XWboard SDK API Manual

(WinCE Based ((eVC & .NET))

Version 1.0.0, August 2010

Written by Sean
Edited by Anna Huang

Warranty

All products manufactured by ICP DAS are under warranty regarding defective materials for a period of one year, beginning from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for any damage resulting from the use of this product.ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, not for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

The names used for identification only may be registered trademarks of their respective companies.

Table of Contents

Preface

This guide introduces XWboard Software Development Kit (SDK). It provides an overview of what you can do with the SDK and the technologies that are available to you through the SDK.

- **Software Development Tool**

  Microsoft eMbedded Visual C++

  Visual Basic.net

  Visual C#

- **Requirements**

  The WinPAC SDK only supports NET Framework 2.0 or above.

- **Installation Path**

  After installing the WinPAC SDKs (.msi), a number of functions can be installed on the Host PC, and this installation puts the header files, libraries into the following public places so they are easily changed by update the WinPAC SDKs.

  Header files:

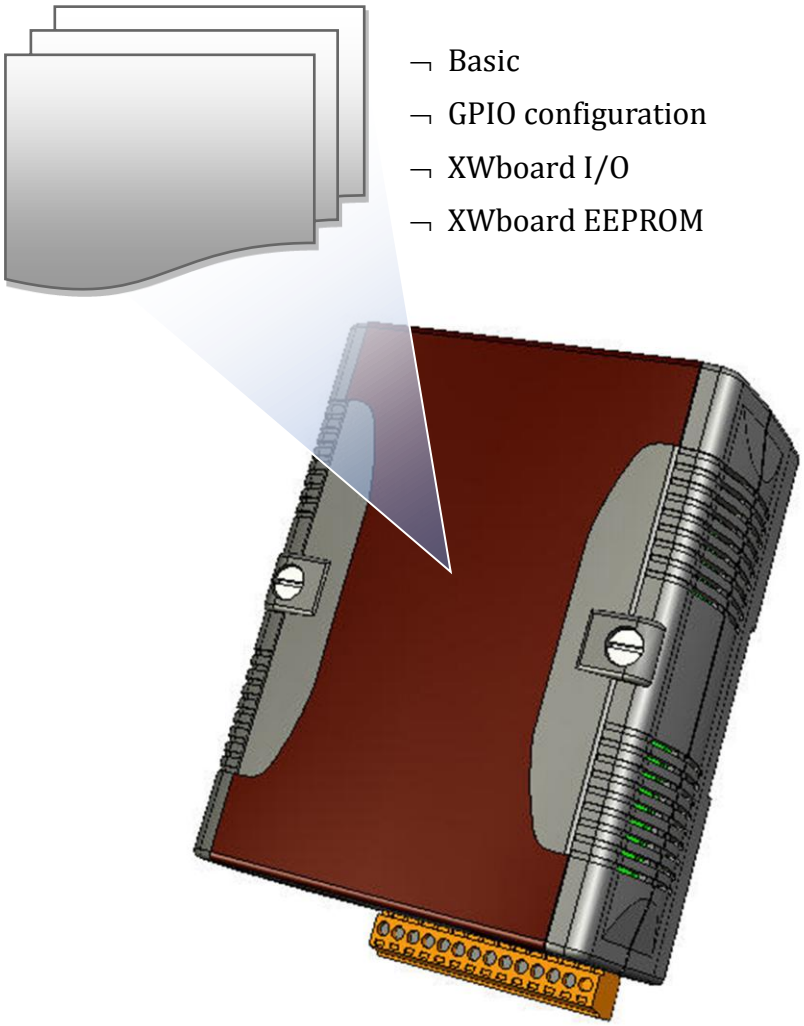
  C:\Program Files\Windows CE Tools\wce500\PAC270\Icpdas\Include\ARMV4I\

  Libraries:

  C:\Program Files\Windows CE Tools\wce500\PAC270\Icpdas\Lib\ARMV4I\

Overview of the XWboard SDK

The XWboard SDK enables applications to exploit the power of WinPAC 5000. The SDK consists of the following API and functional categories:



1. Basic Functions

1.1. pac_GetXWBOARDSDKVersion

This function retrieves the SDK version.

