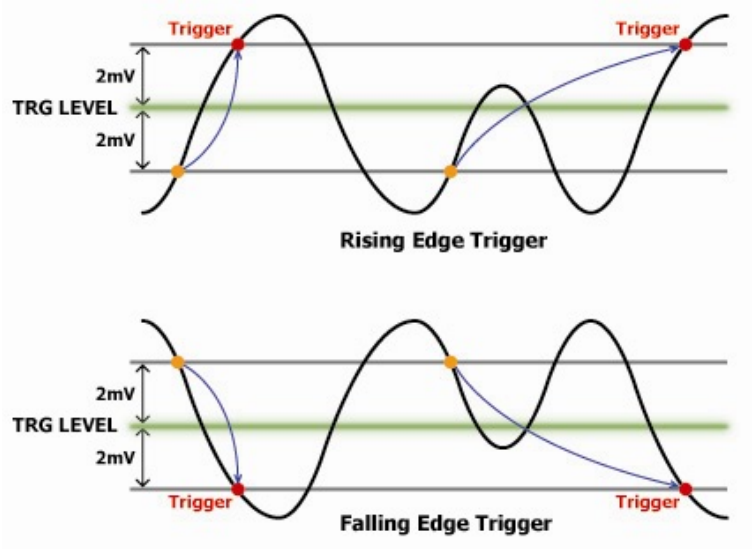
signal reaches 5V which is the trigger level after the trigger generating mode sets in, no trigger happens before it gets over 4.998V.



Conditon of trigger generation that uses the external analogue input

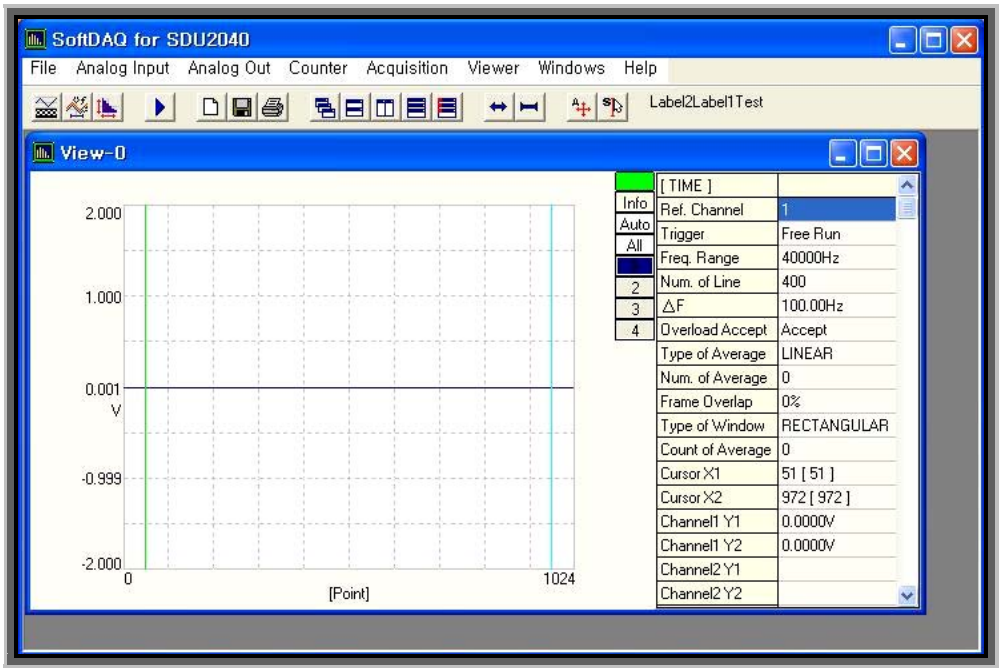
13. How to Use the Software

(1) Basic Measuring Function

① Cliick the execution icon to execute SoftDAQ.

② SoftDAQ program automatically cheks the internal state of SDU 2040 and the connection to computer via USB, and reads the initialization data.

③ The Fig. below shows the initial screen when the SoftDAQ is run.



