Lanner

Hardware Platforms for Network Computing

API User Guide

Version: 0.13

Date of Release: 2018-04-24

Copyright and Trademarks

Revision History

Version	Date	Descriptions
0.10	02/14/2018	Beta release
0.11	02/26/2018	Add PSU functions (3.9.x)
0.12	03/13/2018	Dispersal MISC to HWM, SLED and SWR
0.13	03/16/2018	Add checkenv.sh OK and Error picture on page 115/116

Table of Contents

CHAPTER 1INTRODUCTION

Lanner SDK aims to simplify and enhance the efficiency of customer's application implementation. When developers intend to write an application that involves hardware access, they were required to fully understand the specifications to utilize the drivers. This is often being considered a time-consuming job which requires lots of related knowledge and time. In order to achieve better full-access hardware functionality, Lanner invests great effort to ease customer's development journey with the release of a suite of reliable Software APIs.

Benefits

Faster Time to Market

Lanner's API helps developers to write applications to control the hardware without knowing the hardware specs of the chipsets and driver architecture.

Reduced Project Effort

Accessing the API, developers are being given a more efficient way to utilize chipsets, communication bus, and physical I/Os. In addition, Lanner SDK also provides corresponded tools to prevent unexpected downtime causing from environment setup matters.

Enhances Lanner Platform Reliability

Lanner SDK is released after a series of reliability test and security validations which combines manufacturing test sequences to enhance complete system reliability.

Flexible Upgrade Possibilities

Considering customer's application maintenance and upgrade tasks, Lanner SDK is designed to be flexible to update/upgrade to the module level. The simple re-initial process will bring up the updated and upgraded modules to be functional.

Backward compatibility

Lanner grants the responsibility to control API backward support, allowing customers to worry less about development of new products.

Operating Systems

OS supports for the current version are as below:

- CentOS 7.4/7.3/6.8
- Ubuntu 14.04.5/16.04.1/17.04

CHAPTER 2DEFINITION

Constant and Macro Define

```
#define int8_t        char
#define int16_t       short
#define int32_t       int
#define int64_t       long long
#define uint8_t       unsigned char
#define uint16_t      unsigned short
#define uint32_t      unsigned int
#define uint64_t      unsigned long long

/* WDT timer base ***/
#define BASE_SECOND     1
#define BASE_MINUTE     2

/* WDT type */
#define WDT_TYPE_UNKNOW   0
#define WDT_TYPE_SIO  1
#define WDT_TYPE_TCO  2

/* GPIO Direction Define */
#define DIRECTION_INPUT   0
#define DIRECTION_OUTPUT  1

#define DISABLE       0
#define ENABLE      1
```

/* Bit Define Hardware Monitor Devices */

```
#define HWM_TEMP_CPU1 0x00000001
#define HWM_TEMP_CPU2 0x00000002
#define HWM_TEMP_SYS1 0x00000004
#define HWM_TEMP_SYS2 0x00000008
#define HWM_FAN_CPU1 0x00000010
#define HWM_FAN_CPU2 0x00000020
#define HWM_FAN_SYS1 0x00000040
#define HWM_FAN_SYS2 0x00000080
#define HWM_FAN_1    0x00000100
#define HWM_FAN_2    0x00000200
#define HWM_FAN_3    0x00000400
#define HWM_FAN_4    0x00000800
#define HWM_VCORE_1   0x00001000
#define HWM_VCORE_2   0x00002000
#define HWM_12V      0x00004000
#define HWM_5V     0x00008000
#define HWM_3V3      0x00010000
#define HWM_5VSB     0x00020000
#define HWM_3V3SB    0x00040000
#define HWM_VBAT     0x00080000
#define HWM_POWER_1   0x00100000
#define HWM_POWER_2   0x00200000
```

/* LAN Bypass Controller Slot define */

```
#define SLOT_ONBOARD 0x00
#define SLOT_INDEX_1 0x01
#define SLOT_INDEX_2 0x02
#define SLOT_INDEX_3 0x03
#define SLOT_INDEX_4 0x04
#define SLOT_INDEX_5 0x05
#define SLOT_INDEX_6 0x06
#define SLOT_INDEX_7 0x07
#define SLOT_INDEX_8 0x08
```

/* LAN Slot modules type */

```
#define LAN_TYPE_UNKNOW   0
#define LAN_TYPE_COPPER   1
#define LAN_TYPE_FIBER    2
```

/* LAN Bypass module bit define */

```
#define LBP_MODBIT_SystemOff    0x01
#define LBP_MODBIT_JustOn     0x02
#define LBP_MODBIT_Runtime      0x04
#define LBP_MODBIT_TIMER1     0x08
#define LBP_MODBIT_TIMER2     0x10
#define LBP_MODBIT_TIMER3     0x20
#define LBP_MODBIT_Disconnect  0x40
```

/* LAN Bypass Timer Configure: action item*/

```
#define PAIR_NOASSIGN       0
#define PAIR_BYPASS_ENABLE      1
#define PORT_DISCONNECT       2
```

```
/***** Intrusion Item Define (bit) *****/
/* Hardware Monitor Intrusion */
#define INTRMSG_CASEOPEN    0x00000001
/* Software Reset Button Intrusion */
#define INTRMSG_SWRESET     0x00000001
/* Power Supply Intrusion */
#define INTRMSG_PSU1_INPUT    0x00000001
#define INTRMSG_PSU2_INPUT    0x00000002
#define INTRMSG_PSU1_TEMP   0x00000004
#define INTRMSG_PSU2_TEMP   0x00000008
#define INTRMSG_PSU1_VOUT   0x00000010
#define INTRMSG_PSU2_VOUT   0x00000020
#define INTRMSG_PSU1_IOUT   0x00000040
#define INTRMSG_PSU2_IOUT   0x00000080
#define INTRMSG_PSU1_FAN    0x00000100
#define INTRMSG_PSU2_FAN    0x00000200


/* PSU Watts Select */
#define PSU_WATTS_INPUT   0
#define PSU_WATTS_OUTPUT  1

/* PSU Status Bit define */
#define PSU_FAULT_NONEOFABOVE 0x0001
#define PSU_FAULT_COMM      0x0002
#define PSU_FAULT_TEMP      0x0004
#define PSU_FAULT_VIN_UV    0x0008
#define PSU_FAULT_IOUT_OC   0x0010
#define PSU_FAULT_VOUT_OV   0x0020
#define PSU_FAULT_UINTOFF   0x0040
#define PSU_FAULT_UINTBUSY    0x0080
#define PSU_FAULT_UNKNOWN   0x0100
#define PSU_FAULT_OTHERS    0x0200
#define PSU_FAULT_FAN     0x0400
#define PSU_FAULT_POWERGOOD 0x0800
#define PSU_FAULT_MFRSPEC   0x1000
#define PSU_FAULT_INPUT     0x2000
#define PSU_FAULT_IOUT      0x4000
#define PSU_FAULT_VOUT      0x8000
```

Structure Define

```
typedef struct DEF_DLL_VERSION {
  uint16_t uwDllMajor;
  uint16_t uwDllMinor;
  uint16_t uwDllBuild;
  uint32_t udwBoardProduct;
    uint16_t uwBoardMajor;
  uint16_t uwBoardMinor;
  uint16_t uwBoardBuild;
}DLL_VERSION;

typedef struct DEF_WDT_INFO {
  uint8_t ubType;
  uint16_t uwCountMax;
  uint8_t unMinuteSupport;
} WDT_INFO;

typedef struct DEF_LBPDEV_INFO {
  uint8_t ubSlotIndex;
  uint8_t ubType;      //copper or fiber
  uint8_t ubVersion[2];     //major & minor
  uint8_t ubModules;     //modules support
  uint8_t ubSystemOff_Pairs; //bit indicate of pair
  uint8_t ubJustOn_Pairs;
  uint8_t ubRuntime_Pairs;
  uint16_t uwTimer1_MaxSec;   //maximum seconds
  uint16_t uwTimer2_MaxSec;
  uint16_t uwTimer3_MaxSec;
}LBPDEV_INFO;
```

```
typedef struct DEF_PAIRS_STATUS {
  uint8_t ubSystemOff_PairsBypass;
  uint8_t ubJustOn_PairsBypass;
  uint8_t ubRuntime_PairsBypass;
  uint8_t ubJustOn_PortsDisconnect;
  uint8_t ubRuntime_PortsDisconnect;
}PAIRS_STATUS;

typedef struct DEF_LBP_TIMERCFG {
  uint8_t  ubTimerNo; //timer number
  uint16_t uwTimeSec; //timer value by second
    uint8_t  ubAction;  //timeout action
    uint8_t  ubEffect;  //assign pairs or ports
}LBP_TIMERCFG;

typedef struct DEF_INTRUSION_TIME {
  uint16_t uwYear;
  uint8_t ubMonth;
    uint8_t  ubDay;
    uint8_t  ubHour;
    uint8_t  ubMinute;
    uint8_t  ubSecond;
}INTRUSION_TIME;

/*** Intrusion Callback Function ***/
typedef struct DEF_INTRUSION_MSG {
  uint32_t udwOccurItem;       //Occur item
  uint32_t udwStatus;       //Occur Status
    INTRUSION_TIME stuTime;      //Occur time
}INTRUSION_MSG;
typedef void (*INTRUSION_CALLBACK)(INTRUSION_MSG);
```

```
/***GPI Changed Callback***/
typedef struct DEF_GPI_MSG {
  uint8_t ubGroup;         //GPI Group
  uint32_t udwGpis;         //Occur GPIs
  uint32_t udwStatus;       //Occur Status
    INTRUSION_TIME stuTime;      //Occur time
}GPI_MSG;
typedef void (*GPI_CALLBACK)(GPI_MSG);

/*** LCM Device Information ***/
typedef struct DEF_LCM_INFO {
  uint16_t uwModeNo;
  uint16_t uwVersion;
    uint32_t udwBaudrate;
} LCM_INFO;

/***LCM keys Callback***/
typedef struct DEF_LCMKEY_MSG {
  uint8_t  ubKeys;      //Occur Keys
  uint8_t ubStatus;      //Occur Status
    INTRUSION_TIME stuTime;    //Occur time
}LCMKEY_MSG;
typedef void (*LCMKEY_CALLBACK)(LCMKEY_MSG);
```

```
/*** Power Supply ***/
typedef struct DEF_PSU_INFO {
  uint8_t ubPsuNo;  //input parameter
  uint8_t strubMfrId[33];
  uint8_t strubMfrModel[33];
  uint8_t strubMfrSerial[33];
  uint8_t strubMfrRevision[33];
}PSU_INFO;

typedef struct DEF_PSU_WATTS {
  uint8_t ubPsuNo;   //input parameter
  uint8_t ubInOut;   //input parameter
  float   fVolt;
  float   fAmpere;
  float   fWatts;
}PSU_WATTS;

typedef struct DEF_PSU_SENSORS {
  uint8_t  ubPsuNo;   //input parameter
  float   fTemp_1;
  float   fTemp_2;
  uint16_t uwFanRpm;
}PSU_SENSORS;
```

Status Code

All Lanner API functions immediately return a status code from a common list of possible errors. Any function may return any of the defined status code.

