User Manual

Advantech APAX series

Software Manual

Document History:

Doc Version	Date	Comment
1.0	2015/02/03	First edition

Chapter 1

1. Introduction

1.1. About This Manual

This is the Software Manual for the Advantech APAX- 552X, APAX- 558X, APAX- 562X and APAX- 657X products.
Advantech provides APAX library which allows developers and end users to connect I/O modules, perform configurations, and simple testing of the I/O.

This manual supplies information about Advantech APAX I/O modules, including calling procedure of operating device and descriptions of each function, parameter and data structure.

This manual does not show you how to solve every possible programming problem. To use this manual, you should already be familiar with at least one of the supported programming environments and Windows 2000/XP/Vista/7/Embedded Standard/CE.

1.2. Organization of This Manual

This user manual is divided into the following sections:

- User functions
- Internal functions

User functions

This section provides information about user functions and how to use them.

Internal functions

This section provides information about internal functions.
It's highly recommended NOT to use these functions in your programs.

Functions Reference

- **Call flow**
  This section provides call flow and information about how to use functions.

- **Error Codes**
  This section describes system error codes.

- **Analog I/O Board Settings**
  This section provides range settings for Analog I/O Boards.

1.3. Installation

In order to save your development time, Advantech provides several examples that you can use it as reference to build your own C/C++, C# or VB application program. The default installation directory is **C:/Program Files (x86)/Advantech/AdamApax.NET Class Library** and all examples can be found in /**Sample Code** after installing AdamApax .NET Class Library from Advantech website at http://www.advantech.com in the download area under Support page. For Windows XP users, the examples are under “APAX/Win32”. For Windows CE users, the examples are under “APAX/WinCE”. The sample programs are all build with “Microsoft Visual Studio 2008” for Windows XP and Windows CE.

Chapter 2

2. User functions

2.1. ADAMDrvOpen

Users can use this function to open APAX device. The function returns a handle that can be used to access the APAX device.

Syntax

ADAMDrvOpen(
  LONG *handle
);

Parameters

Name	Direction	Description
handle	Output	[out] The driver handler.

Return Value

If driver initialization succeeded, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Remarks

Use the **ADAMDrvClose** function to close the ADAM/APAX device.

Example

```
LONG lDriverHandle = NULL; /* Driver handler */
if(ERR_SUCCESS != ADAMDrvOpen(&lDriverHandle))
    printf("Fail to open driver\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5013.cpp

2.2. ADAMDrvClose

Close the ADAM/APAX device by calling this function when operation is completed.

Syntax

ADAMDrvClose (

LONG *handle
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.

Return Value

If driver termination succeeded, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
if(NULL != lDriverHandle) {
    ADAMDrvClose(&lDriverHandle);
    lDriverHandle = NULL;
}
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5013.cpp

2.3. AI_GetBurnoutFunEnable

Users can use this function to get the burnout function status (enable/disable)

Syntax

AI_GetBurnoutFunEnable(
LONG handle,
WORD i_wSlot,
DWORD* o_dwEnableMask
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
o_dwEnableMask	Output	[out] The enable mask. ***Note:**

		Each bit indicates one channel. From LSB to MSB of the value indicate the channel-0 to the last channel enabled mask. If the bit is 1, it means that the channel is enabled.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)** and **o_dwEnableMask** contains burnout function status from channel-0 to the last channel. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
const int iChannelNum = 8; /* Set channel number */
WORD wSlotID = 1; /* Slot ID */
LONG lResult = AI_GetBurnoutFunEnable(lDriverHandle, wSlotID,
&dwEnableMask);
if (ERR_SUCCESS == lResult) {
    for (int iCnt = 0; iCnt < iChannelNum; iCnt++) {
        if (dwEnableMask & (0x01 << iCnt)) {
            printf("The channel %d is enabled.\n", iCnt);
            Sleep(100);
        }
        else
            printf("The channel %d is disabled.\n", iCnt);
    }
}
else
    printf("Fail to get burnout detect function\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5013.cpp

2.4. AI_GetBurnoutValue

Users can use this function to get the burnout value.

Syntax

AI_GetBurnoutValue(

```
  LONG handle,
  WORD i_wSlot,
  DWORD* o_dwValue
);
```

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
o_dwValue	Output	[out] The variable to hold the burnout value. ***Note:** The return value is 1 when the detect mode is in "Up Scale"; 0 means "Down Scale".

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)** and **o_dwValue** contains burnout value. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD dwBurnoutMode = 0; /* Burnout Mode */
LONG lResult = AI_GetBurnoutValue(lDriverHandle, wSlotID,
&dwBurnoutMode);
if (ERR_SUCCESS == lResult) {
    if (dwBurnoutMode)
        printf("The burnout detect mode is \" %s \".\n","Up Scale");
    else
        printf("The burnout detect mode is \" %s \".\n","Down Scale");
        Sleep(3000);
}
else
    printf("Fail to get burnout detect mode\n");
```

For more detailed information regarding this function, please see $(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5013.cpp

2.5. AI_GetChValOffset

Users can use this function to get the channel offset value of the indicated slot.

Syntax

AI_GetChValOffset(
  LONG handle,
  WORD i_wSlot,
  WORD i_wChannel,
  DWORD* o_dwOffset
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_wChannel	Input	[in] The channel ID which is ranged from 0 to 31 for normal value offset. ***Note:** Set this value to 0xFE indicates the value offset is for CJC offset.
o_dwValue	Output	[out] The variable to hold the offset value.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
WORD wChannel = 2; /* Channel ID */
DWORD dwValue = 0; /* Channel offset value */

LONG lResult = AI_GetChValOffset (lDriverHandle, wSlotID, wChannel
&dwValue);
if (ERR_SUCCESS == lResult) {
    printf("The offset value is %ld\n", dwValue);
    Sleep(3000);
```

```
}
else
    printf("Fail to get offset value\n");
```

2.6. AI_GetCjcValue

Users can use this function to get the CJC value of the indicated slot.

Syntax

AI_GetCjcValue(
  LONG handle,
  WORD i_wSlot,
  DWORD* o_dwValue,
  BYTE* o_byStatus
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
o_dwValue	Output	[out] The variable to hold the CJC values.
o_byStatus	Output	[out] CJC setting status.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**; **o_dwValue** and **o_byStatus** contain CJC values and setting status. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD dwValue = 0; /* Channel offset value */
BYTE byStatus = 0;
LONG lResult = AI_GetCjcValue (lDriverHandle, wSlotID, &dwValue,
&byStatus);
if (ERR_SUCCESS == lResult) {
    printf("The CJC value is %ld and status is %d\n", dwValue, byStatus);
    Sleep(3000);
```

```
}
else
    printf("Fail to get CJC value\n");
```

2.7. AI_GetSampleRate

Users can use this function to get the sample rate.

Syntax

AI_GetSampleRate(
  LONG handle,
  WORD i_wSlot,
  DWORD* o_dwSampleRate
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
o_dwSampleRate	Output	[out] The variable to hold the sample rate (Hz).

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)** and **o_dwSampleRate** contains sample rate of the indicated slot. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD dwSampleRate = 0;
LONG lResult = AI_GetSampleRate(lDriverHandle, wSlotID, &dwSampleRate);
if (ERR_SUCCESS == lResult) {
    printf("The sampling rate is %d (Hz/Ch).\n",dwSampleRate);
    Sleep(3000);
}
else
    printf("Fail to get sampling rate\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5017.cpp

2.8. AI_SetAutoCalibration

Users can use this function to set to run auto-calibration of the indicated slot.

Syntax

AI_SetAutoCalibration(
  LONG handle,
  WORD i_wSlot
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
if(ERR_SUCCESS != AI_SetAutoCalibration (lDriverHandle, wSlotID))
    printf("Fail to set auto-calibration\n");
```

2.9. AI_SetBurnoutFunEnable

Users can use this function to get the burnout function status (enable/disable)

Syntax

AI_GetBurnoutFunEnable(
  LONG handle,
  WORD i_wSlot,
  DWORD i_dwEnableMask
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_dwEnableMask	Input	[in] The enable mask to be set. ***Note:** Each bit indicates one channel. From LSB to MSB of the value indicate the channel-0 to the last channel enabled mask. If the bit is 1, it means that the channel is enabled.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD dwEnableMask = 0xFFFF; /* Enable all channels */
LONG lResult = AI_SetBurnoutFunEnable(lDriverHandle, wSlotID,
dwEnableMask);
if (ERR_SUCCESS == lResult) {
        printf("Succeed to set burnout function status.\n");
        Sleep(3000);
}
else
    printf("Fail to set burnout function status \n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5013.cpp

2.10. AI_SetBurnoutValue

Users can use this function to set the burnout value.

Syntax

AI_SetBurnoutValue(

```
  LONG handle,
  WORD i_wSlot,
  DWORD    i_dwValue
);
```

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_dwValue	Input	[in] The burnout value to be set. ***Note:** Set value to 1 for “Up Scale” mode; 0 for “Down Scale” mode.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD dwBurnoutMode = 0xFFFF; /* Up scale mode*/
LONG lResult = AI_SetBurnoutValue(lDriverHandle, wSlotID,
dwBurnoutMode);
if (ERR_SUCCESS == lResult) {
    printf("Succeed to set burnout value.\n");
    Sleep(3000);
}
else
    printf("Fail to set burnout value\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5013.cpp

2.11. AI_SetChannelMask

Users can use this function to set channel mask of the indicated slot.

Syntax

AI_SetChannelMask(
  LONG handle,
  WORD i_wSlot,
  DWORD i_dwMask
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_dwMask	Input	[in] The enabled AI channel mask to be set. ***Note:** From LSB to MSB of the value indicate the channel-0 to channel-31 enabled mask. If the bit is 1, it means that the channel is enabled.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD dwMask = 0x0000FFFF; /* Enable all channels */
LONG lResult = AI_SetChannelMask(lDriverHandle, wSlotID, dwMask);
if (ERR_SUCCESS == lResult) {
    printf("Succeed to set channel mask.\n");
    Sleep(1000);
}
else
    printf("Fail to set channel mask\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5013.cpp

2.12. AI_SetChValOffset

Users can use this function to set the channel value offset of the indicated slot.

Syntax

AI_SetChValOffset(
  LONG handle,
  WORD i_wSlot,
  WORD i_wChannel,
  DWORD i_dwOffset
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_wChannel	Input	[in] The channel ID which is ranged from 0 to 31 for normal value offset. ***Note:** Set this value to 0xFE indicates the value offset is for CJC offset.
i_dwValue	Input	[in] The offset value to be set.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
WORD wChannel = 2; /* Channel ID */
DWORD dwValue = 0; /* Channel offset value */

if(ERR_SUCCESS != AI_GetChValOffset (lDriverHandle, wSlotID, wChannel
dwValue))
    printf("Fail to set channel value offset.\n");
```

2.13. AI_SetCjcInitValRecord

Users can use this function to set to record the CJC initial value.

Syntax

AI_SetCjcInitValRecord(
  LONG handle,
  WORD i_wSlot
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
if(ERR_SUCCESS != AI_SetCjcInitValRecord (lDriverHandle, wSlotID))
    printf("Fail to record the CJC initial value\n");
```

2.14. AI_SetIntegrationTime

Users can use this function to set AI integration time of the indicated slot.

Syntax

AI_SetIntegrationTime(
  LONG handle,
  WORD i_wSlot,
  DWORD i_dwIntegration
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_dwIntegration	Input	[in] The AI integration time to be set (Hz).

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD dwIntegration = 50; /* Hz */
if(ERR_SUCCESS != AI_SetIntegrationTime (lDriverHandle, wSlotID,
dwIntegration))
    printf("Fail to set integration time\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5013.cpp

2.15. AI_SetSampleRate

Users can use this function to set the sample rate.

Syntax

AI_SetSampleRate(
  LONG handle,
  WORD i_wSlot,
  DWORD i_dwSampleRate
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_dwSampleRate	Input	[in] The sample rate (Hz) to be set.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD dwSampleRate = 10; /* Hz */
if (ERR_SUCCESS != AI_SetSampleRate(lDriverHandle, wSlotID,
dwSampleRate) {
    printf("Fail to set sampling rate\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5017.cpp

2.16. AI_SetThermoCalibration

Users can use this function to set to calibrate the thermocouple.

Syntax

AI_SetThermoCalibration(
  LONG handle,
  WORD i_wSlot
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
if(ERR_SUCCESS != AI_SetThermoCalibration (lDriverHandle, wSlotID))
    printf("Fail to set thermo-calibration\n");
```

2.17. AIO_GetChannelStatus

Users can use this function to get all channel status of the indicated slot.