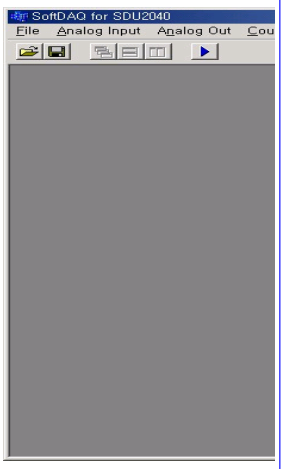
삭제됨: 14. 소프트웨어 사용 방법

1. 간단한 측정기능

1. 실행 아이콘을 클릭하여 softDAQ를 실행합니다.

2. softDAQ 프로그램은 자동으로 SDU 2040의 내부상태 및 USB를 통한 컴퓨터와의 연결상태를 점검하고, 초기화 데이터를 읽어 들이게 됩니다.

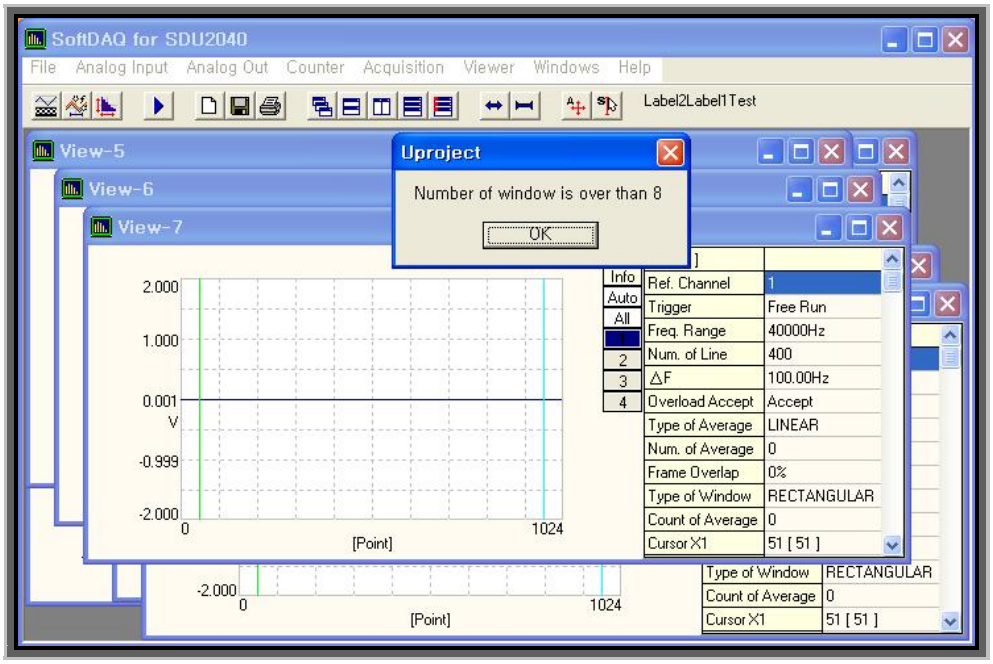
3. SoftDAQ가 실행된 초기 화면은 다음과 같습니다.



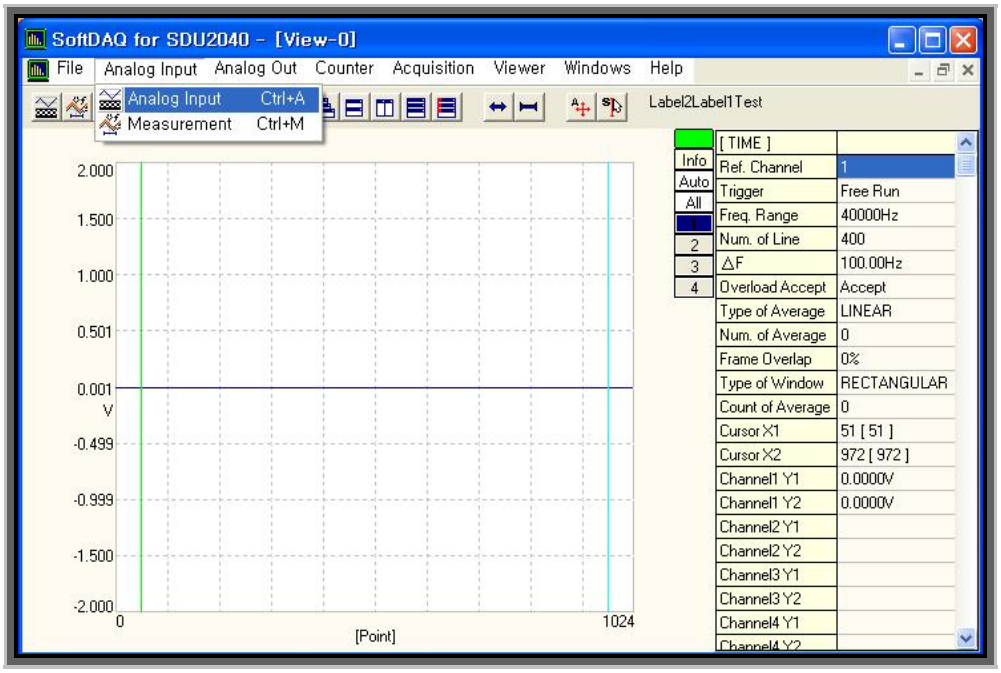
4. Acquisition => Open Viewer 를 클릭하여 출력화면을 활성화 시킵니다.

