& 3.89 FKLABEL()

! รูปแบบ

FKLABEL(<เลขระบุฟังก์ชัน>) --> ชื่อปุ่มฟังก์ชัน

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งชื่อฟังก์ชันเป็นตัวอักษร

: ตัวอย่างที่ 3.103

```
INKEY(0)
? FKLABEL(1)                    // F1
? FKLABEL(20)                   // F20
? FKLABEL(35)                   // F35
? FKMAX()                       //      40
```

& 3.90 FKMAX()

! รูปแบบ

FKMAX() --> เลขระบุฟังก์ชันสูงสุดที่เป็นไปได้

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งค่าสูงสุดของปุ่มฟังก์ชันที่เป็นไปได้ ไม่ให้เกินข้อกำหนด

: ตัวอย่างที่ 3.104

```
INKEY(0)    ;    ? FKMAX()          // 40 เลขฟังก์ชันที่ใช้ได้สูงสุด
```

& 3.91 FLOCK()

! รูปแบบ

FLOCK() --> ผลการจองแฟ้ม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ไม่อนุญาติให้มีการใช้ฐานข้อมูลร่วมกัน มีประโยชน์มากกับระบบเครือข่าย

: ตัวอย่างที่ 3.105

```
USE FILEA SHARED                // เปิดแฟ้มในพื้นที่ 1 และไม่จองไว้
USE FILEA SHARED NEW            // เปิดแฟ้มในพื้นที่ 2 และไม่จองไว้
? FIELD1                        // 101
? FLOCK()                       // .T. จองแฟ้มไม่ให้ใครแก้ไข ในพื้นที่ 2
SELE 1 ;   ? FIELD1             // 101 นำข้อมูลมาแสดงได้
//  DELETE                      แต่ไม่สามารถลบได้ เพราะแฟ้มถูก LOCK
? FLOCK()                       // .F.
```

: ตัวอย่างที่ 3.106

```
USE FILEA
USE FILEA EXCLUSIVE NEW // เปิดพร้อมระบุไม่ให้มีการแบ่งใช้ข้อมูล
SELE 1
? FIELD1                    // 101 นำข้อมูลมาแสดงได้
? FLOCK()                   // .T.
DELETE                      // ลบข้อมูลได้เพราะจองไว้แก้ไข
RECALL ALL
SELE 2
// FIELD1                   นำฟิลด์มาใช้ไม่ได้เพราะเปิดแบบ EXCLUSIVE
? FLOCK()                   // .F.
```

: ตัวอย่างที่ 3.107

```
USE FILEA EXCLUSIVE         // เปิดพร้อมระบุไม่ให้มีการแบ่งใช้ข้อมูล
USE FILEA SHARED NEW
SELE 1
 ? FIELD1                   // 101 นำข้อมูลมาแสดงได้
 ? FLOCK()                  // .T.
 DELETE                     // ลบข้อมูลได้เพราะจองไว้แก้ไขแล้ว
 RECALL ALL
SELE 2
 // FIELD1                  นำฟิลด์มาใช้ไม่ได้เพราะพื้นที่ 1 เปิดแบบ EXCLUSIVE
 ? FLOCK()                  // .F.
SELE 1
 CLOSE                      // ทำให้แฟ้มในพื้นที่ 2 ปิดเช่นกัน
SELE 2
 USE FILEA SHARED
 ? FLOCK()                  // .T.
 ? FIELD1                   // 101
 UNLOCK                     // UNLOCK ต้องใช้ในพื้นที่ที่ LOCK
SELE 1 ;   USE FILEA SHARED
 ? FLOCK()                  // .T.
SELE 2 ;    ? FLOCK()       // .F.
```

& 3.92 FOPEN()

! รูปแบบ

FOPEN(<ชื่อแฟ้ม>, [<แบบในการเปิดแฟ้ม>]) --> ตัวแปรระดับแฟ้ม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งเปิดแฟ้มแบบ BINARY FILE

เปิดแฟ้มได้ 3 ลักษณะดังนี้
0 เปิดแฟ้มแบบ อ่านได้อย่างเดียว
1 เปิดแฟ้มแบบ เขียนให้เขียนลงไปได้อย่างเดียว
2 เปิดแฟ้มแบบ อนุญาติให้ทั้งอ่านและเขียนลงไปในแฟ้ม

: ตัวอย่างที่ 3.108

```
F = FOPEN("TEST.TXT",2)
? FREADSTR(F,1000)          // นำ 1000 ไบต์แรกมาแสดง หรือเท่าที่มี
FSEEK(F,0,0)                // เลื่อนตัวชี้ไปที่ต้นแฟ้ม
? FWRITE(F,"TEST",4)        // 4 จำนวนไบต์ที่เขียนลงไปในแฟ้ม
```

& 3.93 FOUND()

! รูปแบบ

FOUND() --> ผลการค้นหา

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งผลของการ SEARCH เช่น LOCATE หรือ SEEK

: ตัวอย่างที่ 3.109

```
USE FILEA
LOCATE FOR FIELD2 > 900
DO WHILE FOUND()
  ? FIELD1,FIELD2,FIELD3
  SKIP
  LOCATE REST FOR FIELD2 > 900
ENDDO
```

: ตัวอย่างที่ 3.110

```
USE FILEA
CRITERIA = "FIELD2 > 900"
LOCATE FOR &CRITERIA
I = 1
```

```
DO WHILE FOUND()
  ? I,FIELD1,FIELD2,FIELD3
  SKIP
  LOCATE REST FOR &CRITERIA
  I++
ENDDO
```

: ตัวอย่างที่ 3.111

```
USE FILEA
LOCATE FOR FIELD1 = '102'
IF FOUND() ; ? FIELD1,FIELD2,FIELD3 ; ENDIF
GO TOP   ;      LOCATE FOR FIELD2 < 1000
IF FOUND() ; ? FIELD1,FIELD2,FIELD3 ; ENDIF
GO TOP   ;      LOCATE FOR FIELD1 = '102'
? IF(FOUND(),(FIELD1+STR(FIELD2)+STR(FIELD3)),'NOT FOUND')
? IIF(.NOT. FOUND(),,FIELD2+FIELD3)          // ถ้าไม่พบจะไม่แสดงอะไร
```

& 3.94 FREAD()

! รูปแบบ

FREAD(<ตัวแปรระดับแฟ้ม>, @<ตัวแปรรับอักขระ>, <จำนวนอักขระ>) --> จำนวนอักขระ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ อ่านแฟ้มแบบ BINARY FILE แล้วส่งค่าให้ตัวแปร

: ตัวอย่างที่ 3.112

```
X = 10                              // ตัวแปรบอกขนาดการอ่านแต่ละครั้ง
GETDATA = SPACE(X)                  // ตัวแปรที่รับค่าจากการอ่าน
F = FOPEN("C:\AUTOEXEC.BAT",0)      // เปิดแฟ้มแบบอ่านอย่างเดียว
DO WHILE .T.
  IF FREAD(F,@GETDATA,X) <> X
    ? "ERROR READING"               // ตัวอักษรที่เศษจาก 10
    EXIT                            // จะไม่ถูกพิมพ์ออกมา
  ELSE
    ?? GETDATA
  ENDIF
ENDDO
FCLOSE(F)
```

& 3.95 FREADSTR()

! รูปแบบ

FREADSTR(<ตัวแปรระดับแฟ้ม>, <จำนวนอักขระ>) --> ข้อความที่อ่านได้

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ อ่านแฟ้มแบบ BINARY FILE หรือ สั่งให้อ่านโดยไม่ส่งค่า

: ตัวอย่างที่ 3.113

```
F = FOPEN("C:\AUTOEXEC.BAT",0)    // เปิดแฟ้มแบบอ่านอย่างเดียว
X = " "                           // กำหนดค่าเริ่มต้นให้ X
DO WHILE LEN(X) <> 0
  X = FREADSTR(F,40)              // ขนาดของ X ขยายเอง
  ?? X
ENDDO
FCLOSE(F)
```

: ตัวอย่างที่ 3.114

```
USE FILEA
H  = HEADER()
RC = RECCOUNT()
RZ = RECSIZE()
CLOSE ALL
F = FOPEN("FILEA.DBF",0)          // เปิดแฟ้มแบบอ่านอย่างเดียว
FREADSTR(F,H)                     // อ่านหัวของแฟ้มข้อมูล
FOR I = 1 TO RC
   ? FREADSTR(F,RZ)               // นำข้อมูลในเรคอร์ดมาพิมพ์
NEXT
FCLOSE(F)
```

& 3.96 FRENAME()

! รูปแบบ

FRENAME(<ชื่อแฟ้มเดิม>, <ชื่อแฟ้มใหม่>) --> ผลการเปลี่ยนชื่อ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งเปลี่ยนชื่อแฟ้ม

การเปลี่ยนชื่อที่ถูกต้องจะให้ผลเป็น 0 แต่ถ้าผิดพลาดจะให้ผลเป็น -1

จากกรณีต่าง ๆ เช่น แผ่นเต็ม ชื่อแฟ้มใหม่มีอยู่แล้ว หรือไม่พบชื่อแฟ้มเก่าที่จะเปลี่ยน

: ตัวอย่างที่ 3.115

```
? FRENAME("X.BAK","X.BKK")          // ถ้าเปลี่ยนได้จะพิมพ์เลข 0
IF FRENAME("X.BAK","X.BKK") = -1    // ผลของบรรทัดนี้มักเป็น -1
  ? "ERROR OPERATION ON RENAME"     // เพราะบรรทัดแรกเปลี่ยนชื่อแล้ว
ELSE
  ? "O.K."                          // มีโอกาสเป็นไปได้น้อยที่สุด
ENDIF
FRENAME("X.BKK","X.BAK")            // ไม่แสดงผลใด ๆ แต่จะประมวลผลทันที
IF FRENAME("X.BAK","X.BKK") = 0     // ประมวลผลพร้อมส่งค่ามาตรวจสอบ
  ? "O.K."                          // มีโอกาสเป็นไปได้สูง ถ้ามีแฟ้มชื่อ
ENDIF                               // X.BAK อยู่ตั้งแต่แรก
```

& 3.97 FSEEK()

! รูปแบบ

FSEEK(<ตัวแปรระดับแฟ้ม>, <จำนวนตำแหน่ง>, [<แบบของการย้าย>])

--> ที่อยู่ของตำแหน่งปัจจุบัน

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ เซตตำแหน่งของตัวชี้ในแฟ้มแบบ BINARY FILE

แบบของการย้ายมี 3 แบบ
0 การย้ายจากต้นแฟ้มลงไป
1 การย้ายจากตำแหน่งปัจจุบันลงไป
2 การย้ายจากท้ายแฟ้มขึ้นมา

: ตัวอย่างที่ 3.116

```
F = FOPEN("TEST.TXT")
? FSEEK(F,0,0)          //  0 ไปที่ต้นแฟ้ม
? FSEEK(F,0,1)          //  0 จากตำแหน่งปัจจุบันไปอีก 0 ไบต์
? FSEEK(F,0,2)          //  15 ไปที่ท้ายแฟ้ม
? FSEEK(F,0,1)          //  15 จากตำแหน่งปัจจุบันไปอีก 0 ไบต์
? FSEEK(F,0,0)          //  0 เป็นการเซตค่าของตัวชี้ใหม่
? FSEEK(F,2,0)          //  2 จากต้นแฟ้มไปอีก 2 ไบต์
? FSEEK(F,5,1)          //  7 จากไบต์ที่ 2 ไปอีก 5 ไบต์
```

& 3.98 FWRITE()

! รูปแบบ

FWRITE(<ตัวแปรระดับแฟ้ม>, <ตัวอักษรที่ใส่ลงแฟ้ม>, [<จำนวนตัวอักษร>])

--> จำนวนตัวอักษรที่เขียนลงแฟ้ม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งเขียนลงไปในแฟ้มแบบ BINARY FILE

: ตัวอย่างที่ 3.117

```
F = FCREATE("TEST.TXT",0)          // สร้างแฟ้มแบบอ่านเขียนได้
FWRITE(F,'A')                      // แฟ้มจะมีขนาด 1 ไบต์
FCLOSE(F)
```

: ตัวอย่างที่ 3.118

```
F = FCREATE("TEST.TXT",0)          // สร้างแฟ้มแบบอ่านเขียนได้
FWRITE(F,'A',10)                   // เพิ่มข้อมูลในแฟ้มอีก 10 ไบต์
FWRITE(F,'A',10)                   // เพิ่มข้อมูลในแฟ้มอีก 10 ไบต์
FCLOSE(F)                          // แฟ้มมีขนาด 20 ไบต์
// แต่ส่วนที่ไม่ได้ระบุจะกำหนดไม่ได้ในแต่ละ 10 ไบต์นั้น
```