#define ERR_Success 0

Description

The operation is successful.

#define ERR_Error 0xFFFFFFFF

Description

Generic error message.

#define ERR_NotExist 0xFFFFFFFE

Description

Library file not found, device not exist or plugged

#define ERR_NotOpened 0xFFFFFFFD

Description

Library not opened or not ready yet

#define ERR_Invalid 0xFFFFFFFC

Description

Parameter is not valid value or out of range

#define ERR_NotSupport 0xFFFFFFFB

Description

This function wasn't supported in this platform

#define ERR_BusyInUses 0xFFFFFFFA

Description

Library is in using now or device IO is busy now

#define ERR_BoardNotMatch 0xFFFFFFF9

Description

Board library and board isn't matched

#define ERR_IPMI_IDLESTATE 0xFFFFFEFF

Description

IPMI KCS interface not in IDLE State

#define ERR_IPMI_WRITESTATE 0xFFFFFEFE

Description

IPMI KCS interface WRITE State check failure

#define ERR_IPMI_READSTATE 0xFFFFFEFD

Description

IPMI KCS interface READ State check failure

#define ERR_IPMI_IBF0 0xFFFFFEFC

Description

IPMI KCS interface wait IBF '0' State failure

#define ERR_IPMI_OBF1 0xFFFFFEFB

Description

IPMI KCS interface wait OBF '1' State failure

CHAPTER 3API FUNCTIONS

Library Control

LMB_DLL_Init

The **LMB_DLL_Init** function initial the Lanner common API and board libraries

Syntax

int32_t LMB_DLL_Init(void);

Parameters

(none)

Return Value

ERR_Success: initial library successful

ERR_NotExist: board library not found

ERR_BoardNotMatch: library and M/B is different

ERR_Error: initial library failure

Remarks

(none)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{ int32_t iRet;
     iRet = LMB_DLL_Init();
     if ( iRet == ERR_Success ) {
         printf("Initial library successful\n");
         if ( iRet == ERR_Success ) {
             printf("API Library is Ready\n");
         }
    }
    LMB_DLL_DeInit();
 return 0;
 }
```

LMB_DLL_DeInit

The **LMB_DLL_DeInit** function release the Lanner common API library

Syntax

int32_t LMB_DLL_DeInit(void);

Parameters

(none)

Return Value

```
ERR_Success: Release library successful
ERR_BusyInUses: libray busy or some process in uses
```

Remarks

This function will auto releasing board level library

Example

```
Refer function LMB_DLL_Init example.
```

LMB_DLL_Version

The **LMB_DLL_Version** function loads the Lanner board level library

Syntax

int32_t LMB_DLL_Version(DLL_VERSION* pstuDllVersion);

Parameters

pstuDllVersion

[in] obtain the DLL and Board Library version information

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: this platform is not support this function

Remarks

(none)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{
int32_t iRet;
DLL_VERSION stuVer;
    LMB_DLL_Init();
    iRet = LMB_DLL_Version(&stuVer);
    if ( iRet == ERR_Success ) {
  printf("DLL Version: %d.%d.%d\n",stuVer.uwDllMajor,
       stuVer.uwDllMinor,stuVer.uwDllBuild);
        printf("Board Version: %d.%d.%d.%d\n",
                    stuVer.udwBoardProduct, stuVer.uwBoardMajor,
                    stuVer.uwBoardMinor, stuVer.uwBoardBuild);
    }
   LMB_DLL_DeInit();
return 0;
}
```

Hardware Monitor

LMB_HWM_DevInfo

The **LMB_HWM_DevInfo** function gets monitor devices of this platform

Syntax

int32_t LMB_HWM_DevInfo(uint32_t* pudwDevInfo);

Parameters

pudwDevInfo

[in] obtain the monitor devices information.

Please refer chapter 2.1 “Bit Define for Hardware Monitor Devices”

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: this platform is not support this function

Remarks

none

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{
int32_t iRet ;
uint32_t dwDevInfo =0;
     LMB_DLL_Init(ENABLE);
     iRet = LMB_HWM_DevInfo(&dwDevInfo);
     if ( iRet == ERR_Success )
                printf("Monitor devices is %08X\n”, dwDevInfo);
     LMB_DLL_DeInit();
     return 0;
}
```

LMB_HWM_GetCpuTemp

The **LMB_HWM_GetCpuTemp** function reads temperature of the CPU processor

Syntax

int32_t LMB_HWM_GetCpuTemp(uint8_t ubCpuNo, float* pfTemperature);

Parameters

ubCpuNo

[out] select CPU number, single CPU platform this parameter will skip.

pfTemperature

[in] the current temperature value of the CPU.

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_Invalid: parameter input is out of range

ERR_Error: function is failure

ERR_NotSupport: this platform is not support this fucntion

Remarks

none

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{
int32_t iRet ;
float fTemp =0;
     LMB_DLL_Init();
     iRet = LMB_HWM_GetCpuTemp(1, &fTemp);
     if ( iRet == ERR_Success )
                printf("CPU 1 temperature is %f\n”, fTemp);
     LMB_DLL_DeInit();
     return 0;
 }
```

LMB_HWM_GetSysTemp

The **LMB_HWM_GetSysTemp** function reads temperature of the system.

Syntax

int32_t LMB_HWM_GetSysTemp(uint8_t ubSysNo, float* pfTemperature);

Parameters

ubSysNo

[out] select System sensor, single sensor platform this parameter will skip.

pfTemperature

[in] the current temperature value of the CPU.

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_Invalid: parameter input is out of range

ERR_Error: function is failure

ERR_NotSupport: this platform is not support this function

Remarks

none

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{
int32_t iRet ;
float fTemp =0;
     LMB_DLL_Init();
     iRet = LMB_HWM_GetSysTemp(1, &fTemp);
     if ( iRet == ERR_Success )
                printf("SYS 1 temperature is %f\n”, fTemp);
     LMB_DLL_DeInit();
     return 0;
}
```

LMB_HWM_GetVcore

The **LMB_HWM_GetVcore** function reads the CPU Vcore voltage.

Syntax

int32_t LMB_HWM_GetVcore(uint8_t ubCpuNo, float* pfVoltage);

Parameters

ubCpuNo

[out] select CPU number, single CPU platform this parameter will skip.

pfVoltage

[in] obtain the current voltage of CPU Vcore.

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
   ERR_Invalid: parameter input is out of range
ERR_Error: function is failure
   ERR_NotSupport: this platform is not support this function
```

Remarks

```
none
```

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{
int32_t iRet ;
float  fVolt =0;
     LMB_DLL_Init();
     iRet = LMB_HWM_GetVcore(1, &fVolt);
     if ( iRet == ERR_Success )
                printf("CPU 1 Vcore Voltage = %f\n”, fvolt);
     LMB_DLL_DeInit();
     return 0;
 }
```

LMB_HWM_Get12V

The **LMB_Get12V** function reads the positive 12V voltage.

Syntax

int32_t LMB_HWM_Get12V(float* pfVoltage);

Parameters

pfVoltage

[in] obtain the current voltage of positive 12V

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_Error: function is failure

ERR_NotSupport: this platform is not support this fucntion

Remarks

(none)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{
int32_t iRet ;
float  fVolt =0;
     LMB_DLL_Init();
     iRet = LMB_HWM_Get12V(&fVolt);
     if ( iRet == ERR_Success )
                printf("12V Voltage = %f\n”, fvolt);
     LMB_DLL_DeInit();
     return 0;
 }
```

LMB_HWM_Get5V

The **LMB_HWM_Get5V** function reads the positive 5V voltage.

Syntax

int32_t LMB_HWM_Get5V(float* pfVoltage);

Parameters

pfVoltage

[in] obtain the current voltage of positive 5V

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_Error: function is failure

ERR_NotSupport: this platform is not support this fucntion

Remarks

(none)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{
int32_t iRet ;
float  fVolt =0;
     LMB_DLL_Init();
     iRet = LMB_HWM_Get5V(&fVolt);
     if ( iRet == ERR_Success )
                printf("5V Voltage = %f\n”, fVolt);
     LMB_DLL_DeInit();
     return 0;
}
```

LMB_HWM_Get3V3

The **LMB_HWM_Get3V3** function reads the positive 3.3V voltage.

Syntax

int32_t LMB_HWM_Get3V3(float* pfVoltage);

Parameters

pfVoltage

[in] obtain the current voltage of positive 3.3V

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_Error: function is failure

ERR_NotSupport: this platform is not support this fucntion

Remarks

(none)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{
int32_t iRet ;
float  fVolt =0;
     LMB_DLL_Init();
     iRet = LMB_HWM_Get3V3(&fVolt);
     if ( iRet == ERR_Success )
                printf("3.3V Voltage = %f\n”, fVolt);
     LMB_DLL_DeInit();
     return 0;
}
```

LMB_HWM_Get5Vsb

The **LMB_HWM_Get5Vsb** function reads the positive 5V standby power.

Syntax

int32_t LMB_HWM_Get5Vsb(float* pfVoltage);

Parameters

pfVoltage

[in] obtain the current voltage of positive 5V standby power

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_Error: function is failure

ERR_NotSupport: this platform is not support this fucntion

Remarks

(none)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{
int32_t iRet ;
float  fVolt =0;
     LMB_DLL_Init();
     iRet = LMB_HWM_Get5Vsb(&fVolt);
     if ( iRet == ERR_Success )
                printf("5V standby Voltage = %f\n”, fvolt);
     LMB_DLL_DeInit();
     return 0;
}
```

LMB_HWM_Get3V3sb

The **LMB_HWM_Get3Vsb** function reads the positive 3V/3.3V standby power.

Syntax

int32_t LMB_HWM_Get3V3sb(float* pfVoltage);

Parameters

pfVoltage

[in] obtain the current voltage of positive 3.3V standby power

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_Error: function is failure

ERR_NotSupport: this platform is not support this fucntion

Remarks

(none)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{
int32_t iRet ;
float  fVolt =0;
     LMB_DLL_Init();
     iRet = LMB_HWM_Get3V3sb(&fVolt);
     if ( iRet == ERR_Success )
                printf("3.3V standby Voltage = %f\n”, fVolt);
     LMB_DLL_DeInit();
     return 0;
}
```

LMB_HWM_GetVbat

The **LMB_HWM_GetVbat** function reads the battery voltage.

