TELETASK

PROJECT, COMPACT & MICROS
DOMOTICS SYSTEM

'WINDOWS' SOFTWARE LIBRARY

TDS_LIBRARY_EN.doc - 31/01/06

TELETASK DOMOTIC SYSTEMS WINDOWS LIBRARY V2.80

Platform: To be used on all IBM compatible personal computers with Microsoft WINDOWS 98/ME, WINDOWS 2000 (sp 3), WINDOWS XP (sp 2), WINDOWS Server 2003 operating system.

It is the purpose to generate a start-up environment for software developers, who are interested in generating their own PC-based control environment for the TELETASK domotics systems. In this way the developer can create his own user interface and connected solutions and services using a standard personal computer.

This situation should stimulate creative minds in working with their own control environment as an exclusive add-on to a TELETASK system. In this way the developer can generate solutions which are not available at TELETASK but only on his own to be commercialized platform.

Applications: creating mouse- or touch screen controlled PC-screens with ground floors of the particular environment where the TELETASK system is installed. This can be in a very exclusive private villa, but it even can be a central controller which is connected to several houses in a larger project (as there are houses for elderly people). The application list is very large and limited to your imagination and applications.

LICENCE AGREEMENT.

De TELETASK 'TDS Windows Library'- license is free of charge and is only to be used by developers who are the owner of at least one TELETASK central unit.

Due to the unlimited freedom of the developer, it is impossible for TELETASK to give free support or to carry any responsibility on the use and the results of the use of this libraries in any way. Claims in any way for any reason are not allowed.
By using this libraries, the developer and the user agree with this statement.

1 BUILD-UP OF THE EXAMPLE ROUTINES.

The routines are based on the following parts:

- The Windows Custom Control TELETASK.DLL wherein the control functions for the TELETASK systems are included.
  TDSLIBRARY.DLL is a 32-bit Dynamic link library.
- The example programs TDSComport.exe and TDSSocket.exe are simple examples of a graphical user interface which uses the available control functions on a C# 2.0 environment.

Remark: These parts are usable on personal computers which run on a Microsoft WINDOWS 98/ME, WINDOWS 2000 (sp 3), WINDOWS XP (sp 2), WINDOWS Server 2003 operating system.

1.1 Tds_Port

The two examples both use the “Tds_Port” object, the TdsComport example to open a connection to a TDS Central Unit through a “COM” port, the TdsSocket example to connect to a TDS Central Unit over Ethernet (for the connection over Ethernet an Ethernet to RS232 converter is needed, this can be purchased at Teletask as TDS10118).

1.1.1 Constructor

To create a new Instance of “Tds_Port”, three constructors are available

- Tds_Port(int PortNr, int BaudRate, int Tag)
- Tds_Port(string SrvName, int SrvPort, int Tag)
- Tds_Port(IPAddress Address, int SrvPort, int Tag)

The first constructor creates a “Tds_Port” for opening a “COM” port. The parameters “PortNr” and “BaudRate” are the “COM” port you want to open and the associated Baudrate. The “Tag” parameter is a general purpose integer you can use for a variety of reasons (e.g. To identify the connection if you build an application with multiple connections)

The two other constructors can be used to create a “Tds_Port” for opening an Ethernet connection. As first parameter you can either pass a string (e.g. a domain) or an IP-address. The “SrvPort” parameter is the port number to wich you want to connect. “Tag” is again a general purpose integer.

1.1.2 Functions

- `public bool Connect()`

  The Connect method is used to open the “Tds_Port”, if the connection was successfully established “*true*” is returned, if connecting failed the return value is “*false*”.

- `public void Close()`

  The “Close” method closes the connection, it is advisable to call this method before you exit your program or if you do not intend to send or receive data for a longer period.

- `public int FunctionSet(int Fnc, int Opt, int Number, int State)`

  With this function you set the indicated function to a state that correspond to the value in State
  Example: If you set Fnc=FNC_FLAG, Number=3, State=255 the central unit will set the flag with number 3 ON.

