The Lafayette Photovoltaic Research & Development System has a fully functional C++ API that can be used by applications to completely control the LPRDS System. These API functions are listed and explained in this document. Supporting documentation is included in the appendices.
TABLE OF CONTENTS

- CONNECT(ADDRESS, PID) .............................................................................................................................. 2
- DISCONNECT() ............................................................................................................................................... 2
- GETADDRESS() ............................................................................................................................................... 2
- GETDEVICELIST() ............................................................................................................................................ 2
- GETSENSORLIST()........................................................................................................................................... 3
- GETSWITCHLIST() ........................................................................................................................................... 3
- GETDETAILS(DEVICEID) .................................................................................................................................. 4
- GETUNITS(SENSORID) .................................................................................................................................... 4
- GETVALUE(SENSORID) ................................................................................................................................... 5
- GETVALUE(DAQ,CHANNEL) ............................................................................................................................ 5
- GETLASTDATA(SENSORID) ............................................................................................................................. 5
- SETSWITCH(SWITCHID,LEVEL)........................................................................................................................ 6
- SETSWITCH(DAQ,PIN,LEVEL) .......................................................................................................................... 6
- ADDDEVICE(SWITCHID,DAQ,PIN,DETAILS) ..................................................................................................... 6
- ADDDEVICE(SENSORID,TYPE,UNITS,OFFSET,SCALE,DAQ,CHANNEL,MAX,MIN,DETAILS) ..................... 7
- REMOVEDEVICE(DEVICEID) ............................................................................................................................ 7
- UPDATEDEVICES().......................................................................................................................................... 8
- GETPOLLINGINTERVAL() ................................................................................................................................ 8
- SETPOLLINGINTERVAL(SECONDS) .................................................................................................................. 8
- GETSTATE() .................................................................................................................................................... 8
- SETSTATE(STATE) ........................................................................................................................................... 9
- CHECKSAFETY() .............................................................................................................................................. 9
- OPERATESAFETY(LEVEL) ............................................................................................................................... 9
- KILLAPPS() ..................................................................................................................................................... 9
- GETERRORMSG(INT ERR) ............................................................................................................................. 10
- GETFAILS() ................................................................................................................................................... 10
- INTTOSTRING(INT C) .................................................................................................................................... 10
- STRINGTOINT(STRING S)................................................................................................................................ 10

APPENDIX A – LPRDS ERROR CODES .................................................................................................................... 11
APPENDIX B – LPRDS OPERATION STATES........................................................................................................ 12
• **connect(address, pid)**

  *Return Type:* int
  *Parameters:*
  - address: A string object to be used as in creation of the communication pipe between the application and the LPRDS Kernel.
  - pid: The process id of the application as an integer.

  *Description:*
  - This method must be called immediately upon startup of an application that uses the LPRDS Kernel. The LPRDS Kernel must already be running in order for this method to work fully. When this method is called, a communication pipe is created to receive messages from the LPRDS Kernel. Additionally, a message is sent to the LPRDS Kernel through the kernel communication pipe in order to register the application with the kernel process. The message contains the application address, used for communication, and the application’s process id which is used by the LPRDS Kernel process to terminate the application if necessary.

• **disconnect()**

  *Return Type:* none
  *Parameters:*
  - none

  *Description:*
  - This method must be called before termination of an application that is registered with the kernel process. When called, this method sends a packet through the kernel communication pipe in order to remove the application from the kernel registry. Additionally, the communication pipe that was created at the start of the application by use of the connect method is removed.

• **getAddress()**

  *Return Type:* string
  *Parameters:*
  - none

  *Description:*
  - This method checks for and returns the address used by the application when communication with the LPRDS Kernel process.

• **getDeviceList()**

  *Return Type:* list<string>
  *Parameters:*
  - The returned value is an STL List container of string objects. The string objects are the names of each active device recognized by the LPRDS Kernel process. The device names are those specified in the LPRDS Hardware Description XML file.
Description:

- This method communicates with the LPRDS Kernel process through the kernel communication pipe. A packet is sent requesting the name of each active device (sensors and switches) in the system. After sending this packet, the application waits for a response from the kernel process. The response is sent from the kernel process via the application’s communication pipe that was established upon use of the connect method. The received packet is then parsed into the STL List of string objects, containing the device names, which is available for use by the application. This method has two corollary methods, getSensorList and getSwitchList, that perform the same actions except requesting only sensor names and only switch names, respectively.