Syntax

```
void pac_GetXWBOARDSDKVersion(LPSTR sdk_version);
```

Parameters

sdk_version

[out] A pointer to a variable that specifies the SDK version

Return Values

None

Examples

[eVC]

```
char SDK[32];
pac_GetXWBOARDSDKVersion (SDK);
```

[C#]

```
string SDK;
SDK = XWBoard.pac_ GetXWBOARDSDKVersion ();
```

1.2. pac_EnableLed

This function sets the LED on/off.

Syntax

```
void pac_ EnableLed(int Ln, bool bFlag);
```

Parameters

Ln

[in] The number of LED

0: RUN

1: L1

2: L2

bFlag

True: Turn on the LED.

False: Turn off the LED.

Return Values

None

Examples

[eVC]

```
pac_ EnableLed(1, true);
```

[C#]

```
XWBoard.pac_ EnableLed(1, true);
```

1.3. pac_ToggleLed

This function toggle the LED on/off.

Syntax

```
void pac_ToggleLED(int Ln);
```

Parameter

Ln

[in] The number of LED

0: RUN

1: L1

2: L2

Return Values

None

Examples

[eVC]

```
pac_ToggleLED(1);
```

[C#]

```
XWBoard.pac_ToggleLED (1);
```

2. GPIO configuration

User can use those functions to control allowed programmable pin of I/O expansion bus.
If the user wants to design new IO expansion board, these functions in this section helps you to develop your own expansion board.

2.1. Get_GPIO_Pin_Status

This function is used to get the status of the GPIO pin.

Syntax

```
int Get_GPIO_Pin_Status(unsigned int pin);
```

Parameters

pin

[in] 11: GPIO 11- 48MHz Clock

16: GPIO 16

17: GPIO 17

81: GPIO 81

86: GPIO 86

87: GPIO 87

117: GPIO 117

118: GPIO 118

Return Values

0: Low level

1: High level

< 0 : error

Examples

None

2.2. Set_GPIO_Pin_Status

This function is used to set the status of the GPIO pin.

Syntax

```
void Set_GPIO_Pin_Status (unsigned int pin, unsigned int value);
```

Parameters

pin

[in] 11: GPIO 11 - 48MHz Clock

16: GPIO 16

17: GPIO 17

81: GPIO 81

86: GPIO 86

87: GPIO 87

117: GPIO 117

118: GPIO 118

Value

0: output 0

1: output 1

Examples

None

2.3. SetGPIOInput

GPIO pin configured as an input.

Syntax

```
void setGPIOInput(unsigned int pin);
```

Parameters

pin

[in] 11: GPIO 11 - 48MHz Clock

16: GPIO 16

17: GPIO 17

81: GPIO 81

86: GPIO 86

87: GPIO 87

117: GPIO 117

118: GPIO 118

Return Values

None

Examples

None

2.4. SetGPIOOutput

This function is used to set the status of the GPIO pin.

Syntax

```
void setGPIOOutput(unsigned int pin, unsigned int value);
```

Parameters

pin

[in] 11: GPIO 11 - 48MHz Clock

16: GPIO 16

17: GPIO 17

81: GPIO 81

86: GPIO 86

87: GPIO 87

117: GPIO 117

118: GPIO 118

Value

0: output 0

1: output 1

Return Values

None

Examples

None

2.5. SetGPIOAltFunc

Set GPIO pin alternate function.

Syntax

```
Void setGPIOAltFunc(unsigned int pin, unsigned int af, unsigned int mode);
```

Parameters

pin

[in] 11: GPIO 11- 48MHz Clock

16: GPIO 16

17: GPIO 17

81: GPIO 81

86: GPIO 86

87: GPIO 87

117: GPIO 117

118: GPIO 118

af

0 : Alternate Function 0 --> The corresponding GPIO pin is used for as a general purpose I/O

1 : Alternate Function 1 --> The corresponding GPIO pin is used for its alternate function 1

2 : Alternate Function 1 --> The corresponding GPIO pin is used for its alternate function 2

3 : Alternate Function 1 --> The corresponding GPIO pin is used for its alternate function 3

Mode

0: output mode

1: input mode

Return Values

None

Examples

None

2.6. SetGPIO11_ClockOut

Set GPIO11 to generate 48M Hz clock output.

