

# Fintek

# Software Development Kit

## Windows Software Programming Guide

v1.19

Dec. 3, 2025

12/3/2025

**Fintek**

The information contained herein is the exclusive proprietary information of Feature Integration Technology Inc., is provided for customer's internal use only, and shall not be disclosed, distributed to any third party or reproduced in whole or in part by any third party without the prior written permission of Feature Integration Technology Inc. 本資料為精拓科技股份有限公司之智慧財產權，僅供客戶內部使用；非經精拓科技股份有限公司事前書面授權許可，不得透露、散佈予任何第三人或使任何第三人使用本資料之全部或一部，或將之複印、複製或轉變成其它任何形式使用。

## Contents

1.	SDK Driver Install .....	3
1.1	Windows 7/8/8.1/10/11 Series Install.....	3
1.2	Windows XP Install .....	4
2.	LPC DLL Function .....	10
2.1	Support Fintek LPC IC.....	10
2.2	LPC Function .....	11
2.2.1	FintekLPCICSelect.....	11
2.2.2	PORT_LPC_R.....	12
2.2.3	PORT_LPC_W .....	12
2.3	GPIO Function.....	12
2.3.1	GPIO_LPC_R .....	13
2.3.2	GPIO_LPC_W.....	15
2.3.3	SetLPCGpioControl.....	17
2.3.4	GetLPCGpioControl .....	18
2.3.5	SetLPCGpioOutputDataIndividual .....	19
2.3.6	GetLPCGpioOutputDataIndividual .....	20
2.3.7	GetLPCGpioStatusIndividual .....	21
2.3.8	GetLPCGpioStatus .....	22
2.3.9	SetLPCGpioEdgeDetector.....	23
2.3.10	GetLPCGpioEdgeDetector .....	24
2.3.11	SetLPCGpioClearEdgeDetectorStatus .....	24
2.3.12	GetLPCGpioEdgeDetectorStatus.....	25
2.3.13	SetLPCGpioDebounceEnable .....	26
2.3.14	GetLPCGpioDebounceEnable.....	27
2.3.15	SetLPCGpioLevelPulseControl.....	28
2.3.16	GetLPCGpioLevelPulseControl .....	29
2.3.17	SetLPCGpioDebounceTime .....	30
2.3.18	GetLPCGpioDebounceTime.....	31
2.3.19	SetLPCGpioPulseWidth .....	32
2.3.20	GetLPCGpioPulseWidth .....	33
2.3.21	SetLPCGpioInverseEnable .....	34
2.3.22	GetLPCGpioInverseEnable .....	35
2.3.23	SetLPCGpioSMIEnable.....	36
2.3.24	GetLPCGpioSMIEnable .....	37
2.3.25	SetLPCGpioOutputDrivingEnable.....	38
2.3.26	GetLPCGpioOutputDrivingEnable .....	39
2.4	WDT Function: .....	40
2.4.1	WDT_LPC_SetConfig .....	40
2.4.2	WDT_LPC_SetTimeRange .....	42
2.4.3	WDT_LPC_GetConfig .....	43
2.4.4	WDT_LPC_GetTimeRange.....	43

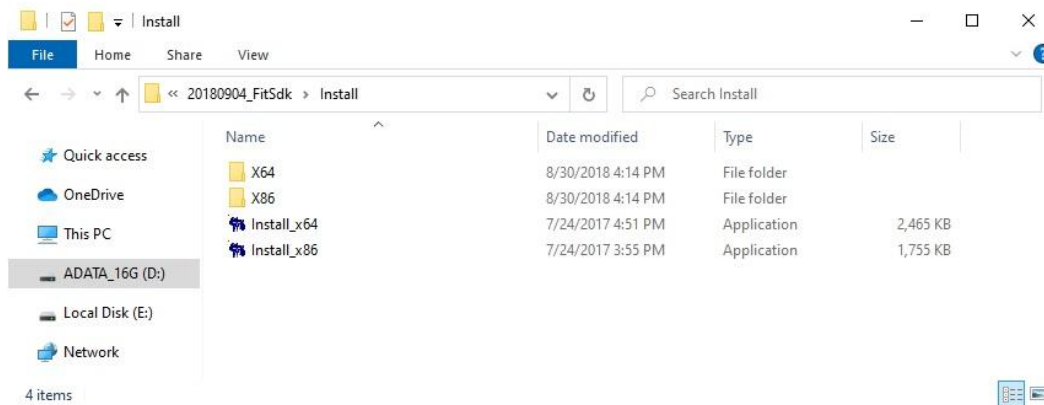
2.4.5	WDT Full Example .....	44
2.5	Fan Control Function .....	46
2.5.1	GetFanMode .....	46
2.5.2	SetFanMode .....	47
2.5.3	GetLPCFanSpeed .....	48
2.5.4	GetLPCFanExpectSpeed .....	49
2.5.5	SetLPCFanExpectSpeed .....	50
2.5.6	GetLPCMaxFanSpeed .....	51
2.5.7	SetLPCMaxFanSpeed .....	52
2.5.8	GetLPCTemperatureThreshold .....	53
2.5.9	SetLPCTemperatureThreshold .....	54
2.5.10	GetLPCFanSpeedSectionValue .....	55
2.5.11	SetLPCFanSpeedSectionValue .....	56
2.6	Temperature Function .....	57
2.6.1	GetLPCTemperatureValue .....	57
2.7	Voltage Function .....	58
2.7.1	GetLPC3VccVoltage .....	58
2.7.2	GetLPCVsbVoltage .....	59
2.7.3	GetLPCVbatVoltage .....	60
2.7.4	GetLPCVinVoltage .....	61
2.7.5	GetLPC5VsbVoltage .....	62
2.7.6	GetLPC5VaVoltage .....	62
2.7.7	GetLPC3VaVoltage .....	63
3.	SMBus DLL Function .....	65
3.1	Support Fintek I2C IC .....	65
3.2	SMBus Function List .....	65
3.3	SMBus Function Flow .....	66
4.	Use Library .....	67
4.1	Load Library .....	67
4.2	Unload Library .....	67
4.3	DLL Parameter .....	68
5.	Known issues .....	68
5.1	Microsoft Security Advisory 3033929 .....	68

# 1. SDK Driver Install

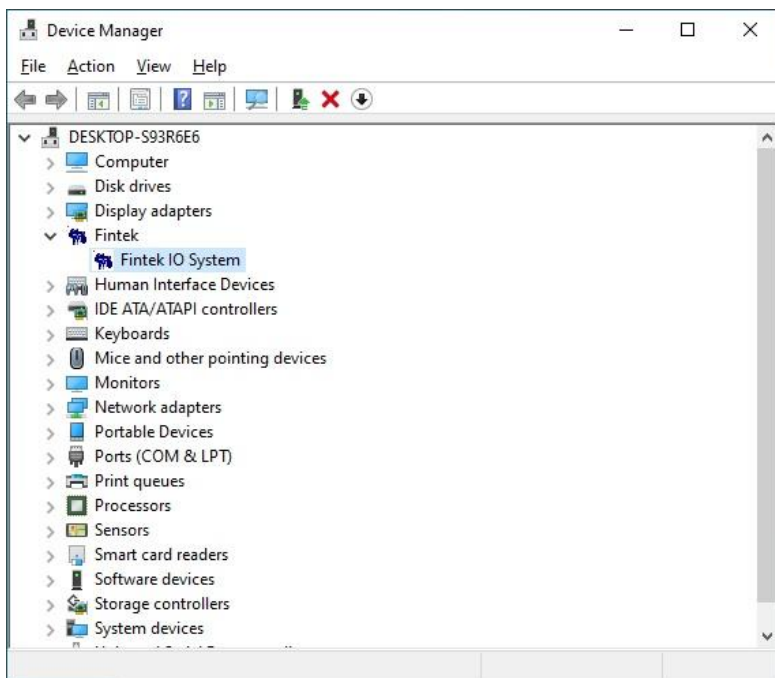
- See below for the driver installation steps:

## 1.1 Windows 7/8/8.1/10/11 Series Install

Step1: Open the [Install] folder, press “Install\_x86/x64” to install driver.

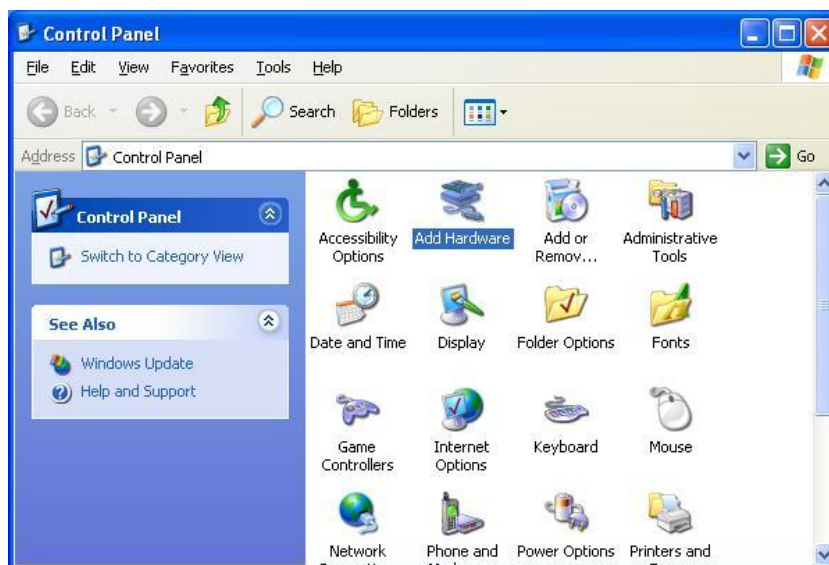


Step2: When the installation is complete, the device manager would show up the [Fintek IO System] as below figure.



