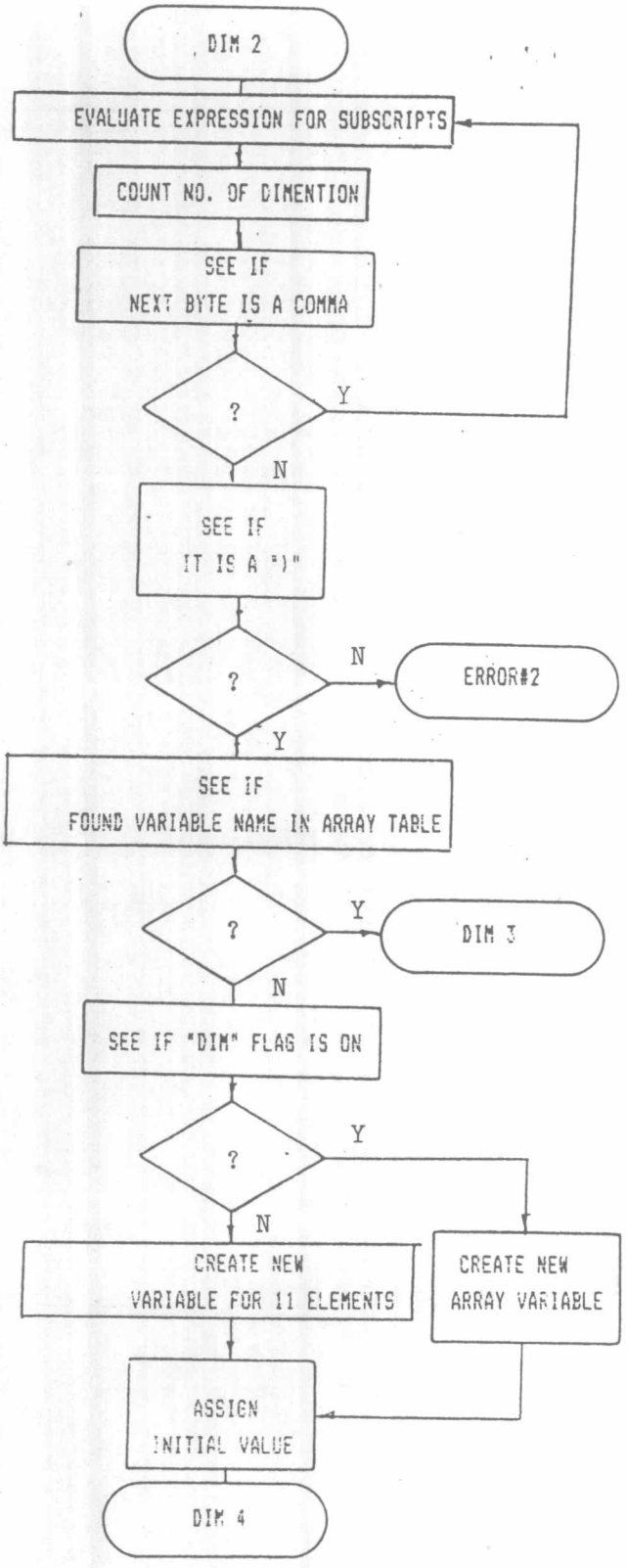
รูปที่ ๓.๒๐ แสดงการเก็บรูปแบบของคำสั่ง DIM ในหน่วยความจำ



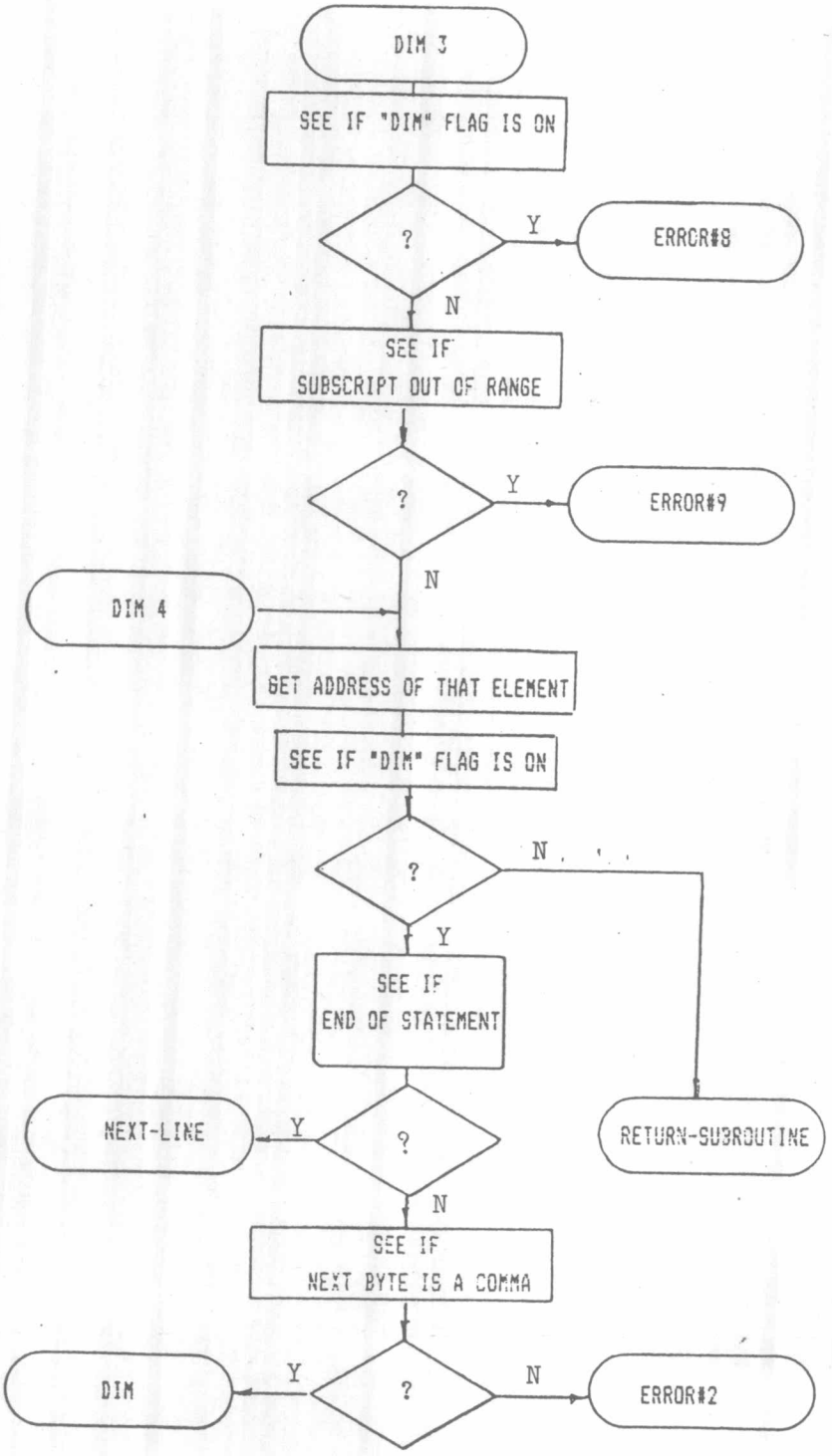
ผังงานที่ ๓.๘ แสดงขั้นตอนการทำงานของคำสั่ง DIM



ผังงานที่ ๓.๘ แสดงขั้นตอนการทำงานของคำสั่ง DIM (ต่อ)



ผังงานที่ ๓.๘ แสดงขั้นตอนการทำงานของคำสั่ง DIM (ต่อ)



ผังงานที่ ๓.๘ แสดงขั้นตอนการทำงานของคำสั่ง DIM (ต่อ)

๓.๒.๓ FOR....NEXT เป็นคำสั่ง ๒ คำสั่งใช้ในการสร้างลูป (LOOP) โดยคำสั่ง FOR เป็นคำสั่งเปิดลูป และเพิ่มหรือลดค่าให้กับตัวแปร คำสั่ง NEXT ใช้ในการปิดลูป และส่งการควบคุมไปยังคำสั่ง FOR รูปแบบของคำสั่งนี้แสดงไว้ในรูปที่ ๓.๒๑ การทำงานของคำสั่ง FOR นี้ อินเตอร์พริเตอร์จะทำการสร้างสแตค เพื่อเก็บข้อมูลในการสร้างลูปและการเพิ่มหรือลดค่า สแตคที่สร้างจะใช้เนื้อที่ ๑๗ ไบท์ ต่อการสร้างลูป ๑ ลูป (ดูรูปที่ ๓.๒๒) ขั้นตอนการทำงานของคำสั่งแสดงไว้ในผังงานที่ ๓.๙ และผังงานที่ ๓.๑๐

Syntax

FOR variable = x TO y [STEP z]

Examples

a)
```
10 REM SIMPLE
   ITERATION
20 FOR I=1 TO 5
30 PRINT I;
40 NEXT I
```

b)
```
10 REM ITERATION
   USING STEP
20 FOR I=5 TO 10 STEP 2
30 PRINT I;
40 NEXT I
```

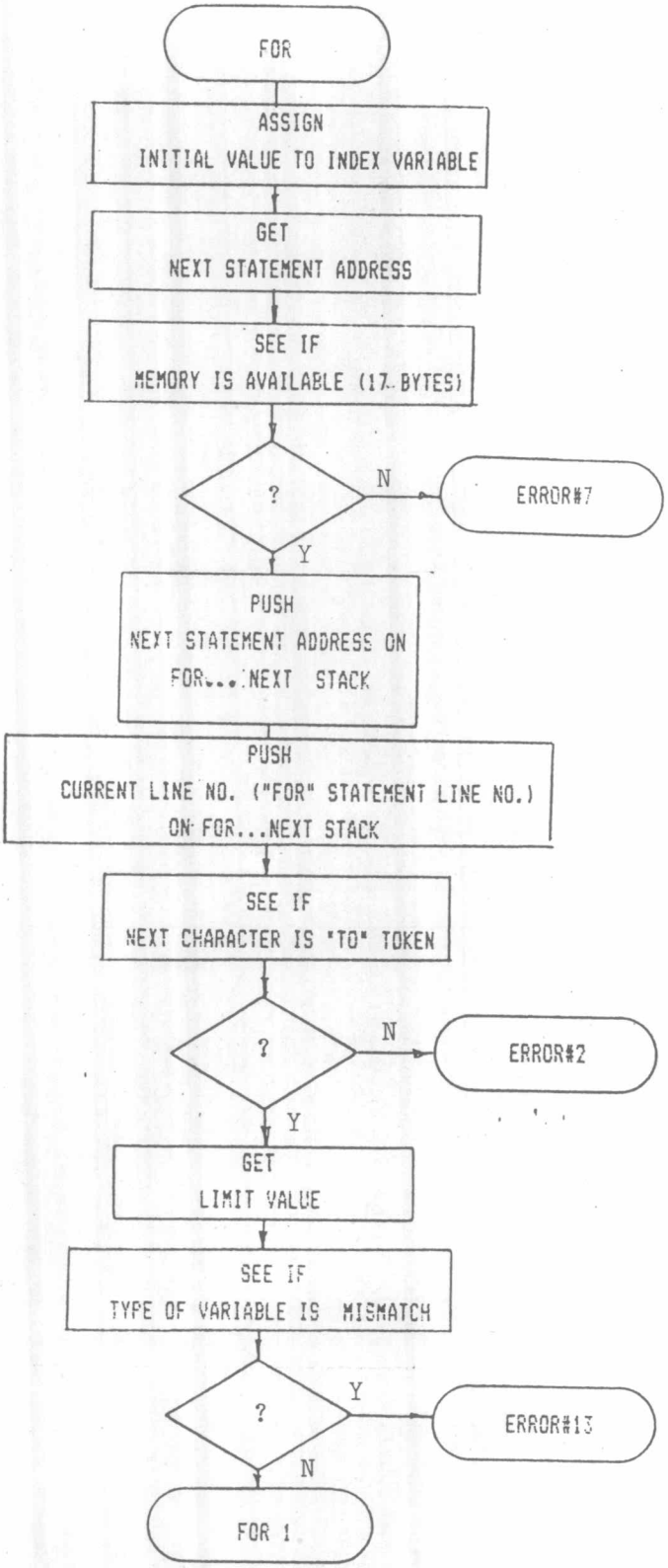
c)
```
10 REM ITERATION
   USING — STEP
20 FOR I=5 TO 1 STEP
   — 1
30 PRINT I;
40 NEXT I
```