Syntax

int32_t LMB_HWM_GetVbat(float* pfVoltage);

Parameters

pfVoltage

[in] obtain the current voltage of the battery

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_Error: function is failure
   ERR_NotSupport: this platform is not support this fucntion
```

Remarks

(none)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{
int32_t iRet ;
float  fVolt =0;
     LMB_DLL_Init();
     iRet = LMB_HWM_GetVbat(&fVolt);
     if ( iRet == ERR_Success )
                printf("Battery Voltage = %f\n”, fVolt);
     LMB_DLL_DeInit();
     return 0;
}
```

LMB_HWM_GetCpuFan

The **LMB_HWM_GetCpuFan** function reads the CPU fan speed

Syntax

int32_t LMB_HWM_GetCpuFan(uint8_t ubCpuNo, uint16_t* puwRpm);

Parameters

ubCpuNo

[out] select CPU number, single CPU platform this parameter will skip.

puwRpm

[in] obtain the CPU fan speed

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_Error: function is failure
   ERR_NotSupport: this platform is not support this fucntion
```

Remarks

```
(none)
```

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{
int32_t iRet ;
uint16_t wRPM =0;
     LMB_DLL_Init();
     iRet = LMB_HWM_GetCpuFan(1, &wRPM);
     if ( iRet == ERR_Success )
                printf("CPU 1 Fan Speed = %d\n”, wRPM);
     LMB_DLL_DeInit();
     return 0;
}
```

LMB_HWM_GetSysFan

The **LMB_HWM_GetSysFan** function reads the system fan speed

Syntax

int32_t LMB_HWM_GetSysFan(uint8_t ubSysNo, uint16_t* puwRpm);

Parameters

ubSysNo

[out] select system fan, single system fan platform this parameter will skip.

puwRpm

[in] obtain the system fan speed

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_Error: function is failure
   ERR_NotSupport: this platform is not support this fucntion
```

Remarks

(none)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{
int32_t iRet ;
uint16_t wRPM =0;
     LMB_DLL_Init();
     iRet = LMB_HWM_GetSysFan(1, &wRPM);
     if ( iRet == ERR_Success )
                printf("System 1 Fan Speed = %d\n”, wRPM);
     LMB_DLL_DeInit();
     return 0;
}
```

LMB_HWM_GetFanSpeed

The **LMB_HWM_GetFanSpeed** function reads the fan speed which was assigned

Syntax

int32_t LMB_HWM_GetFanSpeed(uint8_t ubFanNo, uint16_t* puwRpm);

Parameters

ubFanNo

[out] assign the fan index number which is described in user manual

puwRpm

[in] obtain the fan speed

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_Error: function is failure
   ERR_NotSupport: this platform is not support this fucntion
```

Remarks

(none)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{
int32_t iRet ;
uint16_t wRPM =0;
uint8_t fanno = 2;
     LMB_DLL_Init();
     iRet = LMB_HWM_GetCpuFan(fanno, &wRPM);
     if ( iRet == ERR_Success )
                printf("Fan%d Speed = %d\n”,fanno, wRPM);
     LMB_DLL_DeInit();
     return 0;
}
```

LMB_HWM_GetPowerSupply

The **LMB_HWM_GetPowerSupply** function reads the power supply AC input

Syntax

int32_t LMB_HWM_GetPowerSupply(uint8_t ubPowerNo,
uint16_t* puwVoltage);

Parameters

ubPowerNo

[out] assign the AC power supply number which is described in user manual

puwVoltage

[in] obtain the AC power voltage

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_Error: function is failure

ERR_NotSupport: this platform is not support this fucntion

Remarks

(none)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{
int32_t iRet ;
uint16_t wVolt =0;
uint8_t pwrno = 2;
     LMB_DLL_Init();
     iRet = LMB_HWM_GetPowerSupply(pwrno, &wVolt);
     if ( iRet == ERR_Success )
                printf("Power-%d Voltage = %d\n”,powno, wVolt);
     LMB_DLL_DeInit();
     return 0;
}
```

LMB_HWM_CaseOpenStatus

The **LMB_HWM_CaseOpenStatus** function gets the case cover status

Syntax

int32_t LMB_HWM_CaseOpenStatus(uint8_t *pubStatus);

Parameters

pubStatus

[in] indicate case cover status.

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

none

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
  {
  int32_t iRet;
  uint8_t bStatus= 0;
      LMB_DLL_Init();
      iRet = LMB_HWM_CaseOpenStatus(&bStatus);
      if (iRet == ERR_Success)
                  printf("Case cover status is %d\n", bStatus);
      LMB_DLL_DeInit();
      return 0;
  }
```

LMB_HWM_IntrCallback

The **LMB_HWM_IntrCallback** function hooks some status change callback

Syntax

int32_t LMB_HWM_IntrCallback(INTRUSION_CALLBACK pCallback
uint16_t uwmSec);

Parameters

pCallback

[out] hook callback function pointer for status changed event.

uwmSec

[out] setting period of time for checking status changed
[range: 10 ~ 999ms]

Return Value

```
  ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
    ERR_BusyInUses: callback function is running now
    ERR_Invalid: parameter is out of range
```

Remarks

Supports case open, software reset button and power status changed and power supply fault event (refer to "Intrusion Item Define (bit)" in chapter 2)
When pCallback is NULL pointer this function will disable.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
void hwmCallback(INTRUSION_MSG pIntr){
       printf("item=%04X,status=%04X,\
       time = %04d/%02d/%02d %02d:%02d:%02d\n",\
       pInt.udwOccurItem, pIntr.udwStatus, pIntr.stuTime.year, \
       pIntr.stuTime.month, pIntr.stuTime.day, pIntr.stuTime.hour,\
       pIntr.stuTime.minute, pIntr.stuTime.second”);
  }
int main(){
  LMB_DLL_Init();
  LMB_HWM_IntrCallback(hwmCallback, 100);
  While (1) {
  ……
       }
       LMB_DLL_DeInit();
       return 0;
}
```

3.2.17 LMB_HWM_ClearCaseOpenStatus

The **LMB_HWM_ClearCaseOpenStatus** function reset case open occurred status

Syntax

int32_t LMB_HWM_ClaerCaseOpenStatus(void);

Parameters

(none)

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_NotSupport: function not support
```

Remarks

Some platforms need clear case open status for next event detection.

When clear status the case open status still occurred this is mean case still opened.

Example

```
none
```

Watch Dog Timer

LMB_WDT_QueryInfo

The **LMB_WDT_Queryinfo** function gets Watch Dog Timer information.

Syntax

int32_t LMB_WDT_QueryInfo(WDT_INFO* pstuWdtInfo);

Parameters

pstuWdtInfo

[in] structure pointer for obtaining WDT information

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: this platform is not support this fucntion

Remarks

(none)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{ int32_t iRet;
    WDT_INFO stuWdtInfo;
    LMB_DLL_Init();
    iRet = LMB_WDT_QueryInfo(&stuWdtInfo);
    if ( iRet == ERR_Success ) {
  printf("WDT type = %d\n", stuWdtInfo.ubType);
  printf("WDT maximum count = %d\n", stuWdtInfo.uwCountMax);
  if (stuWdtInfo.unMinuteSupport)
    printf("WDT support MUNUTE and SECOND time base\n");
  else
            printf("WDT support only support SECOND base\n");
    }
    LMB_DLL_DeInit();
 return 0;
}
```

LMB_WDT_Config

The **LMB_WDT_Config** function configure the Watch Dog Timer

Syntax

int32_t LMB_WDT_Config(uint16_t uwCount, uint8_t ubTimeBase);

Parameters

uwCount

[out] the value is setting the timer count down

ubTimeBase

[out] the value is select time base.

1	Second base
2	Minute base

Return Value

```
ERR_Success: function is successful
ERR_Invalid: parameter invalid value
ERR_NotOpened: library not ready or opened yet
   ERR_NotSupport: this platform is not support this function
   ERR_BusyInUse: skip because WDT already starting now
```

Remarks

(none)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{
uint8_t iRet ;
 LMB_DLL_Init();
    iRet = LMB_WDT_Config(200, BASE_SECOND);
    if ( iRet == ERR_Success ) {
printf("WDT Ready: reset system after timeout.\n");
    LMB_WDT_Start(); //starting WDT
  while(1) {
…… Main Loop Process ……
   LMB_WDT_Tick();
  }
    else {
      printf("WDT configure failure\n");
    }
    LMB_DLL_DeInit();
    return 0;
}
```

LMB_WDT_Start

LMB_WDT_Start function is starting the WDT countdown.

Syntax

int32_t LMB_WDT_Start(void);

Parameters

(none)

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
   ERR_NotSupport: this platform is not support this function
```

Remarks

(none)

Example

```
Refer function LMB_WDT_Config example.
```

LMB_WDT_Stop

The **LMB_WDT_Stop** function is stopped the WDT countdown.

Syntax

int32_t LMB_WDT_Stop(void);

Parameters

(none)

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
   ERR_NotSupport: this platform is not support this function
```

Remarks

(none)

Example

```
Refer function LMB_WDT_Config example.
```

LMB_WDT_Tick

The **LMB_WDT_Tick** function reloads the timer then re-computes it.

Syntax

int32_t LMB_WDT_Tick(void);

Parameters

(none)

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
   ERR_NotSupport: this platform is not support this function
```

Remarks

(none)

Example

```
Refer function LMB_WDT_Config example.
```

General Purpose Input Output

LMB_GPIO_GetInfo

The **LMB_GPIO_GetInfo** function gets this platform GPIO supported information

Syntax

int32_t LMB_GPIO_GetInfo(uint8_t ubGroup, uint8_t* pubGpis,
uint8_t* pubGpos);

Parameters

ubGroup

[out] select GPIO group, [range: 1 ~ 10]

pubGpis

[in] obtain the GPIs supports numbers of this platform

pubGpos

[in] obtain the GPOs supports numbers of this platform

Return Value

ERR_Success: function is successful

ERR_Error: function is failure

ERR_NotOpened: ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: platform not support GPIO direction setting

Remarks

none

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{
int32_t iRet;
uint8_t ubGPIs=0, ubGPOs=0;
  LMB_DLL_Init();
  /** pin 0/1 set to output mode, pin 2/3 set to input mode **/
  iRet = LMB_GPIO_GetInfo(0, &ubGPIs, &ubGPOs);
  if  ( iRet == ERR_Success )   {
    printf("GPIs = %d, GPOs = %d\n”, ubGPIs, ubGPOs);
  else
            printf("Functions error \n");
  LMB_DLL_DeInit();
  return 0;
}
```

LMB_GPIO_GpoWrite

The **LMB_GPIO_GpoWrite** function writes data to GPO group

Syntax

int32_t LMB_GPIO_GpoWrite(uint8_t ubGroup, uint32_t udwValue);

Parameters

ubGroup

[out] select GPIO group, if only one group, this parameter will skip.

udwValue

[out] the value for DO writing

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_Error: function is failure
ERR_NotSupport: function not support
```

Remarks

DO group is depend on hardware define, 32 pins is maximum for one group,

GPO group1: GPO_11 ~ GPO_18 or GPO_101 ~ GPO_132

GPO group2: GPO_21 ~ GPO_28

No group: GPO_1 ~ GPO_4

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[] {
  int32_t iRet;
  uint32_t dwSize;
      LMB_DLL_Init();
      /** set DO pin0~1 is '0', Do pin 2~3 is '1' **/
      iRet = LMB_GPIO_GpoWrite(1, 0x0C);
      if  ( iRet == ERR_Success )  printf("Wrote was successful\n");
      LMB_DLL_DeInit();
      return 0;
  }
```

