Lanner

Hardware Platforms for Network Computing

Lanner SDK User Guide For LEC-2580

Version: 1.02

Date of Release: 2018-08-07

Copyright and Trademarks

Revision History

Version	Date	Descriptions
1.00	05/11/2018	Create for LEC-2580 serial
1.01	07/20/2018	Revised define and macro information
1.02	08/03/2018	Revised character decorations

Table of Contents

Chapter 1 Introduction 6

Benefits 6

BIOS Support Information 7

Operating Systems 7

Chapter 2 Definition 8

Constant and Macro Define 8

Structure Define 10

Status Code 11

Chapter 3 API Functions 13

Library Control 13

LMB_DLL_Init 13

LMB_DLL_DeInit 14

LMB_DLL_Version 15

Hardware Monitor 16

LMB_HWM_DevInfo 16

LMB_HWM_GetCpuTemp 17

LMB_HWM_GetSysTemp 18

LMB_HWM_GetVcore 19

LMB_HWM_Get5V 20

LMB_HWM_Get3V3 21

LMB_HWM_Get3V3sb 22

Watch Dog Timer 23

LMB_WDT_QueryInfo 23

LMB_WDT_Config 24
LMB_WDT_Start 26
LMB_WDT_Stop 27
LMB_WDT_Tick 28
Power over Ethernet 29
LMB_POE_Query_Devices 29
LMB_POE_SetPortPower 30
LMB_POE_GetPortStatus 31
Appendix 32
SDK Package Installation & Building Guide 32
Library and Device Dependency 32
Checking Tool 32
Package Directory Introduce 34
Build Sample 34
Install & Run 34
Parameter & Variable Naming Rule 35

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 corresponding tools to prevent unexpected downtime causing from environment setup matters.

Enhances Lanner Platform Reliability

Lanner SDK is released after a series of reliability tests 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-initialization 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.

BIOS Support Information

Version 2.02 and later

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

#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
#define HWM_FAN_5     0x00400000
#define HWM_FAN_6     0x00800000
#define HWM_FAN_7     0x01000000
#define HWM_FAN_8     0x02000000

/* HWM Fan sequence */
#define FAN_A    1
#define FAN_B    2
```

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;
```

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 initializes 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 releases 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-release board level library.

Example

Refer to 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] obtains 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] obtains the monitor devices information.

Please refer to Constant and Macro Define in Chapter 2.

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] selects CPU number.

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] selects System sensor.

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] selects CPU number.

pfVoltage

[in] obtains 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_Get5V

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

Syntax

int32_t LMB_HWM_Get5V(float* pfVoltage);

Parameters

pfVoltage

[in] obtains 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] obtains 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_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] obtains 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;
}
```

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 configures the Watch Dog Timer.

Syntax

int32_t LMB_WDT_Config(uint16_t uwCount, uint8_t ubTimeBase);

Parameters

uwCount

[out] the value sets the timer count down.

ubTimeBase

[out] the value selects 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 starts 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 to function `LMB_WDT_Config` example.

LMB_WDT_Stop

The **LMB_WDT_Stop** function stops 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 to function `LMB_WDT_Config` example.

LMB_WDT_Tick

The **LMB_WDT_Tick** function reloads the timer and 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 to function `LMB_WDT_Config` example.

Power over Ethernet

LMB_POE_Query_Devices

The **LMB_POE_QueryDevices** function queries the PoE supported of LAN ports.

Syntax

int32_t LMB_POE_QueryDevices(uint32_t *pudwPorts);

Parameters

pudwPorts

[in] obtains PoE supported port of LAN ports number.

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_NotSupport: function not support

Remarks

bit0 indicates the LAN1, bit1 indicates LAN2, in a similar fashion bit31 indicates LAN32.

Example

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

`

LMB_POE_SetPortPower

The **LMB_POE_SetPortPower** function enables/disables the PoE port power.

Syntax

int32_t LMB_POE_SetPortPower(uint8_t ubPort, uint8_t ubEnable);

Parameters

ubPort

[out] assigns LAN port number.

When the ubPort assigns 255, will set all PoE LAN ports power.

ubEnable

[out] controls power enabling or disabling.

0	Power Off
1	Power On by auto

Return Value

```
ERR_Success: function is successful
ERR_NotOpened: library not ready or opened yet
ERR_Invalid: parameter is out of range or not a PoE LAN port
```

Remarks

(None)

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{ int32_t iRet;
    LMB_DLL_Init();
    iRet = LMB_POE_SetPortPower(1, 0);
    if ( iRet == ERR_Success ) printf("LAN1 Port Power Off\n");
    iRet = LMB_POE_SetPortPower(3, 1);
    if ( iRet == ERR_Success ) printf("LAN3 Port Power On by Auto\n");
   LMB_DLL_DeInit();
    return 0;
}
```

LMB_POE_GetPortStatus

The **LMB_POE_GetPortStatus** function gets the PoE port power status

Syntax

int32_t LMB_POE_GetPortStatus(uint8_t ubPort, uint32_t* pudwStatus);

Parameters

ubPort

[out] assigns LAN port number.

When the ubPort assigns 255, this function will get all PoE LAN ports power status.

pudwStatus

[in] obtains PoE LAN port power status

bit stat 0	PoE power is disabled
bit stat 1	PoE power is enabled by auto

Return Value

ERR_Success: function is successful

ERR_NotOpened: library not ready or opened yet

ERR_Invalid: parameter is out of range or not a PoE LAN port

Remarks

bit0 indicates the LAN1, bit1 indicates LAN2, in a similar fashion bit31 indicates LAN32

Example

```
#include <stdio.h>
#include <stdlib.h>
#include "lmbinc.h"
int main(int argc, char* argv[])
{ int32_t iRet;
  uint32_t dwStatus =0;
     LMB_DLL_Init();
     iRet = LMB_POE_GetPortStatus(3, &dwStatus);
     if ( iRet == ERR_Success )
           printf("LAN3 Port Power status = %0x8X\n", dwStatus);
     iRet = LMB_POE_ GetPortStatus(255, &dwStatus);
     if ( iRet == ERR_Success )
          printf("All LAN Ports Power status = %0x8X\n", dwStatus);
   LMB_DLL_DeInit();
   return 0;
}
```

APPENDIX

SDK Package Installation & Building Guide

Library and Device Dependency

Device node requires:

/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 all libraries dependency result (the meassage is shown below) when the environment is ready.

```
[legend@localhost sample]$ ./checkenv.sh
The SDK environment is ready

[legend@localhost sample]$
```

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 is for both x86_x64(amd64) and i386/686 architecture, please mark the script which the architecture does not support.

Install & Run

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

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