d)
```
10 REM NESTED LOOPS
20 FOR I=1 TO 5
30 FOR J=1 TO 5
40 PRINT I+J
50 NEXT J
60 PRINT
70 NEXT I
```

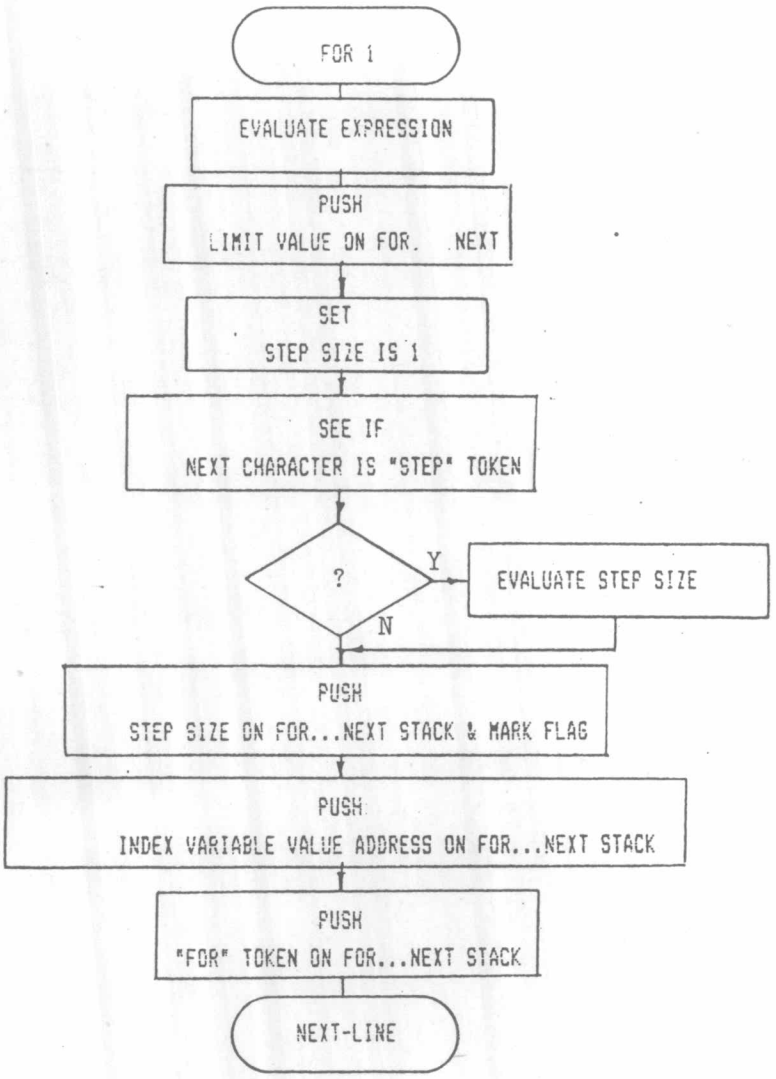
รูปที่ ๓.๒๑ แสดงรูปแบบของคำสั่ง FOR.....NEXT

```
10 FOR I= 1 TO 10 STEP 2
20      PRINT "FOR-NEXT"
30 NEXT I
100 'FOR-NEXT Statement
10 FOR I= 1 TO 10000  STEP 2
20      PRINT "FOR-NEXT"
30 NEXT I
100 'FOR-NEXT Statement
                BEFORE EXECUTION
8020 00 37 80 0A 00 82 20 49 F1 20 12 20 D7 20 1C 10 .7.... I. . . . ..
8030 27 20 20 DA 20 13 00 4D 80 14 00 20 20 20 20 20 '  . ..M...
8040 91 20 22 46 4F 52 2D 4E 45 58 54 22 00 55 80 1E . "FOR-NEXT".U..
8050 00 83 20 49 00 6F 80 64 00 3A 8F E4 46 4F 52 2D .. I.o.d.:..FOR-
8060 4E 45 58 54 20 53 74 61 74 65 6D 65 6E 74 00 00 NEXT Statement..
8070 00 64 FF FF FF FF FF FF FF FF FF FF FF FF FF FF .d..............
8080 FF
                AFTER EXECUTION

8020 00 37 80 0A 00 82 20 49 F1 20 12 20 D7 20 1C 10 .7.... I. . . . ..
8030 27 20 20 DA 20 13 00 4D 80 14 00 20 20 20 20 20 '  . ..M...
8040 91 20 22 46 4F 52 2D 4E 45 58 54 22 00 55 80 1E . "FOR-NEXT".U..
8050 00 83 20 49 00 6F 80 64 00 3A 8F E4 46 4F 52 2D .. I.o.d.:..FOR-
8060 4E 45 58 54 20 53 74 61 74 65 6D 65 6E 74 00 00 NEXT Statement..
8070 00 04 00 49 00 00 60 83 64 FF FF FF FF FF FF FF ...I..`.d.......
8080 FF
              FOR...NEXT STACK

E8C0 82 74 80 01 01 00 00 00 82 00 40 1C 9E 0A 00 36 .t........๔....6
E8D0 80 00
```

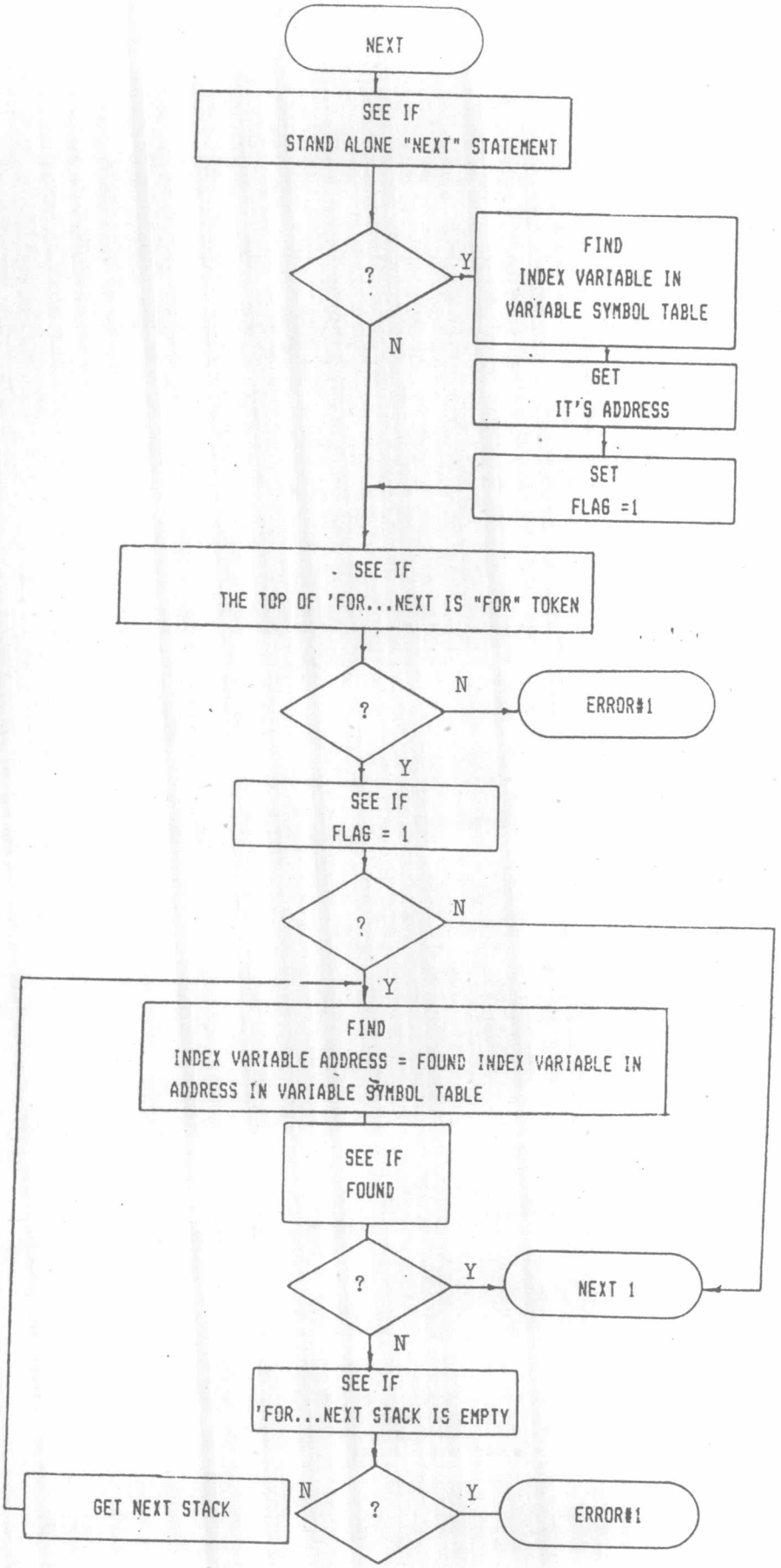
รูปที่ ๓.๒๒ แสดงการเก็บรูปแบบของคำสั่ง FOR ... NEXT ในหน่วยความจำ



