& 2.44 READ

! โครงสร้างไวยากรณ์

READ [SAVE]

P วัตถุประสงค์

คำสั่งนี้ ใช้คู่กับคำสั่ง GET เพื่อรับข้อมูลไปเก็บในตัวแปร

: ตัวอย่างที่ 2.54

```
CLS
X1 = "ABCD"
X2 = 0
@ 5,5 SAY "GET X1 : " GET X1
@ 6,5 SAY "GET X2 : " GET X2
READ
```

& 2.45 RECALL

! โครงสร้างไวยากรณ์

RECALL [<ช่วงที่ต้องการ>] [FOR <เงื่อนไข>] [WHILE <เงื่อนไข>]

P วัตถุประสงค์

คำสั่งนี้ ยกเลิกการทำเครื่องหมายลบหน้าเรคอร์ด

: ตัวอย่างที่ 2.55

```
CLS
USE FILEA
DELETE FOR FIELD2 >= 1000
LIST FIELD1,FIELD2,FIELD3
OPT = " "
@ ROW()+1,10 SAY "ARE YOU SURE TO DELETE?" GET OPT
READ
IF OPT $ [NN]
  RECALL ALL
ELSE
  PACK
ENDIF
LIST FIELD1,FIELD2,FIELD3
```

& 2.46 REINDEX

! โครงสร้างไวยากรณ์

REINDEX

P วัตถุประสงค์

คำสั่งนี้ สั่งให้จัดทำแฟ้มดรรชนีที่กำลังเปิดอยู่ใหม่

: ตัวอย่างที่ 2.56

```
USE FILEA INDEX IFIELD1,IFIELD2,IFIELD3
REINDEX
SET ORDER TO 2
LIST FIELD1,FIELD2,FIELD3
```

& 2.47 RELEASE

! โครงสร้างไวยากรณ์

RELEASE <รายการชื่อตัวแปร>

RELEASE ALL [LIKE | EXCEPT <ชื่อตัวแปร>]

P วัตถุประสงค์

คำสั่งนี้ ยกเลิกตัวแปรแบบ PRIVATE และ PUBLIC

แต่ไม่ยกเลิกตัวแปรแบบ LOCAL และ STATIC

: ตัวอย่างที่ 2.57

```
LOCAL Z
USE FILEA
PRIVATE X
PUBLIC Y
X = 5
Z = 6
RELEASE X                    // ยกเลิกเฉพาะตัวแปร X
RELEASE ALL                  // ยกเลิกการจองเนื้อที่หน่วยความจำให้ Y
? Z
LIST FIELD1,FIELD2,FIELD3
```

& 2.48 RENAME

! โครงสร้างไวยากรณ์

RENAME <ชื่อแฟ้มเดิม> TO <ชื่อแฟ้มใหม่>

P วัตถุประสงค์

คำสั่งนี้ เปลี่ยนชื่อแฟ้ม

: ตัวอย่างที่ 2.58

```
RENAME TEST.TXT TO TEST1.TXT
DIR TEST1.TXT
X = "TEST.TXT"
X1 = "TEST1.TXT"
RENAME &X1 TO &X
DIR TEST.TXT
RENAME (X) TO (X1)
DIR TEST1.TXT
RENAME TEST1.TXT TO TEST.TXT
```

& 2.49 REPLACE

! โครงสร้างไวยากรณ์

```
REPLACE <ชื่อฟิลด์> WITH <ชื่อตัวแปร>
  [,<ชื่อฟิลด์ที่ 2> WITH <ชื่อตัวแปรที่ 2>...]
  [<ช่วงที่ต้องการ>] [FOR <เงื่อนไข>] [WHILE <เงื่อนไข>]
```

P วัตถุประสงค์

คำสั่งนี้ แทนค่าในฟิลด์

: ตัวอย่างที่ 2.59

```
CLS
COPY FILE FILEA.DBF TO FILEC.DBF
USE FILEA
REPLACE FIELD3 WITH 0 ALL
REPLACE FIELD1 WITH "101" FOR FIELD2 >= 1000
? "================="
LIST FIELD1,FIELD2,FIELD3
APPEND BLANK
REPLACE FIELD1 WITH "105",;
      FIELD2 WITH 1000 , FIELD3 WITH 250
? "================="
LIST FIELD1,FIELD2,FIELD3
CLOSE
```

```
COPY FILE FILEC.DBF TO FILEA.DBF
? "================="
USE FILEA
LIST FIELD1,FIELD2,FIELD3
```

& 2.50 RESTORE

! โครงสร้างไวยากรณ์

RESTORE FROM <ชื่อแฟ้มเก็บตัวแปร> [ADDITIVE]

P วัตถุประสงค์

คำสั่งนี้ เรียกตัวแปรที่เคยเก็บไว้ในแฟ้มมาใช้อีกครั้ง

: ตัวอย่างที่ 2.60

```
X1 := 5
X2 := 10
SAVE ALL TO XMEM
CLEAR ALL
RESTORE FROM XMEM
? X1,X2
```

& 2.51 RESTORE SCREEN

! โครงสร้างไวยากรณ์

RESTORE SCREEN [FROM <ตัวแปรเก็บจอภาพ>]

P วัตถุประสงค์

คำสั่งนี้ เรียกรูปแบบของจอภาพที่เคยจัดเก็บไว้มาแสดงบนจออีกครั้ง

: ตัวอย่างที่ 2.61

```
CLS
SET COLOR TO "B+/W"
? "ABC"
SAVE SCREEN                    // ถ้าไม่มีการจัดเก็บรูปแบบของจอภาพหลายครั้ง
SETCOLOR("GR+/B")
? "DEF"
INKEY(0)
RESTORE SCREEN
INKEY(0)
```

: ตัวอย่างที่ 2.62

```
CLS
SETCOLOR("W/B")
@ 0,0 TO 24,79
@ 2,1 TO 2,78
@ 1,10 SAY PADC ("COMPUTER HEADING "+DTOC(DATE()),60)
INKEY(0) ;  SAVE SCREEN TO SCR1
@ 5,10 TO 10,20
@ 7,15 TO 12,25
INKEY(0) ;  SAVE SCREEN TO SCR2
CLS
OPT = " "
@ 5,10 SAY "PRESS 1 OR 2 TO CHOOSE SCREEN : " GET OPT
READ
DO CASE
  CASE OPT = "1" ; RESTORE SCREEN FROM SCR1
  CASE OPT = "2" ; RESTORE SCREEN FROM SCR2
ENDCASE
INKEY(0)
```

& 2.52 RUN

! โครงสร้างไวยากรณ์

RUN | ! <คำสั่งภายนอก>

P วัตถุประสงค์

คำสั่งนี้ สั่งประมวลผลโปรแกรมจากภายนอก

: ตัวอย่างที่ 2.63

```
CLS
? "IF YOU WANT TO RETURN TO APPLICATION,TYPE EXIT."
? "=============================================="
INKEY(3)
RUN COMMAND                    // จะออกจากโปรแกรมไปอยู่ในส่วน DOS SHELL
? "WELCOME TO APPLICATION"
? "======================"
```

: ตัวอย่างที่ 2.64

```
RUN DIR
RUN TYPE X.PRG
```

& 2.53 SAVE

! โครงสร้างไวยากรณ์

SAVE TO <ชื่อแฟ้มเก็บตัวแปร> [ALL [LIKE | EXCEPT <ชื่อแฟ้ม>]]

P วัตถุประสงค์

คำสั่งนี้ จัดเก็บตัวแปรลงแฟ้ม เพื่อนำกลับมาใช้ในภายหลังได้

: ตัวอย่างที่ 2.65

```
X01 := "A"
X02 := 5
Y01 := 6
Y02 := "B"
SAVE ALL TO VARALL
CLEAR ALL
RESTORE FROM VARALL
? X01,X02,Y01,Y02
```