LMB_GPIO_GpoPinWrite

The **LMB_GPIO_GpoPinWrite** function writes data to a specific pin of GPO

Syntax

int32_t LMB_GPIO_GpoPinWrite(uint8_t ubGroup, uint8_t ubPinNo,
uint8_t ubValue);

Parameters

ubGroup

[out] select GPO group, if only one group, this parameter will skip

ubPinNo

[out] t assign a pin of GPO to write data [range: 1~32]

ubValue

[out] the value for write [range: 0/1] (low/high)

Return Value

```
ERR_Success: function is successful
ERR_Invalid: assign DO pin not exist
ERR_NotOpened: library not ready or opened yet
ERR_Error: function is failure
   ERR_NotSupport: function not support
```

Remarks

(none)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[]) {
     LMB_DLL_Init();
     /** set GPO_1/2 is '0', GPO_3/4 is '1' **/
     LMB_GPIO_GpoPinWrite(0, 1, 0);
     LMB_GPIO_GpoPinWrite(0, 2, 0);
     LMB_GPIO_GpoPinWrite(0, 3, 1);
     LMB_GPIO_GpoPinWrite(0, 4, 1);
     LMB_DLL_DeInit();
     return 0;
 }
```

LMB_GPIO_GpoRead

The **LMB_GPIO_GpoRead** function reads data of assigned GPO group.

Syntax

int32_t LMB_GPIO_GpoRead(uint8_t ubGroup, uint32_t* pudwValue);

Parameters

ubGroup

[out] select GPO group, if only one group, this parameter will skip

pudwValue

[in] the value for read, always read '0' when the GPO pin is invalid or not exist

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_Error: function is failure
   ERR_NotSupport: function not support
```

Remarks

(none)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[]) {
  int32_t iRet;
  uint32_t dwData=0;
      LMB_DLL_Init();
      iRet = LMB_GPIO_GpoRead(0, &dwData);
      if (iRet == ERR_Success) printf("GPO status is %08X\n",dwData);
      LMB_DLL_DeInit();
      return 0;
}
```

LMB_GPIO_GpoPinRead

The **LMB_GPIO_GpoPinRead** function reads data of assigned GPO pin.

Syntax

int32_t LMB_GPO_GpoPinRead(uint8_t ubGroup, uint8_t ubPinNo,
uint8_t* pubValue);

Parameters

ubGroup

[out] select GPO group, if only one group, this parameter will skip

ubPinNo

[out] assign a pin of GPO to write data [range: 1~32]

pubValue

[in] the value for read [range: 0/1] (low/high)

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_Error: function is failure
   ERR_NotSupport: function not support
```

Remarks

(none)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{
int32_t iRet;
uint8_t pin=3, bData=0;
     LMB_DLL_Init();
     iRet = LMB_GPIO_GpoPinRead(0, pin &bData);
     if (iRet == ERR_Success)
             printf("GPO pin%d status is %d\n",pin, bData);
     LMB_DLL_DeInit();
     return 0;
 }
```

LMB_GPIO_GpiRead

The **LMB_GPIO_GpiRead** function reads data of assigned GPI group.

Syntax

int32_t LMB_GPIO_GpiRead(uint8_t ubGroup, uint32_t* pudwValue);

Parameters

ubGroup

[out] select GPI group, if only one group, this parameter will skip

pudwValue

[in] the value for read, always read '0' when the GPI pin is invalid or not exist

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: DIO device not opened yet
ERR_Error: function is failure
   ERR_NotSupport: function not support
```

Remarks

(none)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{
 int32_t iRet;
 uint32_t dwData=0;
     LMB_DLL_Init();
 iRet = LMB_GPIO_GpiRead(0, &dwData);
     if (iRet == ERR_Success)
                 printf("GPI pins status is %08X\n",dwData);
     LMB_DLL_DeInit();
     return 0;
 }
```

LMB_GPIO_GpiPinRead

The **LMB_GPIO_GpiPinRead** function reads data of assigned GPI pin.

Syntax

int32_t LMB_GPIO_GpiPinRead(uint8_t ubGroup, uint8_t ubPinNo,
uint8_t* pubValue);

Parameters

ubGroup

[out] select GPI group, if only one group, this parameter will skip

ubPinNo

[out] t assign a pin of GPI to write data [range: 1~32]

pubValue

[in] the value for read [range: 0/1] (low/high)

Return Value

ERR_Success: function is successful

ERR_NotOpened: DIO device not opened yet

ERR_Error: function is failure

ERR_NotSupport: function not support

Remarks

(none)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
  {
  int32_t iRet;
  uint8_t pin=3, bData=0;
      LMB_DLL_Init();
      iRet = LMB_GPIO_GpiPinRead(0, pin &bData);
      if (iRet == ERR_Success)
                  printf("GPI pin%d status is %d\n",pin, bData);
      LMB_DLL_DeInit();
      return 0;
  }
```

LMB_GPIO_GpiCallback

The **LMB_GPIO_GpiCallback** function hooks GPI status changed callback

Syntax

int32_t LMB_GPIO_GpiCallback(GPI_CALLBACK pCallback,
uint16_t uwmSec);

Parameters

pCallback

[out] hook callback function pointer for GPI status changed event.

uwmSec

[out] setting period of time for checking status changed
[range: 10 ~ 999ms]

Return Value

```
  ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
    ERR_BusyInUses: callback function is running now
    ERR_Invalid: parameter is out of range
```

Remarks

Current supports case open, software reset button and power status changed.

When pCallback is NULL pointer this function will disable.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
void gpiCallback(GPI_MSG pGpiMsg){
       printf("Gpis=%02X,status=%02X,\
       time = %04d/%02d/%02d %02d:%02d:%02d\n",\
       pGpiMsg.udwGpis, pGpiMsg.udwStatus, pGpiMsg.stuTime.year, \
       pGpiMsg.stuTime.month, pGpiMsg.stuTime.day, \
       pGpiMsg.stuTime.hour, pGpiMsg.stuTime.minute, \
       pGpiMsg.stuTime.second”);
  }
int main(){
  LMB_DLL_Init();
  LMB_GPIO_GpiCallback(gpiCallback, 250);
  While (1) {
  ……
       }
       LMB_DLL_DeInit();
       return 0;
}
```

System LED

LMB_SLED_SetSystemLED

The **LMB_SLED_SetSystemLED** function sets the system LED display mode

Syntax

int32_t LMB_SLED_SetSystemLED(uint8_t ubLedMode);

Parameters

ubLedMode

[out] setting the LED display mode. (range 0 ~ 3)

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_Invalid: parameter is out of range
ERR_NotSupport: function not support
```

Remarks

The LED colors is depend on the platform hardware configuration

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{
int32_t iRet;
uint8_t sled = 1;//mode1 -> light green LED
    LMB_DLL_Init();
    iRet = LMB_SLED_SetSystemLED(sled);
    if (iRet == ERR_Success) printf("Green LED is ready\n");
    LMB_DLL_DeInit();
    return 0;
}
```

LMB_SLED_GetSystemLED

The **LMB_SLED_GetSystemLED** function gets the system LED display mode status

Syntax

int32_t LMB_SLED_GetSystemLED(uint8_t *pubLedMode);

Parameters

pubLedMode

[in] obtain system LED display mode.

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

The LED colors depend on the platform hardware configuration

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
  {
  int32_t iRet;
  uint8_t sled= 0;
      LMB_DLL_Init();
      iRet = LMB_SLED_GetSysemLED(&sled);
      if (iRet == ERR_Success) printf("LED mode is %d\n", sled);
      LMB_DLL_DeInit();
      return 0;
  }
```

LAN Bypass Control

LMB_LBP_QueryDevices

The **LMB_LBP_QueryDevices** function gets platform installed controller devices.

Syntax

int32_t LMB_LBP_QueryDevices(uint16_t* puwSlotsDev);

Parameters

puwSlotsDev

[in] obtain this platform devices installed status.

bit status 0 is empty, status 1 is installed

bit 0	Onboard LBP controller
bit 1	Slot1 LBP controller status
bit 2	Slot2 LBP controller status
bit 3	Slot3 LBP controller status
bit 4	Slot4 LBP controller status
bit 5	Slot5 LBP controller status
bit 6	Slot6 LBP controller status
bit 7	Slot7 LBP controller status
bit 8	Slot8 LBP controller status
bit 9 ~ bit 15 is not using now	

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_NotSupport: function not support
```

Remarks

```
none
```

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
uint16_t uwSlotsDev;
  LMB_DLL_Init();
  LMB_LBP_QueryDevices(&uwSlotsDev);
  if ( iRet == ERR_Success ) {
    printf("Current hardware devices list:\n");
    if (uwSlotsDev & 0x0001 ) printf("    SLOT_ONBOARD\n");
    if (uwSlotsDev & 0x0002 ) printf("    SLOT_INDEX_1\n");
    if (uwSlotsDev & 0x0004 ) printf("    SLOT_INDEX_2\n");
    if (uwSlotsDev & 0x0008 ) printf("    SLOT_INDEX_3\n");
    if (uwSlotsDev & 0x0010 ) printf("    SLOT_INDEX_4\n");
    if (uwSlotsDev & 0x0020 ) printf("    SLOT_INDEX_5\n");
    if (uwSlotsDev & 0x0040 ) printf("    SLOT_INDEX_6\n");
    if (uwSlotsDev & 0x0080 ) printf("    SLOT_INDEX_7\n");
    if (uwSlotsDev & 0x0100 ) printf("    SLOT_INDEX_8\n");
  }
      LMB_DLL_DeInit();
      return 0;
  }
```

LMB_LBP_DeviceInfo

The **LMB_LBP_DeviceInfo** function gets the controller device information

Syntax

int32_t LMB_LBP_DeviceInfo(uint8_t ubSlot, LBPDEV_INFO *pstuDevInfo);

Parameters

ubSlot

[in] assign the slot for device access

pstuDevInfo

[in] structure pointer for obtain controller device information.