: ตัวอย่างที่ 3.119

```
CLS
F = FCREATE("TEST.TXT",0)          // สร้างแฟ้มแบบอ่านเขียนได้
? 'พิมพ์คำว่า END เมื่อต้องการเลิกจัดเก็บ'
DO WHILE .T.
   ACCEPT TO X                     // ข้อมูลที่ได้จะไม่มีการตัดบรรทัด
   IF X = 'END' ; EXIT ; ENDIF     // เพราะไม่เก็บรหัสตัดบรรทัดให้
   FWRITE(F,X)                     // LASTKEY ใช้กับ ACCEPT ไม่ได้
ENDDO                              // เพราะไม่สามารถกด ESC แล้วเลิกงาน
FCLOSE(F)
```

: ตัวอย่างที่ 3.120

```
#DEFINE CRLF()  (CHR(13)+CHR(10))
CLS
F = FCREATE("TEST.TXT",0)          // สร้างแฟ้มแบบอ่านเขียนได้
LINE = 2
@ 1,5 SAY 'เลิกรับข้อมูลให้กด ENTER มาบรรทัดที่ว่าง แล้วกด ESC'
DO WHILE .T.
```

```
    X = SPACE(50)
    @ LINE,5 GET X
    READ
    LINE++
    IF LASTKEY() = 27 ; EXIT ; ENDIF
    FWRITE(F,RTRIM(X)+CRLF())
  ENDDO
  FCLOSE(F)
```

& 3.99 GETENV()

! รูปแบบ

GETENV(<ชื่อตัวแปรสภาวะที่เคยเซตไว้>) --> ค่าที่เคยเซตไว้ให้ตัวแปรสภาวะ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ รับค่าที่เคยเซตไว้ในระบบปฏิบัติการ เพื่อควบคุมสภาพแวดล้อม

: ตัวอย่างที่ 3.121

```
? GETENV("PATH")          // C:\;C:\CLIPPER5;C:\WINDOWS
? GETENV("OBJ")           // C:\CLIPPER5\OBJ
? GETENV("LIB")           // C:\CLIPPER5\LIB
DIR
```

& 3.100 HARDCR()

! รูปแบบ

HARDCR(<ชุดตัวอักขระที่มี SOFT CR.>) --> ชุดตัวอักขระที่เป็น HARD CR.

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แทนที่ค่า SOFT CARRIAGE RETURN ด้วย HARD CARRIAGE RETURN นั่นคือแทนที่รหัสแอสกี 141 ด้วย 13

: ตัวอย่างที่ 3.122

```
TXT = MEMOREAD("C:\LETTER.TXT")
TXT = HARDCR(TXT)
SET CURSOR OFF
MEMOEDIT(TXT,10,15,20,60,.F.)     // อ่านข้อความมาดู แต่ไม่แก้ไข
SET CURSOR ON
```

& 3.101 HEADER()

! รูปแบบ

HEADER() --> ขนาดของส่วนหัวในแฟ้มข้อมูล

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งค่าความยาวของหัวแฟ้ม สำหรับแฟ้มที่นามสกุล DBF

: ตัวอย่างที่ 3.123

```
USE FILEA
? HEADER()                          // 130
? RECCOUNT()                        //  4
? RECSIZE()                         //  31
#DEFINE DBFSIZE()  (HEADER() + 1 + (RECCOUNT()*RECSIZE()))
USE FILEA
? DBFSIZE()                         // 255
```

& 3.102 IF()

! รูปแบบ

[I]IF(<เงื่อนไข>, <นิพจน์เมื่อเป็นจริง>, <นิพจน์เมื่อเป็นเท็จ>) --> ผลลัพธ์

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งผลการตรวจสอบเงื่อนไข

: ตัวอย่างที่ 3.124

```
A = 5
?  IF(A = 5,"YONOK","LAMPANG")
? IIF(A = 5,"YONOK","LAMPANG")
@ ROW() , IF(A=5,10,20)  SAY "WOW1"
@ IF(ROW() = 20,1,ROW()) , COL()        SAY "WOW2"
@ IIF(ROW() = 20,1,ROW()), IIF(A=5,30,40) SAY "WOW3"
```

& 3.103 INDEXKEY()

! รูปแบบ

INDEXKEY(<ลำดับชื่อแฟ้มดรรชนี>) --> ชื่อแฟ้มดรรชนี

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ บอกชื่อแฟ้ม INDEX ซึ่งระบุลำดับของแฟ้มได้

: ตัวอย่างที่ 3.125

```
USE FILEA INDEX FILEI1,FILEI2,FILEI3
? INDEXKEY(0)                    // FILEI1
? INDEXKEY(1)                    // FILEI1
? INDEXKEY(2)                    // FILEI2
SET ORDER TO 2
? INDEXKEY(0)                    // FILEI2   0 หมายถึงแฟ้ม INDEX ปัจจุบัน
? INDEXKEY(INDEXORD())           // FILEI2
```

& 3.104 INDEXORD()

! รูปแบบ

INDEXORD() --> ลำดับชื่อแฟ้มดรรชนีที่กำลังใช้งานอยู่

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ บอกลำดับของแฟ้มแบบ INDEX ที่กำลังเปิดใช้งาน

: ตัวอย่างที่ 3.126

```
USE FILEA INDEX FILEI1,FILEI2,FILEI3,FILEI4
? INDEXORD()                     // 1
IND := INDEXORD()
SET ORDER TO 2
? INDEXORD()                     // 2
SET ORDER TO 4
IF INDEXORD() = 4
  SET ORDER TO IND
ELSE
  SET ORDER TO 3
ENDIF
```

& 3.105 INKEY()

! รูปแบบ

INKEY([<จำนวนเวลาเป็นวินาที>]) --> ค่าของแป้นพิมพ์ที่ถูกกด

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ รอค่าจาก KEYBOARD

: ตัวอย่างที่ 3.127

```
INKEY(0)                          // WAIT UNTIL KEYPRESSED
INKEY(1)                          // WAIT 1 SECOND
INKEY(2)                          // WAIT 2 SECONDS
? "PLEASE KEY TEXT UNTIL ESC"
CH = SPACE(100)
X  = INKEY(0)
? CHR(X)
DO WHILE .NOT. LASTKEY() = 27
  X = INKEY(0)
  CH = CH + CHR(X)
  ?? CHR(X)
ENDDO
? CH
```

& 3.106 INT()

! รูปแบบ

INT(<จำนวนเลข>) --> จำนวนเต็ม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แปลงจำนวนเลขเป็นจำนวนเต็ม

: ตัวอย่างที่ 3.128

```
? INT(100.25)                     //     100
? INT(25)                         //      25
? INT(-25)                        //     -25
? INT(25.999)                     //      25
? INT(22 / 7)                     //       3
```

& 3.107 ISALPHA()

! รูปแบบ

ISALPHA(<ข้อความ>) --> ผลการตรวจสอบ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบว่าอักษรตัวแรกเป็นตัวอักษรหรือไม่

: ตัวอย่างที่ 3.129

```
? ISALPHA("ABCD")          // .T.  IT WILL CHECK THE FIRST COL.
? ISALPHA("AABB")          // .T.
? ISALPHA("1ABB")          // .F.
? ISALPHA("1234")          // .F.
? ISALPHA("12.4")          // .F.
? ISALPHA("A.A.")          // .T.
```

& 3.108 ISCOLOR()

! รูปแบบ

ISCOLOR() | ISCOLOUR() --> ผลการตรวจจอสี

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบสีของจอภาพ

: ตัวอย่างที่ 3.130

```
IF ISCOLOR()
  SETCOLOR("GR+/N,BG+/B")
ELSE
  SETCOLOR("W/N,N/W")
ENDIF
```

& 3.109 ISDIGIT()

! รูปแบบ

ISDIGIT(<ข้อความ>) --> ผลการตรวจสอบ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบว่าอักษรตัวแรกเป็นตัวเลขหรือไม่

: ตัวอย่างที่ 3.131

```
? ISALPHA("ABCD")          // .F.  IT WILL CHECK THE FIRST COL.
? ISALPHA("AABB")          // .F.
? ISALPHA("1ABB")          // .T.
? ISALPHA("1234")          // .T.
? ISALPHA("12.4")          // .T.
? ISALPHA("A.A.")          // .F.
```

& 3.110 ISLOWER()

! รูปแบบ

ISLOWER(<ข้อความ>) --> ผลการตรวจสอบ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบว่าอักษรตัวแรกเป็นตัวเล็กหรือไม่

: ตัวอย่างที่ 3.132

```
? ISALPHA("abCD")          // .T.  IT WILL CHECK THE FIRST COL.
? ISALPHA("aaBB")          // .T.
? ISALPHA("1abb")          // .F.
? ISALPHA("1234")          // .F.
? ISALPHA("12.4")          // .F.
? ISALPHA("A.a.")          // .F.
```

& 3.111 ISPRINTER()

! รูปแบบ

ISPRINTER() --> ผลการตรวจสอบ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบความพร้อมของเครื่องพิมพ์ก่อนพิมพ์

: ตัวอย่างที่ 3.133

```
IF ISPRINTER()
  SET DEVICE TO PRINTER
  @ PROW(),PCOL() SAY "TEST PRINTER"
  SET DEVICE TO SCREEN
  ? "PRINTER OK.
ELSE
  ? "PRINTER IS NOT READY"
ENDIF
```

& 3.112 ISUPPER()

! รูปแบบ

ISUPPER(<ข้อความ>) --> ผลการตรวจสอบ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบว่าอักษรตัวแรกเป็นตัวใหญ่หรือไม่

: ตัวอย่างที่ 3.134

```
? ISALPHA("aBCD")          // .F.  IT WILL CHECK THE FIRST COL.
? ISALPHA("aaaa")          // .F.
? ISALPHA("1ABB")          // .F.
? ISALPHA("1234")          // .F.
? ISALPHA("12.4")          // .F.
? ISALPHA("A.a.")          // .T.
```

& 3.113 I2BIN()

! รูปแบบ

I2BIN(<จำนวนเลข>) --> เลขในรูปแบบ 16 บิต

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แปลงตัวเลขเป็นเลขในแบบ 16 บิต ซึ่งใช้เนื้อที่ 2 ไบต์ (8*2=16)

: ตัวอย่างที่ 3.135

```
? ASC(SUBSTR(I2BIN(1),1,1))        // 1
? ASC(SUBSTR(I2BIN(1),2,1))        // 0
? ASC(SUBSTR(I2BIN(2),1,1))        // 2
? ASC(SUBSTR(I2BIN(2),2,1))        // 0
? ASC(SUBSTR(I2BIN(20001),1,1))    // 33
? ASC(SUBSTR(I2BIN(20001),2,1))    // 78  โดย 20001=78*256+33
? ASC(SUBSTR(I2BIN(20002),1,1))    // 34
? ASC(SUBSTR(I2BIN(20002),2,1))    // 78  โดย 20002=78*256+34
```

& 3.114 LASTKEY()

! รูปแบบ

LASTKEY() --> ค่าของแป้นพิมพ์ที่ถูกกด

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบการกดแป้นพิมพ์ครั้งสุดท้าย

: ตัวอย่างที่ 3.136

```
SALARY = 0
@ 5,10 SAY "GET SALARY : " GET SALARY
READ
IF LASTKEY() = 27
```

```
    ? "NO SALARY"
  ELSE
    TAX = SALARY * 0.07
    ? TAX
  ENDIF
```

& 3.115 LASTREC()

! รูปแบบ

LASTREC() | RECCOUNT() --> จำนวนเรคอร์ดของแฟ้ม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ บอกจำนวนของเรคอร์ดในแฟ้ม ให้ผลเหมือน RECCOUNT()

: ตัวอย่างที่ 3.137

```
USE FILEB
USE FILEA
? LASTREC()                    //    4
? RECCOUNT()                   //    4
SET FILTER TO FIELD2 >= 800
? RECCOUNT()                   //    4
? LASTREC()                    //    4
COUNT TO CNT
? CNT                          //    3
? FILEB->(LASTREC())           //   20
```

& 3.116 LEFT()

! รูปแบบ

LEFT(<ข้อความ>, <จำนวนอักขระ>) --> อักขระที่ถูกแยกออกมา

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งค่าทางซ้ายของตัวอักษรเท่าที่ระบุ

: ตัวอย่างที่ 3.138

```
A = "YONOK COLLEGE"
? LEFT(A,5)                    // YONOK
B = "SUPERMAN"
? B +" OF "+ LEFT(A,5)         // SUPERMAN OF YONOK
```

