ADLINK SEMA API Programmer Reference Guide

User's Manual

Manual Rev.: 1.0

Revision Date: Oct 24, 2013

Part No.:

Revision History

Revision	Date	Changes
0.01	2013/09/02	β Release
1.0	2013/10/24	Release

Disclaimer

Trademark Information

Conventions

Take note of the following conventions used throughout this manual to make sure that tasks and instructions are performed properly.

Additional information, aids, and tips that help perform specific tasks.

Critical information that users MUST know and instructions that users MUST perform to complete a task.

Information to prevent physical injury, data loss, component damage, program corruption, etc. when trying to complete a specific task.

Using this Manual

Audience and Scope

This manual provides information on accessing and operating low level functions in ADLINK's range of Computer on Modules using a high level API function library. This manual is intended for software programmers with knowledge of Computer-On-Modules (COM).

How this manual is organized

This manual is organized as follows:

Chapter 1, Introduction to SEMA Library: Introduces the SEMA Library, its features and supporting hardware and software environments.

Chapter 2, Installation: Provides information on installing the SEMA Library under different operating systems.

Chapter 3, SEMA Library Programmer's Reference: Detailed API instruction set functions and format

Table of Contents

1 Introduction to SEMA Library

1.1 About the SEMA Library

The SEMA (ADLINK Smart Embedded Management Agent) Library is a software library that enables users to access low level hardware features of ADLINK Computer-on-Modules.

1.2 OS Support

Currently, the following operating systems are supported:

- Windows (win32/64)
  - Microsoft® Windows® 7
- Linux® (3.2.x)

2 OS Support and Installation

All API functions are exported from the `LibSEMA.DLL` dynamic link library and UNICODE is supported. The `LibSEMA.DLL` is binary compatible between Windows® 7. There may be different versions with the same name that support other operating systems.

2.1 Installation for Windows (Win32/64)

The SEMA library contains a Windows® kernel service and library. The library under Windows is a DLL (Dynamic-Link Library) that can be used under Windows® 7. It can work with any Windows® programming language that allows calls to a DLL, such as Microsoft® Visual C/C++® (2008 or above).

Double click `SEMA.msi` in the SEMA Library software package. The installation execution file `SEMA.msi` will guide you to setup the SEMA Library.

NOTE:

After installation, you must restart Windows® to activate the Windows® kernel service.

When the installation process is completed, the installation directory should contain the following files and sub-directories:

File/Sub-directory	Desc
Program Files\Sema<DIR>	Includes Libsema.dll for application programming
WINDOWS\SYSTEM32\Drivers\<DIR>	SYS file

After installing the SEMA Library, the kernel service is entered into the system registry and the related driver files are copied to their appropriate directories. With the registry and driver files (.SYS), Windows® 7 will be able to initiate drivers automatically at system startup.

2.2 Installation for Linux®

A separate software bundle is available with a pre-compiled copy of the SEMA Library for the following Linux® kernels:

► 3.2

The Linux® SEMA Library package in the SEMA_Library directory contains the following files/directories:

```
libsema
|-- CMakeLists.txt
|-- Conv.c
|-- Conv.h
|-- Debug.c
|-- Debug.h
|-- DLL_Exports.h
|-- dmi_info.c
|-- dmi_info.h
|-- ErrorCodes.h
|-- Globals.h
|-- linux
|   |-- SMBus_Linux.c
|   `-- SMBus_Linux.h
|-- Makefile
|-- native.h
|-- resource.h
|-- SemaAdmin.c
|-- Sema.c
|-- Sema.h
|-- SemaLog.c
|-- SemaLog.h
|-- SemaSimulation.h
|-- SemaUser.c
|-- SemaVersion.h
|-- win32
|   |-- SMBus_win32.c
|   `-- SMBus_win32.h
|-- WinCE.c
`-- WinCE.h
2 directories, 27 files
```

2.2.1 Prerequisites

In order to build and install the Linux® SEMA Library and interface library, Cmake in version 2.6 or higher and QT Library at least Version 4.6 are required.

2.2.2 Build and Installation

- Extract the source zip file, sema.zip and enter the directory build.
- Run command line, cmake, will generate a Makefile for Linux.
- Finally run make will create executable files and shared library.

$ cd sema

$ cd build // If there is no folder build, then create it. ($ mkdir build)

$ cmake .. // This will generate a Makefile for Linux.

$ make

If you only need the sema commandline tool for linux, the makefile in the libsema directory can be used without the need for cmake and Qt.