Syntax

```
void setGPIO11_ClockOut(unsigned int enable);
```

Parameters

enable

Enable: 0: disable

1: enable

Return Values

None

Examples

None

2.7. SetGPIO11_PWM

Set GPIO16/17 to generate PWM signal output.

Syntax

```
void setGPIO_PWM(unsigned int pin,DWORD dwFrequency,unsigned int Duty,int Enable);
```

Parameters

pin

16: GPIO 16

17: GPIO 17

dwFrequency

1~20K Hz

Duty

1~100

Return Values

None

Examples

None

3. XWBoard IO API

XWboard API supports to operate XWboard IO modules plugged on the I/O expansion bus.

3.1. XWB_Init(int XWID)

The function is used to initialize the XWboard.

Syntax

```
int XWB_init(
        int xwid
);
```

Parameters

Xwid

Specific an XWboard ID

0 means the WinPAC 5000 will detect which XWboard plug in automatically.

Return Values

0: success
-1: The XWboard is an undefined board.

Examples

[eVC]

```
XWB_init(0);
```

[C#]

```
XWBoard.XWB_init(0);
```

3.2. XWB_WriteDO

This function writes the DO values to DO modules.

Syntax

```
bool XWB_WriteDO(
        int iTotalChannel,
        DWORD lDO_Value
);
```

Parameters

iTotalChannel

[in] The total number of DO channels of the DO modules.

lDO_Value

[in] 1 is to turn on the DO channel; 0 is off.

Return Values

Return true if success, otherwise false.

Examples

[eVC]

```
int total_channel = 8;
DWORD do_value = 3;      // turn on the channel 0,1
XWB_init(0);      // Initialize XWboard
bool ret = XWB_WriteDO(total_channel , do_value );
```

[C#]

```
int total_channel = 8;
uint do_value = 3;      // turn on the channel 0,1
XWBoard.XWB_init(0); // Initialize XWboard
bool ret = XWBoard.XWB_WriteDO(total_channel , do_value );
```

3.3. XWB_WriteDOBit

This function writes a single bit of value to the DO module, that is, only the channel corresponding to the bit is changed.

Syntax

```
bool XWB_WriteDOBit(
        int iTotalChannel,
        int iChannel,
        int iBitValue
);
```

Parameters

iTotalChannel

[in] The total number of DO channels of the DO modules.

iChannel

[in] The DO channel to change.

iBitValue

[in] 1 is to turn on the DO channel; 0 is off.

Return Values

Return true if success, otherwise false.

Examples

[eVC]

```
int iChannel = 2;
int iDO_TotalCh = 8;
int iBitValue = 1;
XWB_init(0);      // Initialize XWboard
bool ret = XWB_WriteDOBit(iDO_TotalCh , iChannel , iBitValue );
```

[C#]

```
int iChannel = 2;
int iDO_TotalCh = 8;
int iBitValue = 1;
XWBoard.XWB_init(0);      // Initialize XWboard
bool ret = XWBoard.XWB_WriteDOBit(miDO_TotalCh , iChannel , iBitValue );
```

3.4. XWB_ReadDO

This function reads the DO value of the DO module.

Syntax

```
bool pac_ReadDO(
        int iTotalChannel,
        DWORD *lDO_Value
);
```

Parameters

iDO_TotalCh

[in] The total number of DO channels of the DO modules.

lDO_Value

[in] The pointer of the DO value to read from the DO module

Return Values

Return true if success, otherwise false.

