ldoc

Modules

Modules

IOSockets	Module that manages IO functions
debugConfig	Default configuration for the debugger.
rinApp	Module manager for L401
rinLibrary.K400	Library for K400 application support.
rinLibrary.K412	Libary for the K412, providing easy functionality for interfacing with all aspects of the device
rinLibrary.L401	Libary for the L401, providing easy functionality for interfacing with all aspects of the device
rinLibrary.ldocUsage	Documentation of the ldoc features
rinLibrary.rinCSV	Functions for working with .CSV files and creating multi-table databases
rinLibrary.rinDebug	Offer functions for converting variables into strings for debugging
rinLibrary.rinINI	Services for saving and restoring settings in a table to .INI config file
rinLibrary.rinRIS	Creates a connection to the M4223
rinLibrary.rinVT100	Functions for working with VT100 terminal interface
rinLibrary.rincon	Creates a connection to the M4223
rinSystem	Framework for interfacing with the L401 for advanced applications
rinSystem.rinSockets	Offer functions for sockets that are compatible with the app framework
rinSystem.rinTimers	Offer functions for timers that are compatible with the app framework

*generated by **LDoc 1.2***

Module `IOSockets`

Module that manages IO functions

Functions

makeSocket (fdin, fdout)	Makes an imitation socket from a LUA file descriptor that can be used with select.
connectDevice ()	Connects to the user IO
getMsg ()	Get a message from the connection

Functions

makeSocket (fdin, fdout)

Makes an imitation socket from a LUA file descriptor that can be used with select.

Parameters:

- *`fdin`*: File descriptor to take input from
- *`fdout`*: File descriptor to send output to

Returns:

An imitation socket with the methods getfd, dirty, receive, send, close

connectDevice ()

Connects to the user IO

Returns:

the connection (not neccessary for use)

getMsg ()

Get a message from the connection

Returns:

message

*generated by **LDoc 1.2***

Module `debugConfig`

Default configuration for the debugger.

This can be overridden if the user calls a script with rinapp with arguments

*generated by **LDoc 1.2***

Module `rinApp`

Module manager for L401

Functions

addK400 (model, ip, portA, portB)	Called to connect to the K400 instrument, and establish the timers, streams and other services
cleanup ()	Called to restore the system to initial state by shutting down services enabled by configure()

Functions

addK400 (model, ip, portA, portB)

Called to connect to the K400 instrument, and establish the timers, streams and other services

Parameters:

- *`model`*: Software model expected for the instrument (eg "K401")
- *`ip`*: IP address for the socket, "127.1.1.1" used as a default
- *`portA`*: port address for the SERA socket (2222 used as default)
- *`portB`*: port address for the SERB socket (2223 used as default)

Returns:

device object for this instrument

cleanup ()

Called to restore the system to initial state by shutting down services enabled by configure()

*generated by **LDoc 1.2***

Module `rinLibrary.K400`

Library for K400 application support.

Provides wrappers for all device services

Register Functions

sysRegisters	System Registers.
rdgRegisters	Instrument Reading Registers.
usrRegisters	Instrument User Variables.
productRegisters	K412 Product Registers.
rinCMD	Main Instrument Commands.
sendReg (cmd, reg, data)	Called to send command to a register but not wait for the response
sendRegWait (cmd, reg, data, t)	Called to send command and wait for response
readReg (reg)	Called to read register contents
writeReg (reg, data)	Called to write data to an instrument register
exReg (reg, data)	Called to run a register execute command with data as the execute parameter

General Utilities

lcdControl (mode)	Called to setup LCD control
model	Called to connect the K400 library to a socket and a system
getRegDP (reg)	Called to read a register value and return value and dp position Used to work out the dp position of a register value so subsequent reads can use the hexadecimal format and convert locally using toFloat
saveSettings ()	Called to save any changed settings and re-initialise instrument
configure ()	Called to configure the instrument library
toPrimary (v)	Called to convert a floating point value to a decimal integer based on then primary instrument weighing settings
toFloat (data, dp)	called to convert hexadecimal return string to a floating point number

Streaming

streamCallback (data, err)	Divide the data stream up and run the relevant callbacks
addStream (streamReg, callback, onChange)	Add a stream to the device (must be connected) Takes parameter 'change' (default) to run callback only if data received changed, 'always' otherwise
removeStream (streamReg)	Remove a stream from the device
streamCleanup ()	Called to cleanup any unused streaming
setStreamFreq (freq)	Set the frequency used for streaming

Status Monitoring

sysstatus	Status Bits for REG_SYSSTATUS.
statusCallback (data, err)	Called when status changes are streamed
setStatusCallback (stat, callback)	Set the callback function for a status bit
setupStatus ()	Setup status monitoring via a stream
setRTCStatus (s)	Control the use of RTC status bit
setRDGStatus (num)	Control the use of reading count status bit.
setIOStatus ()	private function
enableIOStatus (IO)	sets IO status IO bit to recognise this IO
releaseIOStatus (IO)	sets IO status IO bit to ignore this IO
endStatus ()	Cancel status handling

Key Handling

keys	Keys.
setupKeys ()	Setup key handling stream
endKeys (flush)	Cancel keypress handling
setKeyCallback (key, callback)	Set the callback function for an existing key
keygroups	Key Groups.
sendKey (key, status)	Send an artificial key press to the instrument

LCD Services

writeTopLeft (s)	Write string to Top Left of LCD, curTopLeft is set to s
writeTopRight (s)	Write string to Top Right of LCD, curTopRight is set to s
writeBotLeft (s)	Write string to Bottom Left of LCD, curBotLeft is set to s
writeBotRight (s)	Write string to Bottom Right of LCD, curBotRight is set to s
setAutoTopAnnun (reg)	link register address with Top annunciators to update automatically
setAutoTopLeft (reg)	link register address with Top Left display to update automatically.
setAutoBotLeft (reg)	link register address with Bottom Left display to update automatically
BotAnnuns	Bottom LCD Annunciators
setBitsBotAnnuns (d)	Sets the annunciator bits for Bottom Annunciators
clrBitsBotAnnuns (d)	Clears the annunciator bits for Bottom Annunciators

rotWAIT (dir)	Rotate the WAIT annunciator
TopAnnuns	Top LCD Annunciators
setBitsTopAnnuns (d)	Sets the annunciator bits for Top Annunciators
clrBitsTopAnnuns (d)	Clears the annunciator bits for Top Annunciators
Units	Main Units
Other	Additional modifiers on bottom display
writeTopUnits (units)	Set top units
writeBotUnits (units, other)	Set bottom units
restoreLcd ()	Called to restore the LCD to its default state

Buzzer Control

setBuzzLen (len)	Called to set the length of the buzzer sound
buzz (times)	Called to trigger instrument buzzer

Analogue Output Control

setAnalogSource (src)	Set the analog output type
setAnalogType (typ)	Set the analog output type
setAnalogClip (c)	Control behaviour of analog output outside of normal range.
setAnalogRaw (v)	Sets the analog output to minimum 0 through to maximum 50,000
setAnalogVal (val)	Sets the analogue output to minimum 0.0 through to maximum 1.0
setAnalogPC (val)	Sets the analogue output to minimum 0% through to maximum 100%
setAnalogVolt (val)	Sets the analogue output to minimum 0.0V through to maximum 10.0V
setAnalogCur (val)	Sets the analogue output to minimum 4.0 through to maximum 20.0 mA

Setpoint Control

turnOn (IO)	Turns IO Output on
turnOff (IO)	Turns IO Output off
turnOnTimed (IO, t)	Turns IO Output on
enableOutput (IO)	Sets IO Output under LUA control
releaseOutput (IO)	Sets IO Output under instrument control
setpParam (setp, reg, v)	Private function
setpRegAddress (setp, reg)	returns actual register address for a particular setpoint parameter
setNumSetp (n)	Set the number of Setpoints
setpTarget (setp, target)	Set Target for setpoint
setpIO (setp, IO)	Set which Output the setpoint controls
Types	Setpoint Types.
setpLogic (setp, v)	Set the Logic for the setpoint controls
Alarms	Setpoint Alarms Types.
setpAlarm (setp, v)	Set the Alarm for the setpoint
setpName (setp, v)	Set the Name of the setpoint
Source	Setpoint Source Types.
setpHys (setp, v)	Set the Hysteresis for of the setpoint controls

Dialog Control

getKey (keyGroup)	Called to get a key from specified key group
isEditing ()	Check to see if editing routines active
edit (prompt, def, typ)	Called to prompt operator to enter a value
editReg (reg)	Called to edit value of specified register
delayCallback ()	Private function
delay (t)	Called to delay for t msec while keeping event handlers running
askOKCallback (key, state)	Private function
askOK (prompt, q)	Prompts operator and waits for OK or CANCEL key press
selectOption (prompt, options, def, loop)	Prompts operator to select from a list of options using arrow keys and KEY_OK

Printing Utilities

printCustomTransmit (tokenStr, comPort)	Send custom print token string to instrument comms port
reqCustomTransmit (tokenStr)	Called to request response based on custom transmit token string

Real Time Clock

sendDateFormat (fmt)	sets the instrument date format
RTCread (d)	Read Real Time Clock data from instrument into local RTC table
RTCtick ()	Called every second to update local RTC
RTCtostring ()	Returns formated date/time string Private function
RTCdateFormat (first, second, third)	Sets the order of the date string.byte
REG_ADC_ZERO	Commands TODO: Finalise these commands to return proper error messages etc
zero ()	<>
tare ()	<>

presetTare ()	<>
gross ()	<>
net ()	<>
grossNetToggle ()	<>

Register Functions

Functions to read, write and execute commands on instrument registers directly

sysRegisters

System Registers.

Fields:

- *REG_SOFTMODEL*: Software model eg. "K401"
- *REG_SOFTVER*: Software Version eg "V1.00"
- *REG_SERIALNO*: Serial Number

rdgRegisters

Instrument Reading Registers.

Fields:

- *REG_ADCSAMPLE*: Sample number of current reading
- *REG_SYSSTATUS*: System Status Bits
- *REG_SYSERR*: System Error Bits
- *REG_ABSMVV*: Absolute mV/V reading (10,000 = 1mV/V)
- *REG_GROSSNET*: Gross or Net reading depending on operating mode
- *REG_GROSS*: Gross weight
- *REG_NET*: Net Weight
- *REG_TARE*: Tare Weight
- *REG_PEAKHOLD*: Peak Hold Weight
- *REG_MANHOLD*: Manually Held weight
- *REG_GRANDTOTAL*: Accumulated total
- *REG_ALTGROSS*: Gross weight in secondary units
- *REG_RAWADC*: Raw ADC reading (2,560,000 = 1.0 mV/V)
- *REG_ALTNET*: Net weight in secondary units
- *REG_FULLSCALE*: Fullscale weight

usrRegisters

Instrument User Variables.