Simply run make in the libsema directory.

Directory	File(s)	Description
driver/linux/	libsema_hwlinux.so	
gui/	semagui	General SEMA Library test application.
libsema/	liblibsemaadmin.so liblibsema.so	SEMA Library Watchdog test application.
sema/	Sema Semaadmin	
semaweb/	Semaweb	

The SEMA Library will be copied to the appropriate directories:

File(s)	Directories
libsema_hwlinux.so liblibsemaadmin.so liblibsema.so	/usr/local/lib
sema semaadmin semagui semaweb	/usr/local/bin

3 Library Programmer's Reference

The SEMA Library provides control and access to specific board functions of ADLINK Computer-on-Modules. The API programming interface is compatible and identical across all ADLINK boards and all supported operating systems. This section presents detailed functionality for programmers' reference.


NOTE:

Not all functions listed below are supported by all ADLINK modules or by the SEMA Library. Please refer to the `ReleaseNote.txt` file included with your version of the SEMA Library for detailed information.

3.1 DLL Initialization Functions

Before using the SEMA Library, you must initialize the DLL by activating the `Sema_Init()` function. It is also necessary to call the `Sema_Close()` function to properly clear system resources before terminating the application.

3.1.1 Sema_Init

► ***Description***

`Initializes SMBus access and queries BMC capabilities`

► ***Syntax***

```
DLL_EXPORT int Sema_Init(void);
```

► ***Returns***

Ture if requested number of bytes could be read

3.1.2 Sema_Close

► ***Description***

Cleans up

► ***Syntax***

```
DLL_EXPORT void Sema_Close(void);
```

► ***Returns***

3.2 Board Information Functions

SEMA library functions provide an interface to control or get board information. Using the `Sema_GetChipsetID` function gets the board handle which is the unique identification number of a board. Other functions need that handle to identify boards to perform appropriate operations.

3.2.1 Sema_GetChipsetID

► ***Description***

Get chipset ID (Vendor+Device ID).

► ***Syntax***

```
DLL_EXPORT dword Sema_GetChipsetID(void);
```

► ***Parameters***

► ***Returns***

Vendor +Device ID

3.2.2 *Sema_GetChipsetString*

► ***Description***

Gets chipset IDs string .

► ***Syntax***

```
DLL_EXPORT const char* Sema_GetChipsetString(void);
```

► ***Parameters***

► ***Returns***

Pointer to chipset ID string

3.2.3 *Sema_ClearExceptionCode*

► ***Description***

Clears BMC exception code

► ***Syntax***

```
DLL_EXPORT void Sema_ClearExceptionCode(void);
```

► ***Parameters***

► ***Returns***

3.2.4 *Sema_GetVersionString1*

► ***Description***

Get version string #1

► ***Syntax***

```
DLL_EXPORT void Sema_GetVersionString1(byte *Buffer);
```

► ***Parameters***

```
*Buffer        Buffer (at least 33 bytes!!) for version string
```

► ***Returns***

3.2.5 *Sema_GetVersionString2*

► ***Description***

Get version string #2

► ***Syntax***

```
DLL_EXPORT void Sema_GetVersionString2(byte *Buffer);
```

► ***Parameters***

```
*Buffer         Buffer (at least 33 bytes!!) for version string
```

► ***Returns***

3.2.6 *Sema_GetAppVersion*

► ***Description***

Get BMC application version string

► ***Syntax***

```
DLL_EXPORT bool Sema_GetAppVersion(byte *version);
```

► ***Parameters***

```
*Buffer         Buffer (at least 65 bytes!!) for BMC application
version string
```

► ***Returns***

3.2.7 *Sema_SelectBIOS*

► ***Description***

Select BIOS (only available if Faile-Safe-BIOS is supported)

► ***Syntax***

```
DLL_EXPORT bool Sema_SelectBIOS(byte Index);
```

► ***Parameters***

```
Index           BIOS number/index (0 or 1)
```

► ***Returns***

true if successful, otherwise false.

3.2.8 Sema_GetBIOS

► **Description**

Gets selected BIOS (only available if Faile-Safe-BIOS is supported)

► **Syntax**

```
DLL_EXPORT byte Sema_GetBIOS(void);
```

► **Parameters**

► **Returns**

BIOS number/index (0:FS-BIOS not supported or normal BIOS selected 1: FS-BIOS selected)

3.2.9 Sema_GetBootVersion

► **Description**

Get BMC bootloader version string

► **Syntax**

```
DLL_EXPORT bool Sema_GetBootVersion(byte *Buffer);
```

► **Parameters**

```
*Buffer        Buffer for BMC bootloader version string
```

► **Returns**

True if successful, otherwise false

3.2.10 Sema_GetMemSize

► **Description**

Get memory size (currently only implemented for CRR-945GSE)

► **Syntax**

```
DLL_EXPORT unsigned char Sema_GetMemSize(void);
```

► **Parameters**

► **Returns**

1:512MB, 2:1GB, 3:2GB.

