RP Metrix LASOM_LV.DLL

LabView external DLL interface for the RP Metrix LASOM1 (Laboratory Apparatus Scalable Olfactometer Module, Version 1).

The external entry points are described below.

All functions return 0 if success, negative value if otherwise.
For many functions, the following parameters have common usages:

Parameter **uSlaveIndex** selects one LASOM module in a multi-slave system. Range is 1 to 15.
Use 1 for a single module system.

Parameter **uMfcIndex** selects one Mass Flow Controller on the selected slave module. Range is 1 to 2.

Parameter **bActivate** is non-zero to activate a valve or digital output.

Parameter **puActive** points to a 32-bit unsigned integer used to receive the current state of a digital input.

Parameter **fDurationSec** causes an active digital output to be deactivated after the specified delay. If 0, the parameter is ignored.

Parameter type **LABVIEW_BOOL** is equivalent to **LABVIEW_U32.**

__declspec (dllexport)
int LASOM_LV_Open(LABVIEW_U32 uDeviceID, LABVIEW_U32 uTestMode);

All sessions must call this function once, before any other functions are called.
This opens a connection to the LASOM USB device.
All valves are deactivated as a result of this call.

Parameter **uDeviceID** = 0, to open the first RP Metrix device found.
Each device contains a factory assigned 32-bit ID (use LoadBoard GetID to read the value.)
Supply the factory assigned value to open that device specifically. This is only necessary if multiple RP Metrix USB devices are attached to the same computer.

Parameter **uTestMode** = 0 for normal operation.
= 1 for verbose operation (not implemented)
= 2 for demo mode without hardware (not implemented)

__declspec (dllexport)
int LASOM_LV_GetID(LStrHandle hstrDeviceID);

Parameter **hstrDeviceID** is a LabView string handle. This string variable will be modified to contain the length and text of a string identifying the LASOM unit and firmware in use.
Example value: “RP Metrix, LASOM1, Jul 05 2007 09:00:50 ID:2004”.

__declspec (dllexport)
int LASOM_LV_GetID2(char * pszDeviceID, LABVIEW_U32 uBufSize);

The LASOM identification string is returned in the buffer indicated by **pszDeviceID**.
At most, **uBufSize** characters are written, including the null terminator.

__declspec (dllexport)
int LASOM_LV_GetLastError(LStrHandle hstrReason);

Parameter **hstrReason** is a LabView string handle. This string variable will be modified to contain the length and text of a string describing the most recent error. Example value: “No Device”.

__declspec (dllexport)
int LASOM_LV_GetLastError2(char * pszReason, LABVIEW_U32 uBufSize);

The most recent error reason string is returned in the buffer indicated by **pszReason**.
At most, **uBufSize** characters are written, including the null terminator.

__declspec (dllexport)
int LASOM_LV_ScanOneWireBus (LABVIEW_PU32 puDeviceCount);

Scan the 1-Wire bus and determine the number of LASOM cassettes configured on the expansion bus. This count includes the master and all slaves.

Parameter **puDeviceCount** points to a 32-bit unsigned integer to receive the device count.

__declspec (dllexport)
int LASOM_LV_GetOneWireDevice (LABVIEW_U32 uIndex
, LABVIEW_PU32 puDeviceCount
, char * pcDeviceBuf
, LABVIEW_U32 uBufSize
);

Using the results of **ScanOneWireBus()**, get the 1-Wire identification from one device on the bus.

Parameter **uIndex** selects the device for the request, in the range is 1..uDeviceCount.
Parameter **puDeviceCount** points to a 32-bit unsigned integer to receive the device count.
This value should match the result from **ScanOneWireBus()**.

Parameter **pcDeviceBuf** points to a buffer to receive 8 bytes of identification. The bytes will contain:
1 device family code
6 factory assigned ID bytes
1 1-Wire CRC byte computed from the previous data

Parameter **uBufSize** should indicate the byte size of the buffer pointed to by **pcDeviceBuf**.

__declspec (dllexport)
int LASOM_LV_SetBlankState (LABVIEW_U32 uSlaveIndex, LABVIEW_BOOL bVisible);

The valve indicator LEDs normally follow the activation of corresponding odor or gate valves.
LED blanking is an on/off function which can be used to hide all of these LEDs.
An independent mechanism using a DAC channel controls the brightness of these LEDs.

Parameter **bVisible** is non-zero to enable the valve indicator LEDs to be visible.

__declspec (dllexport)
int LASOM_LV_SetOdorValve(LABVIEW_U32 uSlaveIndex, LABVIEW_U32 uOdorValveIndex, LABVIEW_BOOL bActivate);

Parameter **uOdorValveIndex** selects an odor valve. Range is 1 to 11. Odor valve 11 is reserved for the 'dummy' valve, which is active if no other odor valve is active to maintain consistent flow within a LASOM module. Previously activated odor valves will be automatically deactivated each time a new odor valve is activated. Deactivating an active non-dummy odor valve will automatically activate the dummy valve.