## 1.2 Windows XP Install

Step1: Go to the Windows Start menu, select the [Control Panel] option, and then double-click the [Add Hardware] icon.



Step2: The Add Hardware Wizard appears. Click the [Next] button.



Step3: Select the option [Yes, I have already connected the hardware] and then click the [Next] button.



Step4: Look for the [Installed hardware] section, select the [Add a new hardware device] and then click the [Next] button.



Step5: Select [install the hardware that I manually select from a list (Advanced)] and then click the [Next] button.



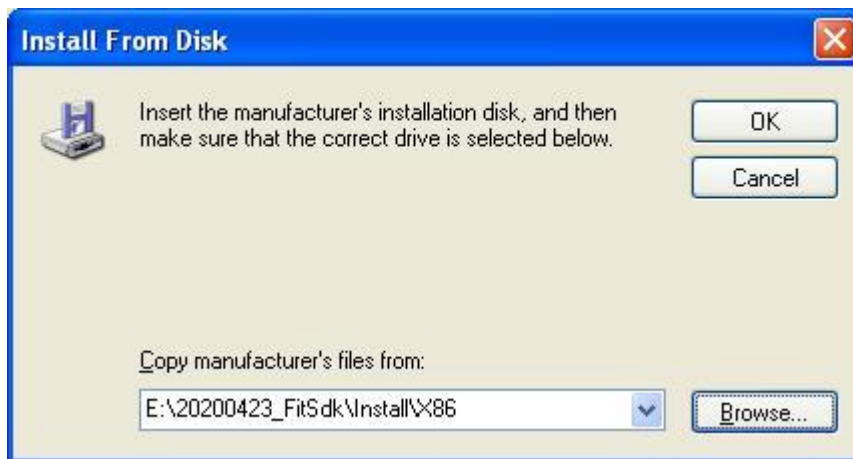
Step6: Look for the [Common hardware types] section, select the [Show All Devices] and then click the [Next] button.



Step7: Select the [Have Disk]. Click the Next button.



Step8: in the window [Install from Disk], use the button "Browse" to locate on [FitSdk\install\X86\] folder, so that the text-box [Copy manufacturer's files from] shows [FitSdk\install\X86\], then select OK.



Step9: In the window "Add Hardware Wizard", you can now select the [Fintek IO System] and then click the [Next] button.



Step10: Click the [Next] button to install [Fintek IO System] driver.

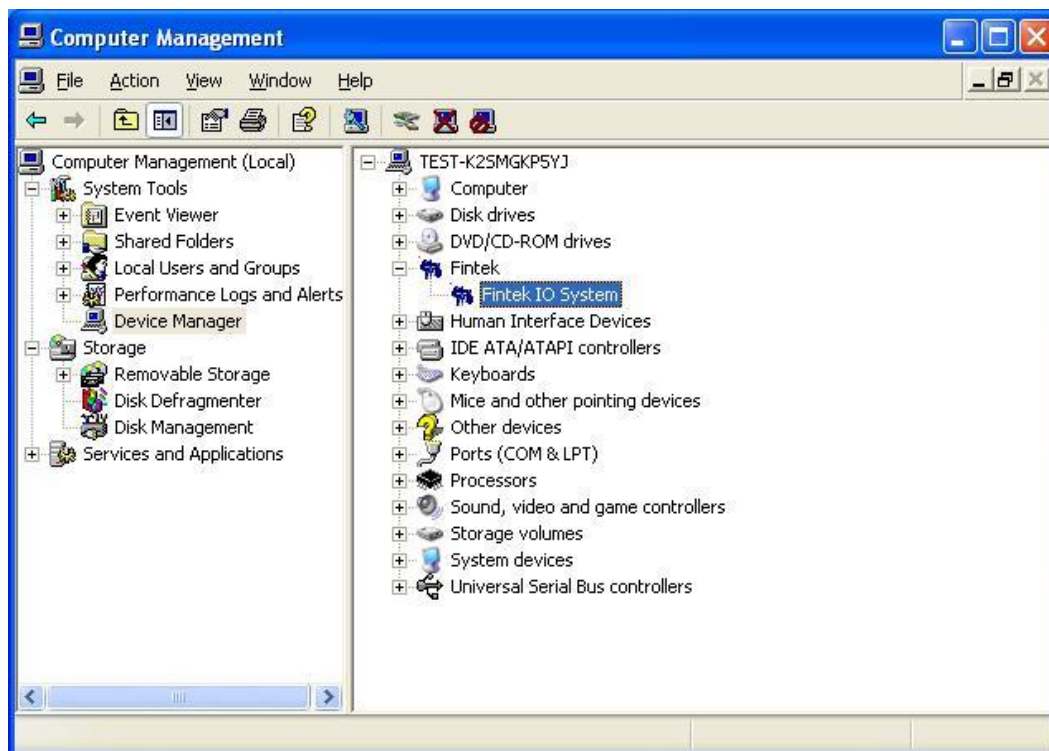


Step11: Click Finish.



Step12: After installing [Fintek IO System] driver, you should confirm the installation using the Device Manager. Device Manager can also be launched from the Computer Management console. It is a snap-in located under System Tools.

This allows you to view the installed [Fintek IO System], as shown in figure below.



## 2. LPC DLL Function

### 2.1 Support Fintek LPC IC

F81801	F81865	F71889E	F71869A	F81867	F81866A
F81804	F81966	F81216	F81768	F81803	F75113
F71808A	F81214E	F81218E	F81968	F81962	F81964
F81967	F81216E				

NOTE: GPIO function only support GPIO\_LPC\_R/ GPIO\_LPC\_W,

F75113/F81866/F81768/F81803/F81804/F81966/F81968/F81962/F81964/F81967 have additional functions, see section 2.3 GPIO function List.

## 2.2 LPC Function

### Function List

ID	Function Name	Description	Support Fintek IC
2.2.1	FintekLPCICSelect	Use while more than 2 Fintek LPC IC on the motherboard	See section 2.1
2.2.2	PORT_LPC_R	LPC port read .	See section 2.1
2.2.3	PORT_LPC_W	LPC port write	See section 2.1

### 2.2.1 FintekLPCICSelect

**int FintekLPCICSelect (int number)**

**Function** : Select the Fintek LPC IC.

**Description** :

You can use this function to change the controlled IC if you have more than 2 Fintek LPC IC on the motherboard.

Default search address as following order:

0x2E 0x67 => 0x2E 0x77 => 0x2E 0x87 => 0x2E 0xA0 => 0x2E 0x50 => 0x4E 0x67 => 0x4E 0x77 => 0x4E 0x87 => 0x4E 0xA0 => 0x4E 0x50

number 1 is first found IC, number 2 is second, etc. Default is the first found IC.

Example: If you have two IC, which address are 0x4E 0x77 and 0x2E 0x87, the 0x2E 0x87 is number 1 and the other is number 2.

**Return Value** :

If the function succeeds, the return value is TRUE. If the function fails, the return value is zero (FALSE). If the Fintek IC selected does not support HWMonitor function, the return value is 2. If the Fintek IC selected does not support GPIO base address mode

function, the return value is 4

## 2.2.2 PORT\_LPC\_R

**int PORT\_LPC\_R(int index, unsigned long \*value, unsigned char bytelen)**

**Function:** Read Port value.

**Parameters:**

Index: IO Port.

Value: Return value.

Bytelen: 1: 8 bits value read. 2: 16 bits value read. 4: 32 bits value read. Others: 8 bit value read.

**Return Value:**

If the function succeeds, the return value is TRUE. If the function fails, the return value is zero (FALSE). Without any FINTEK LPC chip exist, return 3.

## 2.2.3 PORT\_LPC\_W

**int PORT\_LPC\_W(int index, unsigned long value, unsigned char bytelen)**

**Function:** Write Port value.

**Parameters:**

Index: IO Port.

Value: Write value.

Bytelen: 1: 8 bits value write. 2: 16 bits value write. 4: 32 bits value write. Others: 8 bit value write.

**Return Value:**

If the function succeeds, the return value is TRUE. If the function fails, the return value is zero (FALSE). Without any FINTEK LPC chip exist, return 3.

## 2.3 GPIO Function

### Function List

ID	Function Name	Description	Support Fintek IC
2.3.1	GPIO_LPC_R	Read GPIOxx value	See section 2.1
2.3.2	GPIO_LPC_W	Write GPIOxx value.	See section 2.1
2.3.3	SetLPCGpioControl	Set all pins input output control register.	F75113/F81866 F81768/F81803 F81966/F81962 F81964/F81967 F81804/F81968 F81214E/ F81216E F81218E
2.3.4	GetLPCGpioControl	Get all pins input output control register.	
2.3.5	SetLPCGpioOutputDataIndividual	Set one pin output data value.	
2.3.6	GetLPCGpioOutputDataIndividual	Get one pin output data value.	
2.3.7	GetLPCGpioStatusIndividual	Get one pin data.	
2.3.8	GetLPCGpioStatus	Get all pins data.	
2.3.9	SetLPCGpioEdgeDetector	Set all pins edge detector register.	
2.3.10	GetLPCGpioEdgeDetector	Get all pins edge detector register.	
2.3.11	SetLPCGpioClearEdgeDetectorStatus	Clear edge detector status.	F75113 only
2.3.12	GetLPCGpioEdgeDetectorStatus	Get all pins edge detector status register.	F75113 only
2.3.13	SetLPCGpioDebounceEnable	Set all pins debounce enable register.	F75113 only
2.3.14	GetLPCGpioDebounceEnable	Get all pins debounce enable register.	F75113 only
2.3.15	SetLPCGpioLevelPulseControl	Set all pins level/pulse control register.	F75113 only
2.3.16	GetLPCGpioLevelPulseControl	Get all pins level/pulse control register.	F75113 only
2.3.17	SetLPCGpioDebounceTime	Set all pins debounce time.	F75113 only
2.3.18	GetLPCGpioDebounceTime	Get all pins debounce time.	F75113 only
2.3.19	SetLPCGpioPulseWidth	Set all pins pulse width.	F75113 only
2.3.20	GetLPCGpioPulseWidth	Get all pins pulse width.	F75113 only
2.3.21	SetLPCGpioInverseEnable	Set all pins Inverse enable register.	F75113 only
2.3.22	GetLPCGpioInverseEnable	Get all pins Inverse enable register.	F75113 only
2.3.23	SetLPCGpioSMIEnable	Set all pins SMI enable register.	F75113/F81866 F81768/F81803 F81966/F81962 F81964/F81967 F81804/F81968
2.3.24	GetLPCGpioSMIEnable	Get all pins SMI enable register.	
2.3.25	SetLPCGpioOutputDrivingEnable	Set all pins output driving enable register.	
2.3.26	GetLPCGpioOutputDrivingEnable	Get all pins output driving enable register.	