Input:
- Fnc (RELAY, DIMMER, MOTOR, MTRUPDOWN, LOCMOOD, TIMEDMOOD, GENMOOD, FLAG, PROCES, REGIME, SERVICE, MESSAGE)
- Number (dependent on the function)
- State (dependent on the function)
- Opt (dependent on the function: , SETTEMPUP, SETTEMPDOWN, SETTEMPVRIES, SETTEMPNIGHT, SETTEMPDAY, SETTEMPSETNIGHT, SETTEMPSETDAY, AUDIOUP, AUDIODOWN, AUDIOON, AUDIOOFF, AUDIOFM, AUDIOFM2, AUDIOCD, AUDIOCD2, AUDIOTAPE, AUDIOTAPE2, AUDIOVIDEO, AUDIOVIDEO2, AUDIOAUX, AUDIOAUX2, AUDIOKEY

Output:
- Return value: 0 = Message successfully transmitted
  -1 = Communication not opened
  -2 = No Answer

Remark 1:
for Fnc = FNC_RELAY, FNC_DIMMER, temperature functions & audio functions
-> Number = 1 to Maximum
for Fnc = FNC_REGIME (this Event occurs on a change of regime)
-> Number= 0 Automatic workday / weekend-day
1 Workday
2 Weekend-day
3 Simulation
4 Manual
for Fnc = FNC_TPKEY
-> Number = Touch panel number you want to simulate
All other Fnc
-> Number = 0 to Maximum -1

Remark 2:
for Fnc=FNC_DIMMER & FNC_MOTOR
-> State = 0 to 255 (always use the result from function ConvPercToDimVal, may never be 3!!!)
for Fnc=FNC_SETTEMPSETDAY & SETTEMPNIGHT
-> State = 0 to 255 (to calculate with functions ConvTempToSensVal, ConvHumToSensVal or ConvLuxToSensVal)
for Fnc = FNC_AUDIOKEY
-> State = 1 to 9 (preset)
for Fnc = FNC_TPKEY
-> State bit 3-0 = Key number (0 to 7)
-> State bit 7-8 = 00 Normal Short Press, 01 Key Depressed, 10 Key Released
for all other Fnc
-> State = 0 or 255 (or 1) = OFF or ON

- `public int FunctionGet(int Fnc, int Number)`

This function ask for the state of the indicated function
Example. If you call this function with Fnc=GET_SENSORVAL, Number=3 the central unit will transmit the value of sensor 3, and a event will occur (This will always happen regardless if the log-channel is open or not!).

Input:
- fnc (RELAY, DIMMER, MOTOR, MTRUPDOWN, LOCMOOD, TIMEDMOOD, GENMOOD, FLAG, PROCES, REGIME, SERVICE, MESSAGE, COND, GETSENSVAL, GETSENSTARGET, SETTEMPVRIES, SETTEMPNIGHT, SETTEMPDAY, TEMPAUTO, TEMPHEAT, TEMPSPLOW, TEMPSPMED, TEMPSPHIGH, TEMPSPAUTO, TEMPSETSTANDBY, TEMPONOFF, TEMPCOOL, SETTEMPSETNIGHT, SETTEMPSETDAY, AUDIOSTATE)
- Number (dependent on the function, look remarks at FunctionSet)

Output :
- Return value: 0 = Message successfully transmitted
  -1 = Communication not opened
  -2 = No Answer

- `public int FunctionLog(int Fnc, int State)`

  This function will open/close a channel for the function
  Example: If you call this function with the parameter Fnc=FNC_RELAY and State=1, all changes on relays will occur as 'event'! In case you set State=0 no more events will occur from relays.

  Input:
  - Fnc (FNC_DIMMER, FNC_RELAY, FNC_MOTOR, FNC_MTRUPDOWN, FNC_LOCMOOD, FNC_TIMEDMOOD, FNC_GENMOOD, FNC_FLAG , FNC_PROCES , FNC_REGIME , FNC_SERVICE, FNC_MESSAGE, FNC_COND , FNC_GETSENSVAL, FNC_GETSENSTARGET, FNC_AUDIOSTATE ,FNC_TPKEY)
  - State (0 of 1)

  Output:
  - Return value: 0 = Message successfully transmitted
    -1 = Communication not opened
    -2 = No Answer

  Remark: Use only the indicated functions. After a PROSOFT download all channels will be closed.

- `public int WriteDisplayMessage(int Number, String Line1, String Line2, int Beeps, int InterfaceAddress)`

  This function can be used to create "*Dynamic messages"*.
  Example: If you call this functions with the parameter Number=49, two lines of text, Beeps = 10, InterfaceAddress = 12, the next times that message 49 is activated it will show the two lines of text on Interface 12 and Interface 12 will beep 12 times.

