C++ code to build Ladder simple editor used in PLC that is made by me, including functions to add or modify rungs, analog and timers in the ladder logic and functions to get and parser ladder data via rs232.

Creating a comprehensive C++ code for a Ladder Logic editor for PLCs involves multiple components, including handling user inputs for adding/modifying rungs, analog inputs, and timers, as well as managing serial communication over RS232. Below is a simplified version to get you started with the key features and functions. Note that for a full-fledged editor, you'd need additional error handling, a user interface, and more sophisticated data structures.

**Simplified Structure**

1. **Class Definitions for Ladder Logic Components**
2. **Functions to Add or Modify Rungs, Analog Inputs, and Timers**
3. **RS232 Serial Communication Functions**

**Code Example**

#include <iostream>

#include <vector>

#include <string>

#include <fstream>

#include <sstream>

#include <serial/serial.h>

// Basic structure to hold Rung, Analog Input, and Timer

struct Rung {

int id;

std::string logic;

Rung(int id, std::string logic) : id(id), logic(logic) {}

};

struct AnalogInput {

int id;

double value;

AnalogInput(int id, double value) : id(id), value(value) {}

};

struct Timer {

int id;

int duration;

Timer(int id, int duration) : id(id), duration(duration) {}

};

class LadderEditor {

private:

std::vector<Rung> rungs;

std::vector<AnalogInput> analogInputs;

std::vector<Timer> timers;

public:

void addRung(int id, std::string logic) {

rungs.emplace\_back(id, logic);

}

void modifyRung(int id, std::string newLogic) {

for (auto& rung : rungs) {

if (rung.id == id) {

rung.logic = newLogic;

return;

}

}

}

void addAnalogInput(int id, double value) {

analogInputs.emplace\_back(id, value);

}

void modifyAnalogInput(int id, double newValue) {

for (auto& input : analogInputs) {

if (input.id == id) {

input.value = newValue;

return;

}

}

}

void addTimer(int id, int duration) {

timers.emplace\_back(id, duration);

}

void modifyTimer(int id, int newDuration) {

for (auto& timer : timers) {

if (timer.id == id) {

timer.duration = newDuration;

return;

}

}

}

std::string serializeLadderData() {

std::stringstream ss;

for (const auto& rung : rungs) {

ss << "Rung: " << rung.id << " Logic: " << rung.logic << "\n";

}

for (const auto& input : analogInputs) {

ss << "AnalogInput: " << input.id << " Value: " << input.value << "\n";

}

for (const auto& timer : timers) {

ss << "Timer: " << timer.id << " Duration: " << timer.duration << "\n";

}

return ss.str();

}

void deserializeLadderData(const std::string& data) {

std::stringstream ss(data);

std::string line;

while (std::getline(ss, line)) {

std::istringstream iss(line);

std::string type;

int id;

if (line.find("Rung") != std::string::npos) {

std::string logic;

iss >> type >> id >> type >> std::ws;

std::getline(iss, logic);

addRung(id, logic);

} else if (line.find("AnalogInput") != std::string::npos) {

double value;

iss >> type >> id >> type >> value;

addAnalogInput(id, value);

} else if (line.find("Timer") != std::string::npos) {

int duration;

iss >> type >> id >> type >> duration;

addTimer(id, duration);

}

}

}

void sendDataOverRS232(serial::Serial& rs232Port) {

std::string data = serializeLadderData();

rs232Port.write(data);

}

std::string receiveDataOverRS232(serial::Serial& rs232Port) {

return rs232Port.read(rs232Port.available());

}

};

int main() {

// Fake example just for demonstration

LadderEditor editor;

editor.addRung(1, "XIC Start OTE Light");

editor.addAnalogInput(1, 5.5);

editor.addTimer(1, 100);

// Serialize and print the ladder data

std::string ladderData = editor.serializeLadderData();

std

C++

Copy code

Copy

Good response

Bad response

contnue

Sure, I'll continue with the remaining part involving the interaction using serial port.

std::cout << "Serialized Ladder Data:\n" << ladderData << std::endl;

// Assume we have a serial object ready for RS232 communication.

// Replace "/dev/ttyUSB0" with your actual RS232 port address.

serial::Serial rs232Port("/dev/ttyUSB0", 9600, serial::Timeout::simpleTimeout(1000));

// Sending data over RS232

editor.sendDataOverRS232(rs232Port);

// Simulate receiving data over RS232 (for demonstration, using the same data sent)

std::string receivedData = editor.receiveDataOverRS232(rs232Port);

std::cout << "Received Ladder Data:\n" << receivedData << std::endl;

// Deserialize received data

editor.deserializeLadderData(receivedData);

// Print the internal state after deserialization

std::cout << "Deserialized Ladder Data:\n" << editor.serializeLadderData() << std::endl;

return 0;

}

C++

Copy code

**Explanation**

1. **Serialize Ladder Data**: The serializeLadderData function converts the ladder data into a string. This string represents the internal state (rungs, analog inputs, timers).
2. **Send Data Over RS232**: The sendDataOverRS232 method sends the serialized ladder data over an RS232 port. This method uses the serial library for serial communication handling.
3. **Receive Data Over RS232**: The receiveDataOverRS232 method reads data from the RS232 port. This received string simulation was used here for demonstration purposes — in real usage, it would come from an external source.
4. **Deserialize Ladder Data**: The deserializeLadderData method parses the received string to reconstruct the internal state of the ladder logic editor. It reads each line, interprets it, and updates the stored rungs, analog inputs, and timers accordingly.

**Points to Note**

* **Serial Library**: The example uses the serial library which needs to be installed. You can install it using apt-get install libserial-dev on Linux or equivalent for your OS.
* **Error Handling**: Proper error handling is crucial for a real-life application but it’s simplified here for brevity.
* **Port Configuration**: Ensure the right serial port configuration matching the connected RS232 device.

Feel free to ask if you need more specific details or more advanced features for your ladder logic editor!