Examples

[eVC]

```
int total_channel = 8;
DWORD do_value;
XWB_init(0);      // Initialize XWboard
bool ret = XWB_ReadDO(total_channel , &do_value );
```

[C#]

```
int total_channel = 8;
uint do_value = new uint();
XWBoard.XWB_init(0);      // Initialize XWboard
bool ret = XWBoard.pac_ReadDO(hPort, slot , total_channel , ref do_value );
```

3.5. XWB_ReadDI

This function reads the DI value of the DI module.

Syntax

```
bool XWB_ReadDI(
        int iTotalChannel,
        DWORD *lDI_Value
);
```

Parameters

iTotalChannel

[in] The total channels of the DI module.

lDI_Value

[out] The pointer to DI value to read back.

Return Values

Return true if success, otherwise false.

Examples

[eVC]

```
int iDI_TotalCh = 8;
DWORD lDI_Value;
XWB_init(0); // Initialize XWboard
bool ret = pac_ReadDI(iDI_TotalCh, &lDI_Value);
```

[C#]

```
int iDI_TotalCh = 8;
uint lDI_Value = new uint();
XWBoard.XWB_init(0); // Initialize XWboard
bool ret = XWBoard.pac_ReadDI(hPort, iSlot,iDI_TotalCh, ref lDI_Value);
```

3.6. XWB_ReadDILatch

This function reads the DI latch value of the DI module.

Syntax

```
bool XWB_ReadDILatch(
    int iTotalChannel,
    int iLatchType,
    DWORD *lDI_Latch_Value
);
```

Parameters

iTotalChannel

[in] The total number of the DI channels of the DI module.

iLatchType

[in] The latch type specified to read latch value back.

1 = latched high status

0 = latched low status

lDI_Latch_Value

[out] The pointer to the latch value read back from the DI module.

Return Values

Return true if success, otherwise false.

Examples

[eVC]

```
int iDI_TotalCh=8;
int iLatchType=0;
DWORD lDI_Latch_Value;
XWB_init(0);      // Initialize XWboard
bool ret = XWB_ReadDILatch(iDI_TotalCh, iLatchType, &lDI_Latch_Value);
```

[C#]

```
int iDI_TotalCh=8;
int iLatchType=0;
uint lDI_Latch_Value = new uint();
XWBoard.XWB_init(0);      // Initialize XWboard
bool ret = XWBoard.pac_ReadDILatch( iDI_TotalCh, iLatchType, ref lDI_Latch_Value);
```

3.7. XWB_ClearDILatch

This function clears the latch value of the DI module.

Syntax

```
bool XWB_ClearDILatch(void);
```

Parameters

None

Return Values

Return true if success, otherwise false.

Examples

[eVC]

```
XWB_init(0);      // Initialize XWboard
bool ret = XWB_ClearDILatch();
```

[C#]

```
XWBoard.XWB_init(0);      // Initialize XWboard
bool ret = XWB_ClearDILatch();
```

3.8. XWB_ReadDICNT

This function reads the counts of the DI channels of the DI module.

Syntax

```
bool XWB_ReadDICNT(
        int iTotalChannel,
        int iChannel,
        DWORD *lCounter_Value
);
```

Parameters

iTotalChannel

[in] Total number of the DI channels of the DI module.

iChannel

[in] The channel that the counter value belongs.

lCounter_Value

[out] The pointer to the counter value.

Return Values

Return true if success, otherwise false.

Examples

[eVC]

```
int iChannel =2;
int iDI_TotalCh=8;
DWORD lCounter_Value;
XWB_Init(0);      // Initialize XWboard
bool ret = xwb_ReadDICNT(iDI_TotalCh, iSlot,iChannel, &lCounter_Value);
```

[C#]

```
int iChannel =2;
int iDI_TotalCh=8;
uint lCounter_Value = new uint();
XWBoard.XWB_init(0);      // Initialize XWboard
bool ret = XWBoard.XWB_ReadDICNT(iDI_TotalCh, iChannel, ref lCounter_Value);
```

3.9. XWB_ClearDICNT

This function clears the counter value of the DI channel of the DI module.

