

BASIC listing: DIRB.BAS

```
*****
' *                               D I R B                               *
' -----
' * Task      : Displays all files in any directory, including      *
' *            subdirectories and volume label names.                *
' *            QuickBASIC and the QB.LIB must be loaded using      *
' *            QB /L QB                                             *
' *            before loading and running this file.                *
' -----
' * Author    : Michael Tischer                                     *
' * Developed on : 07/08/1987                                       *
' * Last update : 04/07/1995                                       *
' *****
DECLARE FUNCTION MakeWord! (INum AS INTEGER)
DECLARE FUNCTION Dat$ (IVal AS INTEGER)
DECLARE SUB SetDTA (Segment AS LONG, Offset AS LONG)
DECLARE SUB ScreenDesign ()
DECLARE FUNCTION Month$ (Mon AS INTEGER)
DECLARE FUNCTION FindNext% ()
DECLARE FUNCTION FindFirst% (DFilename AS STRING, Attr AS INTEGER)
DECLARE SUB PrintData (DirBuf AS ANY)
DECLARE SUB Dir (DPath AS STRING, Attr AS INTEGER)

' $INCLUDE: 'QB.BI'                                ' Include file for interrupt call

' -- Directory entry structure, returned in DOS functions 4EH and 4FH --

TYPE DirStruct
    Reserved AS STRING * 21
    Attrib AS STRING * 1
    Time AS INTEGER
```

```
Date AS INTEGER
Size AS LONG
DatName AS STRING * 13
END TYPE
```

```
'-- Constants ----
```

```
CONST TRUE = -1                                'Declare truth
CONST FALSE = NOT TRUE
```

```
CONST FCARRY = 1                                'Carry flag
CONST FENTS = 14                                'Number of entries visible at a time
CONST OWINTOP = (20 - FENTS) \ 2                'Top row of output window
```

```
CONST FAReadOnly = &H1                          'File attributes
CONST FAHidden = &H2
CONST FASysFile = &H4
CONST FAVolumeID = &H8
CONST FADirectory = &H10
CONST FAArchive = &H20
CONST FAAnyFile = &H3F
```

```
'-- Main program -----
```

```
IF COMMAND$ = "" THEN                            'No filename provided?
  CALL Dir("*.*", FAAnyFile) 'No --> Display all files in current dir.
ELSE
  CALL Dir(COMMAND$, FAAnyFile) 'Yes --> Display specified files
END IF
```

```
'*****
'* Dat$      : Converts a value to string for date and time display  *
```



```

IF FindFirst(DPath, Attr) THEN      'Find first entry (same attributes)
DO                                  'Display all entries
    NumOfEntries = NumOfEntries + 1    'Another entry found
    NumInScrn = NumInScrn + 1          'Another entry displayed
    IF NumInScrn = FENTS THEN          'Window full?
        '-- Yes --> Wait for keypress, then display next window's worth
        VIEW PRINT (OWINTOP + 5 + FENTS) TO (OWINTOP + 6 + FENTS)
        PRINT "                      Please press a key "
        SLEEP                          'Wait for keypress
        VIEW PRINT (OWINTOP + 4) TO (OWINTOP + 3 + FENTS)
        NumInScrn = 0                  'Display new entries in window
    END IF
    CALL PrintData(DirBuf)              'Display entry data
    LOOP UNTIL NOT FindNext            'Next entry
END IF
VIEW PRINT (OWINTOP + 5 + FENTS) TO (OWINTOP + 6 + FENTS)
CLS
SELECT CASE NumOfEntries
CASE 0
    PRINT "File not found"
CASE 1
    PRINT " One file found"
CASE ELSE
    PRINT STR$(NumOfEntries); " files found"
END SELECT
VIEW PRINT 1 TO 25
END SUB

```

```

! *****
! * FindFirst : Finds the first directory entry *
! * Input    : Filename and file attribute *
! * Output   : TRUE if the entry is found, otherwise FALSE *

```

```

'* Info          : Entry is placed in the DirBuf variable          *
'*****
FUNCTION FindFirst% (DFilename AS STRING, Attr AS INTEGER)
DIM FBuf AS STRING * 65          'Buffer for filename (as text)
DIM Regs AS RegTypeX             'Processor registers

```

```

FBuf = DFilename                'Transfer filename
FBuf = FBuf + CHR$(0)           'Terminate filename with a null
Regs.ax = &H4E00                'AH = Function number: Search for first
Regs.cx = Attr                  'Search for attribute
Regs.ds = VARSEG(FBuf)          'Segment address of filename
Regs.dx = VARPTR(FBuf)          'Offset address of filename
CALL INTERRUPTX(&H21, Regs, Regs) 'Call DOS interrupt
IF (Regs.flags AND FCARRY) = 0 THEN 'Test carry flag
    FindFirst = TRUE             'Unset = file found
ELSE                             'Set = file not found
    FindFirst = FALSE
END IF
END FUNCTION