Syntax

AIO_GetChannelStatus(
  LONG handle,
  WORD i_wSlot,
  BYTE* o_byStatus
);

Parameters

<table>
<tr><th>Name</th><th>Direction</th><th>Description</th></tr>
<tr><td>handle</td><td>Input</td><td>[in] The driver handler.</td></tr>
<tr><td>i_wSlot</td><td>Input</td><td>[in] The slot ID which is ranged from 0 to 15.</td></tr>
<tr><td>o_byStatus</td><td>Output</td><td>[out] The variables array to hold the channel status. The size of this array must be at least 32 BYTEs.
Note:
The value of o_byStatus indicates:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>None</td></tr>
<tr><td>1</td><td>Normal</td></tr>
<tr><td>2</td><td>Over current</td></tr>
<tr><td>3</td><td>Under current</td></tr>
<tr><td>4</td><td>Burn out</td></tr>
<tr><td>5</td><td>Open loop</td></tr>
<tr><td>6</td><td>Not ready</td></tr>
</table></td></tr>
</table>

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)** and **o_byStatus** hold the channel status. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
const int iChannelNum = 8; /* Set channel number */
WORD wSlotID = 1; /* Slot ID */
BYTE bChStatus [32] ={0};
LONG lResult = AIO_GetChannelStatus(lDriverHandle, wSlotID, bChStatus);
if (ERR_SUCCESS == lResult) {
    printf("Succeed to get channel status.\n");
```

```
    for(int iCnt = 0; iCnt < iChannelNum ; iCnt++)
    {
        switch(bChStatus[iCnt])
        {
            case 0:
                printf("[Ch %d] \"None\"\n",iCnt);
                break;
            case 1:
                printf("[Ch %d] \"Normal\"\n",iCnt);
                break;
            case 2:
                printf("[Ch %d] \"Over Current\"\n",iCnt);
                break;
        case 3:
            printf("[Ch %d] \"Under Current\"\n",iCnt);
            break;
        case 4:
            printf("[Ch %d] \"Burn Out\"\n",iCnt);
            break;
        case 5:
            printf("[Ch %d] \"Open Loop\"\n",iCnt);
            break;
        case 6:
            printf("[Ch %d] \"Not Ready\"\n",iCnt);
            break;
        default:
            printf("[Ch %d] \"Unknown\"\n",iCnt);
            break;
        }
    }
    Sleep(100);
}
else
    printf("Fail to get channel status\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5013.cpp

2.18. AIO_GetValue

Users can use this function to get the single AI/AO value from the indicated slot and channel.

Syntax

AIO_GetValue(
  LONG handle,
  WORD i_wSlot,
  WORD i_wChannel,
  WORD* o_wValue
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_wChannel	Input	[in] The channel ID which is ranged from 0 to 31 for normal value offset.
o_wValue	Output	[out] The variable to hold the returned AI/AO value.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)** and **o_wValue** hold the returned AI/AO value. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
WORD wChannel = 7; /* Channel ID */
WORD wValue = 0;
LONG lResult = AIO_GetValue(lDriverHandle, wSlotID, wChannel, &wValue);
if (ERR_SUCCESS == lResult) {
    printf("Channel %d raw data is 0x%04X\n", wChannel, wValue);
    /* Note: Need to scale raw data according to the channel range type*/

    Sleep(3000);
}
```

```
else
    printf("Fail to get value\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5013.cpp

2.19. AIO_GetValues

Users can use this function to get all AI/AO value of the indicated slot.

Syntax

AIO_GetValues(
  LONG handle,
  WORD i_wSlot,
  WORD* o_wValues
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
o_wValues	Output	[out] The variable to hold the returned AI/AO value. ***Note:** The size of this array must be at least 32 WORDs.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)** and **o_wValues** contains AIO values from channel-0 to the last channel. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
const int iChannelNum = 8; /* Set channel number */
WORD wSlotID = 1; /* Slot ID */
WORD wValue[iChannelNum] = {0};
LONG lResult = AIO_GetValues(lDriverHandle, wSlotID, wRawValue);

if (ERR_SUCCESS == lResult) {
```

```
    for (int iCnt =0; iCnt < iChannelNum ; iCnt++) {
        printf("[Ch %d] raw data is 0x%04X\n", iCnt, wValue[iCnt]);
    /* Note: Need to scale raw data according to the channel range type*/
    }
else
    printf("Fail to get values\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5013.cpp

2.20. AIO_SetRanges

Users can use this function to set the channel ranges of the indicated slot.

Syntax

AIO_SetRanges(
  LONG handle,
  WORD i_wSlot,
  WORD i_wChannelTotal,
  WORD* i_wRanges
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_wChannelTotal	Input	[in] The channel total of the module in the indicated slot.
i_wRanges	Input	[in] The ranges to be set. ***Note:** See **APPENDIX C** for valid range settings. The size of this array must be i_wChannelTotal WORDs.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
const int iChannelNum = 8; /* Set channel number */
WORD wSlotID = 1; /* Slot ID */
WORD wChRange[iChannelNum]={0};
wChRange[0] = 0x200; /* PT100_3851_NEG_200_TO_850_C */
wChRange[1] = 0x201; /* PT100_3851_NEG_120_TO_130_C */
wChRange[2] = 0x220; /* PT200_3851_NEG_200_TO_850_C */
wChRange[3] = 0x221; /* PT200_3851_NEG_120_TO_130_C */
wChRange[4] = 0x240; /* PT500_3851_NEG_200_TO_850_C */
wChRange[5] = 0x241; /* PT500_3851_NEG_120_TO_130_C */
wChRange[6] = 0x260; /* PT1000_3851_NEG_200_TO_850_C */
wChRange[7] = 0x261; /* PT1000_3851_NEG_120_TO_130_C */

if(ERR_SUCCESS != AIO_SetRanges(lDriverHandle, wSlotID, iChannelNum,
wChRange))
    printf("Fail to set ranges \n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5013.cpp

2.21. AIO_SetSpanCalibration

Users can use this function to run the span calibraion of the indicated slot and channel.

Syntax

AIO_SetSpanCalibration(
  LONG handle,
  WORD i_wSlot,
  WORD i_wChannel,
  WORD i_wType
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_wChannel	Input	[in] The channel ID which is ranged from 0 to 31 for normal value offset.

i_wType	Input	[in] The type value to be set. Currently, it is ignored.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
WORD wChannel = 7; /* Channel ID */
if(ERR_SUCCESS != AIO_SetSpanCalibration (lDriverHandle, wSlotID,
wChannel, 0))
    printf("Fail to set span-calibration \n");
```

2.22. AIO_SetZeroCalibration

Users can use this function to run the zero calibraion of the indicated slot and channel.

Syntax

AIO_SetZeroCalibration(
  LONG handle,
  WORD i_wSlot,
  WORD i_wChannel,
  WORD i_wType
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_wChannel	Input	[in] The channel ID which is ranged from 0 to 31 for normal value offset.
i_wType	Input	[in] The type value to be set. Currently, it is ignored.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information,

call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
WORD wChannel = 7; /* Channel ID */
if(ERR_SUCCESS != AIO_SetZeroCalibration (lDriverHandle, wSlotID,
wChannel, 0))
    printf("Fail to set zero-calibration \n");
```

2.23. AO_BufValues

Users can use this function to buffer the AO values of the indicated slot. This function is used along with OUT_FlushBufValues for a synchronized write Output. Once all slots are buffered, then OUT_FlushBufValues function triggers the synchronized buffer write of all masked slots.

Syntax

AO_BufValues(
  LONG handle,
  WORD i_wSlot,
  DWORD i_dwMask,
  WORD* i_wValues
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_dwMask	Input	[in] The channels mask. ***Note:** From LSB to MSB of the value indicate the slot-0 to slot-31 mask. If the bit is 1, it means that the channel must buffer value.
i_wValues	Input	[in] The AO values to be bufferred.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information,

call **GetLastError** function.

Example

```
const int iChannelNum = 8; /* Set channel number */
WORD wSlotID = 1; /* Slot ID */
WORD wValue[iChannelNum]={0};
wValue[0] = 0x0011;
DWORD dwMask = 0xFFFF;
if(ERR_SUCCESS != AO_BufValues(DriverHandle, wSlotID, dwMask, wValue))
    printf("Fail to buffer values\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5028.cpp

2.24. AO_GetSaftyValues

Users can use this function to get the all AO safety values of the indicated slot.

Syntax

AO_GetSaftyValues(
  LONG handle,
  WORD i_wSlot,
  WORD i_wChannelTotal,
  WORD* o_wValues
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_wChannelTotal	Input	[in] The channel total of the module in the indicated slot.
o_wValues	Output	[out] The variables array to hold the AIO safety values

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)** and **o_wValues** contains AIO values from channel-0 to the last channel. If the function fails, the

return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
const int iChannelNum = 8; /* Set channel number */
WORD wSlotID = 1; /* Slot ID */
WORD wSafetyValues [iChannelNum] = {0};
LONG lResult = AO_GetSaftyValues(lDriverHandle, wSlotID, iChannelNum,
wSafetyValues);
if (ERR_SUCCESS == lResult) {
    for (int iCnt = 0; iCnt < iChannelNum; iCnt++) {
        printf("The raw safety value is %d.\n", wSafetyValues [iCnt]);
    /* Note: Need to scale raw data according to the channel range type*/
        Sleep(1000);
    }
else
    printf("Fail to get safety values \n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5028.cpp

2.25. AO_GetStartupValues

Users can use this function to get the AO startup values of the indicated slot.

Syntax

```
AO_GetStartupValues(
  LONG handle,
  WORD i_wSlot,
  WORD i_wChannelTotal,
  WORD* o_wValues
);
```

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_wChannelTotal	Input	[in] The channel total of the module in the indicated slot.

o_wValues	Output	[out] The variables array to hold the AO startup values. ***Note:** The size of this array must be at least 32 WORDs.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)** and **o_wValues** contains AO startup values from channel-0 to the last channel. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
const int iChannelNum = 8; /* Set channel number */
WORD wSlotID = 1; /* Slot ID */
WORD wStartUpValues [iChannelNum] = {0};
LONG lResult =  AO_GetStartupValues (lDriverHandle, wSlotID,
iChannelNum, wStartUpValues);
if (ERR_SUCCESS == lResult) {
    for (int iCnt = 0; iCnt < iChannelNum; iCnt++) {
        printf("The raw start up value is %d.\n", wSafetyValues [iCnt]);
    /* Note: Need to scale raw data according to the channel range type*/
        Sleep(1000);
    }
else
    printf("Fail to get startup value \n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5028.cpp

2.26. AO_SetCalibrationMode

Users can use this function to switch to AO calibration mode of the indicated slot.

Syntax

AO_SetCalibrationMode (
  LONG handle,
  WORD i_wSlot,
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
if(ERR_SUCCESS != AO_SetCalibrationMode (lDriverHandle, wSlotID))
    printf("Fail to switch to AO calibration mode\n");
```

2.27. AO_SetSaftyValues

Users can use this function to set the all AO safety values of the indicated slot.

Syntax

AO_GetSaftyValues(
  LONG handle,
  WORD i_wSlot,
  WORD i_wChannelTotal,
  WORD* i_wValues
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_wChannelTotal	Input	[in] The channel total of the module in the indicated slot.
o_wValues	Input	[in] The AO safety values to be set.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails,

the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
const int iChannelNum = 8; /* Set channel number */
WORD wSlotID = 1; /* Slot ID */
WORD wSafteyValue[iChannelNum] = {0};
wSafteyValue[0] = 0xFFFF;
wSafteyValue[1] = 0xFFFF;
if(ERR_SUCCESS != AO_SetSaftyValues(lDriverHandle, wSlotID, iChannelNum,
wSafteyValue))
    printf("Fail to set saftey values \n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5028.cpp

2.28. AO_SetStartupValues

Users can use this function to set the AO startup values of the indicated slot.