- **getSensorList()**
  
  **Return Type:** list<string>
  
  - The returned value is an STL List container of string objects. The string objects are the names of each active sensor recognized by the LPRDS Kernel process. The sensor names are those specified in the LPRDS Hardware Description XML file.

  **Parameters:**
  
  - none
  
  **Description:**
  
  - This is a corollary method to the getDeviceList method. It communicates with the LPRDS Kernel process through the kernel communication pipe. A packet is sent requesting the name of each active sensor in the system. After sending this packet, the application waits for a response from the kernel process. The response is sent from the kernel process via the application’s communication pipe that was established upon use of the connect method. The received packet is then parsed into the STL List of string objects, containing the sensor names, which is available for use by the application.

- **getSwitchList()**
  
  **Return Type:** list<string>
  
  - The returned value is an STL List container of string objects. The string objects are the names of each active switch recognized by the LPRDS Kernel process. The switch names are those specified in the LPRDS Hardware Description XML file.

  **Parameters:**
  
  - none
  
  **Description:**
  
  - This is a corollary method to the getDeviceList method. It communicates with the LPRDS Kernel process through the kernel communication pipe. A packet is sent requesting the name of each active switch in the system. After sending this packet, the application waits for a response from the kernel process. The response is sent from the kernel process via the application’s communication pipe that was established upon use of the
connect method. The received packet is then parsed into the STL List of string objects, containing the switch names, which is available for use by the application.

- **getDetails(deviceID)**
  
  *Return Type:* vector<string>
  
  - The returned value is an STL Vector container of string objects. The string objects are the device details. Depending upon what type of device the details were requested for, the vector will be of two different sizes. If the device details were requested for a sensor device the vector will be of length 10 and if the details were requested for a switch the vector will be of length 5. The following diagram, Figure 1, shows a visual representation of the possible formats of the vector returned by this method.

  **Figure 1: Format of getDetails return vectors**

<table>
<thead>
<tr>
<th>For a sensor:</th>
<th>For a switch:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>Type</td>
<td>Type</td>
</tr>
<tr>
<td>Units</td>
<td></td>
</tr>
<tr>
<td>* Offset</td>
<td>* DAQ</td>
</tr>
<tr>
<td>* Scale</td>
<td>* Pin</td>
</tr>
<tr>
<td>* DAQ</td>
<td></td>
</tr>
<tr>
<td>* Channel</td>
<td>* Details</td>
</tr>
<tr>
<td>* Max</td>
<td></td>
</tr>
<tr>
<td>* Min</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td></td>
</tr>
</tbody>
</table>

  * - indicates the string object will need to be converted to type float before use
  
  * - indicates the string object will need to be converted to type int before use

- **getUnits(sensorID)**
  
  *Return Type:* string
- The returned value is a string object of the units of the sensor device specified by the parameter sensorID.

Parameters:
- **sensorID**: A string object representing the name of the sensor for which the units are requested.

Description:
- This method communicates with the LPRDS Kernel process to request the units of the specified sensor. The units parameter of the sensor is established in the LPRDS Hardware Description XML file. In the case that the specified sensor name is not valid, the string object returned will be “no_device”. If the device specified exists but is not a sensor the returned string will be “not_sensor”. If the device is a sensor but has no units specified the returned string will be “no_units”.

### getValue(sensorID)

**Return Type**: float

- The returned value is either the sensor reading, after adjustment by the scale and offset fields (found in the LPRDS Hardware Description XML file) by the kernel process, or an error code; see LPRDS Error Codes in Appendix A.

Parameters:
- **sensorID**: A string object representing the name of the sensor for which the units are requested.

Description:
- The method communications with the LPRDS Kernel process to poll a sensor, specified by the parameter sensorID, in the system. The method returns either the sensor reading or an error code (as a float value). The error codes can be found in Appendix A.

### getValue(daq,channel)

**Return Type**: float

- The returned value is either the sensor reading, before adjustment, or an error code; see LPRDS Error Codes in Appendix A.

Parameters:
- **daq**: An int value which is the address of the data acquisition board (DAQ) with the sensor to be read.
- **channel**: An int value which is the A-D channel number on the data acquisition board (DAQ) where the sensor is connected.