  Input:
  - Number (the number of the message you want to change)
  - Line1, Line2 (two lines of text)
  - Beeps (the number of times you want the interface to beep)
  - InterfaceAddress (the AUTOBUS address of the interface you want the message to be displayed on)

  Output:
  - Return value: 0 = Message successfully transmitted
    -1 = Communication not opened
    -2 = No Answer

  Remark1
  This function will only work with a MICROS central unit, running V2.80 EPROMS or later. No COMPACT or PROJECT central unit supports this function.

  Remark2
  This function overwrites the message with the given Number, setting the message back to its old value requires a new PROSOFT download or a call to this function with the original data.

  Remark3
  This function only fills the message with the given Data, it does not activate the message. To activate the message use "FunctionSet".

  Remark4
  The lines of text are limited to 16 characters per line for ASCII characters and to 8 characters per line for Unicode characters. If you want to use a combination of Unicode and ASCII characters the maximum characters per line is also 8.

  Remark5
  If you want to generate an alarm, set the InterfaceAddress to '0'.

- `public String GetPortName()`

  This function will return the name of the "Tds_Port". For an Ethernet connection this will be the name or the IP-address, for a "COM" port this will be "COM1", "COM2",…

- `public int GetTag()`

This will return the “Tag” associated with the “Tds_Port”.

- `public int ConvertDimValToPerc(int val)`

  This function converts an integer value (derived from the event ‘Report’) to a Percent.

- `public int ConvertPercToDimVal(int val)`

  This function converts a percentage to an integer required for the Method ‘FunctionSet’.

- `public double ConvertSensValToTemp(int val)`

  This function converts an integer value (derived from the event ‘Report’ with met Fnc= FNC_GETSENSVAL, FNC_GETSENSTARGET, FNC_SETTEMPSETNIGHT, FNC_SETTEMPSETDAY) to degrees Celsius.

- `public int ConvertSensValToHum(int val)`

  This function converts an integer value (derived from the event ‘Report’ with met Fnc= FNC_GETSENSVAL, FNC_GETSENSTARGET, FNC_SETTEMPSETNIGHT, FNC_SETTEMPSETDAY) to % humidity.

- `public double ConvertSensValToLux(int val)`

  This function converts an integer value (derived from the event ‘Report’ with Fnc= FNC_GETSENSVAL, FNC_GETSENSTARGET, FNC_SETTEMPSETNIGHT, FNC_SETTEMPSETDAY) to a lux value.

- `public int ConvertTempToSensVal(double val)`

  This function converts degrees Celsius to an integer required for the Method ‘FunctionSet’ with Fnc= FNC_SETTEMPSETNIGHT, FNC_SETTEMPSETDAY).

- `public int ConvertHumToSensVal(double val)`

  This function converts % humidity to an integer required for the Method ‘FunctionSet’ with Fnc= FNC_SETTEMPSETNIGHT, FNC_SETTEMPSETDAY)

- `public int ConvertLuxToSensVal(double val)`

  This function converts a lux value to an integer required for the Method ‘FunctionSet’ with Fnc= FNC_SETTEMPSETNIGHT, FNC_SETTEMPSETDAY)

1.1.3 Events

- `public delegate void ReportEventHandler(Tds_Port sender, int Fnc, int Number, int State);`

  This event occurs when something changes in the central unit. (on condition that the log channel is open)

  Param:
  - Fnc (FNC_RELAY, FNC_DIMMER, FNC_MOTOR, FNC_MTRUPDOWN, FNC_LOCMOOD, FNC_TIMEDMOOD, FNC_GENMOOD, FNC_FLAG, FNC_PROCES, FNC_REGIME, FNC_SERVICE, FNC_MESSAGE, FNC_COND, FNC_GETSENSVAL, FNC_GETSENSTARGET, FNC_SETTEMPVRIES, FNC_SETTEMPNIGHT, FNC_SETTEMPDAY, FNC_SETTEMPSETNIGHT, FNC_SETTEMPSETDAY, FNC_AUDIOSTATE)
  - Number (dependent on the function, see the remarks in FunctionSet)
  - State(dependent on the function)