& 3.117 LEN()

! รูปแบบ

LEN(<ข้อความ>) --> ความยาว

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งความยาวของตัวอักษรที่ต้องการ

: ตัวอย่างที่ 3.139

```
A = "YONOK COLLEGE"
? LEN(A)                    //   13
B = LEN("SUPERMAN")         //   8
? B +" OF "+ LEFT(A,B)      //   8 OF YONOK CO
```

& 3.118 LOG()

! รูปแบบ

LOG(<จำนวนเลข>) --> ผลการคำนวณ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ คำนวณค่า NATURAL LOGARITHM

: ตัวอย่างที่ 3.140

```
? LOG(10)                   //   2.3
? LOG(10 * 2)               //   3.0
? LOG(2.71)                 //   1.0
? LOG(EXP(10))              //   10.0
```

& 3.119 LOWER()

! รูปแบบ

LOWER(<ข้อความ>) --> ตัวอักษรที่เป็นตัวเล็กหมด

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แปลงอักษรตัวใด ๆ เป็นตัวใหญ่

: ตัวอย่างที่ 3.141

```
X = "superMAN"
? LOWER(X)                  // superman
? LOWER("aABBCc")           // aabbcc
? LOWER("a1a2Aa")           // a1a2aa
? LEN(LOWER("123ABAB")      // 7
```

& 3.120 LTRIM()

! รูปแบบ

LTRIM(<ข้อความ>) --> อักขระที่ถูกตัดช่องว่างทางซ้าย

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ เอาช่องว่างด้านซ้ายของตัวอักษรออกไป

: ตัวอย่างที่ 3.142

```
X = "  ABCDEF"
Y = "ABC"
? X                          // ABCDEF
? Y                          // ABC
Z = 12
? Z                          //      12
? LEN(STR(Z))                //      10
? LTRIM(STR(Z+2))            // 14
? LTRIM(X)                   // ABCDEF
```

& 3.121 LUPDATE()

! รูปแบบ

LUPDATE() --> วันที่ล่าสุดของแฟ้ม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ บอกวันที่ล่าสุดที่มีการปรับปรุงแฟ้ม

: ตัวอย่างที่ 3.143

```
? DATE()                     // 12/31/02   วันที่วันนี้
USE FILEA
? LUPDATE()                  // 12/15/02   วันที่ของแฟ้ม
DELETE
PACK
CLOSE ALL
USE FILEA
? LUPDATE()                  // 12/31/02   วันที่ของแฟ้มเปลี่ยนไป
```

& 3.122 L2BIN()

! รูปแบบ

L2BIN(<จำนวนเลข>) --> เลขในรูปแบบ 32 บิต

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แปลงตัวเลขเป็นเลขในแบบ 32 BIT ซึ่งใช้เนื้อที่ 4 ไบต์ (8*4=32)

: ตัวอย่างที่ 3.144

```
? ASC(SUBSTR(L2BIN(1),1,1))          // 1
? ASC(SUBSTR(L2BIN(1),2,1))          // 0
? ASC(SUBSTR(L2BIN(1),3,1))          // 0
? ASC(SUBSTR(L2BIN(1),4,1))          // 0
? ASC(SUBSTR(L2BIN(2),1,1))          // 2
? ASC(SUBSTR(L2BIN(2),2,1))          // 0
? ASC(SUBSTR(L2BIN(2),3,1))          // 0
? ASC(SUBSTR(L2BIN(2),4,1))          // 0
? ASC(SUBSTR(L2BIN(10001),1,1))      // 17  อีก 3 ไบต์คือ 39 0 0
? ASC(SUBSTR(L2BIN(10001),2,1))      // 39
? ASC(SUBSTR(L2BIN(10002),1,1))      // 18  อีก 3 ไบต์คือ 39 0 0
? ASC(SUBSTR(L2BIN(10002),2,1))      // 39  โดย 10002=39*256+18
```

& 3.123 MAX()

! รูปแบบ

MAX(<นิพจน์ที่ 1>, <นิพจน์ที่ 2>) --> นิพจน์ที่มีค่ามากที่สุด

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งค่าที่มากที่สุดของการเปรียบเทียบเลข 2 จำนวน

: ตัวอย่างที่ 3.145

```
A = 12
B = 15 ;    C = 0
? MAX(A,B)                      //       15
? DATE()                        // 12/31/96
? MAX(DATE(),DATE()+2)          // 01/02/97
? MAX(3,12)                     //       12
```

& 3.124 MAXCOL()

! รูปแบบ

MAXCOL() --> ตัวเลข

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งค่าหลักมากสุดที่เป็นไปได้ของจอ

: ตัวอย่างที่ 3.146

```
X = MAXCOL()
? X                              // 80
@ 0,0 TO 10,MAXCOL()             // TO WRITE BOX
```

& 3.125 MAXROW()

! รูปแบบ

MAXROW() --> ตัวเลข

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งค่าแถวมากที่สุดที่เป็นไปได้ของจอ

: ตัวอย่างที่ 3.147

```
X = MAXROW()
Y = MAXCOL()
? X,Y                            //   25    80
DBEDIT(1,1,X+1,Y-1)
```

& 3.126 MEMOEDIT()

! รูปแบบ

MEMOEDIT([<ข้อความที่ต้องการแก้ไข>],
[<มุมบน>], [<มุมซ้าย>], [<มุมล่าง>], [<มุมขวา>])
--> ข้อความที่ผ่านการแก้ไข

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แสดงหรือแก้ไขฟิลด์แบบ MEMO

ปุ่มที่ใช้ใน MEMOEDIT()

UPARROW	เลื่อนเคอร์เซอร์ขึ้น	CTRL-PGUP	ไปที่หน้าแรกของบันทึก หรือแฟ้มข้อความนี้
DNARROW	เลื่อนเคอร์เซอร์ลง	CTRL-PGDN	ไปที่หน้าสุดท้ายของบันทึก หรือแฟ้มข้อความนี้
LEFTARROW	เลื่อนเคอร์เซอร์ไปทางซ้าย	RETURN	ไปต้นบรรทัดของบรรทัดต่อไป
RIGHTARROW	เลื่อนเคอร์เซอร์ไปทางขวา	DELETE	ลบตัวอักษรที่ตำแหน่งเคอร์เซอร์
CTRL-LEFT	เลื่อนเคอร์เซอร์ไปทางซ้าย 1 คำ	BACKSPACE	ลบตัวอักษรหน้าตำแหน่งเคอร์เซอร์
CTRL-RIGHT	เลื่อนเคอร์เซอร์ไปทางขวา 1 คำ	CTRL-Y	ลบบรรทัดปัจจุบันที่เคอร์เซอร์อยู่
HOME	ไปที่ต้นบรรทัด	CTRL-T	ลบข้อความที่ต่อท้ายเคอร์เซอร์ไปจนหมดบรรทัด
END	ไปที่ท้ายบรรทัด	CTRL-B	จัดรูปแบบของย่อหน้าใหม่
CTRL-HOME	ไปที่บรรทัดแรกของจอภาพ	CTRL-V/INS	สั่งให้ปุ่ม INSERT อยู่ในสถานะเป็นจริง
CTRL-END	ไปที่บรรทัดสุดท้ายของจอภาพ	CTRL-W	ยอมรับการแก้ไข แล้วออกจากฟังก์ชันนี้
PGUP	ไปที่หน้าก่อนหน้านี้	ESC	ยกเลิกการแก้ไข แล้วออกจากฟังก์ชันนี้
PGDN	ไปที่หน้าต่อไป		

: ตัวอย่างที่ 3.148

```
TXT = "ABCDE"+CHR(13)+CHR(10)+"FGHIJ"+CHR(13)+CHR(10)+ ;
    "KLMNO"+CHR(13)+CHR(10)+"PQRST"
SET CURSOR OFF
MEMOEDIT(TXT,10,15,20,60,.F.)        // อ่านข้อความมาดู แต่ไม่แก้ไข
SET CURSOR ON
```

: ตัวอย่างที่ 3.149

```
CLS
TXT = "ABCDE"+CHR(13)+CHR(10)+"FGHIJ"+CHR(13)+CHR(10)+ ;
    "KLMNO"+CHR(13)+CHR(10)+"PQRST"
SETCOLOR("W+/B")
TXT = MEMOEDIT(TXT,10,15,20,60)    // อ่านข้อความมาปรับปรุง
CLS
FOR I = 1 TO MLCOUNT(TXT)
   ?? MEMOLINE(TXT,80,I)
NEXT
```

: ตัวอย่างที่ 3.150

```
CLS
TXT = MEMOREAD("C:\AUTOEXEC.BAT")
SETCOLOR("W+/B")
TXT = MEMOEDIT(TXT,10,15,20,60)    // อ่านแฟ้มข้อความมาปรับปรุง
CLS
FOR I = 1 TO MLCOUNT(TXT)          // ปรับปรุงลง TXT
   ?? MEMOLINE(TXT,80,I)           // ไม่เขียนลง C:\AUTOEXEC.BAT
NEXT
```

: ตัวอย่างที่ 3.151

```
CLS
TXT = MEMOREAD("C:\TEST.TXT")
SETCOLOR("W+/B")
TXT = MEMOEDIT(TXT,10,15,20,60)    // อ่านแฟ้มข้อความมาปรับปรุง
MEMOWRIT("C:\TEST.TXT",TXT)        // ถ้าไม่เคยมีแฟ้มนี้มาก่อน
CLS                                          // คำสั่งนี้จะสร้างแฟ้มใหม่ให้
FOR I = 1 TO MLCOUNT(TXT)          // ปรับปรุงลง TXT และ
   ?? MEMOLINE(TXT,80,I)           // เขียนลง C:\TEST.TXT
```

```
NEXT
// แฟ้มที่สร้างด้วย MEMOEDIT จะมีรหัส 1A ปิดท้ายแฟ้มเสมอ
```

& 3.127 MEMOLINE()

! รูปแบบ

MEMOLINE(<ข้อความ>,
[<ขนาดของบรรทัดที่ต้องการแสดง>],
[<บรรทัดที่ต้องการแสดง>]) --> ข้อความที่ถูกเลือก

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ นำค่าในฟิลด์แบบ MEMO มาใช้โดยเลือกบรรทัดที่ต้องการ

: ตัวอย่างที่ 3.152

```
TXT = "ABCDE"+CHR(13)+CHR(10)+"FGHIJ"+CHR(13)+CHR(10)+ ;
    "KLMNO"+CHR(13)+CHR(10)+"PQRST"
?? MEMOLINE(TXT,80,1)          // ABCDE
?? MEMOLINE(TXT,80,2)          // FGHIJ
?? MEMOLINE(TXT,80,3)          // KLMNO
?? MEMOLINE(TXT,80,4)          // PQRST
INKEY(0)                       // แต่ละบรรทัดมีรหัสตัดบรรทัดของตนเอง จึงไม่ต้องใช้ ?
```

& 3.128 MEMOREAD()

! รูปแบบ

MEMOREAD(<ชื่อแฟ้ม>) --> ข้อความที่รับค่าจากแฟ้ม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ การอ่านค่าแฟ้มข้อความทั้งแฟ้ม มาเก็บในตัวแปรแบบ MEMO

: ตัวอย่างที่ 3.153

```
TXT = MEMOREAD("TEST.TXT")
FOR I = 1 TO MLCOUNT(TXT)
   ? MEMOLINE(TXT,80,I)        // อ่านข้อความจากแฟ้มมาแสดงผล
NEXT
```

& 3.129 MEMORY()

! รูปแบบ

MEMORY(<ระบุพื้นที่>) --> จำนวนกิโลไบต์ที่เหลือใช้

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แสดงหน่วยความจำที่เป็นไปได้

: ตัวอย่างที่ 3.154

```
? MEMORY(2)                          // บอกหน่วยความจำสูงสุดที่ใช้คำสั่ง RUN ได้
? MEMORY(1)                          // พื้นที่เป็น KB ที่เหลือสำหรับโปรแกรม (64KB)
? MEMORY(0)                          // พื้นที่สูงสุด ที่เหลือทั้งหมด
IF MEMORY(2) >= 128
  RUN C:\WORD\CW\CW.EXE
ELSE
  ? "NOT ENOUGH MEMORY TO RUN PROGRAM"
ENDIF
```

& 3.130 MEMOTRAN()

! รูปแบบ

MEMOTRAN(<ข้อความ>,
[<ตัวอักษรที่ใช้แทนที่ HARD CR>],
[<ตัวอักษรที่ใช้แทนที่ SOFT CR>]) --> ข้อความถูกปรับปรุง

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แทนค่า CARRIAGE RETURN/LINE FEEDS ด้วยตัวอักษรตามต้องการ

