

OPEN

Syntax: *OPEN #channel, device* *channel=0..32767*
 OR *OPEN #channel, device, type* *(Minerva v1.80+ only)*
Location: *QL ROM, Toolkit II*

This is the general command used to open a *channel* to a device for input and/or output, so that data can be read from and written to the specified device.

The channel number can be any integer greater than or equal to zero and should be kept as small as possible because QDOS allocates roughly 40 bytes for each possible channel number below the highest one. So if you open *channel #1000*, 40K of memory would be lost - only badly written programs need a thousand channels. After the *channel* has been *OPENED*, if a program needs to access that device in the future, it can do so by passing that *channel* number to the relevant keyword.

Actually, a dozen channels should be sufficient and the Turbo compiler strictly limits the highest channel number to 15, while QLiberator allows you to configure this to the user's needs via a *\$\$chan* directive. The compilers allocate memory for all of the channels when a job is created so that the channel table of the job is independent of other jobs and cannot be extended or decreased. Under the interpreter, the channel table can be freely extended but not decreased - only *NEW* and *KILL_A* clear the channel tables.

When talking about devices, it is necessary to note the difference between drives (file drivers) and serial devices:

A drive is a medium where files can be stored (eg. floppy disks or microdrive cartridges). Since there can always be several drives of a given type, drive names contain a drive number from 1 to 8. Data is always stored in a stream of bytes. Data can be read in any order and from any position.

On the other hand, with a serial device, data has to be read as it comes in: byte by byte or in larger pieces.

Another type of device is a screen device which is a defined section of the TV or Monitor display itself.

There are also mixtures between all of these types.

The difference between the device types becomes obvious when looking at the operations which can be performed on a device: the files on a drive can be listed in a directory and colours are only available for windows, just to give a few examples.

Other operations (especially basic read and write operations) are independent of the device, which is a characteristic of QDOS. This so-called device independence makes it easy to re-direct basic input or output from a program because the program has no need to know specifics about the device other than its name and/or channel number.

If you have Toolkit II installed *OPEN* supports sub-directories and default devices when used on drives. *OPEN* will look in the data directory (see *DATADS*) for the given file if no device is specified.

Basic details of the various standard devices supported by the QL follow (further details appear in the Drivers Appendix):

Device Type	Name	typical uses
Serial device	ser	printers, communication with other computers or modems, control of processes, reading analogue data.
	par	output to printers via a centronics interface
	nul	a dummy device which simply receives incoming data and immediately forgets it, useful for debugging. There are several variants available.
	pipe	pipes are intended for communication between jobs, every pipe has an input and output side - there are both standard pipes and named pipes. This is a First In First Out device.

Device Type	Name	typical uses
Serial Device	history	Similar to a pipe, except that it is a Last In First Out device.
	net	to send or receive data from another network station
	mem	a device to read and write in memory, especially useful to directly access memory on remote network stations via the fileserver
Drives	mdv	microdrives, the original drives on QLs - files are stored on cartridges
	flp	floppy disk drives are regarded as standard today - files are stored on disks, early drivers are called fdk
	win	winchester drives, also called hard disks - files are stored on a permanently installed very large and fast disk
	ram	ramdisks, virtual but extremely fast drives, the files are stored in RAM and are lost when the computer is switched off

	<code>dev</code>	a kind of universal device, see <code>DEV_USE</code> for an introduction
	<code>pth</code>	very similar to <code>dev</code> - see <code>PTH_ADD</code>
	<code>mos</code>	permanent ramdisk, needs specific hardware
	<code>rom</code>	also a permanent ramdisk.
Windows	<code>con</code>	interfaces to a console device (window) for input and output
	<code>scr</code>	the same as <code>con_</code> but for output only
Other devices	<code>n</code>	the fileserv device which allows you to access any device on a remote network station
	<code>sdump</code>	a device for a general window dump

Please refer to other parts of this book for more specific information on the devices. A lot of examples are given throughout the book.