3.2.11 Sema_GetTotalUpTime

► ***Description***

Get total uptime [minutes]

► ***Syntax***

```
DLL_EXPORT unsigned long Sema_GetTotalUpTime(void);
```

► ***Parameters***

► ***Returns***

Total uptime [minutes].

3.2.12 Sema_GetPwrUpTime

► ***Description***

Get power uptime [seconds]

► ***Syntax***

```
DLL_EXPORT unsigned long Sema_GetPwrUpTime(void);
```

► ***Parameters***

► ***Returns***

Uptime since last boot [seconds]

3.2.13 Sema_GetPwrCycles

► ***Description***

Get number of power cycles

► ***Syntax***

```
DLL_EXPORT unsigned long Sema_GetPwrCycles(void);
```

► ***Parameters***

► ***Returns***

Number of power cycles

3.2.14 Sema_GetBootCount

► ***Description***
Get number of boots

► ***Syntax***

```
DLL_EXPORT unsigned long Sema_GetBootCount(void);
```

► ***Parameters***

► ***Returns***
Number of boots

3.2.15 Sema_GetBMCFlags

► ***Description***
Get BMC flags

► ***Syntax***

```
DLL_EXPORT unsigned char Sema_GetBMCFlags(void);
```

► ***Parameters***

► ***Returns***
BMC flags

3.2.16 Sema_GetRestartEvent

► ***Description***
Get last system restart event

► ***Syntax***

```
DLL_EXPORT unsigned char Sema_GetRestartEvent(void);
```

► ***Parameters***

► ***Returns***
Restart event code

3.2.17 Sema_GetRestartEventText

► ***Description***
Get last system restart event

► ***Syntax***

```
DLL_EXPORT const char* Sema_GetRestartEventText(void);
```

► ***Parameters***

► ***Returns***
Restart event description text

3.2.18 Sema_PrintCapabilities

► ***Description***
Print BMC capabilities

► ***Syntax***

```
DLL_EXPORT void Sema_PrintCapabilities(void);
```

► ***Parameters***

► ***Returns***

3.2.19 Sema_GetCapabilities

► ***Description***
Returns BMC capabilities (does not perform any tests!)

► ***Syntax***

```
DLL_EXPORT tSEMACapabilities Sema_GetCapabilities(void);
```

► ***Parameters***

► ***Returns***
Capabilities of BMC

3.2.20 Sema_OverrideCapabilities

► ***Description***
Overrides BMC capabilities (USE WITH CARE!!!)

► ***Syntax***

```
DLL_EXPORT void Sema_OverrideCapabilities(tSEMACapabilities
Cap);
```

► ***Parameters***

```
Cap  Capabilities of BMC
```

► ***Returns***

3.2.21 Sema_GetSysCtrlReg

► ***Description***
Get current status of system control register
Note: First byte of I2C data block is interpreted as most significant byte

► ***Syntax***

```
DLL_EXPORT unsigned short Sema_GetSysCtrlReg(void);
```

► ***Parameters***

► ***Returns***

```
Current status of system control register
```

3.2.22 Sema_SetSysCtrlReg

► ***Description***
Set new status of system control register
Note: Most significant byte will be transferred first

► ***Syntax***

```
DLL_EXPORT void Sema_SetSysCtrlReg(unsigned short val);
```

► ***Parameters***

```
Val  New status of system control register
```

► ***Returns***

3.2.23 Sema_Prog

► ***Description***
Programs file content to specified memory address

► ***Syntax***

```
DLL_EXPORT int Sema_Prog(char *key, word address, char
*filename);
```

► ***Parameters***

► ***Returns***

3.2.24 Sema_Update

► ***Description***
Updates BMC

► ***Syntax***

```
DLL_EXPORT int Sema_Update(char *filename);
```

► ***Parameters***

► ***Returns***

3.2.25 Sema_GetManufacturingData

► ***Description***
Get the Manufacturing Date select by the Data parameter

► ***Syntax***

```
DLL_EXPORT void Sema_GetManufacturingData(byte *Buffer, byte
Data);
```