: ตัวอย่างที่ 3.155

```
TXT = MEMOREAD("TEST.TXT")   ;      TXT = MEMOTRAN(TXT,"!","&")
FOR I = 1 TO MLCOUNT(TXT)
   ? MEMOLINE(TXT,80,I)              // แต่แฟ้ม TEST.TXT จะไม่ถูกปรับปรุง
NEXT
// HARD CARRIAGE RETURN คือ ผลของการกด ENTER ในโปรแกรมทั่วไป
// SOFT CARRIAGE RETURN คือ ผลของโปรแกรมที่ปัดบรรทัดให้กับแฟ้มข้อความ
// รหัส ASCII ของ HARD CARIAGE RETURN คือ 13
// รหัส ASCII ของ SOFT CARIAGE RETURN คือ 141
```

& 3.131 MEMOWRIT()

! รูปแบบ

MEMOWRIT(<ชื่อแฟ้ม>, <ข้อความที่เขียนลงแฟ้ม>) --> ผลการเขียนลงแฟ้ม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ เขียนแฟ้มข้อความ หรือ MEMO FIELD ลงไปในแฟ้ม

: ตัวอย่างที่ 3.156

```
TXT = MEMOREAD("C:\AUTOEXEC.BAT")
MEMOWRIT("TEST.TXT",TXT)          // นำข้อความในแฟ้ม เขียนลงแฟ้ม
CNTLINE = MLCOUNT(MEMOREAD("TEST.TXT"))  ;  ?
FOR I = 1 TO CNTLINE
   ?? MEMOLINE(MEMOREAD("TEST.TXT"),80,I)
NEXT
```

: ตัวอย่างที่ 3.157

```
USE FILEHIST
DO WHILE .NOT. EOF()
  IF MEMOWRIT("MEM.TXT",NOTES)
    ? "VALID TO FILE"
    RUN TYPE MEM.TXT
    INKEY(0)
  ELSE
    ? "INVALID TO FILE"
  ENDIF
  SKIP
ENDDO
```

& 3.132 MIN()

! รูปแบบ

MIN(<นิพจน์ที่ 1>, <นิพจน์ที่ 2>) --> นิพจน์ที่มีค่าน้อยที่สุด

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งค่าที่น้อยที่สุดของการเปรียบเทียบเลข 2 จำนวน

: ตัวอย่างที่ 3.158

```
A = 12
B = 15
C = 0
? MIN(A,B)                    //      12
? DATE()                      // 12/31/96
? MIN(DATE(),DATE()+2)        // 12/31/96
? MIN(3,12)                   //       3
```

& 3.133 MLCOUNT()

! รูปแบบ

MLCOUNT(<ข้อความ>) --> จำนวนบรรทัดของแฟ้มข้อความ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ นับจำนวนบรรทัดในแฟ้มข้อความ หรือ MEMO FIELD

: ตัวอย่างที่ 3.159

```
TXT = MEMOREAD("C:\AUTOEXEC.BAT")
CNTLINE = MLCOUNT(TXT)
? "THIS FILE HAVE "+ STR(CNTLINE)
? "----------"
?                                   // ต้องตัดบรรทัดให้
FOR I = 1 TO CNTLINE                // จึงจะพิมพ์ที่หลักแรกได้ถูกต้อง
   ?? MEMOLINE(TXT,80,I)
NEXT
```

& 3.134 MLCTOPOS()

! รูปแบบ

MLCTOPOS(<ข้อความ>, <ความกว้างของบรรทัด>, <จำนวนบรรทัด>, <ตำแหน่งหลัก>)
--> ตำแหน่งล่าสุด

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งตำแหน่งของข้อมูลเป็นตำแหน่งที่ โดยบอกเป็นแถวและหลัก
โดยการนับแถวจะเริ่มจาก 1 แต่การนับหลักจะเริ่มจาก 0 ของแต่ละแถว
ตรงข้ามกับฟังก์ชัน MPOSTOLC()

: ตัวอย่างที่ 3.160

```
TXT = 'ASFASFASFDASFASFASFASFASDFASFASASDFSAF'
? MLCTOPOS(TXT,5,2,0)               // 6
? MLCTOPOS(TXT,6,3,0)               // 13
? MLCTOPOS(TXT,7,4,2)               // 24
```

& 3.135 MLPOS()

! รูปแบบ

MLPOS(<ข้อความ>, <ความกว้างของบรรทัด>, <จำนวนบรรทัด>) --> ตำแหน่งล่าสุด

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ระบุตำแหน่งแรกของบรรทัดที่ต้องการในแฟ้มข้อความ หรือ MEMO FIELD

: ตัวอย่างที่ 3.161

```
TXT = MEMOREAD("C:\AUTOEXEC.BAT")
CHRINLINE = 40                  // ยิ่งมากยิ่งดี แต่ถ้าน้อยจะมีผลต่อการนับบรรทัด
CNTLINE = 3                     // ต้องการทราบว่าตำแหน่งแรกของบรรทัดที่ 3
X = MLPOS(TXT,CHRINLINE,CNTLINE)
? X                             // 17 หมายถึงตัวเลขแสดงว่าบรรทัดที่ 3 เริ่มไบต์ที่ 17
? SUBSTR(TXT,X,20)              // นำอักษร 20 ตัวจากบรรทัดที่ 3 มาแสดง
```

& 3.136 MOD()

! รูปแบบ

MOD(<ตัวตั้ง>, <ตัวหาร>) --> เศษจากการหาร

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งเศษของการหาร ของตัวแปร 2 จำนวน

: ตัวอย่างที่ 3.162

```
? MOD(3,1)          //   0.00
? MOD(5,2)          //   1.00
? MOD(101,20)       //   1.00
```

& 3.137 MONTH()

! รูปแบบ

MONTH(<วันที่>) --> เลขเดือน

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แปลจาก DATE ให้เป็นเลขเดือน

: ตัวอย่างที่ 3.163

```
? DATE()                    // 12/31/96
? MONTH(DATE())             // 12
? MONTH(CTOD("05/113/95"))  // 05
```

& 3.138 MPOSTOLC()

! รูปแบบ

MPOSTOLC(<ข้อความ>, <ความกว้าง>, <ตำแหน่งหลัก>) --> ชุดอาเรย์เก็บบรรทัดและหลัก

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งคืนค่าตำแหน่งของข้อมูลเป็นแถวและหลัก เก็บในอาเรย์ โดยบอกตำแหน่งที่ตรงข้ามกับฟังก์ชัน MLCTOPOS()

: ตัวอย่างที่ 3.164

```
TXT ='OAKALKLJKHLKJHLKJDFJAAFJKFJASFASF'
X = MPOSTOLC(TXT,5,6)
? X[1],X[2]                                          //  2  0
? MPOSTOLC(TXT,5,0)[1],MPOSTOLC(TXT,5,0)[2]    //  0  0
? MPOSTOLC(TXT,5,1)[1],MPOSTOLC(TXT,5,1)[2]    //  1  0
? MPOSTOLC(TXT,5,2)[1],MPOSTOLC(TXT,5,2)[2]    //  1  1
? MPOSTOLC(TXT,5,3)[1],MPOSTOLC(TXT,5,3)[2]    //  1  2
? MPOSTOLC(TXT,5,4)[1],MPOSTOLC(TXT,5,4)[2]    //  1  3
? MPOSTOLC(TXT,5,5)[1],MPOSTOLC(TXT,5,5)[2]    //  1  4
? MPOSTOLC(TXT,5,7)[1],MPOSTOLC(TXT,5,7)[2]    //  2  1
? MPOSTOLC(TXT,5,8)[1],MPOSTOLC(TXT,5,8)[2]    //  2  2
? MPOSTOLC(TXT,5,9)[1],MPOSTOLC(TXT,5,9)[2]    //  2  3
// ถ้าระบุความกว้างเป็น 5 แล้วต้องการหาว่าตำแหน่งที่ 9 อยู่ที่ใด
// จะได้คำตอบเป็นแถวที่ 2 หลักที่ 3 เพราะการจัดเรียงข้อมูลเป็นตาราง
// ตำแหน่งแรกอยู่แถวที่ 1 หลักที่ 0 หรือตำแหน่งที่หกอยู่แถวที่ 2 หลักที่ 0
```

& 3.139 NEXTKEY()

! รูปแบบ

NEXTKEY() --> ค่าของแป้นพิมพ์ที่ถูกกด

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบการกดแป้นพิมพ์เหมือน INKEY() แต่ INKEY() จะลบค่าในหน่วยความจำ NEXTKEY() จะอ่านค่ามาแต่ไม่ลบค่าออกจากหน่วยความจำ

ระหว่าง INKEY() LASTKEY() และ NEXTKEY() มีการทำงานร่วมกันได้ดีมาก

INKEY() เมื่อได้ค่าจากหน่วยความจำว่ากดอะไร จะส่งค่าไปยัง LASTKEY()

LASTKEY() จะตรวจสอบว่าค่าสุดท้ายในหน่วยความจำ มักรับค่ามาจาก INKEY()

NEXTKEY() อ่านค่าจากหน่วยความจำ แต่จะไม่เปลี่ยนแปลงค่าในหน่วยความจำ

: ตัวอย่างที่ 3.165

```
CLS
KEYBOARD CHR(65)
? NEXTKEY(),NEXTKEY()                //    65     65
```

```
? LASTKEY(),INKEY()              //     0     65
? INKEY(),LASTKEY()              //     0     65
? LASTKEY(),NEXTKEY()            //    65      0
ACCEPT TO X                      // ABCD แล้วกด ENTER
? X                              // ABCD
? NEXTKEY(),INKEY()              //     0      0
? LASTKEY(),NEXTKEY()            //    13      0
WAIT                             // PRESS ANY KEY TO CONTINUE
WAIT "=> Y/N"                    // Y/N   ถ้ากด A จะได้ LASTKEY() = 97
? NEXTKEY(),INKEY()              //     0      0
? LASTKEY(),NEXTKEY()            //    97      0
INKEY(0)                         // รอรับค่าหากกด A จะได้ LASTKEY() = 65
? NEXTKEY(),INKEY()              //     0      0
? LASTKEY(),NEXTKEY()            //    65      0
@ 15,10 GET X                    // ถ้ารับค่าเป็น D จะทำให้ LASTKEY() = 100
READ
? NEXTKEY(),INKEY()              //     0      0
? LASTKEY(),NEXTKEY()            //   100      0
```

: ตัวอย่างที่ 3.166

```
CLS                              // ตัวอย่างที่นี้แสดงให้เป็นการใช้ KEYBOARD
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
```

& 3.140 OS()

! รูปแบบ

OS() --> ชื่อระบบปฏิบัติการ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ บอกชื่อระบบปฏิบัติการที่กำลังใช้อยู่ในระบบคอมพิวเตอร์ของเรา

: ตัวอย่างที่ 3.167

```
CLS
? OS()                                  // DOS 6.20
FOR I = 1 TO 10 ;   ? OS() ;      NEXT
```

& 3.141 OUTERR()

! รูปแบบ

OUTERR(<รายการนิพจน์>) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แสดงข้อความออกทางอุปกรณ์ที่เตรียมไว้แสดงข้อผิดพลาด

: ตัวอย่างที่ 3.168

```
USE FILEA                               // ถ้าเปิดแฟ้มครั้งแรกจะได้ค่า RLOCK() = .T.
IF RLOCK() = .F.                        // รายงานทางอุปกรณ์ที่แสดงข้อผิดพลาด
  OUTERR("INVALID SHARED",DATE(),TIME())
ENDIF
USE FILEA NEW
IF RLOCK() = .F.                        // รายงานว่าแฟ้มถูกจองไว้ ไม่สามารถนำมาใช้ได้
  OUTERR("INVALID SHARED",DATE(),TIME()) // แบบนี้แสดงทางจอภาพ
ENDIF
```

& 3.142 OUTSTD()

! รูปแบบ

OUTSTD(<รายการนิพจน์>) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แสดงข้อมูลออกทางอุปกรณ์มาตรฐาน

ถ้าโปรแกรมนี้ชื่อ X.EXE เมื่อประมวลผลที่ดอสว่า C:\X >AA

โดยแฟ้ม AA คือผลลัพธ์ที่ได้จากการสั่ง OUTSTD ในโปรแกรมนี้ แต่ ? จะไม่มีผลต่อแฟ้ม AA

: ตัวอย่างที่ 3.169

```
USE FILEA
DO WHILE .NOT. EOF()
   OUTSTD(FIELD1,FIELD2,FIELD3)   // ไม่เลื่อนบรรทัดให้เหมือน QOUT
   ?                              // ทำหน้าที่เลื่อนบรรทัด เท่านั้น
   SKIP
ENDDO
```

