MODBUSCLIENT DLL API

I. INTRODUCTION

ModbusClient.dll allows easy remote control of the MProbe measurement via TCP-IP using Modbus protocol communication.
ModbusClient.dll can be located on the same computer as TFCompanioo software (that is connected to MProbe system) or on any other computer on the network.

ModbusClient.dll is a C library that is interfacing java implementation of the Modbus Client (located in Modbus.jar). ModbusClient.dll is communicating (via Modbus.jar) with TFCompanion's build-in ModbusServer software**.**

II. Configuration.

1. Java should be installed on the computer.
If you are using a different computer on a network (not the one where TFCompanion is installed) – check that java is installed.
You can check it by typing: **java –version** at command prompt.
2. ModbusClient.dll and **jvm.dll** should be added to the system path
(jvm.dll is located in java installation directory e.g.
C:\Program Files\Java\jre1.6.0_18\bin\client)
Note. The log file (Modbus.log) will be saved in the working directory of the program that calls **ModbusClient.dll**
The structure of the installation directory:

```
<home_directory>/
                    <executable program that uses ModbusClientDLL..dll >
                    Modbus.properties (optional)
              library/
                        --- ModbusClientDLL..dll
                        --- Modbus.jar
                        --- tflib.jar
                        --- log4j.jar
                        --- comm..jar
            log/
                ---log_config.properties
```

<home_directory> is the directory where executable (calling) program (the program that uses ModbusClientDLL.dll) is located e.g. ModbusExample.exe

83 Pine Hill Rd. Southborough, MA 01772
Phone +1.617.388.6832 Fax. +1.508.858.5473
email: info@semiconsoft.com http://www.semiconsoft.com

III. API DETAILS

1. Initialization.

This need to be the first call to the dll. This method is preparing and starting JVM

```
/**
 @param ip – IP address of the computer with Modbus server. To determine IP address, use ipconfig at command prompt. If computer is not connected – the IP address will a “localhost”
It is expected that TFCompanion server (MProbe) is started at the port 502
 If ip address=0, the program will try to read IP address and port information from the Modbus.properties file
 @ret  0 – success, otherwise failed.
*/
long  init(char* ip);
```

2. Connection to server

```
/*
Connects Modbus Client to Modbus server. The IP address is already set during initialization, the  port#502 is a default Modbus port that is used here.
If you are using firewall or virus protection program – please make sure that port#502 is open.
By default connection timeout is 3000 ms. If response from the server is not received during this time – connection is terminated. The client will try to reestablish connection 2 times before giving up.
To set custom timeout use setConnectionTimeout(int time_ms) method – this method need to be called BEFORE connectClient() otherwise it will have not effect.

return boolean. 1 - true: connection is successful, 0 (false) - problem during connection

*/
 unsigned char connectClient();
```

3. **Setting measurement parameters**.

Measurement parameters need to be set before the first measurement is executed. These parameters are remembered and used for all future measurements. If needed, parameters can be reset at any time.

```
/**
     * Sets measurement parameters in the output Hashtable to prepare request.
     * The content of the Hashtable is maintained and used for all transactions.
     * @param channel - channel number (0 or 1)
     * @param waferID - Sample ID (up to 40 bytes length)
     * @param recipeName - name of the measurement or calibration recipe (up to 10
       bytes long)
     * @param expectedThickness - OPTIONAL PARAMETER (0 - to ignore). Value of the
         Expected thickness - resets the thickness of the currently used filmstack
     * @param intTime - OPTIONAL PARAMETER (0 - to ignore). Resets integration
     time in the specified recipe measurement recipe
     * @return true (1) if successful
     */

unsigned char setMeasurement(int channel,
                             char* waferID,
                             char* recipeName,
                             int expectedThickness,
                             short intTime)
```

4. Measurement

This methods instructs MProbe to perform the measurement. The measurement is fully defined by the measurement recipe (recipe should have a filmstack attached). Measurement call is blocking (it is waiting for the response from the server). The default timeout is set to 3 sec. (see @setConnectionTimeout()) If response is not received – client will disconnect and try connect again (it will repeat it 3 times before giving up)

```
/**
     * Execute measurement or calibration based on the data set in the output
       Hashtable defined by setMeasurement and setRecipe
     * @return response from the server. This response can be parsed directly
       using Response(Read) table spec or
     * use convenience methods that parse response information.
     * @see  hasException(),getThickness()
     */
signed char* measure()
```

83 Pine Hill Rd. Southborough, MA 01772
Phone +1.617.388.6832 Fax. +1.508.858.5473
email: info@semiconsoft.com http://www.semiconsoft.com

5. Checking measurement results

Following are convenience methods that can be called AFTER measurement is performed.