► ***Parameters***

```
*Buffer          Buffer for part number
Data            Manufacturing data index
```

► ***Returns***

3.2.26 Sema_SetBMCAddress

► ***Description***
Get the Manufacturing Date select by the Data parameter

► ***Syntax***
```
DLL_EXPORT void Sema_SetBMCAddress(byte Address);
```

► ***Parameters***

► ***Returns***
Sets BMC SMBus address

3.2.27 Sema_GetLibraryMajor

► ***Description***
Get the Manufacturing Date select by the Data parameter

► ***Syntax***
```
DLL_EXPORT word Sema_GetLibraryMajor(void);
```

► ***Parameters***

► ***Returns***
Returns major library version number

3.2.28 Sema_GetLibraryMinor

► ***Description***

► ***Syntax***
```
DLL_EXPORT word Sema_GetLibraryMinor(void);
```

► ***Parameters***

► ***Returns***
Returns minor library version number

3.2.29 Sema_GetLibraryAddon

► ***Description***

► ***Syntax***

```
DLL_EXPORT const char *Sema_GetLibraryAddon(void);
```

► ***Parameters***

► ***Returns***

Returns the library version add-on string

3.2.30 Sema_GetHardwareAddon

► ***Description***

► ***Syntax***

```
DLL_EXPORT const char* Sema_GetHardwareAddon(void);
```

► ***Parameters***

► ***Returns***

Returns the library version add-on string

3.2.31 Sema_GetHardwareMajor

► ***Description***

► ***Syntax***

```
DLL_EXPORT word Sema_GetHardwareMajor(void);
```

► ***Parameters***

► ***Returns***

Returns major hardware driver/library version number

3.2.32 Sema_GetHardwareMinor

► ***Description***

► ***Syntax***

```
DLL_EXPORT word Sema_GetHardwareMinor(void);
```

► ***Parameters***

► ***Returns***

Returns minor hardware driver/library version number

3.2.33 Sema_IsSimulationMode

► ***Description***

If there is no BMC detected, the SEMA application will run in DEMO mode with dummy values. This is used for presentation or show cases.

► ***Syntax***

```
DLL_EXPORT bool Sema_IsSimulationMode(void);
```

► ***Parameters***

► ***Returns***

Returns true if lib is in demo mode (no hw detected!)

3.3 Storage Areas Functions

Each board can have several storage areas. A storage area is a piece of physical memory that usually provides persistent storage for the user's application(s). The only storage currently supported is EEPROM. Most storage area functions have a type as a second parameter which is a combination of one of the predefined type constants for EEPROM, FLASH, CMOS, or RAM and a zero-based index of the area.

Check what address the SPD memory is using before accessing EEPROM or it will cause system errors.

3.3.1 Sema_MemoryRead

▶ ***Description***

Read from flash memory

▶ ***Syntax***

```
DLL_EXPORT bool Sema_MemoryRead(byte *Buffer, unsigned short
Adr, unsigned char Size);
```

▶ ***Parameters***

```
*Buffer         Buffer for read bytes
Adr             Start address
Size            Number of bytes to read
```

▶ ***Returns***

True if requested number of bytes could be read

3.3.2 *Sema_MemoryWrite*

▶ ***Description***

Writes to flash memory

▶ ***Syntax***

```
DLL_EXPORT void Sema_MemoryWrite(byte *Buffer, unsigned short
Adr, unsigned char Size);
```

▶ ***Parameters***

*Buffer	Buffer for bytes to write
Adr	Start address
Size	Number of bytes to read

▶ ***Returns***

3.3.3 *Sema_MemoryClear*

▶ ***Description***

Clears user flash memory

▶ ***Syntax***

```
DLL_EXPORT void Sema_MemoryClear(void);
```

▶ ***Parameters***

▶ ***Returns***

3.3.4 *Sema_MemoryRestore*

▶ ***Description***

Restores first 64 bytes of user flash memory (partNo, serial, etc)

▶ ***Syntax***

```
DLL_EXPORT void Sema_MemoryRestore(void);
```

▶ ***Parameters***

▶ ***Returns***

3.3.5 *Sema_IsValidAddress*

▶ ***Description***

Tests if address is within valid range

▶ ***Syntax***

```
DLL_EXPORT bool Sema_IsValidAddress(word Adr);
```

▶ ***Parameters***

▶ ***Returns***

3.3.6 *Sema_AdjustSize*

▶ ***Description***

Adjusts size to valid values (boundaries, max length, ...)