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_NotSupport: function not support
```

Remarks

```
none
```

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
LBPDEV_INFO LbpDevInfo;
LMB_DLL_Init();
iRet = LMB_LBP_DeviceInfo(xi, &LbpDevInfo);
if ( iRet == ERR_Success ) {
  printf("===== Lan Bypass Controller information =====\n");
  printf("Slot index is %d\n", LbpDevInfo.ubSlotIndex);
  if  ( LbpDevInfo.ubType == LAN_TYPE_COPPER )
                                    printf("Controller Type is Copper\n");
       else if ( LbpDevInfo.ubType == LAN_TYPE_FIBER  )
                                    printf("Controller Type is Fiber\n");
  else              printf("Controller Type is unknown\n");
  printf("Version: %d.%d\n",LbpDevInfo.ubVersion[0], LbpDevInfo.ubVersion[1]);
       printf("  Modules: %02X\n",LbpDevInfo.ubModules);
       printf("  SystemOff Pairs: %02X\n",LbpDevInfo.ubSystemOff_Pairs);
       printf("  JustOn Pairs: %02X\n",LbpDevInfo.ubJustOn_Pairs);
       printf("  Runtime Pairs: %02X\n",LbpDevInfo.ubRuntime_Pairs);
       printf("  Timer1 MaxSec: %d seconds\n",LbpDevInfo.uwTimer1_MaxSec);
       printf("  Timer2 MaxSec: %d seconds\n",LbpDevInfo.uwTimer2_MaxSec);
       printf("  Timer3 MaxSec: %d seconds\n",LbpDevInfo.uwTimer3_MaxSec);
}
LMB_DLL_DeInit();
   return 0;
}
```

LMB_LBP_FactoryReset

The **LMB_LBP_FactoryReset** function clear setting of the device configuration

Syntax

int32_t LMB_LBP_FactoryReset(uint8_t ubSlot);

Parameters

ubSlot

[in] assign the slot for device access

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

Reset all configure to factory default setting

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
  LMB_DLL_Init();
  LMB_LBP_FactoryReset();
  if ( iRet == ERR_Success ) {
    printf("Lan Bypass Control was set to factory default\n");
  }
      LMB_DLL_DeInit();
      return 0;
  }
```

LMB_LBP_SaveConfig

The **LMB_LBP_SaveConfig** function save the configuration to NVRAM as default

Syntax

int32_t LMB_LBP_SaveConfig(uint8_t ubSlot);

Parameters

ubSlot

[in] assign the slot for device access

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

none

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
  LMB_DLL_Init();
  LMB_LBP_SaveConfig();
  if ( iRet == ERR_Success ) {
    printf("Lan Bypass Control was saved to NVRAM successful\n");
  }
      LMB_DLL_DeInit();
      return 0;
  }
```

LMB_LBP_GetPairsStatus

The **LMB_LBP_GetPairsStatus** function gets all pairs and ports current status

Syntax

int32_t LMB_LBP_GetPairsStatus(uint8_t ubSlot,
PAIRS_STATUS *pstuPairsStatus);

Parameters

ubSlot

[in] assign the slot for device access

pstuPairsStatus

[in] a pointer of PAIRS_STATUS structure

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_NotSupport: function not support
```

Remarks

none

Example

```
  #include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
  int main(){
int32_t iRet;
  PAIRS_STATUS stuPairsStatus
  LMB_DLL_Init();
  iRet =  LMB_LBP_GetPairsStatus(1, &PairsStatus);
  if ( iRet == ERR_Success )  {
    printf("===========GetPairsStatus==============\n");
    printf("Slot index is %d\n", PairsStatus.ubSlotIndex);
    printf("SystemOff_PairsBypass: %02X\n",
           PairsStatus.ubSystemOff_PairsBypass);
    printf("JustOn_PairsBypass: %02X\n",
                          PairsStatus.ubJustOn_PairsBypass);
    printf("Runtime_PairsBypass: %02X\n",
                          PairsStatus.ubRuntime_PairsBypass);
    printf("JustOn_PortsDisconnect: %02X\n",
                          PairsStatus.ubJustOn_PortsDisconnect);
    printf("ubRuntime_PortsDisconnect: %02X\n",
                          PairsStatus.ubRuntime_PortsDisconnect);
  }
      LMB_DLL_DeInit();
      return 0;
  }
```

LMB_LBP_SetPairBypass

The **LMB_LBP_SetPairBypass** function sets the LAN bypass pair new status

Syntax

int32_t LMB_LBP_SetPairBypass(uint8_t ubSlot, uint8_t ubPairNo,
uint8_t ubEnable);

Parameters

ubSlot

[out] assign the slot for device access

ubPairNo

[out] assign a pair to set bypass function

ubEnable

[out] ENABLE or DISABLE bypass function

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

This function will update pair status immediately in Runtime stage

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
LMB_DLL_Init();
iRet = LMB_LBP_SetPairBypass(1, 1, ENABLE);
if ( iRet == ERR_Success ) {
printf("Set Slot 1, Pair 1, Lan Bypass Enable\n");
}
LMB_DLL_DeInit();
return 0;
}
```

LMB_LBP_SetAllPairs

The **LMB_LBP_SetAllPairs** function sets the all pairs new status

Syntax

int32_t LMB_LBP_SetAllPairs(uint8_t ubSlot, uint8_t ubStatus);

Parameters

ubSlot

[out] assign the slot for device access

ubStatus

[out] new status for all pairs

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

This function will update all pair status immediately in Runtime stage

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
uint8_t ubStatus = 0x03;  //pari1 & pair2 bypass enable
LMB_DLL_Init();
iRet =  LMB_LBP_SetAllPairs(1, ubStatus);
if ( iRet == ERR_Success )  {
printf("Set Slot 1, Pair 1&2, Lan Bypass Enable\n");
printf("Set Slot 1, Pair 3&4, Lan Bypass Disable\n");
}
LMB_DLL_DeInit();
return 0;
}
```

LMB_LBP_SetPortDisconnect

The **LMB_LBP_SetPortDisconnect** function sets the fiber port disconnect status

Syntax

int32_t LMB_LBP_SetPortDisconnect(uint8_t ubSlot, uint8_t ubPortNo,
uint8_t ubEnable);

Parameters

ubSlot

[in] assign the slot for device access

ubPortNo

[out] assign a port to set disconnect function

ubEnable

[out] ENABLE or DISABLE fiber port disconnect

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

This function will update fiber port connection status immediately in Runtime stage.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
uint8_t ubStatus = 0x03;  //pari1 & pair2 bypass enable
LMB_DLL_Init();
iRet =  LMB_LBP_SetPortDisconnect(2, 1, ENABLE );
if ( iRet == ERR_Success )  {
printf("Set Slot 2, Fiber port 1, Lan Disconnect Enable\n");     }
LMB_DLL_DeInit();
return 0;
}
```

LMB_LBP_SetAllPortsDisconnect

The **LMB_LBP_SetAllPortsDisconnect** function sets all fiber port disconnect.

Syntax

int32_t LMB_LBP_SetAllPortsDisconnect(uint8_t ubSlot, uint8_t ubStatus);

Parameters

ubSlot

[in] assign the slot for device access

ubStatus

[out] assign new status to set all fiber ports disconnect

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_NotSupport: function not support
```

Remarks

This function will update fiber port connection status immediately in Runtime stage.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
uint8_t ubStatus = 0x0C;  //port3 port4 disconnect enable
LMB_DLL_Init();
iRet =  LMB_LBP_SetAllPortsDisconnect(2, ubStatus);
if ( iRet == ERR_Success )  {
printf("Set Slot 2, Fiber port 3 & 4, Lan Disconnect Enable\n");  }
LMB_DLL_DeInit();
return 0;
}
```

LMB_LBP_TimerConfig

The **LMB_LBP_TimerConfig** function configures Timer count and timeout action

Syntax

int32_t LMB_LBP_TimerConfig(uint8_t ubSlot,
LBP_TIMERCFG stuLbpTimerCfg);

Parameters

ubSlot

[in] assign the slot for device access

stuLbpTimerCfg

[out] structure for configuring LAN bypass timer

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_NotSupport: function not support
```

Remarks

If timer second value large maximum value, the function will use maximum value.

Timer3 count step is 5 seconds, timer count will divisor 5 before setting

Example

```
   #include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
   int main(){
int32_t iRet;
   uint8_t bSlot=1;
LMB_DLL_Init();
   printf("===========LMB_LBP_TimerConfig==============\n");
stuTimerCfg.ubTimerNo = 1;
stuTimerCfg.uwTimeSec = 10;
stuTimerCfg.ubAction = PAIR_BYPASS_ENABLE;
stuTimerCfg.ubEffect = 0x08; //effect pair
   printf("Slot=%d, Timer=%d, Second=%d, Action=%d, Pair/Port Effect %02X\n",
            bSlot, stuTimerCfg.ubTimerNo, stuTimerCfg.uwTimeSec,
            stuTimerCfg.ubAction, stuTimerCfg.ubEffect);
iRet = LMB_LBP_TimerConfig(bSlot, stuTimerCfg);
if ( iRet == ERR_Success ) {
  printf("---> set Slot1 Time1 10 seconds timeout pair4 bypass enable\n");
  LMB_LBP_TimerStart(bSlot, stuTimerCfg.ubTimerNo);
  printf("Wait 5 seconds....... tick 1\n");   sleep(5);
  LMB_LBP_TimerTick(bSlot, stuTimerCfg.ubTimerNo);
  printf("Wait 5 seconds....... tick 2\n");   sleep(5);
  LMB_LBP_TimerTick(bSlot, stuTimerCfg.ubTimerNo);
  printf("----> stop timer\n");
  LMB_LBP_TimerStop(bSlot, stuTimerCfg.ubTimerNo);
  printf("hit <Enter> to start timer !!!\n");   getchar();
  LMB_LBP_TimerStart(bSlot, stuTimerCfg.ubTimerNo);
  printf("Wait 12 seconds....... wait timeout\n");
  sleep(12);
}
else
  printf("Error ===> pairs setting may be not exist\n");
   LMB_DLL_DeInit();
   return 0;
}
```

LMB_LBP_TimerStart

The **LMB_LBP_TimerStart** function is starting the controller timer function.

Syntax

int32_t LMB_LBP_TimerStart(uint8_t ubSlot, uint8_t ubTimerNo);

Parameters

ubSlot

[in] assign the slot for device access

ubTimerNo

[out] assign timer of LAN bypass controller

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
   ERR_NotSupport: this platform is not support this function
```

Remarks

(none)

Example

```
Refer function LMB_LBP_TimerrConfig example.
```

LMB_LBP_TimerStop

The **LMB_LBP_TimerStop** function is stopped the controller timer function.

Syntax

int32_t LMB_LBP_TimerStop(uint8_t ubSlot, uint8_t ubTimerNo);

Parameters

ubSlot

[in] assign the slot for device access

ubTimerNo

[out] assign timer of LAN bypass controller

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
   ERR_NotSupport: this platform is not support this function
```

Remarks

(none)

Example

```
Refer function LMB_LBP_TimerrConfig example.
```

LMB_LBP_TimerTick

The **LMB_LBP_TimerTick** function reloads the timer then re-compute it.

Syntax

int32_t LMB_LBP_TimerTick(uint8_t ubSlot, uint8_t ubTimerNo);

Parameters

ubSlot

[in] assign the slot for device access

ubTimerNo

[out] assign timer of LAN bypass controller

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
   ERR_NotSupport: this platform is not support this function
```

Remarks

(none)

Example

```
Refer function LMB_LBP_TimerrConfig example.
```

MB_LBP_TimerStatus

The **LMB_LBP_TimerStatus** function gets the Timer current status.