… [1]

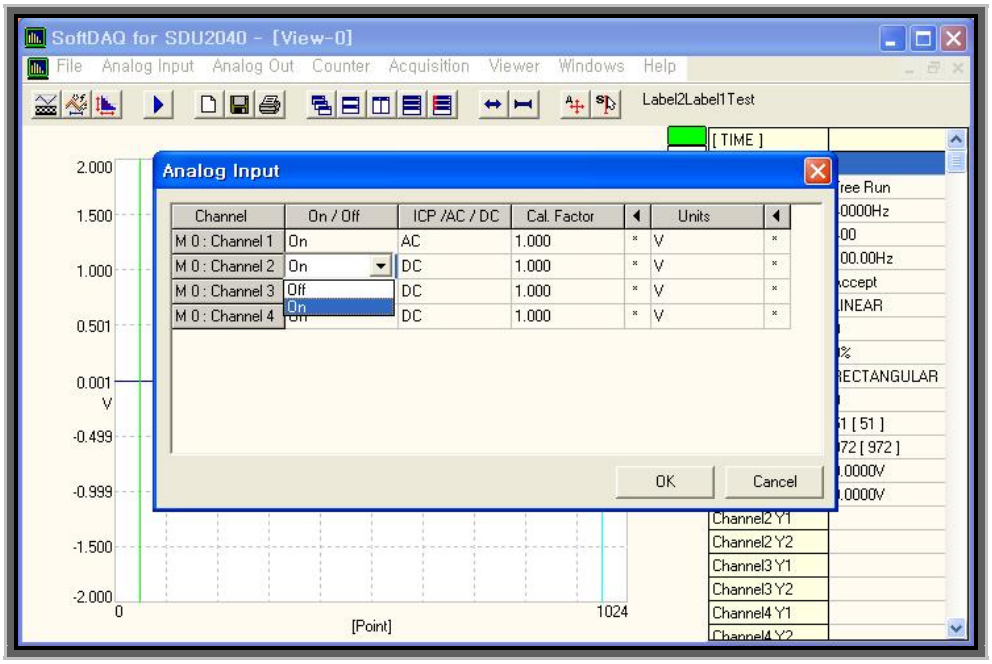
④ The basic window provides a screen. If you want more windows, click the New button on the 5th row in the menu window. Then, you can get up to 8 windows that you want.