▶ ***Syntax***

```
DLL_EXPORT void Sema_AdjustSize(word *size, word address);
```

▶ ***Parameters***

```
*size           number of bytes
address         Start address
```

▶ ***Returns***

3.3.7 *Sema_PrintMem*

▶ ***Description***

Prints memory contents to formatted string

▶ ***Syntax***

```
DLL_EXPORT void Sema_PrintMem(word address, word size, char
*output);
```

▶ ***Parameters***

```
address         Start address
size            number of bytes
*output         Pointer to output buffer
```

▶ ***Returns***

3.4 I2C Bus Functions

The SEMA I2C functions provide an interface to access the onboard I2C or SMBus. You can use these functions to access any I2C device.

Use caution when accessing I2C devices.

3.4.1 Sema_Trans

► ***Description***

Initiates Raw I2C transfer

► ***Syntax***

```
DLL_EXPORT byte Sema_Trans(byte Address, tTransType Type, byte
*Buffer);
```

► ***Parameters***

► ***Returns***

1: success, 0: fail

3.4.2 Sema_TransExt

► ***Description***

Initiates transfer on external I2C busses

► ***Syntax***

```
DLL_EXPORT byte Sema_TransExt(tTransType Type, byte Length,
byte *Buffer);
```

► ***Parameters***

```
Type      Transfer type
Length         Number of bytes transferred to BMC
Buffer         Data to transfer/receive
```

► ***Returns***

1: success, 0: fail

3.5 GPIO Functions

GPIO library support is limited to GPIO signals that originate from a COM (Computer on Module) and are extended to the carrier board. The COM Express specification allows 4 GPO and 4 GPI lines to extend to the carrier board, while the ETX specification does not include any GPIO signals. Therefore SEMA library's GPIO functions only relates to COM Express modules.

3.5.1 Sema_GetGPIOReg

► ***Description***

Get current status of GPIO register (only on 86DX platform)

► ***Syntax***

```
DLL_EXPORT unsigned char Sema_GetGPIOReg(void);
```

► ***Parameters***

► ***Returns***

Current status of GPIO register.

3.5.2 Sema_SetGPIOReg

► ***Description***

Set status of GPIO register (only on 86DX platform)

► ***Syntax***

```
DLL_EXPORT void Sema_SetGPIOReg(unsigned char GpioReg);
```

► ***Parameters***

GpioReg New status of GPIO register

► ***Returns***

3.5.3 *Sema_SetGPIODirection*

► ***Description***

Set GPIO direction of the I/O pins.

► ***Syntax***

```
DLL_EXPORT bool Sema_SetGpioDirection(unsigned int dwData);
```

► ***Parameters***

Value	New status of Configuration register

► ***Returns***

3.5.4 *Sema_GetGPIODirection*

► ***Description***

Get GPIO direction of the I/O pins.

► ***Syntax***

```
DLL_EXPORT bool Sema_GetGpioDirection(unsigned int *dwData);
```

► ***Parameters***

*Buffer	Pointer to current status of Configuration register

► ***Returns***

3.5.5 *Sema_GetGPIORead*

► ***Description***

Get GPIO Input of the I/O pins.

► ***Syntax***

```
DLL_EXPORT bool Sema_GetGpioRead(unsigned int *dwData);
```

► ***Parameters***

Value	Pointer to current status of Input register

► ***Returns***

3.5.6 *Sema_SetGPIOWrite*

► ***Description***

Get GPIO Output of the I/O pins.

► ***Syntax***

```
DLL_EXPORT bool Sema_SetGpioWrite(unsigned int dwData);
```

► ***Parameters***

*Buffer	New status of Output register

► ***Returns***

3.5.7 *Sema_SetGPIOXorAndXor*

► ***Description***

Set GPIO Output of the I/O pins

► ***Syntax***

```
DLL_EXPORT bool Sema_GpioXorAndXor(unsigned int Value1,
unsigned int Value2, unsigned int Value3);
```

► ***Parameters***

Value1	value to Xor the I/O pins
Value2	value to And the I/O pins
Value3	value to Xor the I/O pins

► ***Returns***

3.6 Watchdog Functions

The SEMA library Watchdog functions support Watchdog control of the board. If the Watchdog begins and reaches the preset time, it will access the CPU's RESET signal to reset the system. This application must call another function named `Sema_SetWatchdog` that resets the Watchdog to prevent system reset.

3.6.1 Sema_SetWatchdog

► ***Description***

Sets watchdog timer.