Fields:

- *REG_USERID_NAME1*: Names of 5 User ID strings
- *REG_USERID_NAME2*:
- *REG_USERID_NAME3*:
- *REG_USERID_NAME4*:
- *REG_USERID_NAME5*:
- *REG_USERNUM_NAME1*: Names of 5 User ID numbers
- *REG_USERNUM_NAME2*:
- *REG_USERNUM_NAME3*:
- *REG_USERNUM_NAME4*:
- *REG_USERNUM_NAME5*:
- *REG_USERID1*: Data for 5 User ID strings
- *REG_USERID2*: the first 2 are integers
- *REG_USERID3*: the last 3 are weight values
- *REG_USERID4*:
- *REG_USERID5*:
- *REG_USERNUM1*: Data for 5 User ID numbers
- *REG_USERNUM2*:
- *REG_USERNUM3*:
- *REG_USERNUM4*:
- *REG_USERNUM5*:

productRegisters

K412 Product Registers.

Fields:

- *REG_ACTIVE_PRODUCT_NO*: Read the Active Product Number, Write to set the active product by number
- *REG_ACTIVE_PRODUCT_NAME*: Read the Active Product Name, Write to set Active Product by name
- *REG_CLR_ALL_TOTALS*: Clears all product totals (EXECUTE)
- *REG_CLR_DOCKET_TOTALS*: Clears all docket sub-totals (EXECUTE)
- *REG_SELECT_PRODUCT_NO*: Read the Selected Product Number, Write to set the Selected product by number
- *REG_SELECT_PRODUCT_NAME*: Read the Selected Product Name, Write to set the Selected product by Name
- *REG_SELECT_PRODUCT_DELETE*: Delete Selected Product (EXECUTE)
- *REG_SELECT_PRODUCT_RENAME*: Write to change name of selected product

rinCMD

Main Instrument Commands.

Fields:

- `CMD_RDLIT`: Read literal data
- `CMD_RDFINALHEX`: Read data in hexadecimal format
- `CMD_RDFINALDEC`: Read data in decimal format
- `CMD_WRFINALHEX`: Write data in hexadecimal format
- `CMD_WRFINALDEC`: Write data in decimal format
- `CMD_EX`: Execute with data as execute parameter

sendReg (cmd, reg, data)

Called to send command to a register but not wait for the response

Parameters:

- `cmd`: CMD_ command
- `reg`: REG_ register
- `data`: to send

sendRegWait (cmd, reg, data, t)

Called to send command and wait for response

Parameters:

- `cmd`: CMD_ command
- `reg`: REG_ register
- `data`: to send
- `t`: timeout in msec

Returns:

1. reply received from instrument, nil if error
2. err error string if error received, nil otherwise

readReg (reg)

Called to read register contents

Parameters:

- `reg`: REG_ register

Returns:

1. data received from instrument, nil if error
2. err error string if error received, nil otherwise

writeReg (reg, data)

Called to write data to an instrument register

Parameters:

- `reg`: REG_ register
- `data`: to send

exReg (reg, data)

Called to run a register execute command with data as the execute parameter

Parameters:

- `reg`: REG_ register
- `data`: to send

General Utilities

General Functions for configuring the instrument

lcdControl (mode)

Called to setup LCD control

Parameters:

- `mode`: is 'lua' to control display from script or 'default' to return control to the default instrement application

model

Called to connect the K400 library to a socket and a system

getRegDP (reg)

Called to read a register value and return value and dp position Used to work out the dp position of a register value so subsequent reads can use the hexadecimal format and convert locally using toFloat

Parameters:

- *reg*: register to read

Returns:

register value number and dp position

---

saveSettings ()

Called to save any changed settings and re-initialise instrument

---

configure ()

Called to configure the instrument library

Returns:

nil if ok or error string if model doesn't match

---

toPrimary (v)

Called to convert a floating point value to a decimal integer based on then primary instrument weighing settings

Parameters:

- *v*: is value to convert

Returns:

floating point value suitable for a WRFINALDEC

---

toFloat (data, dp)

called to convert hexadecimal return string to a floating point number

Parameters:

- *data*: returned from _CMD_RDFINALHEX or from stream
- *dp*: decimal position

Returns:

floating point number

Streaming

This section is for functions associated with streaming registers

---

streamCallback (data, err)

Divide the data stream up and run the relevant callbacks

Parameters:

- *data*: Data received from register
- *err*: Potential error message

---

addStream (streamReg, callback, onChange)

Add a stream to the device (must be connected) Takes parameter 'change' (default) to run callback only if data received changed, 'always' otherwise

Parameters:

- *streamReg*: Register to stream from (_M.REG_*)
- *callback*: Function to bind to streaming register
- *onChange*: Change parameter return streamReg indentity

---

removeStream (streamReg)

Remove a stream from the device

Parameters:

- *streamReg*: Register to be removed(_M.REG_*)

---

streamCleanup ()

Called to cleanup any unused streaming

---

setStreamFreq (freq)

Set the frequency used for streaming

Parameters:

- *freq*: Frequency of streaming (_M.STM_FREQ_*)

Status Monitoring

Functions are associated with the status monitoring

sysstatus

Status Bits for REG_SYSSTATUS.

Fields:

- *SYS_OVERLOAD*: Scale overloaded
- *SYS_UNDERLOAD*: Scale underload
- *SYS_ERR*: Error active
- *SYS_SETUP*: Instrument in setup mode
- *SYS_CALINPROG*: Instrument calibration in progress
- *SYS_MOTION*: Weight unstable
- *SYS_CENTREOFZERO*: Centre of Zero (within 0.25 divisions of zero)
- *SYS_ZERO*: Weight within zero band setting
- *SYS_NET*: Instrument in Net mode

statusCallback (data, err)

Called when status changes are streamed

Parameters:

- *data*: Data on status streamed
- *err*: Potential error message

setStatusCallback (stat, callback)

Set the callback function for a status bit

Parameters:

- *stat*: STAT_ status bit
- *callback*: Function to run when there is an event on change in status

setupStatus ()

Setup status monitoring via a stream

setRTCStatus (s)

Control the use of RTC status bit

Parameters:

- *s*: true to enable RTC change monitoring, false to disable

setRDGStatus (num)

Control the use of reading count status bit. This is useful if weight readings are not collected via an on change stream register directly

Parameters:

- *num*: Sets amount of readings to trigger a reading count status change

setIOStatus ()

private function

enableIOStatus (IO)

sets IO status IO bit to recognise this IO

Parameters:

- *IO*: is output 1..32

releaseIOStatus (IO)

sets IO status IO bit to ignore this IO

Parameters:

- *IO*: is output 1..32

endStatus ()

Cancel status handling

Key Handling

Functions associated with the handing key presses

keys

Keys.

Fields:

- *KEY_0*:
- *KEY_1*:

- *KEY_2*:
- *KEY_3*:
- *KEY_4*:
- *KEY_5*:
- *KEY_6*:
- *KEY_7*:
- *KEY_8*:
- *KEY_9*:
- *KEY_POWER*:
- *KEY_ZERO*:
- *KEY_TARE*:
- *KEY_SEL*:
- *KEY_F1*:
- *KEY_F2*:
- *KEY_F3*:
- *KEY_PLUSMINUS*:
- *KEY_DP*:
- *KEY_CANCEL*:
- *KEY_UP*:
- *KEY_DOWN*:
- *KEY_OK*:
- *KEY_SETUP*:
- *KEY_PWR_ZERO*:
- *KEY_PWR_TARE*:
- *KEY_PWR_SEL*:
- *KEY_PWR_F1*:
- *KEY_PWR_F2*:
- *KEY_PWR_F3*:
- *KEY_PWR_CANCEL*:

---

setupKeys ()

Setup key handling stream

---

endKeys (flush)

Cancel keypress handling

Parameters:

- *flush*: Flush the current keypresses that have not yet been handled

---

setKeyCallback (key, callback)

Set the callback function for an existing key

Parameters:

- *key*: to monitor (KEY_*)
- *callback*: Function to run when there is an event for that key. Callback function parameters are key (.KEY_OK etc) and state ('short' or 'long')

Usage:

```
local function F1Pressed(key, state)
  if state == 'short' then
      dbg.info('F1 pressed')
   end
   return true    -- F1 handled here so don't send back to instrument for handling
  end
  dwi.setKeyCallback(dwi.KEY_F1, F1Pressed)
```

---

keygroups

Key Groups.

Fields:

- *keyGroup.all*:
- *keyGroup.primary*:
- *keyGroup.functions*:
- *keyGroup.keypad*:
- *keyGroup.numpad*:
- *keyGroup.cursor*:
- *keyGroup.extended*: Set the callback function for an existing key group

---

sendKey (key, status)

Send an artificial key press to the instrument

Parameters:

- *key*: (.KEY_*)
- *status*: 'long' or 'short'

LCD Services

Functions to configure the LCD

writeTopLeft (s)

Write string to Top Left of LCD, curTopLeft is set to s

Parameters:

- *s*: string to display

writeTopRight (s)

Write string to Top Right of LCD, curTopRight is set to s

Parameters:

- *s*: string to display

writeBotLeft (s)

Write string to Bottom Left of LCD, curBotLeft is set to s

Parameters:

- *s*: string to display

writeBotRight (s)

Write string to Bottom Right of LCD, curBotRight is set to s

Parameters:

- *s*: string to display

setAutoTopAnnun (reg)

link register address with Top annunciators to update automatically

Parameters:

- *reg*: address of register to link Top Annunciator state to. Set to 0 to enable direct control of the area

setAutoTopLeft (reg)

link register address with Top Left display to update automatically.

