Описание импортируемых функций из библиотеки A100DLL

bool **a100UIntToData(*const unsigned int arg_val, unsigned char data_arr[4]*)**
Преобразует ***arg_val*** в массив ***data_arr; true - успешно***

unsigned int **a100DataToUInt(*const unsigned char data_arr[4]*);**
Преобразует массив ***data_arr*** в беззнаковый целый тип

bool **a100FloatToData(*const float arg_val, unsigned char float_point_count, unsigned char data_arr[4]*);**
Преобразует ***arg_val*** в массив ***data_arr***, ***float_point_count*** - количество знаков после запятой от 1 до 3; ***true - успешно***

float **a100DataToFloat(const unsigned char data_arr[4]);**
Преобразует массив ***data_arr*** в тип с плавающей запятой

bool **a100OpenSerialPort(*const std::string &interface_name, const int &interface_speed*);**
Открывает последовательный порт ***interface_name*** на скорости ***interface_speed; true – успешно***.
interface_name принимает значения "COM1", "COM2", "COM3", ... ,"COM255"
interface_speed принимает значения 9600, 19200, 38400

int **a100GetSerialPortCurrentState();**
Возвращает состояние последовательного порта открытого функцией **a100OpenSerialPort**
Возвращаемые значения:
0 - интерфейс закрыт
1 - интерфейс открыт и доступен для отправки сообщения контроллеру
2 - интерфейс открыт и находится в состоянии чтения информации (получение ответного сообщения от контроллера)
3 - интерфейс открыт и находится в состоянии записи информации (отправка сообщения контроллеру)

void a100CloseSerialPort();
Закрывает последовательный порт, открытый функцией **a100OpenSerialPort**

int **a100SendCommand(*unsigned char receiver_address, unsigned char sender_address, unsigned char command_code, unsigned char command_param, unsigned char data_array[4], int write_timeout, int read_timeout, pA100CallBackFunc a100callBackFunc, bool b_wait*);**
Данная функция не является thread-safe т.е. для многопоточных приложений нужно продумать синхронизацию потоков.
Отправка осуществляется в асинхронном режиме (***параметр b_wait*** равен ***false***) т.е. после вызова данной функции приложение продолжит работу, а не будет "зависать" на этой функции до прихода и обработки ответа. ***ВНИМАНИЕ!!!*** если ***параметр b_wait*** равен ***true*** то после вызова данной функции приложение не продолжит работу до тех пор пока не завершится сеанс передачи/приёма сообщения

Отправка сообщения возможна только в случае когда функция **a100GetSerialPortCurrentState** возвращает 1

Возвращаемые значения:
0 - ОК, нет препятствий для отправки сообщения, процесс отправления начался
1 - ошибка интерфейс занят т.е. функция getSerialPortCurrentState вернула значение отличное от 1
2 - ошибка data_array == NULL (не определен)
3 - ошибка "упаковки" сообщения
4 - ошибка дескриптора последовательного порта

Описание параметров:
receiver_address - адрес получателя т.е. адрес контроллера который выставляется в соответствующем меню самого контроллера
sender_address - адрес отправителя, обычно выставляется в 0x1f, теоретически в приложении может быть несколько потоков которые посылают сообщения одному или разным контроллерам, тогда адреса отправителя будут свои для каждого потока
command_code - код команды, коды поддерживаемых команд написаны в документации к контроллеру
command_param - параметр команды (дополнительные данные, байт номер 3), параметры поддерживаемых команд написаны в документации к контроллеру
data_array - данные, целого типа либо типа с плавающей запятой, формируются при помощи функций **a100UIntToData** и **a100FloatToData**
write_timeout - максимальное время в течение которого будет осуществляться отправка сообщения, в мс
read_timeout - максимальное время в течение которого будет ожидаться первый байт ответного сообщения, последующие байты принимаются до тех пор пока они буду посылаться контроллером, в мс
a100callBackFunc - указатель на функцию пользователя, данная функция вызывается при получении ответа он контроллера либо при возникновении ошибок приёма/передачи

Описание параметроф функции пользователя которая вызывается при получении ответа от контроллера

void a100callBackFunc (*unsigned char dest, unsigned char source, unsigned char code, unsigned char param, unsigned char status, unsigned char error, unsigned char data[4], int error_status*);

Описание параметров:
dest - адрес получателя ответного сообщения сообщения (равен параметру *sender_address* в вункции **a100SendCommand**)
source - адрес отправителя ответного сообщения (равен параметру *receiver_address* в вункции **a100SendCommand**)
code - код команды (равен параметру *command_code* в вункции **a100SendCommand**)
param - параметр команды (равен параметру *command_param* в функции **a100SendCommand**)
status - слово состояния отправителя
error - номера ошибок, код фазы выполняемой работы
data[4] - данные от контроллера
error_status - принимает следующие знаения:
0 - ОК, принят корректный ответ
1 - ошибка отправки данных, превышен *write_timeout* (нет связи?)
2 - ошибка отправки данных (нет связи? драйвер? дескриптор?)

3 - ошибка чтения данных, превышен *read_timeout* (нет связи?)
4 - ошибка чтения данных (нет связи? драйвер? дескриптор?)
5 - ошибка декодирования сообщения

ВАЖНО!!! До тех пор, пока выполняется тело a100CallBackFunc интерфейс считается занятым и послать новое сообщение не удастся.

Дополнительные источники

Использование DLL в программе на Visual C++ :

http://www.rsdn.ru/article/baseserv/dlluse.xml