Syntax

```
AO_SetStartupValues(
  LONG handle,
  WORD i_wSlot,
  WORD i_wChannelTotal,
  WORD* i_wValues
);
```

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_wChannelTotal	Input	[in] The channel total of the module in the indicated slot.
i_wValues	Input	[in] The values array to be set. ***Note:** The size of this array must be i_wChannelTotal WORDs.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
const int iChannelNum = 8; /* Set channel number */
WORD wSlotID = 1; /* Slot ID */
WORD wStartValue[iChannelNum] = {0};
wStartValue [0] = 0x0011;
wStartValue [1] = 0x1100;
if(ERR_SUCCESS != AO_SetStartupValues (lDriverHandle, wSlotID,
iChannelNum, wSafteyValue))
    printf("Fail to set startup values\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5028.cpp

2.29. CNT_ClearAlarmFlags

Users can use this function to clear the counter alarm of the indicated slot.

Syntax

CNT_ClearAlarmFlags(
  LONG handle,
  WORD i_wSlot,
  DWORD i_dwMask
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_dwMask	Input	[in] The alarm flag mask.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails,

the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD dwMask = 0xFFFF;
if(ERR_SUCCESS != CNT_ClearAlarmFlags (lDriverHandle, wSlotID, dwMask))
    printf("Fail to clear counter alarm \n");
```

2.30. CNT_ClearOverflows

Users can use this function to clear the counter overflow of the indicated slot.

Syntax

CNT_ClearOverflows(
  LONG handle,
  WORD i_wSlot,
  DWORD i_dwMask
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_dwMask	Input	[in] The counter channel mask to be set. ***Note:** From LSB to MSB of the value indicate the channel-0 to the last channel mask. If the bit value is 1, it means that the overflow of the channel must be cleared.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
```

```
DWORD dwMask = 0xFFFF;
if(ERR_SUCCESS != CNT_ClearOverflows (lDriverHandle, wSlotID, dwMask))
    printf("Fail to clear overflows\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5080.cpp

2.31. CNT_ClearValues

Users can use this function to clear the masked counter values to startup values of the indicated slot.

Syntax

CNT_ClearValues(
  LONG handle,
  WORD i_wSlot,
  DWORD i_dwMask
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_dwMask	Input	[in] The counter channel mask to be set. ***Note:** From LSB to MSB of the value indicate the channel-0 to the last channel mask. If the bit value is 1, it means that the channel must set to startup value.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD dwMask = 0xFFFF;
if(ERR_SUCCESS != CNT_ClearValues (lDriverHandle, wSlotID, dwMask))
```

```
printf("Fail to clear counter values \n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5080.cpp

2.32. CNT_GetAlarmConfig

Users can use this function to get the counter alarm configuration of the indicated slot.

Syntax

```
CNT_GetAlarmConfig(
  LONG handle,
  WORD i_wSlot,
  WORD i_wAlarmIndex,
  BOOL* o_bEnable,
  BOOL* o_bAutoReload,
  BYTE* o_byType,
  BYTE* o_byMapChannel,
  DWORD* o_dwLimit,
  BYTE* o_byDoType,
  DWORD* o_dwDoPulseWidth
);
```

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_wAlarmIndex	Input	[in] The alarm index
o_bEnable	Output	[out] The variable to hold the alarm enabled status.
o_bAutoReload	Output	[out] The variable to hold the alarm auto reload status.
o_byType	Output	[out] The variable to hold the alarm type. ***Note:** Set value to 1 for “High alarm”; 0 for “Low alarm”.
o_byMapChannel	Output	[out] The variable to hold the counter channel.
o_dwLimit	Output	[out] The variable to hold the counter limit.

o_byDoType	Output	[out] The variable to hold the DO type. ***Note:** The value of o_byDoType indicates: Value — Meaning 0 — Low level 1 — High level 2 — Low pulse 3 — High pulse
o_dwDoPulseWidth	Output	[out] The variable to hold the DO pulse width.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
WORD wChannel = 1;
BOOL bEnabled = false; /* Enabled */
BOOL bAutoReload = false; /* Auto reload */
BYTE byAlarmType = 0; /* Low alarm */
BYTE byMapChannel = 0;
DWORD dwLimitValue = 0;
BYTE byDoType = 0;
DWORD dwDoPulseWidth = 0; /* DO pulse width to be set (ms) */
CHAR *szEnabledStatus[] = {"false","true"};
CHAR *szAlarmType[] = {"low alarm","high alarm"};
CHAR *szDoType[] = {"low level","high level","low pulse","high pulse"};
LONG lResult = CNT_GetAlarmConfig(lDriverHandle, wSlotID, wChannel,
&bEnabled,&bAutoReload,&byAlarmType,&byMapChannel,&dwLimitValue,&byDo
Type,&dwDoPulseWidth);
if (ERR_SUCCESS == lResult) {
    printf("Succeed to get CNT alarm configuration.\n");
    printf("Alarm enabled status is %s.\n",szEnabledStatus[bEnabled]);
    printf("Auto reload status is %s.\n",szEnabledStatus[bAutoReload]);
    printf("Alarm type is %s.\n",szAlarmType[byAlarmType]);
    printf("The CNT channel %d is mapped.\n",byMapChannel);
    printf("The counter limit value is %d.\n",dwLimitValue);
```

```
    printf("The DO type is %s.\n",szDoType[byDoType]);
    printf("The DO pulse width is %d(ms).\n",dwDoPulseWidth);
    Sleep(1000);
}
else
    printf("Fail to get CNT alarm configuration\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5080.cpp

2.33. CNT_GetAlarmFlags

Users can use this function to get the counter alarm of the indicated slot.

Syntax

CNT_GetAlarmFlags(
  LONG handle,
  WORD i_wSlot,
  DWORD* o_dwFlags
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
o_dwFlags	Output	[out] The variable to hold the counter alarm flags.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD wFlag = 0;
if(ERR_SUCCESS != CNT_GetAlarmFlags (lDriverHandle, wSlotID, &wFlag))
    printf("Fail to get alarm flags\n");
```

2.34. CNT_GetChannelMask

Users can use this function to get enabled counter channel mask of the indicated slot.

Syntax

CNT_GetChannelMask(
  LONG handle,
  WORD i_wSlot,
  DWORD* o_dwMask
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
o_dwMask	Output	[out] The variable to hold the enabled counter channel mask. ***Note:** From LSB to MSB of the value indicate the channel-0 to the last channel mask. If the bit value is 1, it means that the channel is enabled.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD wMask = 0;
if(ERR_SUCCESS != CNT_GetChannelMask (lDriverHandle, wSlotID, &wMask))
    printf("Fail to get channel mask\n");
```

2.35. CNT_GetChannelStatus

Users can use this function to get all channels status of the indicated slot.

Syntax

CNT_GetChannelStatus(
  LONG handle,
  WORD i_wSlot,
  BYTE* o_byStatus
);

Parameters

<table>
<tr><th>Name</th><th>Direction</th><th>Description</th></tr>
<tr><td>handle</td><td>Input</td><td>[in] The driver handler.</td></tr>
<tr><td>i_wSlot</td><td>Input</td><td>[in] The slot ID which is ranged from 0 to 15.</td></tr>
<tr><td>o_byStatus</td><td>Output</td><td>[out] The variables array to hold the channels status.
*Note:
The size of this array must be at least 8 BYTEs.
The value of o_byDoType indicates:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Normal</td></tr>
<tr><td>8</td><td>Over flow</td></tr>
<tr><td>9</td><td>Under flow</td></tr>
<tr><td>10</td><td>Over and under flow</td></tr>
</table></td></tr>
</table>

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
const int iChannelCNTNum = 8;
BYTE bChStatus [32] = {0};
LONG lResult = CNT_GetChannelStatus(lDriverHandle, wSlotID, bChStatus);
if (ERR_SUCCESS == lResult) {
    printf("Succeed to get CNT channel status.\n");
    for(int iCnt = 0; iCnt < iChannelCNTNum ; iCnt++) {
        switch(bChStatus[iCnt]) {
            case 0:
                printf("Ch %d status: \"None.\"\n",iCnt);
```

```
                break;
            case 1:
                printf("Ch %d status: \"Normal.\"\n",iCnt);
                break;
            case 8:
                printf("Ch %d status: \"Over flow.\"\n",iCnt);
                break;
            case 9:
                printf("Ch %d status: \"Under flow.\"\n",iCnt);
                break;
            case 10:
                printf("Ch %d status: \"Over and Under flow.\"\n",iCnt);
                break;
            default:
                printf("Ch %d status is \"Unknown.\"\n",iCnt);
                    break;
            }
        }
        Sleep(100);
    }
    else
            printf("Fail to get CNT channel status\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5080.cpp

2.36. CNT_GetCntTypeConfig

Users can use this function to get the counter counting type of the indicated slot.

Syntax

```
CNT_GetCntTypeConfig(
  LONG handle,
  WORD i_wSlot,
  WORD i_wChannel,
  BOOL* o_bRepeat,
  BOOL* o_bReload
);
```

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_wChannel	Input	[in] The channel index.
o_bRepeat	Output	[out] The variable to hold the repeat enabled status.
o_bReload	Output	[out] The variable to hold the startup enabled status.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
const int iChannelCNTNum = 8;
WORD wChannel = 2; /* Channel ID */
BOOL byRepeat = 0; /* Repeat enabled status */
BOOL byReload = 0; /* Startup enabled status */
CHAR *szEnabledStatus[] = {"false","true"};
LONG lResult = CNT_GetCntTypeConfig(lDriverHandle, wSlotID, wChannel,
&byRepeat, &byReload);
if (ERR_SUCCESS == lResult) {
    printf("Repeat enabled status %s.\n",szEnabledStatus[byRepeat]);
    printf("Reload enabled status is %s.\n",szEnabledStatus[byReload]);
    Sleep(1000);
}
else
    printf("Fail to get CNT type configuration.\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5080.cpp

2.37. CNT_GetFilter

Users can use this function to get the counter filter width of the indicated slot.

Syntax

CNT_GetFilter(
  LONG handle,
  WORD i_wSlot,
  DWORD* o_dwWidth
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
o_dwWidth	Output	[out] The variable to hold the counter digital filter width.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD dwWidth = 0; /* filter width */
LONG lResult = CNT_GetFilter(lDriverHandle, wSlotID, &dwWidth);
if (ERR_SUCCESS == lResult) {
    printf("The CNT filter width is %d (us).\n", dwWidth);
        Sleep(1000);
    }
    else
        printf("Fail to get CNT filter width\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5080.cpp

2.38. CNT_GetGateConfig

Users can use this function to get the counter gate configuration of the indicated slot.

Syntax

```
CNT_GetGateConfig(
  LONG handle,
  WORD i_wSlot,
  WORD i_wChannel,
  BOOL* o_bEnable,
  BYTE* o_byTriggerMode,
  BYTE* o_byGateActiveType,
  BYTE* o_byMapGate
);
```

Parameters

<table>
<tr><th>Name</th><th>Direction</th><th>Description</th></tr>
<tr><td>handle</td><td>Input</td><td>[in] The driver handler.</td></tr>
<tr><td>i_wSlot</td><td>Input</td><td>[in] The slot ID which is ranged from 0 to 15.</td></tr>
<tr><td>i_wChannel</td><td>Input</td><td>[in] The channel index.</td></tr>
<tr><td>o_bEnable</td><td>Output</td><td>[out] The variable to hold the gate enabled status.</td></tr>
<tr><td>o_byTriggerMode</td><td>Output</td><td>[out] The variable to hold the trigger mode.
*Note:
The value of o_byTriggerMode indicates:
<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Non re-trigger</td></tr><tr><td>1</td><td>Re-trigger</td></tr><tr><td>2</td><td>Edge start</td></tr></table></td></tr>
<tr><td>o_byGateActiveType</td><td>Output</td><td>[out] The variable to hold the gate active type.
*Note:
The value of o_byGateActiveType indicates:
<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>0</td><td>Low level</td></tr><tr><td>1</td><td>Falling edge</td></tr><tr><td>2</td><td>High level</td></tr><tr><td>3</td><td>Rising edge</td></tr></table></td></tr>
<tr><td>o_byMapGate</td><td>Output</td><td>[out] The variable to hold the map gate.</td></tr>
</table>

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information,

call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
WORD wChannel = 2; /* Channel ID */
BOOL byGateEnabled = false; /* Gate enabled status */
BYTE byTrig = 0; /* Trigger mode */
BYTE byActive = 0; /* Gate active type */
BYTE byMapGate = 0; /* Map gate */
CHAR *szEnabledStatus[] = {"false","true"};
CHAR *szTriggerMode[] = {"non re-trigger","re-trigger","edge start"};
CHAR *szActiveType[] = {"low level","falling edge","high level","rising
edge"};

LONG lResult =
CNT_GetGateConfig(lDriverHandle,wSlotID,wChannel,&byGateEnabled,&byTr
ig,&byActive,&byMapGate);

if (ERR_SUCCESS == lResult) {
    printf("Succeed to get CNT gate configuration.\n");
    printf("Enabled status : %s.\n",szEnabledStatus[byGateEnabled]);
    printf("The trigger mode is %s.\n",szTriggerMode[byTrig]);
    printf("Active type : %s.\n",szActiveType[byActive]);
    printf("The CNT channel %d is mapped.\n",byMapGate);
    Sleep(1000);
}
else
        printf("Fail to get CNT gate configuration.\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5080.cpp

2.39. CNT_GetStartupValues

Users can use this function to get the counter startup values of the indicated slot.

Syntax

CNT_GetStartupValues(