Description:
- The method communications with the LPRDS Kernel process to poll a sensor, specified by the parameters daq and channel, in the system. The method returns either the sensor reading or an error code (as a float value). The error codes can be found in Appendix A.

### getLastData(sensorID)

**Return Type**: float

- The returned value is either the previously recorded sensor reading from the LPRDS database or an error code; see LPRDS Error Codes in Appendix A.
Parameters:
- **sensorID**: A string object representing the name of the sensor for which the units are requested.

Description:
- The method communicates with the LPRDS Kernel process to request the most recent reading from a sensor, specified by the parameter sensorID, in the LPRDS database. The method returns either the sensor reading or an error code (as a float value). The error codes can be found in Appendix A.

- **setSwitch(switchID,level)**
  
  **Return Type**: int
  - The returned value is an error code; see LPRDS Error Codes in Appendix A.

  **Parameters**:
  - **switchID**: A string object representing the name of the switch to be set.
  - **level**: The level is an integer 1 or 0 representing on or off, respectively.

  **Description**:
  - This method operates a switch in the LPRDS system, specified by switch name, by sending a packet to the LPRDS Kernel through the kernel communication pipe. The value returned by this method is an integer error code.

- **setSwitch(daq,pin,level)**
  
  **Return Type**: int
  - The returned value is an error code; see LPRDS Error Codes in Appendix A.

  **Parameters**:
  - **daq**: An int value the is the address of the DAQ that has the switch to be set.
  - **pin**: An int value that is the pin number of the switch on the DAQ.
  - **level**: The level is an integer 1 or 0 representing on or off, respectively.

  **Description**:
  - This method operates a switch in the LPRDS system, specified by DAQ and pin number, by sending a packet to the LPRDS Kernel through the kernel communication pipe. The value returned by this method is an integer error code.

- **addDevice(switchID,daq,pin,details)**
  
  **Return Type**: int
  - The returned value is an error code; see LPRDS Error Codes in Appendix A.

  **Parameters**:
  - **switchID**: A string object representing the name of the switch.
  - **daq**: An int value that is the address of the DAQ that has the switch.
  - **pin**: An int value that is the pin number of the switch on the DAQ.
  - **details**: A string object that contains a brief human readable description of the device. No spaces are permitted in the description.

  **Description**:
  - This method allows an application to add a new switch to the LPRDS system. In order to do this a packet is sent through the kernel communication pipe containing the device parameters specified by switchID, daq, pin, and details. The LPRDS Kernel process updates the LPRDS Hardware Description
XML file and updates the internal device lists to include the new switch. The kernel process sends a packet to through the application’s communication pipe acknowledging the success or failure of the addition. The packet contains an error code; these error codes can be found in Appendix A.

- **addDevice(sensorID,type,units,offset,scale,daq,channel,max,min,details)**

  *Return Type: int*
  
  - The returned value is an error code; see LPRDS Error Codes in Appendix A.

  *Parameters:*
  
  - **sensorID**: A string object representing the name of the sensor to be added to the system.
  - **type**: A string object representing the type of sensor to be added. Valid sensor types are temperature (“temp”), voltage (“volt”), or current (“curr”).
  - **units**: A string object representing the units of the sensor to be added.
  - **offset**: A float value that is the offset to be applied to sensor readings.
  - **scale**: A float value that is the scale factor to be applied to sensor readings.
  - **daq**: An int value that is the address of the DAQ that has the sensor.
  - **pin**: An int value that is the channel number of the sensor on the A-D on the DAQ.
  - **max**: An int value that is the maximum safe reading of the sensor. A reading greater than this value will trigger a fault in the LPRDS system.
  - **min**: An int value that is the minimum safe reading of the sensor. A reading below this value will trigger a fault in the LPRDS system.
  - **details**: A string object that contains a brief human readable description of the device. No spaces are permitted in the description.

  *Description:*
  
  - This method allows an application to add a new sensor to the LPRDS system. In order to do this a packet is sent through the kernel communication pipe containing the device parameters specified by sensorID, type, units, offset, scale, daq, channel, max, min, and details. The LPRDS Kernel process updates the LPRDS Hardware Description XML file and updates the internal device lists to include the new sensor. The kernel process sends a packet to through the application’s communication pipe acknowledging the success or failure of the addition. The packet contains an error code; these error codes can be found in Appendix A.