Parameters:

- *reg*: address of register to link Top Left display to. Set to 0 to enable direct control of the area

setAutoBotLeft (reg)

link register address with Bottom Left display to update automatically

Parameters:

- *reg*: address of register to link Top Left display to. Set to 0 to enable direct control of the area

BotAnnuns

Bottom LCD Annunciators

Fields:

- *BATTERY*:
- *CLOCK*:
- *BAT_LO*:
- *BAT_MIDL*:
- *BAT_MIDH*:
- *BAT_HI*:
- *BAT_FULL*:
- *WAIT*:
- *WAIT45*:
- *WAIT90*:
- *WAIT135*:
- *WAITALL*:

setBitsBotAnnuns (d)

Sets the annunciator bits for Bottom Annunciators

Parameters:

- *d*: holds bit locations

clrBitsBotAnnuns (d)

Clears the annunciator bits for Bottom Annunciators

Parameters:

- *d*: holds bit locations

rotWAIT (dir)

Rotate the WAIT annunciator

Parameters:

- *dir*: 1 clockwise, -1 anticlockwise 0 no change

TopAnnuns

Top LCD Annunciators

Fields:

- *SIGMA*:
- *BALANCE*:
- *COZ*:
- *HOLD*:
- *MOTION*:
- *NET*:
- *RANGE*:
- *ZERO*:
- *BAL_SEGA*:
- *BAL_SEGB*:
- *BAL_SEGC*:
- *BAL_SEGD*:
- *BAL_SEGE*:
- *BAL_SEGF*:
- *BAL_SEGG*:
- *RANGE_SEGADG*:
- *RANGE_SEGC*:
- *RANGE_SEGE*:

setBitsTopAnnuns (d)

Sets the annunciator bits for Top Annunciators

Parameters:

- *d*: holds bit locations

clrBitsTopAnnuns (d)

Clears the annunciator bits for Top Annunciators

Parameters:

- *d*: holds bit locations

Units

Main Units

Fields:

- *UNITS_NONE*:
- *UNITS_KG*:
- *UNITS_LB*:
- *UNITS_T*:
- *UNITS_G*:
- *UNITS_OZ*:
- *UNITS_N*:
- *UNITS_ARROW_L*:
- *UNITS_P*:
- *UNITS_L*:
- *UNITS_ARROW_H*: REG_DISP UNITS BIT SETTINGS

Other

Additional modifiers on bottom display

Fields:

- *UNITS_OTHER_PER_H*:
- *UNITS_OTHER_PER_M*:
- *UNITS_OTHER_PER_S*:
- *UNITS_OTHER_PC*:
- *UNITS_OTHER_TOT*:

writeTopUnits (units)

Set top units

Parameters:

- *units*: (.UNITS_NONE etc)

writeBotUnits (units, other)

Set bottom units

Parameters:

- *units*: (.UNITS_NONE etc)
- *other*: (.UNITS_OTHER_PER_H etc)

restoreLcd ()

Called to restore the LCD to its default state

Buzzer Control

Functions to control instrument buzzer

setBuzzLen (len)

Called to set the length of the buzzer sound

Parameters:

- *len*: - length of buzzer sound (BUZZ_SHORT, BUZZ_MEDIUM, BUZZ_LONG)

buzz (times)

Called to trigger instrument buzzer

Parameters:

- *times*: - number of times to buzz, 1..4

Analogue Output Control

Functions to configure and control the analogue output module

setAnalogSource (src)

Set the analog output type

Parameters:

- *src*: Source for output. Must be set to ANALOG_COMMS to control directly

setAnalogType (typ)

Set the analog output type

Parameters:

- *typ*: Type for output (.CUR or .VOLT)

setAnalogClip (c)

Control behaviour of analog output outside of normal range. If clip is active then output will be clipped to the nominal range otherwise the output will drive to the limit of the hardware

Parameters:

- *c*: 0 for clipping disabled, 1 for clipping enabled

setAnalogRaw (v)

Sets the analog output to minimum 0 through to maximum 50,000

Parameters:

- *v*: value in raw counts (0..50000)

setAnalogVal (val)

Sets the analogue output to minimum 0.0 through to maximum 1.0

Parameters:

- *val*: value 0.0 to 1.0

setAnalogPC (val)

Sets the analogue output to minimum 0% through to maximum 100%

Parameters:

- *val*: value 0 to 100 %

setAnalogVolt (val)

Sets the analogue output to minimum 0.0V through to maximum 10.0V

Parameters:

- *val*: value 0.0 to 10.0

setAnalogCur (val)

Sets the analogue output to minimum 4.0 through to maximum 20.0 mA

Parameters:

- *val*: value 4.0 to 20.0

Setpoint Control

Functions to setup adn control setpoint outputs

turnOn (IO)

Turns IO Output on

Parameters:

- *IO*: is output 1..32

turnOff (IO)

Turns IO Output off

Parameters:

- *IO*: is output 1..32

turnOnTimed (IO, t)

Turns IO Output on

Parameters:

- *IO*: is output 1..32
- *t*: is time in milliseconds

enableOutput (IO)

Sets IO Output under LUA control

Parameters:

- *IO*: is input 1..32

releaseOutput (IO)

Sets IO Output under instrument control

Parameters:

- *IO*: is output 1..32

setpParam (setp, reg, v)

Private function

Parameters:

- *setp*:
- *reg*:
- *v*:

setpRegAddress (setp, reg)

returns actual register address for a particular setpoint parameter

Parameters:

- *setp*: is setpoint 1..16
- *reg*: is REG_SETP_*

Returns:

address of this registet for setpoint setp

setNumSetp (n)

Set the number of Setpoints

Parameters:

- *n*: is the number of setpoints 0..8

setpTarget (setp, target)

Set Target for setpoint

Parameters:

- *setp*: Setpoint 1..16

- *target*: Target value

setpIO (setp, IO)

Set which Output the setpoint controls

Parameters:

- *setp*: is setpount 1..16
- *IO*: is output 1..32, 0 for none

Types

Setpoint Types.

Fields:

- *TYPE_OFF*:
- *TYPE_ON*:
- *TYPE_OVER*:
- *TYPE_UNDER*:
- *TYPE_COZ*:
- *TYPE_ZERO*:
- *TYPE_NET*:
- *TYPE_MOTION*:
- *TYPE_ERROR*:
- *TYPE_LGC_AND*:
- *TYPE_LGC_OR*:
- *TYPE_LGC_XOR*:
- *TYPE_BUZZER*: Set the TYPE of the setpoint controls

setpLogic (setp, v)

Set the Logic for the setpoint controls

Parameters:

- *setp*: is setpount 1..16
- *v*: is setpoint logic type (.LOGIC_HIGH, .LOGIC_LOW)

Alarms

Setpoint Alarms Types.

Fields:

- *ALARM_NONE*:
- *ALARM_SINGLE*:
- *ALARM_DOUBLE*:
- *ALARM_FLASH*:

setpAlarm (setp, v)

Set the Alarm for the setpoint

Parameters:

- *setp*: is setpount 1..16
- *v*: is alarm type

setpName (setp, v)

Set the Name of the setpoint

Parameters:

- *setp*: is setpount 1..16
- *v*: is setpoint name (8 character string)

Source

Setpoint Source Types.

Fields:

- *SOURCE_GROSS*:
- *SOURCE_NET*:
- *SOURCE_DISP*:
- *SOURCE_ALT_GROSS*:
- *SOURCE_ALT_NET*:
- *SOURCE_ALT_DISP*:
- *SOURCE_PIECE*:
- *SOURCE_REG*: Set the data source of the setpoint controls

setpHys (setp, v)

Set the Hysteresis for of the setpoint controls

Parameters:

- *setp*: is setpount 1..16
- *v*: is setpoint hysteresis

Dialog Control

Functions for user interface dialogues

getKey (keyGroup)

Called to get a key from specified key group

Parameters:

- *keyGroup*: keyGroup.all is default group

Returns:

key (KEY_), state ('short','long','up')

isEditing ()

Check to see if editing routines active

Returns:

true of editing false otherwise

edit (prompt, def, typ)

Called to prompt operator to enter a value

Parameters:

- *prompt*: string displayed on bottom right LCD
- *def*: default value
- *typ*: type of value to enter ('integer','number','string'

Returns:

value and true if ok pressed at end

editReg (reg)

Called to edit value of specified register

Parameters:

- *reg*: is the address of the register to edit

delayCallback ()

Private function

delay (t)

Called to delay for t msec while keeping event handlers running

Parameters:

- *t*: delay time in msec

askOKCallback (key, state)

Private function

Parameters:

- *key*:
- *state*:

askOK (prompt, q)

Prompts operator and waits for OK or CANCEL key press

Parameters:

- *prompt*: string to put on bottom right LCD
- *q*: string to put on bottom left LCD

Returns:

either KEY_OK or KEY_CANCEL

selectOption (prompt, options, def, loop)

Prompts operator to select from a list of options using arrow keys and KEY_OK

Parameters:

- *prompt*: string to put on bottom right LCD
- *options*: table of option strings
- *def*: default selection string.byte
- *loop*: If true, top option loops to the bottom option and vice versa

Returns:

selected string (default selection if KEY_CANCEL pressed)

Printing Utilities

Functions for printing

printCustomTransmit (tokenStr, comPort)

Send custom print token string to instrument comms port

Parameters:

- *tokenStr*: string containing custom print tokens
- *comPort*: - port to use PRINT_SER1A (default) .. PRINT_SER2B

reqCustomTransmit (tokenStr)

Called to request response based on custom transmit token string

Parameters:

- *tokenStr*: custom token string

Real Time Clock

Functions to control Real Time Clock

sendDateFormat (fmt)

sets the instrument date format

Parameters:

- *fmt*: TM_MMDDYYYY or TM_DDMMYYYY

RTCread (d)

Read Real Time Clock data from instrument into local RTC table

Parameters:

- *d*: 'date' or 'time' to read these fields only, or 'all' for both

RTCtick ()

Called every second to update local RTC

RTCtostring ()

Returns formated date/time string Private function

RTCdateFormat (first, second, third)

Sets the order of the date string.byte

Parameters:

- *first*: = 'day', 'month' or 'year'
- *second*: = 'day', 'monht','year'
- *third*: = 'day','month','year'

REG_ADC_ZERO

Commands TODO: Finalise these commands to return proper error messages etc

zero ()

<>

tare ()

<>

presetTare ()

<>

gross ()

<>

net ()

<>

grossNetToggle ()

	<>	

generated by ***LDoc 1.2***

Module `rinLibrary.K412`

Libary for the K412, providing easy functionality for interfacing with all aspects of the device