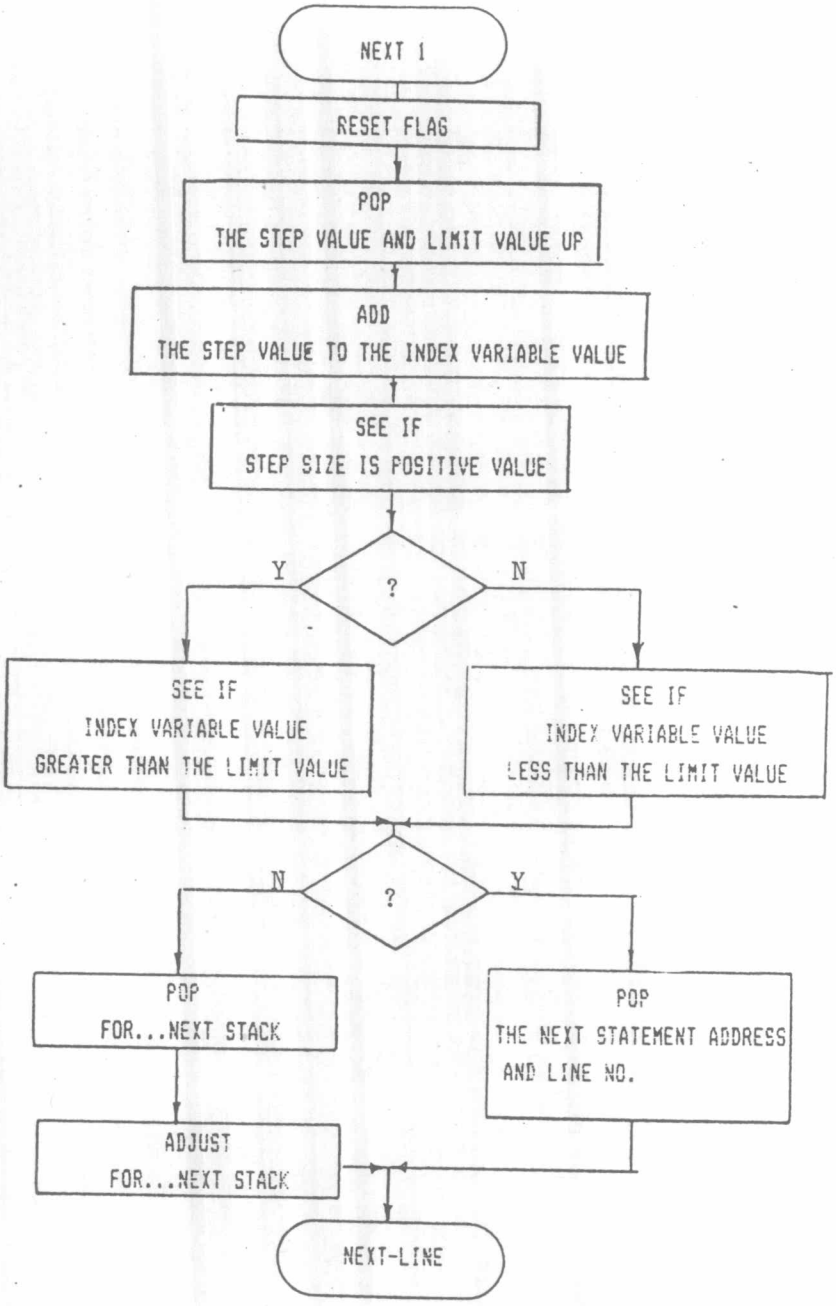
ผังงานที่ ๓.๙ แสดงขั้นตอนการทำงานของคำสั่ง FOR



ผังงานที่ ๓.๙ แสดงขั้นตอนการทำงานของคำสั่ง FOR (ต่อ)



ผังงานที่ ๓.๑๐ แสดงขั้นตอนการทำงานของคำสั่ง NEXT



ผังงานที่ ๓.๑๐ แสดงขั้นตอนการทำงานของคำสั่ง NEXT (ต่อ)

๓.๒.๔ GOSUB....RETURN เป็นคำสั่ง ๒ คำสั่ง คำสั่ง GOSUB ใช้เรียกโปรแกรมย่อย และส่งการควบคุมไปยังโปรแกรมย่อยนั้น ส่วนคำสั่ง RETURN ใช้ในการส่งการควบคุมกลับมายังโปรแกรมหลัก เพื่อให้โปรแกรมหลักทำงานต่อ โดยจะปฏิบัติคำสั่งหลังจากคำสั่ง GOSUB เป็นต้นไป รูปแบบของคำสั่งแสดงไว้ในรูปที่ ๓.๒๓ การทำงานของคำสั่ง GOSUB นี้ อินเตอร์พรีเตอร์จะทำการสร้างสแตค เพื่อกลับมายังโปรแกรมหลักในแอดเดรสที่ถูกต้อง สแตคที่สร้างจะใช้เนื้อที่ ๗ ไบท์ต่อการใช้คำสั่ง GOSUB ๑ คำสั่ง (ดูรูปที่ ๓.๒๔) ขั้นตอนการทำงานของคำสั่งแสดงไว้ในผังงานที่ ๓.๑๑ และผังงานที่ ๓.๑๒

Syntax

GOSUB line number

Example

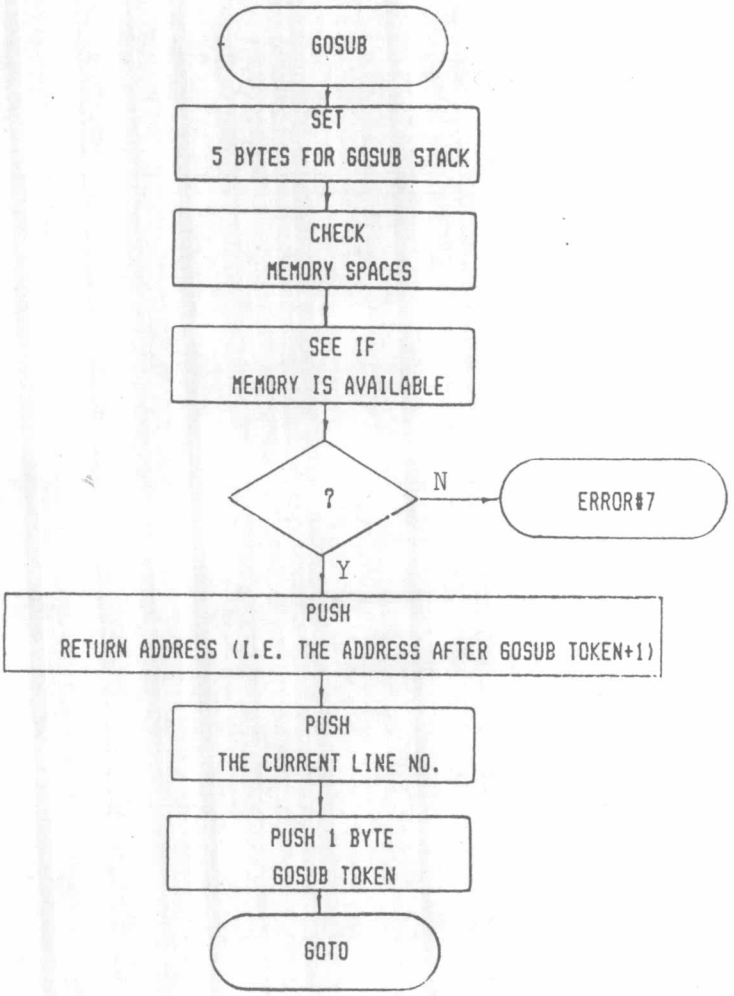
```
10 PRINT "BEFORE SUBROUTINE J=";J
20 GOSUB 60
30 PRINT
40 PRINT "AFTER SUBROUTINE J= ";J
50 END
60 J=J+5
70 RETURN
run
BEFORE SUBROUTINE J=0
AFTER SUBROUTINE J=5
```

รูปที่ ๓.๒๓ แสดงรูปแบบของคำสั่ง GOSUB.....RETURN

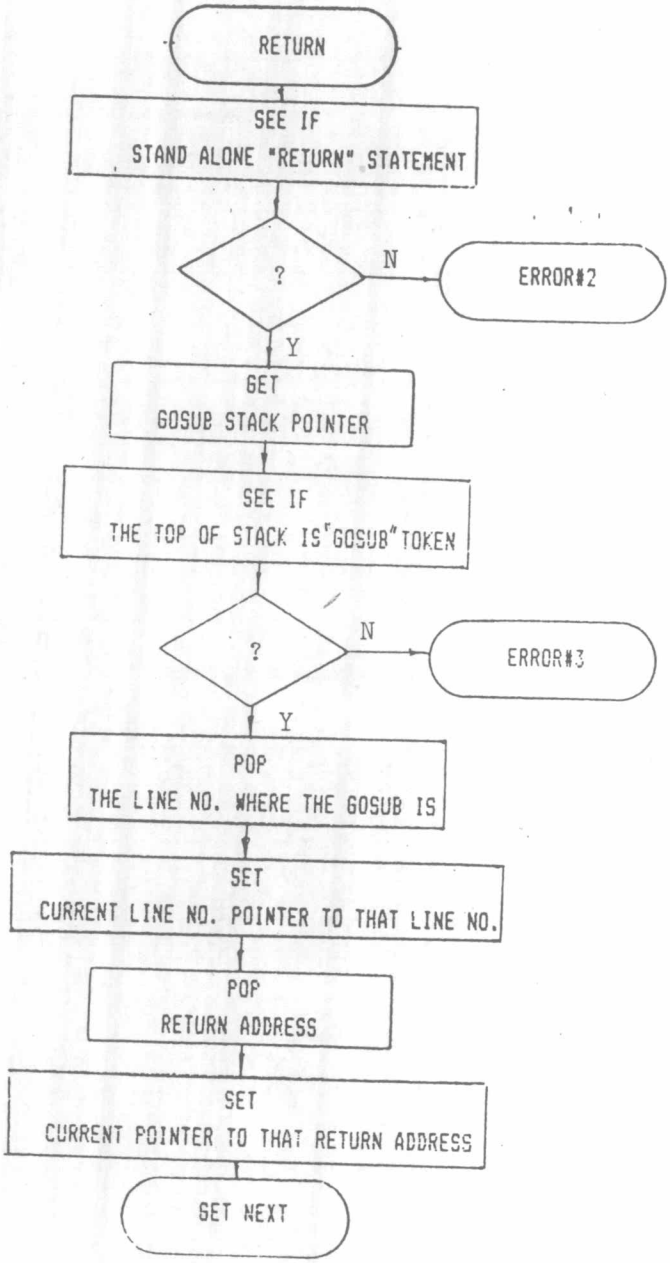
```
10 GOSUB 100
20 END
100 'GOSUB & RETURN Statement
200 RETURN
```

```
                    BEFORE EXECUTION
8020 00 2B 80 0A 00 8D 20 0E 64 00 00 31 80 14 00 81 .+.... .d..1....
8030 00 51 80 64 00 3A 8F E4 47 4F 53 55 42 20 26 20 .Q.d.:..GOSUB &
8040 52 45 54 55 52 4E 20 53 74 61 74 65 6D 65 6E 74 RETURN Statement
8050 00 57 80 C8 00 8E 00 00 00 4F FF FF FF FF FF FF .W.......O......
8060 FF
                    GOSUB...RETURN STACK

E8C0 74 0D 10 20 D5 A0 D1 A0 32 A0 FA 41 8D 0A 00 2A t.. ....2..A...*
E8D0 80
                    AFTER EXECUTION

8020 00 2B 80 0A 00 8D 20 0D 30 80 00 31 80 14 00 81 .+.... .0..1....
8030 00 51 80 64 00 3A 8F E4 47 4F 53 55 42 20 26 20 .Q.d.:..GOSUB &
8040 52 45 54 55 52 4E 20 53 74 61 74 65 6D 65 6E 74 RETURN Statement
8050 00 57 80 C8 00 8E 00 00 00 FF FF FF FF FF FF FF .W.............
8060 FF
```

รูปที่ ๓.๒๔ แสดงการเก็บรูปแบบของคำสั่ง GOSUB ... RETURN ในหน่วยความจำ



ผังงานที่ ๓.๑๑ แสดงขั้นตอนการทำงานของคำสั่ง GOSUB



ผังงานที่ ๓.๑๒ แสดงขั้นตอนการทำงานของคำสั่ง RETURN

๓.๒.๔ GOTO เป็นคำสั่งที่ใช้ในการข้ามไปยังหมายเลขบรรทัดที่ต้องการในโปรแกรมโดยไม่มีเงื่อนไข มีรูปแบบดังรูปที่ ๓.๒๔ เพื่อการปฏิบัติงานให้รวดเร็วขึ้น อินเตอร์พรีเตอร์จะเปลี่ยนหมายเลขบรรทัดให้เป็นแอดเดรสของหมายเลขบรรทัดนั้น หลังจากปฏิบัติงานคำสั่งนี้ ครั้งแรกการที่อินเตอร์พรีเตอร์ทราบว่า หลังคำสั่ง GOTO จะเป็นหมายเลขบรรทัดหรือแอดเดรสของบรรทัดนั้น โดยการแสดงแฟลกว่าเป็นหมายเลขบรรทัด (Line number flag) ซึ่งแทนด้วยรหัส ØEH หรือ แอดเดรส (Address flag) ซึ่งแทนด้วยรหัส ØDH (ดูรูปที่ ๓.๒๖) ขั้นตอนการทำงานแสดงไว้ในผังงานที่ ๓.๑๓