## 2.3.1 GPIO\_LPC\_R

int GPIO\_LPC\_R(int index, int \*value, unsigned char mode)

**Function:** Get GPIO pin value.

**Parameters:**

index: GPIO Output Register.

value: Return GPIO group status.

mode:

Value	Description
-------	-------------

0x01	LDN mode (default)
0x02	Use GPIO index/data port. Write index to index port first and then read/write the register.
0x04	Use digital I/O port. The way only access GPIO data register. Write data to this port will control the data output register. And read this port will read the pin status register.

mode index: **The actual index value may vary with different FINTEK IC, please contact the FINTEK members.**

Value	Index value
0x01	0x5x(0x50~): GPIOC register. 0x4x(0x48~): GPIOB register. 0x4x(0x40~): GPIOA register.
0x02	0x9x(0x98~): GPIO9 register. 0x8x(0x88~): GPIO8 register. 0x8x(0x80~): GPIO7 register. 0x9x: GPIO6 register. 0xAx: GPIO5 register. 0xBx: GPIO4 register. 0xCx: GPIO3 register. 0xDx: GPIO2 register. 0Ex: GPIO1 register. 0Fx: GPIO0 register.
0x04	2: GPIO8 register. 3: GPIO7 register. 4: GPIO6 register. 5: GPIO5 register. 6: GPIO0 register. 7: GPIO1 register. 8: GPIO2 register. 9: GPIO3 register. 10: GPIO4 register.

**Return Value:**

If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

**Example :** Get GPIOOX mode control Register

```
void XXXX(void)
{
    int value = 0;
```

```
GETINT2UCHARPROC ProcAdd;
```

```
ProcAdd = (GETINT2UCHARPROC) GetProcAddress(hinstLib, " GPIO_LPC_R");
```

```
if (NULL != ProcAdd)
```

```
{
```

```
    if (!(*ProcAdd)( 0xF0, & value, 1))
```

```
        // Fail to get this value
```

```
}
```

```
else
```

```
{
```

```
    // Fail to get this procedure address
```

```
}
```

```
}
```

## 2.3.2 GPIO\_LPC\_W

**int GPIO\_LPC\_W(int index, int value, unsigned char mode)**

**Function:** Set GPIO pin status.

**Parameters:**

index: GPIO Output Register.

value: Write GPIO group status.

mode:

Value	Description
0x01	LDN mode (default)
0x02	Use GPIO index/data port. Write index to index port first and then read/write the register.
0x04	Use digital I/O port. The way only access GPIO data register. Write data to this port will control the data output register. And read this port will read the pin status register.

mode index: **The actual index value may vary with different FINTEK IC, please contact the FINTEK members.**

Value	Index value
0x01	0x5x(0x50~): GPIOC register. 0x4x(0x48~): GPIOB register. 0x4x(0x40~): GPIOA register. 0x9x(0x98~): GPIO9 register.

0x02	0x8x(0x88~): GPIO8 register. 0x8x(0x80~): GPIO7 register. 0x9x: GPIO6 register. 0xAx: GPIO5 register. 0xBx: GPIO4 register. 0xCx: GPIO3 register. 0xDx: GPIO2 register. 0Ex: GPIO1 register. 0xFx: GPIO0 register.
0x04	2: GPIO8 register. 3: GPIO7 register. 4: GPIO6 register. 5: GPIO5 register. 6: GPIO0 register. 7: GPIO1 register. 8: GPIO2 register. 9: GPIO3 register. 10: GPIO4 register.

**Return Value:**

If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

**Example :** Set GPIO0X mode control Register

```

void XXXX(void)
{
    int value = 0x03;
    SETINT2UCHARPROC ProcAdd;

    ProcAdd = (SETINT2UCHARPROC) GetProcAddress(hinstLib, "GPIO_LPC_W");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 0xF0, value, 1))
            // Fail to get this value
    }
    else
    {

```

```

        // Fail to get this procedure address
    }
}
    
```

## 2.3.3 SetLPCGpioControl

The function will help you to write GPIO all pins input/output control value.

**bool SetGpioControl(unsigned char ucGpioX, unsigned char value)**

**Parameters:**

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)

value: EX. 0x21: set GPIO 10 and GPIO 15 to output mode(if ucGpioX is 0x10)

**Return Value:**

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

**Example :** Set GPIO1X mode control

```

void XXXX(void)
{
    int value = 0x21;
    SETUCHAR2PROC ProcAdd;

    ProcAdd = (SETUCHAR2PROC) GetProcAddress(hinstLib, "SetLPCGpioControl");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)(0x10, value)) // set GPIO 10 and GPIO 15 to output mode
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
    
```

## 2.3.4 GetLPCGpioControl

The function will help you to read GPIO all pins input/output control value.

**bool GetGpioControl(unsigned char ucGpioX, unsigned char \*value)**

**Parameters:**

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)

value: Pointer to a variable that stores the valid value

**Return Value:**

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

**Example :** Get GPIO4X mode control

```
void XXXX(void)
{
    int value = 0;
    GETUCHAR2PROC ProcAdd;

    ProcAdd = (GETUCHAR2PROC) GetProcAddress(hinstLib, "GetLPCGpioControl");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 0x40, &value))
            // Fail to get this value
        }
    else
    {
        // Fail to get this procedure address
    }
}
```

## 2.3.5 SetLPCGpioOutputDataIndividual

The function will help you to set output data by GPIO pin index

**bool SetLPCGpioOutputDataIndividual (unsigned char ucGpioX, unsigned char value)**

**Parameters:**

ucGpioX: EX. 0x32 is GPIO 32 (0x45 is GPIO 45, etc)

value: Pin value, 0 or 1

**Return Value:**

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

**Example :** Set GPIO32 data to 1

```
void XXXX(void)
{
    int value = 1;
    SETUCHAR2PROC ProcAdd;

    ProcAdd = (SETUCHAR2PROC) GetProcAddress(hinstLib, "SetLPCGpioOutputDataIndividual");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 0x32, value)) // Set GPIO32 data to 1
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
```

## 2.3.6 GetLPCGpioOutputDataIndividual

The function will help you to get data by GPIO pin index

**bool GetLPCGpioOutputDataIndividual (unsigned char ucGpioX, unsigned char \*value)**

### Parameters:

ucGpioX: EX. 0x32 is GPIO 32 (0x45 is GPIO 45, etc)

value: Pin value, 0 or 1

### Return Value:

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

**Example :** Get GPIO45 data

```
void XXXX(void)
{
    int value = 0;
    GETUCHAR2PROC ProcAdd;

    ProcAdd = (GETUCHAR2PROC) GetProcAddress(hinstLib, "GetLPCGpioOutputDataIndividual");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 0x45, &value)) // value = 0 or 1
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
```

## 2.3.7 GetLPCGpioStatusIndividual

The function will help you to get status by GPIO pin index

**bool GetLPCGpioStatusIndividual (unsigned char ucGpioX, unsigned char \*value)**

**Parameters:**

ucGpioX: EX. 0x32 is GPIO 32 (0x45 is GPIO 45, etc)

value: Pin status value, 0 or 1

**Return Value:**

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

**Example :** Get GPIO45 status

```
void XXXX(void)
{
    int value = 0;
    GETUCHAR2PROC ProcAdd;

    ProcAdd = (GETUCHAR2PROC) GetProcAddress(hinstLib, " GetLPCGpioStatusIndividual");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 0x45, &value)) // value = 0 or 1
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
```

## 2.3.8 GetLPCGpioStatus

The function will help you to read GPIO all pins status value.

**bool GetLPCGpioStatus (unsigned char ucGpioX, unsigned char \*value)**

**Parameters:**

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)

value: Pointer to a variable that stores the valid value

**Return Value:**

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

**Example :** Get GPIO4X status

```
void XXXX(void)
{
    int value = 0;
    GETUCHAR2PROC ProcAdd;

    ProcAdd = (GETUCHAR2PROC) GetProcAddress(hinstLib, " GetLPCGpioStatus");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 0x40, &value))
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
```

## 2.3.9 SetLPCGpioEdgeDetector

The function will help you to enable GPIO all pins edge detector.

If set to 1 and GPIO set to input mode will enable GPIO edge detection.

**bool SetLPCGpioEdgeDetector (unsigned char ucGpioX, unsigned char value)**

**Parameters:**

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)

value: EX. 0x21: enable GPIO 10 and GPIO 15 edge detector (if ucGpioX is 0x10)

**Return Value:**

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

**Example :** Set GPIO1X edge detector enable value

```
void XXXX(void)
{
    int value = 0x21;
    SETUCHAR2PROC ProcAdd;

    ProcAdd = (SETUCHAR2PROC) GetProcAddress(hinstLib, "SetLPCGpioEdgeDetector");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 0x10, value)) // enable GPIO 10 and GPIO 15 edge detector
            // Fail to get this value
        }
    else
    {
        // Fail to get this procedure address
    }
}
```

## 2.3.10 GetLPCGpioEdgeDetector

The function will help you to read GPIO all pins edge detector enable value.

