Software Documentation V1.09
for
CVMLIB!pro
V02.11

Win2000/XP - DLL
for Cell Voltage Monitoring System CVMS

SMART Electronic Development GmbH • Hölzelweg 2 • D-70191 Stuttgart • Phone: ++49 (0) 711/25521-0 • Fax: ++49 (0) 711/25521-10 • Internet: http://www.smart-gmbh.de

Contents

1 Brief description

The CVMLIB!pro – DLL is the interface between the SMART cell voltage monitoring system CVMS and user applications on a PC. The user can access all important functions through this DLL.

A graphic user interface, "CVMView!pro", can be obtained as standard for configuration of the CVM units and visualisation of the measuring data. This front-end, coupled with the relevant card drivers of the SMART PCC-CAN-cards, enables easy communication through the internal CAN-Bus (iCAN) of the CVMS units. All data access is encapsulated and cannot be viewed by the user within the driver layers.

Windows 2000/XP
Operation with 1 CAN interface card (e.g. USB or PCMCIA) or 2 cards (of same type) is possible.

The following description has been intentionally kept simple. For a quick start, you'll find a simple test program in the DLL's installation path.

2 System requirements, licencing

CVMLIB!pro-V2 requires Windows 2000 or XP. We assume that users are familiar with integrating external DLLs into their application and have appropriate programming knowledge.

SMART recommends at least a Pentium4, 1 Ghz processor with 256MB of RAM and about 5 MB of free disk space.

The CVMLIB!pro-DLL communicates via the SMART PCCOMLIB library which is protected by a code protection based on the serial number of the CAN interface card.

Together with the shipment, you have received a CDROM containing the latest licence file "pccomls.dll". Replace the existing DLL under "C:\Program Files\SMART\Pccomlib\" with the newer file from the CDROM.
If you have any questions or problems, please contact SMART Electronic Development GmbH (please also indicate the serial number and type of your CAN interface card !)

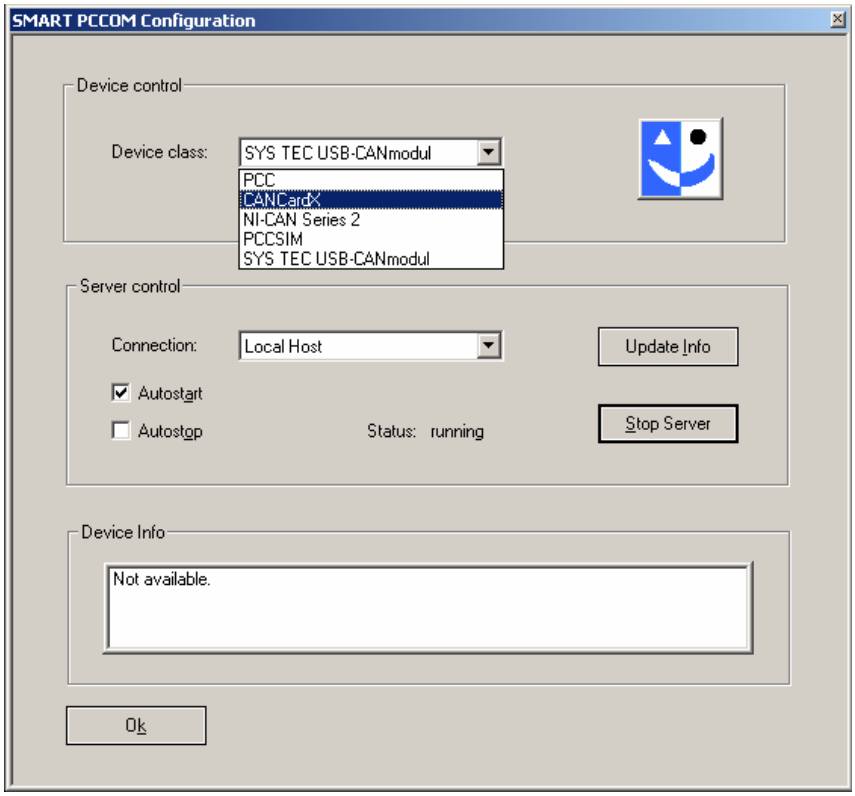
3 Selection of the interface card

CVMLIB!pro-V2 supports operation with different CAN interface cards:

- PCC: SMART interface cards PCCom-PCI and PCCom-PCMCIA
- CANCardX: interface cards of Vector Informatik (CANcardXL only !)
- NI-CAN Series2: interface cards of National Instruments
- Systec USB-CANmodul: USB interface cards of Systec (can be ordered via SMART)

After installing the interface card according to the manufacturer's instructions, you can select the desired interface card using the "SMART PCCOM Configuration Tool". You will find it under

START ⇒ Programs ⇒ SMART ⇒ PCCOMLIB ⇒ PCCOM Config Tool.



To select an interface card ("device") follow the steps:

- stop the PCCOM server („Stop Server“)
- select the card from the dropdown list
- restart the PCCOM server („Start Server“).

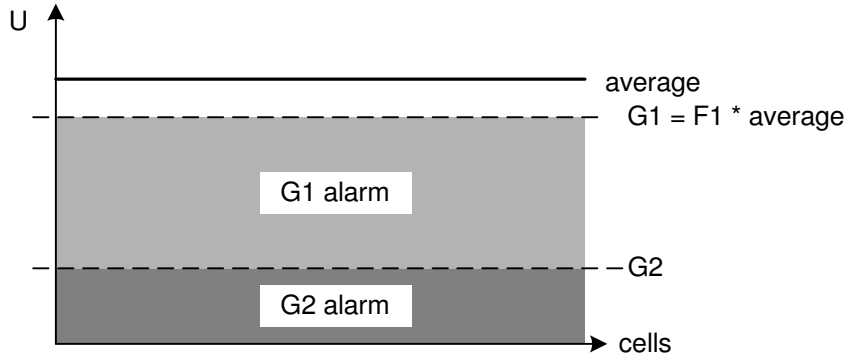
4 Terms and abbreviations used

Term	Meaning
CAN	Controller Area Network / Two-wire bus for controller communication
CVM	Cell Voltage Monitoring
FC	Full Cells
iCAN	Internal CAN bus
line	All stacks on the same CAN bus form a "line"
nd	not defined or not required
PCC	PC-Card, CAN interface card: SMART PCCom-PCI or –PCMCIA, Systec CAN-USB-interface, NiCAN card, Vector CANcardXL
stack	Component composed of up to 180 individual fuel cells
tbd	to be defined
UAU	Universal Adapter Unit. Flexible adapter units to enhance input voltage range.
UZUE	Old designation for CVM
VI	Virtual instrument (necessary for intergration with LabView)

Limit values:

The CVM system has two limit values: G1 and G2.
G1 is dynamically computed using a factor F1 and the average voltage of a stack. G2 is an absolute limit. The following rules apply:

G2 ≤ cell voltage < G1	cell is reported as "G1 alarm"	(1)
0 < cell voltage < G2	cell is reported as "G2 alarm"	(2)



Average voltage, total voltage:

The average value (with GetStackValues) is calculated from the total voltage (in V) and the number of cells. The total voltage is rounded. If you need a higher accuracy, your application should re-calculate the total voltage using the individual cell voltages returned by GetCellValues().