Parameter **bActivate** is non-zero to activate a valve.

__declspec (dllexport)
int LASOM_LV_SetGateValve(LABVIEW_U32 uSlaveIndex, LABVIEW_U32 uGateValveIndex, LABVIEW_BOOL bActivate);

Parameter **uGateValveIndex** selects a gate valve. Range is 12 to 16. These valves are external to the LASOM module. Each may be operated independently.

Parameter **bActivate** is non-zero to activate a valve.

__declspec (dllexport)
int LASOM_LV_SetDigOutMask(LABVIEW_U32 uOutputMask);

Set the new output state of digital output pins on the master LASOM module.

Parameter **uOutputMask** activates digital outputs as described in "LASOM_DigMasks.txt". Set specific bits in this 32-bit parameter to activate the corresponding selected outputs.

__declspec (dllexport)
int LASOM_LV_GetDigInMask(LABVIEW_PU32 puInputMask);

Read the current input state of digital input pins on the master LASOM module.

Parameter **puInputMask** points to a 32-bit unsigned integer to receive the sampled state of all digital input pins as described in "LASOM_DigMasks.txt". AND the result with a 32-bit mask to select particular digital inputs to observe.

__declspec (dllexport)
int LASOM_LV_ArmGateValves(LABVIEW_U32 uSlaveIndex, LABVIEW_U32 uValveMask);

Prepare to activate selected gate valves on a future Strobe signal.

Parameter **uValveMask** selects the gate valves to activate. Set bits in the mask based on the selected gate valve index of each valve to activate. Each mask bit is computed as 2^(GateValveIndex-1), so that for example the mask bit for valve 16 is 0x8000. Set multiple bits to activate multiple valves.

__declspec (dllexport)
int LASOM_LV_SetStrobeMask(LABVIEW_U32 uStrobeMask);

Set the Strobe mask, which either generates a software Strobe signal, enables an external Strobe input, or both.

Parameter **uStrobeMask** enables Strobe inputs, outputs, and internal strobe generation as described in "LASOM_DigMasks.txt".

If gate valves have been armed and the software Strobe signal is activated, the valves activate immediately when this function is called.

If gate valves have been armed and the Strobe In BNC input is enabled with this function, the valves will not activate until a TTL high level is present at the Strobe In input.

__declspec (dllexport)
int LASOM_LV_SetMfcFlowRate(LABVIEW_U32 uSlaveIndex
, LABVIEW_U32 uMfcIndex
, LABVIEW_FLOAT fPercentOfCapacity
);

Set the MFC mass flow rate set point.

Parameter **fPercentOfCapacity** requests a flow as a percentage of total capacity. Range is 0.0 to 100.0.

__declspec (dllexport)
int LASOM_LV_GetMfcCapacity(LABVIEW_U32 uSlaveIndex
, LABVIEW_U32 uMfcIndex
, LABVIEW_PFLOAT pfCapacity
, LStrHandle hstrUnits
);

__declspec (dllexport)
int LASOM_LV_GetMfcCapacity2(LABVIEW_U32 uSlaveIndex
, LABVIEW_U32 uMfcIndex
, LABVIEW_PFLOAT pfCapacity
, char * pszUnits
, LABVIEW_U32 uBufSize
);

Inquire the MFC to determine the total flow rate capacity and the physical units in which the capacity is expressed.

Parameter **pfCapacity** points to a single precision float variable to receive the capacity value. Example values: 100.0 or 1000.0.

Parameter **hstrUnits** is a LabView string handle. This string variable will be modified to contain the length and text of the capacity physical unit string. Example value: "sccm".

The capacity physical unit string will be returned in the buffer indicated by **pszUnits**. At most **uBufSize** characters will be written, including the null terminator.

```
__declspec (dllexport)
int LASOM_LV_GetMfcFlowRateSetting(LABVIEW_U32 uSlaveIndex
                                   , LABVIEW_U32 uMfcIndex
                                   , LABVIEW_PFLOAT pfPercentOfCapacity
                                   );
```

Get the current mass flow rate setting as a percentage of total capacity. Expected range is 0.0 to 100.0.

Parameter **pfPercentOfCapacity** points to a single precision float variable to receive the current setting.

```
__declspec (dllexport)
int LASOM_LV_GetMfcFlowRateMeasure(LABVIEW_U32 uSlaveIndex
                                   , LABVIEW_U32 uMfcIndex
                                   , LABVIEW_PFLOAT pfPercentOfCapacity
                                   );
```

Get the current mass flow rate measurement as a percentage of total capacity.

Parameter **pfPercentOfCapacity** points to a single precision float variable to receive the current measurement. Expected range is 0.0 to 100.0.