Functions

connect (sock, system)	Called to connect the L401 library to a socket and a system
getRegDP (reg)	Called to read a register value and return value and dp position Used to work out the dp position of a register value so subsequent reads can use the hexadecimal format and convert locally using toFloat
saveSetting ()	Called to save any changed settings and re-initialise instrument
toPrimary (v)	Called to convert a floating point value to a decimal integer based on then primary instrument weighing settings
toFloat (data, dp)	called to convert hexadecimal return string to a floating point number
streamCallback (data, err)	Divide the data stream up and run the relevant callbacks
addStream (streamReg, callback, onChange)	Add a stream to the device (must be connected) Takes parameter 'change' (default) to run callback only if data received changed, 'always' otherwise
removeStream (streamReg)	Remove a stream from the device
streamCleanup ()	Called to cleanup any unused streaming
setStreamFreq (freq)	Set the frequency used for streaming
statusCallback (data, err)	Called when status changes are streamed
setStatusCallback (stat, callback)	Set the callback function for a status bit
setupStatus ()	Setup status monitoring via a stream
endStatus ()	Cancel status handling
keyCallback (data, err)	Called when keys are streamed, send the keys to each group it is bound to in order of priority, until one of them returns true.
setKeyCallback (key, callback)	Set the callback function for an existing key
setKeyGroupCallback (keyGroup, callback)	Set the callback function for an existing key group
setupKeys ()	Setup key handling stream
endKeys (flush)	Cancel keypress handling
writeBotLeft (s)	Write string to Bottom Left of LCD, curBotLeft is set to s
writeBotRight (s)	Write string to Bottom Right of LCD, curBotRight is set to s
setBitsBotAnnuns (d)	Sets the annunciator bits for Bottom Annunciators
clrBitsBotAnnuns (d)	Clears the annunciator bits for Bottom Annunciators
rotWAIT (dir)	Rotate the WAIT annunciator
setBitsTopAnnuns (d)	Sets the annunciator bits for Top Annunciators
clrBitsTopAnnuns (d)	Clears the annunciator bits for Top Annunciators
restoreLcd ()	Called to restore the LCD to its default state
setAnalogType (typ)	Set the analog output type
setAnalogVal (val)	Sets the analogue output to minimum 0.0 through to maximum 1.0
setAnalogPC (val)	Sets the analogue output to minimum 0% through to maximum 100%
setAnalogVolt (val)	Sets the analogue output to minimum 0.0V through to maximum 10.0V
setAnalogCur (val)	Sets the analogue output to minimum 4.0 through to maximum 20.0 mA
turnOn (IO)	Turns IO Output on
turnOff (IO)	Turns IO Output off
enableOutput (IO)	Sets IO Output under LUA control
releaseOutput (IO)	Sets IO Output under instrument control
setpParam (setp, reg, v)	Private function
setpRegAddress (setp, reg)	returns actual register address for a particular setpoint parameter
setpTarget (setp, target)	Set Target for setpoint
setpIO (setp, IO)	Set which Output the setpoint controls
setpType (setp, v)	Set the TYPE of the setpoint controls
setpLogic (setp, v)	Set the Logic for the setpoint controls
setpAlarm (setp, v)	Set the Alarm for the setpoint
setpName (setp, v)	Set the Name of the setpoint
setpSource (setp, v)	Set the data source of the setpoint controls
setpHys (setp, v)	Set the Hysteresis for of the setpoint controls
setNumSetp (n)	Set the number of Setpoints
setBuzzLen (len)	Called to set the length of the buzzer sound
buzz (times)	Called to trigger instrument buzzer
getKey (keyGroup)	Called to get a key from specified key group
edit (prompt, def, typ)	Called to prompt operator to enter a value
sendRegCallback (data, err)	Private function

sendRegWait (cmd, reg, data, t)	Called to send command and wait for response
readRegWait (reg)	Called to read register contents
editRegister (reg)	Called to edit value of specified register
delayCallback ()	Private function
delay (t)	Called to delay for t msec while keeping event handlers running
askOKCallback (key, state)	Private function
askOK (prompt, q)	Prompts operator and waits for OK or CANCEL key press
selectOption (prompt, options, def, loop)	Prompts operator to select from a list of options using arrow keys and KEY_OK
printCustomTransmit (tokenStr, comPort)	Send custom print token string to instrument comms port
reqCustomTransmit (tokenStr)	Called to request response based on custom transmit token string
sendDateFormat (fmt)	sets the instrument date format
RTCread (d)	Read Real Time Clock data from instrument into local RTC table
RTCtick ()	Called every second to update local RTC
RTCtostring ()	Returns formated date/time string Private function
RTCdateFormat (first, second, third)	Sets the order of the date string.byte
tare ()	<>
presetTare ()	<>
gross ()	<>
net ()	<>
grossNetToggle ()	<>

Fields

fullscale	Called to collect key system data from instrument to configure the local L401 library
s	Control the use of RTC status bit
num	Control the use of reading count status bit.
mask	private function [[function _M.setIOStatus(mask)
curIOStatus	sets IO status IO bit to recognise this IO
curIOStatus	sets IO status IO bit to ignore this IO
REG_DISP_LAYOUT	LCD Services LCD display registers
REG_ANALOGUE_DATA	Routines for Analogue Output control
REG_IO_STATUS	Routines for Setpoint Output control
getKeyPressed	Routines for Dialog control
REG_TIMECUR	RTC Routines Time and Date
REG_ADC_TARE	Commands TODO: Finalise these commands to return proper error messages etc [[_M.REG_ADC_ZERO = 0x0300 -- Execute registers

Functions

connect (sock, system)

Called to connect the L401 library to a socket and a system

Parameters:

- *sock*: TCP socket to connect
- *system*: System that manages L401 (nil if stand-alone library)

getRegDP (reg)

Called to read a register value and return value and dp position Used to work out the dp position of a register value so subsequent reads can use the hexadecimal format and convert locally using toFloat

Parameters:

- *reg*: register to read

Returns:

register value number and dp position

saveSetting ()

Called to save any changed settings and re-initialise instrument

toPrimary (v)

Called to convert a floating point value to a decimal integer based on then primary instrument weighing settings

Parameters:

- *v*: is value to convert

Returns:

floating point value suitable for a WRFINALDEC

toFloat (data, dp)

called to convert hexadecimal return string to a floating point number

Parameters:

- *data*: returned from _CMD_RDFINALHEX or from stream
- *dp*: decimal position

Returns:

floating point number

streamCallback (data, err)

Divide the data stream up and run the relevant callbacks

Parameters:

- *data*: Data received from register
- *err*: Potential error message

addStream (streamReg, callback, onChange)

Add a stream to the device (must be connected) Takes parameter 'change' (default) to run callback only if data received changed, 'always' otherwise

Parameters:

- *streamReg*: Register to stream from (_M.REG_*)
- *callback*: Function to bind to streaming register
- *onChange*: Change parameter return streamReg indentity

removeStream (streamReg)

Remove a stream from the device

Parameters:

- *streamReg*: Register to be removed(_M.REG_*)

streamCleanup ()

Called to cleanup any unused streaming

setStreamFreq (freq)

Set the frequency used for streaming

Parameters:

- *freq*: Frequency of streaming (_M.STM_FREQ_*)

statusCallback (data, err)

Called when status changes are streamed

Parameters:

- *data*: Data on status streamed
- *err*: Potential error message

setStatusCallback (stat, callback)

Set the callback function for a status bit

Parameters:

- *stat*: given in _M.statBinds
- *callback*: Function to run when there is an event on change in status

setupStatus ()

Setup status monitoring via a stream

endStatus ()

Cancel status handling

keyCallback (data, err)

Called when keys are streamed, send the keys to each group it is bound to in order of priority, until one of them returns true. key states are 'short','long','up' Note: keybind tables should be sorted by priority

Parameters:

- *data*: Data on key streamed
- *err*: Potential error message

setKeyCallback (key, callback)

Set the callback function for an existing key

Parameters:

- *key*: A key given in _M.keyBinds
- *callback*: Function to run when there is an event on the keygroup

setKeyGroupCallback (keyGroup, callback)

Set the callback function for an existing key group

Parameters:

- *keyGroup*: A keygroup given in _M.keyGroup.*
- *callback*: Function to run when there is an event on the keygroup

setupKeys ()

Setup key handling stream

endKeys (flush)

Cancel keypress handling

Parameters:

- *flush*: Flush the current keypresses that have not yet been handled

writeBotLeft (s)

Write string to Bottom Left of LCD, curBotLeft is set to s

Parameters:

- *s*: string to display

writeBotRight (s)

Write string to Bottom Right of LCD, curBotRight is set to s

Parameters:

- *s*: string to display

setBitsBotAnnuns (d)

Sets the annunciator bits for Bottom Annunciators

Parameters:

- *d*: holds bit locations

clrBitsBotAnnuns (d)

Clears the annunciator bits for Bottom Annunciators

Parameters:

- *d*: holds bit locations

rotWAIT (dir)

Rotate the WAIT annunciator

Parameters:

- *dir*: 1 clockwise, -1 anticlockwise 0 no change

setBitsTopAnnuns (d)

Sets the annunciator bits for Top Annunciators

Parameters:

- *d*: holds bit locations

clrBitsTopAnnuns (d)

Clears the annunciator bits for Top Annunciators

Parameters:

- *d*: holds bit locations

restoreLcd ()

Called to restore the LCD to its default state

setAnalogType (typ)

Set the analog output type

Parameters:

- *typ*: Type for output (.CUR or .VOLT)

---

setAnalogVal (val)

Sets the analogue output to minimum 0.0 through to maximum 1.0

Parameters:

- *val*: value 0.0 to 1.0

---

setAnalogPC (val)

Sets the analogue output to minimum 0% through to maximum 100%

Parameters:

- *val*: value 0 to 100 %

---

setAnalogVolt (val)

Sets the analogue output to minimum 0.0V through to maximum 10.0V

Parameters:

- *val*: value 0.0 to 10.0

---

setAnalogCur (val)

Sets the analogue output to minimum 4.0 through to maximum 20.0 mA

Parameters:

- *val*: value 4.0 to 20.0

---

turnOn (IO)

Turns IO Output on

Parameters:

- *IO*: is output 1..32

---

turnOff (IO)

Turns IO Output off

Parameters:

- *IO*: is output 1..32

---

enableOutput (IO)

Sets IO Output under LUA control

Parameters:

- *IO*: is input 1..32

---

releaseOutput (IO)

Sets IO Output under instrument control

Parameters:

- *IO*: is output 1..32

---

setpParam (setp, reg, v)

Private function

Parameters:

- *setp*:
- *reg*:
- *v*:

---

setpRegAddress (setp, reg)

returns actual register address for a particular setpoint parameter

Parameters:

- *setp*: is setpoint 1..16
- *reg*: is REG_SETP_*

Returns:

address of this registet for setpoint setp

setpTarget (setp, target)

Set Target for setpoint

Parameters:

- *setp*: Setpoint 1..16
- *target*: Target value

setpIO (setp, IO)

Set which Output the setpoint controls

Parameters:

- *setp*: is setpount 1..16
- *IO*: is output 1..32, 0 for none

setpType (setp, v)

Set the TYPE of the setpoint controls

Parameters:

- *setp*: is setpount 1..16
- *v*: is setpoint type

setpLogic (setp, v)

Set the Logic for the setpoint controls

Parameters:

- *setp*: is setpount 1..16
- *v*: is setpoint logic type

setpAlarm (setp, v)

Set the Alarm for the setpoint

Parameters:

- *setp*: is setpount 1..16
- *v*: is alarm type

setpName (setp, v)

Set the Name of the setpoint

Parameters:

- *setp*: is setpount 1..16
- *v*: is setpoint name

setpSource (setp, v)

Set the data source of the setpoint controls

Parameters:

- *setp*: is setpount 1..16
- *v*: is setpoint source type

setpHys (setp, v)

Set the Hysteresis for of the setpoint controls

Parameters:

- *setp*: is setpount 1..16
- *v*: is setpoint hysteresis

setNumSetp (n)

Set the number of Setpoints

Parameters:

- *n*: is the number of setpoints 0..16

setBuzzLen (len)

Called to set the length of the buzzer sound

Parameters:

- *len*: - length of buzzer sound (BUZZ_SHORT, BUZZ_MEDIUM, BUZZ_LONG)

buzz (times)

Called to trigger instrument buzzer

Parameters:

- *times*: - number of times to buzz, 1..4

getKey (keyGroup)

Called to get a key from specified key group

Parameters:

- *keyGroup*: keyGroup.all is default group

Returns:

key (KEY_), state ('short','long','up')

edit (prompt, def, typ)

Called to prompt operator to enter a value

Parameters:

- *prompt*: string displayed on bottom right LCD
- *def*: default value
- *typ*: type of value to enter ('integer','number','string'

Returns:

value and true if ok pressed at end

sendRegCallback (data, err)

Private function

Parameters:

- *data*:
- *err*:

sendRegWait (cmd, reg, data, t)

Called to send command and wait for response

Parameters:

- *cmd*: CMD_ command
- *reg*: REG_ register
- *data*: to send
- *t*: timeout in msec

Returns:

1. data received from instrument, nil if error
2. err error string if error received, nil otherwise

readRegWait (reg)

Called to read register contents

Parameters:

- *reg*: REG_ register

Returns:

1. data received from instrument, nil if error
2. err error string if error received, nil otherwise

editRegister (reg)

Called to edit value of specified register

Parameters:

- *reg*: is register to edit

delayCallback ()

Private function

delay (t)

Called to delay for t msec while keeping event handlers running

Parameters:

- *t*: delay time in msec

askOKCallback (key, state)

Private function

Parameters:

- *key*:
- *state*:

askOK (prompt, q)

Prompts operator and waits for OK or CANCEL key press

Parameters:

- *prompt*: string to put on bottom right LCD
- *q*: string to put on bottom left LCD

Returns:

either KEY_OK or KEY_CANCEL

selectOption (prompt, options, def, loop)

Prompts operator to select from a list of options using arrow keys and KEY_OK

Parameters:

- *prompt*: string to put on bottom right LCD
- *options*: table of option strings
- *def*: default selection string.byte
- *loop*: If true, top option loops to the bottom option and vice versa

Returns:

selected string (default selection if KEY_CANCEL pressed)

printCustomTransmit (tokenStr, comPort)

Send custom print token string to instrument comms port

Parameters:

- *tokenStr*: string containing custom print tokens
- *comPort*: - port to use PRINT_SER1A (default) .. PRINT_SER2B

reqCustomTransmit (tokenStr)

Called to request response based on custom transmit token string

Parameters:

- *tokenStr*: custom token string

sendDateFormat (fmt)

sets the instrument date format

Parameters:

- *fmt*: TM_MMDDYYYY or TM_DDMMYYYY

RTCread (d)

Read Real Time Clock data from instrument into local RTC table

Parameters:

- *d*: 'date' or 'time' to read these fields only, or 'all' for both

RTCtick ()

Called every second to update local RTC

RTCtostring ()

Returns formated date/time string Private function

RTCdateFormat (first, second, third)

Sets the order of the date string.byte

Parameters:

- *first*: = 'day', 'month' or 'year'
- *second*: = 'day', 'monht','year'
- *third*: = 'day','month','year'

tare ()

<>

presetTare ()

<>

gross ()

<>

net ()

<>

grossNetToggle ()

<>

Fields

fullscale

Called to collect key system data from instrument to configure the local L401 library

s

Control the use of RTC status bit

num

Control the use of reading count status bit. This is useful if weight readings are not collected via an on change stream register directly

mask

private function [[function _M.setIOStatus(mask)

curIOStatus

sets IO status IO bit to recognise this IO

curIOStatus

sets IO status IO bit to ignore this IO

REG_DISP_LAYOUT

LCD Services LCD display registers

REG_ANALOGUE_DATA

Routines for Analogue Output control

REG_IO_STATUS

Routines for Setpoint Output control

getKeyPressed

Routines for Dialog control

REG_TIMECUR

RTC Routines Time and Date

REG_ADC_TARE

Commands TODO: Finalise these commands to return proper error messages etc [[_M.REG_ADC_ZERO = 0x0300 -- Execute registers

*generated by **LDoc 1.2***

Module `rinLibrary.L401`

Libary for the L401, providing easy functionality for interfacing with all aspects of the device

Functions

connect (sock, system)	Called to connect the L401 library to a socket and a system
getRegDP (reg)	Called to read a register value and return value and dp position Used to work out the dp position of a register value so subsequent reads can use the hexadecimal format and convert locally using toFloat
saveSetting ()	Called to save any changed settings and re-initialise instrument
toPrimary (v)	Called to convert a floating point value to a decimal integer based on then primary instrument weighing settings
toFloat (data, dp)	called to convert hexadecimal return string to a floating point number
streamCallback (data, err)	Divide the data stream up and run the relevant callbacks
addStream (streamReg, callback, onChange)	Add a stream to the device (must be connected) Takes parameter 'change' (default) to run callback only if data received changed, 'always' otherwise
removeStream (streamReg)	Remove a stream from the device
streamCleanup ()	Called to cleanup any unused streaming
setStreamFreq (freq)	Set the frequency used for streaming
statusCallback (data, err)	Called when status changes are streamed
setStatusCallback (stat, callback)	Set the callback function for a status bit
setupStatus ()	Setup status monitoring via a stream
setRTCStatus (s)	Control the use of RTC status bit
setRDGStatus (num)	Control the use of reading count status bit.
setIOStatus ()	private function
enableIOStatus (IO)	sets IO status IO bit to recognise this IO
releaseIOStatus (IO)	sets IO status IO bit to ignore this IO
endStatus ()	Cancel status handling
keyCallback (data, err)	Called when keys are streamed, send the keys to each group it is bound to in order of priority, until one of them returns true.
setKeyCallback (key, callback)	Set the callback function for an existing key
setKeyGroupCallback (keyGroup, callback)	Set the callback function for an existing key group
setupKeys ()	Setup key handling stream
endKeys (flush)	Cancel keypress handling
writeBotLeft (s)	Write string to Bottom Left of LCD, curBotLeft is set to s
writeBotRight (s)	Write string to Bottom Right of LCD, curBotRight is set to s
setBitsBotAnnuns (d)	Sets the annunciator bits for Bottom Annunciators
clrBitsBotAnnuns (d)	Clears the annunciator bits for Bottom Annunciators
rotWAIT (dir)	Rotate the WAIT annunciator
setBitsTopAnnuns (d)	Sets the annunciator bits for Top Annunciators
clrBitsTopAnnuns (d)	Clears the annunciator bits for Top Annunciators
restoreLcd ()	Called to restore the LCD to its default state
setAnalogType (typ)	Set the analog output type
setAnalogVal (val)	Sets the analogue output to minimum 0.0 through to maximum 1.0
setAnalogPC (val)	Sets the analogue output to minimum 0% through to maximum 100%
setAnalogVolt (val)	Sets the analogue output to minimum 0.0V through to maximum 10.0V
setAnalogCur (val)	Sets the analogue output to minimum 4.0 through to maximum 20.0 mA
turnOn (IO)	Turns IO Output on
turnOff (IO)	Turns IO Output off
enableOutput (IO)	Sets IO Output under LUA control
releaseOutput (IO)	Sets IO Output under instrument control
setpParam (setp, reg, v)	Private function
setpRegAddress (setp, reg)	returns actual register address for a particular setpoint parameter
setpTarget (setp, target)	Set Target for setpoint
setpIO (setp, IO)	Set which Output the setpoint controls
setpType (setp, v)	Set the TYPE of the setpoint controls
setpLogic (setp, v)	Set the Logic for the setpoint controls
setpAlarm (setp, v)	Set the Alarm for the setpoint
setpName (setp, v)	Set the Name of the setpoint
setpSource (setp, v)	Set the data source of the setpoint controls
setpHys (setp, v)	Set the Hysteresis for of the setpoint controls
setNumSetp (n)	Set the number of Setpoints

setBuzzLen (len)	Called to set the length of the buzzer sound
buzz (times)	Called to trigger instrument buzzer
getKey (keyGroup)	Called to get a key from specified key group
edit (prompt, def, typ)	Called to prompt operator to enter a value
sendRegCallback (data, err)	Private function
sendRegWait (cmd, reg, data, t)	Called to send command and wait for response
readRegWait (reg)	Called to read register contents
editRegister (reg)	Called to edit value of specified register
delayCallback ()	Private function
delay (t)	Called to delay for t msec while keeping event handlers running
askOKCallback (key, state)	Private function
askOK (prompt, q)	Prompts operator and waits for OK or CANCEL key press
selectOption (prompt, options, def, loop)	Prompts operator to select from a list of options using arrow keys and KEY_OK
printCustomTransmit (tokenStr, comPort)	Send custom print token string to instrument comms port
reqCustomTransmit (tokenStr)	Called to request response based on custom transmit token string
sendDateFormat (fmt)	sets the instrument date format
RTCread (d)	Read Real Time Clock data from instrument into local RTC table
RTCtick ()	Called every second to update local RTC
RTCtostring ()	Returns formated date/time string Private function
RTCdateFormat (first, second, third)	Sets the order of the date string.byte
zero ()	<>
tare ()	<>
presetTare ()	<>
gross ()	<>
net ()	<>
grossNetToggle ()	<>

Fields

fullscale	Called to collect key system data from instrument to configure the local L401 library
REG_DISP_LAYOUT	LCD Services LCD display registers
REG_ANALOGUE_DATA	Routines for Analogue Output control
REG_IO_STATUS	Routines for Setpoint Output control
getKeyPressed	Routines for Dialog control
REG_TIMECUR	RTC Routines Time and Date
REG_ADC_ZERO	Commands TODO: Finalise these commands to return proper error messages etc

Functions

connect (sock, system)

Called to connect the L401 library to a socket and a system

Parameters:

- *sock*: TCP socket to connect
- *system*: System that manages L401 (nil if stand-alone library)

getRegDP (reg)

Called to read a register value and return value and dp position Used to work out the dp position of a register value so subsequent reads can use the hexadecimal format and convert locally using toFloat

Parameters:

- *reg*: register to read

Returns:

register value number and dp position

saveSetting ()

Called to save any changed settings and re-initialise instrument

toPrimary (v)

Called to convert a floating point value to a decimal integer based on then primary instrument weighing settings

Parameters:

- *v*: is value to convert

Returns:

floating point value suitable for a WRFINALDEC

---

toFloat (data, dp)

called to convert hexadecimal return string to a floating point number

Parameters:

- *data*: returned from _CMD_RDFINALHEX or from stream
- *dp*: decimal position

Returns:

floating point number

---

streamCallback (data, err)

Divide the data stream up and run the relevant callbacks

Parameters:

- *data*: Data received from register
- *err*: Potential error message

---

addStream (streamReg, callback, onChange)

Add a stream to the device (must be connected) Takes parameter 'change' (default) to run callback only if data received changed, 'always' otherwise

Parameters:

- *streamReg*: Register to stream from (_M.REG_*)
- *callback*: Function to bind to streaming register
- *onChange*: Change parameter return streamReg indentity

---

removeStream (streamReg)

Remove a stream from the device

Parameters:

- *streamReg*: Register to be removed(_M.REG_*)

---

streamCleanup ()

Called to cleanup any unused streaming

---

setStreamFreq (freq)

Set the frequency used for streaming

Parameters:

- *freq*: Frequency of streaming (_M.STM_FREQ_*)

---

statusCallback (data, err)

Called when status changes are streamed

Parameters:

- *data*: Data on status streamed
- *err*: Potential error message

---

setStatusCallback (stat, callback)

Set the callback function for a status bit

Parameters:

- *stat*: given in _M.statBinds
- *callback*: Function to run when there is an event on change in status

---

setupStatus ()

Setup status monitoring via a stream

---

setRTCStatus (s)

Control the use of RTC status bit

Parameters:

- *s*: true to enable RTC change monitoring, false to disable

---

setRDGStatus (num)

Control the use of reading count status bit. This is useful if weight readings are not collected via an on change stream register directly

Parameters:

- *num*: Sets amount of readings to trigger a reading count status change

---

setIOStatus ()

private function

---

enableIOStatus (IO)

sets IO status IO bit to recognise this IO

Parameters:

- *IO*: is output 1..32

---

releaseIOStatus (IO)

sets IO status IO bit to ignore this IO

Parameters:

- *IO*: is output 1..32

---

endStatus ()

Cancel status handling

---

keyCallback (data, err)

Called when keys are streamed, send the keys to each group it is bound to in order of priority, until one of them returns true. key states are 'short','long','up' Note: keybind tables should be sorted by priority

Parameters:

- *data*: Data on key streamed
- *err*: Potential error message

---

setKeyCallback (key, callback)

Set the callback function for an existing key

Parameters:

- *key*: A key given in _M.keyBinds
- *callback*: Function to run when there is an event on the keygroup

---

setKeyGroupCallback (keyGroup, callback)

Set the callback function for an existing key group

Parameters:

- *keyGroup*: A keygroup given in _M.keyGroup.*
- *callback*: Function to run when there is an event on the keygroup

---

setupKeys ()

Setup key handling stream

---

endKeys (flush)

Cancel keypress handling

Parameters:

- *flush*: Flush the current keypresses that have not yet been handled

---

writeBotLeft (s)

Write string to Bottom Left of LCD, curBotLeft is set to s

Parameters:

- *s*: string to display

---

writeBotRight (s)

Write string to Bottom Right of LCD, curBotRight is set to s

Parameters:

- *s*: string to display

---

setBitsBotAnnuns (d)

Sets the annunciator bits for Bottom Annunciators

Parameters:

- *d*: holds bit locations

clrBitsBotAnnuns (d)

Clears the annunciator bits for Bottom Annunciators

Parameters:

- `d` holds bit locations

rotWAIT (dir)

Rotate the WAIT annunciator

Parameters:

- `dir`: 1 clockwise, -1 anticlockwise 0 no change

setBitsTopAnnuns (d)

Sets the annunciator bits for Top Annunciators

Parameters:

- `d` holds bit locations

clrBitsTopAnnuns (d)

Clears the annunciator bits for Top Annunciators

Parameters:

- `d` holds bit locations

restoreLcd ()

Called to restore the LCD to its default state

setAnalogType (typ)

Set the analog output type

Parameters:

- `typ`: Type for output (.CUR or .VOLT)

setAnalogVal (val)

Sets the analogue output to minimum 0.0 through to maximum 1.0

Parameters:

- `val`: value 0.0 to 1.0

setAnalogPC (val)

Sets the analogue output to minimum 0% through to maximum 100%

Parameters:

- `val`: value 0 to 100 %

setAnalogVolt (val)

Sets the analogue output to minimum 0.0V through to maximum 10.0V

Parameters:

- `val`: value 0.0 to 10.0

setAnalogCur (val)

Sets the analogue output to minimum 4.0 through to maximum 20.0 mA

Parameters:

- `val`: value 4.0 to 20.0

turnOn (IO)

Turns IO Output on

Parameters:

- `IO` is output 1..32

turnOff (IO)

Turns IO Output off

Parameters:

- `IO` is output 1..32

enableOutput (IO)

Sets IO Output under LUA control

Parameters:

- *`IO`*: is input 1..32

releaseOutput (IO)

Sets IO Output under instrument control

Parameters:

- *`IO`*: is output 1..32

setpParam (setp, reg, v)

Private function

Parameters:

- *`setp`*:
- *`reg`*:
- *`v`*:

setpRegAddress (setp, reg)

returns actual register address for a particular setpoint parameter

Parameters:

- *`setp`*: is setpoint 1..16
- *`reg`*: is REG_SETP_*

Returns:

address of this registet for setpoint setp

setpTarget (setp, target)

Set Target for setpoint

Parameters:

- *`setp`*: Setpoint 1..16
- *`target`*: Target value

setpIO (setp, IO)

Set which Output the setpoint controls

Parameters:

- *`setp`*: is setpount 1..16
- *`IO`*: is output 1..32, 0 for none

setpType (setp, v)

Set the TYPE of the setpoint controls

Parameters:

- *`setp`*: is setpount 1..16
- *`v`*: is setpoint type

setpLogic (setp, v)

Set the Logic for the setpoint controls

Parameters:

- *`setp`*: is setpount 1..16
- *`v`*: is setpoint logic type

setpAlarm (setp, v)

Set the Alarm for the setpoint

Parameters:

- *`setp`*: is setpount 1..16
- *`v`*: is alarm type

setpName (setp, v)

Set the Name of the setpoint

Parameters:

- *`setp`*: is setpount 1..16

- *v*: is setpoint name

setpSource (setp, v)

Set the data source of the setpoint controls

Parameters:

- *setp*: is setpount 1..16
- *v*: is setpoint source type

setpHys (setp, v)

Set the Hysteresis for of the setpoint controls

Parameters:

- *setp*: is setpount 1..16
- *v*: is setpoint hysteresis

setNumSetp (n)

Set the number of Setpoints

Parameters:

- *n*: is the number of setpoints 0..16

setBuzzLen (len)

Called to set the length of the buzzer sound

Parameters:

- *len*: - length of buzzer sound (BUZZ_SHORT, BUZZ_MEDIUM, BUZZ_LONG)

buzz (times)

Called to trigger instrument buzzer

Parameters:

- *times*: - number of times to buzz, 1..4

getKey (keyGroup)

Called to get a key from specified key group

Parameters:

- *keyGroup*: keyGroup.all is default group

Returns:

key (KEY_), state ('short','long','up')

edit (prompt, def, typ)

Called to prompt operator to enter a value

Parameters:

- *prompt*: string displayed on bottom right LCD
- *def*: default value
- *typ*: type of value to enter ('integer','number','string'

Returns:

value and true if ok pressed at end

sendRegCallback (data, err)

Private function

Parameters:

- *data*:
- *err*:

sendRegWait (cmd, reg, data, t)

Called to send command and wait for response

Parameters:

- *cmd*: CMD_ command
- *reg*: REG_ register
- *data*: to send
- *t*: timeout in msec

Returns:

1. data received from instrument, nil if error

2. err error string if error received, nil otherwise

readRegWait (reg)

Called to read register contents

Parameters:

- *reg*: REG_ register

Returns:

1. data received from instrument, nil if error
2. err error string if error received, nil otherwise

editRegister (reg)

Called to edit value of specified register

Parameters:

- *reg*: is register to edit

delayCallback ()

Private function

delay (t)

Called to delay for t msec while keeping event handlers running

Parameters:

- *t*: delay time in msec

askOKCallback (key, state)

Private function

Parameters:

- *key*:
- *state*:

askOK (prompt, q)

Prompts operator and waits for OK or CANCEL key press

Parameters:

- *prompt*: string to put on bottom right LCD
- *q*: string to put on bottom left LCD

Returns:

either KEY_OK or KEY_CANCEL

selectOption (prompt, options, def, loop)

Prompts operator to select from a list of options using arrow keys and KEY_OK

Parameters:

- *prompt*: string to put on bottom right LCD
- *options*: table of option strings
- *def*: default selection string.byte
- *loop*: If true, top option loops to the bottom option and vice versa

Returns:

selected string (default selection if KEY_CANCEL pressed)

printCustomTransmit (tokenStr, comPort)

Send custom print token string to instrument comms port

Parameters:

- *tokenStr*: string containing custom print tokens
- *comPort*: - port to use PRINT_SER1A (default) .. PRINT_SER2B

reqCustomTransmit (tokenStr)

Called to request response based on custom transmit token string

Parameters:

- *tokenStr*: custom token string

sendDateFormat (fmt)

sets the instrument date format

Parameters:

- *fmt*: TM_MMDDYYYY or TM_DDMMYYYY

RTCread (d)

Read Real Time Clock data from instrument into local RTC table

Parameters:

- *d* 'date' or 'time' to read these fields only, or 'all' for both

RTCtick ()

Called every second to update local RTC

RTCtostring ()

Returns formated date/time string Private function

RTCdateFormat (first, second, third)

Sets the order of the date string.byte

Parameters:

- *first*: = 'day', 'month' or 'year'
- *second*: = 'day', 'monht','year'
- *third*: = 'day','month','year'

zero ()

<>

tare ()

<>

presetTare ()

<>

gross ()

<>

net ()

<>

grossNetToggle ()

<>

Fields

fullscale

Called to collect key system data from instrument to configure the local L401 library

REG_DISP_LAYOUT

LCD Services LCD display registers

REG_ANALOGUE_DATA

Routines for Analogue Output control

REG_IO_STATUS

Routines for Setpoint Output control

getKeyPressed

Routines for Dialog control

REG_TIMECUR

RTC Routines Time and Date

REG_ADC_ZERO

Commands TODO: Finalise these commands to return proper error messages etc

*generated by **LDoc 1.2***

Module `rinLibrary.ldocUsage`

Documentation of the ldoc features

Section Comments

tableName	Table Definition.
myFunction (p1, p2)	General Function Description goes here.
anotherFunction (p1, p2)	Another Function Description goes here.
localFunction	Private Method You need -a flag or 'all=true' to see these

Section Comments

Description of the section goes here

tableName

Table Definition.

Fields:

- *FIELD1*: Field 1 Description
- *FIELD2*: Field 2 DescriptionREG_SOFTVER Software Version eg "V1.00"
- *FIELD3*: Field 3 DescriptionREG_SERIALNO Serial Number

myFunction (p1, p2)

General Function Description goes here.

Parameters:

- *p1*: Parameter 1 description
- *p2*: Parameter 2 description

Usage:

```
a = myFunction(1,2)  -- comment in example code
```

Returns:

return description

see also:

anotherFunction

anotherFunction (p1, p2)

Another Function Description goes here. You can add multiple lines of description and link to standard Lua functions also.