: ตัวอย่างที่ 2.66

```
CLS
? "PASSWORD SYSTEM"
? "---------------"
IF FILE("PASSF.MEM")
  RESTORE FROM PASSF
ELSE
  ? "INVALID ON PASSWORD LIST"
  USER = "    "
  PASS = "    "
  @ 5,5 SAY "    NEW USER : " GET USER
  @ 6,5 SAY "NEW PASSWORD : " GET PASS
  READ
  PASS = STR(ASC(LEFT(PASS,1)))+RIGHT(PASS,3)
ENDIF
_USER := "    "
```

```
_PASS := "    "
WHILE .T.
  @ 8,5 SAY "    YOUR USER : " GET _USER
  @ 9,5 SAY "YOUR PASSWORD : " GET _PASS
  READ
  IF _USER = USER .AND. ;
    _PASS = CHR(VAL(LEFT(PASS,LEN(PASS)-3)))+RIGHT(PASS,3)
    SAVE ALL TO PASSF
    EXIT
  ENDIF
END
```

& 2.54 SAVE SCREEN

! โครงสร้างไวยากรณ์

SAVE SCREEN [TO <ตัวแปรเก็บจอภาพ>]

P วัตถุประสงค์

คำสั่งนี้ จัดเก็บรูปแบบของจอภาพ

: ตัวอย่างที่ 2.67

```
SETCOLOR("GR+/B")
CLS
@ 5,10 TO 15,20
@ 15,5 TO 21,21 DOUBLE
SAVE SCREEN TO SCR1
SETCOLOR("W/BR+")
@ 6,7,10,20 BOX '123456789'
? '----------------------'
SAVE SCREEN TO SCR2
SETCOLOR("BG+/R+")
@ 7,10,9,15 BOX '.'
@ 9,21 CLEAR TO 18,24
SAVE SCREEN TO SCR3
SAVE ALL TO SCRF
CLEAR ALL
RESTORE FROM SCRF
```

```
RESTORE SCREEN FROM SCR1   ; INKEY(3)
RESTORE SCREEN FROM SCR2   ; INKEY(3)
RESTORE SCREEN FROM SCR3   ; INKEY(3)
```

& 2.55 SEEK

! โครงสร้างไวยากรณ์

SEEK <EXPSEARCH>

P วัตถุประสงค์

คำสั่งนี้ การค้นหาจากฟิลด์ที่ถูกกำหนดให้จัดเรียงในแฟ้มดรรชนี

: ตัวอย่างที่ 2.68

```
USE FILEA  INDEX IFIELD1
SEEK "101"
IF FOUND()
  ? FIELD1,FIELD2,FIELD3
ENDIF
```

: ตัวอย่างที่ 2.69

```
USE FILEA  INDEX IFIELD1
X = "101"
@ ROW()+1,10 SAY "WHAT ID : " GET X
READ
SEEK X
IF FOUND()
  ? FIELD1,FIELD2,FIELD3
  ? RECNO()
ENDIF
```

& 2.56 SELECT

! โครงสร้างไวยากรณ์

SELECT <เลขประจำพื้นที่> | <สมนาม>

P วัตถุประสงค์

คำสั่งนี้ เลือกพื้นที่ทำงาน

: ตัวอย่างที่ 2.70

```
USE FILEA
USE FILEB
LIST FIELD1,FIELD2,FIELD3          // แสดงข้อมูลจากแฟ้ม FILEB
```

```
SELE 1
LIST FIELD1,FIELD2,FIELD3          // แสดงข้อมูลจากแฟ้ม FILEA
SELE FILEB
? RECCOUNT()                       // แสดงจำนวนเรคอร์ดของ FILEB
```

& 2.57 SET ALTERNATE

! โครงสร้างไวยากรณ์

SET ALTERNATE TO [<ชื่อแฟ้ม> [ADDITIVE]]

SET ALTERNATE ON | OFF | <ตรรกนิพจน์>

P วัตถุประสงค์

คำสั่งนี้ ใช้สร้างสร้างแฟ้ม เพื่อรองรับการเขียนลงแฟ้มด้วยคำสั่ง ? หรือ LIST

การใช้คำสั่ง ? หรือ LIST จะเห็นผลบนจอภาพปกติ และเขียนลงแฟ้มพร้อม ๆ กัน

หากระบุชื่อแฟ้มแต่ไม่มีนามสกุล โปรแกรมจะตั้งนามสกุลชื่อ TXT ให้อัตโนมัติ

โดย ON หมายถึง เปิดแฟ้มเพื่อรับรองรับการเขียนด้วยคำสั่ง ?

และ OFF หมายถึง ไม่เปิดแฟ้มที่ระบุไว้นั้นขึ้นมา

: ตัวอย่างที่ 2.71

```
SET ALTERNATE TO XX ADDITIVE       // TO MAKE XX.TXT
SET ALTERNATE ON
? 'ABC'
? 'DEF'  ;      ? 'GHI'
@ 5,5 SAY 'ABC'                    // NO EFFECT ON XX.TXT
SET ALTERNATE TO XX.XXX            // TO MAKE XX.XXX
? 'ABC'
CLOSE ALTERNATE
```

& 2.58 SET BELL

! โครงสร้างไวยากรณ์

SET BELL ON | OFF | <ตรรกนิพจน์>

P วัตถุประสงค์

คำสั่งนี้ ใช้ควบคุมการเปล่งเสียง เมื่อป้อนข้อมูลจนล้นช่องรับข้อมูล

ในขณะที่ใช้คำสั่ง GET แต่ไม่มีผลต่อฟังก์ชัน TONE() และ CHR(7)

โดย ON หมายถึง ให้เกิดเสียงเมื่อป้อนข้อมูลจนเกินช่องที่เตรียมไว้

และ OFF หมายถึง ไม่ให้เกิดเสียงเตือน

: ตัวอย่างที่ 2.72

```
SET BELL OFF
X = 5
@ ROW(),10 SAY "GET X1 " GET X       // ไม่มีเสียงเมื่อป้อนข้อมูลมากเกิน
READ
TONE(200,20) ;  ?? CHR(7)
SET BELL ON
X = 5
@ ROW(),60 SAY "GET X2 " GET X       // มีเสียงเมื่อป้อนข้อมูลมากเกิน
READ
TONE(200,20) ;  ?? CHR(7)
```

& 2.59 SET CENTURY

! โครงสร้างไวยากรณ์

SET CENTURY ON | OFF | <ตรรกนิพจน์>

P วัตถุประสงค์

คำสั่งนี้ ทำให้การแสดงผลของปี ค.ศ. เป็น 2 หลัก หรือ 4 หลัก

โดย ON หมายถึง ให้แสดงปี ค.ศ. เป็น 4 หลัก เช่น 2002

และ OFF หมายถึง ให้แสดงปี ค.ศ. เป็น 2 หลัก เช่น 02

: ตัวอย่างที่ 2.73

```
SET CENTURY ON
? DATE()                    //  08/23/2002
SET CENTURY OFF
? DATE()                    //  08/23/02
```

& 2.60 SET COLOR

! โครงสร้างไวยากรณ์

SET COLOR | COLOUR TO

[[<สีของตัวปกติ>] [,<สีของการรับค่า>] [,<สีของเส้น>] [,<สีของพื้น>]

[<สีของส่วนที่ไม่ได้รับค่า>]] | (<ระบุสี>)

P วัตถุประสงค์

คำสั่งนี้ จะกำหนดสีให้กับส่วนต่าง ๆ ของจอภาพ

โดย สีของตัวปกติ หมายถึง สีของตัวอักษรที่พิมพ์ด้วยคำสั่ง ? หรือ SAY

สีของการรับค่า หมายถึง สีที่ปรากฏในส่วนที่กำลังรอรับค่าด้วยคำสั่ง GET

สีของเส้น หมายถึง สีของเส้นขอบ ซึ่งเป็นเส้นนอกสุด

สีของพื้น หมายถึง

สีของส่วนที่ไม่ได้รับค่า หมายถึง สีที่ปรากฏในส่วนที่ใช้คำสั่ง GET แต่ยังไม่รอรับค่า

