

# lua4882

Access to National Instrument's NI-488.2 (GPIB) driver for [OneLuaPro](#).

## Summary

The following commands (mostly traditional NI-488.2 calls; listed in alphabetical order) are implemented.

Function	Purpose
<a href="#">ibask</a>	Return information about software configuration parameters.
<a href="#">ibclr</a>	Clear a specific device.
<a href="#">ibconfig</a>	Change the software configuration input.
<a href="#">ibdev</a>	Open and initialize a GPIB device handle.
<a href="#">ibonl</a>	Place the device or controller interface online or offline.
<a href="#">ibrdr</a>	Read data from a device into a user buffer.
<a href="#">ibrsp</a>	Conduct a serial poll.
<a href="#">ibtrg</a>	Trigger selected device.
<a href="#">ibwait</a>	Wait for GPIB events.
<a href="#">ibwrt</a>	Write data to a device from a user buffer.

## Function Reference

### ibask()

Purpose: Return information about software configuration parameters. This functions is the complement to `ibconfig()`. Valid option identifiers are:

```
"IbcPAD", "IbcSAD", "IbcTMO", "IbcEOT", "IbcPPC", "IbcREADDR", "IbcAUTOPOLL",  
"IbcSC", "IbcSRE", "IbcEOSrd", "IbcEOSwrt", "IbcEOScmp", "IbcEOSchar", "IbcPP2",  
"IbcTIMING", "IbcDMA", "IbcSendLLO", "IbcSPollTime", "IbcPPollTime",  
"IbcEndBitIsNormal", "IbcUnAddr", "IbcHSCableLength", "IbcIst", "IbcRsv",  
"IbcLON", "IbcEOS"
```

The detailed documentation on these options is available at:

- ibconfig Board Configuration Parameter Options: <https://documentation.help/NI-488.2/func7Zjn.html>
- ibconfig Device Configuration Parameter Options: <https://documentation.help/NI-488.2/func9vcj.html>

```

local gpib = require "lua4882"

-- Example 1: Get Autopolling setting from controller interface 0
local optval, stat, errmsg = gpib.ibask(0,"IbCAUTOPOLL")

-- On success:
optval = <AUTOPOLL_STATUS> -- IbCAUTOPOLL may be 0 or 1
stat = <STATUS_TABLE> -- see description for ibclr()
errmsg = nil -- no error message
-- On failure:
optval = nil -- no info available
stat = <STATUS_TABLE> -- see description for ibclr()
errmsg = "Error code and detailed description"

```

## ibclr()

Purpose: Clear a specific device.

```

local gpib = require "lua4882"

-- Returns content of IBSTA as table and an error message
local stat, errmsg = gpib.ibclr(3) -- clears device 3

-- On success:
stat = <STATUS_TABLE> -- see below
errmsg = nil
-- On failure:
handle = <STATUS_TABLE> -- see below
errmsg = "Error code and detailed description"

-- Example with no GPIO-adapter attached, therefore ERR = true and errmsg != nil
-- Individual table element access via normal Lua means, e.g. stat["RQS"] or
stat.RQS
for i,v in pairs(stat) do print(i,v) end
RQS      false
LACS     false
END      false
CMPL     false
TIMO     false
ERR      true
DTAS     false
TACS     false
LOK      false
REM      false
CIC      false
SRQI     false
DCAS     false
ATN      false

print(errmsg)
EHDL:The input handle is invalid

```

For the meaning of the status bits see <https://documentation.help/NI-488.2/gpib2o8j.html>.

## ibconfig()

Purpose: Change the software configuration input. This functions is the complement to `ibask()`.