```
  LONG handle,
  WORD i_wSlot,
  WORD i_wChannelTotal,
  DWORD* o_dwValues
);
```

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_wChannelTotal	Input	[in] The channel total of the module in the indicated slot. ***Note:** The maximum value is 8.
o_dwValues	Output	[out] the variables array to hold the counter startup values. ***Note:** The size of this array must be at least 8 DWORDs.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
const int iChannelCNTNum = 8;
WORD wSlotID = 1; /* Slot ID */
DWORD dStartValues[iChannelCNTNum] = {0};
LONG lResult = CNT_GetStartupValues(lDriverHandle, wSlotID,
iChannelCNTNum, dStartValues);
if (ERR_SUCCESS == lResult)
{
    printf("Succeed to get CNT startup values.\n");
    for (int iCnt = 0; iCnt < iChannelCNTNum; iCnt++) {
        printf("Startup value is %d.\n", dStartValues[iCnt]);
        Sleep(1000);
    }
}
```

```
else
    printf("Fail to get CNT startup values\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5080.cpp

2.40. CNT_GetValue

Users can use this function to get the counter value of the indicated slot and channel.

Syntax

CNT_GetValue (
  LONG handle,
  WORD i_wSlot,
  WORD i_wChannel,
  DWORD * o_dwValue
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_wChannel	Input	[in] The channel ID which is ranged from 0 to 7.
o_dwValue	Output	[out] The variable to hold the counter value.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)** and **o_dwValue** contains counter value. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wChannel = 2; /* Channel ID */
WORD wSlotID = 1; /* Slot ID */
DWORD dwValue = 0;
LONG lResult = CNT_GetValue (lDriverHandle, wSlotID, wChannel, &
dwValue);
if (ERR_SUCCESS == lResult) {
```

```
    printf("The value of CNT channel %d is %d.\n",wChannel, dwValue);
}
else
    printf("Fail to get CNT channel value \n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5080.cpp

2.41. CNT_GetValues

Users can use this function to get the all counter values of the indicated slot.

Syntax

CNT_GetValues (
  LONG handle,
  WORD i_wSlot,
  DWORD * o_dwValues
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
o_dwValues	Output	[out] The variable to hold the counter value. ***Note:** The size of this array must be at least 8 DWORDs.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)** and **o_dwValues** contains counter values from channel-0 to the last channel. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
const int iChannelCNTNum = 8;
WORD wSlotID = 1; /* Slot ID */
DWORD dValues[iChannelCNTNum] = {0};
LONG lResult = CNT_GetValues(lDriverHandle, wSlotID, dValues);
```

```
if (ERR_SUCCESS == lResult) {
    for (int iCnt = 0; iCnt < iChannelCNTNum; iCnt++) {
        printf("The value is %d.\n", dValues[iCnt]);
        Sleep(1000);
    }
}
else
    printf("Fail to get CNT channel value\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5080.cpp

2.42. CNT_SetAlarmConfig

Users can use this function to set the counter alarm configuration of the indicated slot.

Syntax

CNT_SetAlarmConfig(
  LONG handle,
  WORD i_wSlot,
  WORD i_wAlarmIndex,
  BOOL* i_bEnable,
  BOOL* i_bAutoReload,
  BYTE* i_byType,
  BYTE* i_byMapChannel,
  DWORD* i_dwLimit,
  BYTE* i_byDoType,
  DWORD* i_dwDoPulseWidth
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_wAlarmIndex	Input	[in] The alarm index
o_bEnable	Input	[in] The variable to hold the alarm enabled status.
o_bAutoReload	Input	[in] The variable to hold the alarm auto reload

<table>
<tr><td></td><td></td><td>status.</td></tr>
<tr><td>o_byType</td><td>Input</td><td>[in] The variable to hold the alarm type.
Note:
Set value to 1 for “High alarm”; 0 for “Low alarm”.</td></tr>
<tr><td>o_byMapChannel</td><td>Input</td><td>[in] The variable to hold the counter channel.</td></tr>
<tr><td>o_dwLimit</td><td>Input</td><td>[in] The variable to hold the counter limit.</td></tr>
<tr><td>o_byDoType</td><td>Input</td><td>[in] The variable to hold the DO type.
Note:
The value of o_byDoType indicates:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>Low level</td></tr>
<tr><td>1</td><td>High level</td></tr>
<tr><td>2</td><td>Low pulse</td></tr>
<tr><td>3</td><td>High pulse</td></tr>
</table></td></tr>
<tr><td>o_dwDoPulseWidth</td><td>Output</td><td>[in] The variable to hold the DO pulse width.</td></tr>
</table>

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
WORD wChannel = 1;
BOOL bEnabled = true; /* Enabled */
BOOL bAutoReload = true; /* Auto reload */
BYTE byAlarmType = 0; /* Low alarm */
BYTE byMapChannel = 0;
DWORD dwLimitValue = 100;
BYTE byDoType = 2; /* Low pulse */
DWORD dwDoPulseWidth = 500; /* DO pulse width to be set (ms) */
CHAR *szEnabledStatus[] = {"false","true"};
CHAR *szAlarmType[] = {"low alarm","high alarm"};
CHAR *szDoType[] = {"low level","high level","low pulse","high pulse"};
if (ERR_SUCCESS != CNT_SetAlarmConfig(lDriverHandle, wSlotID, wChannel,
bEnabled,bAutoReload,byAlarmType,byMapChannel,dwLimitValue,byDoType,
dwDoPulseWidth))
    printf("Fail to set CNT alarm configuration\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5080.cpp

2.43. CNT_SetChannelMask

Users can use this function to set enabled counter channel mask of the indicated slot.

Syntax

CNT_SetChannelMask(
  LONG handle,
  WORD i_wSlot,
  DWORD i_dwMask
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_dwMask	Input	[in] The enabled counter channel mask to be set. ***Note:** From LSB to MSB of the value indicate the channel-0 to the last channel mask. Set bit value to 1 and the channel will be enabled.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD dwMask = 0xFFFF;
if(ERR_SUCCESS != CNT_SetChannelMask (lDriverHandle, wSlotID, dwMask))
    printf("Fail to set channel mask\n");
```

For more detailed information regarding this function, please see

$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5080.cpp

2.44. CNT_SetCntTypeConfig

Users can use this function to set the counter counting type of the indicated slot.

Syntax

CNT_SetCntTypeConfig(
  LONG handle,
  WORD i_wSlot,
  WORD i_wChannel,
  BOOL* o_bRepeat,
  BOOL* o_bReload
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_wChannel	Input	[in] The channel index.
i_bRepeat	Input	[in] The repeat enabled status.
i_bReload	Input	[in] The startup enabled status.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
const int iChannelCNTNum = 8;
WORD wChannel = 2; /* Channel ID */
BOOL byRepeat = 1; /* Repeat enabled status */
BOOL byReload = 1; /* Startup enabled status */
if(ERR_SUCCESS != CNT_SetCntTypeConfig(lDriverHandle, wSlotID, wChannel,
byRepeat, byReload))
    printf("Fail to set CNT type configuration.\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5080.cpp

2.45. CNT_SetFilter

Users can use this function to set the counter filter width of the indicated slot.

Syntax

CNT_SetFilter(
  LONG handle,
  WORD i_wSlot,
  DWORD i_dwWidth
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_dwWidth	Input	[in] The counter filter width to be set and ranging from 0 to 40000 (us)

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD dwWidth = 1000;
if(ERR_SUCCESS != CNT_SetFilter(lDriverHandle, wSlotID, dwWidth))
    printf("Fail to set CNT filter width\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5080.cpp

2.46. CNT_SetFreqAcqTime

Users can use this function to set the counter frequency acquisition time of the

indicated slot.

Syntax

CNT_SetFreqAcqTime(
  LONG handle,
  WORD i_wSlot,
  DWORD i_dwFreqAcqTime
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_dwFreqAcqTime	Input	[in] The frequency acquisition time, ranging from 0 to 10000 (ms)

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD dwFreqAcqTime = 1000;
if(ERR_SUCCESS != CNT_SetFreqAcqTime(lDriverHandle, wSlotID,
dwFreqAcqTime))
    printf("Fail to set CNT frequency acquisition time\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5080.cpp

2.47. CNT_SetGateConfig

Users can use this function to set the counter gate configuration of the indicated slot.

Syntax

CNT_SetGateConfig(
  LONG handle,

```
  WORD i_wSlot,
  WORD i_wChannel,
  BOOL i_bEnable,
  BYTE i_byTriggerMode,
  BYTE i_byGateActiveType,
  BYTE i_byMapGate
);
```

Parameters

<table>
<tr><th>Name</th><th>Direction</th><th>Description</th></tr>
<tr><td>handle</td><td>Input</td><td>[in] The driver handler.</td></tr>
<tr><td>i_wSlot</td><td>Input</td><td>[in] The slot ID which is ranged from 0 to 15.</td></tr>
<tr><td>i_wChannel</td><td>Input</td><td>[in] The channel index.</td></tr>
<tr><td>i_bEnable</td><td>Input</td><td>[in] The gate enabled status to be set.</td></tr>
<tr><td>i_byTriggerMode</td><td>Input</td><td>[in] The trigger mode to be set.
*Note:
The value of i_byTriggerMode indicates:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>Non re-trigger</td></tr>
<tr><td>1</td><td>Re-trigger</td></tr>
<tr><td>2</td><td>Edge start</td></tr>
</table></td></tr>
<tr><td>i_byGateActiveType</td><td>Input</td><td>[in] The gate active type to be set.
*Note:
The value of i_byGateActiveType indicates:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>Low level</td></tr>
<tr><td>1</td><td>Falling edge</td></tr>
<tr><td>2</td><td>High level</td></tr>
<tr><td>3</td><td>Rising edge</td></tr>
</table></td></tr>
<tr><td>i_byMapGate</td><td>Input</td><td>[in] The map gate to be set, ranged from 0 to 3</td></tr>
</table>

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
```

```
WORD wChannel = 2; /* Channel ID */
BOOL byGateEnabled = true; /* Gate enabled status */
BYTE byTrig = 2; /* Edge start */
BYTE byActive = 2; /* High level */
BYTE byMapGate = 2; /* Map gate */
if(ERR_SUCCESS != CNT_SetGateConfig(lDriverHandle, wSlotID, wChannel,
byGateEnabled, byTrig, byActive, byMapGate))
    printf("Fail to set CNT gate configuration.\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5080.cpp

2.48. CNT_SetRanges

Users can use this function to set the channel ranges of the indicated slot.

Syntax

```
CNT_SetRanges(
  LONG handle,
  WORD i_wSlot,
  WORD i_wChannelTotal,
  WORD* i_wRanges
);
```

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_wChannelTotal	Input	[in] The channel total of the module in the indicated slot. ***Note:** The maximum value is 8.
i_wRanges	Input	[in] The ranges to be set. The value of i_wRanges indicates: Value — Meaning 0x01C0 — Bi-directory 0x01C1 — Up/Down 0x01C2 — Up

		0x01C3	High Frequency
		0x01C4	A/B-1X
		0x01C5	A/B-2X
		0x01C6	A/B-4X
		0x01C7	Low Frequency
		0x01C8	Wave Width
		***Note:** **Bi-direction mode, Up/Down mode and A/B phase mode must be set in pairs**	

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
const int iChannelCNTNum = 8;
WORD wModes[iChannelCNTNum] = {0};
wModes[0] = 0x01C1; /* UP_AND_DOWN */
wModes[1] = 0x01C1; /* UP_AND_DOWN */
wModes[2] = 0x01C4; /* AB_1X */
wModes[3] = 0x01C4; /* AB_1X */
wModes[4] = 0x01C2; /* UP */
wModes[5] = 0x01C3; /* HIGH_FREQUENCY */
wModes[6] = 0x01C2; /* UP */
wModes[7] = 0x01C2; /* UP */
if(ERR_SUCCESS != CNT_SetRanges(lDriverHandle, wSlotID, iChannelCNTNum,
wModes))
    printf("Fail to set the channel ranges \n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5080.cpp

2.49. CNT_SetStartupValues

Users can use this function to set the counter startup values of the indicated slot.