Syntax

int32_t LMB_LBP_TimerStatus(uint8_t ubSlot, uint8_t ubTimerNo,
uint8_t *pubStatus);

Parameters

ubSlot

[in] assign the slot for device access

ubTimerNo

[out] assign timer of LAN bypass controller

pubStatus

[in] obtain timer current status

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: this platform is not support this function

Remarks

(none)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
  int main(){
int32_t iRet;
uint8_t ubStatus;
  LMB_DLL_Init();
  iRet =  LMB_LBP_TimerStatus(2, 1, &ubStatus);
  if ( iRet == ERR_Success )  {
    if (ubStatus ==0 )     printf("Slot2, Timer1, Stopped\n");
    else if (ubStatus==1 ) printf("Slot2, Timer1, Running\n");
    else if (ubStatus==2 ) printf("Slot2, Timer1, Expired\n");
    else          printf("Unknown status\n");
      }
      LMB_DLL_DeInit();
      return 0;
  }
```

Serial EEPROM

LMB_EEP_QueryDevices

The **LMB_EEP_QueryDevices** function gets platform installed controller devices.

Syntax

int32_t LMB_EEP_QueryDevices(uint16_t* puwSlotsDev);

Parameters

puwSlotsDev

[in] obtain this platform devices installed status.

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_NotSupport: function not support
```

Remarks

```
None
```

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
uint16_t uwSlotsDev;
  LMB_DLL_Init();
  LMB_EEP_QueryDevices(&uwSlotsDev);
  if ( iRet == ERR_Success ) {
    printf("Current EEPROM devices list:\n");
    if (uwSlotsDev & 0x0001 ) printf("    SLOT_ONBOARD\n");
    if (uwSlotsDev & 0x0002 ) printf("    SLOT_INDEX_1\n");
    if (uwSlotsDev & 0x0004 ) printf("    SLOT_INDEX_2\n");
    if (uwSlotsDev & 0x0008 ) printf("    SLOT_INDEX_3\n");
    if (uwSlotsDev & 0x0010 ) printf("    SLOT_INDEX_4\n");
    if (uwSlotsDev & 0x0020 ) printf("    SLOT_INDEX_5\n");
    if (uwSlotsDev & 0x0040 ) printf("    SLOT_INDEX_6\n");
    if (uwSlotsDev & 0x0080 ) printf("    SLOT_INDEX_7\n");
    if (uwSlotsDev & 0x0100 ) printf("    SLOT_INDEX_8\n");
  }
      LMB_DLL_DeInit();
      return 0;
  }
```

LMB_EEP_WriteByte

The **LMB_EEP_WriteByte** function write byte data to EEPROM device.

Syntax

int32_t LMB_EEP_WriteByte(uint8_t ubDeviceNo, uint32_t udwAddr,
uint8_t ubData);

Parameters

ubDeviceNo

[out] assign the Serial EEPROM device.

udwAddr

[out] assign the location to write.

ubData

[out] byte data for writing

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_NotSupport: function not support
```

Remarks

```
None
```

Example

```
  #include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
  int main(){
int32_t iRet;
uint8_t aryData[0x20];
  LMB_DLL_Init();
      //erase onboard eeprom
      LMB_EEP_Erase(0);
      //Write data
      printf("Write addr.0x00,data 0x41\n");
  LMB_EEP_WriteByte(0, 0, 0x41);
      printf("Write addr.0x04,data 0x3132\n");
  LMB_EEP_WriteWord(0, 0x04, 0x3132);
      printf("Write addr.0x08, data 0x61626364\n");
  LMB_EEP_WriteDWord(0, 0x08, 0x61626364);
      printf("Write addr.0x10, string AbCdEfgHiJk\n")
      LMB_EEP_WriteBlock(0, 0x10, 11, "AbCdEfgHiJk" );
      //Read data
  LMB_EEP_ReadByte(0, 0, &aryData[0]);
  printf("Read Addr.0x00,data=%02X\n", aryData[0]);
  LMB_EEP_ReadWord(0, 4, (uint16_t*)&aryData[4]);
  printf("Read Addr.0x04,data=%04X\n", *(uint16_t*)&aryData[4]);
      LMB_EEP_ReadDWord(0, 0x08, (uint32_t*)&aryData[8]);
  printf("Read Addr.0x08,data=%08X\n",*(uint32_t*)&aryData[8]);
      LMB_EEP_ReadBlock(0, 0x10, 11,  &aryData[0x10]);
  printf("Read Addr.0x10,data=%s\n", &aryData[0x10]);
      LMB_DLL_DeInit();
      return 0;
  }
```

LMB_EEP_WriteWord

The **LMB_EEP_WriteWord** function write word data to EEPROM device.

Syntax

int32_t LMB_EEP_WriteWord(uint8_t ubDeviceNo, uint32_t udwAddr,
uint16_t uwData);

Parameters

ubDeviceNo

[out] assign the Serial EEPROM device.

udwAddr

[out] assign the location to write.

uwData

[out] word data for writing

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

None

Example

Refer function LMB_EEP_WriteByte example.

LMB_EEP_WriteDWord

The **LMB_EEP_WriteDWord** function write dword data to EEPROM device.

Syntax

int32_t LMB_EEP_WriteDWord(uint8_t ubDeviceNo, uint32_t udwAddr,
uint32_t udwData);

Parameters

ubDeviceNo

[out] assign the Serial EEPROM device.

udwAddr

[out] assign the location to write.

udwData

[out] double word data for writing

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_NotSupport: function not support
```

Remarks

```
None
```

Example

```
Refer function LMB_EEP_WriteByte example.
```

LMB_EEP_WriteBlock

The **LMB_EEP_WriteBlock** function write block data to EEPROM device.

Syntax

int32_t LMB_EEP_WriteBlock(uint8_t ubDeviceNo, uint16_t udwAddr,
uint16_t uwLength, uint8_t* pubBlock);

Parameters

ubDeviceNo

[out] assign the Serial EEPROM device.

udwAddr

[out] assign the location to write.

uwLength

[out] double word data for writing

pubBlock

[out] byte block data for writing

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_NotSupport: function not support
```

Remarks

```
None
```

Example

```
Refer function LMB_EEP_WriteByte example.
```

LMB_EEP_ReadByte

1. The **LMB_EEP_ReadByte** function write byte data from EEPROM device.

Syntax

int32_t LMB_EEP_ReadByte(uint8_t ubDeviceNo, uint32_t udwAddr,
uint8_t* pubData);

Parameters

ubDeviceNo

[out] assign the Serial EEPROM device.

udwAddr

[out] assign the location to write.

pubData

[in] obtain byte data for reading

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_NotSupport: function not support
```

Remarks

```
None
```

Example

```
Refer function LMB_EEP_WriteByte example.
```

LMB_EEP_ReadWord

The **LMB_EEP_ReadWord** function read word data from EEPROM device.

Syntax

int32_t LMB_EEP_ReadWord(uint8_t ubDeviceNo, uint32_t udwAddr,
uint16_t* puwData);

Parameters

ubDeviceNo

[out] assign the Serial EEPROM device.

udwAddr

[out] assign the location to write.

puwData

[in] obtain word data for reading

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_NotSupport: function not support
```

Remarks

```
None
```

Example

```
Refer function LMB_EEP_WriteByte example.
```

LMB_EEP_ReadDWord

The **LMB_EEP_ReadDWord** function read dword data from EEPROM device.

Syntax

int32_t LMB_EEP_ReadDWord(uint8_t ubDeviceNo, uint32_t udwAddr,
uint32_t* pudwData);

Parameters

ubDeviceNo

[out] assign the Serial EEPROM device.

udwAddr

[out] assign the location to write.

pudwData

[in] obtain double word data for reading

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_NotSupport: function not support
```

Remarks

```
None
```

Example

```
Refer function LMB_EEP_WriteByte example.
```

LMB_EEP_ReadBlock

The **LMB_EEP_ReadBlock** function read block data from EEPROM device.

Syntax

int32_t LMB_EEP_ReadBlock(uint8_t ubDeviceNo, uint32_t udwAddr,
uint16_t uwLength, uint8_t* pubBlock);

Parameters

ubDeviceNo

[out] assign the Serial EEPROM device.

udwAddr

[out] assign the location to write.

uwLength

[out] obtain double word data for reading

pubBlock

[in] obtain byte block data for reading

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_NotSupport: function not support
```

Remarks

```
None
```

Example

```
Refer function LMB_EEP_WriteByte example.
```

LMB_EEP_Erase

The **LMB_EEP_Erase** function erases EPROM device.

Syntax

int32_t LMB_EEP_Erase(uint8_t ubDeviceNo);

Parameters

ubDeviceNo

[out] assign the Serial EEPROM device.

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_NotSupport: function not support
```

Remarks

```
Erase EEPROM all data to 0x00 value
```

Example

```
Refer function LMB_EEP_WriteByte example.
```

LCD Module

LMB_LCM_DeviceOpen

The **LMB_LCM_DeviceOpen** function opens and connected LCD module.

Syntax

int32_t LMB_LCM_DeviceOpen(void);

Parameters

(none)

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

None

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
uint8_t aryData[0x20];
LMB_DLL_Init();
    iRet = LMB_LCM_DeviceOpen();
    if ( iRet != ERR_Success ) {
     printf("LCD Module not exist\n");
     return -1;
    }
    … … … … … …
LMB_LCM_DeviceClose();
    LMB_DLL_DeInit();
    return 0;
}
```

LMB_LCM_DeviceClose

The **LMB_LCM_DeviceClose** function closes and disconnected LCD module.

Syntax

int32_t LMB_LCM_DeviceClode(void);

Parameters

(none)

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_NotSupport: function not support
```

Remarks

None

Example

Refer function LMB_EEP_WriteByte example.

LMB_LCM_DeviceInfo

The **LMB_LCM_DeviceInfo** function gets platform LCM device information.

Syntax

int32_t LMB_LCM_DeviceInfo(LCM_INFO* pstuLcmInfo);

Parameters

pstuLcmInfo

[in] obtain this LCM device information.

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

None

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
LCM_INFO stuLcmInfo;
LMB_DLL_Init();
    LMB_LCM_DeviceOpen();
    iRet = LMB_LCM_DeviceInfo(&stuLcmInfo);
    if (iRet == ERR_Success ){
     printf("LCM Mode No. is %04X\n", stuLcmInfo.uwModeNo);
        printf("LCM Firmware Ver. is %04X\n", stuLcmInfo.uwVersion);
        printf("LCM Speed =%d\n", stuLcmInfo.udwBaudrate);
    }
LMB_LCM_DeviceClose();
    LMB_DLL_DeInit();
    return 0;
}
```

LMB_LCM_LightCtrl

The **LMB_LCM_LightCtrl** function sets LCD module light on/off status.