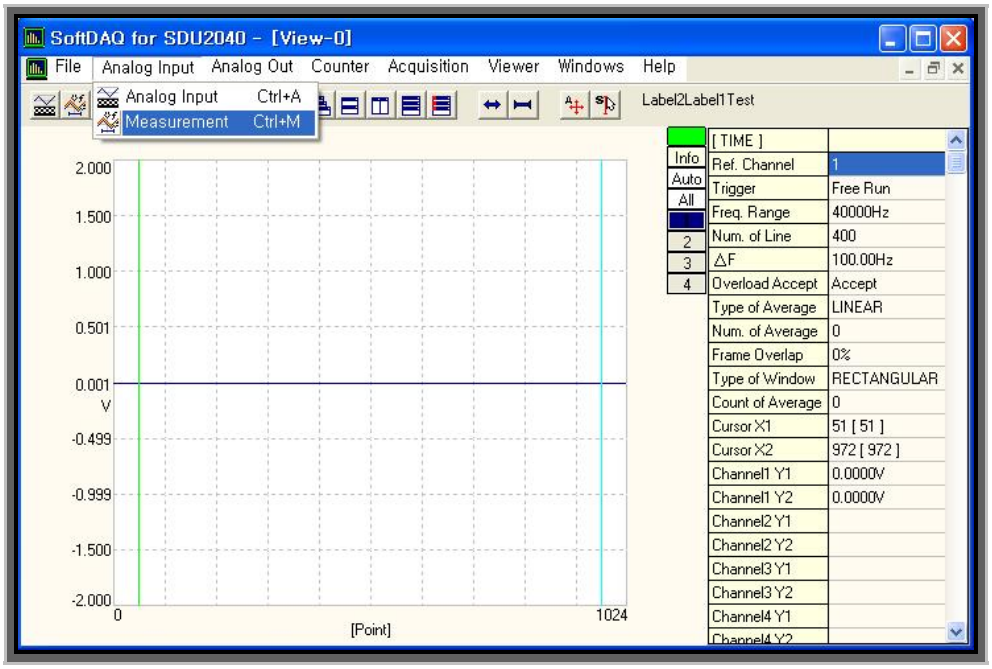
⑤ Activate the output screen for the Analog Input . Then, choose the Analog Input.



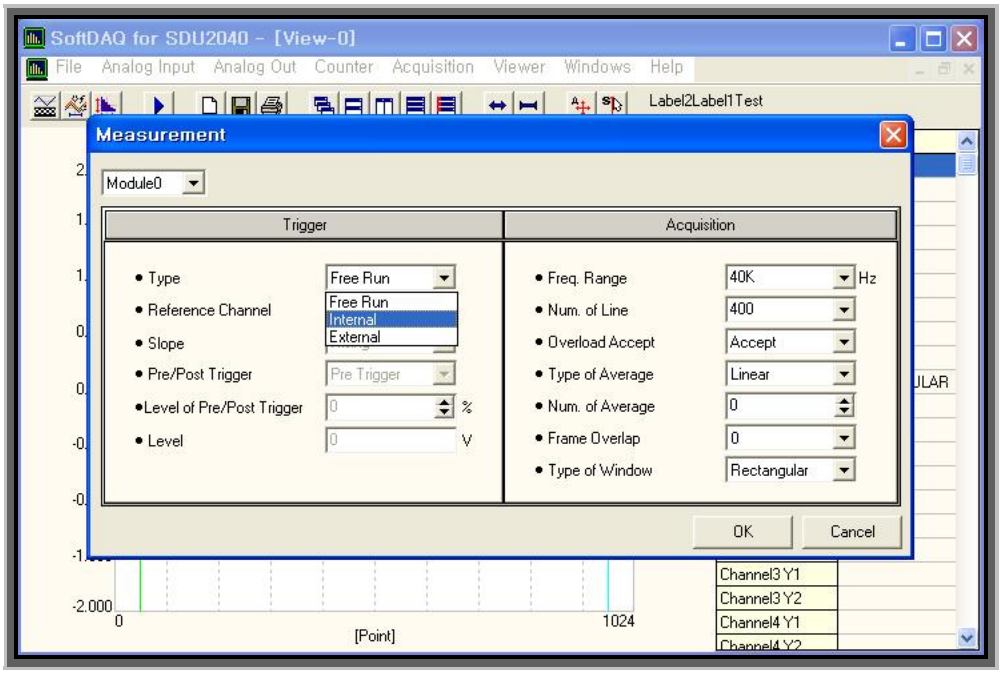
⑥ Activate the corresponding channel in the analogue input, and set the condition that meets the input condition.