Syntax

GOTO line number

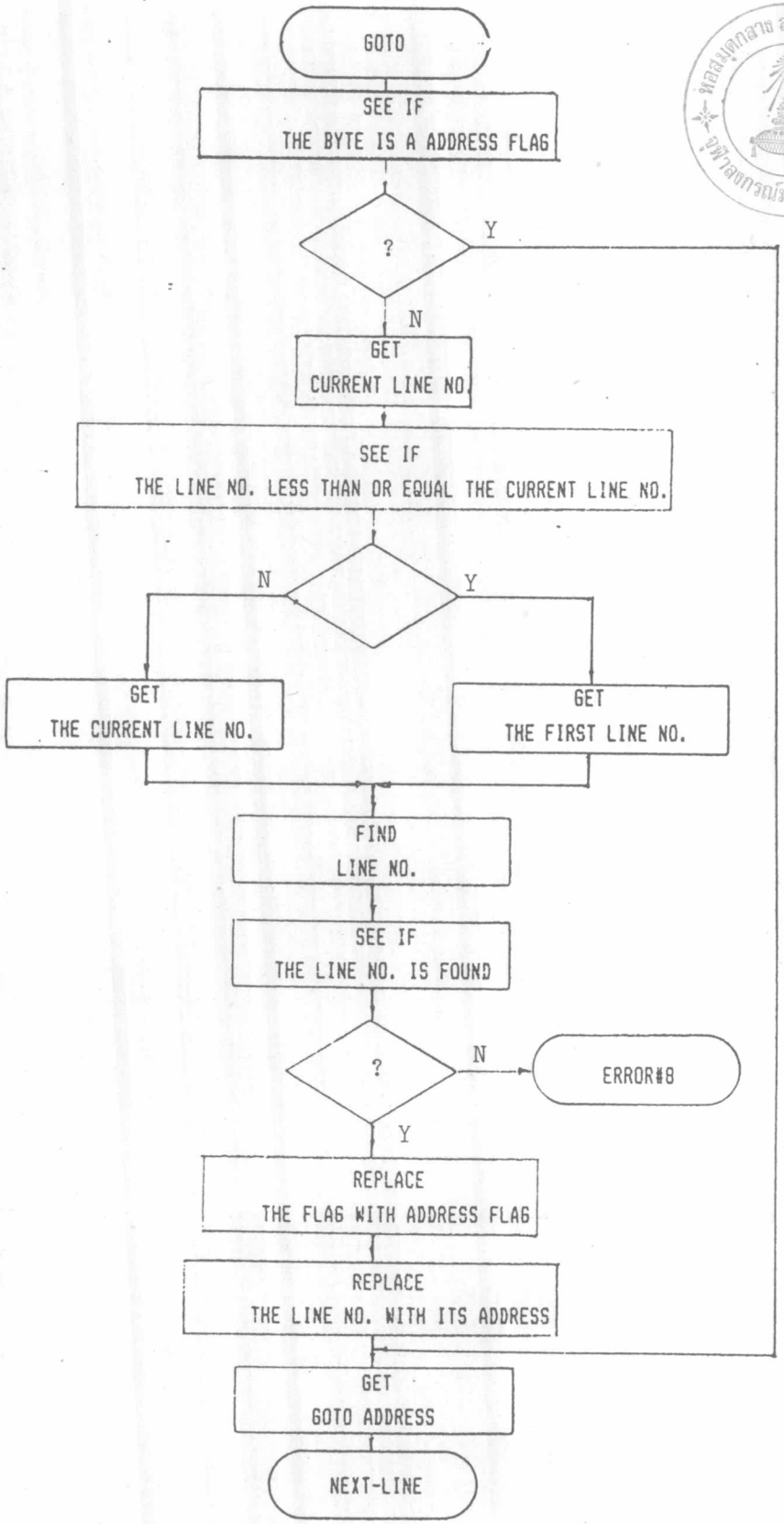
Example

```
10 READ R
20 PRINT "R= ";R;
30 A=3.14*R^2
40 PRINT "AREA= ";A
50 GOTO 10
60 DATA 5,7,12
```

รูปที่ ๓.๒๔ แสดงรูปแบบของคำสั่ง GOTO

```
10 GOTO  100
100 'GO TO Statement

8020 00 2C 80 0A 00 89 20 20 0E 64 00 00 43 80 64 00 .,.... .d..C.d.
8030 3A 8F E4 47 4F 20 54 4F 20 53 74 61 74 65 6D 65 :..GO TO Stateme
8040 6E 74 00 00 00 FF FF FF FF FF FF FF FF FF FF FF nt..............
8050 FF
```

รูปที่ ๓.๒๖ แสดงการเก็บรูปแบบของคำสั่ง GOTO ในหน่วยความจำ

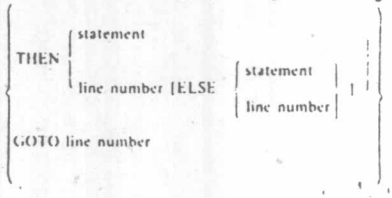


ผังงานที่ ๓.๑๓ แสดงขั้นตอนการทำงานของคำสั่ง GOTO

๓.๒.๖ IF.....THEN เป็นคำสั่งที่ใช้ในการสร้างเงื่อนไข เพื่อเลือกเส้นทางการปฏิบัติงานของโปรแกรม เงื่อนไขที่สร้างขึ้นเป็นเงื่อนไขทางตรรก รูปแบบของคำสั่งแสดงไว้ในรูปที่ ๓.๒๗ ขั้นตอนการทำงานแสดงไว้ในผังงานที่ ๓.๑๔

Syntax

IF logical expression [AND logical expression]



Examples

Simple IF Statement

```
10 PRINT TIME$
20 INPUT "TYPE YES OR NO TO CONTINUE";A$
30 IF A$="YES" THEN 10
40 PRINT
50 PRINT "GOODBY"
60 END
```

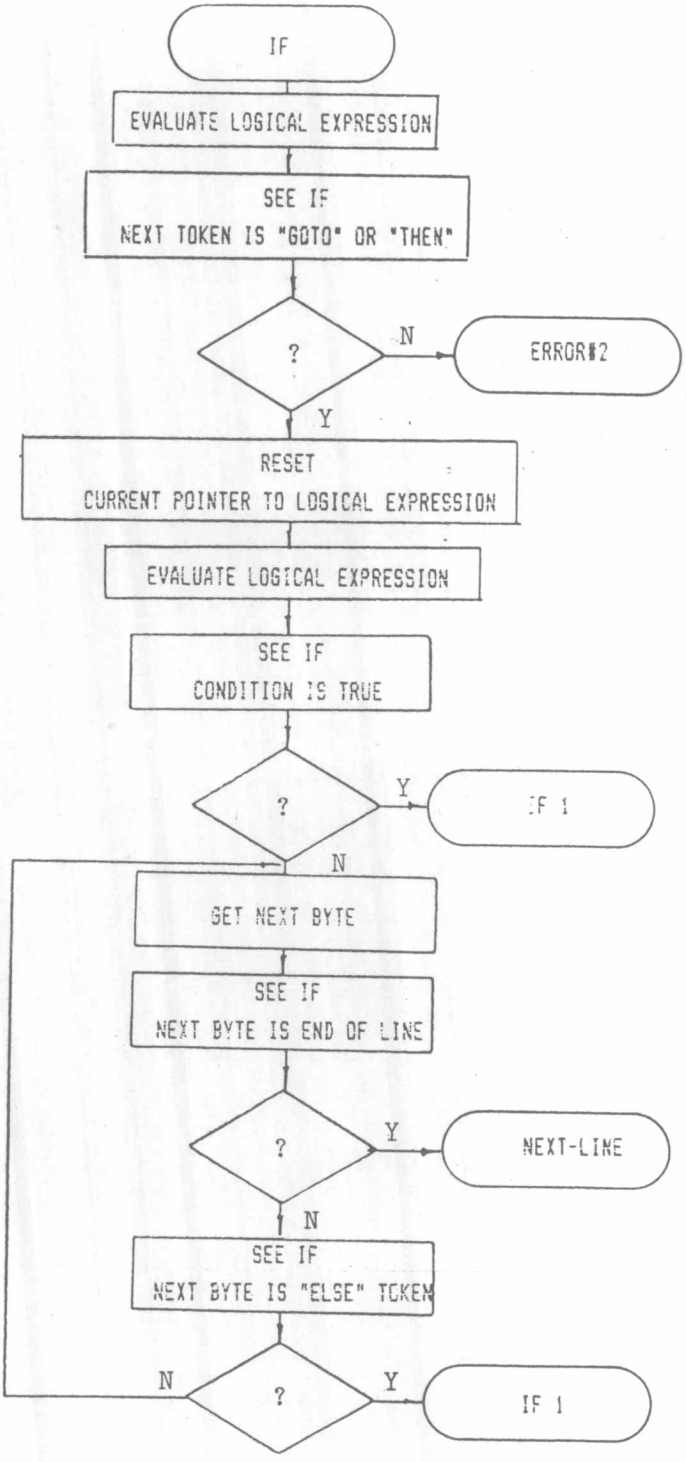
The program prints the current time on the system clock repetitively if the test condition in line 30 is met. Program execution continues to the next line (40) if the test condition is not met.

รูปที่ ๓.๒๗ แสดงรูปแบบของคำสั่ง IF.....THEN....

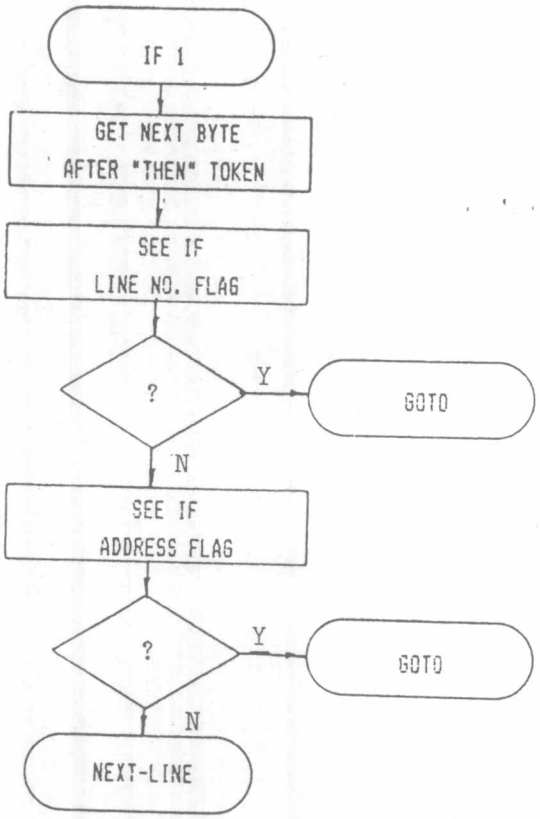
```
10 LET A=2 :LET B=1
20 LET C=A*B :LET B=C+1
30 IF B>100 THEN END ELSE 20
100 'LET & IF....THEN....ELSE Statement
```

```
8020 00 32 80 0A 00 88 20 41 F1 13 20 3A 88 20 42 F1 .2.... A.. :. B.
8030 12 00 47 80 14 00 88 20 43 F1 41 F5 42 20 3A 88 ..G.... C.A.B :.
8040 20 42 F1 43 F3 12 00 5D 80 1E 00 8B 20 42 F0 0F  B.C...]....  B..
8050 64 20 D8 20 81 20 3A A1 20 0D 31 80 00 87 80 64 d . . :. .1....d
8060 00 3A 8F E4 4C 45 54 20 26 20 49 46 2E 2E 2E 2E .:..LET & IF....
8070 54 48 45 4E 2E 2E 2E 2E 45 4C 53 45 20 53 74 61 THEN....ELSE Sta
8080 74 65 6D 65 6E 74 00 00 00 04 00 41 00 00 00 82 tement.....A....
8090 04 00 42 00 00 7E 87 04 00 43 00 00 7C 87 04 FF ..B..~...C..|...
```

รูปที่ ๓.๒๘ แสดงการเก็บรูปแบบของคำสั่ง IF ... THEN ในหน่วยความจำ



ผังงานที่ ๓.๑๔ แสดงขั้นตอนการทำงานของคำสั่ง IF ... THEN



ผังงานที่ ๓.๑๔ แสดงขั้นตอนการทำงานของคำสั่ง IF ... THEN (ต่อ)

๓.๒.๗ INPUT เป็นคำสั่งรับข้อมูลจากแป้นพิมพ์ (key board) รูปแบบของคำสั่งแสดงไว้ในรูปที่ ๓.๒๙ ขั้นตอนการทำงานแสดงไว้ในผังงานที่ ๓.๑๕

Syntax

INPUT ["prompt string";] variable, variable . . .