Valid option identifiers are:

```
"IbcPAD", "IbcSAD", "IbcTMO", "IbcEOT", "IbcPPC", "IbcCREADDR", "IbcAUTOPOLL",  
"IbcSC", "IbcSRE", "IbcEOSrd", "IbcEOSwrt", "IbcEOScmp", "IbcEOSchar", "IbcPP2",  
"IbcTIMING", "IbcDMA", "IbcSendLLO", "IbcSPollTime", "IbcPPollTime",  
"IbcEndBitIsNormal", "IbcUnAddr", "IbcHSCableLength", "IbcIst", "IbcRsv",  
"IbcLON", "IbcEOS"
```

The detailed documentation on these options is available at:

- ibconfig Board Configuration Parameter Options: <https://documentation.help/NI-488.2/func7Zjn.html>
- ibconfig Device Configuration Parameter Options: <https://documentation.help/NI-488.2/func9vcj.html>

```
local gpib = require "lua4882"  
  
-- Example 1: Disable Autopolling on controller interface 0  
local stat, errmsg = gpib.ibconfig(0,"IbcAUTOPOLL",0)  
  
-- Example 2: Change timeout on device 4 to 10s (see timeout table in ibdev()  
below)  
local stat, errmsg = gpib.ibconfig(4,"IbcTMO",13)  
  
-- On success:  
stat = <STATUS_TABLE>    -- see description for ibclr()  
errmsg = nil             -- no error message  
-- On failure:  
stat = <STATUS_TABLE>    -- see description for ibclr()  
errmsg = "Error code and detailed description"
```

## ibdev()

Purpose: Open and initialize a GPIB device handle.

```
local gpib = require "lua4882"  
local boardIndex = 0    -- Commonly used index of bus controller  
local primaryAddr = 1   -- Primary GPIB address of remote device  
local secondaryAddr = 0 -- No secondary GPIB address of remote device  
local timeout = 11      -- Timeout 1 s (see table below)  
local eoiMode = 1       -- Asserts GPIB End-or-Identify (EOI) line at end of  
transfer  
local eosMode = 0       -- No end-of-string character  
  
-- Returns device handle and error message  
local handle, errmsg = gpib.ibdev(boardIndex, primaryAddr, secondaryAddr,  
timeout, eoiMode, eosMode)  
  
-- On success:  
handle = <DEVICE_HANDLE>
```

```

errmsg = nil      -- no error message
-- On failure:
handle = nil      -- no device handle
errmsg = "Error code and detailed description"

-- Interactive example (without an GPIB adapter installed)
Lua 5.4.7 Copyright (C) 1994-2024 Lua.org, PUC-Rio
> gpib=require "lua4882"
> gpib.ibdev(0,3,0,3,1,0)
nil      ENEB:Non-existent interface board

```

The following timeout index values may be used.

Index	0	1	2	3	4	5	6	7	8
Timeout	none	10 μs	30 μs	100 μs	300 μs	1 ms	3 ms	10 ms	30 ms

Index	9	10	11	12	13	14	15	16	17
Timeout	100 ms	300 ms	1 s	3 s	10 s	30 s	100 s	300 s	1000 s

## ibonl()

Purpose: Place the device or controller interface online or offline.

```

local gpib = require "lua4882"

-- Example 1: Enable device 3
local stat, errmsg = gpib.ibonl(3,true)

-- Example 2: Disable device 4
local stat, errmsg = gpib.ibonl(4,false)

-- On success:
stat = <STATUS_TABLE>  -- see description for ibclr()
errmsg = nil           -- no error message
-- On failure:
stat = <STATUS_TABLE>  -- see description for ibclr()
errmsg = "Error code and detailed description"

```

## ibrd()

Purpose: Read data from a device into a user buffer.

Data may be read as contiguous ASCII-string, as table of single ASCII-characters, or as table of raw binary data. Actual number of bytes read may be less than the specified value. This usually happens when the addressed device raises the `END` line during transmission, indicating that no more data is available for transmission.

```

local gpib = require "lua4882"

-- Example 1: read 16 bytes from device 3 as contiguous string