**bool GetLPCGpioEdgeDetector (unsigned char ucGpioX, unsigned char \*value)**

### Parameters:

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)  
value: Pointer to a variable that stores the valid value

### Return Value:

If the function succeeds, the return value is nonzero (TRUE).  
If the function fails, the return value is zero (FALSE).

**Example :** Get GPIO4X edge detector enable value

```
void XXXX(void)
{
    int value = 0;
    GETUCHAR2PROC ProcAdd;

    ProcAdd = (GETUCHAR2PROC) GetProcAddress(hinstLib, "GetLPCGpioEdgeDetector");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 0x40, &value))
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
```

## 2.3.11 SetLPCGpioClearEdgeDetectorStatus

The function will help you to clear GPIO all pins edge detector status.

Write 1 to clear edge detector status bit. Writing 0 is invalid.

**bool SetLPCGpioClearEdgeDetectorStatus (unsigned char ucGpioX, unsigned char value)**

**Parameters:**

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)

value: EX. 0x21: clear GPIO 10 and GPIO 15 edge detector status (if ucGpioX is 0x10)

**Return Value:**

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

**Example :** Clear GPIO1X edge detector status

```
void XXXX(void)
{
    int value = 0x21;
    SETUCHAR2PROC ProcAdd;

    ProcAdd = (SETUCHAR2PROC) GetProcAddress(hinstLib, "SetLPCGpioClearEdgeDetectorStatus");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 0x10, value)) // Clear GPIO 10 and GPIO 15 edge detector status
            // Fail to get this value
        }
        else
        {
            // Fail to get this procedure address
        }
    }
}
```

## 2.3.12 GetLPCGpioEdgeDetectorStatus

The function will help you to read GPIO all pins edge detector status.

Indicate GPIO Edge Status. If set to 1, the edge of GPIO has occurred.

**bool GetLPCGpioEdgeDetectorStatus (unsigned char ucGpioX, unsigned char \*value)**

**Parameters:**

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)  
 value: Pointer to a variable that stores the valid value

**Return Value:**

If the function succeeds, the return value is nonzero (TRUE).  
 If the function fails, the return value is zero (FALSE).

**Example :** Get GPIO4X edge detector status

```
void XXXX(void)
{
    int value = 0;
    GETUCHAR2PROC ProcAdd;

    ProcAdd = (GETUCHAR2PROC) GetProcAddress(hinstLib, "GetLPCGpioEdgeDetectorStatus");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 0x40, &value))
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
```

## 2.3.13 SetLPCGpioDebounceEnable

The function will help you to enable GPIO all pins input debounce.

**bool SetLPCGpioDebounceEnable (unsigned char ucGpioX, unsigned char value)**

**Parameters:**

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)

value: EX. 0x21: enable GPIO 10 and GPIO 15 input debounce (if ucGpioX is 0x10)

**Return Value:**

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

**Example :** Set GPIO1X debounce enable

```
void XXXX(void)
{
    int value = 0x21;
    SETUCHAR2PROC ProcAdd;

    ProcAdd = (SETUCHAR2PROC) GetProcAddress(hinstLib, "SetLPCGpioDebounceEnable");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)(0x10, value)) // set GPIO 10 and GPIO 15 to debounce enable
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
```

## 2.3.14 GetLPCGpioDebounceEnable

The function will help you to read GPIO all pins debounce enable value.

**bool GetLPCGpioDebounceEnable (unsigned char ucGpioX, unsigned char \*value)**

**Parameters:**

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)

value: Pointer to a variable that stores the valid value

**Return Value:**

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

**Example :** Get GPIO4X debounce enable value

```
void XXXX(void)
{
    int value = 0;
    GETUCHAR2PROC ProcAdd;

    ProcAdd = (GETUCHAR2PROC) GetProcAddress(hinstLib, "GetLPCGpioDebounceEnable");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 0x40, &value))
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
```

## 2.3.15 SetLPCGpioLevelPulseControl

The function will help you to write GPIO all pins Level/Pulse control value.

If set to 1, output mode is pulse, else if set to 0, output mode is level.

**bool SetLPCGpioLevelPulseControl (unsigned char ucGpioX, unsigned char value)****Parameters:**

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)

value: EX. 0x21: set GPIO 10 and GPIO 15 to pulse mode (if ucGpioX is 0x10)

**Return Value:**

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

**Example :** Set GPIO1X Level/Pulse control

```
void XXXX(void)
{
    int value = 0x21;
    SETUCHAR2PROC ProcAdd;

    ProcAdd = (SETUCHAR2PROC) GetProcAddress(hinstLib, "SetLPCGpioLevelPulseControl");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 0x10, value)) // set GPIO 10 and GPIO 15 to pulse mode
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
```

## 2.3.16 GetLPCGpioLevelPulseControl

The function will help you to read GPIO all pins Level/Pulse control value.

**bool GetLPCGpioLevelPulseControl (unsigned char ucGpioX, unsigned char \*value)**

**Parameters:**

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)

value: Pointer to a variable that stores the valid value

**Return Value:**

If the function succeeds, the return value is nonzero (TRUE).



If the function fails, the return value is zero (FALSE).

**Example :** Get GPIO4X Level/Pulse control

```
void XXXX(void)
{
    int value = 0;
    GETUCHAR2PROC ProcAdd;

    ProcAdd = (GETUCHAR2PROC) GetProcAddress(hinstLib, "GetLPCGpioLevelPulseControl");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 0x40, &value))
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
```

## 2.3.17 SetLPCGpioDebounceTime

The function will help you to write GPIO all pins input de-bounce time value.

If set to 1 debounce time is 25ms else if set to 0 debounce time is 10μs.

**bool SetLPCGpioDebounceTime (unsigned char ucGpioX, unsigned char value)**

**Parameters:**

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)

value: EX. 0x21: set GPIO 10 and GPIO 15 debounce time to 25ms (if ucGpioX is 0x10)

**Return Value:**

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

**Example :** Set GPIO1X debounce time

```
void XXXX(void)
{
    int value = 0x21;
    SETUCHAR2PROC ProcAdd;

    ProcAdd = (SETUCHAR2PROC) GetProcAddress(hinstLib, "SetLPCGpioDebounceTime");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)(0x10, value)) // set GPIO 10 and GPIO 15 debounce time to 25ms
            // Fail to get this value
        }
    }
    else
    {
        // Fail to get this procedure address
    }
}
```

## 2.3.18 GetLPCGpioDebounceTime

The function will help you to read GPIO all pins input de-bounce time value.

If set to 1 debounce time is 25ms else if set to 0 debounce time is 10 $\mu$ s.

**bool GetLPCGpioDebounceTime (unsigned char ucGpioX, unsigned char \*value)**

### Parameters:

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)

value: Pointer to a variable that stores the valid value

### Return Value:

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

**Example :** Get GPIO4X debounce time

```
void XXXX(void)
```

```

{
    int value = 0;
    GETUCHAR2PROC ProcAdd;

    ProcAdd = (GETUCHAR2PROC) GetProcAddress(hinstLib, "GetLPCGpioDebounceTime");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 0x40, &value))
            // Fail to get this value
        }
    else
    {
        // Fail to get this procedure address
    }
}

```

## 2.3.19 SetLPCGpioPulseWidth

The function will help you to write GPIO all pins pulse width value.

If set the GPIO to pulse mode, the pulse width can be defined as follows:

00b: 500us / 01b: 1ms / 10b: 20ms / 11b: 100ms

**bool SetLPCGpioPulseWidth (unsigned char ucGpioX, unsigned char value)**

### Parameters:

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)

value: 00b: 500us/01b: 1ms/10b: 20ms/11b: 100ms

### Return Value:

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

**Example :** Set GPIO1X pulse width to 100ms

```

void XXXX(void)
{

```

```

int value = 0x03;
SETUCHAR2PROC ProcAdd;

ProcAdd = (SETUCHAR2PROC) GetProcAddress(hinstLib, "SetLPCGpioPulseWidth");
if (NULL != ProcAdd)
{
    if (!(*ProcAdd)( 0x10, value)) // set GPIO 1x pulse width to 100ms
        // Fail to get this value
    }
else
{
    // Fail to get this procedure address
}
}
    
```

## 2.3.20 GetLPCGpioPulseWidth

The function will help you to read GPIO all pins pulse width value.

If set the GPIO to pulse mode, the pulse width can be defined as follows:

00b: 500us / 01b: 1ms / 10b: 20ms / 11b: 100ms

**bool GetLPCGpioPulseWidth (unsigned char ucGpioX, unsigned char \*value)**

### Parameters:

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)

value: Pointer to a variable that stores the valid value

### Return Value:

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

**Example :** Get GPIO4X pulse width

```

void XXXX(void)
{
    
```

```

int value = 0;
GETUCHAR2PROC ProcAdd;

ProcAdd = (GETUCHAR2PROC) GetProcAddress(hinstLib, "GetLPCGpioPulseWidth");
if (NULL != ProcAdd)
{
    if (!(*ProcAdd)( 0x40, &value))
        // Fail to get this value
    }
else
{
    // Fail to get this procedure address
}
}

```

## 2.3.21 SetLPCGpioInverseEnable

The function will help you to write GPIO output signal inverse value.  
If the pin inverse was selected, the output signal would be inverted.