Syntax

```
CNT_SetStartupValues(
  LONG handle,
  WORD i_wSlot,
  WORD i_wChannelTotal,
  DWORD* i_dwValues
);
```

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_wChannelTotal	Input	[in] The channel total of the module in the indicated slot. ***Note:** The maximum value is 8.
o_dwValues	Input	[in] The counter startup values to be set. ***Note:** The size of this array must be i_wChannelTotal DWORDs.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
const int iChannelCNTNum = 8;
WORD wSlotID = 1; /* Slot ID */
DWORD dStartValues[iChannelCNTNum] = {0};
dStartValues[0] = 0x0011;
dStartValues[1] = 0x1100;
if(ERR_SUCCESS != CNT_SetStartupValues(lDriverHandle, wSlotID,
iChannelCNTNum, dStartValues))
    printf("Fail to set the counter startup values.\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5080.cpp

2.50. DI_GetFilters

Users can use this function to get DI filter mask and width of the indicated slot.

Syntax

```
DI_GetFilters(
  LONG handle,
  WORD i_wSlot,
  DWORD* o_dwHighMask,
  DWORD* o_dwLowMask,
  DWORD* o_dwWidth
);
```

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 31.
o_dwHighMask	Output	[out] The variable to hold the DI filter mask from channel 32 to 63. The LSB indicates the channel 32.
o_dwLowMask	Output	[out] The variable to hold the DI filter mask from channel 0 to 31. The LSB indicates the channel 0.
o_dwWidth	Output	[out] The variable to hold the DI filter width.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
const int iChannelDINum = 8;
WORD wSlotID = 1; /* Slot ID */
DWORD dwHighMask = 0;
DWORD dwLowMask = 0;
DWORD dwWidth = 0;
LONG lResult = DI_GetFilters(lDriverHandle, wSlotID, &dwHighMask,
&dwLowMask, &dwWidth);
if (ERR_SUCCESS == lResult) {
```

```
    printf("The filter width is %d (0.1 ms).\n", dwWidth);
    for (int iCnt = 0; iCnt < iChannelDINum ; iCnt++) {
        if (dwLowMask & (0x0001 << iCnt)) {
            printf("Channel %d filter mask is true.\n", iCnt);
        else
            printf("Channel %d filter mask is false.\n", iCnt);
        }
    }
}
else
    printf("Fail to get filter mask\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5045.cpp

2.51. DI_SetFilters

Users can use this function to set DI filter mask and width of the indicated slot.

Syntax

DI_GetFilters(
  LONG handle,
  WORD i_wSlot,
  DWORD i_dwHighMask,
  DWORD i_dwLowMask,
  DWORD i_dwWidth
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 31.
i_dwHighMask	Input	[in] The DI filter mask from channel 32 to 63. The LSB indicates the channel 32.
i_dwLowMask	Input	[in] The DI filter mask from channel 0 to 31. The LSB indicates the channel 0.
i_dwWidth	Input	[in] The DI filters width, ranging from 5 to 400 (0.1 ms).

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD dwHighMask = 0x0;
DWORD dwLowMask = 0x00FF;
DWORD dwWidth = 300;
if(ERR_SUCCESS != DI_SetFilters(lDriverHandle, wSlotID, dwHighMask,
dwLowMask, dwWidth))
    printf("Fail to set filter mask\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5045.cpp

2.52. DIO_GetSaftyValues

Users can use this function to get the all DI/DO safety values of the indicated slot.

Syntax

DIO_GetSaftyValues(
  LONG handle,
  WORD i_wSlot,
  DWORD* o_dwHighValue,
  DWORD* o_dwLowValue
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 31.
o_dwHighValue	Output	[out] The variable to hold the DI/DO safety values from channel 32 to 63. The LSB indicates the channel 32.
o_dwLowValue	Output	[out] The variable to hold the DI/DO safety values from channel 0 to 31.

		The LSB indicates the channel 0.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)** and **o_dwHighValue** and **o_dwLowValue** contain DI/DO values from channel-0 to the last channel. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
const int iChannelDONum = 12;
WORD wSlotID = 1; /* Slot ID */
DWORD dHiValue = 0x0;
DWORD dLowValue = 0x0;
LONG lResult = DIO_GetSaftyValues(lDriverHandle, wSlotID, &dHiValue,
&dLowValue);
if (ERR_SUCCESS == lResult) {
    for (int iCnt = 0; iCnt < iChannelDONum ; iCnt++) {
        if (dLowValue & (0x0001 << iCnt) )
            printf("The safety value of channel %d is true.\n", iCnt);
        else
            printf("The safety value of channel %d is false.\n", iCnt);
        }
}
else
    printf("Fail to get DO safety values\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5045.cpp

2.53. DIO_GetValue

Users can use this function to get the DI/DO value of the indicated slot and channel.

Syntax

DIO_GetValue (
 LONG handle,
 WORD i_wSlot,
 WORD i_wChannel,

BOOL* o_bValue
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 31.
i_wChannel	Input	[in] The channel ID which is ranged from 0 to 63.
o_bValue	Output	[out] The variable to hold the DI/DO value.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)** and **o_bValue** contains DI/DO value. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
const int iChannelDONum = 12;
WORD wSlotID = 1; /* Slot ID */
WORD wChannel = 1;
BOOL bHoldValue = false;
LONG lResult = DIO_GetValue(lDriverHandle, wSlotID, wChannel,
&bHoldValue);
if (ERR_SUCCESS == lResult) {
    if (bHoldValue)
        printf("Channel %d is true.\n", wChannel);
    else
        printf("Channel %d is false.\n",wChannel);
}
else
    printf("Fail to get DI/DO single channel value \n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5045.cpp

2.54. DIO_GetValues

Users can use this function to get all DIO values of the indicated slot.

Syntax

DIO_GetValues (
  LONG handle,
  WORD i_wSlot,
  DWORD* o_dwHighValue,
  DWORD * o_dwLowValue
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 31.
o_dwHighValue	Output	[out] The variable to hold the DIO values from channel 32 to 63. The LSB indicates the channel 32.
o_dwLowValue	Output	[out] The variable to hold the DIO values from channel 0 to 31. The LSB indicates the channel 0.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)** and **o_dwHighValue** and **o_dwLowValue** contain DIO values from channel 0 to the last channel. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
const int iChannelDONum = 12;
WORD wSlotID = 1; /* Slot ID */
DWORD dHiValue = 0x0;
DWORD dLowValue = 0x0;
LONG lResult = DIO_GetValues(lDriverHandle, wSlotID, &dHiValue,
&dLowValue);
if (ERR_SUCCESS == lResult) {
    for (int iCnt = 0; iCnt < iChannelDONum ; iCnt++) {
        if (dLowValue & (0x0001 << (iCnt + iChannelDONum) ) )
            printf("Channel %d is true.\n", iCnt);
        else
            printf("Channel %d is false.\n", iCnt);
    }
}
```

```
else
    printf("Fail to get DI/DO multiple channel values \n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5045.cpp

2.55. DIO_SetSaftyValues

Users can use this function to set the all DI/DO safety values of the indicated slot.

Syntax

DIO_SetSaftyValues(
  LONG handle,
  WORD i_wSlot,
  DWORD i_dwHighValue,
  DWORD i_dwLowValue
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 31.
i_dwHighValue	Input	[in] The DI/DO safety values from channel 32 to 63. The LSB indicates the channel 32.
i_dwLowValue	Input	[in] The DI/DO safety values from channel 0 to 31. The LSB indicates the channel 0.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
const int iChannelDONum = 12;
WORD wSlotID = 1; /* Slot ID */
DWORD dHiValue = 0x0;
DWORD dLowValue = 0x000000F;
if(ERR_SUCCESS != DIO_SetSaftyValues(lDriverHandle, wSlotID, dHiValue,
```

```
dLowValue))
    printf("Fail to set DO safety values \n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5045.cpp

2.56. DO_BufValues

Users can use this function to buffer the DO values of the indicated slot.

Syntax

DO_BufValues (
  LONG handle,
  WORD i_wSlot,
  DWORD i_dwHighValue,
  DWORD i_dwLowValue
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 31.
i_dwHighValue	Input	[in] The DO values from channel 32 to 63. The LSB indicates the channel 32.
i_dwLowValue	Input	[in] The DO values from channel 0 to 31. The LSB indicates the channel 0.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID[2] = {0};
DWORD dHiValue[2] = {0};
DWORD dLowValue[2] ={0};

wSlotID[0] = 1;
```

```
dHiValue[0] = 0x0;
dLowValue[0] = 0x000F0F0F;
wSlotID[1] = 2;
dHiValue[1] = 0x0;
dLowValue[1] = 0x00F0F0F0;
LONG lResult = 0;
for (int iCnt = 0; iCnt < 2 ; iCnt++ ) {
   lResult = DO_BufValues(lDriverHandle, wSlotID[iCnt], dHiValue[iCnt],
dLowValue[iCnt]);
    if (ERR_SUCCESS == lResult)
        printf("Succeed to buffer values of slot %d.\n",wSlotID[iCnt]);
    else
        printf("Fail to buffer values\n");
}
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5045.cpp

2.57. DO_SetValue

Users can use this function to set DO value of the indicated slot and channel.

Syntax

DO_SetValue (
  LONG handle,
  WORD i_wSlot,
  WORD i_wChannel,
  BOOL i_bValue
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 31.
i_wChannel	Input	[in] The channel ID which is ranged from 0 to 63.
i_bValue	Input	[in] The DO value to be set.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
WORD wChannel = 7; //the channel ID from 0 to 11
BOOL bSetValue = true; // DO value to be set

if(ERR_SUCCESS != DO_SetValue(lDriverHandle, wSlotID, wChannel,
bSetValue))
    printf("Fail to set single DO value.\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5045.cpp

2.58. DO_SetValues

Users can use this function to set all DO values of the indicated slot.

Syntax

DO_SetValues (
  LONG handle,
  WORD i_wSlot,
  DWORD i_dwHighValue,
  DWORD i_dwLowValue
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 31.
i_dwHighValue	Input	[in] The DO values from channel 32 to 63. The LSB indicates the channel 32.
i_dwLowValue	Input	[in] The DO values from channel 0 to 31. The LSB indicates the channel 0.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD dHiValue = 0x0;
DWORD dLowValue = 0x0000F0F;
if(ERR_SUCCESS != DO_SetValues(lDriverHandle, wSlotID, dHiValue,
dLowValue))
    printf("Fail to set DO multiple channel values.\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5045.cpp

2.59. LED_DeviceOpen

Users can use this function to initialize LED devices. The function returns a handle that can be used to control LED lights.

Syntax

LED_DeviceOpen (
  ULONG DeviceNum,
  LONG *handle
);

Parameters

Name	Direction	Description
DeviceNum	Input	[in] The device number. Currently, it is ignored.
handle	Output	[out] The LED driver handler.

Return Value

If driver initialization succeeded, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **LED_GetErrorMessage** function.

Remarks

Use the **LED_DeviceClose** function to terminate the LED devices.

Example

```
LONG lDriverHandle = NULL; /* Driver handler */
if(ERR_SUCCESS != LED_DeviceOpen (0, &lDriverHandle))
    printf("Fail to initialize LED devices\n");
```

2.60. LED_DeviceClose

Users can use this function to terminate the LED devices.

Syntax

LED_DeviceClose (
 LONG *handle
);

Parameters

Name	Direction	Description
handle	Input	[in] The LED driver handler.

Return Value

If driver termination succeeded, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **LED_GetErrorMessage** function.