Parameters:

- *p1*: Parameter 1 description
- *p2*: Parameter 2 description

Usage:

```
b = anotherFunction(1,2)  -- comment in example code
```

Returns:

returns description

see also:

string.sub

localFunction

Private Method You need -a flag or 'all=true' to see these

Parameters:

- *p1*:
- *p2*:

generated by ***LDoc 1.2***

Module `rinLibrary.rinCSV`

Functions for working with .CSV files and creating multi-table databases

CSV Utilities

escapeCSV (s)	Adds '"' around s if it contains ',' or '"' and replaces '"' with '""'
toCSV (t)	Converts a table (1d array) to a CSV string with fields escaped if required
padCSV (t, w)	Converts a table (1d array) to a CSV string with fields escaped if required
fromCSV (s)	Takes an escaped CSV string and returns a line (1d array)
equalCSV (labels, check)	Checks labels to ensure database table is the same structure Tolerant of additional whitespace in labels and ignores case

CSV Functions

saveCSV (t)	Save table t to a .CSV file
loadCSV (t)	Reads a .CSV file and returns a table with the loaded contents If no CSV file found or contents different then file created with structure in t
logLineCSV (t, line)	Adds line of data to a CSV file but does not update local data in table
addLineCSV (t, line)	Adds line of data to a table
dupLineCSV (line)	Makes a duplicate copy of a line of data
remLineCSV (t, row)	Removes line of data in a table (does not save .CSV file)
getLineCSV (t, val, col)	Removes line of data in a table (does not save .CSV file)
replaceLineCSV (t, row, line)	Replaces a line of data in the table (does not save the .CSV file)
labelCol (t, label)	Returns the column number of a particular label
tostringCSV (t, w)	Converts contents of the CSV table into a print friendly string

Database Utilities

addTableDB (db, name, t)	Adds a database table to the database, updates contents with t if already present
loadDB (db)	Restores database contents from CSV files Only loads in database tables already registered with database
addLineDB (db, name, l)	Adds line of data to a table in the database
remLineDB (db, name, line)	Removes last line of data in a database table
saveDB (db)	Save database to multiple CSV files
tostringDB (db, w)	Converts contents of the database into a print friendly string

CSV Utilities

Functions to convert data to and from .CSV format

escapeCSV (s)

Adds '"' around s if it contains ',' or '"' and replaces '"' with '""'

Parameters:

- *s*: string to escape

Returns:

escaped string

toCSV (t)

Converts a table (1d array) to a CSV string with fields escaped if required

Parameters:

- *t*: table to convert

Returns:

escaped CSV string

padCSV (t, w)

Converts a table (1d array) to a CSV string with fields escaped if required

Parameters:

- *t*: table to convert
- *w*: is width of each cell

Returns:

escaped CSV string padded to w characters in each cell

fromCSV (s)

Takes an escaped CSV string and returns a line (1d array)

Parameters:

- *s*: CSV string

Returns:

table (1d array)

equalCSV (labels, check)

Checks labels to ensure database table is the same structure Tolerant of additional whitespace in labels and ignores case

Parameters:

- `labels`: 1d array of labels from a database table
- `check`: 1d array of labels to check

Returns:

true if labels and check are the same, false otherwise

CSV Functions

Functions to manage CSV files directly

saveCSV (t)

Save table t to a .CSV file

Parameters:

- `t`: database table to save. table is in the format: fname name of .csv file associated with table - used to save/restore table contents labels{} 1d array of column labels data{{}} 2d array of data

Returns:

table in same format:

loadCSV (t)

Reads a .CSV file and returns a table with the loaded contents If no CSV file found or contents different then file created with structure in t

Parameters:

- `t`: is table with structure of expected CSV included

Returns:

table in same format: fname name of .csv file associated with table - used to save/restore table contents labels{} 1d array of column labels data{{}} 2d array of data

logLineCSV (t, line)

Adds line of data to a CSV file but does not update local data in table

Parameters:

- `t`: is table describing CSV data
- `line`: is a row of data (1d array) to save

addLineCSV (t, line)

Adds line of data to a table

Parameters:

- `t`: is table holding CSV data
- `line`: of data (1d array) to add to the table

Returns:

row location of line new line in table

dupLineCSV (line)

Makes a duplicate copy of a line of data

Parameters:

- `line`: is the line of data (1-d array)

Returns:

duplicate copy of line

remLineCSV (t, row)

Removes line of data in a table (does not save .CSV file)

Parameters:

- `t`: is table holding CSV data
- `row`: is row number of table data 1..n to remove. removes last line of data if row is nil

getLineCSV (t, val, col)

Removes line of data in a table (does not save .CSV file)

Parameters:

- *t*: is table holding CSV data
- *val*: is value of the cell to find
- *col*: is the column of data to match (default is col 1)

Returns:

1. row that val found in or nil if not found
2. line of data found at that row with matching val data in column col

---

replaceLineCSV (t, row, line)

Replaces a line of data in the table (does not save the .CSV file)

Parameters:

- *t*: is table holding CSV data
- *row*: is the row number of the line of data
- *line*: is the line of data

---

labelCol (t, label)

Returns the column number of a particular label

Parameters:

- *t*: is table holding CSV data
- *label*: is name of column to find (not case sensitive)

Returns:

column number of the label or nil if not found

---

tostringCSV (t, w)

Converts contents of the CSV table into a print friendly string

Parameters:

- *t*: table to convert
- *w*: width to pad each cell to

Database Utilities

Functions to manage multiple tables in a database

---

addTableDB (db, name, t)

Adds a database table to the database, updates contents with t if already present

Parameters:

- *db*: is the database table to populate
- *name*: is the name of table
- *t*: is the csv table to add database table is in the format fname name of .csv file associated with table - used to save/restore table contents labels{} 1d array of column labels data{{}} 2d array of data

---

loadDB (db)

Restores database contents from CSV files Only loads in database tables already registered with database

Parameters:

- *db*: database table to populate

---

addLineDB (db, name, l)

Adds line of data to a table in the database

Parameters:

- *db*: database table
- *name*: name of table in database to use
- *l*: line (1d array) of data to save

---

remLineDB (db, name, line)

Removes last line of data in a database table

Parameters:

- *db*: database table
- *name*: name of table to use
- *line*: is row number of table data 1..n to remove. removes last line if line is nil

---

saveDB (db)

Save database to multiple CSV files

Parameters:

- *db*: database table

---

tostringDB (db, w)

Converts contents of the database into a print friendly string

Parameters:

- *db*: database table
- *w*: width of each cell

*generated by **LDoc 1.2***

Module `rinLibrary.rinDebug`

Offer functions for converting variables into strings for debugging

Functions

setLevel (level)	Set Debug level
restoreLevel ()	Restores Debug level to previous setting
configureDebug (config, ip)	Configures the level for debugging
getDebugConfig ()	returns debug configuration
tableString (t)	Converts table t into a string
varString (arg)	Converts arg into a string
print (prompt, ...)	Prints variable contents to debugger at current debug level with optional prompt
debug (prompt, ...)	Prints variable contents to debugger at DEBUG level with optional prompt
info (prompt, ...)	Prints variable contents to debugger at INFO level with optional prompt
warn (prompt, ...)	Prints variable contents to debugger at WARN level with optional prompt
error (prompt, ...)	Prints variable contents to debugger at ERROR level with optional prompt
fatal (prompt, ...)	Prints variable contents to debugger at FATAL level with optional prompt
printVar (prompt, v, level)	Prints variable v contents to stdio with optional prompt included for backward compatibility - replaced by print

Functions

setLevel (level)

Set Debug level

Parameters:

- *`level`*: enumerated level constant or matching string. If no match level set to INFO

restoreLevel ()

Restores Debug level to previous setting

configureDebug (config, ip)

Configures the level for debugging

Parameters:

- *`config`*: is table of settings
- *`ip`*: optional tag printed with each message (usually IP address of the instrument)

Usage:

```
dbg.configureDebug({level = 'DEBUG',timestamp = true, logger = 'console'},'testDebug')
```

getDebugConfig ()

returns debug configuration

Returns:

1. level lualogging level
2. timestamp: true if timestamp logging is to included
3. ip

tableString (t)

Converts table t into a string

Parameters:

- *`t`*: is a table

varString (arg)

Converts arg into a string

Parameters:

- *`arg`*: is any variable

print (prompt, ...)

Prints variable contents to debugger at current debug level with optional prompt

Parameters:

- *`prompt`*: is an optional prompt printed before the arguments
- `...`: arguments to be printed

debug (prompt, ...)

Prints variable contents to debugger at DEBUG level with optional prompt

Parameters:

- `prompt`: is an optional prompt printed before the arguments
- `...`: arguments to be printed

info (prompt, ...)

Prints variable contents to debugger at INFO level with optional prompt

Parameters:

- `prompt`: is an optional prompt printed before the arguments
- `...`: arguments to be printed

warn (prompt, ...)

Prints variable contents to debugger at WARN level with optional prompt

Parameters:

- `prompt`: is an optional prompt printed before the arguments
- `...`: arguments to be printed

error (prompt, ...)

Prints variable contents to debugger at ERROR level with optional prompt

Parameters:

- `prompt`: is an optional prompt printed before the arguments
- `...`: arguments to be printed

fatal (prompt, ...)

Prints variable contents to debugger at FATAL level with optional prompt

Parameters:

- `prompt`: is an optional prompt printed before the arguments
- `...`: arguments to be printed

printVar (prompt, v, level)

Prints variable v contents to stdio with optional prompt included for backward compatibility - replaced by print

Parameters:

- `prompt`: is an optional prompt
- `v`: is a variable whose contents are to be printed
- `level`: is the debug level for the message INFO by default

generated by ***LDoc 1.2***

ldoc

Module `rinLibrary.rinINI`

Services for saving and restoring settings in a table to .INI config file

Functions

saveINI (fname, t)	Saves table t as a .INI name (fname)
loadINI (fname, def)	populates table t with contents of INI file fname if fname is not found a new file is created with table def contents
stringINI (t)	returns table t contents in an INI format string

Functions

saveINI (fname, t)

Saves table t as a .INI name (fname)

Parameters:

- `fname`: name of file
- `t`: is table of settings

Returns:

t if successful, nil otherwise

loadINI (fname, def)

populates table t with contents of INI file fname if fname is not found a new file is created with table def contents

Parameters:

- `fname`: name of file
- `def`: default table of settings

Returns:

table t or nil if file invalid

stringINI (t)

returns table t contents in an INI format string

Parameters:

- `t`: is table of settings

Returns:

A string in INI format

*generated by **LDoc 1.2***

Module `rinLibrary.rinRIS`

Creates a connection to the M4223

Functions

load (filename, ip, port)	Read a RIS file and send valid commands to the device

Functions

load (filename, ip, port)

Read a RIS file and send valid commands to the device

Parameters:

- *`filename`*: Name of the RIS file
- *`ip`*: IP Address to transmit file contents to
- *`port`*: Port to transmit file contents to

generated by ***LDoc 1.2***

Module `rinLibrary.rinVT100`

Functions for working with VT100 terminal interface

Functions

set (s)	Send an escape sequence to the terminal
csrHome ()	Move Cursor to top left of screen
csrXY (x, y)	Move Cursor to position x,y on screen
csrUp (r)	Move Cursor up relative to current position
csrDown (r)	Move Cursor down relative to current position
csrLeft (c)	Move Cursor left relative to current position
csrRight (c)	Move Cursor right relative to current position
csrSave ()	Save current cursor position
csrRestore ()	Restore cursor to saved position
clrScreen ()	Clear terminal screen
clrEol ()	Clear to end of line from cursor position
clrSol ()	Clear to start of line from cursor position
clrRow ()	Clear current row
clrAbove ()	Clear all lines above current cursor position
clrBelow ()	Clear all lines below current cursor position
clrAttr ()	Clear terminal attributes to default settings
scrollAll ()	set terminal to scroll entire screen
scrollSet (r1, r2)	set terminal to scroll only a section of the screen
setAttr (fg, bg)	set terminal attributes
restoreAttr ()	restores last terminal attributes