**bool SetLPCGpioInverseEnable (unsigned char ucGpioX, unsigned char value)**

**Parameters:**

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)  
value: EX. 0x21: set GPIO 10 and GPIO 15 to output signal inverse (if ucGpioX is 0x10)

**Return Value:**

If the function succeeds, the return value is nonzero (TRUE).  
If the function fails, the return value is zero (FALSE).

**Example :** Set GPIO1X output signal inverse

```

void XXXX(void)
{
    int value = 0x21;
}

```

```
SETUCHAR2PROC ProcAdd;
```

```
ProcAdd = (SETUCHAR2PROC) GetProcAddress(hinstLib, "SetLPCGpioInverseEnable");
if (NULL != ProcAdd)
{
    if (!(*ProcAdd)(0x10, value)) // set GPIO 10 and GPIO 15 to output signal inverse
        // Fail to get this value
    }
else
{
    // Fail to get this procedure address
}
}
```

## 2.3.22 GetLPCGpioInverseEnable

The function will help you to read GPIO output signal inverse value.

**bool GetLPCGpioInverseEnable (unsigned char ucGpioX, unsigned char \*value)**

### Parameters:

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)  
 value: Pointer to a variable that stores the valid value

### Return Value:

If the function succeeds, the return value is nonzero (TRUE).  
 If the function fails, the return value is zero (FALSE).

**Example :** Get GPIO4X output signal inverse value

```
void XXXX(void)
{
    int value = 0;
    GETUCHAR2PROC ProcAdd;
```

```

ProcAdd = (GETUCHAR2PROC) GetProcAddress(hinstLib, "GetLPCGpioInverseEnable");
if (NULL != ProcAdd)
{
    if (!(*ProcAdd)( 0x40, &value))
        // Fail to get this value
    }
else
{
    // Fail to get this procedure address
}
}

```

## 2.3.23 SetLPCGpioSMIEnable

The function will help you to enable GPIO SMI Generation.

If this bit set to 1, enable GPIO to generate SMI.

**bool SetLPCGpioSMIEnable (unsigned char ucGpioX, unsigned char value)**

### Parameters:

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)

value: EX. 0x21: enable GPIO 10 and GPIO 15 SMI Generation (if ucGpioX is 0x10)

### Return Value:

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

**Example :** Enable GPIO1X SMI Generation

```

void XXXX(void)
{
    int value = 0x21;
    SETUCHAR2PROC ProcAdd;

    ProcAdd = (SETUCHAR2PROC) GetProcAddress(hinstLib, "SetLPCGpioSMIEnable");

```

```

if (NULL != ProcAdd)
{
    if (!(*ProcAdd)( 0x10, value)) // enable GPIO 10 and GPIO 15 SMI Generation
        // Fail to get this value
    }
else
{
    // Fail to get this procedure address
}
}

```

## 2.3.24 GetLPCGpioSMIEnable

The function will help you to read GPIO SMI Generation enable value.

**bool GetLPCGpioSMIEnable (unsigned char ucGpioX, unsigned char \*value)**

**Parameters:**

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)  
value: Pointer to a variable that stores the valid value

**Return Value:**

If the function succeeds, the return value is nonzero (TRUE).  
If the function fails, the return value is zero (FALSE).

**Example :** Get GPIO4X SMI Generation enable value

```

void XXXX(void)
{
    int value = 0;
    GETUCHAR2PROC ProcAdd;

    ProcAdd = (GETUCHAR2PROC) GetProcAddress(hinstLib, "GetLPCGpioSMIEnable");
    if (NULL != ProcAdd)
    {

```

```

        if (!(*ProcAdd)( 0x40, &value))
            // Fail to get this value
        }
    else
    {
        // Fail to get this procedure address
    }
}
    
```

## 2.3.25 SetLPCGpioOutputDrivingEnable

The function will help you to write GPIO drive enable value.

If set to 0, the pin will be open drain in output mode, if set to 1, the pin will be push pull in output mode.

**bool SetLPCGpioOutputDrivingEnable (unsigned char ucGpioX, unsigned char value)**

### Parameters:

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)

value: EX. 0x21: set GPIO 10 and GPIO 15 output mode to push pull (if ucGpioX is 0x10)

### Return Value:

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

**Example :** Set GPIO1X drive enable

```

void XXXX(void)
{
    int value = 0x21;
    SETUCHAR2PROC ProcAdd;

    ProcAdd = (SETUCHAR2PROC) GetProcAddress(hinstLib, "SetLPCGpioOutputDrivingEnable");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 0x10, value)) // set GPIO 10 and GPIO 15 output mode to push pull
            // Fail to get this value
    }
}
    
```

```

    }
    else
    {
        // Fail to get this procedure address
    }
}
    
```

## 2.3.26 GetLPCGpioOutputDrivingEnable

The function will help you to read GPIO drive enable value.

If value is 0, the pin will be open drain in output mode, if value is 1, the pin will be push pull in output mode.

**bool GetLPCGpioOutputDrivingEnable (unsigned char ucGpioX, unsigned char \*value)**

### Parameters:

ucGpioX: EX. 0x10 is GPIO 1x (0x40 is GPIO 4x, etc)

value: Pointer to a variable that stores the valid value

### Return Value:

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

**Example :** Get GPIO4X drive enable value

```

void XXXX(void)
{
    int value = 0;
    GETUCHAR2PROC ProcAdd;

    ProcAdd = (GETUCHAR2PROC) GetProcAddress(hinstLib, "GetLPCGpioOutputDrivingEnable");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 0x40, &value))
            // Fail to get this value
    }
}
    
```

```

else
{
    // Fail to get this procedure address
}
}
    
```

## 2.4 WDT Function:

### Function List

ID	Function Name	Description
2.4.1	WDT_LPC_SetConfig	Set WatchDog configuration
2.4.2	WDT_LPC_SetTimeRange	Set WatchDog timing range
2.4.3	WDT_LPC_GetConfig	Get WatchDog configuration
2.4.4	WDT_LPC_GetTimeRange	Get WatchDog time range

### 2.4.1 WDT\_LPC\_SetConfig

**int WDT\_LPC\_SetConfig(int Group, unsigned int value)**

**Function:** Set WatchDog configuration.

**Parameters:**

Group: WDT group.

- Group 1: Reserved
- Group 2: For F81801/F81865/F71889E/F71869A/F81867/F81866A/F81804/F81966/F81768/F81803/F71808A/F81968  
F81962/F81964/F81967
- Group 3: For F81216 series
- Group 4: For F75113

Value: configuration value, consulted the IC SPEC.

- Group 2: For F81801/F81865/F71889E/F71869A/F81867/F81866A/F81804/F81966/F81768/F81803/F71808A/F81968  
F81962/F81964/F81967

Define	Value	Description
LPC_WDT_WDOUT_EN_FLAG1	0x8000	Enable Watchdog time out output via WDTRST#. For F81801/F71889E/F71869A/F81768/F81803/F71808A

LPC_WDT_WDOUT_EN_FLAG2	0x0100	Enable Watchdog time out output via WDTRST#. For F81865/F81867/F81866A/F81804/F81966/F81968 F81962/F81964/F81967
LPC_WDT_TIMEOUT_FLAG	0x40	When watchdog timeout. This bit will be set to 1.
LPC_WDT_ENABLE_FLAG	0x20	Enable watchdog timer
LPC_WDT_PULSE_FLAG	0x10	Configure WDT output mode 0: Level Mode 1: Pulse Mode
LPC_WDT_UNIT_FLAG	0x08	Watchdog unit select. 0: Select second. 1: Select minute.
LPC_WDT_PSWIDTH_1MS	0x00	When select Pulse mode: 1 ms.
LPC_WDT_PSWIDTH_25MS	0x01	When select Pulse mode: 25 ms.
LPC_WDT_PSWIDTH_125MS	0x02	When select Pulse mode: 125 ms.
LPC_WDT_PSWIDTH_5000MS	0x03	When select Pulse mode: 5000 ms.

■ Group 3: For F81216 series

Define	Value	Description
LPC_WDT3_UNIT_10MS	0x00	Timer Unit is 10ms..
LPC_WDT3_UNIT_SEC	0x02	Timer Unit is 1 second.
LPC_WDT3_UNIT_MIN	0x04	Timer Unit is 1 minute.
LPC_WDT_TIMEOUT_OCCUR	0x01	0 : no time out occur. 1 : time out has occurred. Write "1" to this bit will clear the status.

■ Group 4: For F75113

Define	Value	Description
LPC_WDT_TIMEOUT_OCCUR	0x01	0 : no time out occur. 1 : time out has occurred. Write "1" to this bit will clear the status.

**Return Value:**

If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

**Example:** Enable Group2 WDT and parameter setting.

```
void XXXX(void)
{
    unsigned int value = 0x22; // 0x20: enable WDT; 0x02: pulse width;
```

```

SETINTUINTPROC ProcAdd;

ProcAdd = (SETINTUINTPROC) GetProcAddress(hinstLib, " WDT_LPC_SetConfig ");
if (NULL != ProcAdd)
{
    if (! (*ProcAdd)( 2, value))
        // Fail to get this value
    }
else
{
    // Fail to get this procedure address
}
}
    
```

## 2.4.2 WDT\_LPC\_SetTimeRange

**int WDT\_LPC\_SetTimeRange (int Group, int value)**

**Function:** Set WDT timing range.

**Parameters:**

Group: WDT group.

- Group 1: Reserved
- Group 2: For F81801/F81865/F71889E/F71869A/F81867/F81866A/F81804/F81966/F81768/F81803/F71808A/F81968/F81962/F81964/F81967

Setting timing range from 0 - 255. The unit is either second or minute programmed by WDT\_LPC\_SetConfig.

- Group 3: For F81216 series  
Setting timing range from 0 – 255 and **start WDT**. The unit is 10ms, second or minute programmed by WDT\_LPC\_SetConfig.

- Group 4: For F75113

Value: Timing range, consulted the IC SPEC.