ชื่อสี	ตัวอักษรย่อ	ตัวเลข	สำหรับจอสีเดียว
ดำ (BLACK)	N,SPACE	0	BLACK
น้ำเงิน (BLUE)	B	1	UNDERLINE
เขียว (GREEN)	G	2	WHITE
คราม (CYAN)	BG	3	WHITE
แดง (RED)	R	4	WHITE
ม่วง (MAGENTA)	RB	5	WHITE
น้ำตาล (BROWN)	GR	6	WHITE
ขาว (WHITE)	W	7	WHITE
เทา (GRAY)	N+	8	BLACK
น้ำเงินสว่าง (BRIGHT BLUE)	B+	9	BRIGHT UNDERLINE
เขียวสว่าง (BRIGHT GREEN)	G+	10	BRIGHT WHITE
ครามสว่าง (BRIGHT CYAN)	BG+	11	BRIGHT WHITE
แดงสว่าง (BRIGHT RED)	R+	12	BRIGHT WHITE
ม่วงสว่าง (BRIGHT MAGENTA)	RB+	13	BRIGHT WHITE
เหลือง (YELLOW)	GR+	14	BRIGHT WHITE
ขาวสว่าง (BRIGHT WHITE)	W+	15	BRIGHT WHITE

: ตัวอย่างที่ 2.74

```
SET COLOR TO "B/G,BG/R,RB,,N+/GR"
CLS
@ 5,5 TO 10,15 DOUBLE              //  B/G
? "TEST COLOR"                     //  B/G
X = 5
Y = 15
@ 15,5 SAY "TEST COLOR X " GET X   // BG/R
@ 16,5 SAY "TEST COLOR Y " GET Y   // N+/GR
READ
WAIT TO X
```

& 2.61 SET CONFIRM

! โครงสร้างไวยากรณ์

SET CONFIRM ON | OFF | <ตรรกนิพจน์>

P วัตถุประสงค์

คำสั่งนี้ ใช้ร่วมกับคำสั่ง GET เพราะจะควบคุมการออกจาก GET อย่างไม่ถูกต้อง
จนกว่าจะกดปุ่มเอ็นเทอร์ (ENTER) เพื่อข้ามไปยังบรรทัดต่อไป
โดย ON หมายถึง ถ้าป้อนข้อมูลจนล้นช่องแล้วจะไม่หลุดไปยังบรรทัดต่อไป
และ OFF หมายถึง ถ้าป้อนข้อมูลจนล้น จะข้ามไปบรรทัดต่อไปอัตโนมัติ

: ตัวอย่างที่ 2.75

```
SET CONFIRM ON
CLS
X = 5
@ 15,5 SAY "TEST CONFIRM " GET X
READ
```

& 2.62 SET CONSOLE

! โครงสร้างไวยากรณ์

SET CONSOLE ON | OFF | <ตรรกนิพจน์>

P วัตถุประสงค์

คำสั่งนี้ ใช้ระบุให้การแสดงผลทางจอภาพเกิดขึ้นหรือไม่
โดย ON หมายถึง ให้แสดงผลทางจอภาพ
และ OFF หมายถึง ไม่ให้แสดงผลทางจอภาพ

: ตัวอย่างที่ 2.76

```
SET CONSOLE ON
USE FILEA
LIST FIELD1 TO PRINTER      // ออกทั้งจอภาพ และเครื่องพิมพ์
SET CONSOLE OFF
LIST FIELD1 TO PRINTER      // ออกเฉพาะเครื่องพิมพ์
```

& 2.63 SET CURSOR

! โครงสร้างไวยากรณ์

SET CURSOR ON | OFF | <ตรรกนิพจน์>

P วัตถุประสงค์

คำสั่งนี้ ทำให้ตัวกระพริบ(CURSOR) ปรากฏ หรือไม่ปรากฏบนจอภาพ

โดย ON หมายถึง ปรากฏตัวกระพริบ(CURSOR) บนจอภาพ

และ OFF หมายถึง ไม่ปรากฏตัวกระพริบ(CURSOR) บนจอภาพ

: ตัวอย่างที่ 2.77

```
SET CURSOR OFF
WAIT
INKEY(5)
X = 5
@ ROW(),COL() GET X
READ
```

& 2.64 SET DATE

! โครงสร้างไวยากรณ์

SET DATE FORMAT [TO] <รูปแบบวันที่ ที่ต้องการ>

SET DATE [TO] AMERICAN | ANSI | BRITISH | FRENCH |
GERMAN | ITALIAN | JAPAN | USA

P วัตถุประสงค์

คำสั่งนี้ ใช้เลือกรูปแบบของวันที่ ที่จะนำไปใช้ในโปรแกรม

ค่าติดตั้ง	รูปแบบที่ได้
AMERICAN	MM/DD/YY
ANSI	YY.MM.DD
BRITISH	DD/MM/YY
FRENCH	DD/MM/YY
GERMAN	DD.MM.YY
ITALIAN	DD-MM-YY
JAPAN	YY/MM/DD
USA	MM-DD-YY

: ตัวอย่างที่ 2.78

```
SET DATE FORMAT TO "DD/MM/YYYY"
? DATE()                          // 28/08/2002
SET CENTURY OFF
SET DATE TO GERMAN
```

```
? DATE()                    // 28.08.02
SET DATE TO ANSI
? DATE()                    // 02.08.28
SET CENTURY ON
SET DATE TO GERMAN
? DATE()                    // 28.08.2002
SET DATE TO ANSI
? DATE()                    // 2002.08.28
```

& 2.65 SET DECIMALS

! โครงสร้างไวยากรณ์

SET DECIMALS TO [<จำนวนทศนิยม>]

P วัตถุประสงค์

คำสั่งนี้ ใช้กำหนดจำนวนทศนิยม แต่ผลการคำนวณจะได้ทศนิยมไม่เกิน 15 หลัก

: ตัวอย่างที่ 2.79

```
SET DECIMALS TO 2
? 22/7                      // 3.14
SET DECIMALS TO 20
? 22/7                      // 3.14285714285714300000
SET DECIMALS TO 30
? 22/7                      // 3.142857142857143000000000000000
SET DECIMALS TO
? 22/7                      // 3
```

& 2.66 SET DEFAULT

! โครงสร้างไวยากรณ์

SET DEFAULT TO [<ข้อความระบุไดรฟ์และไดเรกทรอรี่ที่เก็บข้อมูล>]

P วัตถุประสงค์

คำสั่งนี้ ใช้กำหนดไดเรกทรอรี่ที่เก็บข้อมูล แต่ไม่เปลี่ยนไดเรกทรอรี่

เหมือนคำสั่งเปลี่ยนไดเรกทรอรี่ในดอส (CD : CHANGE DIRECTORY)

: ตัวอย่างที่ 2.80

```
SET DEFAULT TO C:\LANGUAGE\CLIPPER5
! DIR
DIR                         // ค้นแฟ้มในไดเรกทรอรี่นี้
```

```
SET DEFAULT TO C:                    // เปลี่ยนมาใช้ไดรฟ์ C
SET DEFAULT TO ..                    // ออกไปนอกไดเรกทรอรี่ปัจจุบัน 1 ระดับ
! DIR                                // ยังอยู่ที่ LANGUAGE\CLIPPER5
DIR                                  // ไม่พบแฟ้มใน LANGUAGE
SET PATH TO C:\;C:\WINDOWS;C:\DOS;C:\LANGUAGE\CLIPPER5
SET DEFAULT TO \                     // กลับไปที่ไดเรกทรอรี่ราก
! DIR                                // ยังอยู่ที่ LANGUAGE\CLIPPER5
DIR                                  // ไม่พบแฟ้มใน C:\
```

& 2.67 SET DELETED

! โครงสร้างไวยากรณ์

SET DELETED ON | OFF | <ตรรกนิพจน์>

P วัตถุประสงค์

คำสั่งนี้ ทำให้เรคอร์ดที่ถูกทำเครื่องหมายลบ ถูกมองว่าไม่อยู่จริง ๆ

ซึ่งให้ผลคล้ายคำสั่ง SET FILTER TO !DELETE()