Examples

```
10 INPUT "INPUT A NUMBER";X
20 PRINT X " SQUARED IS"; X^2
```

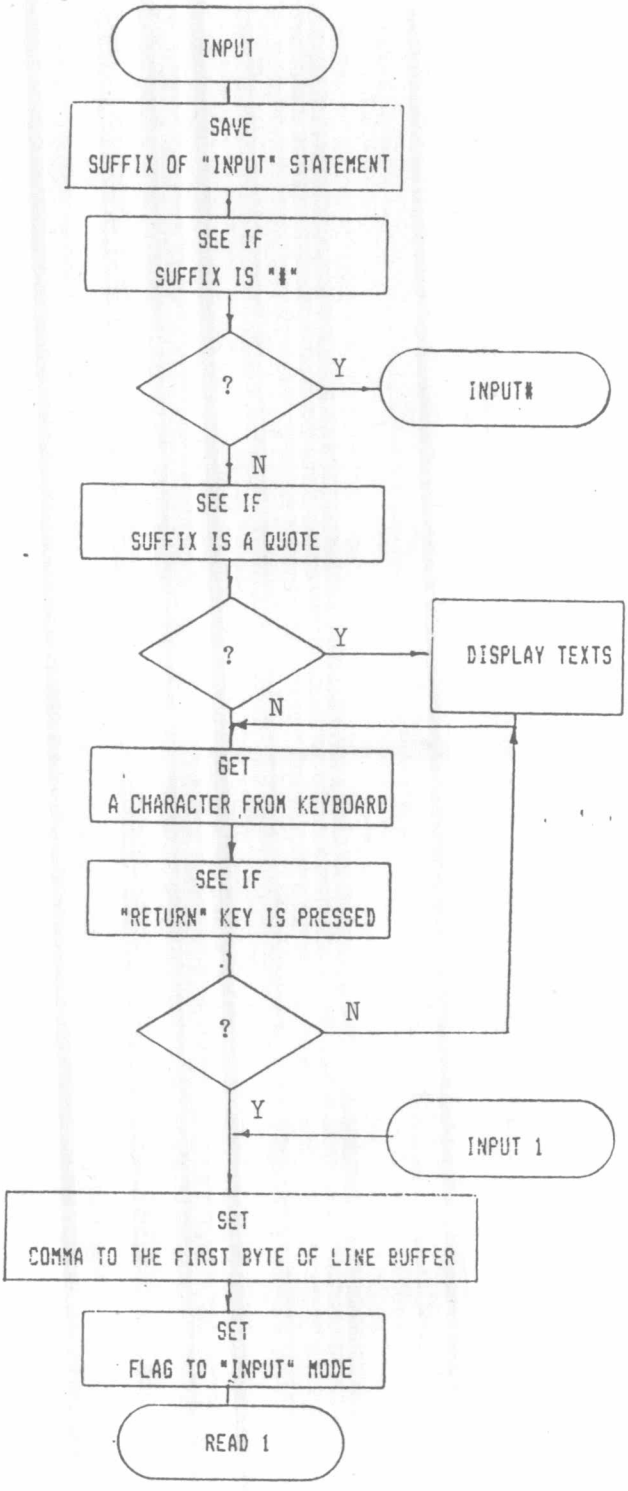
Data items must be separated by commas and coincide with the number of variables listed after the INPUT statement. If too few data items are supplied, the system displays two question marks "??" and waits for further input. If too many data items are entered, the message "EXTRA IGNORED" is displayed and execution continues. Data input must correspond to the data types of the variables.

You also have the option of including a prompt string with the INPUT statement to aid in proper data input. When using a prompt string, the system displays the prompt immediately followed by a question mark.

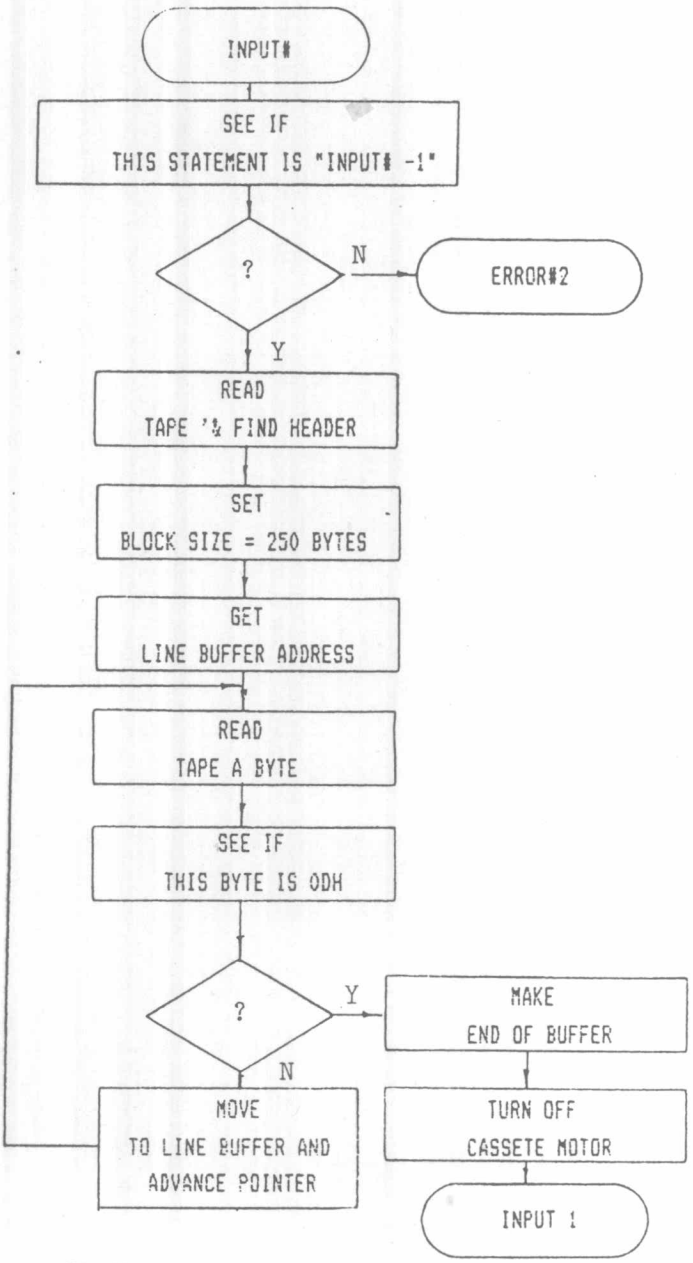
รูปที่ ๓.๒๙ แสดงรูปแบบของคำสั่ง INPUT

```
10 INPUT"STRING ";ST$
20 INPUT"INTEGER ";IT%
30 INPUT"SINGLE PRECISION ";SG
40 INPUT"DOUBLE PRECISION ";DB#
100 'INPUT Statement

8020 00 34 80 0A 00 85 22 53 54 52 49 4E 47 20 22 3B .4....."STRING ";
8030 53 54 24 00 48 80 14 00 85 22 49 4E 54 45 47 45 ST$.H...."INTEGE
8040 52 20 22 3B 49 54 25 00 64 80 1E 00 85 22 53 49 R ";IT%.d...."SI
8050 4E 47 4C 45 20 50 52 45 43 49 53 49 4F 4E 20 22 NGLE PRECISION "
8060 3B 53 47 00 81 80 28 00 85 22 44 4F 55 42 4C 45 ;SG...(.."DOUBLE
8070 20 50 52 45 43 49 53 49 4F 4E 20 22 3B 44 42 23  PRECISION ";DB#
8080 00 98 80 64 00 3A 8F E4 49 4E 50 55 54 20 53 74 ...d.:..INPUT St
8090 61 74 65 6D 65 6E 74 00 00 00 03 54 53 1A E6 E9 atement....TS...
80A0 02 54 49 0A 00 04 47 53 00 00 20 84 08 42 44 00 .TI...GS.. ..BD.
80B0 00 00 00 00 00 20 84 64 FF FF FF FF FF FF FF FF ..... .d........
80C0 00

E9E0 00 00 00 00 00 00 73 74 72 69 6E 67 20 66 6F 72 ......string for
E9F0 20 69 6E 70 75 74 20 73 74 61 74 65 6D 65 6E 74  input statement
EA00 D3
```

รูปที่ ๓.๓๐ แสดงการเก็บรูปแบบของคำสั่ง INPUT ในหน่วยความจำ



ผังงานที่ ๓.๑๕ แสดงขั้นตอนการทำงานของคำสั่ง INPUT



ผังงานที่ ๓.๑๕ แสดงขั้นตอนการทำงานของคำสั่ง INPUT (ต่อ)

๓.๒.๘ INPUT # - 1 เป็นคำสั่งที่ใช้ในการอ่านระเบียน (Record) จากแฟ้มข้อมูลแบบลำดับ (Sequential file) ในเทปคาสเซท รูปแบบของคำสั่งแสดงไว้ในรูปที่ ๓.๓๑ ขั้นตอนการทำงานใช้ผังงานเดียวกับคำสั่ง INPUT (ผังงานที่ ๓.๑๕)

Syntax

INPUT# file number, variable, variable, . . .

Variables in an INPUT# statement must be identical in data type to the stored record items. For example, the variable XS cannot be initialized to a record item stored as an integer.

รูปที่ ๓.๓๑ แสดงรูปแบบของคำสั่ง INPUT # - 1

```
10 INPUT#-1,A,B%,C$
20 'INPUT#-1 Statement
```

```
8020 00 32 80 0A 00 85 23 F4 12 2C 41 2C 42 25 2C 43 .2....#..,A,B%,C
8030 24 00 4C 80 14 00 3A 8F E4 49 4E 50 55 54 23 2D $.L...:..INPUT#-
8040 31 20 53 74 61 74 65 6D 65 6E 74 00 00 00 14 00 1 Statement.....
8050 00
```

รูปที่ ๓.๓๒ แสดงการเก็บรูปแบบของคำสั่ง INPUT # -1 ในหน่วยความจำ

๓.๒.๔ PRINT เป็นคำสั่งใช้ในการแสดงผลค่าของตัวแปร หรือค่าคงที่บนจอภาพ ตำแหน่งในการพิมพ์ขึ้นอยู่กับเครื่องหมายวรรคตอน เช่น , ; ถ้าใช้จุลภาค (,) จะแยกระหว่างข้อมูล ๑๔ ตัวอักษร แต่ถ้าใช้อัฒภาค (;) จะพิมพ์ข้อมูลตำแหน่งต่อมาจากข้อมูลเดิม รูปแบบของคำสั่งนี้แสดงไว้ในรูปที่ ๓.๓๓ ถ้าต้องการแสดงผลทางเครื่องพิมพ์ ให้ใช้คำสั่ง LPRINT รูปแบบของคำสั่ง LPRINT จะเหมือนคำสั่ง PRINT ทุกประการ ขั้นตอนการทำงานของคำสั่ง LIST และ LLIST แสดงไว้ในผังงานที่ ๓.๑๖

Syntax

PRINT ["prompt" [;] item,item,item. . .]