  For Fnc=FNC_DIMMER & FNC_MOTOR
  State = 0 to 255 (you can afterwards calculate the % value with ConvDimValToPerc)
  For Fnc=FNC_SETTEMPSETDAY & SETTEMPNIGHT
  State = 0 tot 255 (you can calculate the real value u with ConvSensValToTemp, ConvSensValToHum of ConvSensValToLux)
  For Fnc=FNC_AUDIOSTATE
  State = 36 ON 37 OFF
  38 FM 39 CD
  40 TAPE 41 VIDEO
  42 AUX
  For Fnc=FNC_TPKEY
  State bit 5-0 = Key number
  State bit 6 = 1 Key down, 0 Key Up
  State bit 7 = 1 Remote Audio Key, 0 normal
  For All other
  State = 0 OR 255 (or. 1) = OFF or ON

- `public delegate void WriteEventHandler(Tds_Port sender, int Fnc, int Number, int State);`

  This event occurs after a WriteDisplayMessage was processed by the MICROS central unit.

  Param:
  - Fnc (FNC_MESSAGE)
  - Number (the Number of the message)
  - State (1 if the message was Written, 0 if the message was not writter)

1.2 Included examples

The examples included are build on the Microsoft .Net framework 2.0. This framework is included on your TELETASK V2.80 CD-ROM an if necessary it will be installed when installing PROSOFT.

The program sources are included on your CD-ROM and have to be opened in Visual Studio 2005.

If you need support, it will be on the level of your c# compiler. If you need support concerning the TELETASK softtools, this will be charged at normal cost. For more information please contact your local distributor.

1.2.1 TdsComport Example

To be able to test this demo program, your pc has to be connected with the TDS by means of a standard serial port. After starting up the program you must enter a valid port number and choose the appropriate baud rate (default baud rate is 19200).

On the screen you will find several buttons:

“Log Function” : For every function you can activate the log with this function
“Get Function” : A request to get the actual state of the function, a bit later you will receive a event
“Set Function” : Activate the function

All the received messages are put in a list.

REMARK: There are some actions(commands) which are not able to be controlled in an OFF status or which have no result. Please keep in mind these (logic) features of the system.

1.2.2 TdsSocket Example

For this demo program your pc has to be connected to a MICROS central unit over Ethernet or over a RS232 connection. The ethernet connection can by made by means of the TDS10118, Ethernet to RS232 convertor (For more information about this product or to purchase it please contact your local TELETASK distributor).

This demo allows you to connect to two MICROS central units (Only one for RS232 communication), and to activate a dynamic message on one or both the central units.

The Socket information (IP-address and port number) for each of the ethernet connections and for the serial port can be set in the top part of the Form.
In the Center of the Form you can enter the AUTOBUS address of the interface you want to display the message on and the two lines of text for the message.
At the bottom there are four buttons, the first sends the message to MICROS A and if the message was successfully written it will activate the message. The third button does the same but to MICROS B and the middle button will simultaneously handle the writing and the activation of the message on both the MICROS central unit (the two central units need to have display interface on the same address). The button labelled "Send using COM" will do the same over the RS232 connection