โดย ON หมายถึง มองว่าเรคอร์ดที่ถูกทำเครื่องหมายลบ ถูกลบไปแล้วจริง ๆ

หรือ OFF หมายถึง มองว่าเรคอร์ดที่ถูกทำเครื่องหมายลบ ยังมีอยู่ในแฟ้ม

: ตัวอย่างที่ 2.81

```
USE FILEA                            // แฟ้มนี้มีข้อมูล 4 เรคอร์ด
RECALL ALL
X = 0
DELETE RECORD 2
DELETE RECORD 4
LIST FIELD1,FIELD2,FIELD3   // เห็น * หน้าเรคอร์ดที่ 2 และ 4
COUNT TO X ; ? X                     // 4
SET DELETED ON
COUNT TO X ; ? X                     // 2
SET DELETED OFF                      // ต้องเซตให้ OFF จึงจะสั่ง RECALL ได้
RECALL ALL
LIST FIELD1,FIELD2,FIELD3            // ไม่มีเรคอร์ดที่ถูกทำเครื่องหมายลบ
COUNT TO X ; ? X                     // 4
DELETE RECORD 1
SET FILTER TO !DELETED()
COUNT TO X ; ? X                     // 3
```

& 2.68 SET DELIMITERS

! โครงสร้างไวยากรณ์

SET DELIMITERS ON | OFF | <ตรรกนิพจน์>

SET DELIMITERS TO [<ตัวอักษร> | DEFAULT]

P วัตถุประสงค์

คำสั่งนี้ ใช้กำหนดตัวอักษรที่จะอยู่ข้างหน้าและข้างหลัง
ของตัวแปรที่รับค่าด้วยคำสั่ง GET ซึ่งทำให้ผู้ใช้เห็นฟิลด์ได้ชัดเจนขึ้น
ตัวอักษรจะใช้ 2 ตัวอักษร คืออยู่ข้างหน้า และข้างหลัง อย่างละตัว
โดย ON หมายถึง ให้ใช้ตัวอักษรคลุมฟิลด์ในขณะใช้คำสั่ง GET
และ OFF หมายถึง ไม่ให้มีตัวอักษรคลุมฟิลด์

: ตัวอย่างที่ 2.82

```
X := Y := SPACE(5)
CLS
SET DELIMITERS ON
SET DELIMITERS TO ": "          // มีเฉพาะหน้าฟิลด์
@ 5,5 SAY "GET X" GET X
SET DELIMITERS TO "[ ]"         // มี [ อยู่หน้าฟิลด์ และ ] อยู่หลังฟิลด์
@ 6,5 SAY "GET Y" GET Y
READ
```

& 2.69 SET DEVICE

! โครงสร้างไวยากรณ์

SET DEVICE TO SCREEN | PRINTER

P วัตถุประสงค์

คำสั่งนี้ ใช้ระบุอุปกรณ์ที่จะแสดงผล และมักใช้คู่กับคำสั่ง SAY

: ตัวอย่างที่ 2.83

```
SET DEVICE TO PRINTER
@ 5,5 SAY "WOW"
@ 6,5 SAY "WOW"
@ 1,5 SAY "WOW"          // จะไปพิมพ์ที่บรรทัดที่ 1 ของหน้าต่อไป
EJECT
```

: ตัวอย่างที่ 2.84

```
SET PRINTER TO "OUT.TXT"
SET DEVICE TO PRINTER
```

```
? "==="                    // ให้ผลทางจอภาพเท่านั้น
@ 5,5 SAY "WOW"            // ผลลัพธ์ทั้ง 3 บรรทัดปรากฏในแฟ้ม OUT.TXT
@ 6,5 SAY "WOW"            // แต่ไม่ปรากฏบนจอภาพ
@ 8,5 SAY "WOW"
```

& 2.70 SET EPOCH

! โครงสร้างไวยากรณ์

SET EPOCH TO <ปี>

P วัตถุประสงค์

คำสั่งนี้ ทำให้โปรแกรมฉลาดขึ้นในเรื่องการมองปี จากการที่โปรแกรมมักรับปีเป็นเลข 2 หลัก แล้วเติมศตวรรษให้ หากต้องการปี 2005 ควรเติมเพียง 05 หากใช้ค่าปริยาย จะได้ปีเป็น 1905 ทันที ซึ่งค่าปริยายกำหนดให้ EPOCH เป็น 1900 ช่วงปีที่เป็นไปได้คือ 1900 ถึง 1999 ดังนั้นเมื่อเติม 12 จึงเป็น 1912

ถ้าต้องการเติม 01 แล้วให้เป็น 2001 ควรเซต EPOCH เป็น 1960 ทำให้ปีที่เป็นไปได้เมื่อเติมเลข 2 หลักคือ 1960 ถึง 2059

: ตัวอย่างที่ 2.85

```
SET DATE FORMAT TO "DD/MM/YYYY"
? CTOD("15/06/01")         // 15/06/1901
SET EPOCH TO 1960          // ช่วงของปีคือ 1960 - 2059
? CTOD("15/06/01")         // 15/06/2001
SET EPOCH TO 1995          // ช่วงของปีคือ 1995 - 2094
? CTOD("15/06/92")         // 15/06/2092
```

& 2.71 SET ESCAPE

! โครงสร้างไวยากรณ์

SET ESCAPE ON | OFF | <ตรรกนิพจน์>

P วัตถุประสงค์

คำสั่งนี้ ใช้อนุญาติหรือจำกัด การใช้ปุ่มเอสเคป (ESC KEY) กับคำสั่ง READ โดย ON หมายถึง ให้ออกจากคำสั่ง READ ด้วยการกดปุ่มเอสเคป (ESC KEY) หรือ OFF หมายถึง ไม่ให้ออกจากคำสั่ง READ ด้วยการกดปุ่มเอสเคป (ESC KEY)

: ตัวอย่างที่ 2.86

```
SET ESCAPE OFF
X := Y:= Z:= 5
CLS
```

```
@ 5,5 GET X                    // ไม่สามารถใช้ปุ่มเอสเคปตรงนี้ได้
@ 6,5 GET Y
@ 7,5 GET Z
READ
SET ESCAPE ON
@ 8,5 GET X                    // ถ้ากด ESC ตรงนี้ จะหลุดจากการรับ Y ด้วย
@ 9,5 GET Y
READ
```

& 2.72 SET EXACT

! โครงสร้างไวยากรณ์

SET EXACT ON | OFF | <ตรรกนิพจน์>

P วัตถุประสงค์

คำสั่งนี้ ใช้คุมการเปรียบเทียบในเรื่องความเท่ากันของนิพจน์ 2 ตัว

โดย ON หมายถึง เป็นจริงต่อเมื่อทั้งนิพจน์แรกและหลังเหมือนกันทุกประการ

หรือ OFF หมายถึง เป็นจริงต่อเมื่อทั้งหมดของนิพจน์หลังเท่ากับนิพจน์แรก

: ตัวอย่างที่ 2.87

```
SET EXACT OFF
? "123" = "123"                // .T.
? "123 " = "123"               // .T.
? "123" = "12"                 // .T.
? "12" = ""                    // .T.
? "123" = "123 "               // .F.
? "12" = "123"                 // .F.
? "" = "123"                   // .F.
SET EXACT ON
? "123" = "123"                // .T.
? "123 " = "123"               // .T.
? "123" = "12"                 // .F.
? "12" = ""                    // .F.
? "123" = "123 "               // .T.
? "12" = "123"                 // .F.
? "" = "123"                   // .F.
```

