

## Calling CSPD.COM Functions from Quickbasic

---

### Initial configuration

The Quickbasic libraries, as shipped from Microsoft, do not have support for the *Call Interrupt* statement that must be used to talk to CSPD.COM. *Call Interrupt* is supplied in the library QB.LIB. The easiest way to solve this problem is to merge QB.LIB with BCOM45.LIB and BRUN45.LIB with Microsoft's library manager LIB.EXE. Here's a quick way to do it at the command line.

```
Rename BCOM45.LIB _BCOM45.LIB
Run LIB.EXE
At the Library Name prompt, enter BCOM45.LIB
At the Create? prompt, enter Y
At the Operations? prompt, enter _BCOM45.LIB QB.LIB
```

Ignore the warning message. The resulting BCOM45.LIB now has support for the *Call Interrupt* statement. The same operations can be performed on BRUN45.LIB if you use it instead of BCOM45.LIB.

The following short program can be used to test the merged BCOM45 and BRUN45 libraries. The program will call interrupt 1Ah to get the timer tick and will display it on the console until a key is hit. The program is saved as TICK.BAS

```
start:
    '$include: 'qb.bi'                ' include call interrupt basic code
    open "o",1,"cons:"                ' open console out

    dim InRegs as RegType, OutRegs as RegType 'setup arrays for registers

TickLoop:
    Inregs.ax=0                        ' AH = 0, get timer tick
    call interrupt (&h1A,Inregs,OutRegs) ' call interrupt 1Ah
    print #1,hex$(OutRegs.dx);chr$(&hD); ' print dx followed by return
    if inkey$="" then goto TickLoop     ' loop til key hit
end
```

Compile the program at the command line and run it. It should work on both your development system and your controller board. Note that the ability to call DOS and BIOS interrupts from Quickbasic is a powerful tool and can be used for other purposes besides accessing CSPD.

### Some CSPD Examples

Before running any of these programs on your controller, be sure to first run CSPD.COM. Running CSPD causes it to hook interrupt 63h and remain resident in memory.

The following example, HELLO.BAS will set serial port 0 to 4800 baud with no handshaking and send the text string "hello, world" out the port.

## Calling CSPD.COM Functions from Quickbasic

---

```
start:
    '$include: 'qb.bi'           ' include call interrupt basic code
    open "o",1,"cons:"         ' open console out

    dim InRegs as RegType, OutRegs as RegType 'setup arrays for registers

'
    Set Port 0 baud rate to 4800

    Inregs.ax=&h20*&h100        ' ah=function 20h, al=0
    Inregs.dx=0                 ' port 0
    Inregs.bx=4800              ' 4800 baud
    call interrupt (&h63,Inregs,OutRegs)

'
    Set Port 0 transmit to no handshaking

    Inregs.ax=&h13*&h100        ' ah=function 13h, al=0
    Inregs.dx=0                 ' port 0
    call interrupt (&h63,Inregs,OutRegs)

'
    Send Hello World out Port 0

    OutString$="hello, world"   ' here's our string
    OutString$=OutString$+chr$(13)+chr$(10) ' add CR,LF to string

    for i=1 to len(OutString$)
        Inregs.ax=(&h10*&h100)+asc(mid$(OutString$,i,1))
        Inregs.dx=0
        call interrupt (&h63,Inregs,OutRegs)
    next i
end
```

Here is an example of receiving a string of characters from Serial Port 0. The string is assumed to be terminated with a carriage return. After the string is received, it is printed on the console. 4800 baud and no handshaking is used.

```
start:
    '$include: 'qb.bi'           ' include call interrupt support code
    open "o",1,"cons:"         ' open console out

    dim InRegs as RegType, OutRegs as RegType 'setup arrays for registers

'
    Set Port 0 baud rate to 4800

    Inregs.ax=&h20*&h100        ' ah=function 20h, al=0
    Inregs.dx=0                 ' port 0
    Inregs.bx=4800              ' 4800 baud
    call interrupt (&h63,Inregs,OutRegs)

'
    Set Port 0 receive to no handshaking

    Inregs.ax=&h3*&h100         ' ah=function 13h, al=0
    Inregs.dx=0                 ' port 0
    call interrupt (&h63,Inregs,OutRegs)

    InString$=""
    LastChar$=""
```