```

```

local data, stat, errmsg = gpib.ibrd(3,16)
-- On success:
data = "<SOME_ASCII_STRING>"
stat = <STATUS_TABLE> -- see description for ibclr()
errmsg = nil -- no error message
-- On failure:
data = nil
handle = <STATUS_TABLE> -- see description for ibclr()
errmsg = "Error code and detailed description"

-- Example 2: read 12 bytes from device 4 as table of ASCII-characters
--           devData = "ABC" : data[1]="A" data[2]="B" data[3]="C"
local data, stat, errmsg = gpib.ibrd(4,12,"charTable")
-- On success:
data = <TABLE_OF_CHARACTERS> -- with Lua 1-based indexing
stat = <STATUS_TABLE> -- see description for ibclr()
errmsg = nil -- no error message
-- On failure:
data = nil
handle = <STATUS_TABLE> -- see description for ibclr()
errmsg = "Error code and detailed description"

-- Example 3: read 8 bytes from device 5 as table of numbers (raw data)
--           devData = "ABC" : data[1]=0x41 data[2]=0x42 data[3]=0x63
local data, stat, errmsg = gpib.ibrd(5,8,"binTable")
-- On success:
data = <TABLE_OF_NUMBERS> -- with Lua 1-based indexing
stat = <STATUS_TABLE> -- see description for ibclr()
errmsg = nil -- no error message
-- On failure:
data = nil
handle = <STATUS_TABLE> -- see description for ibclr()
errmsg = "Error code and detailed description"

```

## ibrsp()

Purpose: Conduct a serial poll.

The Serial Poll Response Byte (SPRB) is presented as a Lua table with boolean values and descriptive keys for easy bitwise access.

```

local gpib = require "lua4882"

-- Conduct serial poll on device 3.
local sprByte, stat, errmsg = gpib.ibrsp(3)

-- On success:
sprByte = <SPRB_TABLE> -- see below
stat = <STATUS_TABLE> -- see description for ibclr()
errmsg = nil -- no error message
-- On failure:
sprByte = nil -- no SPRB data available
handle = <STATUS_TABLE> -- see description for ibclr()

```

```

errmsg = "Error code and detailed description"

-- Serial Poll Response Byte bitwise access (<VARNAME>.bit0 ... <VARNAME>.bit7)
if sprByte.bit6 == true then
    -- If bit 6 (hex 40) of the response is set, the device is requesting
    service.
    -- Usage of bit6 defined in IEEE 488 standard.
    ...
else
    -- No service requested by device.
    ...
end

```

## ibtrg()

Purpose: Trigger selected device.

```

local gpib = require "lua4882"

-- Trigger device 3
local stat, errmsg = gpib.ibtrg(3)

-- On success:
stat = <STATUS_TABLE> -- see description for ibclr()
errmsg = nil -- no error message
-- On failure:
handle = <STATUS_TABLE> -- see description for ibclr()
errmsg = "Error code and detailed description"

```

## ibwait()

Purpose: Wait for GPIB events. Valid wait mask identifiers are:

```

"DCAS", "DTAS", "LACS", "TACS", "ATN", "CIC", "REM", "LOK", "CMPL", "RQS",
"SRQI", "END", "TIMO"

```

For GPIB devices the only valid wait masks are `TIMO`, `END`, `RQS`, and `CMPL`. GPIB controllers accept all wait masks except for `RQS`. Detailed wait mask information is available at <https://documentation.help/NI-488.2/func3kfo.html>.

```

local gpib = require "lua4882"

-- Example 1: Wait (endlessly) for device 3 requesting service
local stat, errmsg = gpib.ibwait(3,"RQS")

-- Example 2: Wait for device 3 requesting service or for timeout
-- Notice that more than one wait mask may be handed over when put into a Lua
table.
local stat, errmsg = gpib.ibwait(3,{"RQS","TIMO"})

-- On success:
stat = <STATUS_TABLE> -- see description for ibclr()
errmsg = nil -- no error message

```

```
-- On failure:
stat = <STATUS_TABLE>    -- see description for ibclr()
errmsg = "Error code and detailed description"
```

## ibwrt()

Purpose: Write data to a device from a user buffer.

```
local gpib = require "lua4882"

-- Write SCPI reset command to device 3
local bytes, stat, errmsg = gpib.ibwrt(3, "*RST\n") -- assumes \n message
terminator

-- On success:
bytes = 5    -- 5 bytes written
stat = <STATUS_TABLE>    -- see description for ibclr()
errmsg = nil    -- no error message
-- On failure:
bytes = nil
handle = <STATUS_TABLE> -- see description for ibclr()
errmsg = "Error code and detailed description"
```

## License

---

See <https://github.com/OneLuaPro/lua4882/blob/master/LICENSE>.