► ***Syntax***

```
DLL_EXPORT void Sema_SetWatchdog(unsigned short Value);
```

► ***Parameters***

Value Watchdog timeout in seconds (0 disables watchdog)

► ***Returns***

3.6.2 Sema_SetPwrUpWatchdog

► ***Description***

Sets watchdog timer.

► ***Syntax***

```
DLL_EXPORT void Sema_SetPwrUpWatchdog(unsigned short Value);
```

► ***Parameters***

Value Watchdog timeout in seconds (0 disables watchdog)

► ***Returns***

3.7 Hardware Monitor Functions

The SEMA library hardware monitor functions support temperature, fan, and voltage. These functions read the temperature, fan, and voltage status of the system. When the status exceeds the upper or lower limit parameters, the SEMA Library is able to issue an alarm.

3.7.1 Sema_GetFanSpeed

► ***Description***

Get fan speed (unit: rpm)

► ***Syntax***

```
DLL_EXPORT unsigned short Sema_GetFanSpeed(void);
```

► ***Parameters***

► ***Returns***

Fan speed

3.7.2 Sema_GetFanSpeedSystem

► ***Description***

Get fan speed (unit: rpm)

► ***Syntax***

```
DLL_EXPORT unsigned short Sema_GetFanSpeedSystem(void);
```

► ***Parameters***

► ***Returns***

Fan speed

3.7.3 *Sema_SmartFanGetTempSetpoints*

▶ ***Description***

SmartFan: Get temperature setpoints (signed, degree Celsius)

▶ ***Syntax***

```
DLL_EXPORT bool Sema_SmartFanGetTempSetpoints(signed char
*lvl1, signed char *lvl2, signed char *lvl3, signed char lvl4);
```

▶ ***Parameters***

```
lvl1 value for TEMP_LEVEL1
lvl2 value for TEMP_LEVEL2
lvl3 value for TEMP_LEVEL3
lvl4 value for TEMP_LEVEL4
```

▶ ***Returns***

true if operation was successful

3.7.4 *Sema_SmartFanGetPWMSetpoints*

▶ ***Description***

SmartFan: Get PWM setpoints (valid range: 0..100)

▶ ***Syntax***

```
DLL_EXPORT bool Sema_SmartFanGetPWMSetpoints(signed char *lvl1,
signed char *lvl2, signed char *lvl3, signed char *lvl4);
```

▶ ***Parameters***

```
lvl1 value for TEMP_LEVEL1
lvl2 value for TEMP_LEVEL2
lvl3 value for TEMP_LEVEL3
lvl4 value for TEMP_LEVEL4
```

▶ ***Returns***

true if operation was successful

3.7.5 Sema_SmartFanSetTempSetpoints

▶ ***Description***

SmartFan: Set temperature setpoints (signed, degree Celsius)

▶ ***Syntax***

```
DLL_EXPORT bool Sema_SmartFanSetTempSetpoints(signed char lvl1,
signed char lvl2, signed char lvl3, signed char lvl4);
```

▶ ***Parameters***

```
lvl1 value for TEMP_LEVEL1
lvl2 value for TEMP_LEVEL2
lvl3 value for TEMP_LEVEL3
lvl4 value for TEMP_LEVEL4
```

▶ ***Returns***

true if operation was successful

3.7.6 Sema_SmartFanSetPWMSetpoints

▶ ***Description***

SmartFan: Set PWM setpoints (valid range: 0..100)

▶ ***Syntax***

```
DLL_EXPORT bool Sema_SmartFanSetPWMSetpoints(signed char *lvl1,
signed char *lvl2, signed char *lvl3, signed char *lvl4);
```

▶ ***Parameters***

```
lvl1 value for TEMP_LEVEL1
lvl2 value for TEMP_LEVEL2
lvl3 value for TEMP_LEVEL3
lvl4 value for TEMP_LEVEL4
```

▶ ***Returns***

true if operation was successful

3.7.7 Sema_SmartFanGetTempSetpointsSystem

► ***Description***

SmartFan: Get temperature setpoints (signed, degree Celsius)

► ***Syntax***

```
DLL_EXPORT bool Sema_SmartFanGetTempSetpointsSystem(signed char
*lvl1, signed char *lvl2, signed char *lvl3, signed char
*lvl4);
```

► ***Parameters***

```
lvl1 value for TEMP_LEVEL1
lvl2 value for TEMP_LEVEL2
lvl3 value for TEMP_LEVEL3
lvl4 value for TEMP_LEVEL4
```