__declspec (dllexport)
int LASOM_LV_SetCue(LABVIEW_U32 uIndex
, LABVIEW_BOOL bActivate
, LABVIEW_FLOAT fDurationSec
);
Set a Cue output state on Leaf Board 1 (via J51).
Parameter **uIndex** selects the output. Range 1..3 corresponds to leaf J44..J46.

__declspec (dllexport)
int LASOM_LV_SetDigOut(LABVIEW_U32 uIndex
, LABVIEW_BOOL bActivate
, LABVIEW_FLOAT fDurationSec
);
Set a DigOut output state on Leaf Board 2 (via J52).
Parameter **uIndex** selects the output. Range 1..3 corresponds to leaf J44..J46.

__declspec (dllexport)
int LASOM_LV_SetXLogicOut(LABVIEW_U32 uIndex
, LABVIEW_BOOL bActivate
, LABVIEW_FLOAT fDurationSec
);
Set an XLogic output state (via J53).
Parameter **uIndex** selects the output. Range 1..4 correspond to Xlogic 5..8.

__declspec (dllexport)
int LASOM_LV_GetBeam(LABVIEW_U32 uIndex
, LABVIEW_PU32 puActive
);
Get a Beam input state from Leaf Board 1 (via J51).
Parameter **uIndex** selects the input. Range 1..3 corresponds to leaf J41..J43.

__declspec (dllexport)
int LASOM_LV_GetDigIn(LABVIEW_U32 uIndex
, LABVIEW_PU32 puActive
);
Get a DigIn input state from Leaf Board 2 (via J52).
Parameter **uIndex** selects the input. Range 1..3 corresponds to leaf J41..J43.

__declspec (dllexport)
int LASOM_LV_GetXLogicIn(LABVIEW_U32 uIndex
, LABVIEW_PU32 puActive
);
Get an XLogic input state (via J53).
Parameter **uIndex** selects the input. Range 1..4 corresponds to Xlogic 1..4.

__declspec (dllexport)
int LASOM_LV_MacroEnable(void);

Enable automatic use of LASOM sequencer to perform operations with repeatable timing. Normally, operations are initiated individually from the host, with host dependent timing. The following operations are affected:

SetOdorValve
SetGateValve
SetGateValve2
SetCue
SetDigOut
SetXLogicOut

__declspec (dllexport)
int LASOM_LV_MacroDisable(void);

Disable automatic use of LASOM sequencer.

__declspec (dllexport)
int LASOM_LV_ParseSeqFile(LPCSTR pszFileName);

Read and parse a LASOM sequencer source file (LSQ file). The parameterized sequence will be stored in host memory, awaiting further action.

__declspec (dllexport)
int LASOM_LV_SetParamU32(LPCSTR pszName, LABVIEW_U32 u Value**);**

Define a sequencer parameter and assign the value. This function is used to override default parameter values stored within an LSQ file. All override assignments must occur BEFORE calling ParseSeqFile to have any effect. An error occurs if the parameter is already defined.

__declspec (dllexport)
int LASOM_LV_GetSeqParseResult(LStrHandle hstrResult);

Parameter **hstrResult** is a LabView string handle. This string variable will be modified to contain the length and text of a string describing the most recent parser result from **LASOM_LV_ParseSeqFile()** . Example value: “End of file, no errors”.

__declspec (dllexport)
int LASOM_LV_ GetSeqParseResult 2(char * pszResult, LABVIEW_U32 uBufSize);

The most recent parser result string is returned in the buffer indicated by **pszResult**.
At most, **uBufSize** characters are written, including the null terminator.

int LASOM_LV_ GetSeqNumParsedLines (LABVIEW_PU32 puValue);

Returns the number of lines parsed by **LASOM_LV_ParseSeqFile()** up to the end of the file if the parse is successful, or if not, the first line containing an error.

__declspec (dllexport)
int LASOM_LV_ClearSequence(void);

Delete any previously defined sequencer parameter definitions resulting from parsing an LSQ file or setting override parameter values.

__declspec (dllexport)
int LASOM_LV_CompileSequence(void);

Convert the parsed sequence to LASOM sequencer instructions. All parameter substitution and conditional and repeat generation takes place during this operation.

__declspec (dllexport)
int LASOM_LV_LoadAndRunSequencer(LABVIEW_BOOL bClearStates);

Load the compiled sequencer instructions into the LASOM and start execution.

Parameter **bClearStates** must be zero to preserve the sequencer states from a previous execution.

__declspec (dllexport)
int LASOM_LV_StopSequencer(void);

Stop the LASOM sequencer, and set the current instruction position to zero.

__declspec (dllexport)
int LASOM_LV_SetSequencerVar(LABVIEW_U32 uVarIndex, LABVIEW_U32 uNewValue);

Change the value of a sequencer variable.