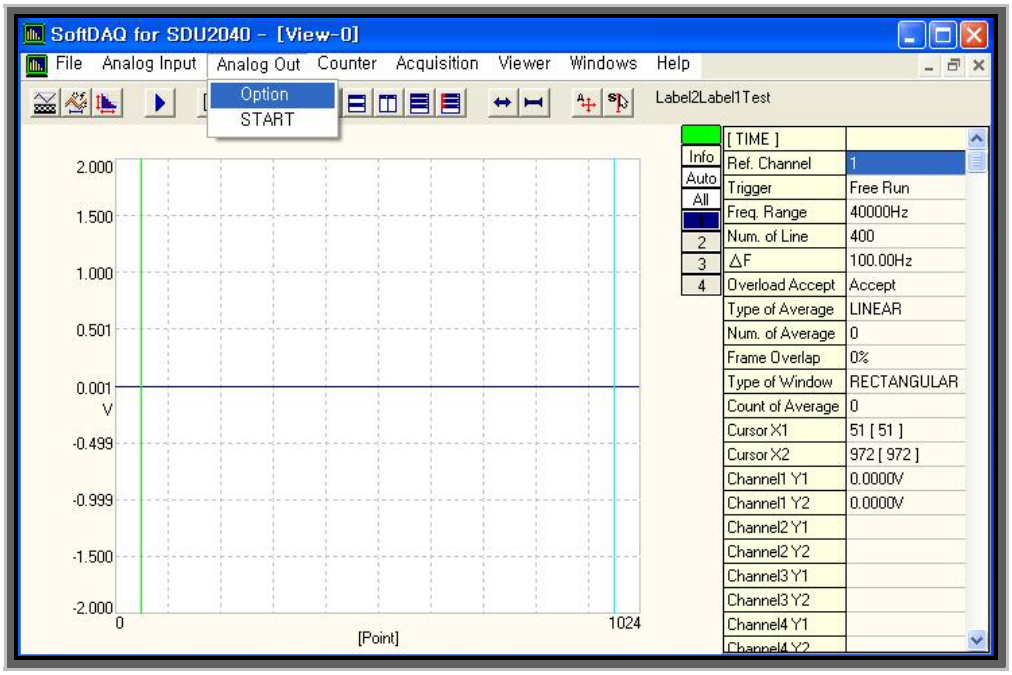
⑦ Activate the output screen for the Analog input. And then choose the Measurement.



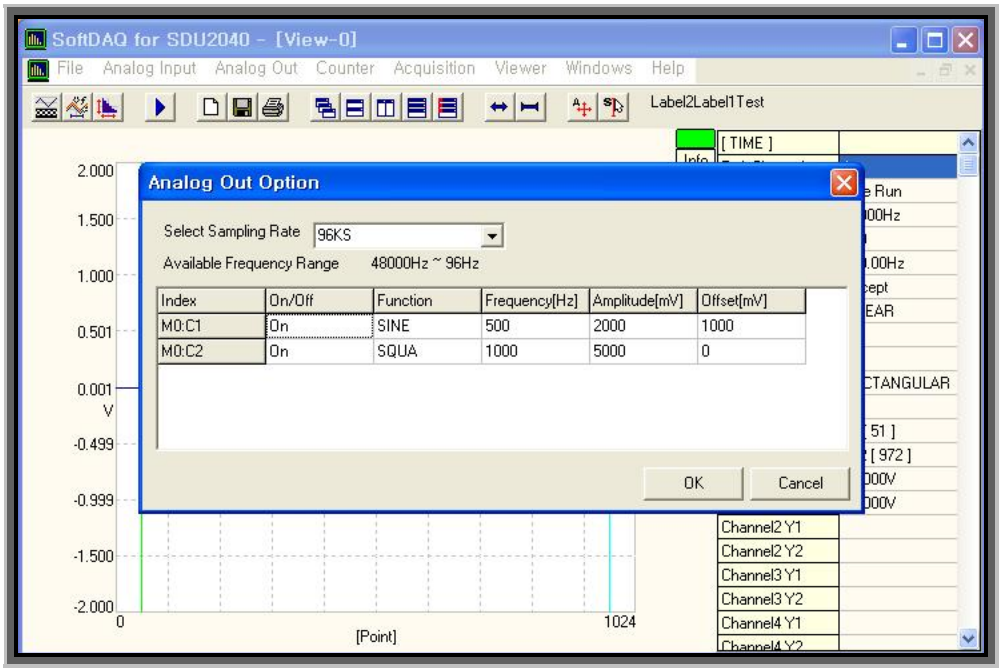
⑧ Activate the corresponding window in the Analog Input set the condition that meets the input condition.