► ***Returns***

true if operation was successful

3.7.8 Sema_SmartFanGetPWMSetpointsSystem

► ***Description***

SmartFan: Get PWM setpoints (valid range: 0..100)

► ***Syntax***

```
DLL_EXPORT bool Sema_SmartFanGetPWMSetpointsSystem(unsigned
char *lvl1, unsigned char *lvl2, unsigned char *lvl3, unsigned
char *lvl4);
```

► ***Parameters***

```
lvl1 value for TEMP_LEVEL1
lvl2 value for TEMP_LEVEL2
lvl3 value for TEMP_LEVEL3
lvl4 value for TEMP_LEVEL4
```

► ***Returns***

true if operation was successful

3.7.9 Sema_SmartFanSetTempSetpointsSystem

► ***Description***

SmartFan: Set temperature setpoints (signed, degree Celsius)

► ***Syntax***

```
DLL_EXPORT bool Sema_SmartFanSetTempSetpointsSystem(signed char
*lvl1, signed char *lvl2, signed char *lvl3, signed char
*lvl4);
```

► ***Parameters***

```
lvl1 value for TEMP_LEVEL1
lvl2 value for TEMP_LEVEL2
lvl3 value for TEMP_LEVEL3
lvl4 value for TEMP_LEVEL4
```

► ***Returns***

true if operation was successful

3.7.10 Sema_SmartFanSetPWMSetpointsSystem

► ***Description***

SmartFan: Set PWM setpoints (valid range: 0..100)

► ***Syntax***

```
DLL_EXPORT bool Sema_SmartFanGetPWMSetpointsSystem(unsigned
char *lvl1, unsigned char *lvl2, unsigned char *lvl3, unsigned
char *lvl4);
```

► ***Parameters***

```
lvl1 value for TEMP_LEVEL1
lvl2 value for TEMP_LEVEL2
lvl3 value for TEMP_LEVEL3
lvl4 value for TEMP_LEVEL4
```

► ***Returns***

true if operation was successful

3.7.11 Sema_GetVoltage

► ***Description***

Get voltage of selected channel (unit: V, resolution=1/1000 V)

► ***Syntax***

```
DLL_EXPORT float Sema_GetVoltage(unsigned char Ch);
```

► ***Parameters***

► ***Returns***

voltage

3.7.12 Sema_GetVoltageDescription

► ***Description***

Get voltage of selected channel (unit: V, resolution=1/1000 V)

► ***Syntax***

```
DLL_EXPORT bool Sema_GetVoltageDescription(unsigned char Ch,
char *Buffer);
```

► ***Parameters***

► ***Returns***

voltage

3.7.13 Sema_GetMainPowerCurrent

► ***Description***

Get main power current (unit: mA, resolution=1/10 mA)

► ***Syntax***

```
DLL_EXPORT unsigned short Sema_GetMainPowerCurrent(void);
```

► ***Parameters***

► ***Returns***

main power current

3.7.14 Sema_GetCurrentCPUTemp

► ***Description***

Get current CPU temperature

► ***Syntax***

```
DLL_EXPORT float Sema_GetCurrentCPUTemp(void);
```

► ***Parameters***

► ***Returns***

Current CPU temperature (resolution=1/10 degC)

3.7.15 Sema_GetCurrentBoardTemp

► ***Description***

Get current Board temperature

► ***Syntax***

```
DLL_EXPORT float Sema_GetCurrentBoardTemp(void);
```

► ***Parameters***

► ***Returns***

Current Board temperature

3.7.16 Sema_GetMinMaxTemp

► ***Description***

Get min and max temperatures of CPU and board

► ***Syntax***

```
DLL_EXPORT void Sema_GetMinMaxTemp(signed char *MinCPU, signed
char *MaxCPU, signed char *MinBoard, signed char *MaxBoard);
```

► ***Parameters***

```
*MinCPU         Location for min CPU temperature
*MaxCPU         Location for max CPU temperature
*MinBoard       Location for min board temperature
*MaxBoard       Location for max board temperature
```

► ***Returns***

3.7.17 Sema_GetStartupTemp

► ***Description***

Get startup temperatures of CPU and board

► ***Syntax***

```
DLL_EXPORT void Sema_GetStartupTemp(signed char *StartCPU,
signed char *StartBoard);
```

► ***Parameters***

```
*StartCPU       Location for startup CPU temperature
*StartBoard     Location for startup board temperature
```

► ***Returns***

3.8 VGA Functions

The SEMA Library VGA functions support backlight brightness and backlight enable.