& 2.73 SET EXCLUSIVE

! โครงสร้างไวยากรณ์

SET EXCLUSIVE ON | OFF | <ตรรกนิพจน์>

P วัตถุประสงค์

คำสั่งนี้ ใช้ควบคุมแฟ้มโดยตรง

โดย ON หมายถึง แฟ้มไม่สามารถถูกแบ่งให้ใครใช้ได้ในขณะที่โปรแกรมนี้ใช้งาน

และ OFF หมายถึง แฟ้มสามารถถูกแบ่งให้ผู้ใช้คนอื่น ๆ ใช้ร่วมกันได้

: ตัวอย่างที่ 2.88

```
USE FILEA
SET EXCLUSIVE ON
LIST FIELD1,FIELD2,FIELD3
```

& 2.74 SET FILTER

! โครงสร้างไวยากรณ์

SET FILTER TO [<เงื่อนไข>]

P วัตถุประสงค์

คำสั่งนี้ ใช้จำกัดช่วงข้อมูลที่นำมาใช้ ให้มีขอบเขตที่แคบลงมา เช่นต้องการ

ข้อมูลของคนที่เงินเดือน มากกว่า 5000 บาทเป็นต้น

: ตัวอย่างที่ 2.89

```
USE FILEA
SET FILTER TO FIELD2 > 5000
LIST FIELD1,FIELD2,FIELD3
SET FILTER TO                    // ยกเลิกการใช้ตัวกรอง
LIST FIELD1,FIELD2,FIELD3
SET FILTER TO DELETE()           // เลือกเฉพาะข้อมูลที่ถูกทำเครื่องหมายลบ
LIST FIELD1,FIELD2,FIELD3
```

& 2.75 SET FIXED

! โครงสร้างไวยากรณ์

SET FIXED ON | OFF | <ตรรกนิพจน์>

P วัตถุประสงค์

คำสั่งนี้ ใช้ควบคุมให้การกำหนดตำแหน่งทศนิยมทำงาน หรือใช้ค่าโดยปริยายแทน

โดย ON หมายถึง ให้ใช้จำนวนตำแหน่งทศนิยมตามคำสั่ง SET DECIMALS

หรือ OFF หมายถึง ให้จำนวนตำแหน่งทศนิยมเป็นไปตามค่าปริยาย

: ตัวอย่างที่ 2.90

```
SET FIXED ON
SET DECIMALS TO 5
X = 5
Y = 2
Z = X / Y
? Z                                    // 2.50000
```

& 2.76 SET FORMAT

! โครงสร้างไวยากรณ์

SET FORMAT TO [<ชื่อโปรแกรมย่อย>[.<ส่วนขยาย>]]

P วัตถุประสงค์

คำสั่งนี้ ใช้กำหนดโปรแกรมย่อยที่จะทำงาน มักใช้ร่วมกับคำสั่ง READ ทำให้สามารถนำไปเลือกรูปแบบที่จะใช้รับข้อมูลได้

ในบางกรณีอาจเห็นผลของคำสั่ง DO ให้ผลเหมือนคำสั่ง SET FORMAT แต่ตามตัวอย่างที่ข้างล่างไม่สามารถใช้คำสั่ง DO ได้ เพราะคำสั่ง DO จะไม่ส่งรูปแบบคืนมาให้ READ ทำงานในลูปได้อีก ทำให้โปรแกรมค้าง (HANG)

: ตัวอย่างที่ 2.91

```
CLS
FRM = 1
X = 5
Y = 10
CLS
SET DELIMITERS TO "[]"
SET DELIMITERS ON
@ 4,5 SAY "CHOOSE FORM 1 TO 3 " GET FRM PICT "9"
READ                                   // เลือกรูปแบบที่จะป้อนข้อมูลได้
DO CASE
  CASE FRM = 1 ;  SET FORMAT TO TESTPROC1
  CASE FRM = 2 ;  SET FORMAT TO TESTPROC2
  CASE FRM = 3 ;  SET FORMAT TO TESTPROC3
ENDCASE
WHILE LASTKEY() != 27
  READ
```

```
  ? X,Y,X+Y,X-Y,X*Y,X/Y
END
PROCEDURE TESTPROC1
  @ 5,5  SAY "GET X" GET X
  @ 6,5  SAY "GET Y" GET Y
RETURN
PROCEDURE TESTPROC2
  @ 5,5  SAY "GET X" GET X
  @ 5,55 SAY "GET Y" GET Y
RETURN
PROCEDURE TESTPROC3
  @ 5,4 TO 8,35 DOUBLE
  @ 6,5  SAY "GET X" GET X
  @ 7,5  SAY "GET Y" GET Y
RETURN
```

& 2.77 SET FUNCTION

! โครงสร้างไวยากรณ์

SET FUNCTION <เลขที่ปุ่มฟังก์ชัน> TO <ข้อความ>

P วัตถุประสงค์

คำสั่งนี้ ใช้กำหนดข้อความประจำปุ่มฟังก์ชัน เมื่อกดปุ่มฟังก์ชันแล้วจะนำข้อความไปป้อนในฟิลด์ที่กำลังรอรับค่าอยู่ได้ ทำให้สะดวกในการใส่ข้อความที่มักต้องใช้บ่อย ๆ เช่นชื่อบริษัท หรือชื่อผู้จัดการเป็นต้น

เลขประจำปุ่มฟังก์ชัน	ชื่อปุ่มฟังก์ชัน
1-10	F1 - F10
11-20	SHIFT-F1 - SHIFT-F10
21-30	CTRL-F1 - CTRL-F10
31-40	ALT-F1 - ALT-F10

: ตัวอย่างที่ 2.92

```
SET KEY -2 TO EXITPROC              // -2 หมายถึงปุ่ม F3
SET FUNCTION 5 TO "YONOK COLLEGE"
SET FUNCTION 6 TO "DR.NIRUND JIVASANTIKARN"
CLS
X = SPACE(25)                       // กด F5 จะส่งคำว่า YONOK COLLEGE ลงใน GET
```

```
CNT = 0                                          // ระบุคำที่ใช้บ่อย ๆ ในปุ่มฟังก์ชันเพื่อความสะดวก
@ 1,1 SAY "F3:EXIT  F5:YNK   F6:NIRUND"
DO WHILE .T.
  CNT++
  @ CNT+1,10 SAY STR(CNT,3)+" TEST FUNCTION :" GET X
  READ
ENDDO
PROCEDURE EXITPROC
   QUIT
RETURN
```

& 2.78 SET INDEX

! โครงสร้างไวยากรณ์

SET INDEX TO <รายการชื่อแฟ้มดรรชนี>

P วัตถุประสงค์

คำสั่งนี้ ใช้เปิดแฟ้มดรรชนี ในกรณีที่เปิดแฟ้มแล้วไม่ได้เปิดแฟ้มดรรชนีพร้อมกัน
เพราะปกติโปรแกรมสามารถเปิดแฟ้มพร้อมเปิดแฟ้มดรรชนีได้พร้อมกัน
คำสั่งนี้จึงใช้เปิดแฟ้มดรรชนีใหม่ หรือยกเลิกการเปิดแฟ้มดรรชนีได้

: ตัวอย่างที่ 2.93

```
USE FILEA INDEX IFIELD1,IFIELD2
LIST FIELD1,FIELD2,FIELD3
SET ORDER TO 2
LIST FIELD1,FIELD2,FIELD3
```

: ตัวอย่างที่ 2.94

```
USE FILEA
LIST FIELD1,FIELD2,FIELD3
SET INDEX TO IFIELD1,IFIELD2
LIST FIELD1,FIELD2,FIELD3
SET INDEX TO
LIST FIELD1,FIELD2,FIELD3
```

& 2.79 SET INTENSITY

! โครงสร้างไวยากรณ์

SET INTENSITY ON | OFF | <ตรรกนิพจน์>

P วัตถุประสงค์

คำสั่งนี้ ใช้คุมการใช้สีของคำสั่ง GET ที่จะใช้สีของการรับค่า หรือสีมาตรฐาน
โดย ON หมายถึง ให้คำสั่ง GET ใช้สีของการรับค่า หรือส่วนที่ยังไม่ได้รับค่า
และ OFF หมายถึง ให้คำสั่ง GET ใช้สีมารตฐานทุกกรณี