& 3.143 PAD()

! รูปแบบ

PADL(<นิพจน์ที่ต้องการแสดง>, <ความยาว>) --> ข้อความที่ชิดซ้าย

PADC(<นิพจน์ที่ต้องการแสดง>, <ความยาว>) --> ข้อความที่อยู่กึ่งกลาง

PADR(<นิพจน์ที่ต้องการแสดง>, <ความยาว>) --> ข้อความที่ชิดขวา

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ จัดชุดตัวอักษรให้เป็นระเบียบ เช่น ชิดซ้าย ชิดขวา หรือกึ่งกลาง

: ตัวอย่างที่ 3.170

```
CLS
SETCOLOR("W+/B")
X = PADC("DATE : "+DTOC(DATE()), 80)
@ 21, 0 SAY X
@ 22, 0 SAY PADR("TIME OF DAY : " + TIME(),80)
@ 23, 0 SAY PADL("SECOND : "+ STR(SECOND()),80)
INKEY(0)
```

& 3.144 PCOL()

! รูปแบบ

PCOL() --> จำนวนสูงสุดของหลัก

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งค่าหลักปัจจุบันที่อยู่บนเครื่องพิมพ์ คล้ายกับ COL() แต่ใช้กับเครื่องพิมพ์

: ตัวอย่างที่ 3.171

```
SET DEVICE TO PRINTER
FOR I = 1 TO 5                    // ! *
  @ PROW()+1,1 SAY "!"            // ! * *
  FOR J = 1 TO I                  // ! * * *
    @ PROW() , PCOL()+1 SAY "*"   // ! * * * *
  NEXT                            // ! * * * * *
NEXT                              // * จะห่างกันเพราะ PCOL()+1
EJECT                             // สั่งเลื่อนกระดาษออกจากเครื่องพิมพ์
SET DEVICE TO SCREEN
```

& 3.145 PCOUNT()

! รูปแบบ

PCOUNT() --> เลขแสดงการตรวจสอบการรับค่าของโปรแกรมย่อย

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบการรับตัวแปรของโปรแกรมย่อย ถ้าได้ค่า 0 หมายถึงไม่มีการรับตัวแปรเข้ามา

: ตัวอย่างที่ 3.172

```
DO XX WITH (5)                  // 1
DO XX                           // 0
DO XX WITH ("")                 // 1
DO XXX WITH (5,5,5)             // 1
PROCEDURE XX (NUM)
  ? PCOUNT()
RETURN
PROCEDURE XXX (NUM1,NUM2,NUM3)
  ? PCOUNT()
RETURN
```

& 3.146 PROCLINE()

! รูปแบบ

PROCLINE([<ลำดับของโปรแกรมที่พิจารณา>]) --> เลขที่บรรทัดในโปรแกรม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ บอกเลขบรรทัดของโปรแกรมที่ใช้งานอยู่จนถึงคำสั่งปัจจุบัน

: ตัวอย่างที่ 3.173

```
DO P1
PROCEDURE P1
  DO P2
RETURN
PROCEDURE P2 ;  DO P3 ;   RETURN
PROCEDURE P3
  ? PROCNAME(0),PROCLINE(0)      // P3   9
  ? PROCNAME(1),PROCLINE(1)      // P2   6
  ? PROCNAME(2),PROCLINE(2)      // P1   3
  ? PROCNAME(3),PROCLINE(3)      // X    1
RETURN
```

: ตัวอย่างที่ 3.174

```
DO P1
PROCEDURE P1
  DO P2
RETURN
PROCEDURE P2
  DO P3
RETURN
PROCEDURE P3                    // ตัวอย่างที่นี้ทำให้รู้ว่าเรียกโปรแกรมอะไรมาบ้าง
  I = 0
  DO WHILE LEN(PROCNAME(I)) > 0 // P3   11
    ? PROCNAME(I),PROCLINE(I)   // P2    6
    I++                         // P1    3
  ENDDO                         // X     1
RETURN
```

& 3.147 PROCNAME()

! รูปแบบ

PROCNAME([<ลำดับของโปรแกรมที่พิจารณา>]) --> ชื่อโปรแกรมย่อย

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ บอกชื่อโปรแกรมย่อยที่มีการเรียกใช้งานอยู่เป็นชั้น ๆ จนถึงชั้นนอกสุด

: ตัวอย่างที่ 3.175

```
DO P1                           // PROC1
DO P2                           // PROC2
DO P3                           // PROC3
? PROCNAME(0)                   // X  (CURRENT PROCEDURE)
PROCEDURE P1
  ? "PROC1"
RETURN
PROCEDURE P2
  ? "PROC2"
RETURN
PROCEDURE P3
  ? "PROC3"
```

```
  ? PROCNAME(0)                    // P3
  ? PROCNAME(1)                    // X
RETURN
```

: ตัวอย่างที่ 3.176

```
DO P1
PROCEDURE P1
  DO P2
RETURN
PROCEDURE P2 ;  DO P3  ;  RETURN
PROCEDURE P3
  ? PROCNAME(0)                    // P3
  ? PROCNAME(1)                    // P2
  ? PROCNAME(2)                    // P1
  ? PROCNAME(3)                    // X
RETURN
```

& 3.148 PROW()

! รูปแบบ

PROW() --> ตำแหน่งแถวบนเครื่องพิมพ์

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งค่าแถวปัจจุบันที่ระบุบนเครื่องพิมพ์ คล้ายกับ ROW() แต่ใช้กับเครื่องพิมพ์

: ตัวอย่างที่ 3.177

```
USE FILEA
SET DEVICE TO PRINTER
DO WHILE .NOT. EOF()
  @ PROW()+1,10 SAY FIELD1
  @ PROW()  ,20 SAY FIELD2
  @ PROW()  ,30 SAY FIELD3
  SKIP
ENDDO
EJECT
SET DEVICE TO SCREEN
```

& 3.149 QOUT()

! รูปแบบ

QOUT([<รายการนิพจน์>]) --> NIL

QQOUT([<รายการนิพจน์>]) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แสดงข้อมูลออกทางจอภาพ(CONSOLE) โดยมีการตัดบรรทัดให้อัตโนมัติทุกครั้ง

QOUT พิมพ์แต่ละเรคอร์ดจะตัดบรรทัดให้ คล้ายคำสั่ง ?

QQOUT พิมพ์แต่ละเรคอร์ดจะไม่ตัดบรรทัดให้ คล้ายคำสั่ง ??

: ตัวอย่างที่ 3.178

```
X = {1,2}
XXX = {|Y|QOUT(Y)}
AEVAL (X,{|Y|QOUT(Y)})                  // 1
X = {"A = A +1",120,.T.,"THAILAND"}  // 2
AEVAL (X,XXX)                           // การนำข้อมูลในอาเรย์มาแสดง
Z = ARRAY(4)
Z[1] = [RED]
Z[2] = [GREEN]
Z[3] = [YELLOW]
AEVAL (Z,XXX)                           // อาเรย์ตัวที่ 4 ไม่มีจะออกคำว่า NIL
```

: ตัวอย่างที่ 3.179

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

& 3.150 RAT()

! รูปแบบ

RAT(<ตัวอักษรที่ต้องการนำไปค้น>, <ข้อความเป้าหมาย>) --> ตำแหน่งที่ค้นพบ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งตำแหน่งสุดท้ายที่พบตัวอักษรในชุดตัวอักษรที่ต้องการหา

: ตัวอย่างที่ 3.180

```
X = "ABCDEFABCDEF"
```

```
? RAT("E",X)                    // 11
X = "C:\WINDOWS\WIN.COM"
POS = RAT("\",X)
? POS                           // 11
? LEN(X)                        // 18
? SUBSTR(X,POS+1,LEN(X)-POS)    // WIN.COM
? SUBSTR(X,1,POS)               // C:\WINDOWS\
```

& 3.151 READEXIT()

! รูปแบบ

READEXIT([<ตรรกนิพจน์>]) --> ตรรกนิพจน์แสดงค่าปัจจุบัน

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบการออกจากคำสั่ง READ ด้วยลูกศรขึ้นลง หรือการสั่งอนุญาติให้ออกหรือไม่ออก

: ตัวอย่างที่ 3.181

```
CLS
X = SPACE(5)
Y = SPACE(5)
Z = SPACE(5)
? READEXIT()                    // .F. ไม่ให้ออกจาก READ ด้วยลูกศร
OLDMODE = READEXIT()            // เก็บค่าเดิมไว้
READEXIT(.T.)                   // คือการอนุญาติให้ออกจาก READ ด้วยลูกศร
@ 4,10 SAY "TEST ARROW ON READ " GET X
@ 5,10 SAY "TEST ARROW ON READ " GET Y
@ 6,10 SAY "TEST ARROW ON READ " GET Z
READ                            // อนุญาติให้เลิกด้วยลูกศรได้
READEXIT(OLDMODE)               // เซตค่าใหม่ตามค่าที่เคยเก็บไว้
@ 7,10 SAY "TEST ARROW ON READ " GET X
@ 8,10 SAY "TEST ARROW ON READ " GET Y
@ 9,10 SAY "TEST ARROW ON READ " GET Z
READ                            // ไม่อนุญาติให้เลิก READ ด้วยลูกศร
```

& 3.152 READINSERT()

! รูปแบบ

READINSERT([<ตรรกนิพจน์>]) --> ตรรกนิพจน์แสดงค่าปัจจุบัน

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบการกดปุ่ม INSERT หรือ สั่งเปิดปุ่ม INSERT

: ตัวอย่างที่ 3.182

```
CLS
X = 0
? READINSERT()                  // .F. เสมอ เมื่อเริ่มโปรแกรม
OLDMODE = READINSERT()
READINSERT(.T.)
@ 5,10 SAY "TEST INSERT ON READ " GET X
READ
? READINSERT()                  // .T.
READINSERT(.F.)                 // สั่งปิด INSERT
@ 6,10 SAY "TEST INSERT ON READ " GET X
READ
READINSERT(OLDMODE)             // คืนสถานะการกด INSERT
@ 7,10 SAY "TEST INSERT ON READ " GET X
READ
```

& 3.153 READKEY()

! รูปแบบ

READKEY() --> ค่าของแป้นพิมพ์ที่ถูกกด

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบการออกจากคำสั่ง GET ... READ

ตารางค่าที่ส่งค่ามาให้ READKEY() ซึ่งเป็น EXIT CODE

UPARROW	5	ESC	12
DNARROW	24	CTRL-END,CTRL-W	14
PGUP	6	TYPE PAST END	15
PGDN	7	RETURN	15
CTRL-PGUP	31	CTRL-PGDN	30

: ตัวอย่างที่ 3.183

```
CLS
X = 0
@ 5,10 SAY "TEST READKEY() " GET X
READ
```

```
? READKEY()                    // บอกค่าที่เกิดจากการหยุดคำสั่ง READ
? LASTKEY()                    // ค่าของ LASTKEY คือการกดปุ่มครั้งสุดท้าย
```

& 3.154 READVAR()

! รูปแบบ

READVAR() --> ชื่อตัวแปรที่กำลังถูกเปิดอยู่

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ บอกชื่อตัวแปรปัจจุบันที่กำลังทำงานอยู่ สามารถนำไปสร้างส่วนช่วยเหลือได้อย่างดี

: ตัวอย่างที่ 3.184

```
CLS
SET KEY 28 TO HELPFUNCTION       // F1
_NAME = SPACE(20)
_AGE  = 0
_SALARY = 0
@ 5,10 SAY "NAME   : " GET _NAME
@ 6,10 SAY "AGE    : " GET _AGE
@ 7,10 SAY "SALARY : " GET _SALARY
READ
FUNCTION HELPFUNCTION
LOCAL SCR1 := SAVESCREEN(0,0,24,79)
DO CASE
  CASE READVAR() = "_NAME"
     ?? "IT SHOULD HAVE MR. OR MS. AT THE FONT OF NAME"
  CASE READVAR() = "_AGE"
     ?? "IT SHOULD BE 0 - 99"
  CASE READVAR() = "_SALARY"
     ?? "IT SHOULD > 1000 AND < 1,000,000"
ENDCASE
INKEY(0)
RESTSCREEN(0,0,24,79,SCR1)
RETURN
```

& 3.155 RECCOUNT()

! รูปแบบ

RECCOUNT() | LASTREC() --> จำนวนเรคอร์ดของแฟ้ม

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ บอกจำนวนของเรคอร์ดในแฟ้ม ให้ผลเหมือน LASTREC()