**Return Value:**

If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

**Example:** Set WDT timing range to 10 sec.

```

void XXXX(void)
{
    
```

```

int value = 0x0A;
SETINT2PROC ProcAdd;

ProcAdd = (SETINT2PROC) GetProcAddress(hinstLib, " WDT_LPC_ SetTimeRange");
if (NULL != ProcAdd)
{
    if (!(*ProcAdd)( 2, value))
        // Fail to get this value
    }
else
{
    // Fail to get this procedure address
}
}
    
```

### 2.4.3 WDT\_LPC\_GetConfig

**int WDT\_LPC\_GetConfig (int Group, unsigned int \*value)**

**Function:** Get WatchDog configuration.

**Parameters:**

Group: WDT group.

- Group 1: Reserved
- Group 2: For F81801/F81865/F71889E/F71869A/F81867/F81866A/F81804/F81966/F81768/F81803/F71808A/F81968  
F81962/F81964/F81967
- Group 3: For F81216 series
- Group 4: For F75113

Value: configuration value, consulted the IC SPEC..

**Return Value:**

If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

### 2.4.4 WDT\_LPC\_GetTimeRange

**int WDT\_LPC\_GetTimeRange (int Group, int \*value)**

**Function:** Get WDT timing range.

**Parameters:**

Group: WDT group.

- Group 1: Reserved
- Group 2: For F81801/F81865/F71889E/F71869A/F81867/F81866A/F81804/F81966/F81768/F81803/F71808A/F81968  
F81962/F81964/F81967
- Group 3: For F81216 series
- Group 4: For F75113

Value: Timing range, consulted the IC SPEC.

**Return Value:**

If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

## 2.4.5 WDT Full Example

**Group 2 Example:** F81866 Start watch dog, set timeout 10 sec.

```

Void xxxx(void)
{

    ProcAdd = (SETINT2PROC) GetProcAddress(hinstLib, "WDT_LPC_SetTimeRange");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)(2, 10)) // Group 2, 10 sec
        {
            ShowMessage("WDT_LPC_SetTimeRange Fail");
        }
    }

    ProcAdd = (SETINTUINTPROC) GetProcAddress(hinstLib, "WDT_LPC_SetConfig");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)(2, 0x60)) // Group 2 , clear status, set unit is second. , and start watch dog
        {
            ShowMessage("WDT_LPC_SetConfig Fail");
        }
    }
}
    
```

**Group 3 Example:** F81216 Start watch dog, set timeout 10 sec.

```
Void xxxx(void)
{

    ProcAdd = (SETINT2PROC) GetProcAddress(hinstLib, "WDT_LPC_SetConfig");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)(3, 0x3)) // Group 3 , clear status and set unit is second.
        {
            ShowMessage("WDT_LPC_SetConfig Fail");
        }
    }
    ProcAdd = (SETINT2PROC) GetProcAddress(hinstLib, "WDT_LPC_SetTimeRange");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)(3, 10 )) // Group 3, 10 sec , and start watch dog
        {
            ShowMessage("WDT_LPC_SetTimeRange Fail");
        }
    }
}
```

## 2.5 Fan Control Function

### Function List

ID	Function Name	Description
2.5.1	GetFanMode	Get fan mode.
2.5.2	SetFanMode	Set fan mode.
2.5.3	GetLPCFanSpeed	Get fan speed.
2.5.4	GetLPCFanExpectSpeed	Get fan expect speed.
2.5.5	SetLPCFanExpectSpeed	Set fan expect speed.
2.5.6	GetLPCMaxFanSpeed	Get fan maximum speed.
2.5.7	SetLPCMaxFanSpeed	Set fan maximum speed.
2.5.8	GetLPCTemperatureThreshold	Get fan temperature threshold.
2.5.9	SetLPCTemperatureThreshold	Set fan temperature threshold.
2.5.10	GetLPCFanSpeedSectionValue	Get fan speed section threshold.
2.5.11	SetLPCFanSpeedSectionValue	Set fan speed section threshold.

### 2.5.1 GetFanMode

**int GetFanMode(int \*FanMode)**

**Function :** Get current Fan Mode setting value.

Ex: RPM mode 、 Duty mode 、 RPM manual 、 Duty manual

**Description :** None

**Return Value :**

If the function succeeds, the return value is nonzero (TRUE). If the function fails, the return value is zero (FALSE). ◦

**Example :**

```
void XXXX(void)
{
    int FanMode=0;
    GETINT1PROC ProcAdd;

    ProcAdd = (GETINT1PROC) GetProcAddress(hinstLib, " GetFanMode ");
    if (NULL != ProcAdd)
```

```

    {
        if (!(*ProcAdd>(&FanMode))
    {
        // Fail to get this value
    }
}
else
{
    // Fail to get this procedure address
}
}
    
```

## 2.5.2 SetFanMode

**int SetFanMode(int FanMode)**

**Function** : Set Fan Mode.

Ex: RPM mode 、 Duty mode 、 RPM manual 、 Duty manual

**Description** : None

**Return Value** :

If the function succeeds, the return value is nonzero (TRUE). If the function fails, the return value is zero (FALSE). ◦

**Example** :

```

void XXXX(void)
{
    int FanMode = 0x3F; // all fan are duty manual
    SETINTPROC ProcAdd;

    ProcAdd = (SETINTPROC) GetProcAddress(hinstLib, " SetFanMode ");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)(FanMode))
        {
            // Fail to get this value
        }
    }
}
else
    
```

```

    {
        // Fail to get this procedure address
    }
}
    
```

## 2.5.3 GetLPCFanSpeed

**int GetLPCFanSpeed(int Group, int \*RPMValue)**

**Function** : Get current FAN group speed.

**Parameter** :

Group : Fan Group.

RPMValue : return FAN speed °

**Description** :

Maximum 9999 rpm °

**Return Value** :

If the function succeeds, the return value is nonzero (TRUE). If the function fails, the return value is zero (FALSE). °

**Example** :

```

void XXXX(void)
{
    int RPMValue = 0;
    GETINT2PROC ProcAdd;

    ProcAdd = (GETINT2PROC) GetProcAddress(hinstLib, " GetLPCFanSpeed ");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 1, &RPMValue))
            // Fail to get this value

        if (!(*ProcAdd)( 2, &RPMValue))
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
    
```



## 2.5.4 GetLPCFanExpectSpeed

**int GetLPCFanExpectSpeed(int Group, WORD\* ExpectFanRPM)**

**Function** : Get expect FAN speed.

**Parameter** :

Group : FAN Group.

ExpectFanRPM : return expect FAN speed ◦

**Return Value** :

If the function succeeds, the return value is nonzero (TRUE). If the function fails, the return value is zero (FALSE). ◦

**Example** :

```
void XXXX(void)
{
    WORD ExpectFanCount = 0;
    GETINTWORDPROC ProcAdd;

    ProcAdd = (GETINTWORDPROC) GetProcAddress(hinstLib, " GetLPCFanExpectSpeed ");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 1, &ExpectFanCount))
            // Fail to get this value

        if (!(*ProcAdd)( 2, &ExpectFanCount))
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
```

## 2.5.5 SetLPCFanExpectSpeed

**int SetLPCFanExpectSpeed(int Group, WORD ExpectFanRPM)**

**Function** : Set expect FAN speed.

**Parameter** :

Group : FAN Group

ExpectFanRPM : expect FAN speed.

**Description** :

This function uses under fan control mode is manual mode and needs to consider the FAN is Duty or RPM mode, detailed please consult IC Datasheet.

**Return Value** :

If the function succeeds, the return value is nonzero (TRUE). If the function fails, the return value is zero (FALSE). °

**Example** :

```
void XXXX(void)
{
    WORD ExpectFanRPM = 0;    // maximum RPM is 0 ; maximum Duty is 0xFF
    SETINTWORDPROC ProcAdd;

    ProcAdd = (SETINTWORDPROC) GetProcAddress(hinstLib, " SetLPCFanExpectSpeed ");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 1, ExpectFanRPM))
            // Fail to get this value

        if (!(*ProcAdd)( 2, ExpectFanRPM))
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
```

## 2.5.6 GetLPCMaxFanSpeed

**int GetLPCMaxFanSpeed(int Group, int \*MaxRPMValue)**

**Function** : Get Fan maximum speed.

**Parameter** :

Group : FAN Group.

MaxRPMValue : return Fan maximum speed.

**Description** : None ◦

**Return Value** :

If the function succeeds, the return value is nonzero (TRUE). If the function fails, the return value is zero (FALSE). ◦

**Example** :

```
void XXXX(void)
{
    int MaxRPMValue = 0;
    GETINT2PROC ProcAdd;

    ProcAdd = (GETINT2PROC) GetProcAddress(hinstLib, " GetLPCMaxFanSpeed ");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 1, &MaxRPMValue))
            // Fail to get this value

        if (!(*ProcAdd)( 2, &MaxRPMValue))
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
```

## 2.5.7 SetLPCMaxFanSpeed

**int SetLPCMaxFanSpeed(int Group, int MaxRPMValue)**

**Function** : Set Fan maximum speed

**Parameter** :

Group : FAN Group.

MaxRPMValue : Fan maximum speed.

**Description** :

Parameter MaxRPMValue must be operated in RPM mode.

**Return Value** :

If the function succeeds, the return value is nonzero (TRUE). If the function fails, the return value is zero (FALSE). ◦

**Example** :

```
void XXXX(void)
{
    int MaxRPMValue = 0x177;    // ex: 4000rpm = 0x177 h ; (1500000 / 375 = 4000 rpm)
    SETINT2PROC ProcAdd;

    ProcAdd = (SETINT2PROC) GetProcAddress(hinstLib, " SetLPCMaxFanSpeed ");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 1, MaxRPMValue))
            // Fail to get this value

        if (!(*ProcAdd)( 2, MaxRPMValue))
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
```

## 2.5.8 GetLPCTemperatureThreshold

```
int GetLPCTemperatureThreshold( int Group,  
                               int *TemperatureThr1, int *TemperatureThr2,  
                               int *TemperatureThr3, int *TemperatureThr4)
```

**Function** : Get temperature threshold of FAN control.

**Parameter** :

Group : FAN Group.