: ตัวอย่างที่ 2.95

```
SET COLOR TO "B/G,BG/R,RB,,N+/GR"
CLS
X = 5
Y = 15
@ 13,5 SAY "TEST COLOR X " GET X    // BG/R เมื่อรอรับค่า
@ 14,5 SAY "TEST COLOR Y " GET Y   // N+/GR
READ
SET INTENSITY OFF                   // ยกเลิกการเซตสีในส่วนของคำสั่ง GET
CLS                                 // โดยให้ส่วนของคำสั่ง GET มาใช้สีมาตรฐาน
@ 15,5 SAY "TEST COLOR X " GET X   // B/G
@ 16,5 SAY "TEST COLOR Y " GET Y   // B/G
READ
```

& 2.80 SET KEY

! โครงสร้างไวยากรณ์

SET KEY <รหัสการกดแป้นพิมพ์> TO [<ชื่อโปรแกรมย่อย>]

P วัตถุประสงค์

คำสั่งนี้ ใช้กำหนดโปรแกรมย่อยประจำปุ่มฟังก์ชัน นั่นหมายถึงเมื่อ
กดปุ่มฟังก์ชันจะทำให้โปรแกรมย่อยที่ระบุไว้ทำงานทันที

: ตัวอย่างที่ 2.96

```
SET KEY 28 TO HELPPROC          // 28 หมายถึงปุ่ม F1
SET KEY -1 TO CALPROC           // -1 หมายถึงปุ่ม F2
SET KEY -2 TO EXITPROC          // -2 หมายถึงปุ่ม F3
CLS
X = SPACE(25)
CNT = 0
@ 1,1 SAY "F1:HELP"
DO WHILE .T.
  CNT++
```

```
    @ CNT+1,10 SAY STR(CNT,3)+" TEST FUNCTION :" GET X
    READ
  ENDDO
  PROCEDURE HELPPROC
    X = ALERT("F1:HELP  F2:PUT CAL. VALUE  F3:EXIT",{"OK"})
    // เมื่อแสดงส่วนช่วยเหลือแล้ว ถ้ากด  Enter  จะส่ง 1 กลับไป
    // แต่ถ้ากดปุ่ม ESC จะส่งเลข 0 กลับไปให้ X
  RETURN
  PROCEDURE EXITPROC
    QUIT
  RETURN
  PROCEDURE CALPROC
    IF X < 5 ; X = CNT * 100  ; ENDIF
    IF X = 5 ; X = CNT * 500  ; ENDIF
    IF X > 5 ; X = CNT * 1000 ; ENDIF
  RETURN
```

& 2.81 SET MARGIN

! โครงสร้างไวยากรณ์

```
SET MARGIN TO [<เลขระบุกั้นหน้า>]
```

P วัตถุประสงค์

คำสั่งนี้ ใช้กำหนดตำแหน่งกั้นหน้าเพื่อพิมพ์รายงานออกเครื่องพิมพ์

: ตัวอย่างที่ 2.97

```
USE FILEA
SET MARGIN TO 8
LIST FIELD1,FIELD2 TO PRINTER       // ได้กั้นหน้าเป็น 8
SET MARGIN TO 5
LIST SECOND(),FIELD1 TO PRINTER  // ได้กั้นหน้าเป็น 5
EJECT                                // ให้ผลทั้งเครื่องพิมพ์และจอภาพ
SET DEVICE TO SCREEN
```

& 2.82 SET MESSAGE

! โครงสร้างไวยากรณ์

```
SET MESSAGE TO [<แถว> [CENTER | CENTRE]]
```

P วัตถุประสงค์

คำสั่งนี้ ใช้ระบุแถวที่จะแสดงข่าวสารจากที่เคยระบุข่าวสารในคำสั่ง PROMPT

: ตัวอย่างที่ 2.98

```
SET MESSAGE TO 22 CENTRE
SET WRAP ON
MSS = "แปลว่า "
@ 6,5 PROMPT "ANT"    MESSAGE MSS+"มด"
@ 7,5 PROMPT "BAT"    MESSAGE MSS+"ค้างคาว"
@ 8,5 PROMPT "CAT"    MESSAGE MSS+"แมว"
@ 9,5 PROMPT "DOG"    MESSAGE MSS+"สุนัข"
MENU TO OPT
DO CASE
  CASE OPT = 0
     QUIT
  CASE OPT > 0
     ? "O.K."
ENDCASE
```

& 2.83 SET ORDER

! โครงสร้างไวยากรณ์

SET ORDER TO [<เลขลำดับแฟ้มดรรชนี>]

P วัตถุประสงค์

คำสั่งนี้ ใช้เลือกแฟ้มดรรชนีที่จะมาทำงาน ซึ่งจัดเรียงตามฟิลด์ที่กำหนด

คำสั่งนี้จึงเป็นคำสั่งที่เลือกแฟ้มดรรชนีที่เคยเปิดรอไว้แล้วเท่านั้น

: ตัวอย่างที่ 2.99

```
USE FILEA
INDEX ON FIELD1 TO IFIELD1
INDEX ON FIELD2 TO IFIELD2
SET INDEX TO IFIELD1,IFIELD2
SET ORDER TO 2
LIST FIELD1,FIELD2,FIELD3
SET ORDER TO 1
LIST FIELD1,FIELD2,FIELD3
```

& 2.84 SET PATH

! โครงสร้างไวยากรณ์

SET PATH TO [<รายการระบุเส้นทาง>]

P วัตถุประสงค์

คำสั่งนี้ ใช้ระบุเส้นทางที่จะเรียกใช้แฟ้ม เมื่อไม่พบในไดเรกทรอรี่ปัจจุบัน

: ตัวอย่างที่ 2.100

```
? FILE("COMMAND.COM")          // .F.
SET PATH TO C:\;C:\DOS;C:\WINDOWS
? FILE("COMMAND.COM")          // .T.
? FILE("FORMAT.COM")           // .T.
TYPE AUTOEXEC.BAT
// แฟ้ม AUTOEXEC.BAT อยู่ใน C:\ ที่หาพบ เพราะเซตเส้นทางไว้
```

& 2.85 SET PRINTER

! โครงสร้างไวยากรณ์

SET PRINTER ON | OFF | <ตรรกนิพจน์>

SET PRINTER TO [<ชื่ออุปกรณ์> | <ชื่อแฟ้ม> [ADDITIVE]]

P วัตถุประสงค์

คำสั่งนี้ ใช้ระบุปลายทางของผลลัพธ์เป็นเครื่องพิมพ์ หรือแฟ้ม

โดย ON หมายถึง ให้แสดงผลทางเครื่องพิมพ์ หรือนำข้อมูลออกแฟ้มได้

และ OFF หมายถึง ไม่อนุญาติให้ใช้เครื่องพิมพ์ ด้วยคำสั่ง SET PRINTER TO

: ตัวอย่างที่ 2.101

```
SET PRINTER ON
SET CONSOLE OFF
USE FILEA
WHILE .NOT. EOF()
  ? FIELD1,FIELD2,FIELD3       // ไม่แสดงผลทางจอภาพ
  SKIP
END
```

: ตัวอย่างที่ 2.102

```
SET PRINTER TO OUT.TXT
SET PRINTER ON
SET CONSOLE OFF
```

```
USE FILEA
WHILE .NOT. EOF()
  ? FIELD1,FIELD2,FIELD3          // นำผลลัพธ์ออกทางแฟ้มเท่านั้น
  SKIP
END
```

& 2.86 SET PROCEDURE

! โครงสร้างไวยากรณ์

SET PROCEDURE TO [<ชื่อโปรแกรมย่อย>[.<ส่วนขยาย>]]

P วัตถุประสงค์

คำสั่งนี้ ใช้ระบุโปรแกรมย่อยจากภายนอกที่จะนำมาแปล (COMPILE) เพราะบางครั้งโปรแกรมต้องเรียกใช้โปรแกรมจากภายนอก แต่ไม่ได้ระบุชื่อของโปรแกรมไว้แต่แรก