Examples

a) 10 FOR I=1 TO 6
   20 PRINT I
   30 NEXT I

b) 20 PRINT I;I+1;

The screen width is divided into 14 space zones. A comma used to separate items places the data every 14 spaces. A semicolon places data every other space.

A PRINT statement that is terminated by a comma or a semicolon causes the next PRINT statement to begin printing on the same line, spaced accordingly. A PRINT statement not terminated by a comma or a semicolon is followed by a carriage return.

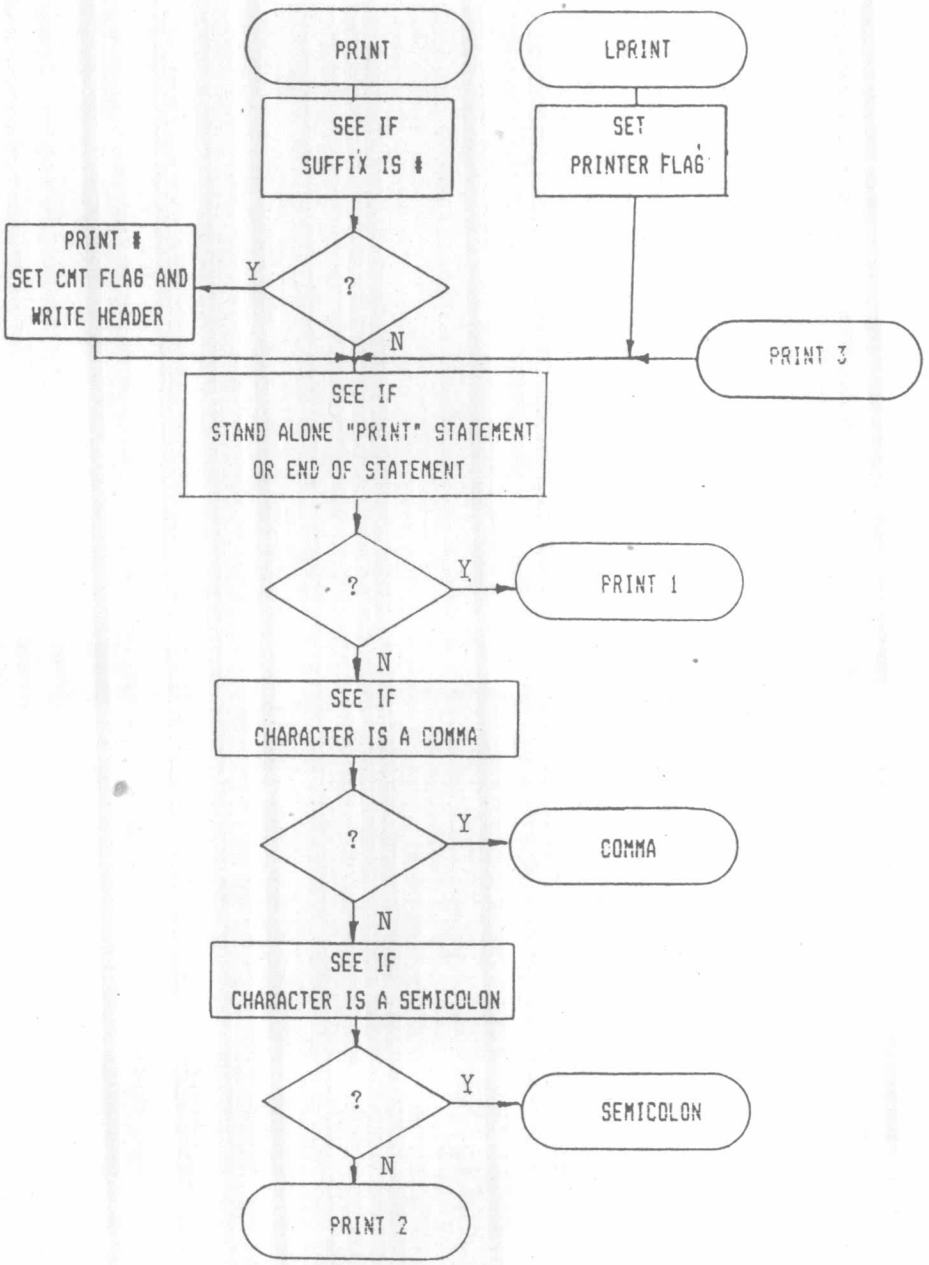
A question mark can be used as an abbreviated form of PRINT.

รูปที่ ๓.๓๓ แสดงรูปแบบของคำสั่ง PRINT

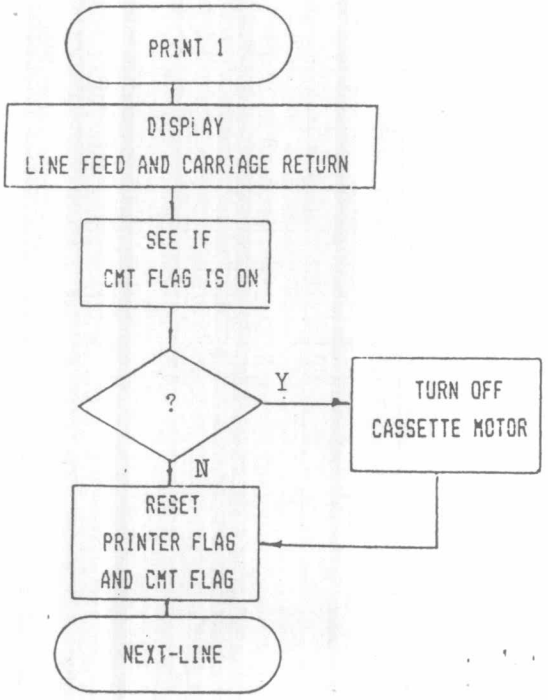
```
10 PRINT "PRINT Statement"
20 'INPUT#-1 Statement
```

```
8020 00 39 80 0A 00 91 20 22 50 52 49 4E 54 20 53 74 .9.... "PRINT St
8030 61 74 65 6D 65 6E 74 22 00 00 00 14 00 3A 8F E4 atement".....:..
```

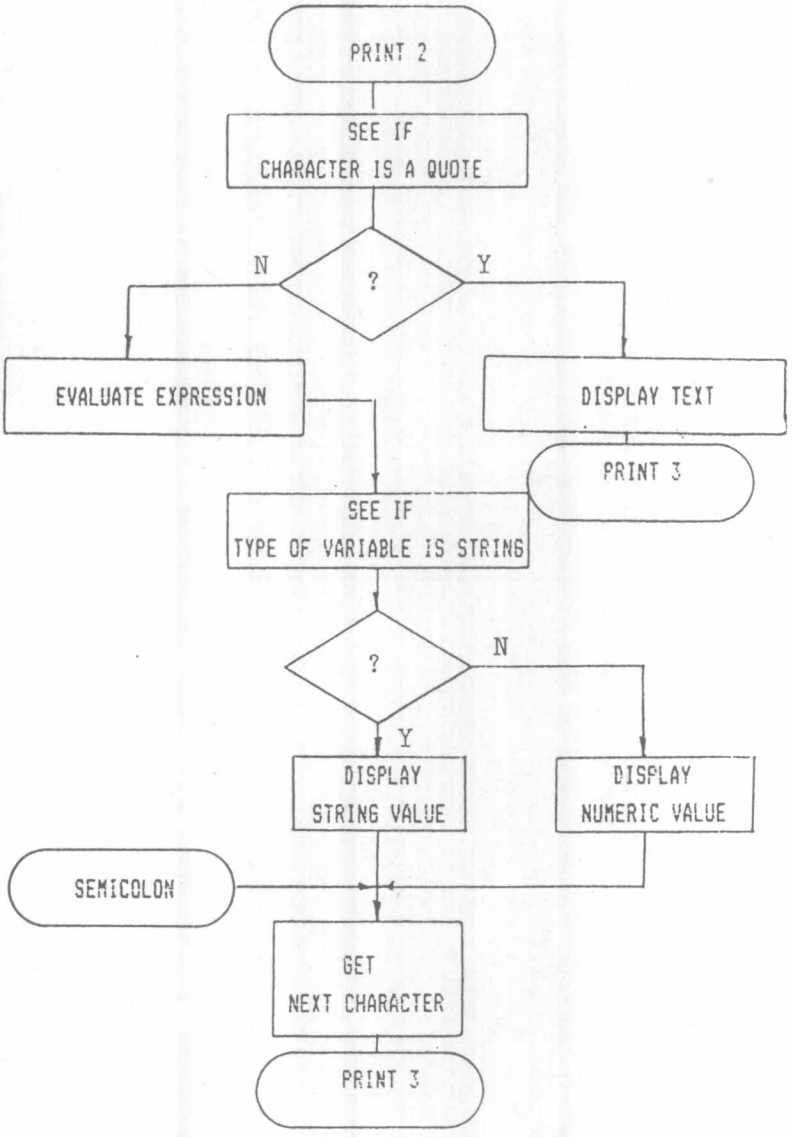
รูปที่ ๓.๓๔ แสดงการเก็บรูปแบบของคำสั่ง PRINT ในหน่วยความจำ



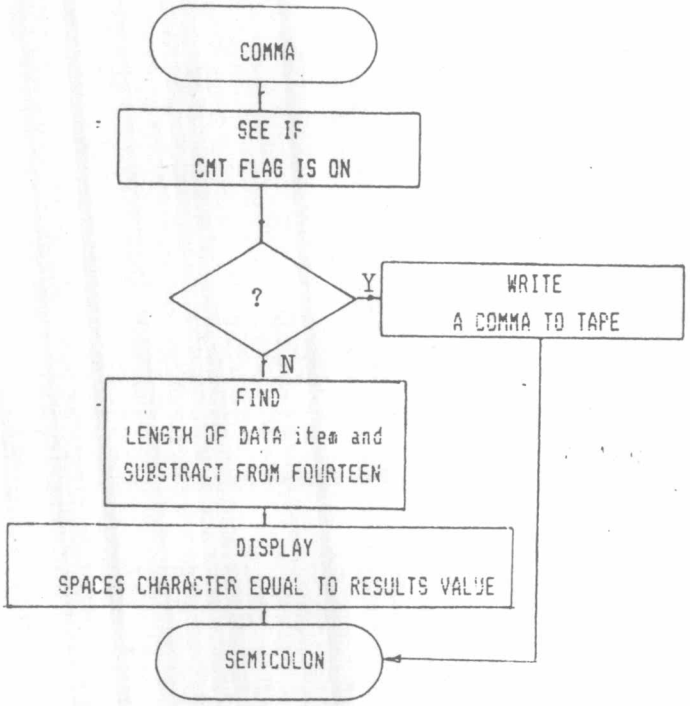
ผังงานที่ ๓.๑๖ แสดงขั้นตอนการทำงานของคำสั่ง PRINT



ผังงานที่ ๓.๑๖ แสดงขั้นตอนการทำงานของคำสั่ง PRINT (ต่อ)



ผังงานที่ ๓.๑๖ แสดงขั้นตอนการทำงานของคำสั่ง PRINT (ต่อ)



ผังงานที่ ๓.๑๖ แสดงขั้นตอนการทำงานของคำสั่ง PRINT (ต่อ)

๓.๒.๑๐ PRINT # -1 เป็นคำสั่งที่ใช้ในการเขียนระเบียนลง แฟ้มข้อมูลแบบลำดับ ในเทปคาสเซท รูปแบบของคำสั่งแสดงไว้ในรูปที่ ๓.๓๕

Syntax

PRINT# filenumber, variable, variable, . . .