TemperatureThr1 : return the 1st BOUNDARY temperature.

TemperatureThr2 : return the 2nd BOUNDARY temperature.

TemperatureThr3 : return the 3rd BOUNDARY temperature.

TemperatureThr4 : return the 4th BOUNDARY temperature.

**Description** : None ◦

**Return Value** :

If the function succeeds, the return value is nonzero (TRUE). If the function fails, the return value is zero (FALSE). ◦

**Example** :

```
void XXXX(void)  
{  
    GETINT5PROC  ProcAdd;  
  
    ProcAdd = (GETINT5PROC) GetProcAddress(hinstLib, " GetLPCTemperatureThreshold ");  
    if (NULL != ProcAdd)  
    {  
        if (!(*ProcAdd)( 1, &TemperatureThr1, &TemperatureThr2, &TemperatureThr3, &TemperatureThr4))  
            // Fail to get this value  
  
        if (!(*ProcAdd)( 2, &TemperatureThr1, &TemperatureThr2, &TemperatureThr3, &TemperatureThr4))  
            // Fail to get this value  
    }  
    else  
    {  
        // Fail to get this procedure address  
    }  
}
```

## 2.5.9 SetLPCTemperatureThreshold

```
int SetLPCTemperatureThreshold( int Group,
                               int TemperatureThr1, int TemperatureThr2,
                               int TemperatureThr3, int TemperatureThr4)
```

**Function** : Set temperature threshold of FAN control.

**Parameter** :

Group : FAN Group.

TemperatureThr1 : The 1st BOUNDARY temperature.

TemperatureThr2 : The 2nd BOUNDARY temperature.

TemperatureThr3 : The 3rd BOUNDARY temperature.

TemperatureThr4 : The 4th BOUNDARY temperature.

**Description** : None ◦

**Return Value** :

If the function succeeds, the return value is nonzero (TRUE). If the function fails, the return value is zero (FALSE). ◦

**Example** :

```
void XXXX(void)
{
    SETINT5PROC  ProcAdd;

    ProcAdd = (SETINT5PROC) GetProcAddress(hinstLib, " SetLPCTemperatureThreshold ");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 1, 60,50,40,30))
            // Fail to get this value

        if (!(*ProcAdd)( 2, 60,50,40,30))
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
```

## 2.5.10 GetLPCFanSpeedSectionValue

```
int GetLPCFanSpeedSectionValue ( int Group,
                                int *FanSpeed1, int *FanSpeed2, int *FanSpeed3, int *FanSpeed4, int *FanSpeed5)
```

**Function :** Get speed threshold of FAN control.

**Parameter :**

Group : FAN Group.

FanSpeed1 : return the 1st SEGMENT fan speed.

FanSpeed2 : return the 2nd SEGMENT fan speed.

FanSpeed3 : return the 3rd SEGMENT fan speed.

FanSpeed4 : return the 4th SEGMENT fan speed.

FanSpeed5 : return the 5th SEGMENT fan speed.

**Description :**

Parameter FanSpeed1~5, Percentage of the MAXIMUM fan speed ◦

**Return Value :**

If the function succeeds, the return value is nonzero (TRUE). If the function fails, the return value is zero (FALSE). ◦

**Example :**

```
void XXXX(void)
{
    GETINT6PROC  ProcAdd;

    ProcAdd = (GETINT6PROC) GetProcAddress(hinstLib, " GetLPCFanSpeedSectionValue ");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 1, &FanSpeed1, &FanSpeed2, &FanSpeed3, &FanSpeed4, &FanSpeed5))
            // Fail to get this value

        if (!(*ProcAdd)( 2, &FanSpeed1, &FanSpeed2, &FanSpeed3, &FanSpeed4, &FanSpeed5))
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
```



## 2.6 Temperature Function

### Function List

ID	Function Name	Description
2.6.1	GetLPCTemperatureValue	Get temperature value.

### 2.6.1 GetLPCTemperatureValue

**int GetLPCTemperatureValue(int Group, UCHAR \*TemperatureValue)**

**Function :** Get current temperature value

**Parameter :**

Group : Temperature Group.

TemperatureValue : Return current temperature value

**Description :**

Temperature range: 0 ~ 255 °

**Return Value :**

If the function succeeds, the return value is nonzero (TRUE). If the function fails, the return value is zero (FALSE). °

**Example :**

```
void XXXX(void)
{
    UCHAR temperatureValue = 0;
    GETINTUCHAR2PROC ProcAdd;

    ProcAdd = (GETINTUCHAR2PROC) GetProcAddress(hinstLib, " GetLPCTemperatureValue ");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( 1, &temperatureValue))
            // Fail to get this value

        if (!(*ProcAdd)( 2, , &temperatureValue))
            // Fail to get this value
    }
}
```

```

    }
    else
    {
        // Fail to get this procedure address
    }
}
    
```

## 2.7 Voltage Function

### Function List

ID	Function Name	Description
2.7.1	GetLPC3VccVoltage	Get VCC.
2.7.2	GetLPCVsbVoltage	Get VSB.
2.7.3	GetLPCVbatVoltage	Get VBAT.
2.7.4	GetLPCVinVoltage	Get Vinx.
2.7.5	GetLPC5VsbVoltage	Get 5VSB.
2.7.6	GetLPC5VaVoltage	Get 5VA.
2.7.7	GetLPC5VaVoltage	Get 3VA.

### 2.7.1 GetLPC3VccVoltage

**int GetLPC3VccVoltage(float \*VccVoltageValue)**

**Function :** Get Vcc voltage.

**Description :**

\*VccVoltageValue x Dividing Resistor = REAL VCC voltage

**Return Value :**

If the function succeeds, the return value is nonzero (TRUE). If the function fails, the return value is zero (FALSE). ◦

**Example :**

```

void XXXX(void)
{
    float vccVoltageValue = 0;
    
```

```

GETFLOATPROC ProcAdd;

ProcAdd = (GETFLOATPROC) GetProcAddress(hinstLib, " GetLPC3VccVoltage ");
if (NULL != ProcAdd)
{
    if (! (*ProcAdd)( & vccVoltageValue))
        // Fail to get this value
}
else
{
    // Fail to get this procedure address
}
}
    
```

## 2.7.2 GetLPCVsbVoltage

**int GetLPCVsbVoltage (float \* VsbVoltageValue)**

**Function :** Get Vsb voltage.

**Description :**

\* VsbVoltageValue x Dividing Resistor = REAL VSB voltage

**Return Value :**

If the function succeeds, the return value is nonzero (TRUE). If the function fails, the return value is zero (FALSE). ◦

**Example :**

```

void XXXX(void)
{
    float vsbVoltageValue = 0;
    GETFLOATPROC ProcAdd;

    ProcAdd = (GETFLOATPROC) GetProcAddress(hinstLib, " GetLPCVsbVoltage ");
    if (NULL != ProcAdd)
    {
        if (! (*ProcAdd)( &vsbVoltageValue))
            // Fail to get this value
    }
}
    
```

```

else
{
    // Fail to get this procedure address
}
}
    
```

## 2.7.3 GetLPCVbatVoltage

**int GetLPCVbatVoltage (float \* VbatVoltageValue)**

**Function :** Get VBat voltage.

**Description :**

\* VbatVoltageValue x Dividing Resistor = REAL VBAT voltage

**Return Value :**

If the function succeeds, the return value is nonzero (TRUE). If the function fails, the return value is zero (FALSE). ◦

**Example :**

```

void XXXX(void)
{
    float vbatVoltageValue = 0;
    GETFLOATPROC ProcAdd;

    ProcAdd = (GETFLOATPROC) GetProcAddress(hinstLib, " GetLPCVbatVoltage");
    if (NULL != ProcAdd)
    {
        if (! (*ProcAdd)( &vbatVoltageValue))
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
    
```

## 2.7.4 GetLPCVinVoltage

**int GetLPCVinVoltage(int Group, float \*Vin)**

**Function** : Get Vin1 - Vin8 Voltage.

**Parameter** :

Group : Voltage Group.

Vin : VinX Value.