Syntax

```
bool XWB_ClearDICNT(
        int iTotalChannel,
        int iChannel,
);
```

Parameters

iTotalChannel

[in] Total number of the DI channels of the DI module.

iChannel

[in] The channel that the counter value belongs.

Return Values

Return true if success, otherwise false.

Examples

[eVC]

```
int iChannel=2;
int iDI_TotalCh=8;
XWB_Init(0);      // Initialize XWboard
bool ret = XWB_ClearDICNT(,iDI_TotalCh, iChannel);
```

[C#]

```
int iChannel=2;
int iDI_TotalCh=8;
XWBoard. XWB_Init(0);      // Initialize XWboard
bool ret = XWBoard.XWB_ClearDICNT(iDI_TotalCh,iChannel);
```

3.10. XWB_ReadDICNTOverflow

This function is used to read the counter overflow value of the DI modules.

Syntax

```
bool pac_ReadDICNTOverflow(
        int iTotalChannel,
        int iChannel,
        int *iOverflow
);
```

Parameters

iTotalChannel

[in] Total number of the DI channels of the DI module.

iChannel

[in] The channel that reads the counter overflows value back from the DI module.

iOverflow

[out] The pointer to the counter overflow that is read back from the DI module.

Return Values

Return true if success, otherwise false.

Examples

[eVC]

```
int iChannel=0;
int iOverflow;
int iDI_TotalCh=8;
XWB_Init(0);      // Initialize XWboard
bool ret = pac_ReadDICNTOverflow(iDI_TotalCh,iChannel,&iOverflow);
```

[C#]

```
int iChannel = 0;
int iOverflow = new int();
int iDI_TotalCh=8;
XWBoard. XWB_Init(0);      // Initialize XWboard
bool ret = XWBoard. pac_ReadDICNTOverflow (iDI_TotalCh, iChannel, ref iOverflow);
```

3.11. XWB_WriteAO

This function writes the AO value to the AO modules.

Syntax

```
bool XWB_WriteAO(
        int iTotalChannel,
int iChannel,
        float fValue
);
```

Parameters

iTotalChannel

[in] The total number of the AO channels of the AO module.

iChannel

[in] The channel that is written thee AO value to.

float fValue

[in] The AO value to write to the AO module.

Return Values

Return true if success, otherwise false.

Examples

[eVC]

```
int iChannel=2;
int iAO_TotalCh=4;
float fValue=5;
XWB_Init(0);      // Initialize XWboard
bool ret = XWB_WriteAO(iAO_TotalCh,iChannel,,fValue);
```

[C#]

```
int iChannel=2;
int iAO_TotalCh=4;
float fValue=5;
XWBoard. XWB_Init(0);      // Initialize XWboard
bool ret = XWBoard.XWB_WriteAO(iAO_TotalCh,iChannel,fValue);
```

3.12. XWB_ReadAO

This function reads the AO value of the AO module.

Syntax

```
bool XWB_ReadAO(
    int iTotalChannel,
    int iChannel,
    float *fValue
);
```

Parameters

iTotalChannel

[in] The total number of the AO channels of the AO module.

iChannel

[in] Read the AO value from the channel.

float fValue

[in] The pointer to the AO value that is read back from the AO module.

Return Values

Return true if success, otherwise false.

Examples

[eVC]

```
int iChannel=2;
int iAO_TotalCh=4;
float fValue;
XWB_Init(0);      // Initialize XWboard
bool ret = XWB_ReadAO(iAO_TotalCh, iChannel, &fValue);
```

[C#]

```
int iChannel=2;
int iAO_TotalCh=4;
float fValue = new float();
XWBoard. XWB_Init(0);      // Initialize XWboard
bool ret = XWBoard.XWB_ReadAO(iAO_TotalCh,iChannel,ref fValue);
```

3.13. XWB_ReadAI

This function reads the engineering-mode AI value of the AI module.

Syntax

```
bool pac_ReadAI(
        int iTotalChannel,
        int iChannel,
        float *fValue
);
```

Parameters

iTotalChannel

[in] The total number of the AI channels of the AI module.

iChannel

[in] Read the AI value from the channel.

fValue

[in] The pointer to the AI value that is read back from the AI module.