Functions

set (s)

Send an escape sequence to the terminal

Parameters:

- *s*: string containing escape sequence to send

Usage:

```
VT100 = require rinLibrary.rinVT100

 VT100.set(VT100.csrHome()..VT100.setAttr(VT100.FGRed,VT100.BGBlack)

 print('Hello'..VT100.csrLeft(10)..'There')
```

csrHome ()

Move Cursor to top left of screen

csrXY (x, y)

Move Cursor to position x,y on screen

Parameters:

- *x*: horizontal position from left, 0 (default) is start
- *y*: vertical position from top, 0 (default) is top of screen

csrUp (r)

Move Cursor up relative to current position

Parameters:

- *r*: is number of rows to move

csrDown (r)

Move Cursor down relative to current position

Parameters:

- *r*: is number of rows to move

csrLeft (c)

Move Cursor left relative to current position

Parameters:

- *c*: is number of columns to move

csrRight (c)

Move Cursor right relative to current position

Parameters:

- `c`: is number of columns to move

csrSave ()

Save current cursor position

csrRestore ()

Restore cursor to saved position

clrScreen ()

Clear terminal screen

clrEol ()

Clear to end of line from cursor position

clrSol ()

Clear to start of line from cursor position

clrRow ()

Clear current row

clrAbove ()

Clear all lines above current cursor position

clrBelow ()

Clear all lines below current cursor position

clrAttr ()

Clear terminal attributes to default settings

scrollAll ()

set terminal to scroll entire screen

scrollSet (r1, r2)

set terminal to scroll only a section of the screen

Parameters:

- `r1`: row to start scrolling area
- `r2`: row to end scrolling area

setAttr (fg, bg)

set terminal attributes

Parameters:

- `fg`: foreground colours FG*
- `bg`: background colours BG*

restoreAttr ()

restores last terminal attributes

*generated by **LDoc 1.2***

Module `rinLibrary.rincon`

Creates a connection to the M4223

Functions

defaultErrHandler (addr, cmd, reg, data, s)	Default Error Handler, logs error to debug at WARN level with error string and received command takes arguments: Address, Command, Register, Data, Err String from message processing
setErrHandler (errHandler)	Set your own routine to handle errors reported from the instrument
removeErrHandler ()	Removes the error handler
socketACallback ()	Designed to be registered with rinSystem.
pushQ (msg)	Add 1 to the queue and put the message on the end of the queue
popQ ()	Remove the message from the front of the queue, and return the message
Qempty ()	Check if the queue is empty
sendQueueCallback ()	Designed to be registered with rinSystem.
disconnect ()	Disconnect from the R400
recMsg ()	Receive a rinCMD message from a socket linked to SERA.
CCITT (data)	Creates a CRC-CCITT (0xFFFF) of the given ASCII data
processMsg (msg, err)	Processes the message and feeds back the individual parts
sendRaw (raw)	Sends a raw message
sendMsg (msg, crc)	Sends a message with delimiters added optionally with CRC
send (addr, cmd, reg, data, reply, crc)	Sends a structured message built up from individual parameters as follows
preconfigureMsg (reg, cmd, reply, crc)	Return a function allowing for repeatable commands
bindRegister (reg, callback)	Set up a callback for when data on a specific register is received
unbindRegister (reg)	Unbind a register
socketBCallback ()	Designed to be registered with rinSystem.
setDelimiters (start, end1, end2)	Set delimiters for messages received from the socket linked to SERB E.g.
setSerBCallback (f)	Set delimiters for messages received from the socket linked to SERB E.g.

Tables

rinCMD	Instrument Commands.

Functions

defaultErrHandler (addr, cmd, reg, data, s)

Default Error Handler, logs error to debug at WARN level with error string and received command takes arguments: Address, Command, Register, Data, Err String from message processing

Parameters:

- *addr*:
- *cmd*:
- *reg*:
- *data*:
- *s*:

setErrHandler (errHandler)

Set your own routine to handle errors reported from the instrument

Parameters:

- *errHandler*: Function for handling errors, should take arguments: Address, Command, Register, Data, Err String.

removeErrHandler ()

Removes the error handler

Returns:

original registered handler

socketACallback ()

Designed to be registered with rinSystem. If a message error occurs, pass it to the error handler.

pushQ (msg)

Add 1 to the queue and put the message on the end of the queue

Parameters:

- *msg*: Message to add to the end of the queue

popQ ()

Remove the message from the front of the queue, and return the message

Returns:

Message removed from the queue

Qempty ()

Check if the queue is empty

Returns:

True if empty, false otherwise

sendQueueCallback ()

Designed to be registered with rinSystem. If a message error occurs, pass it to the error handler.

disconnect ()

Disconnect from the R400

recMsg ()

Receive a rinCMD message from a socket linked to SERA. Receives one byte at a time, and ends the message based on specified delimiters

Returns:

1. A string bounded by delimiters (nil if error)
2. An error message (nil if no error)

CCITT (data)

Creates a CRC-CCITT (0xFFFF) of the given ASCII data

Parameters:

- *data*: Data to be processed

Returns:

CRC-CCITT (0xFFFF) of message This code is relatively slow. Further speed gains will require a switch to C which should bump the speed a thousand fold.

processMsg (msg, err)

Processes the message and feeds back the individual parts

Parameters:

- *msg*: Message to be processed
- *err*: Error in receive (nil if none)

Returns:

1. address (0x00 to 0x1F)
2. command (CMD_*)
3. register (REG_*)
4. data
5. error

sendRaw (raw)

Sends a raw message

Parameters:

- *raw*: string to send

sendMsg (msg, crc)

Sends a message with delimiters added optionally with CRC

Parameters:

- *msg*: message string to send
- *crc*: if crc = 'crc' then SOH msg CRC EOT sent, msg CRLF otherwise (default)

send (addr, cmd, reg, data, reply, crc)

Sends a structured message built up from individual parameters as follows

Parameters:

- *addr*: Indicator address (0x00 to 0x1F)
- *cmd*: Command (CMD_*)

- *reg*: Register (REG_*)
- *data*: Data to be sent
- *reply*: - 'reply' (default) if reply required, sent with ADDR_NOREPLY otherwise
- *crc*: - 'crc' if message sent with crc, false (default) otherwise

preconfigureMsg (reg, cmd, reply, crc)

Return a function allowing for repeatable commands

Parameters:

- *reg*: register (REG_*)
- *cmd*: command (CMD_*)
- *reply*: - 'reply' (default) if reply required, sent with ADDR_NOREPLY otherwise
- *crc*: - 'crc' if message sent with crc, false (default) otherwise

Returns:

preconfigured function

bindRegister (reg, callback)

Set up a callback for when data on a specific register is received

Parameters:

- *reg*: Register to give callback, (REG_*), 0 is used to match anything received that has no other binding
- *callback*: Function to be run when data is received

unbindRegister (reg)

Unbind a register

Parameters:

- *reg*: Register to remove callback, (REG_*)

socketBCallback ()

Designed to be registered with rinSystem.

setDelimiters (start, end1, end2)

Set delimiters for messages received from the socket linked to SERB E.g. for \r\n delimiting use parameters: nil, '\r', '\n'

Parameters:

- *start*: start character, nil if not used
- *end1*: first end character, nil if not used
- *end2*: second end character, nil if not used

setSerBCallback (f)

Set delimiters for messages received from the socket linked to SERB E.g. for \r\n delimiting use parameters: nil, '\r', '\n'

Parameters:

- *f*: callback function that takes a message string as an argument

Tables

rinCMD

Instrument Commands.

Fields:

- *CMD_RDTYPE*: Read Register Type
- *CMD_RDRANGEMIN*: Read data range minimum
- *CMD_RDRANGEMAX*: Read data range maximum
- *CMD_RDRAW*: Read Raw data
- *CMD_RDLIT*: Read literal data
- *CMD_WRRAW*: Write Raw data
- *CMD_RDDEFAULT*: Read default setting
- *CMD_RDNAME*: Read Name
- *CMD_RDITEM*: Read Item from item list
- *CMD_RDPERMISSION*: Read register permissions
- *CMD_RDFINALHEX*: Read data in hexadecimal format
- *CMD_RDFINALDEC*: Read data in decimal format
- *CMD_WRFINALHEX*: Write data in hexadecimal format
- *CMD_WRFINALDEC*: Write data in decimal format
- *CMD_EX*: Execute with data as execute parameter

*generated by **LDoc 1.2***

Module `rinSystem`

Framework for interfacing with the L401 for advanced applications

Functions

handleEvents ()	Main function for handling events Issues a callback to any connection or timer that has an event on it.

Functions

handleEvents ()

Main function for handling events Issues a callback to any connection or timer that has an event on it.

*generated by **LDoc 1.2***

Module `rinSystem.rinSockets`

Offer functions for sockets that are compatible with the app framework

Functions

addSocket (sock, callback)	Add a socket to the socket list
socketCallback (sock)	Run the callback function associated with a socket
removeSocket (sock)	Remove a socket from the socket list

Functions

addSocket (sock, callback)

Add a socket to the socket list

Parameters:

- *sock*: Socket to add to the list
- *callback*: Callback for the socket

socketCallback (sock)

Run the callback function associated with a socket

Parameters:

- *sock*: Socket whose callback function should be executed.

removeSocket (sock)

Remove a socket from the socket list

Parameters:

- *sock*: Socket to remove from the list

*generated by **LDoc 1.2***

Module `rinSystem.rinTimers`

Offer functions for timers that are compatible with the app framework

Functions

addTimer (time, delay, callback, ...)	Add a timer to the timer list
removeTimer (key)	Remove a timer from the timer list
getSoonest ()	Get the time until the next timer expires
runKey (key)	Attempt to run a timer designated by the key.

Functions

addTimer (time, delay, callback, ...)

Add a timer to the timer list

Parameters:

- *`time`*: Time until the timer will go off (milliseconds)
- *`delay`*: Initial delay for timer
- *`callback`*: Function to run when timer is complete
- `...`: Function variables

Returns:

Timer key

removeTimer (key)

Remove a timer from the timer list

Parameters:

- *`key`*: Key for a timer

Returns:

key if success, nil and error if otherwise

getSoonest ()

Get the time until the next timer expires

Returns:

Timer key

runKey (key)

Attempt to run a timer designated by the key.

Parameters:

- *`key`*: Timer key given by add timer

Returns:

true is timer callback ran, false if it did not

*generated by **LDoc 1.2***