3.8.1 Sema_BacklightSetPWM

▶ ***Description***

Set new backlight PWM value (range: 0..255)

▶ ***Syntax***

```
DLL_EXPORT void Sema_BacklightSetPWM(byte Value);
```

▶ ***Parameters***

```
Value     new backlight PWM value (range: 0..255)
```

▶ ***Returns***

3.8.2 Sema_BacklightGetPWM

▶ ***Description***

Get current backlight PWM value (range: 0..255)

▶ ***Syntax***

```
DLL_EXPORT byte Sema_BacklightGetPWM(void);
```

▶ ***Parameters***

▶ ***Returns***

Current backlight PWM value

4 Getting Service

Contact us should you require any service or assistance.

ADLINK Technology, Inc.
Address: 9F, No.166 Jian Yi Road, Zhonghe District
New Taipei City 235, Taiwan
新北市中和區建一路 166 號 9 樓
Tel: +886-2-8226-5877
Fax: +886-2-8226-5717
Email: service@adlinktech.com

Ampro ADLINK Technology, Inc.
Address: 5215 Hellyer Avenue, #110, San Jose, CA 95138, USA
Tel: +1-408-360-0200
Toll Free: +1-800-966-5200 (USA only)
Fax: +1-408-360-0222
Email: info@adlinktech.com

ADLINK Technology (China) Co., Ltd.
Address: 上海市浦东新区张江高科技园区芳春路 300 号 (201203)
300 Fang Chun Rd., Zhangjiang Hi-Tech Park,
Pudong New Area, Shanghai, 201203 China
Tel: +86-21-5132-8988
Fax: +86-21-5132-3588
Email: market@adlinktech.com

ADLINK Technology Beijing
Address: 北京市海淀区上地东路 1 号盈创动力大厦 E 座 801 室(100085)
Rm. 801, Power Creative E, No. 1, B/D
Shang Di East Rd., Beijing, 100085 China
Tel: +86-10-5885-8666
Fax: +86-10-5885-8625
Email: market@adlinktech.com

ADLINK Technology Shenzhen
Address: 深圳市南山区科技园南区高新南七道 数字技术园 A1 栋 2 楼 C 区 (518057)
2F, C Block, Bldg. A1, Cyber-Tech Zone, Gao Xin Ave. Sec. 7,
High-Tech Industrial Park S., Shenzhen, 518054 China
Tel: +86-755-2643-4858
Fax: +86-755-2664-6353
Email: market@adlinktech.com

LiPPERT ADLINK Technology GmbH
Address: Hans-Thoma-Strasse 11, D-68163, Mannheim, Germany
Tel: +49-621-43214-0
Fax: +49-621 43214-30
Email: emea@adlinktech.com

ADLINK Technology, Inc. (French Liaison Office)
Address: 15 rue Emile Baudot, 91300 Massy CEDEX, France
Tel: +33 (0) 1 60 12 35 66
Fax: +33 (0) 1 60 12 35 66
Email: france@adlinktech.com

ADLINK Technology Japan Corporation
Address: 〒101-0045 東京都千代田区神田鍛冶町 3-7-4
神田 374 ビル 4F
KANDA374 Bldg. 4F, 3-7-4 Kanda Kajicho,
Chiyoda-ku, Tokyo 101-0045, Japan
Tel: +81-3-4455-3722
Fax: +81-3-5209-6013
Email: japan@adlinktech.com

ADLINK Technology, Inc. (Korean Liaison Office)
Address: 서울시 서초구 서초동 1675-12 모인터빌딩 8 층
8F Mointer B/D,1675-12, Seocho-Dong, Seocho-Gu,
Seoul 137-070, Korea
Tel: +82-2-2057-0565
Fax: +82-2-2057-0563
Email: korea@adlinktech.com

ADLINK Technology Singapore Pte. Ltd.
Address: 84 Genting Lane #07-02A, Cityneon Design Centre,
Singapore 349584
Tel: +65-6844-2261
Fax: +65-6844-2263
Email: singapore@adlinktech.com

ADLINK Technology Singapore Pte. Ltd. (Indian Liaison Office)
Address: 1st Floor, #50-56 (Between 16th/17th Cross) Margosa Plaza,
Margosa Main Road, Malleswaram, Bangalore-560055, India
Tel: +91-80-65605817, +91-80-42246107
Fax: +91-80-23464606
Email: india@adlinktech.com