NOTE 1:

The `OPEN` command will close a channel which is already open with the same channel number prior to opening the new channel - do not try to `OPEN #0` (except from within a compiled program) unless you have Minerva or SMS - even then, do not try to `OPEN #0` as anything other than a `CON_` device, except from within a MultiBASIC / Multiple SBASIC.

NOTE 2:

On AH ROMs, if two tasks tried to read the same file at the same time, the second task was likely to miss the start of the file and read the directory header instead.

NOTE 3:

On QL ROMs (pre MG) there is a maximum of 32767 `OPENS` in a session.

NOTE 4:

The pointer environment has a little bug in it which can lead to odd results when `OPENING` screen windows. Try, for a laugh (and beware that this will crash the QL eventually), the following:

```
FOR I=1 TO 32768: OPEN #3,scr: PRINT#3,'Hello';I
```

This is fixed under SMSQ/E and VMAN v1.52.

NOTE 5:

The maximum number of channels which can be opened at the same time depends on the amount of memory available, but in current implementations, there is an overall maximum of 360 channels, unless you are using Minerva (see below).

SMS seems to allow a much larger number of channels to be open at the same time.

NOTE 6:

Any attempt to open more than one channel to a serial port will report the error 'in use', unless you are using the ST/QL Emulator which allows more than one input channel to be opened to a serial port.

NOTE 7:

On the QXL (pre v2.50 of SMS), an attempt to `OPEN #ch,ser2` would fail if `ser1` was not available to the operating system for any reason.

MINERVA NOTES:

On v1.80 (and later), a third parameter is supported on this command which can be used to specify the 'open type'. This is only of any use on drives and pipes.

Drives	
Open type	Effect
0	Open existing file for exclusive use (same as <code>OPEN</code>)
1	Open existing file for shared use (same as <code>OPEN_IN</code>)
2	Open new file (same as <code>OPEN_NEW</code>)
3	Open file and overwrite if already exists (same as <code>OPEN_OVER</code>)
4	Open directory file (same as <code>OPEN_DIR</code>)

(Compare this list with the list at `FILE_OPEN!`)

Minerva Example:

```
OPEN#3,ram1_test_device,3
```

opens a new file called `ram1_test_device` whether or not it already exists.

Pipes

The extra parameter supplies the QDOS channel number of the source end of the `pipe`. This is therefore only of use when opening the 'read' end of the `pipe`. This gets around the necessity for commands like `QLINK`. For example these two lines are the same:

```
OPEN#4,'pipe_4000':QLINK#4 TO #3
OPEN#4,'pipe_4000':pipe_id=PEEK_W(\48\4*40+2):OPEN#3,'pipe_',pipe_id
```

Unfortunately, Toolkit II replaces this variant of `OPEN` with its own, but all of the above facilities (apart from pipe channel numbers) are provided by specific Toolkit II commands in any event.

Due to Minerva's System Xtensions, the maximum number of permitted channels open at any one time has been reduced to 304 on an expanded machine (earlier ROMs allow 360). On an unexpanded machine, you can only open 112 under Minerva.

In MultiBasics, both channel #0 and channel #1 can be inextricably linked. Due to the fact that the *OPEN* command closes an existing channel before setting up the new channel with the given parameters, *OPEN #0* or *OPEN #1* from within a MultiBasic will remove the MultiBasic in certain instances - see MultiBasic appendix.

CROSS-REFERENCE:

Opened channels are closed with *CLOSE* and can be listed with *CHANNELS*.

FOPEN is the same as *OPEN* except it works as a function and *OPEN_IN* / *FOP_IN* open a device for input only.

OPEN_DIR (*FOP_DIR*) opens a directory (or a sub-directory on level-2 drivers).

Also see *OPEN_NEW*, *FOP_OVER*, *TTEOPEN* and *FILE_OPEN*.

NEWCHAN% can be quite useful when *OPENING* channels.