Example

```
if(NULL != lDriverHandle) {
    LED_DeviceClose (&lDriverHandle);
    lDriverHandle = NULL;
}
```

2.61. LED_EnableLedAndSetMode

Users can use this function to set LED modes

Syntax

LED_EnableLedAndSetMode (
 LONG handle,
 USHORT mode

);

Parameters

<table>
<tr><th>Name</th><th>Direction</th><th>Description</th></tr>
<tr><td>handle</td><td>Input</td><td>[in] The LED driver handler.</td></tr>
<tr><td>mode</td><td>Input</td><td>[in] The LED modes.
Note:
The value of mode indicates:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>Turn off</td></tr>
<tr><td>1</td><td>Turn on</td></tr>
<tr><td>3</td><td>Blinking (fast)</td></tr>
<tr><td>5</td><td>Blinking (normal)</td></tr>
<tr><td>7</td><td>Blinking (slow)</td></tr>
</table></td></tr>
</table>

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **LED_GetErrorMessage** function.

Remarks

This function ONLY supports for APAX-557x and APAX-657x controllers.

Need to be used with **LED_ToggleLedFlashing** function.

Example

```
USHORT  mode = 7; /* Slow blinking mode */
int  iLedNum = 4; /* The total number of LED */

if(ERR_SUCCESS == LED_EnableLedAndSetMode (lDriverHandle, mode)) {
    DWORD  dwErrCde;
    for(int i = 0; i < iLedNum; i++)
    {
        if (ERR_SUCCESS != LED_ToggleLedFlashing(lDriverHandle, 1, i))
        {
            printf("Fail to flash LED light\n");
        }
    }
}
```

```
else
    printf("Fail to set LED mode\n");
```

2.62. LED_GetErrorMessage

Users can use this function to get error information.

Syntax

LED_GetErrorMessage (
 LONG handle,
 LPTSTR lpszErrMsg
);

Parameters

Name	Direction	Description
handle	Input	[in] The LED driver handler.
lpszErrMsg	Output	[out] The error message.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
LONG lDriverHandle = NULL; /* Driver handler */
LONG lResult = LED_DeviceOpen (&lDriverHandle);
if(ERR_SUCCESS != lResult) {
    TCHAR szErrMsg[180];
    printf("Fail to open driver\n");
    LED_GetErrorMessage(lResult, szErrMsg);
    printf("Error message is %s\n", szErrMsg);
}
```

2.63. LED_GetPLEDInformation

Users can use this function to get the total number of LEDs.

Syntax

LED_GetPLEDInformation (
 LONG handle,
 UCHAR *LEDCount
);

Parameters

Name	Direction	Description
handle	Input	[in] The LED driver handler.
PLEDCount	Output	[out] The total number of LEDs

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **LED_GetErrorMessage** function.

Example

```
UCHAR  LEDCount = 0;
if(ERR_SUCCESS == LED_GetPLEDInformation(lDriverHandle, &LEDCount))
    printf("There are %d LEDs", LEDCount);
```

2.64. LED_ReadProgramLedByte

Users can use this function to get LED status.

Syntax

LED_ReadProgramLedByte (
 LONG handle,
 UCHAR *value
);

Parameters

Name	Direction	Description
handle	Input	[in] The LED driver handler.
value	Output	[out] The LED status. The value of status indicates: Value — Meaning 0 — Turned off 1 — Turned on

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **LED_GetErrorMessage** function.

Remarks

Use the **ADAMDrvClose** function to close the ADAM/APAX device.

Example

```
int  iLedNum = 4; /* The total number of LED */
UCHAR  byReadBack = 0;
int i = 0;
do {
    if(ERR_SUCCESS == LED_ReadProgramLedByte(lDriverHandle,
&byReadBack)
        printf("[LED %d] The current state is %d\n", i, byReadBack);
    i++;
 }while(i < iLedNum);
```

2.65. LED_ToggleLedFlashing

Users can use this function to flash LEDs.

Syntax

LED_ToggleLedFlashing (
 LONG handle,
 USHORT mode,
 int index
);

Parameters

Name	Direction	Description
handle	Input	[in] The LED driver handler.
mode	Input	[in] The LED flash mode ***Note:** The value of mode indicates: Value Meaning

		0 Turn off 1 Turn on
index	Input	[in] The LED index

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **LED_GetErrorMessage** function.

Example

```
USHORT  mode = 7; /* Slow blinking mode */
int  iLedNum = 4; /* The total number of LED */

if(ERR_SUCCESS == LED_EnableLedAndSetMode (lDriverHandle, mode)) {
    DWORD  dwErrCde;
    for(int i = 0; i < iLedNum; i++)
    {
        if (ERR_SUCCESS != LED_ToggleLedFlashing(lDriverHandle, 1, i))
        {
            printf("Fail to flash LED light\n");
        }
    }
}
else
    printf("Fail to set LED mode\n");
```

2.66. LED_WriteProgramLedByte

Users can use this function to set LED status.

Syntax

LED_WriteProgramLedByte (
  LONG handle,
  UCHAR value
);

Parameters

Name	Direction	Description

handle	Input	[in] The LED driver handler.
value	Input	[in] The LED status. The value of status indicates: Value — Meaning 0 — Turned off 1 — Turned on

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **LED_GetErrorMessage** function.

Example

```
int  iLedNum = 4; /* The total number of LED */
UCHAR  byData = 1;
int i = 0;
do {
    if(ERR_SUCCESS == LED_WriteProgramLedByte (lDriverHandle, byData ))
        printf("LED %d is turned on\n", i);
    byteData = (byData << 1) + 1;
    i++;
 }while(i < iLedNum);
```

2.67. OUT_FlushBufValues

Users can use this function to flush the buffered values.

Syntax

OUT_FlushBufValues (
 LONG handle,
 DWORD i_dwSlotMask
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_dwSlotMask	Input	[in] The flush slot mask. ***Note:**

		The LSB indicates the slot 0. Set bit value to 1 and values of the slot will be buffered.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
DWORD dSlotMask = 0x05; /*slot 0 and slot 2*/
if(ERR_SUCCESS != OUT_FlushBufValues(lDriverHandle, dSlotMask));
    printf("Fail to flush buffered values\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5045.cpp

2.68. OUT_SetSaftyEnable

Users can use this function to set the safety value enabled status of the indicated slot.

Syntax

```
OUT_SetSaftyEnable(
  LONG handle,
  WORD i_wSlot,
  BOOL i_bEnable
);
```

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_bEnable	Input	[in] The safety value enabled status.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
BOOL bEnabled = true;
if(ERR_SUCCESS != OUT_SetSaftyEnable(lDriverHandle, wSlotID, bEnabled))
    printf("Fail to set DO safety function status\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5045.cpp

2.69. PWM_GetChannelMask

Users can use this function to get PWM channel mask of the indicated slot.

Syntax

PWM_GetChannelMask(
  LONG handle,
  WORD i_wSlot,
  DWORD* o_dwMask
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
o_dwMask	Output	[out] The variable to hold the PWM channel mask. From LSB to MSB of the value indicate the channel 0 to channel 31 enabled mask.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
const int g_ChannelPWMNum = 8; /* CNT channel number is 8 */
WORD wSlotID = 1; /* Slot ID */
DWORD dwMask = 0;
```

```
LONG lResult = PWM_GetConfig(lDriverHandle, wSlotID, &dwMask);
if (ERR_SUCCESS == lResult)
    {
        printf("Succeed to get PWM channel mask.\n");
        for (int iCnt = 0; iCnt < g_ChannelPWMNum; iCnt++)
        {
            if (dwMask & (0x01 << iCnt))
                printf("The PWM channel %d is enabled.\n", iCnt);
            else
                printf("The PWM channel %d is disabled.\n", iCnt);
        }
    }
    else
        printf("Fail to get PWM channel mask");
```

For more detailed information regarding this function, please see $(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5082.cpp

2.70. PWM_GetConfig

Users can use this function to get channel configuration of the indicated slot.

Syntax

PWM_GetConfig(
  LONG handle,
  WORD i_wSlot,
  WORD i_wChannel,
  DWORD *o_dwFreq,
  float *o_fDuty
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_wChannel	Input	[in] The channel index.
o_dwFreq	Output	[out] The variable to hold the channel's frequency. (1 - 30K Hz)
o_fDuty	Output	[out] The variable to hold the channel's duty cycle.

		(0.1 ~ 99.9 %)

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
WORD wChannel = 5;
DWORD dwFreq = 0; /* Frequency */
float fDuty = 0.0; /* Duty cycle */
LONG lResult = PWM_GetConfig(lDriverHandle, wSlotID, wChannel, &dwFreq,
&fDuty);
if (ERR_SUCCESS == lResult) {
    printf("The frequency is %d (Hz) and duty cycle is %2.2f %%.\n",
    dwFreq, fDuty);
}
else
    printf("Fail to get PWM channel configuration\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5082.cpp

2.71. PWM_SetChannelMask

Users can use this function to set PWM channel mask of the indicated slot.

Syntax

PWM_SetChannelMask(
  LONG handle,
  WORD i_wSlot,
  DWORD i_dwMask
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.

i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_dwMask	Input	[in] The PWM channel mask. From LSB to MSB of the value indicate the channel 0 to channel 31 enabled mask.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD dwMask = 0x0000FFFF;
if(ERR_SUCCESS != PWM_SetChannelMask(lDriverHandle, wSlotID, dwMask))
    printf("Fail to set PWM channel mask\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5082.cpp

2.72. PWM_SetConfig

Users can use this function to set channel configuration of the indicated slot.

Syntax

PWM_SetConfig(
  LONG handle,
  WORD i_wSlot,
  WORD i_wChannel,
  DWORD i_dwFreq,
  float i_fDuty
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 15.
i_wChannel	Input	[in] The channel index.
i_dwFreq	Input	[in] The channel frequency. (1 - 30K Hz)

i_fDuty	Input	[in] The channel duty cycle. (0.1 ~ 99.9 %)

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
WORD wChannel = 5;
DWORD dwFreq = 1000;
float fDuty = 66.5;
if(ERR_SUCCESS != PWM_SetConfig(lDriverHandle, wSlotID, wChannel,
dwFreq, fDuty))
    printf("Fail to set PWM channel configuration\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5082.cpp

2.73. SYS_GetControllerID

Users can use this function to get the controller ID.

Syntax

SYS_GetControllerID (
  LONG handle,
  DWORD * o_dwControllerID
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
o_dwControllerID	Output	[out] The variable to hold controller ID

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
DWORD dwID = 0;
if(ERR_SUCCESS == SYS_GetControllerID (lDriverHandle, &dwID))
    printf("The controller ID is %x", dwID);
else
    printf("Fail to get controller ID\n");
```

2.74. SYS_GetModuleID

Users can use this function to get the module ID of the indicated slot.

Syntax

SYS_GetModuleID (
  LONG handle,
  WORD i_wSlot,
  DWORD * o_dwModuleID
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 31.
o_dwModuleID	Output	[out] The variable to hold module ID.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD dwModuleID = 0;
LONG lResult = SYS_GetModuleID(lDriverHandle, wSlotID, &dwModuleID);
if (ERR_SUCCESS == lResult) {
    DWORD dwHiWord = (dwModuleID & 0xFFFF0000) >> 16;
    DWORD dwLoWord =  dwModuleID & 0x0000FFFF;
    if (0 == dwLoWord)
```

```
        printf("APAX-%x\n", dwHiWord);
    else if (0 != (dwLoWord & 0x0000FF00) && (dwLoWord & 0x000000FF) )
        printf("APAX-%x%c%c\n", dwHiWord, dwLoWord >> 8, (dwLoWord &
0x000000FF) );
    else
        printf("APAX-%x%c \n", dHiWord, dLoWord >> 8 );
}
else
    printf("Fail to get module ID\n");
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5013.cpp

2.75. SYS_GetModuleID_EX

Users can use this function to get APAX-5090P/5095P ID and order of the indicated slot.

Syntax

SYS_GetModuleID_EX (
  LONG handle,
  WORD i_wSlot,
  DWORD * o_dwModuleID,
  int *o_iOrder
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 31.
o_dwModuleID	Output	[out] The variable to hold module ID.
o_iOrder	Output	[out] The variable to hold module order.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Remarks

This function ONLY supports for APAX-5090P and APAX-5095P under Windows CE.

Example

```
WORD wSlotID = 1; /* Slot ID */
DWORD dwModuleID = 0;
int iOrder = 0;
LONG lResult = SYS_GetModuleID_EX(lDriverHandle, wSlotID, &dwModuleID,
&iOrder);
if (ERR_SUCCESS == lResult) {
    DWORD dwHiWord = (dwModuleID & 0xFFFF0000) >> 16;
    printf("APAX-%x\n" with order %d\n, dwHiWord, iOrder);
}
else
    printf("Fail to get module ID\n");
```

2.76. SYS_GetSlotInfo

Users can use this function to get the slot information.

Syntax

SYS_GetSlotInfo (
  LONG handle,
  WORD i_wSlot,
  struct SlotInfo * o_stSlotInfo
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wSlot	Input	[in] The slot ID which is ranged from 0 to 31.
o_stSlotInfo	Output	[out] The SlotInfo structure to store information

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Remarks

The SlotInfo structure contains all system information of the indicated slot. Please use other function to get more specific information.

Example

```
WORD wSlotID = 1; /* Slot ID */
struct SlotInfo sSlotInfo;
memset(&sSlotInfo, 0, sizeof(struct SlotInfo));
if (ERR_SUCCESS != SYS_GetSlotInfo(lDriverHandle,wSlotID, &sSlotInfo))
    printf("Fail to get slot information\n");
```

For more detailed information regarding this function, please see $(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5013.cpp

2.77. SYS_GetVersion

Users can use this function to get the Advantech library version.

Syntax

SYS_GetVersion (
  LONG handle,
  DWORD *o_dwVersion
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
o_dwVersion	Output	[out] The variable to hold the library version.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
DWORD dwVer = 0;
if (ERR_SUCCESS == SYS_GetVersion(lDriverHandle, &dwVer))
```

```
    printf("The APAX Library version is %x\n", dwVer);
```

For more detailed information regarding this function, please see
$(Default install directory)\APAX\Win32\CPlusPlus\APAX-PAC-Sample\APAX-5013.cpp

2.78. SYS_SetInnerTimeout

Users can use this function to set the inner-timeout of the configuration functions that use internal communication channel. The default timeout value is 3 seconds.