: ตัวอย่างที่ 3.185

```
USE FILEA
? RECCOUNT()                    //      4
? LASTREC()                     //      4
SET FILTER TO FIELD2 > 1000
COUNT TO FILTER_COUNT
? FILTER_COUNT                  //      1
? RECCOUNT()                    //      4
? LASTREC()                     //      4
```

& 3.156 RECNO()

! รูปแบบ

RECNO() --> เลขที่เรคอร์ดปัจจุบัน

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ บอกลำดับของเรคอร์ดปัจจุบันที่อยู่ในแฟ้ม การ INDEXED แฟ้มไม่มีผลกับลำดับ

: ตัวอย่างที่ 3.186

```
USE FILEA
? RECNO()                       //      1
GO BOTTOM
? RECNO()                       //      4
```

& 3.157 RECSIZE()

! รูปแบบ

RECSIZE() --> ขนาดของเรคอร์ด

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ บอกความยาวของ 1 เรคอร์ด

: ตัวอย่างที่ 3.187

```
USE FILEA
? RECSIZE()                    //    31
? LASTREC()                    //     4
? HEADER()                     //   130
FSIZE = RECSIZE()*LASTREC()+HEADER()+1
? "FILE SIZE = " + STR(FSIZE,4)     // FILE SIZE =  255
```

& 3.158 REPLICATE()

! รูปแบบ

REPLICATE(<ข้อความ>, <จำนวนทำซ้ำ>) --> ข้อความที่ผ่านการทำซ้ำ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ฟังก์ชันที่ใช้คัดลอกตัวอักษร ให้ได้หลาย ๆ ชุด

: ตัวอย่างที่ 3.188

```
? REPLICATE("A",3)             // AAA
? REPLICATE("AB ",2)           // AB AB
X = 5
? REPLICATE(STR(X,1),3)        // 555
FOR I = 1 TO 3                 // *
   ? REPLICATE("*",I)          // **
NEXT                           // ***
```

& 3.159 RESTSCREEN()

! รูปแบบ

RESTSCREEN(<มุมบน>, <มุมซ้าย>, <มุมล่าง>, <มุมขวา>,
<ตัวแปรเก็บจอภาพ>) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ คืนผลการ SAVESCREEN ให้จอภาพ

: ตัวอย่างที่ 3.189

```
SETCOLOR("GR+/B")              // ตัวอย่างที่นี้จะใช้พักจอภาพได้
CLS                            // ทุก 0.1วินาทีจะพิมพ์ข้อความ
SETPOS(5,10)                   // ที่ไม่ซ้ำกับตำแหน่งเดิม
?? "CLIPPER FOR DOS"
SCR1 = SAVESCREEN(5,10,5,24)
```

```
DO WHILE LASTKEY() != 27
  X = MOD(SECOND()*1000,24)
  Y = MOD(SECOND()*1000,65)
  RESTSCREEN(X,Y,X,Y+14,SCR1)
  INKEY(0.1)
ENDDO
```

& 3.160 RIGHT()

! รูปแบบ

RIGHT(<ข้อความ>, <จำนวนอักขระ>) --> อักษรที่ถูกแยกออกมา

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ เลือก SUBSTRING จากทางขวา

: ตัวอย่างที่ 3.190

```
? RIGHT("ABCD",2)              // CD
? RIGHT("1234",2)              // 34
X = "A1B2"
Y =[A2B3]
? RIGHT(X,2)+RIGHT(Y,2)        // B2B3
? RIGHT(X+Y,4)                 // A2B3
```

& 3.161 RLOCK()

! รูปแบบ

RLOCK() --> ผลการจองเรคอร์ด

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบการจองเรคอร์ดปัจจุบัน เพื่อทำการ ลบ เรียกคืน หรือเขียนทับลงไป

: ตัวอย่างที่ 3.191

```
USE FILEA
USE FILEA NEW
IF RLOCK()
  ? "LOCK OK."  ;     ELSE
  ? "CAN NOT LOCK."            // CAN NOT LOCK.
ENDIF
SELE 1  ;     CLOSE            // ปิดแฟ้มในพื้นที่ 1
SELE 2
```

```
IF RLOCK()                    // ตรวจสอบการจองเรคอร์ด
  ? "LOCK OK."
ELSE
    ? "CAN NOT LOCK."         // CAN NOT LOCK.
ENDIF
USE FILEA                     // เปิดแฟ้มนี้ใหม่ในพื้นที่ 2
IF RLOCK()                    // ตรวจสอบการจองเรคอร์ด
  ? "LOCK OK."                // LOCK OK. การจองเรคอร์ดถูกต้อง
  DELETE                      // สามารถ DELETE , RECALL หรือ REPLACE
ELSE
  ? "CAN NOT LOCK."
ENDIF
```

: ตัวอย่างที่ 3.192

```
USE FILEA                     // โปรแกรมนี้จะไม่นำเรคอร์ดมาแสดง
USE FILEA NEW                 // เพราะเรคอร์ดถูกจองไว้
? RLOCK()                     // .F.
DO WHILE .NOT. EOF()          // การวนลูปจึงนำเรคอร์ดออกมาไม่ได้
  ? FIELD1
  SKIP
ENDDO
```

: ตัวอย่างที่ 3.193

```
USE FILEA
USE FILEA NEW
SELE 1                        // เปิดแฟ้มแบบนี้จะนำเรคอร์ด
? RLOCK()                     // .T.
DO WHILE .NOT. EOF()          // มาแสดงได้เพราะการจองถูกใช้
  ? FIELD1                    // ในพื้นที่ ที่จองอย่างถูกต้อง
  SKIP
ENDDO
```

: ตัวอย่างที่ 3.194

```
USE FILEA SHARED              // เปิดแฟ้มแบบนี้ ทำให้พื้นที่อื่นใช้แฟ้มร่วมได้
USE FILEA SHARED NEW
? RLOCK()                     // .T.
DO WHILE .NOT. EOF()
```

```
    ? FIELD1
    SKIP
  ENDDO
```

& 3.162 ROUND()

! รูปแบบ

ROUND(<ตัวเลข>, <จำนวนทศนิยม>) --> ตัวเลขที่ผ่านการปัดเศษ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ปัดเศษให้กับตัวแปรตัวเลข

: ตัวอย่างที่ 3.195

```
? ROUND(101.1,0)          // 0 หมายถึงจำนวนทศนิยม จะได้ 101
? ROUND(106.1,-1)         // ปัดเศษตั้งแต่หลักสิบ จะได้ 110
? ROUND(106.1,-2)         // ปัดเศษตั้งแต่หลักร้อย จะได้ 100
? ROUND(106.1,-1)         // 110
? ROUND(106.1,-2)         // 100
? ROUND(106.164,2)        // 106.16
? ROUND(106.164,1)        // 106.2
? ROUND(106.164,3)        // 106.164
SET DECIMALS TO 3
SET FIXED ON
? ROUND(106.164,1)        // 106.200
? ROUND(106.164,-1)       // 110.000 (-1 หมายถึงไม่เอาหน้าจุด 1 หลัก)
? ROUND(106.164,2)        // 106.160 ( 2 หมายถึงเอาหลังจุด 2 หลัก)
```

& 3.163 ROW()

! รูปแบบ

ROW() --> เลขแถวปัจจุบัน บนจอภาพ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ คืนตำแหน่งของแถวที่แสดงบนจอภาพ

: ตัวอย่างที่ 3.196

```
USE FILEA
DO WHILE .NOT. EOF()
  @ ++ROW(),10 SAY FIELD1
  @   ROW(),20 SAY FIELD1
```

```
    @  ROW(),30 SAY FIELD1
    SKIP
  ENDDO
```

& 3.164 RTRIM()

! รูปแบบ

[R]TRIM(<ข้อความ>) --> ข้อความที่ผ่านการตัดช่องว่างทางขวา

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ นำช่องว่างทางขวาของตัวแปรตัวอักษรออกไป

: ตัวอย่างที่ 3.197

```
? RTRIM("ABCD")                    // ABCD
? RTRIM(" A  ")+RTRIM(" B.")       //  A B.
? RTRIM([ A  ])+RTRIM([ B ])+"C"   //  A BC
X = [ABC  ]
Y = [  ABC]
? X+Y                              // ABC    ABC
? RTRIM(X)+RTRIM(Y)                // ABC  ABC
? RTRIM(X)+Y                       // ABC  ABC
? X+RTRIM(Y)                       // ABC    ABC
```

& 3.165 SAVESCREEN()

! รูปแบบ

SAVESCREEN(<มุมบน>, <มุมซ้าย>, <มุมล่าง>, <มุมขวา>) --> ตัวแปรเก็บจอภาพ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ จัดเก็บจอภาพบางส่วนไว้ แล้วนำออกมาแสดงอีกได้ด้วยคำสั่ง RESTSCREEN

: ตัวอย่างที่ 3.198

```
CLS
SETPOS(5,10); ?? [==========================]
SETPOS(6,10); ?? [TEST OF SAVESCREEN FUNCTION]
SETPOS(7,10); ?? [==========================]
? "TEST ALL SCREEN"
? "--------------"
SAVE SCREEN TO SCR1
SCR2 = SAVESCREEN(5,10,7,36)
```

```
CLS
? "THIS IS THE RESULT OF SCR1"
RESTORE SCREEN FROM SCR1
INKEY(0)
CLS
? "THIS IS THE RESULT OF SCR2"
RESTSCREEN(5,10,7,36,SCR2)
RESTSCREEN(8,10,10,36,SCR2)
RESTSCREEN(15,5,17,31,SCR2)
RESTSCREEN(15,45,17,71,SCR2)
INKEY(0)
```

& 3.166 SCROLL()

! รูปแบบ

SCROLL([<มุมบน>], [<มุมซ้าย>], [<มุมล่าง>], [<มุมขวา>], [<จำนวนบรรทัด>])

--> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ สั่งเลื่อนขึ้นลงแต่ละบรรทัด ตามขอบเขตที่ระบุ

: ตัวอย่างที่ 3.199

```
CLS
SETPOS(1,5) ; ?? "ABCD"
SETPOS(2,5) ; ?? "EFGH"
SETPOS(3,5) ; ?? "IJKL"
SETPOS(4,5) ; ?? "MNOP"
SETPOS(5,5) ; ?? "QRST"
SETPOS(6,5) ; ?? "UVWX"
SETPOS(7,5) ; ?? "YZ.."          // แสดงข้อมูลมาให้ดู เห็น 7 บรรทัด
INKEY(0)                         // ขณะนี้บรรทัดแรกอยู่บรรทัดที่ 1
SCROLL(0,5,7,9,1)                // ขยับขึ้น และเห็นทั้ง 7 บรรทัด
INKEY(0)                         // ขณะนี้บรรทัดแรกอยู่บรรทัดที่ 0
SCROLL(0,5,7,9,-1)               // ขยับลง และยังเห็นทั้ง 7 บรรทัด
INKEY(0)                         // ขณะนี้บรรทัดแรกอยู่บรรทัดที่ 1
SCROLL(0,5,7,9,-1)               // ขยับลง ไม่เห็นบรรทัดสุดท้าย
INKEY(0)                         // ขณะนี้บรรทัดแรกอยู่บรรทัดที่ 2
```

```
SCROLL(0,5,7,9,-1)                    // ขยับลง เห็นเพียง 5 บรรทัดแรก
INKEY(0)                              // ขณะนี้บรรทัดแรกอยู่บรรทัดที่ 3
CLS
SETCURSOR(0)
@ 0,0,24,79 BOX '.........'
SETCOLOR("W/B")
SETPOS( 5,10); ?? [******************************]
SETPOS( 6,10); ?? [--    --  --    --  -------]
SETPOS( 7,10); ?? [ --  ---  --  ----  --   ]
SETPOS( 8,10); ?? [  -- --    ---------  --   ]
SETPOS( 9,10); ?? [  ---     ----  --  --   ]
SETPOS(10,10); ?? [  ---     --    --  -------]
SETPOS(11,10); ?? [******************************]
FOR I  = 5 TO 15
  SCROLL(5,10,15,40,-1)
  INKEY(0.5)
NEXT
```

& 3.167 SECONDS()

! รูปแบบ

SECONDS() --> จำนวนวินาทีปัจจุบัน

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ หาค่าวินาทีจากเวลา ของวันนั้น