2 Constants

The functions in the DLL use a parameter *"Fnc"* and can have following values

FNC_RELAY = 1	(control or get the status of a relay)
FNC_DIMMER = 2	(control or get the status of a dimmer)
FNC_MOTOR = 6	(control or get the status of a Motor: On/Off)
FNC_MTRUPDOWN = 55	(control or get the status of a Motor: Op/Down)
FNC_LOCMOOD = 8	(control or get the status of a Local Mood)
FNC_TIMEDMOOD = 9	(control or get the status of a Timed Local Mood)
FNC_GENMOOD = 10	(control or get the status of a General Mood)
FNC_FLAG = 15	(control or get the status of a Flag)
FNC_PROCES = 3	(control or get the status of a Process function)
FNC_REGIME = 14	(control or get the status of a Regime function)
FNC_SERVICE = 53	(control or get the status of a Service function)
FNC_MESSAGE = 54	(control or get the status of a Messages or Alarms)
FNC_COND = 60	(get the status of a Condition)
FNC_TPKEY = 52	(simulate a key press on an interface)
FNC_GETSENSTARGET = 21	(get the status of a Sensor setting)
FNC_GETSENSVAL = 20	(get the status of a Sensor value)
FNC_SETTEMPUP = 21	(control the Sensor Function: Up)
FNC_SETTEMPDOWN = 22	(control the Sensor Function: Down)
FNC_SETTEMPFROST = 24	(control the Sensor Function: Go to frost)
FNC_SETTEMPNIGHT = 25	(control the Sensor Function: Go to night)
FNC_SETTEMPDAY = 26	(control the Sensor Function: Go to day)
FNC_SETTEMPSETNIGHT = 27	(control the Sensor Function: Set night preset)
FNC_SETTEMPSETDAY = 29	(control the Sensor Function: Set day preset)
FNC_AUDIOSTATE = 31	(get status of an Audio Zone)
FNC_AUDIOUP = 32	(Control Audio Function: Vol. Up)
FNC_AUDIODOWN = 33	(Control Audio Function: Vol. Down)
FNC_AUDIOON = 36	(Control Audio Function: On)
FNC_AUDIOOFF = 37	(Control Audio Function: Off)
FNC_AUDIOFM = 38	(Control Audio Function: Tuner)
FNC_AUDIOFM2 = 47	(Control Audio Function: Tuner Extra)
FNC_AUDIOCD = 39	(Control Audio Function: CD)
FNC_AUDIOCD2 = 48	(Control Audio Function: CD Extra)
FNC_AUDIOTAPE = 40	(Control Audio Function: Tape)
FNC_AUDIOTAPE2 = 49	(Control Audio Function: Tape Extra)
FNC_AUDIOVIDEO = 41	(Control Audio Function: Video)
FNC_AUDIOVIDEO2 = 43	(Control Audio Function: Video Extra)
FNC_AUDIOAUX = 42	(Control Audio Function: Aux)
FNC_AUDIOAUX2 = 35	(Control Audio Function: Aux Extra)
FNC_AUDIOKEY = 44	(Control Audio Function: Key)

3 Communication

This section describes what you have to know if you write your own interface but you won't use the DLL. If you use the DLL you don't have to read this section.

The TDS CENTRAL is connected to a CCT (custom command terminal), which has to be operated by the user and can send commands to the TDS CENTRAL. The communication between the CCT and the TDS CENTRAL is a standard RS-232 interface. The cable will be connected to the TDS CENTRAL by means of a 9 pin female SubD type connector. The interface will use only 3 of the 9 pins:

Pin #	Description
3	Transmit data
2	Receive data
5	Signal Ground

This interface protocol will have 1 stop bit and no parity by default. Possible baud rates are: 2400, 4800, 9600 & 19200 (Default)

All commands and messages in both directions will use the same frame format:

STX (02h)	Length	Command Number	Parameter 1		Parameter n	ChkSm

The length does not include the ChkSm-byte.
The ChkSm is calculated on Command Number + Command Parameters + Length + STX
After the ChkSm the central unit send an acknowledge byte 0A (hex). If no acknowledge byte is send the command is not handled.

Command details

Function Set

- Description: This command allows the CCT to set individual functions. See "methods" for detailed descriptions
- Command number: 01h
- Length: 6
- Direction: From TDS to CCT.
- Parameter 1 = Fnc
- Parameter 2 = Outp
- Parameter 3 = State

Function Get

- Description: When the TDS receives this command it reports the level of the specified load. See methods for detailed descriptions
- Command number: 02h
- Length: 5
- Direction: From CCT to TDS
- Parameter 1 = fnc
- Parameter 2 = Outp

Function Log On/Off

- Description: When the TDS receives this command it (de-)activates it's channel for reporting the function!
- Command number: 03h
- Length: 5
- Direction: From CCT to TDS

- Parameter 1 = fnc
- Parameter 2 = state

Event Report

- Description: TDS sends this command it the level of the specified load has changed
- Command number: 08h
- Length: 5
- Direction: From TDS to CCT
- Parameter 1 = Fnc
- Parameter 2 = Outp
- Parameter 3 = State

Sensor values

Temperatures

To change from byte to °C: (byte/2)-40
To change from °C to byte: (temp+40)x2

Humidity

Byte = % humidity

Lux values

To change from byte to lux = $(10^{(byte / 40)}) - 1$
To change from lux to byte = Log10(lux + 1) * 40

CHANGES since Version 2.73

- WriteDisplayMessage
- Option byte
- Changed from vb OCX naar C# DLL