: ตัวอย่างที่ 2.103

```
SET PROCEDURE TO Y.PRG
PROCEXT = "Y"
DO &PROCEXT
// ถ้าโปรแกรมนี้ไม่ใช้คำสั่ง SET PROCEDURE จะประมวลผลไม่ผ่าน
// เพราะไม่ได้เรียกโปรแกรม Y มาแปลพร้อม ๆ กัน
```

& 2.87 SET RELATION

! โครงสร้างไวยากรณ์

SET RELATION TO [<นิพจน์> | <ลำดับของเรคอร์ด> INTO <สมนาม>]
[, [TO] <นิพจน์ที่ 2> | <ลำดับของเรคอร์ดที่ 2> INTO <สมนามที่ 2>...]
[ADDITIVE}

P วัตถุประสงค์

คำสั่งนี้ ใช้เชื่อมความสัมพันธ์ของแฟ้มอัตโนมัติ ทำให้เชื่อมกันและถูกเรียกใช้ได้อย่างง่ายดาย โดยไม่ต้องใช้คำสั่งค้นหาให้ยุ่งยาก

: ตัวอย่างที่ 2.104

```
USE INVF INDEX IINV NEW
USE ZIPF INDEX IZIP NEW
USE CUSF
SET RELATION TO CUSID INTO INVF , ZIPCODE INTO ZIPF
LIST CUSID , INVF->INVNUM , INVF->DATE , ZIPF->PROVINCE
```

& 2.88 SET SCOREBOARD

! โครงสร้างไวยากรณ์

SET SCOREBOARD ON | OFF | <ตรรกนิพจน์>

P วัตถุประสงค์

คำสั่งนี้ ใช้อนุญาติให้แสดงข้อผิดพลาดจากคำสั่ง READ หรือ MEMOEDIT ในบรรทัด 0

เช่นการใส่ค่าในคำสั่ง READ เกิดช่วงที่ระบุ จะแสดงข้อผิดพลาดทางในบรรทัด 0

โดย ON หมายถึง อนุญาติให้ใช้บรรทัด 0 แสดงข้อผิดพลาด

และ OFF หมายถึง ไม่แสดงข้อผิดพลาดในบรรทัด 0

: ตัวอย่างที่ 2.105

```
X = 5
@ 5,5 GET X RANGE 0 , 99
READ
SET SCOREBOARD OFF              // ไม่แสดงข้อผิดพลาด
@ 6,5 GET X RANGE 0 , 99        // ถ้าค่าผิด จะใช้ค่าปริยายแทนค่าที่ผิด
READ
```

& 2.89 SET SOFTSEEK

! โครงสร้างไวยากรณ์

SET SOFTSEEK ON | OFF | <ตรรกนิพจน์>

P วัตถุประสงค์

คำสั่งนี้ ใช้ควบคุมเกี่ยวกับการค้นหาเรคอร์ด หากค้นหาไม่พบ

โดย ON หมายถึง ถ้าค้นหาไม่พบ ตัวชี้จะไปชี้ที่เรคอร์ดมีค่าสูงกว่า

และ OFF หมายถึง ถ้าค้นหาไม่พบ ตัวชี้จะไปชี้ที่ท้ายเรคอร์ดสุดท้าย

: ตัวอย่างที่ 2.106

```
USE FILEA INDEX IFIELD2
SEEK 1001
IF .NOT. FOUND()
  ? FIELD2                      // 0 ท้ายเรคอร์ดสุดท้ายไม่มีค่า
ENDIF
SET SOFTSEEK ON
SEEK 1001
IF .NOT. FOUND()
  ? FIELD2                      // 6500 ค่าต่อมาที่สูงกว่า 1001
ENDIF
```

& 2.90 SET TYPEAHEAD

! โครงสร้างไวยากรณ์

SET TYPEAHEAD TO <จำนวนเลขระบุขนาด>

P วัตถุประสงค์

คำสั่งนี้ ใช้กำหนดขนาดหน่วยความจำที่เก็บค่าของแป้นพิมพ์ เมื่อผู้ใช้กด
แป้นพิมพ์จนหน่วยความจำรับไปประมวลผลไม่ทัน จึงต้องเก็บค่าที่เกินไว้
ในหน่วยความจำสำหรับแป้นพิมพ์ (KEYBOARD BUFFER)
โดยค่าที่เป็นไปได้อยู่ระหว่าง 0 ถึง 4096

: ตัวอย่างที่ 2.107

```
SET TYPEAHEAD TO 128
CLS                                  // เก็บค่าลงหน่วยความจำของแป้นพิมพ์
KEYBOARD "ABCDEF"+CHR(13)+CHR(13)+CHR(65)+CHR(66)
? NEXTKEY(),INKEY()                  //    97    97
? NEXTKEY(),INKEY()                  //    98    98
? NEXTKEY(),INKEY()                  //    99    99
? NEXTKEY(),NEXTKEY()                //   100   100
WAIT ""                              // D
ACCEPT TO TESTKEY1                   // EF
INKEY(0)                             // A
WAIT ""                              //
? NEXTKEY(),LASTKEY()                //    66    65
```

& 2.91 SET WRAP

! โครงสร้างไวยากรณ์

SET WRAP ON | OFF | <ตรรกนิพจน์>

P วัตถุประสงค์

คำสั่งนี้ อนุญาติให้การใช้คำสั่ง PROMPT สะดวกขึ้น เพราะเมื่อเลือก
ตัวเลือกไปถึงตัวบนสุดแล้ว กดลูกศรขึ้นสามารถคุมให้อยู่กับที่ หรือให้
ไปยังตัวล่างสุด ในทางกลับกันเมื่ออยู่ที่ตัวล่างสุด กดลูกศรลงอีกครั้ง
จะมาชี้ยังตัวแรก หรือยังคงอยู่ที่ตัวล่างสุดได้
โดย ON หมายถึง อนุญาติให้มองเห็นตัวเลือกตัวแรก และสุดท้ายต่อกัน
และ OFF หมายถึง ไม่ให้ตัวเลือกแรก และสุดท้ายต่อกัน

: ตัวอย่างที่ 2.108

```
SET MESSAGE TO 22 CENTRE
SET WRAP ON
MSS = "แปลว่า "
@ 6,5 PROMPT "ANT"    MESSAGE MSS+"มด"
@ 7,5 PROMPT "BAT"    MESSAGE MSS+"ค้างคาว"
@ 8,5 PROMPT "CAT"    MESSAGE MSS+"แมว"
@ 9,5 PROMPT "DOG"    MESSAGE MSS+"สุนัข"
MENU TO OPT
DO CASE
  CASE OPT = 0
     QUIT
  CASE OPT > 0
     ? "O.K."
ENDCASE
```

& 2.92 SKIP

! โครงสร้างไวยากรณ์

SKIP [<จำนวนเรคอร์ด>] [ALIAS <สมนาม> | <เลขระบุพื้นที่ทำงาน>]

P วัตถุประสงค์

คำสั่งนี้ เลื่อนตัวชี้ไปเรคอร์ดต่อไป

: ตัวอย่างที่ 2.109

```
USE FILEA
DO WHILE .NOT. EOF()
  ? FIELD1,FIELD2,FIELD3
  SKIP
ENDDO
```

: ตัวอย่างที่ 2.110

```
USE FILEA
GO BOTTOM
DO WHILE .NOT. BOF()
  ? FIELD1,FIELD2,FIELD3
  SKIP -1
ENDDO
```

& 2.93 SORT

! โครงสร้างไวยากรณ์

SORT TO <ชื่อแฟ้ม> ON <ชื่อฟิลด์> [/[A | D][C]]
[, <ชื่อฟิลด์> [/[A | D][C]...]
[<ช่วงที่ต้องการ>] [WHILE <เงื่อนไข>] [FOR <เงื่อนไข>]