Syntax

int32_t LMB_LCM_LightCtrl(uint8_t ubOnOff);

Parameters

ubOnOff

[out] control the LCM light status. [range: 0/1] (Off/On)

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

None

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
LCM_INFO stuLcmInfo;
LMB_DLL_Init();
    LMB_LCM_DeviceOpen();
    iRet = LMB_LCM_LightCtrl(DISABLE);
    if (iRet == ERR_Success ) printf("LCM Light Off\n");
    sleep(2);//delay 2 seconds
    iRet = LMB_LCM_LightCtrl(ENABLE);
    if (iRet == ERR_Success ) printf("LCM Light On\n");
LMB_LCM_DeviceClose();
    LMB_DLL_DeInit();
    return 0;
}
```

LMB_LCM_SetCursor

The **LMB_LCM_SetCursor** function writes string to LCD module.

Syntax

int32_t LMB_LCM_SetCursor(uint8_t ubColumn, uint8_t ubRow);

Parameters.

ubColumn

[out] assigned column value of LCM display cursor.

ubRow

[out] assigned row value of LCM display cursor

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

None

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
LMB_DLL_Init();
    LMB_LCM_DeviceOpen();
    iRet = LMB_LCM_SetCursor(5,1); //column 5, row 1
    if (iRet == ERR_Success )
        printf("LCM set cursor to column 5, row 1\n");
LMB_LCM_DeviceClose();
    LMB_DLL_DeInit();
    return 0;
}
```

LMB_LCM_WriteString

The **LMB_LCM_WriteString** function writes string to LCD module.

Syntax

int32_t LMB_LCM_WriteString(int8_t* pstrubString);

Parameters

pstrubString

[out] message string to LCM display.

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

Please refer appendix: LCD Module Character Table

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
LMB_DLL_Init();
    LMB_LCM_DeviceOpen();
    iRet = LMB_LCM_WriteString("123ABCD");
    if (iRet == ERR_Success )
        printf("Write string to LCM is successful\n");
LMB_LCM_DeviceClose();
    LMB_DLL_DeInit();
    return 0;
}
```

LMB_LCM_DisplayClear

The **LMB_LCM_DisplayClear** function clears LCM display.

Syntax

int32_t LMB_LCM_DisplayClear(void);

Parameters

(none)

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

None

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
LMB_DLL_Init();
    LMB_LCM_DeviceOpen();
    iRet = LMB_LCM_DisplayClear();
    if (iRet == ERR_Success ) printf("Clear display successful\n");
LMB_LCM_DeviceClose();
    LMB_DLL_DeInit();
    return 0;
}
```

LMB_LCM_Brightness

The **LMB_LCM_Brightness** function sets the brightness of LCD module.

Syntax

int32_t LMB_LCM_Brightness(uint8_t ubBrightness);

Parameters

ubBrightness

[out] set brightness value of LCM. [range 1 ~ 8]

When value is out of range, will auto set to maximum or minimum.

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

None

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
LMB_DLL_Init();
    LMB_LCM_DeviceOpen();
    iRet = LMB_LCM_Brightness(1);
    if (iRet == ERR_Success ) printf("Set brightness step 8\n");
    iRet = LMB_LCM_Brightness(8);
    if (iRet == ERR_Success ) printf("Set brightness step 1\n");
LMB_LCM_DeviceClose();
    LMB_DLL_DeInit();
    return 0;
}
```

LMB_LCM_Reset

The **LMB_LCM_Reset** function resets the LCD module.

Syntax

int32_t LMB_LCM_Reset(void);

Parameters

(none)

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

None

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
LMB_DLL_Init();
    LMB_LCM_DeviceOpen();
    iRet = LMB_LCM_Reset();
    if (iRet == ERR_Success ) printf("Soft Reset LCM\n");
LMB_LCM_DeviceClose();
    LMB_DLL_DeInit();
    return 0;
}
```

LMB_LCM_SetSpeed

The **LMB_LCM_SetSpeed** function sets communication baud rate of LCM.

Syntax

int32_t LMB_LCM_SetSpeed(uint32_t udwBaudrate);

Parameters

udwBaudrate

[out] set communication baudrate of LCM.

Supports speed: 115200, 57600, 38400, 19200, 9600

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

After changed connect speed, the application needs close LCM device and re-open LCM device again.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
LMB_DLL_Init();
    LMB_LCM_DeviceOpen();
    iRet = LMB_LCM_SetSpeed(115200);
    if (iRet == ERR_Success ) printf("set LCM speed is 115200\n");
LMB_LCM_DeviceClose();
    LMB_DLL_DeInit();
    return 0;
}
```

LMB_LCM_WrapCtrl

The **LMB_LCM_WrapCtrl** function sets LCM line wrap format.

Syntax

int32_t LMB_LCM_WrapCtrl(uint8_t ubOnOff);

Parameters

ubOnOff

[out] control wrap is on or off. [range: 0/1] (Off/On)

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

None

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
LMB_DLL_Init();
    LMB_LCM_DeviceOpen();
    iRet = LMB_LCM_WrapCtrl(DISABLE);
    if (iRet == ERR_Success ) printf("Line wrap is off\n");
    iRet = LMB_LCM_WrapCtrl(ENABLE);
    if (iRet == ERR_Success ) printf("Line wrap is on\n");
LMB_LCM_DeviceClose();
    LMB_DLL_DeInit();
    return 0;
}
```

LMB_LCM_CursorModeCtrl

The **LMB_LCM_CursorModeCtrl** function sets LCM cursor format.

Syntax

int32_t LMB_LCM_CursorModeCtrl(uint8_t ubCursorMode);

Parameters

ubCursorMode

[out] control cursor display mode.

0	Off
1	Underline
2	Blinking Block

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

Set cursor display type will auto set cursor to HOME(0,0) position

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
LMB_DLL_Init();
    LMB_LCM_DeviceOpen();
    iRet = LMB_LCM_CursorModeCtrl(2);
    if (iRet == ERR_Success ) printf("Cursor is blinking & block\n");
    iRet = LMB_LCM_CursorModeCtrl(1);
    if (iRet == ERR_Success ) printf("Cursor is underline\n");
LMB_LCM_DeviceClose();
    LMB_DLL_DeInit();
    return 0;
}
```

LMB_LCM_KeysStatus

The **LMB_LCM_KeysStatus** function reads LCM key status.

Syntax

int32_t LMB_LCM_KeysStatus(uint8_t * pubKeys);

Parameters

pubKeys

[in] obtain keys status of LCD module.

Bit 0	Key 1 pressed
Bit 1	Key 2 pressed
Bit 2	Key 3 pressed
Bit 3	Key 4 pressed

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_NotSupport: function not support
ERR_Error: no any key pressed
```

Remarks

none

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
uint8_t ubKeys;
LMB_DLL_Init();
    LMB_LCM_DeviceOpen();
    iRet = LMB_LCM_KeysStatus(&ubKeys);
    if (iRet == ERR_Success )
        printf("key was pressed: %02X\n",ubKeys);
LMB_LCM_DeviceClose();
    LMB_DLL_DeInit();
    return 0;
}
```

LMB_LCM_KeysCallback

The **LMB_LCM_KeysCallback** function hooks keys event callback of LCM

Syntax

int32_t LMB_LCM_KeysCallback(LCMKEY_CALLBACK pCallback,
uint16_t uwmSec);

Parameters

pCallback

[out] hook callback function pointer for LCM keys status changed event.

uwmSec

[out] setting period of time for checking status changed
[range: 10 ~ 999ms]

Return Value

```
  ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
    ERR_BusyInUses: callback function is running now
    ERR_Invalid: parameter is out of range
```

Remarks

Current supports case open, software reset button and power status changed.

When pCallback is NULL pointer this function will disable.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
void lcmCallback(LCMKEY_MSG pLcmMsg){
       printf("Keys=%02X,status=%02X,\
       time = %04d/%02d/%02d %02d:%02d:%02d\n",\
       pLcmMsg.ubKeys, pLcmMsg.ubStatus, pLcmMsg.stuTime.year, \
       pLcmMsg.stuTime.month, pLcmMsg.stuTime.day, \
       pLcmMsg.stuTime.hour, pLcmMsg.stuTime.minute, \
       pLcmMsg.stuTime.second”);
  }
int main(){
  LMB_DLL_Init();
  LMB_LCM_KeysCallback(lcmCallback, 150);
  While (1) {
  ……
       }
       LMB_DLL_DeInit();
       return 0;
  }
```

LMB_LCM_StartupMsg

The **LMB_LCM_StartupMsg** function set default message after LCM reset.

Syntax

int32_t LMB_LCM_StartupMsg(uint8_t * pubMsg, uint8_t ubLength);

Parameters

pubMessage

[out] default display message after LCM reset.

ubLength

[out] message string length.

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_NotSupport: function not support
```

Remarks

```
The string maximum length depends on LCD module,
normally is 40 characters.
```

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
uint8_t *pstrMsg = (uint8_t*)“Lanner Electronics Inc.”;
LMB_DLL_Init();
    LMB_LCM_DeviceOpen();
    LMB_LCM_StartupMsg(pstrMsg, strlen((char*)pstrMsg);
    LMB_LCM_Reset();
LMB_LCM_DeviceClose();
    LMB_DLL_DeInit();
    return ;
}
```

Power Supply

LMB_PSU_QueryDevices

The **LMB_PSU_QueryDevices** function gets platform power supply devices.

Syntax

int32_t LMB_PWR_QueryDevices(uint16_t* puwPsuDev);

Parameters

puwPsuDev

[in] obtain this platform devices installed status.

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

None

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
uint16_t uwPsuDev;
LMB_DLL_Init();
        iRet = LMB_PSU_QueryDevices(&uwPsuDev);
        if (iRet == ERR_Success )
                printf("PSU devices = %02X\n", uwPsuDev);
        LMB_DLL_DeInit();
        return 0;
}
```

LMB_PSU_DeviceInfo

The **LMB_PSU_DeviceInfo** function gets power supply information

Syntax

int32_t LMB_PSU_DeviceInfo(PSU_INFO* pstuPsuInfo);

Parameters

pstuPsuInfo

[in/out] obtain this power supply device information

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

ERR_Invalid: parameter is out of range

Remarks

The "ubPsuNo" parameter of PSU_INFO needs to assign PSU number.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
PSU_INFO stuPsuInfo;
LMB_DLL_Init();
    stuPsuInfo.ubPsuNo = 1;
    iRet = LMB_PSU_DeviceInfo(&stuPsuInfo);
    if (iRet == ERR_Success ) {
       printf("PSU 1 Device information\n");
       printf(" MFRID = %s\n", stuPsuInfo.strubMfrId);
       printf(" MFRMODEL = %s\n", stuPsuInfo.strubMfrModel);
       printf(" MFRSERIAL = %s\n", stuPsuInfo.strubMfrSerial);
}
    LMB_DLL_DeInit();
    return 0;
}
```

LMB_PSU_SensorInfo

The **LMB_PSU_SensorInfo** function gets temperature and fan speed information.