Return Values

Return true if success, otherwise false.

Examples

[eVC]

```
int iChannel=2;
int iAI_TotalCh=8;
float fValue;
XWB_Init(0);      // Initialize XWboard
bool ret = XWB_ReadAI(iAI_TotalCh,,iChannel, &fValue);
```

[C#]

```
int iChannel=2;
int iAI_TotalCh=8;
float fValue = new float();
XWBoard. XWB_Init(0);      // Initialize XWboard
bool ret = XWBoard.XWB_ReadAI(iAI_TotalCh, iChannel, ref fValue);
```

3.14. XWB_ReadAIHex

This function reads the 2's complement-mode AI value of the AI module.

Syntax

```
bool XWB_ReadAIHex(
        int iTotalChannel,
        int iChannel,
        int *iValue
);
```

Parameters

iTotalChannel

[in] The total number of the AI channels of the AI module.

iChannel

[in] Read the AI value from the channel.

iValue

[in] The pointer to the AI value that is read back from the AI module

Return Values

Return true if success, otherwise false.

Examples

[eVC]

```
int iChannel=2;
int iAI_TotalCh=8;
int iValue;
XWB_Init(0);      // Initialize XWboard
bool ret = XWB_ReadAIHex(iAI_TotalCh, iChannel, &iValue);
```

[C#]

```
int iChannel=2;
int iAI_TotalCh=8;
int iValue = new int();
XWBoard. XWB_Init(0);      // Initialize XWboard
bool ret = XWBoard.XWB_ReadAIHex(iAI_TotalCh, iChannel, ref iValue);
```

3.15. XWB_ReadAIAll

This function reads all the AI values of all channels in engineering-mode of the AI module.

Syntax

```
bool XWB_ReadAIAll(
        float fValue[]
);
```

Parameters

fValue[]

[out] The array contains the AI values that read back from the AI module.

Return Values

Return true if success, otherwise false.

Examples

[eVC]

```
float fValue[8];
XWB_Init(0);      // Initialize XWboard
bool ret = XWB_ReadAIAll(fValue);
```

[C#]

```
float[] fValue = new float[8];
XWBoard. XWB_Init(0);      // Initialize XWboard
bool ret =XWBoard.XWB_ReadAIAll(fValue);
```

3.16. XWB_ReadAIAllHex

This function reads all the AI values of all channels in 2's complement-mode of the AI module.

Syntax

```
bool XWB_ReadAIAllHex(
        int iValue[]
);
```

Parameters

iValue[]

[out] The array contains the AI values that read back from the AI module.

Return Values

Return true if success, otherwise false.

Examples

[eVC]

```
int iValue[8];
XWB_Init(0);      // Initialize XWboard
bool ret = XWB_ReadAIAllHex(iValue);
```

[C#]

```
int[] iValue = new int[8];
XWBoard. XWB_Init(0);      // Initialize XWboard
bool ret = XWBoard.XWB_ReadAIAllHex(iValue);
```

4. XWBoard EEPROM API

The EEPROM on the XWboard series is 24LC16, contains 8 blocks (block 0 to 7). Each block has 256 bytes (address 0 to 255), so the total size of the EEPROM is 2048 (2K) bytes. The block 7 is reserved for the XWboard series.

The default mode for EEPROM is write-protected mode.

Before data can be written to the EEPROM, EE_WriteEnable() must be called to set EEPROM to write-enabled mode. After a write operation to the EEPROM is completed, it is recommended that the EE_WriteProtect() is called to set the EEPROM to write-protect mode.

4.1. XEE_Init

Initialize the EEPROM on XWboard.
The functions (XEE_xxxxxx) must be used after XEE_Init is called.

Syntax

```
int XEE_Init(
int clk_pin, int sda_pin, int wp_pin, int need_pullhigh
);
```

Parameters

Clk_pin

[in] The GPIO pin number of the CPU that is connected to the CLK pin of the 24LC16
sda_pin

[in] The GPIO pin number of the CPU that is connected to the SDA pin of the 24LC16
wp_pin

The PIO pin number of the CPU that is connected to the WP pin of the 24LC16

need_pullhigh

If the sda_pin is set to input mode\

0: Pullhigh resistance is not needed.