P วัตถุประสงค์

คำสั่งนี้ จัดเรียงตามฟิลด์แล้วเก็บลงแฟ้มใหม่

: ตัวอย่างที่ 2.111

```
USE FILEA
SORT ON FIELD2 TO SFILEA
CLOSE ALL
USE SFILEA
LIST FIELD1,FIELD2,FIELD3
```

& 2.94 STORE

! โครงสร้างไวยากรณ์

STORE <นิพจน์> TO <รายการตัวแปร>

<ตัวแปร> = <นิพจน์>

<ตัวแปร> := [<ตัวแปรที่ 2 := ...] <นิพจน์>

P วัตถุประสงค์

คำสั่งนี้ จัดเก็บค่าลงในตัวแปร

: ตัวอย่างที่ 2.112

```
STORE  5 TO A,B,C
X := Y := Z := 'QC'
AX1 = 2
STORE AX1 TO AX2
? A,Z,B,Y,C,X
? AX1,AX2
```

& 2.95 SUM

! โครงสร้างไวยากรณ์

SUM <ชื่อฟิลด์> TO <รายการตัวแปร>
[<ช่วงที่ต้องการ>] [WHILE <เงื่อนไข>] [FOR <เงื่อนไข>]

P วัตถุประสงค์

คำสั่งนี้ หาผลรวมของฟิลด์

: ตัวอย่างที่ 2.113

```
USE FILEA
COUNT TO CNT
SUM FIELD2 TO X1
SUM FIELD2,FIELD3 TO SUM1,SUM2
? SUM1/CNT                    // 2271.25
? SUM2/CNT                    //  367.50
```

& 2.96 TEXT

! โครงสร้างไวยากรณ์

```
TEXT [TO PRINTER] [TO FILE <ชื่อแฟ้ม>]
  <ข้อความ>...
ENDTEXT
```

P วัตถุประสงค์

คำสั่งนี้ แสดงข้อความออกทางอุปกรณ์แสดงผล

: ตัวอย่างที่ 2.114

```
TEXT
  THIS IS THE TEST OF TEXT COMMAND.
  ================================
  ABCD
  EFGH
  IJKL                        // ข้อความทั้ง 5 บรรทัดจะแสดงออกทางจอภาพ
ENDTEXT
```

: ตัวอย่างที่ 2.115

```
XDATE := DTOC(DATE())
TEXT TO PRINTER
  TODAY IS &XDATE
  HAPPY BIRTHDAY TO YOU
ENDTEXT
EJECT
```

& 2.97 TOTAL

! โครงสร้างไวยากรณ์

TOTAL ON <ชื่อฟิลด์> [FIELDS <รายการชื่อฟิลด์>] TO <ชื่อแฟ้ม>
[<ช่วงที่ต้องการ>] [WHILE <เงื่อนไข>] [FOR <เงื่อนไข>]

P วัตถุประสงค์

คำสั่งนี้ หาผลรวม โดยรวมค่าเฉพาะที่เป็นกลุ่มเดียวกัน เก็บลงแฟ้มใหม่ ในฟิลด์เดิม

: ตัวอย่างที่ 2.116

```
USE FILEF INDEX IFIELD1
// จับกลุ่มตาม FIELD1 โดยหาผลรวมของ FIELD2
TOTAL ON &(INDEXKEY(0)) FIELDS FIELD2 TO TFILEF1
// จับกลุ่มตาม FIELD1 โดยหาผลรวมของ FIELD2 และ FIELD3
TOTAL ON &(INDEXKEY(0)) FIELDS FIELD2,FIELD3 TO TFILEF2
```

& 2.98 TYPE

! โครงสร้างไวยากรณ์

TYPE <ชื่อแฟ้ม> [TO PRINTER] [TO FILE <ชื่อแฟ้มผลลัพธ์>]

P วัตถุประสงค์

คำสั่งนี้ แสดงแฟ้มตัวอักษรออกทางอุปกรณ์แสดงผล

: ตัวอย่างที่ 2.117

```
TYPE X.PRG
INKEY(0)
TYPE X.PRG TO PRINTER
```

& 2.99 UNLOCK

! โครงสร้างไวยากรณ์

UNLOCK [ALL]

P วัตถุประสงค์

คำสั่งนี้ ยกเลิกการจองแฟ้ม

: ตัวอย่างที่ 2.118

```
USE FILEA
USE FILEA NEW
SELE 1
UNLOCK
CLOSE
```

```
SELE 2
USE FILEA NEW
DELETE
LIST FIELD1,FIELD2,FIELD3
INKEY(0)
```

& 2.100 UPDATE

! โครงสร้างไวยากรณ์

```
UPDATE FROM <สมนาม>
 ON <ชื่อฟิลด์> [RANDOM]
 REPLACE <ชื่อฟิลด์> WITH <ชื่อตัวแปร>
      [,<ชื่อฟิลด์ที่ 2> WITH <ชื่อตัวแปรที่ 2>...]
```

P วัตถุประสงค์

คำสั่งนี้ ปรับปรุงแฟ้มข้อมูล โดยใช้อีกแฟ้มหนึ่งมาแก้ไข

: ตัวอย่างที่ 2.119

```
USE FILEU                          // มีโครงสร้างเหมือนแฟ้ม FILEA
USE FILEA INDEX IFIELD1 NEW   ;      REINDEX
UPDATE FROM FILEU ON FIELD1 RANDOM;
     REPLACE FIELD2 WITH FIELD2 + FILEU->FIELD2,;
          FIELD3 WITH FIELD3 + FILEU->FIELD3
```

& 2.101 USE

! โครงสร้างไวยากรณ์

```
USE  [<ชื่อแฟ้ม>
  [INDEX <รายการชื่อแฟ้มดรรชนี>]
  [ALIAS <สมนาม>] [EXCLUSIVE | SHARED]
  [NEW] [READONLY]
```

P วัตถุประสงค์

คำสั่งนี้ เปิดแฟ้มมาประมวลผล

: ตัวอย่างที่ 2.220

```
USE FILEA READONLY
// DELETE   ใช้คำสั่งนี้ไม่ได้ เพราะแฟ้มถูกอนุญาติให้อ่านเท่านั้น
LIST FIELD1,FIELD2,FIELD3
? RECCOUNT(),RECNO()               // 4    5
```

& 2.102 WAIT

! โครงสร้างไวยากรณ์

WAIT [<ข้อความ>] [TO <ชื่อตัวแปร>]

P วัตถุประสงค์

คำสั่งนี้ หยุดรอให้กดปุ่มใด ๆ

: ตัวอย่างที่ 2.221

```
USE FILEA
WHILE .NOT. EOF()
   DISPLAY FIELD1,FIELD2,FIELD3
   WAIT                          // แสดงคำว่า PRESS ANY KEY TO CONTINUE...
   SKIP
END
```

: ตัวอย่างที่ 2.222

```
WAIT "PRESS A TO D" TO GETCHR    // แสดงตัวอักษรที่กดให้เห็น
IF GETCHR $ [ABCDABCD]
  ? "YOU ARE RIGHT."
ELSE
  ? "YOU ARE WRONG."
ENDIF
```

& 2.103 ZAP

! โครงสร้างไวยากรณ์

ZAP

P วัตถุประสงค์

คำสั่งนี้ ลบเรคอร์ดทั้งหมดจากแฟ้ม โดยไม่จำเป็นต้องใช้คำสั่ง PACK

: ตัวอย่างที่ 2.223

```
USE FILEA
ZAP
? RECCOUNT()                     // 0
CLOSE
COPY FILE FABAK.DBF TO FILEA.DBF
USE FILEA.DBF
LIST FIELD1,FIELD2,FIELD3
? RECCOUNT()                     // 4
```

ยิ่งเรียน ยิ่งรู้ว่าไม่รู้
จึงต้องพยายามมากยิ่งขึ้น
เพื่อลดความไม่รู้ให้มากที่สุด
และนำความรู้ไปหาเลี้ยงชีพ .. ต่อไป