Parameter **nVarIndex** is in the range 1..8.

__declspec (dllexport)
int LASOM_LV_GetSequencerVar(LABVIEW_U32 uVarIndex, LABVIEW_PU32 puValue);

Get the value of a sequencer variable.

Parameter **uVarIndex** is in the range 1..8.

__declspec (dllexport)
int LASOM_LV_GetSequencerTimer(LABVIEW_U32 uTimerIndex
, LABVIEW_PFLOAT pfSeconds);

Get the value of a sequencer timer.

Parameter **uTimerIndex** is in the range 1..8.

Sequencer Status Updates

The sequencer action "EmitStatus" causes a status update message to be emitted from the LASOM master to the host computer. LASOMX maintains a copy of the most recent status message. Status updates also occur when the sequencer is initialized and when it drops to the idle state.

The status information is maintained in the properties:

LABVIEW_U32	**SeqUpdateEnable**	**1 if Running, 0 if Idle**
LABVIEW_U32	**SeqUpdateStep**	**Current sequencer instruction index**
LABVIEW_U32	**SeqUpdateCount**	**Increments mod 256 on each update**
LABVIEW_U32	**SeqUpdateInState**	**16-bit digital input mask**
LABVIEW_U32	**SeqUpdateOutState**	**16-bit digital output mask**
LABVIEW_U32	**SeqUpdateVarState (LABVIEW_U32 uVarIndex)**	**$Vark, where k = nVarIndex = 1..8**

Gets can be issued for these property values, with the functions:

```
__declspec (dllexport)
int LASOM_LV_GetSeqUpdateEnable(LABVIEW_PU32 puValue);
__declspec (dllexport)
int LASOM_LV_GetSeqUpdateStep(LABVIEW_PU32 puValue);
__declspec (dllexport)
int LASOM_LV_GetSeqUpdateCount(LABVIEW_PU32 puValue);
__declspec (dllexport)
int LASOM_LV_GetSeqUpdateInState(LABVIEW_PU32 puValue);
__declspec (dllexport)
int LASOM_LV_GetSeqUpdateOutState(LABVIEW_PU32 puValue);
__declspec (dllexport)
int LASOM_LV_GetSeqUpdateVarState(LABVIEW_U32 uVarIndex, LABVIEW_PU32 puValue);
```

Parameter **uVarIndex** is in the range 1..8.

Use **GetSeqUpdateCount()** to check for the arrival of a new status message. When it differs from the previous call, a new message has arrived and the other status properties may have changed.

Sequencer Execution Timing

In addition to the use of timers to record event times, delays, and durations the sequencer maintains a record of the last execution time of every instruction. Although the mapping of a source code statement to the corresponding instruction offset is determined by the compiler, a user can access the time for any number of specific steps in a sequencer program which are prefixed with Label statements. The Label statement itself consumes no instruction space in the sequencer. The relative time in seconds from execution of the first sequencer instruction can be obtained by either of the following entry points:

```
__declspec (dllexport)
int LASOM_LV_GetSequencerLabelTime(LPCSTR pszLabel, LABVIEW_PFLOAT pfSeconds);
```

Parameter **pszLabel** is a string which should match a label in the source code, such as "@EndLoop1". Parameter **pfSeconds** is a pointer to a float variable in which the relative time in seconds is returned. The return value of **GetSequencerLabelTime** indicates the success of the operation, and if the label was recognized.

LASOM Analog Input and Output

LASOM connector J33 as described in "LASOM_Analog.txt" provides access to the onboard 4-channel ADC/DAC interface chip (Analog Devices ADT7517). This device supports analog control of the MFC modules, and adjustable dimming of the valve indicator LEDs. If those functions are not needed, the connector provides for other external uses.

DAC channels 1 to 4 are available on connector J33.
ADC channels 1 to 4 are available on connector J33.
ADC channel 5 measures the chip temperature in degrees C (not volts).
ADC channel 6 measures the chip power supply input, nominally 5 volts.

```
__declspec (dllexport)
int LASOM_LV_SetDac(LABVIEW_U32 uSlaveIndex
                    , LABVIEW_U32 uDacIndex
                    , LABVIEW_FLOAT fVoltage
                    );
```

Set the DAC output voltage.

Parameter **uDacIndex** range is 1 to 4.
Parameter **fVoltage** range is 0.0 to 5.0.

```
__declspec (dllexport)
int LASOM_LV_GetAdcMeasure(LABVIEW_U32 uSlaveIndex
                    , LABVIEW_U32 uAdcIndex
                    , LABVIEW_PFLOAT  pfVoltage
                    );
```

Get the current ADC measurement.

Parameter **uAdcIndex** range is 1 to 6.
Parameter **pfVoltage** points to a single precision float variable to receive the current measurement. Expected range is 0.0 to 2.5 volts for channels 1 to 4.