1: Pullhigh resistance is needed.

Return Values

Return true if success, otherwise false.

Examples

[eVC]

```
XEE_Init (11, 87, 17, 0);
/* GPIO 11 configured as CLK pin
   GPIO 87 configured as SDA pin without Pull-high resistance.
   GPIO 17 configured as WP pin
*/
```

[C#]

```
XWBoard. XEE_Init (11, 87, 17, 0);
/* GPIO 11 configured as CLK pin
   GPIO 87 configured as SDA pin without Pull-high resistance.
   GPIO 17 configured as WP pin
*/
```

4.2. XEE_InitByName

Initialize the EEPROM on XWboard using the XWBoard ID
The functions (XEE_xxxxxx) can be used after XEE_InitByName is called.
XEE_InitByName is the same as XEE_Init except for the input parameters.
The user just chooses either XEE_Init or XEE_InitByName to initialize the EEPROM.

Syntax

```
int XEE_InitByName(
int XWboard
);
```

Parameters

XWboard

[in] Specific an XWboard ID

Refer to XWboardSDK.h to get the definition of XWboard ID.

```
#define _XW107_     107
#define _XW110_     110
#define _XW201_     201
#define _XW202_     202
#define _XW203_     203
#define _XW205_     205
#define _XW206_     206
#define _XW300_     300
#define _XW301_     301
#define _XW302_     302
#define _XW303_     303
#define _XW304_     304
#define _XW305_     305
#define _XW306_     306
```

#define _XW308_ 308
#define _XW309_ 309
#define _XW310_ 310
#define _XW314_ 314

Return Values

On success
-1 The board is unsupported.

Examples

[eVC]

```
XEE_InitByName(_XW300_);      //Initialize the XW300 board
```

XEE_InitByName [C#]

```
XWBoard. XEE_InitByName(_XW300_);      //Initialize the XW300 board
```

4.3. XEE_WriteEnable

This function sets the EEPROM to write-enabled mode.

Syntax

```
void XEE_WriteEnable(
void
);
```

Parameters

None

Return Values

None

Examples

[eVC]

```
XEE_InitByName(_XW300_);
XEE_WriteEnable ();
```

[C#]

```
XWBoard.XEE_InitByName(_XW300_);
XWBoard. XEE_WriteEnable();
```

4.4. XEE_WriteProtect

This function sets the EEPROM to write-protected mode.

Syntax

```
int XEE_WriteProtect (
void
);
```

Parameters

None

Return Values

None

Examples

[eVC]

```
XEE_InitByName(_XW300_);
XEE_WriteProtect();
```

[C#]

```
XWBoard.XEE_InitByName(_XW300_);
XWBoard. XEE_WriteProtect();
```

4.5. XEE_RandomRead

Read data (one byte) from the specific block and address of the EEPROM memory.

Syntax

```
int XEE_RandomRead(
int Block, int Addr
);
```

Parameters

Block

[in] 0 to 7. (Total is 8 blocks)

Addr

[in] 0 to 255. (Each block is 256 bytes)

Return Values

>= 0 The value in the Block:Addr address.
-9 (AddrError) Addr is invalid.
-10 (BlockError) Block is invalid.

Examples

[eVC]

```
unsigned char ucData;
XEE_InitByName(_XW300_);
ucData =XEE_RandomRead (0,0x10);
//read a byte data from address 0x10 block 0 on EEPROM
```

[C#]

```
Byte bData;
XWBoard.XEE_InitByName(_XW300_);
bData = XWBoard.XEE_RandomRead (0,0x10);
//read a byte data from address 0x10 block 0 on EEPROM
```

4.6. XEE_ReadNext

Read data (one byte) from the next address of the specified block and address on calling the XEE_RandomRead.