รูปที่ ๓.๓๕ แสดงรูปแบบของคำสั่ง PRINT # -1

```
10 PRINT #-1,A,B%,C$
20 'PRINT#-1 Statement
```

```
8020 00 33 80 0A 00 91 20 23 F4 12 2C 41 2C 42 25 2C .3.... #...,A,B%,
8030 43 24 00 4D 80 14 00 3A 8F E4 50 52 49 4E 54 23 C$.M...:..PRINT#
8040 20 31 20 53 74 61 74 65 6D 65 6E 74 00 00 00 0A -1 Statement....
```

รูปที่ ๓.๓๖ แสดงการเก็บรูปแบบของคำสั่ง PRINT # -1 ในหน่วยความจำ

การเก็บข้อมูลไว้ในเทปคาสเซท มีรายละเอียดในการเก็บดังนี้ (ดูรูปที่ ๓.๓๗ และรูปที่ ๓.๓๘)

๓.๒.๑๐.๑ หัวข้อมูล (Header) ใช้ระหัส 9CH จำนวน ๖ ไบท์

๓.๒.๑๐.๒ ข้อมูล (Data) ข้อมูลเก็บเป็นระหัสแอสกี แต่ละข้อมูลกั้นด้วยจุลภาค

๓.๒.๑๐.๓ เครื่องหมายแสดงการหมดระเบียนใช้ระหัส 0DH 0AH

```
10 '  DATA REPRESENT IN TAPE
20 INPUT"DATA= ";A,B%,C#,D$
30 PRINT #-1,A,B%,C#,D$
40 END
```

รูปที่ ๓.๓๗ แสดงตัวอย่างโปแกรมที่ต้องการเขียนข้อมูลลงในเทปคาสเซท

```
FC FE 01 01 01 01 01 01 01 01 01 01 01 01 01 01 ................
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 ................
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 ................
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 ................
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 ................
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 ................
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 ................
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 ................
01 01 01 01 01 01 01 01 FF 9C 9C 9C 9C 9C 9C 20 ...............
31 30 30 20 2C 20 32 30 30 20 2C 20 33 30 30 20 100 , 200 , 300
2C 48 65 6C 6C 6F 0D 0A 01 01 01 01 01 01 01 D0 ,Hello.........
FF 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 ................
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 ................
01
```

รูปที่ ๓.๓๘ แสดงการเก็บข้อมูลในเทปคาสเซท

๓.๒.๑๑ READ เป็นคำสั่งที่ใช้ในการอ่านข้อมูล เพื่อกำหนดให้ตัวแปรจากคำสั่ง DATA จำนวนข้อมูลในคำสั่ง DATA จะต้องเพียงพอกับตัวแปรที่ต้องการกำหนดค่าในคำสั่ง READ มิฉะนั้นจะเกิดความผิดพลาดด้วยคำว่า Out of DATA รูปแบบของคำสั่งแสดงไว้ในรูปที่ ๓.๓๙ ขั้นตอนการทำงานแสดงไว้ในผังงานที่ ๓.๑๗

Syntax

READ variable, variable, . . .

Examples

a)
```
10 FOR I=1 TO 5
20 READ A
30 PRINT A;
40 NEXT I
50 DATA 1,2,3,4,5,7
```

b)
```
10 FOR I= 1 TO 5
20 READ A,B
30 PRINT A;B
40 NEXT I
50 DATA 1,2,3,4,5,
6,7,8,9,10
```

READ initializes variables to the data items of a DATA statement on a one-to-one basis. The READ statement variables can be of any data type, but must agree with the data type of the items in the DATA statement.

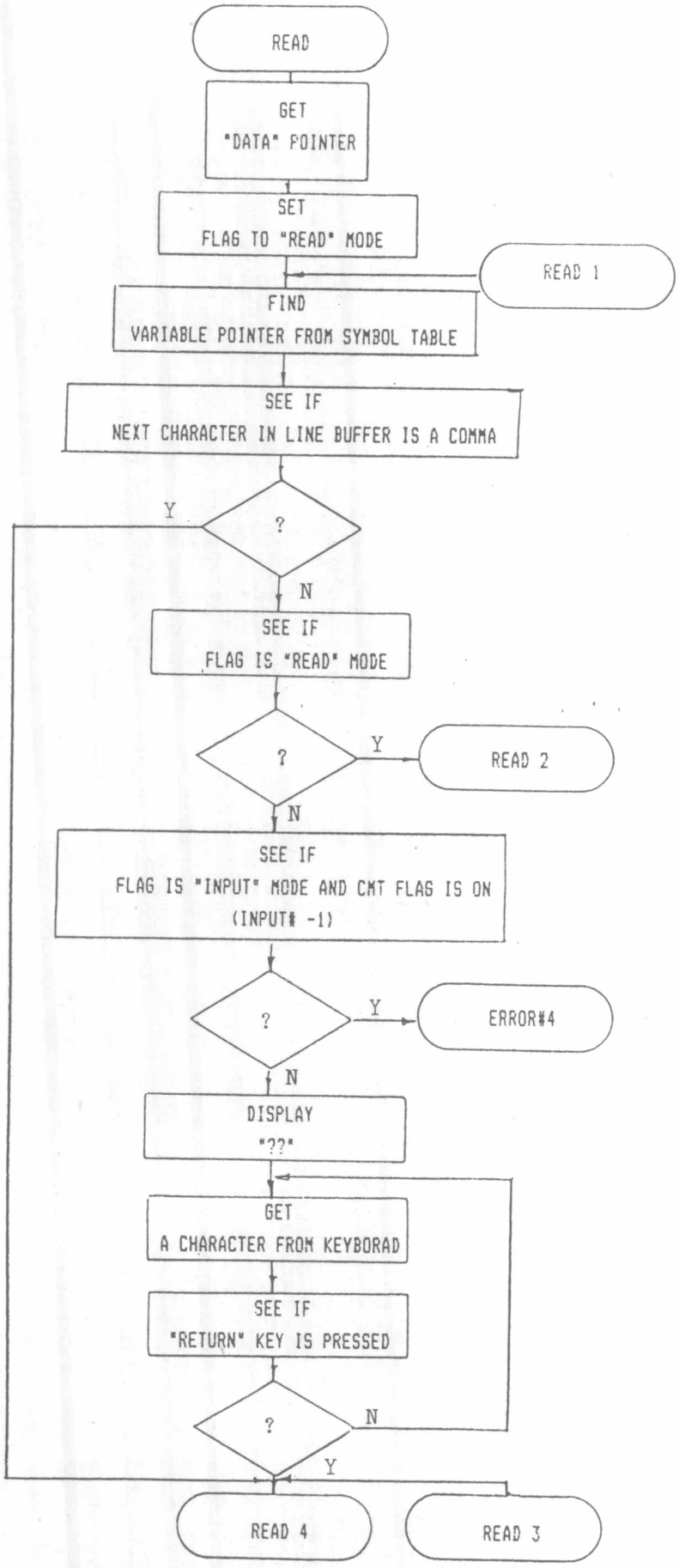
A single READ statement can access several DATA statements; several READ statements can access one DATA statement.

รูปที่ ๓.๓๙ แสดงรูปแบบของคำสั่ง READ

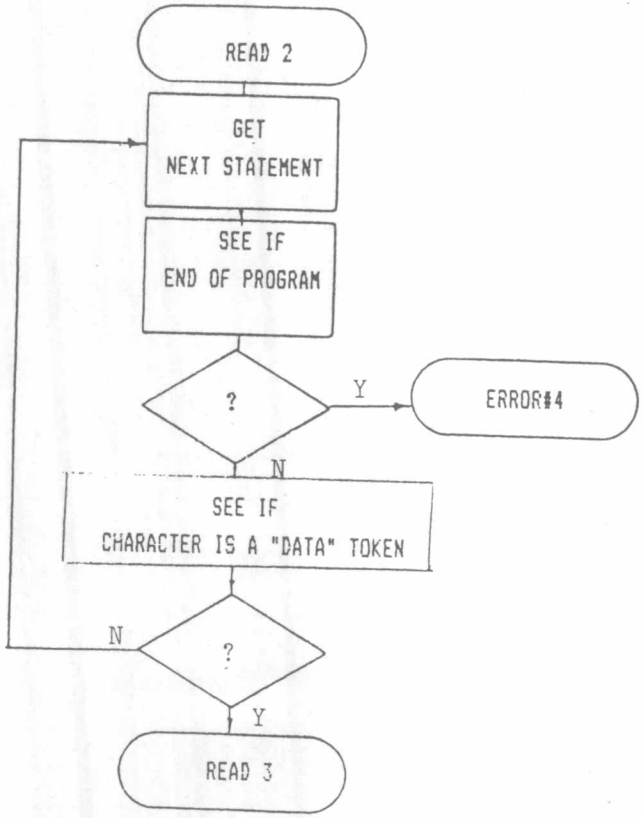
```
10 READ ST$,IT%
20 IF IT%=0 THEN 40
30 GOTO 10
40 RESTORE 50
50 DATA hello,1,BASIC,2,INTERPRETER,0
100 'READ DATA & RESTORE Statement
```

```
8020 00 2F 80 0A 00 87 20 53 54 24 2C 49 54 25 00 41 ./.... ST$,IT%.A
8030 80 14 00 8B 20 49 54 25 F1 11 20 D8 20 0D 4A 80 .... IT%.. . .J.
8040 00 4B 80 1E 00 89 20 0D 20 80 00 55 80 28 00 8C .K.... . ..U.(..
8050 20 0E 32 00 00 79 80 32 00 84 20 68 65 6C 6C 6F  .2..y.2.. hello
8060 2C 31 2C 42 41 53 49 43 2C 32 2C 49 4E 54 45 52 ,1,BASIC,2,INTER
8070 50 52 45 54 45 52 2C 30 00 9E 80 64 00 3A 8F E4 PRETER,0...d.:..
8080 52 45 41 44 20 44 41 54 41 20 26 20 52 45 53 54 READ DATA & REST
8090 4F 52 45 20 53 74 61 74 65 6D 65 6E 74 00 00 00 ORE Statement...
80A0 03 54 53 0B 6B 80 02 54 49 00 00 65 FF FF FF FF .TS.k..TI..e....
80B0 FF
```