- **removeDevice(deviceID)**

  *Return Type: int*
  
  - The returned value is an error code; see LPRDS Error Codes in Appendix A.

  *Parameters:*
  
  - **deviceID**: A string object representing the name of the device to be removed from the system.

  *Description:*
  
  - This method allows an application to remove a device from the LPRDS system. To do this a packet is sent through the kernel communication pipe containing the name of the device to be removed, specified by deviceID. An error code is sent back to the application through the application’s
communication pipe. The error codes for the LPRDS system are explained in Appendix A.

- **updateDevices()**
  
  **Return Type:** int
  
  - The returned value is an error code; see LPRDS Error Codes in Appendix A.
  
  **Parameters:**
  
  - none
  
  **Description:**
  
  - The method sends a packet to the LPRDS Kernel process instructing it to re-scan the LPRDS Hardware Description XML file. This method can be used to update the kernel process’ internal device lists if the hardware description file has been updated manually or by another means than the supplied addDevice and removeDevice methods. (*WARNING:* Changing the LPRDS Hardware Description XML file by any means other than the supplied addDevice and removeDevice methods is not recommended and may result in corruption of the device lists and/or the hardware description file!)

- **getPollingInterval()**
  
  **Return Type:** int
  
  - The returned value is the current system polling interval in seconds.
  
  **Parameters:**
  
  - none
  
  **Description:**
  
  - The method requests the current system polling interval from the LPRDS Kernel process. The polling interval is how often the system polls each sensor for a reading.

- **setPollingInterval(seconds)**
  
  **Return Type:** int
  
  - The returned value is an error code; see LPRDS Error Codes in Appendix A.
  
  **Parameters:**
  
  - seconds: An int value that is the number of seconds the polling interval will be changed to.
  
  **Description:**
  
  - This method instructs the LPRDS Kernel process to change how often each sensor in the LPRDS system is polled and how often sensor readings are saved in the LPRDS database. An error code is returned to the application by the LPRDS Kernel process.

- **getState()**
  
  **Return Type:** int
  
  - The returned value is an integer representing the current operational system state.
  
  **Parameters:**
  
  - none
  
  **Description:**
- The method requests the current operational system state from the LPRDS Kernel process. The system state is managed by the LPRDS Operational State Manager application. The LPRDS operational states and corresponding integers are explained in Appendix B.

- **setState(state)**
  
  **Return Type:** int
  - The returned value is an error code; see LPRDS Error Codes in Appendix A.
  
  **Parameters:**
  - state: An int value representing the next operational state of the LPRDS system.
  
  **Description:**
  - The method is used to change the current operational state of the LPRDS system in the LPRDS Kernel process. The LPRDS operational states and corresponding integers are explained in Appendix B.

- **checkSafety()**
  
  **Return Type:** int
  - The returned value is 1 or 0, representing closed or open, respectively.
  
  **Parameters:**
  - none
  
  **Description:**
  - This method can be used by an application to check the status of the safety loop. The status of the loop is returned as an integer value, a 1 for closed or a 0 for open.

- **operateSafety(level)**
  
  **Return Type:** int
  - The returned value is an error code; see LPRDS Error Codes in Appendix A.
  
  **Parameters:**
  - level: An integer 1 or 0 representing closed or open, respectively.
  
  **Description:**
  - This method can be used by an application to operate the SCADA safety loop relay. The input parameter, level, as an integer 1 or 0 determines what action is performed. If level = 0 the safety loop relay is opened. If it is a 1 the safety loop relay is closed.

- **killApps()**
  
  **Return Type:** int
  - The returned int value is a LPRDS error code.
  
  **Parameters:**
  - none
  
  **Description:**
  - This method communicates with the LPRDS Kernel process via the kernel communication pipe. The purpose of the method is to provide a means of terminating all applications registered with the LPRDS Kernel process through the connect method. The only application permitted to use this
method is the LPRDS Operational State Manager application. The only applications that are not terminated by use of this method are the LPRDS Kernel process and the LPRDS Operational State manager. The applications are terminated by the SIGURG signal, which should be handled by the application.

