**General Rules:**
A command will be executed upon the reception of <LF>.
Any wrong command will return „E“.
A return value ends with <CR> <LF>.
A list of return values is separated by <CR> <LF>.

---

**Exit**
Terminate session. Keep state of master.

**SetTimeout(timeoutSec)**
Enables or disables Connection State supervision.
Will terminate connection after {timeoutSec} sec without traffic.

*Parameters:*

| | | |
|---|---|---|
| timeoutSec | n | timeout in seconds (default = 30, 0 = disabled) |

*Return:*

| | | |
|---|---|---|
| timeout= | n | in seconds |

**KeepAlive**
prevents Connection Timeout

*Return:*

1

---

**ListDevices**
Returns List of found USB FTDI devices

*Return:*

| | | |
|---|---|---|
| | n | Number of devices, list with devices follows |
| | dev0 | Only if devices are found |
| | ... | |

**OpenSerial(serial)**
Open connection to selected FTDI Master by device serial number.

*Parameters:*

| | | |
|---|---|---|
| serial number | n | serial number of device |

*Return:*

| | |
|---|---|
| connected= | [0\|1] |

**Close**
Close connection to FTDI Master. Stop Periodic Transfers.

*Return:*

| | |
|---|---|
| connected= | [0\|1] |

---

**SetBitrate(bitrate)**
Select Bitrate

*Parameters:*

| | | |
|---|---|---|
| bitrate | n | in baud (default = 500000) |

*Return:*

| | |
|---|---|
| bitrate= | n |

**SetSync(sync)**
Select SYNC

*Parameters:*

| | | |
|---|---|---|
| sync | 8 (default) | bit |
| | 32 | bit |

*Return:*

| | |
|---|---|
| sync= | n |

**SetHeader(header)**
Select Header format

*Parameters:*

| | | |
|---|---|---|
| header | 3 (default) | byte header |
| | 4 | byte header |

*Return:*

| | |
|---|---|
| header= | n |

**SetParity(parity)**
Select Parity

*Parameters:*

| | | |
|---|---|---|
| parity | 0 (default) | Even |
| | 1 | Odd |
| | 2 | Space |
| | 3 | None |

*Return:*

| | |
|---|---|
| parity= | n |

**SetBreak(break)**
Select Break Length

*Parameters:*

| | | |
|---|---|---|
| break | x,y | double format (default = 13,5) |

*Return:*

| | |
|---|---|
| break= | x,y |

---

**SetNodeAddr(addr)**
Select address of node.

*Parameters:*

| | |
|---|---|
| addr | n |

*Return:*

| | |
|---|---|
| addr= | n |

**SendWakeup(symbol, ack)**
Send Wakeup, symbol if selected, ack if selected.

*Parameters:*

| | | |
|---|---|---|
| symbol | [0\|1] | 1: send wakeup symbol + sleep 25ms |
| ack | [0\|1] | 1: send wakeup ack |

*Return:*

1

**SendSleepBroadcast**
Send Broadcast for Sleep.

*Return:*

1

---

**Write(addr, words, data)**
Write {words} data words starting with {addr} with the same {data}.
Address will be auto-incremented by 2.

*Parameters:*

| | | |
|---|---|---|
| addr | | Byte address in decimal or hex |
| words | | number of words in decimal |
| data | | write data in decimal or hex |

*Return:*

[0|1]

**Read(addr, words)**
Read {words} data words starting with {addr}.
Address will be auto-incremented by 2.

*Parameters:*

| | |
|---|---|
| addr | Byte address in decimal or hex |
| words | number of words in decimal |

*Return:*

| | |
|---|---|
| [0\|1] | 1: list with read data follows |
| data0 | |
| … | |

**Verify(addr, words, mask, expected)**
Verify {words} data words starting with {addr},
checking that {expected} =? (rdata & {mask}).
Address will be auto-incremented by 2.

*Parameters:*

| | |
|---|---|
| addr | Byte address in decimal or hex |
| words | number of words in decimal |
| mask | verify mask in decimal or hex |
| expected | expected check value in decimal or hex |

*Return:*

| | |
|---|---|
| [0\|1] | 0: list with read data follows |
| data0 | |
| … | |

---

**SetPeriodicWrite(addr, data, words)**
Enable periodic write of {words} data words starting with {addr} with the same {data}.
Address will be auto-incremented by 2.

*Parameters:*

| | |
|---|---|
| addr | Byte address in decimal or hex |
| words | number of words in decimal (0 = disable) |
| data | write data in decimal or hex |

*Return:*

1

**SetPeriodicVerify(addr, words, mask, expected)**
Enable periodic verify of {words} data words starting with {addr}, checking that {expected} =? (rdata & {mask}).
Address will be auto-incremented by 2.

*Parameters:*

| | |
|---|---|
| addr | Byte address in decimal or hex |
| words | number of words in decimal (0 = disable) |
| mask | verify mask in decimal or hex |
| expected | expected check value in decimal or hex |

*Return:*

1

**SetPeriodicIntervalMs(interval)**
Set interval in ms between periodic transfers.

*Parameters:*

| | | |
|---|---|---|
| interval | n | in ms |

*Return:*

| | |
|---|---|
| interval= | n |

**StartPeriodic**
Start/Enable configured periodic transfers.

*Return:*

| | |
|---|---|
| periodic= | [0\|1] |

**StopPeriodic**
Stop/Disable periodic transfers.

*Return:*

| | |
|---|---|
| periodic= | [0\|1] |

---

**GetStatus**
Returns and clears status flags

*Return:*

| | | |
|---|---|---|
| com_error= | [0\|1] | Any error detected during transfer (readback, response, timeout, …) |
| verify_error= | [0\|1] | Verify error has occurred. |

**GetLastLiveCounter**
Returns last used Live Counter

*Return:*
n

---

**SetDebugHeaderCrcError(debugHeaderCrcError)**
Select injection of Header CRC Error

*Parameters:*
debugHeaderCrcError [0|1]

*Return:*
debugHeaderCrcError= [0|1]

**SetDebugWriteCrcError(debugWriteCrcError)**
Select injection of Write CRC Error

*Parameters:*
debugWriteCrcError [0|1]

*Return:*
debugWriteCrcError= [0|1]

**SetDebugUseLastLiveCounter(debugUseLastLiveCounter)**
Select injection of LiveCounter Error

*Parameters:*
debugUseLastLiveCounter [0|1]

*Return:*
debugUseLastLiveCounter= [0|1]