**Description** :

\*Vin x Dividing Resistor = REAL VIN voltage

**Return Value** :

If the function succeeds, the return value is nonzero (TRUE). If the function fails, the return value is zero (FALSE). °

**Example** :

```
void XXXX(void)
{
    float vin = 0;
    GETINTFLOATPROC ProcAdd;

    ProcAdd = (GETINTFLOATPROC) GetProcAddress(hinstLib, "GetLPCVinVoltage");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)(1, &vin))
            // Fail to get this value

        if (!(*ProcAdd)(2, &vin))
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
```

## 2.7.5 GetLPC5VsbVoltage

**int GetLPC5VsbVoltage(float \*Vsb5VoltageValue)**

**Function** : Get 5Vsb voltage.

**Description** :

\* Vsb5VoltageValue x Dividing Resistor = REAL 5VSB voltage

**Return Value** :

If the function succeeds, the return value is nonzero (TRUE). If the function fails, the return value is zero (FALSE). °

**Example** :

```
void XXXX(void)
{
    float vsb5VoltageValue = 0;
    GETFLOATPROC ProcAdd;

    ProcAdd = (GETFLOATPROC) GetProcAddress(hinstLib, " GetLPC5VsbVoltage ");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( &vsb5VoltageValue))
            // Fail to get this value
    }
    else
    {
        // Fail to get this procedure address
    }
}
```

## 2.7.6 GetLPC5VaVoltage

**int GetLPC5VaVoltage(float \*Va5VoltageValue)**

**Function** : Get 5VA voltage.

**Description** :

\* Va5VoltageValue x Dividing Resistor = REAL 5VA voltage

**Return Value :**

If the function succeeds, the return value is nonzero (TRUE). If the function fails, the return value is zero (FALSE). ◦

**Example :**

```
void XXXX(void)
{
    float va5VoltageValue = 0;
    GETFLOATPROC ProcAdd;

    ProcAdd = (GETFLOATPROC) GetProcAddress(hinstLib, " GetLPC5VaVoltage ");
    if (NULL != ProcAdd)
    {
        if (!(*ProcAdd)( & va5VoltageValue))
            // Fail to get this value
        }
        else
        {
            // Fail to get this procedure address
        }
    }
}
```

## 2.7.7 GetLPC3VaVoltage

**int GetLPC3VaVoltage(float \*Va3VoltageValue)**

**Function :** Get 3VA voltage.

**Description :**

\* Va3VoltageValue x Dividing Resistor = REAL 3VA voltage

**Return Value :**

If the function succeeds, the return value is nonzero (TRUE). If the function fails, the return value is zero (FALSE). ◦

**Example :**

```
void XXXX(void)
{
    float va3VoltageValue = 0;
    GETFLOATPROC ProcAdd;
```

```
ProcAdd = (GETFLOATPROC) GetProcAddress(hinstLib, " GetLPC3VaVoltage ");
if (NULL != ProcAdd)
{
    if (! (*ProcAdd)( & va3VoltageValue))
        // Fail to get this value
    }
else
{
    // Fail to get this procedure address
}
}
```

## 3. SMBus DLL Function

### 3.1 Support Fintek I2C IC

F75308	F75387	F75113			

### 3.2 SMBus Function List

<b>SMBus Initial Function</b>
int InitialSMBus(unsigned long I2cAddress);
<b>GPIO Function</b>
int GPIO_SMBus_R(int index, int *value, unsigned long reserved);
int GPIO_SMBus_W(int index, int value, unsigned long reserved);
<b>HWMonitor and Fan Control Function</b>
int GetI2CVccVoltage(float *VccVoltageValue, unsigned long reserved);
int GetI2C3VccVoltage(float *VccVoltageValue, unsigned long reserved);
int GetI2CVinVoltage(int Group, float *Vin, unsigned long reserved);
int GetI2CTemperatureValue(int Group, UCHAR *TemperatureValue, unsigned long reserved);
int GetI2CFanMode(int Group, int *FanMode, unsigned long reserved);
int SetI2CFanMode(int Group, int FanMode, unsigned long reserved);
int GetI2CFanSpeed(int Group, int *RPMValue, unsigned long reserved);
int GetI2CMaxFanSpeed(int Group, int *MaxRPMValue, unsigned long reserved);
int SetI2CMaxFanSpeed(int Group, int MaxRPMValue, unsigned long reserved);
int GetI2CFanExpectSpeed(int Group, WORD *ExpectFanRPM, unsigned long reserved);
int SetI2CFanExpectSpeed(int Group, WORD ExpectFanRPM, unsigned long reserved);
int GetI2CFanSpeedSectionValue(int Group, int *FanSpeed1, int *FanSpeed2, int *FanSpeed3, int *FanSpeed4, int

```

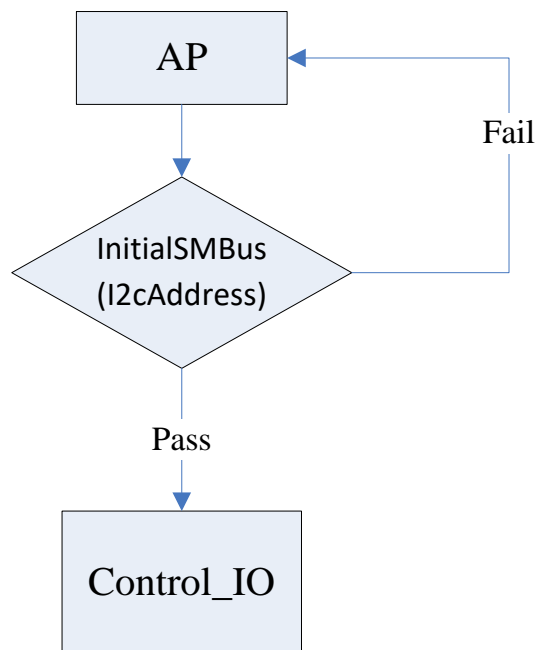
*FanSpeed5, unsigned long reserved);
int SetI2CFanSpeedSectionValue(int Group, int FanSpeed1, int FanSpeed2, int FanSpeed3, int FanSpeed4, int FanSpeed5,
unsigned long reserved);
int GetI2CTemperatureThreshold(int Group, int *TemperatureThr1, int *TemperatureThr2, int *TemperatureThr3, int
*TemperatureThr4, unsigned long reserved);
int SetI2CTemperatureThreshold(int Group, int TemperatureThr1, int TemperatureThr2, int TemperatureThr3, int
TemperatureThr4, unsigned long reserved);
    
```

## WDT Function

```

int WDT_SMBus_SetConfig(unsigned int value, unsigned long reserved);
int WDT_SMBus_SetTimeRange(unsigned int value, unsigned long reserved);
int WDT_SMBus_GetConfig(unsigned int *value, unsigned long reserved);
int WDT_SMBus_GetTimeRange(unsigned int *value, unsigned long reserved);
    
```

### 3.3 SMBus Function Flow

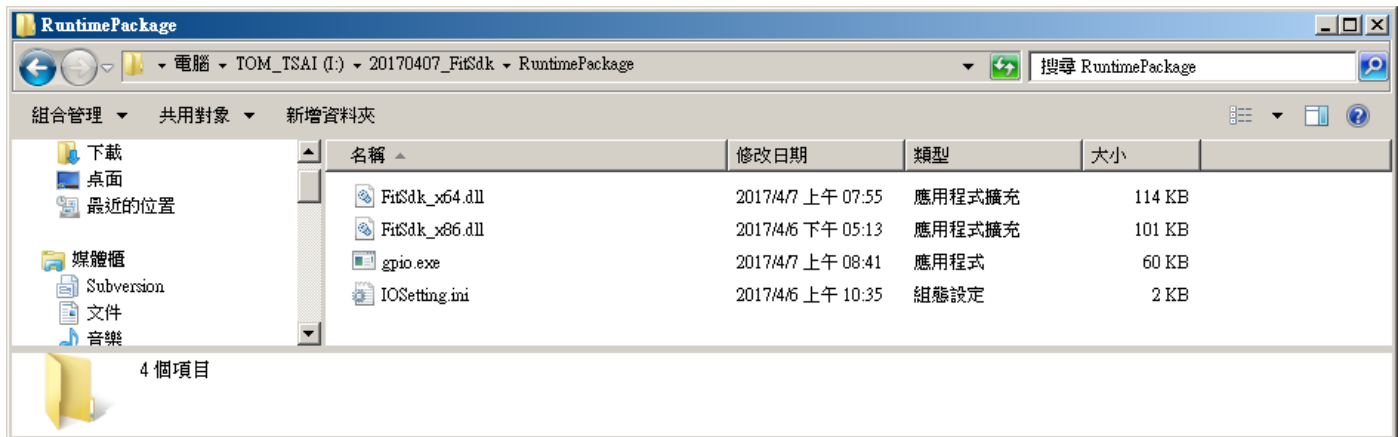


66

12/3/2025

## 4. Use Library

Copy .exe, IOSetting.ini and FitSdk\_x86/64.dll in the same directory.



### 4.1 Load Library

```

HINSTANCE hinstLib;
hinstLib = LoadLibrary("FitSdk_x86/64.dll");
if (hinstLib == NULL)
{
    printf("Load fail FitSdk_x86/64.dll");
    return 1;
}
    
```

### 4.2 Unload Library

```

if (hinstLib != NULL)
{
    FreeLibrary(hinstLib);
}
    
```

## 4.3 DLL Parameter

```
// Get parameter type
typedef int (*GETINT1PROC) (int*);
typedef int (*GETFLOATPROC)(float*);
typedef int (*GETINTUCHAR2PROC)(int, UCHAR*);
typedef int (*GETINT2PROC) (int, int*);
typedef int (*GETINTFLOATPROC)(int, float*);
typedef int (*GETINTWORDPROC) (int, WORD*);
typedef int (*GETUCHAR2PROC) (unsigned char, unsigned char *);

// Set parameter type
typedef int (*SETINTPROC) (int);
typedef int (*SETINT2PROC) (int,int);
typedef int (*SETINTWORDPROC) (int, WORD);
typedef int (*SETINT5PROC)(int, int, int, int, int);
typedef int (*SETINT6PROC)(int, int, int, int, int, int);
typedef int (*SETUCHAR2PROC) (unsigned char, unsigned char);

// Other
typedef int (*SETVOIDPROC)();
```

## 5. Known issues

### 5.1 Microsoft Security Advisory 3033929

Installation on 64-bit versions of Windows 7 and Windows Server 2008 R2 fails if Microsoft security update 3033929 is not installed. Because SDK driver is signed by SHA-256 certificate. Without this update Windows 7 and Windows Server 2008 R2 does not recognize the signature properly and fails to load the driver. A security issue has been identified in a Microsoft software product that could affect your system. You can help protect your system by installing this update KB3033929 from Microsoft. For a complete listing of the issues that are included in this update, see the associated Microsoft Knowledge Base article (<https://technet.microsoft.com/en-us/library/security/3033929>). After you install this update, you may have to

restart your system.