Syntax

```
int XEE_ReadNext(
int Block
);
```

Parameters

Block

[in] 0 to 7. (Total is 8 blocks)

Return Values

Return the EEPROm data
-10 (BlockError) StartBlock is invalid.

Examples

[eVC]

```
unsigned char ucData, ucData1;
XEE_InitByName(_XW300_);
//read a byte data from address 0x10 block 0 on EEPROM
ucData =XEE_RandomRead (0,0x10);
ucData1= XEE_ReadNext(0); //read a byte data from the next address of 0x10 (0x11)
```

[C#]

```
byte bData, bData1;
XWBoard.XEE_InitByName(_XW300_);
//read a byte data from address 0x10 block 0 on EEPROM
bData = XWBoard.XEE_RandomRead (0,0x10);
//read a byte data from the next address of 0x10 (0x11)
ucData1= XWBoard.XEE_ReadNext(0);
```

4.7. XEE_MultiRead

This function reads multiple bytes of data from the EEPROM on the XWboard.

Syntax

```
int XEE_MultiRead(
int StartBlock,
int StartAddr,
int no,
unsigned char *databuf
);
```

Parameters

StartBlock

[in] 0 to 7. (Total is 8 blocks)

StartAddr

[in] 0 to 255. (Each block is 256 bytes)

no

[in] 1 to 2048

**databuf*

[Out] The address where the data is to be stored

Return Values

0 (NoError) On success.
-9 (AddrError) StartAddr is invalid.
-10 (BlockError) StartBlock is invalid.

Examples

[eVC]

```
char usData[10];
XEE_InitByName(_XW300_);
//read 10 bytes of data to usData buffer from address 0 block 0 on EEPROM
XEE_MultiRead(0,0,10, usData);
```

[C#]

```
byte bData, bData1;
XWBoard.XEE_InitByName(_XW300_);
//read 10 bytes of data to usData buffer from address 0 block 0 on EEPROM
XWBoard.XEE_MultiRead(0,0,10, usData);
```

4.8. XEE_RandomWrite

Write one byte of data to the Flash memory.

Syntax

```
int XEE_ RandomWrite (
int Block,
int Addr,
int Data,
);
```

Parameters

Block

[in] 0 to 7. (Total is 8 blocks)

Addr

[in] 0 to 255. (Each block is 256 bytes)

Data

[in] 0 to 255

Return Values

0 (NoError) On success.
-9 (AddrError) StartAddr is invalid.
-10 (BlockError) StartBlock is invalid.

Examples

[eVC]

```
XEE_InitByName(_XW300_);
//write a byte data (0x10) to address 0 block 0 on EEPROM
XEE_ RandomWrite(0,0,0x10);
```

[C#]

```
XWBoard.XEE_InitByName(_XW300_);
//write a byte data (0x10) to address 0 block 0 on EEPROM
XWBoard. XEE_ RandomWrite (0,0x10);
```

4.9. XEE_MultiWrite

This function writes data to the EEPROM on the XW-board. The maximun size for multi-write is 16 bytes, and must be written to the same block.

Syntax

```
int XEE_ MultiWrite (
int Block,
int Addr,
int Num,
char *Data
);
```

Parameters

Block

[in] 0 to 7. (Total is 8 blocks)

Addr

[in] 0 to 255. (Each block is 256 bytes)

Num

[in] 1 to 16

Data

[Out] The starting address of the buffer where the data will be written into EEPROM

Return Values

0 (NoError) On success.
-1 The EEPROM is busy, the Block or Addr is invalid.

Examples

[eVC]

```
char usData[10]={0,1,2,3,4,5,6,7,8,9};
XEE_InitByName(_XW300_);
//read 10 bytes of data to address 0 block 0 on EEPROM
XEE_MultiWrite(0,0,10, usData);
```

[C#]

```
byte[] bValue = {0,1,2,3,4,5,6,7,8,9};
XWBoard. XWB_Init(0); // Initialize XWboard
//read 10 bytes of data to address 0 block 0 on EEPROM
bool ret = XWBoard. XEE_MultiWrite(0,0,10, bValue);
```