: ตัวอย่างที่ 3.200

```
? TIME()                              // 19:56:50
? SECONDS()                           //  71810.41
// ถ้า 20:00:00 จะได้ 72000 วินาที ซึ่งเกิดจาก (20 * 60 * 60)
// ดังนั้น 71810.41 จึงเกิดจาก
// (19 * 3600) + (56 * 60) + 50 ได้ 71810
// ส่วน 0.41 เป็นความละเอียด ที่ไม่สามารถมองเห็นใน TIME()
// โปรแกรมข้างล่างนี้ แสดงการจับเวลาว่า ใช้เวลาเท่าใดในการกดปุ่ม
START = SECONDS()
INKEY(0)
ELAPSED = SECONDS() - START
```

```
? ELAPSED                                  //       79.12
? "MINUTE : " + STR((ELAPSED / 60),2)      // MINUTE :  1
? "SECOND : " + STR(MOD(ELAPSED,60),5,2)   // SECOND : 19.12
```

& 3.168 SELECT()

! รูปแบบ

SELECT([<สมนาม>]) --> เลขที่ของพื้นที่ทำงาน

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ระบุพื้นที่ทำงาน (WORK AREA)

: ตัวอย่างที่ 3.201

```
SELECT 1                        // ต้องสั่งเลือกพื้นที่ก่อนเปิดแฟ้ม
USE FILEA
? SELECT()                      // 1
SELECT 2                        // ต้องสั่งเลือกพื้นที่ก่อนเปิดแฟ้ม
USE FILEB
? SELECT()                      // 2
USE FILEA                       // เปลี่ยนแฟ้มที่เปิดในพื้นที่ ที่ 2
? SELECT()                      // 2
CLEAR ALL                       // ยกเลิกตัวแปร และการใช้แฟ้มในทุกพื้นที่
USE FILEA  NEW                  // เติม NEW หลัง USE จะเปิดพื้นที่ใหม่ให้
? SELECT()                      // 1
SS = SELECT()
USE FILEB  NEW
? SELECT()                      // 2
SELECT(SS)
USE FILEA                       // เปลี่ยน หรือเปิดแฟ้มเดิมในพื้นที่ ที่ 1
? SELECT()                      // 1
? SELECT("FILEB")               // 2
? SELECT("FILEA")               // 1
USE FILEB
? SELECT("FILEB")               // 2
? SELECT("FILEA")               // 0
? SELECT()                      // 1
```

& 3.169 SET()

! รูปแบบ

SET(<เลขระบุตัวเลือก>, [<ค่าใหม่>]) --> ค่าที่ถูกตั้งใหม่

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ปรับปรุง หรือตรวจสอบระบบ

SET VALUES DEFINED IN SET.CH

CONSTANT	VALUE TYPE	CONSTANT	VALUE TYPE
1 _SET_EXACT	LOGICAL	20 _SET_DEVICE	CHARACTER
2 _SET_FIXED	LOGICAL	21 _SET_EXTRA	LOGICAL
3 _SET_DECIMALS	NUMERIC	22 _SET_EXTRAFILE	CHARACTER
4 _SET_DATEFORMAT	CHARACTER	23 _SET_PRINTER	LOGICAL
5 _SET_EPOCH	NUMERIC	24 _SET_PRINTFILE	CHARACTER
6 _SET_PATH	CHARACTER	25 _SET_MARGIN	NUMERIC
7 _SET_DEFAULT	CHARACTER	26 _SET_BELL	LOGICAL
8 _SET_EXCLUSIVE	LOGICAL	27 _SET_CONFIRM	LOGICAL
9 _SET_SOFTSEEK	LOGICAL	28 _SET_ESCAPE	LOGICAL
10 _SET_UNIQUE	LOGICAL	29 _SET_INSERT	LOGICAL
11 _SET_DELETED	LOGICAL	30 _SET_EXIT	LOGICAL
12 _SET_CANCEL	LOGICAL	31 _SET_INTENSITY	LOGICAL
13 _SET_DEBUG	LOGICAL	32 _SET_SCOREBOARD	LOGICAL
14 _SET_TYPEAHEAD	NUMERIC	33 _SET_DELIMITERS	LOGICAL
15 _SET_COLOR	CHARACTER	34 _SET_DELIMCHARS	CHARACTER
16 _SET_CURSOR	NUMERIC	35 _SET_WRAP	LOGICAL
17 _SET_CONSOLE	LOGICAL	36 _SET_MESSAGE	NUMERIC
18 _SET_ALTERNATE	LOGICAL	37 _SET_MCENTER	LOGICAL
19 _SET_ALTFILE	CHARACTER	38 _SET_SCROLLBREAK	LOGICAL

: ตัวอย่างที่ 3.202

```
//  #INCLUDE "SET.CH"          // ไม่จำเป็นต้องมีบรรทัดนี้
? _SET_COUNT                   // มีค่าเป็น 38
FOR I = 1 TO _SET_COUNT
  ? SET(I)                     // พิมพ์ค่าของการเซตทั้ง 38 ตัว
  INKEY(0.5)                   // ทำให้การแสดงผลหยุด 0.5 วินาที
NEXT
SET FIXED ON                   // ต้องมีบรรทัดนี้ จึงทำให้ตำแหน่งทศนิยมถูกต้อง
SET(_SET_DECIMALS,5)           // ค่าของ _SET_DECIMALS เท่ากับ 3
? SET(3)                       //       5.00000
```

```
? 1                          //     1.00000
SET(3,6)
? SET(3)                     //     6.000000
? 1                          //     1.000000
```

& 3.170 SETBLINK()

! รูปแบบ

SETBLINK([<ตรรกนิพจน์>]) --> ผลการตั้งค่าใหม่

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ทำให้เกิดการกระพริบ

: ตัวอย่างที่ 3.203

```
SETCOLOR("*W/B,*B/W")        // ใช้คู่กับ SETBLINK
A = SPACE(3)
SETBLINK(.F.)
@ 10,10 SAY "ABCDE" GET A
READ
SETBLINK(.T.)                // ในเซตสีต้องมี * จึงทำให้กระพริบ
@ 11,10 SAY "ABCDE" GET A
READ
SETBLINK(.F.)
@ 12,10 SAY "ABCDE" GET A
READ
```

& 3.171 SETCANCEL()

! รูปแบบ

SETCANCEL([<ตรรกนิพจน์>]) --> ผลการตั้งค่าใหม่

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ระบุไม่ให้หยุดโปรแกรม แบบทันทีทันใดด้วยปุ่ม CTRL-BREAK หรือ ALT-C

: ตัวอย่างที่ 3.204

```
SETCANCEL(.F.)
FOR I = 1 TO 10              // แต่โปรแกรมจะหลุด เมื่อกดปุ่มใดๆ 10 ครั้ง
  X = INKEY(0)
  IF X = 1                   // กดปุ่ม HOME เพื่ออนุญาติให้หยุดโปรแกรมได้
    SETCANCEL(.T.)           // โดยหยุดโปรแกรมเมื่อกดปุ่ม
```

```
    ENDIF                          // ALT-C หรือ CTRL-BREAK
  NEXT
```

& 3.172 SETCOLOR()

! รูปแบบ

SETCOLOR([<ข้อความแสดงสี>]) --> ข้อความแสดงสี

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ระบุสีให้จอภาพ

: ตัวอย่างที่ 3.205

```
SETCOLOR("W/N,BG+/B,B,,B/W")      // STANDARD   W/N
@ 5,5 TO 8,75 DOUBLE              // ENHANCED   BG+/B สำหรับ GET
X = SPACE(5)                      // BORDER     B เส้นขอบนอกสุด
Y = SPACE(5)                      // BACKGROUND (NO SUPPORTED)
@ 6,10 SAY "TEST 1 :" GET X  // UNSLECTED  B/W  GET ที่ยังไม่ถูกเลือก
@ 7,10 SAY "TEST 2 :" GET Y
READ
FORM1 = "B/W,B/BG+,W,,GR+/W"
SETCOLOR(FORM1)
@ 10,10 SAY "TEST 3 :" GET Y
READ
```

& 3.173 SETCURSOR()

! รูปแบบ

SETCURSOR([<เลขระบุแบบตัวกระพริบ>]) --> เลขระบุแบบตัวกระพริบ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ เปลี่ยนรูปแบบของตัวกระพริบ (CURSOR)

: ตัวอย่างที่ 3.206

```
SETCURSOR(0)                      // NONE CURSOR
INKEY(0)
SETCURSOR(1)                      // UNDERLINE CURSOR & SET CURSOR ON
INKEY(0)
SETCURSOR(2)                      // LOWER HALF-BLOCK CURSOR
INKEY(0) ;     SETCURSOR(3)       // FULL BLOCK CURSOR
INKEY(0) ;     SETCURSOR(4)       // UPPER HALF-BLOCK CURSOR
```

```
INKEY(0) ;      SET CURSOR OFF        // NONE CURSOR
INKEY(0)
```

& 3.174 SETKEY()

! รูปแบบ

SETKEY(<เลขปุ่มบรรแป้นพิมพ์>, [<กล่องปฏิบัติการ>]) --> กล่องปฏิบัติการ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ มอบหมายการกดปุ่มให้ไปทำโปรแกรมย่อยที่ต้องการ

: ตัวอย่างที่ 3.207

```
#DEFINE HOME 1
SETKEY(HOME,{|X|H()})              // HOME
SETKEY(27  ,{|X|E()})              // ESC
A = INKEY(0)
Z = SETKEY(A)
IF Z != NIL ; EVAL(Z) ; ENDIF
FUNCTION H
  ? "YOU PRESS HOME"
RETURN
FUNCTION E
  ? "YOU PRESS ESC"
RETURN
```

& 3.175 SETMODE()

! รูปแบบ

SETMODE(<จำนวนแถว>, <จำนวนหลัก>) --> ผลการตั้งค่าใหม่

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบ MODE การแสดงผล ว่าใช้กับ MODE ใดได้

: ตัวอย่างที่ 3.208

```
? SETMODE(25,80)                   // .T.
? SETMODE(43,80)                   // .T.
? SETMODE(10,50)                   // .F.
IF SETMODE(25,80)
  ? "25 ROWS AND 80 COLUMNS AVAILABLE"
ELSE
```

```
   ? "INVALID MODE"
  ENDIF
```

& 3.176 SETPOS()

! รูปแบบ

SETPOS(<เลขแถว>, <เลขหลัก>) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ย้ายตำแหน่งของตัวกระพริบ(CURSOR) บนจอภาพ

: ตัวอย่างที่ 3.209

```
SETPOS(5,0)
?? "---"
SETPOS(5,4)
?? "---"                              // --- ---
?  "---"                              // ---
FOR I = 1 TO 3                        //  12
 FOR J = 1 TO 2                       //  12
   SETPOS(7+I,J) ; ?? LTRIM(STR(J))   //  12
 NEXT
NEXT
FOR I = 1 TO 3                        //  11
 FOR J = 1 TO 2                       //  22
   SETPOS(10+I,J) ; ?? STR(I,1,1)     //  33
 NEXT
NEXT
```

& 3.177 SETPRC()

! รูปแบบ

SETPRC(<เลขแถว>, <เลขหลัก>) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ เซตตำแหน่งของหัวพิมพ์บนเครื่องพิมพ์ เพื่อเปลี่ยนค่า PROW() และ PCOL()

: ตัวอย่างที่ 3.210

```
SETPRC(1,1)
SETPRC(PROW()+1,PCOL())
SETPRC(PROW(),PCOL()+1)
SETPRC(PROW()+1,10)
```

& 3.178 SPACE()

! รูปแบบ

SPACE(<จำนวนเลข>) --> ข้อความที่เป็นช่องว่าง

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ เปลี่ยนตัวอักษรเป็นช่องว่าง

: ตัวอย่างที่ 3.211

```
X = SPACE(5)
Y = SPACE(10)
Z = 20
A = SPACE(Z)
FILEA->FIELD1 = SPACE(LEN(FILEB->FIELD1))
INKEY(0)
FILEA->FIELD1 = LASTKEY()
```

& 3.179 SQRT()

! รูปแบบ

SQRT(<จำนวนเลข>) --> ผลการถอดรากที่ 2

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ส่งค่ารากที่สอง

: ตัวอย่างที่ 3.212

```
SET DECIMALS TO 7
? SQRT(4)                    // 2.0000000
SET DECIMALS TO 5
? SQRT(2)                    // 1.41421
SET DECIMALS TO 3
? SQRT(9)                    // 3.000
? SQRT(9) ** 2               // 9.000
? SQRT(9) ** 0               // 1.000
? SQRT(9) ** 1               // 3.000
```

```
? SQRT(4) ** 3                  // 8.000
? SQRT(4) ^ 3                   // 8.000
? SQRT(81) ^ 2                  // 81.000
X = 5 * 5
? SQRT(X)                       // 5.000
```

& 3.180 STR()