Syntax

SYS_SetInnerTimeout (
  LONG handle,
  WORD i_wTimeout
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_wTimeout	Input	[in] The timeout value (in milliseconds). Default is 500 milliseconds.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

Example

```
WORD wTimeout = 1000; /* milliseconds */
if (ERR_SUCCESS != SYS_SetInnerTimeout (lDriverHandle, wTimeout))
    printf("Fail to set inner-timeout \n");
```

Chapter 3

3. Internal functions

3.1. SYS_GetAllSlotErrorFlag

Users can use this function to get the presence of a module for each slot.

Syntax

SYS_GetAllSlotErrorFlag(
  LONG handle,
  DWORD* o_wError
)

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
o_wError	Output	[out] The variable to hold all slots status. ***Note:** From LSB to MSB of the value indicates the slot 0 to the last slot status. If the bit is 1, it means that the slot has no module present.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

3.2. SYS_GetAppAStatus

For backup system, users can use this function to get the application A status.

Syntax

SYS_GetAppAStatus(
  LONG handle,
  BOOL* o_bAlive
)

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
o_bAlive	Output	[out] The variable to hold the application A status.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call GetLastError function.

3.3. SYS_GetAppBStatus

For backup system, users can use this function to the application B status.

Syntax

SYS_GetAppAStatus(
  LONG handle,
  BOOL* o_bAlive
)

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
o_bAlive	Output	[out] The variable to hold the application B status.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

3.4. SYS_GetDspChannelFlag

Users can use this function to get DSP channel flag.

Syntax

SYS_GetDspChannelFlag (
  LONG handle,
  WORD* o_wFlag
)

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
o_wFlag	Output	[out] The variable to hold the DSP channel flag.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

3.5. SYS_GetFpgaVersion

Users can use this function to get the FPGA version.

Syntax

SYS_GetFpgaVersion (
  LONG handle,
  DWORD * o_dwVer
);

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
o_dwVer	Output	[out] The variable to hold the FPGA version.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails,

the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

3.6. SYS_GetGlobalActive

For backup system, users can use this function to get global active.

Syntax

```
SYS_GetGlobalActive (
  LONG handle,
  WORD * o_wActive
);
```

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
o_wActive	Output	[out] The variable to hold the global active.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the function returns error code.

3.7. SYS_GetGlobalMachineID

For backup system, users can use this function to get the machine ID.

Syntax

```
SYS_GetGlobalMachineID (
  LONG handle,
  WORD * o_wID
);
```

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
o_wID	Output	[out] The variable to hold the machine ID.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the function returns error code.

3.8. SYS_GetGlobalStatus

For backup system, users can use this function to get global status.

Syntax

```
SYS_GetGlobalStatus (
  LONG handle,
  WORD * o_wStatus
);
```

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
o_wStatus	Output	[out] The variable to hold the global status.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

3.9. SYS_GetHeartbeatRun

For backup system, users can use this function to get heartbeat run.

Syntax

```
SYS_GetHeartbeatRun (
  LONG handle,
  BOOL * o_bRun
);
```

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
o_bRun	Output	[out] The variable to hold the heartbeat run.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

3.10. SYS_GetRawData

Users can use this function to get raw data from Dual Port RAM.

Syntax

```
SYS_GetRawData(
  LONG handle,
  DWORD i_dwStart,
  DWORD i_dwLength,
  BYTE *o_dwRawData
)
```

Parameters

Name	Direction	Description
handle	Input	[in] The driver handler.
i_dwStart	Input	[in] The start address. ***Note:** The address MUST be Multiple of 4
i_dwLength	Input	[in] The data length. ***Note:** The maximum length is 256 BYTE
o_dwRawData	Output	[out] The variable to hold raw data.

Return Value

If the function succeeds, the return value is **0 (ERR_SUCCESS)**. If the function fails, the return value is **325 (ERR_INTERNAL_FAILED)**. To get extended error information, call **GetLastError** function.

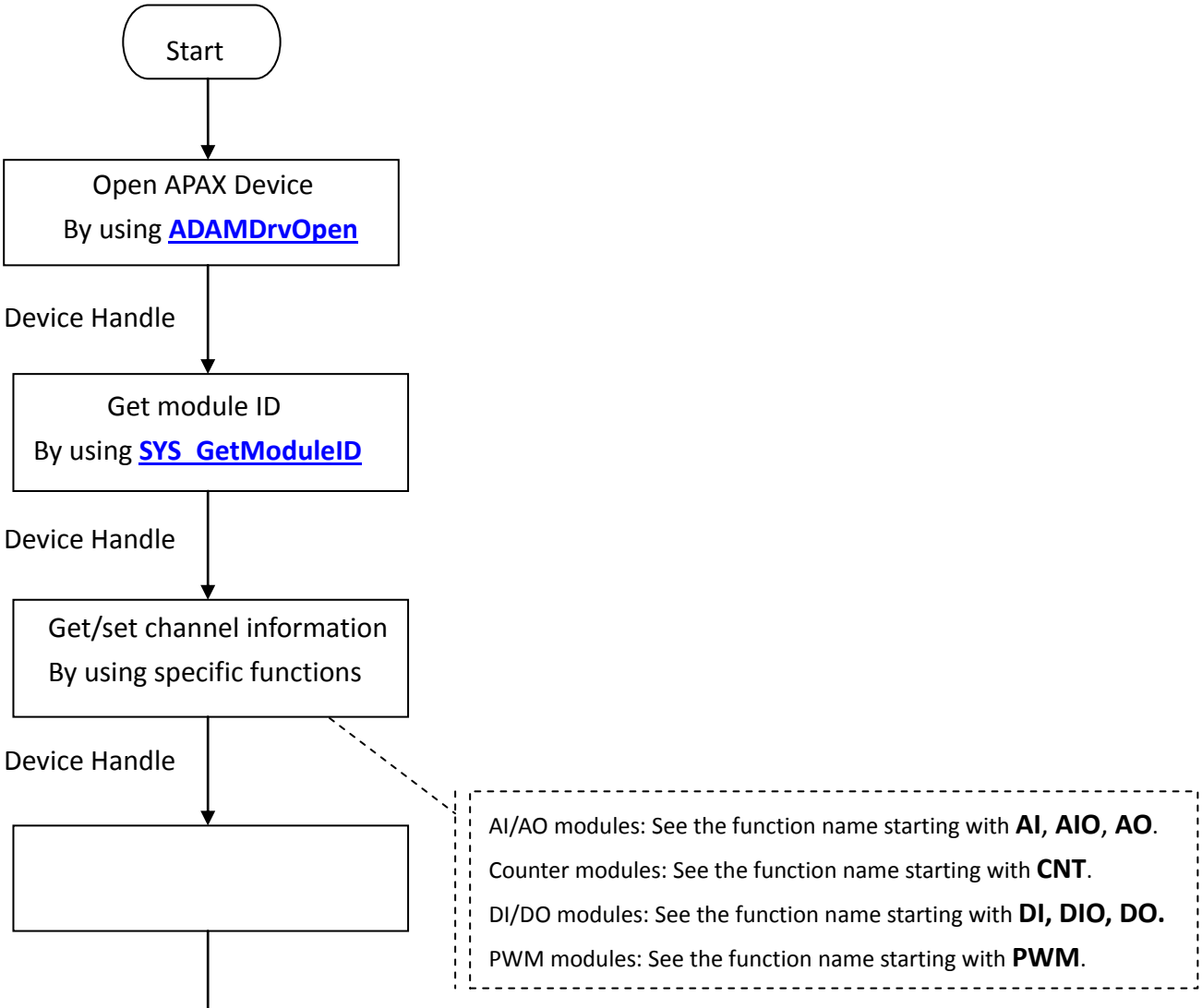
Appendix A

A1. Call flow

Users can directly access drivers with ADSDIO API. Necessary files for developing applications are listed below. Suppose installation paths of all header files in the example are C:\Program Files(x86)\Advantech\AdamApax.NET Class Library\Sample Code\APAX\Win32\CPlusPlus\include

Common Call Flow

The following figure describes the common call flow of the APAX modules.