```

```

'*****
'* FindNext : Finds the next directory entry          *
'* Input    : None                                    *
'* Output    : TRUE if the entry is found, otherwise FALSE *
'* Info      : This function should execute after GetFirst. The entry *
'*            is placed in the DirBuf variable.        *
'*****
FUNCTION FindNext%

```

```

DIM Regs AS RegTypeX             'Processor registers for interrupt call
Regs.ax = &H4F00                 'AH = 4F: Function number: Search for next

```

```

CALL INTERRUPT(&H21, Regs, Regs)                'Call DOS interrupt
IF (Regs.flags AND FCARRY) = 0 THEN              'Test carry flag
    FindNext = TRUE                             'Unset = file found
ELSE                                             'Set = file not found
    FindNext = FALSE
END IF
END FUNCTION

```

```

*****
' * Makeword : Converts an integer to a long number, which permits      *
' *          BASIC to perform bit shift operations on a negative        *
' *          number using integer division.                             *
' * Input    : The integer number                                       *
' * Output   : Corresponding long number                                *
*****

```

```

FUNCTION MakeWord! (INum AS INTEGER)

```

```

IF INum < 0 THEN
    MakeWord = 65536! + INum
ELSE
    MakeWord = INum
END IF
END FUNCTION

```

```

*****
' * Month    : Displays the month as a string (Jan, Feb, etc.).         *
' * Input    : Number of the month                                       *
' * Output   : Month name as a string                                    *
*****

```

```

FUNCTION Month$(Mon AS INTEGER)

```

```

SELECT CASE Mon

```

```

CASE 1
  Month$ = "Jan"
CASE 2
  Month$ = "Feb"
CASE 3
  Month$ = "Mar"
CASE 4
  Month$ = "Apr"
CASE 5
  Month$ = "May"
CASE 6
  Month$ = "Jun"
CASE 7
  Month$ = "Jul"
CASE 8
  Month$ = "Aug"
CASE 9
  Month$ = "Sep"
CASE 10
  Month$ = "Oct"
CASE 11
  Month$ = "Nov"
CASE 12
  Month$ = "Dec"
END SELECT
END FUNCTION

```

```

! *****
! * PrintData : Display information about a file entry *
! * Input    : DirBufType with file information *
! * Output   : None *
! * Info     : Information for this SUB is taken from the *

```

```

'*                DirBuf variable                *
'*****
SUB PrintData (DirBuf AS DirStruct)

DIM LCounter AS INTEGER                        'Loop counter

PRINT                                           'Display new line in table
LOCATE OWINTOP + FENTS + 3, 15      'Cursor at last line of output window
PRINT "| ";
LCounter = 1
WHILE MID$(DirBuf.DatName, LCounter, 1) <> CHR$(0)  'Repeat until null
    PRINT MID$(DirBuf.DatName, LCounter, 1);      'Display name character
    LCounter = LCounter + 1                        'Next character
WEND

'-- Compute and display file length -----

LOCATE OWINTOP + FENTS + 3, 30                'Position cursor
PRINT USING "| ##### "; DirBuf.Size;

'-- Display date and time -----
LOCATE OWINTOP + FENTS + 3, 40
PRINT "| "; Month$((MakeWord(DirBuf.Date) \ 32) AND 15);      'Show month
PRINT " "; Dat$(MakeWord(DirBuf.Date) AND 31);                'Show day
PRINT USING " ####"; (MakeWord(DirBuf.Date) \ 512) + 1980;    'Show year
LOCATE OWINTOP + FENTS + 3, 53
PRINT "| "; Dat$(MakeWord(DirBuf.Time) \ 2048); " : ";        'Show hour
PRINT Dat$((MakeWord(DirBuf.Time) \ 32) AND 63);              'Show minutes

'-- Display file attributes -----

LOCATE OWINTOP + FENTS + 3, 63

```



```
VIEW PRINT (OWINTOP + 4) TO (OWINTOP + 3 + FENTS)
```

```
END SUB
```

```
! *****  
! * SetDTA : Set DTA address *  
! * Input : Segment and offset address of DTA buffer *  
! * Output : None *  
! *****
```

```
SUB SetDTA (Segment AS LONG, Offset AS LONG)
```

```
DIM Regs AS RegTypeX
```

```
'Processor registers
```

```
Regs.ax = &H1A00
```

```
'AH = 1AH: Function number: Set DTA
```

```
Regs.ds = Segment
```

```
'Segment address of DS register
```

```
Regs.dx = Offset
```

```
'Offset address of DX register
```

```
CALL INTERRUPTX(&H21, Regs, Regs)
```

```
'Call DOS interrupt
```

```
END SUB
```