- **getErrorMsg(int err)**
  
  **Return Type:** string
  - The error message corresponding to the specified error code.
  
  **Parameters:**
  - `err`: The error code
  
  **Description:**
  - This method is provided as a means of standardizing error codes and error messages. The error codes handled are described in Appendix A.

- **getFails()**
  
  **Return Type:** vector<string>
  - The returned STL container type vector of strings contains each LPRDS system device that has experienced an LV fault since the previous check.
  
  **Parameters:**
  - none
  
  **Description:**
  - The method communicates with the LPRDS Kernel process via the kernel communication pipe in order to get a list of the devices that have experienced an LV fault. An LV fault consists of a sensor reading that is out of the bounds specified in the XML Hardware Description file for that particular device.

- **intToString(int c)**
  
  **Return Type:** string
  - The specified integer value converted to a C++ string.
  
  **Parameters:**
  - `c`: The integer to be converted
  
  **Description:**
  - This method converts the specified integer to the string type.

- **stringToInt(string s)**
  
  **Return Type:** int
  - The specified string converted to an integer value.
  
  **Parameters:**
  - `s`: The string to be converted
  
  **Description:**
  - This method converts the specified string to an integer value, if possible.
### Appendix A – LPRDS Error Codes

<table>
<thead>
<tr>
<th>Error Code (int)</th>
<th>Error Code (float)</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>-99991</td>
<td>Device does not exist</td>
</tr>
<tr>
<td>2</td>
<td>-99992</td>
<td>Wrong device type</td>
</tr>
<tr>
<td>3</td>
<td>-99993</td>
<td>DAQ communication timeout</td>
</tr>
<tr>
<td>4</td>
<td>-99994</td>
<td>DAQ board communication error</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Pin no. is out of bounds</td>
</tr>
<tr>
<td>6</td>
<td>-99996</td>
<td>A-D channel no. is out of bounds</td>
</tr>
<tr>
<td>7</td>
<td>-99997</td>
<td>Sensor reading exceeds maximum value</td>
</tr>
<tr>
<td>8</td>
<td>-99998</td>
<td>Sensor reading below minimum value</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>Unable to access LPRDS Hardware Description XML file</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>Device ID already exists</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>LPRDS Kernel is already running</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>LPRDS Kernel is already shutdown</td>
</tr>
<tr>
<td>13</td>
<td>-99913</td>
<td>Unable to perform action, LPRDS Kernel is shutdown</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>SQL command conversion error</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>SQL query error</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>Unknown SQL error</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>SQL invalid name</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>SQL invalid DAQ/pin combination</td>
</tr>
<tr>
<td>19</td>
<td>-99919</td>
<td>Write to pipe error</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>Kernel &amp; Results pipe do not exists, LPRDS Kernel may not be running</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>Kernel pipe does not exists, LPRDS Kernel may not be running</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>Results pipe does not exists, LPRDS Kernel may not be running</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>Requested operation is not permitted</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>Unable to connect to LPRDS Kernel</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>SIB is not connected</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>SIB communication error</td>
</tr>
<tr>
<td>27</td>
<td>-99927</td>
<td>LPRDS API communication pipe read timeout</td>
</tr>
</tbody>
</table>

These error codes may be returned if the method return type is a float or integer type. If the return type is of type float, the error codes possible are found in the Error Code (float) column. If the return type is of type int, the error codes possible are found in the Error Code (int) column. If the return type of a method is not of type int or float, there are different possibilities for errors. If the return type is a string, the error will be described in the string. If the return type is an STL container, such as a List or Vector, the container will be empty if there was an error.
## Appendix B – LPRDS Operation States

<table>
<thead>
<tr>
<th>State Number</th>
<th>State Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Shutdown</td>
</tr>
<tr>
<td>2</td>
<td>Booting</td>
</tr>
<tr>
<td>3</td>
<td>LV Standby Unsafe</td>
</tr>
<tr>
<td>4</td>
<td>LV Standby Safe</td>
</tr>
<tr>
<td>5</td>
<td>HV Standby</td>
</tr>
<tr>
<td>6</td>
<td>HV Unsafe</td>
</tr>
<tr>
<td>7</td>
<td>Operational</td>
</tr>
<tr>
<td>8</td>
<td>Off</td>
</tr>
<tr>
<td>9</td>
<td>Clueless</td>
</tr>
<tr>
<td>10</td>
<td>LED Demo</td>
</tr>
</tbody>
</table>