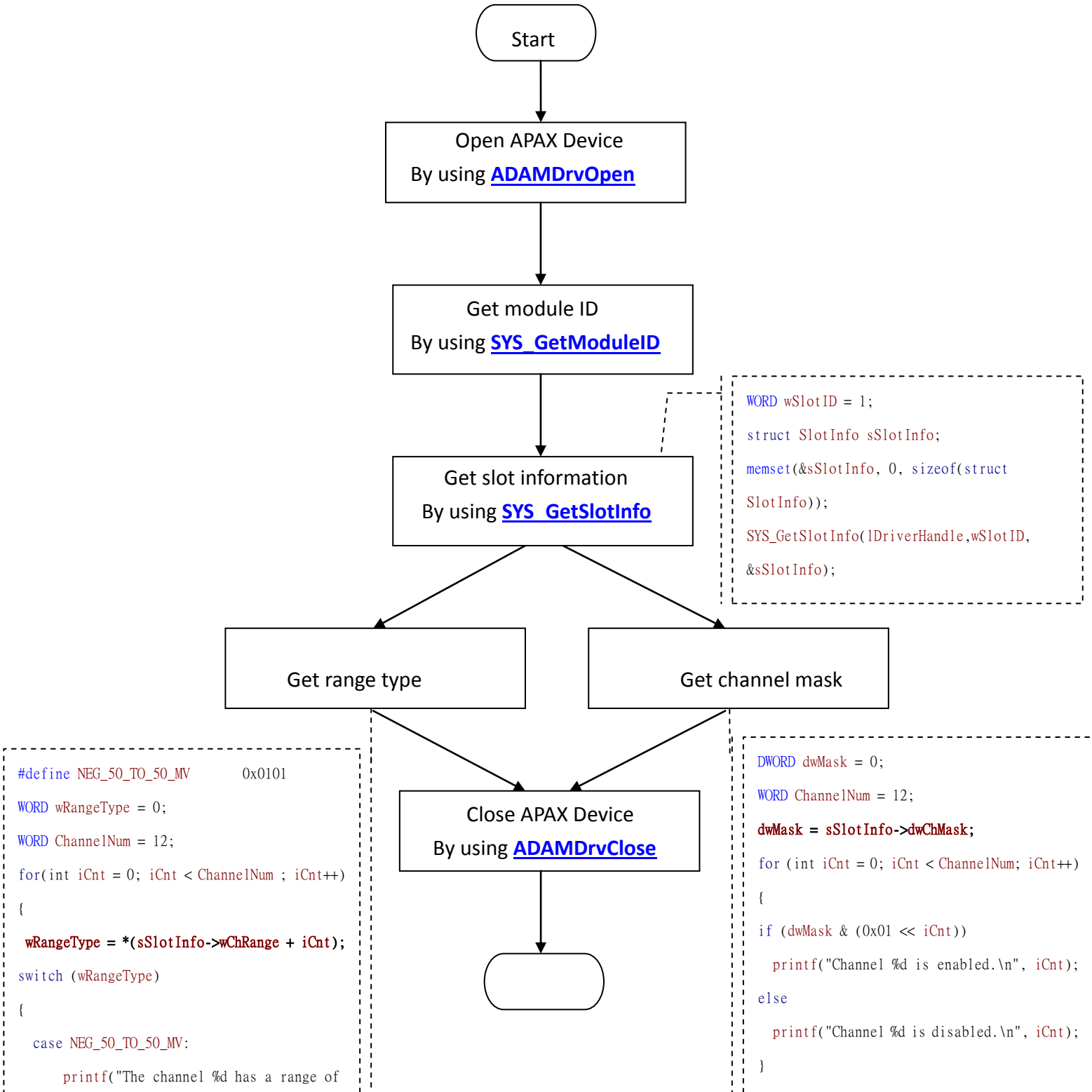


Close APAX Device

By using **ADAMDrvClose**

End

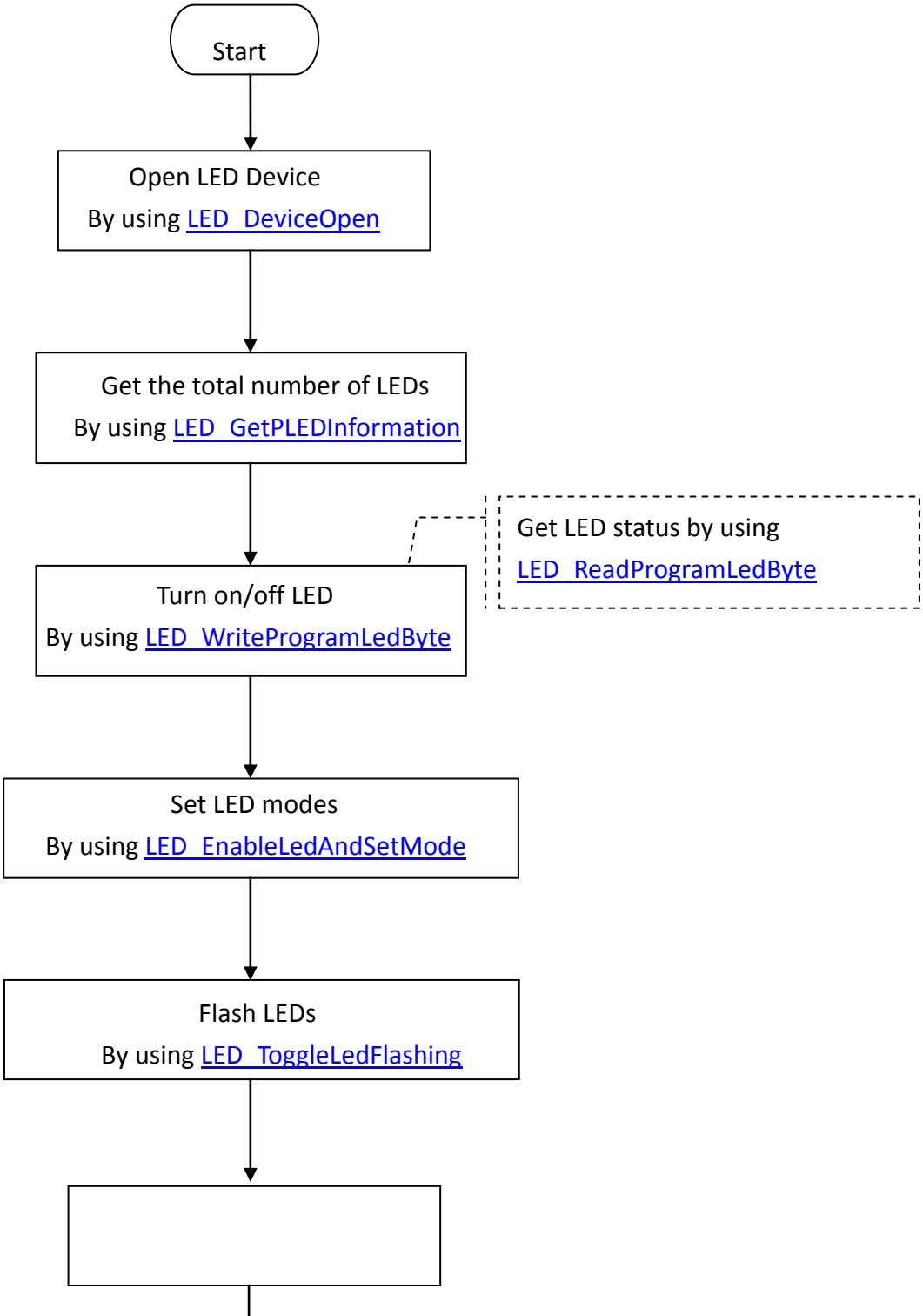
AI/AO Modules Call Flow for Getting Range Type or Channel Mask



End

LED Call Flow

The following figure describes how to control LEDs.



Close LED Device

By using LED_DeviceClose

End

Appendix B

B1. Error Codes

The information on this page is intended to be used by programmers so that the software they write can better deal with errors.
The following list describes system error codes. They are returned by the **GetLastError** function when many functions fail.

ERROR_SUCCESS
    0 (0x0)
    The operation completed successfully.

ERR_MALLOC_FAILED
    300 (0x12C)
    The system fails to allocate memory.

ERR_MAPADDR_FAILED
    301 (0x12D)
    The system fails to map address.

ERR_HANDLE_INVALID
    302 (0x12E)
    The handle is invalid.

ERR_MODULE_INVALID
    303 (0x12F)
    The module is invalid.

ERR_SLOT_INVALID
    304 (0x130)

The slot index is invalid.

ERR_CHANNEL_INVALID

305 (0x131)

The channel index is invalid.

ERR_FUNC_INVALID

306 (0x132)

The function is invalid.

ERR_INTRINIT_FAILED

307 (0x133)

The internal Initialization is failed.

ERR_FREQMEASU_FAILED

308 (0x134)

The system fails to measure frequency.

ERR_PARAM_INVALID

309 (0x135)

The parameter is invalid.

ERR_FIFO_NOTREADY

310 (0x136)

The internal fifo is not ready.

ERR_FIFO_FULL

311 (0x137)

The internal fifo is full.

ERR_FIFO_DATAFAILED

312 (0x138)

The fifo data is failed.

ERR_ACQSTOP_FAILED

313 (0x139)

The system fails to stop data acquisition.

ERR_FREQ_INVALID

314 (0x13A)
The channel frequency is invalid.
To set configuration for PWM modules, the channel frequency is ranged from 1 to 30K (Hz).

ERR_DUTY_INVALID
315 (0x13B)
The channel duty cycle is invalid.
To set configuration for PWM modules, the channel duty cycle is ranged from 0.1 to 99.9 %.

ERR_DEVICE_NON
320 (0x140)
The module didn't exist.

ERR_ACCESS_DENIED
321 (0x141)
Access is denied.

ERR_LENGTH_INVALID
322 (0x142)
The length of input data exceeds the maximum allowed.

ERR_CONFIG_FAILED
323 (0x143)
The system configuration is failed.

ERR_DSPFLAG_INVALID
324 (0x144)
The DSP flag is invalid.

ERR_INTERNAL_FAILED
325 (0x145)
The internal file is failed

ERR_TIMEOUT
326 (0x146)
The processing time exceeds the maximum allowed.

ERR_COMMAND_FAILED

390 (0x186)

The system command is failed.

Appendix C

C1. Analog I/O Board Settings

Range Settings for Analog I/O Boards. These ranges are provided for reference. Not all boards support all ranges. Please see hardware manual for valid ranges for a particular board.

	Setting Type	Value (Hex)
Millivolts DC	+/- 15mV	0x0100
(mV)	+/- 50mV	0x0101
	+/- 100mV	0x0102
	+/- 150mV	0x0103
	+/- 500mV	0x0104
	0~150mV	0x0105
	0~500mV	0x0106
Volts DC	+/- 1V	0x0140
(V)	+/- 2.5V	0x0141
	+/- 5V	0x0142
	+/- 10V	0x0143
	+/- 15V	0x0144
	0~1V	0x0145
	0~2.5V	0x0146
	0~5V	0x0147
	0~10V	0x0148
	0~15V	0x0149
Milliamps (mA)	4~20mA	0x0180
	+/-20mA	0x0181
	0~20mA	0x0182
Counter settings	Pulse/DIR	0x01C0

	Up/Down	0x01C1
	Up	0x01C2
	Frequency	0x01C3
	AB 1X	0x01C4
	AB 2X	0x01C5
	AB 4X	0x01C6
Pt-100 (3851)	Pt-100 (3851) -200~850 'C	0x0200
	Pt-100 (3851) -120~130 'C	0x0201
	Pt-100 (3851) -200~200 'C	0x0202
	Pt-100 (3851) -100~100 'C	0x0203
	Pt-100 (3851) -50~150 'C	0x0204
	Pt-100 (3851) 0~100 'C	0x0205
	Pt-100 (3851) 0~200 'C	0x0206
	Pt-100 (3851) 0~400 'C	0x0207
	Pt-100 (3851) 0~600 'C	0x0208
Pt-200 (3851)	Pt-200 (3851) -200~850 'C	0x0220
	Pt-200 (3851) -120~130 'C	0x0221
Pt-500 (3851)	Pt-500 (3851) -200~850 'C	0x0240
	Pt-500 (3851) -120~130 'C	0x0241
Pt-1000 (3851)	Pt-1000 (3851) -200~850 'C	0x0260
	Pt-1000 (3851) -120~130 'C	0x0261
	Pt-1000 (3851) -40~160 'C	0x0262
Pt-100 (3916)	Pt-100 (3916) -200~850 'C	0x0280
	Pt-100 (3916) -120~130 'C	0x0281
	Pt-100 (3916) -200~200 'C	0x0282
	Pt-100 (3916) -100~100 'C	0x0283
	Pt-100 (3916) -50~150 'C	0x0284
	Pt-100 (3916) 0~100 'C	0x0285
	Pt-100 (3916) 0~200 'C	0x0286
	Pt-100 (3916) 0~400 'C	0x0287
	Pt-100 (3916) 0~600 'C	0x0288
Pt-200 (3916)	Pt-200 (3916) -200~850 'C	0x02A0
	Pt-200 (3916) -120~130 'C	0x02A1
Pt-500 (3916)	Pt-500 (3916) -200~850 'C	0x02C0
	Pt-500 (3916) -120~130 'C	0x02C1
Pt-1000 (3916)	Pt-1000 (3916) -200~850 'C	0x02E0

	Pt-1000 (3916) -120~130 'C	0x02E1
	Pt-1000 (3916) -40~160 'C	0x02E2
Balco 500	Balcon(500) -30~120	0x0300
Ni 518	Ni(518) -80~100 'C	0x0320
	Ni(518) 0~100 'C	0x0321
Ni 508	Ni(508) 0~100 'C	0x0340
	Ni(508) -50~200 'C	0x0341
Thermistor 3K	Thermistor 3K 0~100 'C	0x0360
Thermistor 10K	Thermistor 10K 0~100 'C	0x0380
	Thermistor 10K -50~100 'C	0x0381
T/C TypeJ	T/C TypeJ 0~760 'C	0x0400
	T/C TypeJ -200~1200 'C	0x0401
T/C TypeK	T/C TypeK 0~1370 'C	0x0420
	T/C TypeK -270~1372 'C	0x0421
T/C TypeT	T/C TypeT -100~400 'C	0x0440
	T/C TypeT -270~400 'C	0x0441
T/C TypeE	T/C TypeE 0~1000 'C	0x0460
	T/C TypeE -270~1000 'C	0x0461
T/C TypeR	T/C TypeR 500~1750 'C	0x0480
	T/C TypeR 0~1768	0x0481
T/C TypeS	T/C TypeS 500~1750 'C	0x04A0
	T/C TypeS 0~1768 'C	0x04A1
T/C TypeB	T/C TypeB 500~1800 'C	0x04C0
	T/C TypeB 300~1820 'C	0x04C1