Syntax

int32_t LMB_PSU_SensorInfo(PSU_SENSORS* pstuPsuSensors);

Parameters

pstuPsuSensors

[in/out] obtain this power supply sensors information

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

ERR_Invalid: parameter is out of range

Remarks

The "ubPsuNo" parameter of PSU_SENSORS needs to assign PSU number.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
PSU_SENSORS stuPsuSensors;
LMB_DLL_Init();
    stuPsuSensors.ubPsuNo = 1;
    iRet = LMB_PSU_SensorInfo(&stuPsuSensors);
    if (iRet == ERR_Success ) {
       printf("PSU 1 Sensor information\n");
       printf(" Temp-1 = %4.1f\n", stuPsuSensors.fTemp_1);
       printf(" Temp-2 = %4.1f\n", stuPsuSensors.fTemp_2);
       printf(" FanSpeed = %d\n", stuPsuSensors.uwFanRpm);
}
    LMB_DLL_DeInit();
    return 0;
}
```

LMB_PSU_WattsInfo

The **LMB_PSU_WattsInfo** function gets power supply Watts information.

Syntax

int32_t LMB_PSU_WattsInfo(PSU_WATTS* pstuPsuWatts);

Parameters

pstuPsuWatts

[in/out] obtain this power supply Watts, Volt and Amperes information

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

The "ubPsuNo" parameter of PSU_WATTS needs to assign PSU number.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
PSU_WATTS stuPsuWatts;
LMB_DLL_Init();
    stuPsuWatts.ubPsuNo = 1;
    stuPsuWatts.ubPsuNo = PSU_WATTS_OUTPUT;
    iRet = LMB_PSU_WattsInfo(&stuPsuWatts);
    if (iRet == ERR_Success ) {
       printf("PSU 1 Output Watts information\n");
       printf(" Volts = %4.3f\n", stuPsuWatts.fVolts);
       printf(" Amperes = %4.3f\n", stuPsuWatts.fAmperes);
       printf(" Watts = %4.3f\n", stuPsuWatts.Watts);
}
    LMB_DLL_DeInit();
    return 0;
}
```

LMB_PSU_Status

The **LMB_PSU_Status** function gets power supply status information.

Syntax

int32_t LMB_PSU_Status(uint8_t ubPsuNo, uint16_t* puwStatus);

Parameters

ubPsuNo

[out] assign the PSU number

puwStatus

[in] obtain this power supply status word

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_NotSupport: function not support
```

Remarks

```
Please refer the "PSU Status Bit define" in chapter 2.1
```

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(){
int32_t iRet;
uint16_t uwStatus =0;
LMB_DLL_Init();
    iRet = LMB_PSU_Status(1, &uwStatus);
    if (iRet == ERR_Success ) {
       printf("PSU 1 status Status = %04X\n", uwStatus);
}
    LMB_DLL_DeInit();
    return 0;
}
```

LMB_PSU_IntrCallback

The **LMB_PSU_IntrCallback** function hooks power supply alarm callback

Syntax

int32_t LMB_PSU_IntrCallback(INTRUSION_CALLBACK pCallback
uint16_t uwmSec);

Parameters

pCallback

[out] hook callback function pointer for status changed event.

uwmSec

[out] setting period of time for checking status changed
[range: 10 ~ 999ms]

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_BusyInUses: callback function is running now
ERR_Invalid: parameter is out of range
```

Remarks

Supports case open, software reset button and power status changed and power supply fault event (refer "Intrusion Item Define (bit)" in chapter 2.1)

When pCallback is NULL pointer this function will disable.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
void psuCallback(INTRUSION_MSG pIntr){
       printf("item=%04X,status=%04X,\
       time = %04d/%02d/%02d %02d:%02d:%02d\n",\
       pInt.udwOccurItem, pIntr.udwStatus, pIntr.stuTime.year, \
       pIntr.stuTime.month, pIntr.stuTime.day, pIntr.stuTime.hour,\
       pIntr.stuTime.minute, pIntr.stuTime.second”);
  }
int main(){
  LMB_DLL_Init();
  LMB_PSU_IntrCallback(psuCallback, 100);
  While (1) {
  ……
       }
       LMB_DLL_DeInit();
       return 0;
}
```

Software Reset Button

LMB_SWR_GetStatus

The **LMB_SWR_GetStatus** function gets software reset button status

Syntax

int32_t LMB_SWR_GetStatus(uint8_t *pubStatus);

Parameters

pubStatus

[in] indicate software reset button status.

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

none

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
  {
  int32_t iRet;
  uint8_t bStatus= 0;
      LMB_DLL_Init();
      iRet = LMB_SW_GetStatus(&bStatus);
      if (iRet == ERR_Success)
              printf("Software reset button status is %d\n", bStatus);
      LMB_DLL_DeInit();
      return 0;
  }
```

LMB_SWR_IntrCallback

The **LMB_SWR_IntrCallback** function hooks some status change callback

Syntax

int32_t LMB_SWR_IntrCallback(INTRUSION_CALLBACK pCallback
uint16_t uwmSec);

Parameters

pCallback

[out] hook callback function pointer for status changed event.

uwmSec

[out] setting period of time for checking status changed
[range: 10 ~ 999ms]

Return Value

```
  ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
    ERR_BusyInUses: callback function is running now
    ERR_Invalid: parameter is out of range
```

Remarks

Supports case open, software reset button and power status changed and power supply fault event (refer "Intrusion Item Define (bit)" in chapter 2.1)

When pCallback is NULL pointer this function will disable.

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
void swrCallback(INTRUSION_MSG pIntr){
       printf("item=%04X,status=%04X,\
       time = %04d/%02d/%02d %02d:%02d:%02d\n",\
       pInt.udwOccurItem, pIntr.udwStatus, pIntr.stuTime.year, \
       pIntr.stuTime.month, pIntr.stuTime.day, pIntr.stuTime.hour,\
       pIntr.stuTime.minute, pIntr.stuTime.second”);
  }
int main(){
  LMB_DLL_Init();
  LMB_SWR_IntrCallback(swrCallback, 100);
  While (1) {
  ……
       }
       LMB_DLL_DeInit();
       return 0;
}
```

APPENDIX

SDK Package Install & Build guide

Library and Device dependency:

Device node require

/dev/mem

x86_x64 (amd64) library (base functions)

linux-vdso.so.1

libdl.so.2

libstdc++.so.6

libm.so.6

libc.so.6

libgcc_s.so.1

/lib64/ld-linux-x86-64.so.2

i386/i686 library (base functions)

linux-gate.so.1

libdl.so.2

libstdc++.so.6

libm.so.6

libc.so.6

libgcc_s.so.1

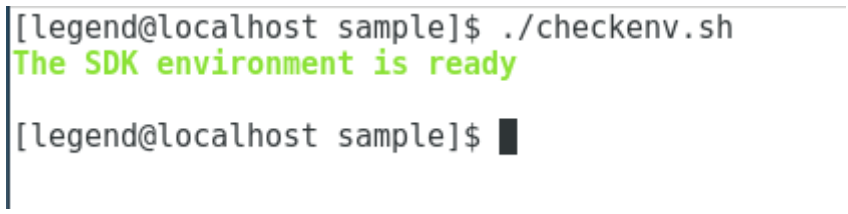
/lib/ld-linux.so.2

Checking tool:

The shell script file “amd64/sample/checkenv.sh” will report

the all libraries dependency result

when the environment is ok like below picture



If any library is found missing in the system, the message shows:

```
[legend@localhost sample]$ ./checkenv.sh
Warning !!! libraries is not exist
==================================
../sdk-lanner-eep-0.0.1.so :: libstdc++.so.6 => not found

Warning !!! libraries is not exist
==================================
../sdk-lanner-lbp-0.0.1.so :: libstdc++.so.6 => not found

Warning !!! libraries is not exist
==================================
../sdk-lanner-lcm-0.0.1.so :: libstdc++.so.6 => not found

Warning !!! libraries is not exist
==================================
../sdk-lanner-psu-0.0.1.so :: libstdc++.so.6 => not found

Warning !!! libraries is not exist
==================================
../sdk-lmbapi-0.1.27.so :: libstdc++.so.6 => not found

Warning !!! libraries is not exist
==================================
../sdk-nca6210-board-0.1.37.so :: libstdc++.so.6 => not found

Warning !!! libraries is not exist
==================================
../sdk-nca6210-gpio-0.0.1.so :: libstdc++.so.6 => not found

Warning !!! libraries is not exist
==================================
../sdk-nca6210-hwm-0.0.1.so :: libstdc++.so.6 => not found

Warning !!! libraries is not exist
==================================
../sdk-nca6210-sled-0.0.1.so :: libstdc++.so.6 => not found

Warning !!! libraries is not exist
==================================
../sdk-nca6210-swr-0.0.1.so :: libstdc++.so.6 => not found

Warning !!! libraries is not exist
==================================
../sdk-nca6210-wdt-0.0.1.so :: libstdc++.so.6 => not found

[legend@localhost sample]$
```

Package directory introduce:

"amd64" directory

The libraries of product and sample code for x86_x64 Linux

"i386" directory

The libraries of product and sample code for i686/i386 Linux

"include" directory

The include file of the SDK library

"sample" directory

The source code of the sample program

Build sample

Require g++ compiler

install g++ method:

CentOS: $ sudo yum install gcc-c++

Ubuntu: $ sudo apt-get install g++

Build:

$ cd sample

$ ./make-sample.sh

Notes: this script for both x86_x64(amd64) and i386/686 architecture,

please mark the script which the architecture is not supported.

Install & Running

1. Copy amd64 or i386 directory all library file with a link to /usr/lib or your directory path
2. $ export LD_LIBRARY_PATH=*your_path*:$LD_LIBRARY_PATH
3. Running the sample program or yours

Parameter & Variable Naming Rule

bValue : 8 bit size variable with sign
ubValue: 8 bit size variable without sign
wValue: 16 bit size variable with sign
uwValue: 16 bit size variable without sign
dwValue: 32 bit size variable with sign
udwValue: 32 bit size variable without sign
fValue: float type variable with sign
dlValue: 64 bit size variable with sign
udlValue: 64 bit size variable without sign

pbValue : pointer of 8 bit size variable with sign
pubValue: pointer of 8 bit size variable without sign
pwValue: pointer of 16 bit size variable with sign
puwValue: pointer of 16 bit size variable without sign
pdwValue: pointer of 32 bit size variable with sign
pudwValue: pointer of 32 bit size variable without sign
pfValue: float type variable with sign
pdlValue: pointer of 64 bit size variable with sign
pudlValue: pointer of 64 bit size variable without sign

strbString: 8bit with sign string or array pointer
strubString: 8bit without sign string or array pointer
strwString: 16bit with sign string or array pointer
struwString: 16bit without sign string or array pointer
strdwString: 32bit with sign string or array pointer
strudwString: 32bit without sign string or array pointer

stuStruct: structure name
pstuStruct: pointer of structure name

LCD Module Character Table

NO.7066-0A