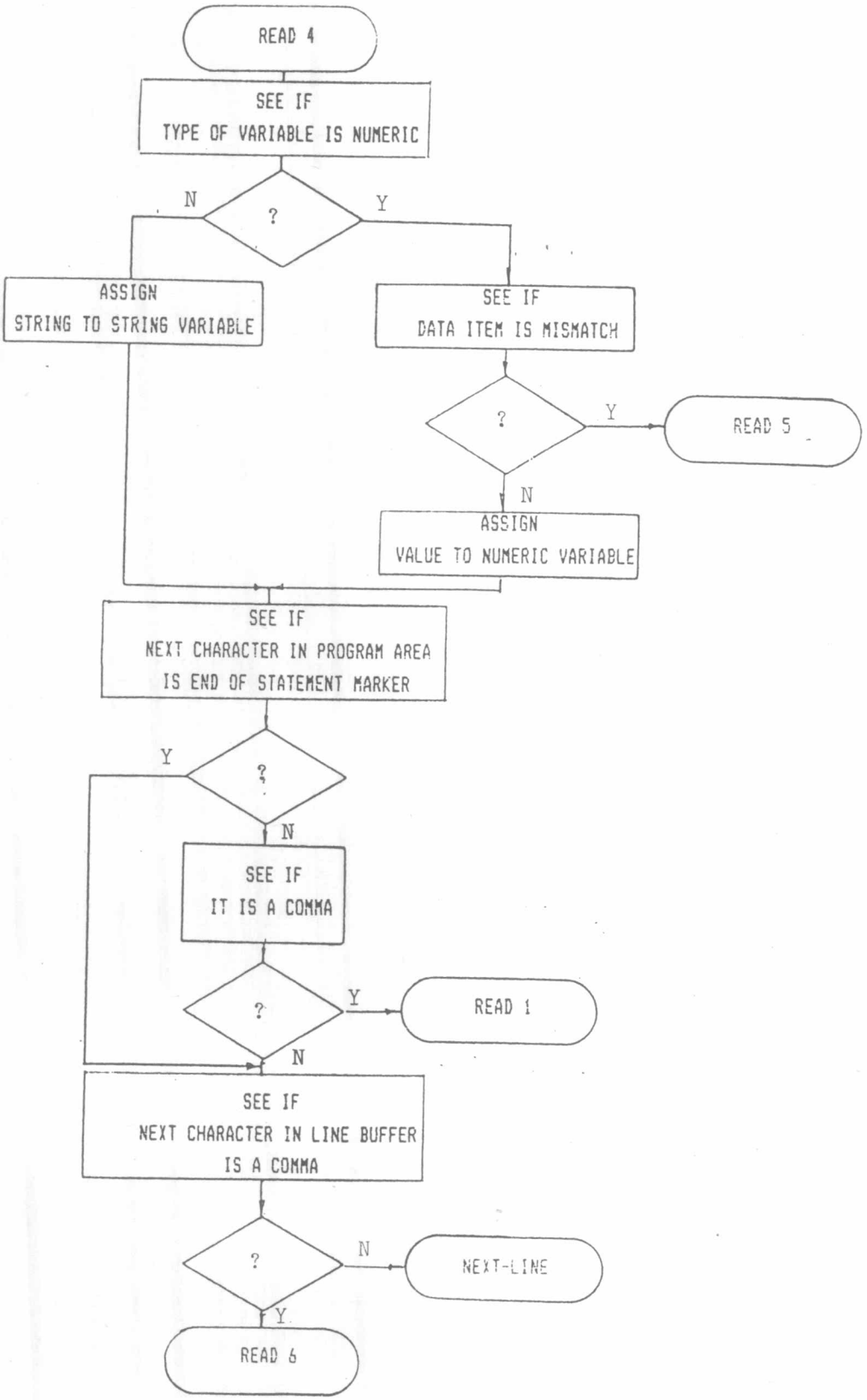
รูปที่ ๓.๔๐ แสดงการเก็บรูปแบบของคำสั่ง READ ในหน่วยความจำ



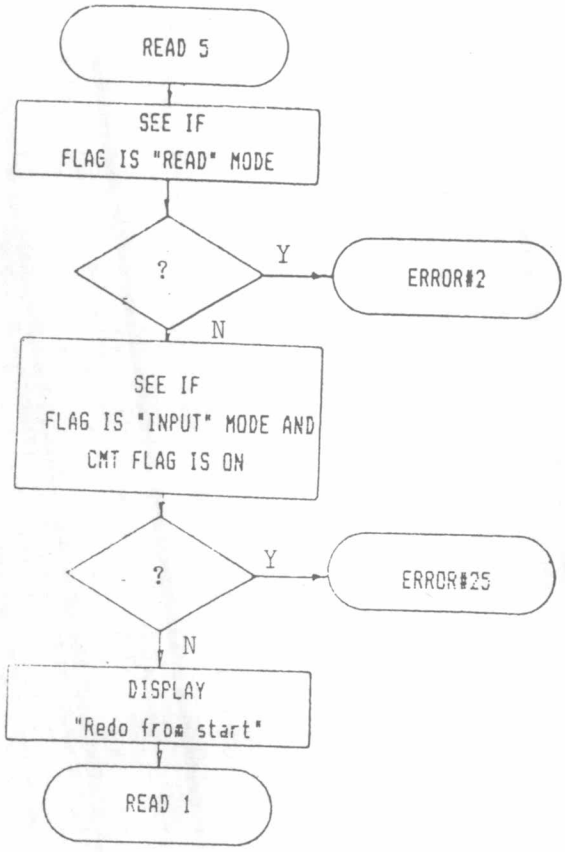
ผังงานที่ ๓.๑๗ แสดงขั้นตอนการทำงานของคำสั่ง READ



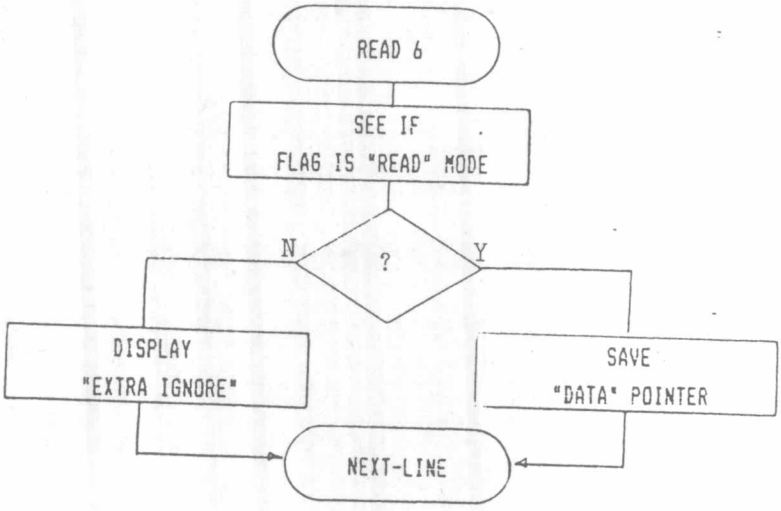
ผังงานที่ ๓.๑๗ แสดงขั้นตอนการทำงานของคำสั่ง READ (ต่อ)



ผังงานที่ ๓.๑๗ แสดงขั้นตอนการทำงานของคำสั่ง READ (ต่อ)



ผังงานที่ ๓.๑๗ แสดงขั้นตอนการทำงานของคำสั่ง READ (ต่อ)



ผังงานที่ ๓.๑๗ แสดงขั้นตอนการทำงานของคำสั่ง READ (ต่อ)

๓.๒.๑๒ RESTORE เป็นคำสั่งบังคับ ตัวชี้ตำแหน่งข้อมูล (Data pointer) ในคำสั่ง DATA ให้กลับมาเริ่มต้นใหม่ ดังนั้น สามารถใช้คำสั่ง READ อ่านข้อมูลตั้งแต่ต้นใหม่ได้ นอกจากนี้ยังสามารถเลือกหมายเลขบรรทัดของคำสั่ง DATA ที่ต้องการจะเริ่มต้นอ่านใหม่ได้ รูปแบบของคำสั่งแสดงไว้ในรูปที่ ๓.๔๑ ขั้นตอนการทำงานแสดงไว้ในผังงานที่ ๓.๑๘

Syntax

RESTORE [line number]

Example

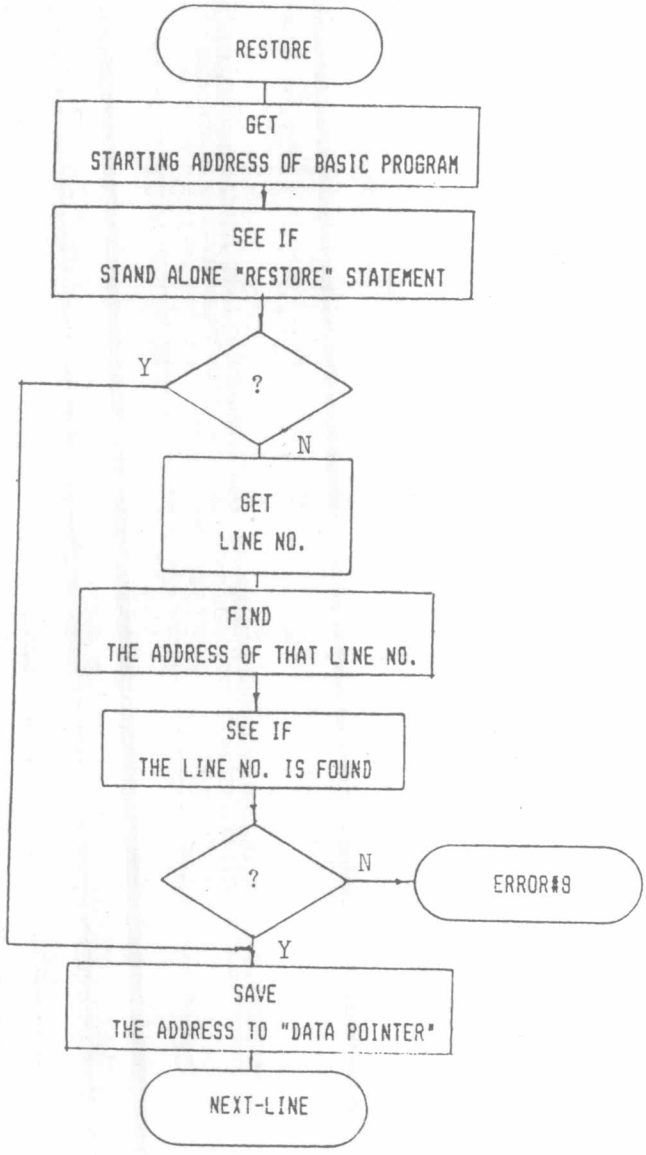
```
10 READ A,B,C
20 PRINT A;B;C
30 READ A,B,C,D,E,F
40 RESTORE 80
50 READ G,H,I
60 PRINT A;B;C;D;E;F;G;H;I
70 DATA 1,2,3,4,5,6,7,8
80 DATA 9,10,11,12
```

RESTORE resets the items of a DATA statement so they can be reread by a READ statement. It also provides the option of choosing the particular DATA statement to be read.

รูปที่ ๓.๔๑ แสดงรูปแบบของคำสั่ง RESTORE

```
10 RESTORE 100
30 'RESTORE Statement
100 DATA 1,2,3

8020 00 2B 80 0A 00 8C 20 0E 64 00 00 44 80 1E 00 3A .+.... .d..D...:
8030 8F E4 52 45 53 54 4F 52 45 20 53 74 61 74 65 6D ..RESTORE Statem
8040 65 6E 74 00 50 80 64 00 84 20 31 2C 32 2C 33 00 ent.P.d.. 1,2,3.
8050 00 00 0A 2C 43 23 2C 44 24 00 6F 80 1E 00 91 20 ...,C#,D$.o....
8060 23 F4 12 2C 41 2C 42 25 2C 43 23 2C 44 24 00 75 #..,A,B%,C#,D$.u
8070 80 28 00 81 00 00 00 14 20 20 20 20 20 20 20 20 .(...... 
```

รูปที่ ๓.๔๒ แสดงการเก็บรูปแบบของคำสั่ง RESTORE ในหน่วยความจำ



ผังงานที่ ๓.๑๘ แสดงขั้นตอนการทำงานของคำสั่ง RESTORE