5 Headerfile for CVMLIBpro - DLL (V02.11)

The following is only a list of the exported functions of the DLL. The complete header file is part of the shipment.

```
// DLL functions

int  EXPORT_DECL __stdcall Init(unsigned long ulPccNo);
int  EXPORT_DECL __stdcall Exit(unsigned long ulPccNo);
int  EXPORT_DECL __stdcall BusScan(unsigned long ulPccNo);
int  EXPORT_DECL __stdcall GetConfig(int* paiConfig, int* paiSeriesNo, unsigned long ulPccNo);
int  EXPORT_DECL __stdcall SetConfig(int* paiConfig, int* paiSeriesNo, unsigned long ulPccNo);
int  EXPORT_DECL __stdcall SetConfigPassive(int* paiConfig, int* paiSeriesNo,
                                            unsigned long ulPccNo);
int  EXPORT_DECL __stdcall GetStackValues(int iStackNo, int iNumberOfCells,
                                            int* paiStackValues, unsigned long ulPccNo);
int  EXPORT_DECL __stdcall StartStack(int iStackIdx, int iStackNo, unsigned long ulPccNo);
int  EXPORT_DECL __stdcall StopStack(int iStackIdx, int iStackNo, unsigned long ulPccNo);
int  EXPORT_DECL __stdcall GetCellValues(int iStackIdx, int iNumberOfCells, int* paiUcell,
                                            int* paiFlags, unsigned long ulPccNo);
int  EXPORT_DECL __stdcall SetLimitValue(int iF1, int iG2, unsigned long ulPccNo);
int  EXPORT_DECL __stdcall GetLimitValue(int* piF1, int* piG2, unsigned long ulPccNo);
int  EXPORT_DECL __stdcall SetCVMMode(int iPolarity, int iNCell, int iStackNo,
                                            unsigned long ulPccNo);
int  EXPORT_DECL __stdcall GetCVMMode(int *piPolarity, int *piNCell, int iStackNo,
                                            unsigned long ulPccNo);
int  EXPORT_DECL __stdcall SetCellMask(unsigned long *pulCellMask, int iStackNo,
                                            unsigned long ulPccNo);
int  EXPORT_DECL __stdcall GetCellMask(unsigned long *pulCellMask, int iStackNo,
                                            unsigned long ulPccNo);
int  EXPORT_DECL __stdcall SetECANMsgID(int iTxId, int iRxId, unsigned long ulPccNo);
int  EXPORT_DECL __stdcall GetDLLVersion(char* pszVersion);
int  EXPORT_DECL __stdcall GetPccomlibVersion(char* pszVersion);
long EXPORT_DECL __stdcall GetLastErrNo(unsigned long ulPccNo);
int  EXPORT_DECL __stdcall GetLastErrTxt(char *pcDest, int iLen, unsigned long ulPccNo);
```

Recommendations for your software design:

The main task of CVMLIB!pro-DLL is to collect all information sent by the CVM units and maintain a complete set that can be queried by your application.
Therefore, avoid redundant data structures in your own application. Use the Get() functions of CVMLIB!pro-DLL to retrieve information whenever you need it.

6 DLL functions

Overview of functions

1. **Initialisation**

Init	-- Initialises resources of the operating system and underlying drivers and checks the licence (Exit() is the pendant to deinitialize the DLL)

2. **Read and write configuration**

BusScan	-- Determines the number, serial numbers and properties of all CVM units connected to the iCAN
GetConfig	-- Returns the system configuration as reported by the CVM units
SetConfig	-- Writes the configuration and starts the CAN communication
SetConfigPassive	-- **nop**

3. **Reading fuel cell stack values**

GetStackValues	-- Reads the current fuel cell stack values such as total voltage, min and max values.

4. **Reading individual cell values**

StartStack	-- Starts transmission of individual cell values
GetCellValues	-- Reads the current individual cell values of a stack
StopStack	-- Stops transmission of individual cell values

5. **Limits and operation mode**

GetLimitValue	-- Returns the current limit values F1 and G2
SetLimitValue	-- Sets new limit values
GetCVMMode	-- Returns current polarity and mode (single/double cell mode)
SetCVMMode	-- Sets new polarity and mode (single/double cell mode)
GetCellMask	-- Returns the current cell mask (visibility mask)
SetCellMask	-- Sets a new cell mask
SetECANMsgID	-- Sets new CAN Identifiers for eCAN messages

6. **Error handling**

GetLastErrNo	-- Returns the number of the last error
GetLastErrTxt	-- Returns an error text for the last error

7. **Version info**

GetDLLVersion	-- Returns the DLL version string
GetPccomlibVersion	-- Auslesen des Versionsstrings der verwendeten Kommunikations-DLL (PCCOMLIB)

8. **Shutdown**

Exit	-- Deinitializes the DLL