5.1 Checking for exception (Problems during measurement, calculation or communication)

```
/**
     * Parse server response to extract exception code during transaction
        (measurement/calibration)
     * Can be called after response was received
     * @return 0 - no exception, 1,2 or 3 - general Modbus errors (communication/
        format problem),
     * 4 - unknown sensor error, 5 - data acquisition problem, 6 - calculation
        problem,  7 - system problem (generic exception)
     */
signed char getExceptionCode()
```

5.2 Thickness data

```
/**
Parse server response to extract thickness data
@param channel – the measurement channel
@return  - thickness in Angstroms
*/

long getThickness(int channel)
```

5.3 Thickness confidence interval (90%)

```
/**
* Parse server response to extract confidence interval data
@param channel: measurement channel
@return confidence interval value
*/
double getThicknessError(int channel)
```

5.4 Goodness of Fit

```
/**
* Parse server response to extract GOF data
@param channel: measurement channel
@return GOF value
*/
double getGOF(int channel)
```

83 Pine Hill Rd. Southborough, MA 01772
Phone +1.617.388.6832 Fax. +1.508.858.5473
email: info@semiconsoft.com http://www.semiconsoft.com

6. Convenience methods to change measurement parameters

6.1 Changing measurement recipe

```
/**
     * A convenience method to change measurement recipe used in transaction.
     * @ see setMeasurement method.
     * The data is stored in the hashtable and will be used to prepare next
       response
     * @param recipeName the name of measurement recipe
     * @param channel measurement channel number
     * @return true (1)- recipe set successfully, false (0) - problem setting
        recipe.
     */

 unsigned char setRecipe(int channel, char* recipeName)
```

6.2. Toggle between measurement and measurement/diagnostics type.
measurement/diagnostics is the same as measurement only server send more information about the system

```
/**
     * Convenience method to switch between standard measurement and measurement
       w/ diagnostics Can be applied after setMeasurement was called
     * @see setMeasurement method
     * @param diag true (1) to enable measurement with diagnostics, false (0)
        standard measurement
     * standard measurement is the default
        *@return  true (1) if successful, false (0) if problem
     */
unsigned char setDiagnostics(unsigned char flag)
```

7. Diagnostic convenience methods.

These methods can be used ONLY when measurement with diagnostics is enabled. They parse server response to extract requested data, so it should be called after the measurement is completed.

7.1. Warnings

```
/**
@return warning code
NO_WARNING=0
LOW_SIGNAL=1 (low light intensity)
HIGH_SIGNAL 2 (intensity maybe too high - saturation)
LOW_MEMORY 3 (low memory available to the program)
*/
signed char getWarning()
```

7.2. Signal level

Need to be called to determine actual signal level if there is a low or high signal warning.

```
/*
@param channel: measurement channel
```

```
@return actual signal level (intensity) in % of the total range
*/
short getSignalLevel(int channel)
```

7.3. Integration time

Actual integration time used by spectrometer

```
/**
Return actual integration time used during the measurement
@param channel channel number
@return integration time in ms
*/
short getIntegrationTime(int channel)
```

8. System status.

Checking system status after the last measurement call.

```
/*

     * Parse server response to check system status
     * Can be called after response was received
     * @return 0 - ready/idle,1 - measuring, 2 - calculating, 3 - exception
state,4 - busy/initializing
*/
signed char getSystemStatus()
```

9. Connection checking.

```
/*
Checking the status of the connection
@return  0 – false (disconnected), 1-true (connected)

*/
 unsigned char isClientConnected()

/*
Sets connection timeout in milliseconds. Default timeout id 3000ms
Setting timeout to 0 –means connection will never timout.
IMPORTANT: This method should be called BEFORE calling connectClient();
Otherwise it will have no effect
*/
void setConnectionTimeout(int time_ms);

/**
@return current timout in milliseconds
*/
int getConnectionTimeout();
```

83 Pine Hill Rd. Southborough, MA 01772
Phone +1.617.388.6832 Fax. +1.508.858.5473
email: info@semiconsoft.com http://www.semiconsoft.com

10. Checking intensity (Added Jan. 10, 2017)

10.1. Check Maximum Signal.

This method allows to check maximum signal from the system before doing actual measurement. A full spectrum is acquired and a maximum value of the intensity is determined. Averaging of the 10 spectra is done to have accurate data.
In case of the system with Flush lamp – setStrobe() command should be send first to activate synchronization pulses.
In case of all other systems, integration time should be checked to make sure it is the same as will be used during measurement (signal value will depend directly on integration time)

```
/**
@return maximum intensity as counts of a 16 bit ADC from 0 to 65500
*/
unsigned short getMaxIntensity()
```

10.2 Set Intensity

This method allows to set intensity of the light source.

```
/**
@param value – value in the range of 0 to 100, 100 – maximum intensity
@return a response from the server  (byte[])
*/
signed char* setIntensity(unsigned char value)
```

83 Pine Hill Rd. Southborough, MA 01772
Phone +1.617.388.6832 Fax. +1.508.858.5473
email: info@semiconsoft.com http://www.semiconsoft.com