! รูปแบบ

STR(<ตัวเลข>, <ความยาว>, <จำนวนทศนิยม>) --> ข้อความ

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ เปลี่ยนตัวเลขเป็นตัวอักษร

: ตัวอย่างที่ 3.213

```
? "*"                           // *
? STR(5,6,2)                    //   5.00
? STR(5.25,6,2)                 //   5.25
? STR(5.25,7,2)                 //    5.25
? STR(5.25,8,2)                 //     5.25
? STR(5.25,1,2)                 // 5
? STR(5.25,2,2)                 // **  แสดงทศนิยมไม่ได้
? STR(5.25,3,2)                 // *** แสดงทศนิยมไม่ได้
? STR(5.25,4,2)                 // 5.25
? STR(5.25,4)                   //    5
? STR(12.69,1,2)                // *   ขนาดเล็กเกินไปที่จะแสดงตัวเลข
? STR(12.69,2,2)                // **
? STR(12.69,3,2)                // ***
? STR(12.69,3)                  //  13
? STR(12.69,5,2)                // 12.69
? STR(12.69,4,1)                // 12.7
X = 5
? STR(X,3)                      //   5
X = 5.2499
? STR(X,3,1)                    // 5.2
? STR(X,3)                      // 5
```

& 3.181 STRTRAN()

! รูปแบบ

STRTRAN(<ข้อความหลัก>, <ข้อความที่นำไปค้นหา>,
[<ข้อความที่นำไปแทนที่>], [<ตำแหน่งเริ่ม>], [<จำนวนตำแหน่ง>])
--> ข้อความใหม่

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ค้นหาและแทนที่ข้อความด้วยข้อความ

: ตัวอย่างที่ 3.214

```
? STRTRAN("ABCDEFGH","C","C")       // ABCDEFGH
? STRTRAN("ABABABAB","B","A")       // AAAAAAAA
X = UPPER(STRTRAN("ABCDEFGH","C","C"))
? X
X = X + X
? STRTRAN(X,"ABC","123")            // 123DEFGH123DEFGH
```

& 3.182 STUFF()

! รูปแบบ

STUFF(<ข้อความหลัก>, <ตำแหน่งเริ่ม>, <จำนวนลบ>, <ตัวอักษรที่แทรกเข้าไป>)
--> ข้อความใหม่

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ลบและแทรกข้อความ ด้วยข้อความที่ต้องการในคำสั่งเดียวกัน

: ตัวอย่างที่ 3.215

```
? STUFF("ABCDEF",3,1,"")            // ABDEF
? STUFF("ABCDEF",3,1,"C")           // ABCDEF
? STUFF("ABCDEF",3,2,"C")           // ABCEF
? STUFF("ABCDEF",4,3,"CBA")         // ABCCBA
X = "MANY PEOPLE AT BANGKOK"
? STUFF(X,1,15,"")                  // BANGKOK
? STUFF(X,1,0,"HI! ")               // HI! MANY PEOPLE AT BANGKOK
? STUFF(X,13,2,"IN")                // MANY PEOPLE IN BANGKOK
```

& 3.183 SUBSTR()

! รูปแบบ

SUBSTR(<ข้อความหลัก>, <ตำแหน่งเริ่ม>, [<จำนวนตำแหน่ง>]) --> ข้อความใหม่

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แยกข้อความที่ต้องการจากข้อความ

: ตัวอย่างที่ 3.216

```
X = "LAMPANG , THAILAND"
X1 = SUBSTR(X,1,7)
X2 = SUBSTR(X,11,8)
Y = X1 + " , " + X2
? X                                  // LAMPANG , THAILAND
? Y                                  // LAMPANG , THAILAND
```

& 3.184 TIME()

! รูปแบบ

TIME() --> ข้อความเวลา

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แสดงเวลาของระบบ

: ตัวอย่างที่ 3.217

```
? TIME()                             // 13:52:29
X = TIME()
? SUBSTR(X,1,2)                      // 13
? SUBSTR(X,4,2)                      // 52
? SUBSTR(X,7,2)                      // 29
A1 = VAL(SUBSTR(TIME(),1,2))
A2 = SUBSTR(TIME(),4,2)  ;     A3 = TIME()
A = A1 + VAL(A2) + VAL(SUBSTR(A3,7,2))
? A                                  // 94
```

& 3.185 TONE()

! รูปแบบ

TONE(<ความถี่>, <ความนาน>) --> NIL

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ฟังก์ชันออกเสียงผ่านลำโพง

โน็ต และความถี่			
C 130.80	G 196.00	MID C 261.70	G 392.00
C# 138.60	G# 207.70	C# 277.20	G# 415.30
D 146.80	A 200.00	D 293.70	A 440.00
D# 156.60	A# 233.10	D# 311.10	A# 466.20
E 164.80	B 246.90	E 329.60	B 493.90
F 174.60	F 349.20	C 523.30	
F# 185.00	F# 370.00		

: ตัวอย่างที่ 3.218

```
TONE(130.80,18)          // เสียงโน็ต C นาน 1 วินาที
TONE(146.80,36)          // เสียงโน็ต D นาน 2 วินาที
TONE(130.80,1)           // เสียงโน็ต C นาน 1/18 วินาที
TONE(130.80,48)          // เสียงโน็ต C นาน 3 วินาที
```

& 3.186 TRANSFORM()

! รูปแบบ

TRANSFORM(<นิพจน์>, <รูปแบบที่ต้องการ>) --> นิพจน์ใหม่

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แปลงค่าให้ไปอยู่ในรูปแบบของตัวอักษร

ตัวแบบที่ใช้เปลี่ยนรูปแบบ	
A,N,X,9,#	สำหรับการแปลงอักษร และตัวเลข
L	แสดงค่า T,F สำหรับค่าแบบตรรกะ
Y	แสดงค่า Y,N สำหรับค่าแบบตรรกะ
!	แสดงตัวอักษรตัวใหญ่
$,*	แสดงเครื่องหมายนี้หน้าตัวเลข
. หรือ ,	แต่งข้อมูลตัวเลขให้เข้าใจได้ง่าย
ฟังก์ชันที่ใช้เปลี่ยนรูปแบบ	
B	ให้ข้อมูลแสดงแบบชิดซ้าย และมักใช้กับตัวเลข
C	เติม CR หลังค่าที่เป็น +
X	เติม DR หลังค่าที่เป็น -
D	แสดงวันที่ตามรูปแบบที่กำหนดด้วย SET DATE
E	แสดงวันที่ตามแบบของอังกฤษ (BRITISH)
R	ไม่มีการจัดรูปแบบ
Z	ถ้ามี 0 หน้าตัวเลขให้แสดงช่องว่างแทน
(	ค่าลบ ให้อยู่ภายในเครื่องหมาย ()
!	ให้เปลี่ยนเป็นตัวอักษรใหญ่

: ตัวอย่างที่ 3.219

```
? TRANSFORM(123,"@C")            //      123 CR
? TRANSFORM(-123,"@X")           //      123 DB
? TRANSFORM(-123,"@R")           //     -123
? TRANSFORM(-123,"")             //     -123
? TRANSFORM(-123,"C")            // C
? TRANSFORM(-123,"@B")           // -123
? TRANSFORM(-0123,"@B")          // -123
? TRANSFORM(-01230,"@Z")         //     -1230
? TRANSFORM(1234.56,"9999999")   //   1235
? TRANSFORM(1234.56,"$99,999")   // $ 1,235
? TRANSFORM("ABCD","@!")         // ABCD
? TRANSFORM("ABCD","!AA")        // ABC
SET DATE TO ANSI                 // YY.MM.DD
? TRANSFORM(DATE(),"@D")         // 96.03.13
SET DATE TO AMERICAN
? TRANSFORM("04/15/96","@D")     // 04/15/96 ตามที่ SET DATE
? TRANSFORM("04/15/96","@E")     // 15/04/96  BRITISH
? TRANSFORM(123,"$9999")         // $ 123
? TRANSFORM(-123,"@(")           // (     123)
? TRANSFORM("-123","@(")         // -123
```

& 3.187 TYPE()

! รูปแบบ

TYPE(<นิพจน์>) --> แบบของนิพจน์

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ระบุแบบของข้อมูลในชุดเงื่อนไขของนิพจน์

แบบของนิพจน์		
A ARRAY	L LOGICAL	U NIL,LOCAL OR STATIC
B BLOCK	M MEMO	UE ERROR SYNTACTICAL
C CHARACTER	N NUMERIC	UI ERROR INDETERMINATE
D DATE	O OBJECT	

: ตัวอย่างที่ 3.220

```
? TYPE("LEFT('12AA',2)")          // C
? TYPE("VAL(LEFT('12AA',2))")     // N
? TYPE("IF(5>2,'MORE',10)")       // C
? TYPE("IF(5<2,'MORE',10)")       // N
? TYPE("A")                       // U
? TYPE('B')                       // U
? TYPE("C()")                     // U
? TYPE("UDF()")                   // U
? TYPE('IF(2-3)')                 // UE
? TYPE('.T.')                     // L
? TYPE("IF(2<3,.T.,.F.)")         // L
```

& 3.188 UPDATED()

! รูปแบบ

UPDATED() --> ตรรกแจ้งว่ามีการปรับปรุงข้อมูลหรือไม่

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบการปรับปรุงข้อมูล จากผลการ GET ... READ

: ตัวอย่างที่ 3.221

```
A = 5
B = "TEST"
@ 5,5 SAY "GET A : " GET A
@ 6,5 SAY "GET B : " GET B
READ
IF UPDATED()
  ? "YOU CAN MANAGE WITH YOUR FILES."
ELSE
  ? "NO UPDATED"
ENDIF
```

& 3.189 UPPER()

! รูปแบบ

UPPER(<ข้อความ>) --> ข้อความที่เป็นตัวใหญ่หมด

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ เปลี่ยนตัวอักษรให้เป็นตัวใหญ่หมด

: ตัวอย่างที่ 3.222

```
? UPPER("AbC")                    // ABC
? UPPER("abC")                    // ABC
```

& 3.190 USED()

! รูปแบบ

USED() --> ตรรกแจ้งว่ามีการเปิดแฟ้มหรือไม่

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ตรวจสอบว่ามีการใช้แฟ้มอยู่หรือไม่

: ตัวอย่างที่ 3.223

```
USE FILEA
? USED()                    // .T.
SELE 2
? USED()                    // .F.
SELE 1
? USED()                    // .T.
CLOSE
? USED()                    // .F.
```

& 3.191 VAL()

! รูปแบบ

VAL(<ข้อความที่เป็นตัวเลข>) --> ตัวเลข

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ เปลี่ยนตัวอักษรเป็นตัวเลข

: ตัวอย่างที่ 3.224

```
? VAL("1.94499")            // 1.94499
? VAL("1.945")              // 1.945
? VAL(" 1.945")             //  1.945
? VAL(" 1.945 ")            //  1.9450
? VAL(" 1. 945 ")           //  1.00000
? VAL(" 5. ") + VAL(" 11 ") //      16.0
? VAL("A")                  //  0
? VAL(" A")                 //  0
? VAL(SPACE(0))             //  0
? VAL(SPACE(1))             //  0
? VAL(SPACE(3))             //  0
```

& 3.192 VALTYPE()

! รูปแบบ

VALTYPE(<นิพจน์>) --> แบบของนิพจน์

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ ระบุแบบข้อมูลของนิพจน์

แบบที่ใช้กำหนดนิพจน์		
A ARRAY	L LOGICAL	U NIL
B BLOCK	M MEMO	
C CHARACTER	N NUMERIC	
D DATE	O OBJECT	

: ตัวอย่างที่ 3.225

```
X = ARRAY(20)
? VALTYPE("UNIVERSITY")          // C
? VALTYPE(22/7)                  // N
? VALTYPE(X)                     // A
? VALTYPE(CTOD("12-31-96"))      // D
? VALTYPE(.T.)                   // L
? VALTYPE(3 > 5)                 // L
? VALTYPE("")                    // C
? VALTYPE(NIL)                   // U
```

& 3.193 VERSION()

! รูปแบบ

VERSION() --> ชื่อเวอร์ชั่นของภาษาที่ใช้

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แสดงเลข VERSION ของ CLIPPER

: ตัวอย่างที่ 3.226

```
? VERSION()                      // CLIPPER (R) 5.01
? OS()                           // DOS 6.20
```

& 3.194 YEAR()

! รูปแบบ

YEAR(<วันที่>) --> ปี ค.ศ.

P หน้าที่

ฟังก์ชันนี้จะทำหน้าที่ แปลงวันที่เป็นเลขปีที่ต้องการ

: ตัวอย่างที่ 3.227

```
? DATE()                         // 12/31/02
? YEAR(DATE())                   // 2002
? YEAR(CTOD("05/11/95"))         // 1995
? YEAR(DATE()+5)                 // 2003
```