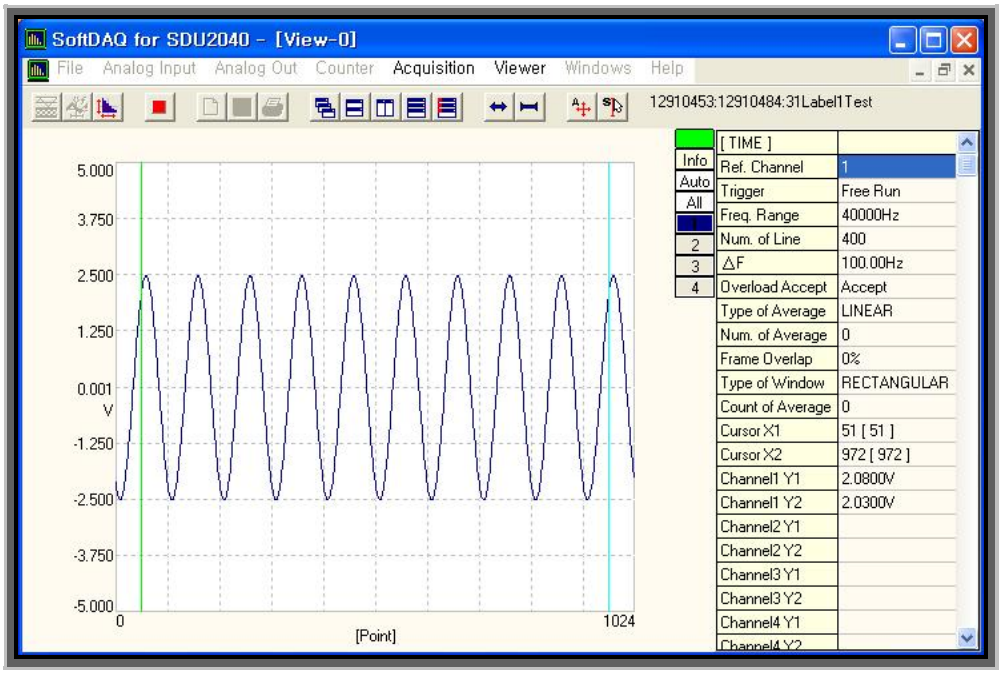
⑨ Activate the output screen for the Analog input. And then choose the Option.



⑩ Activate the Analog out in the corresponding window and set the condition that meets the input condition.

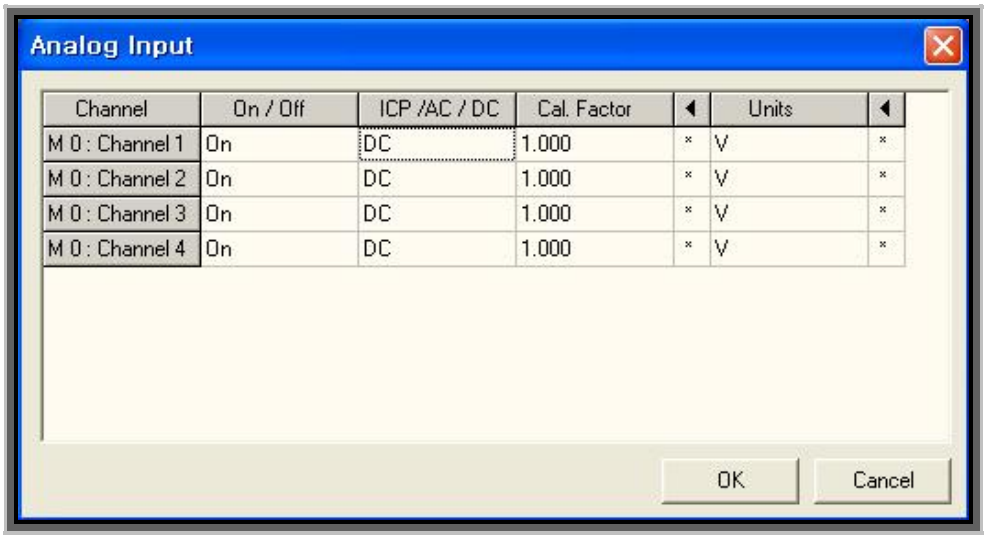


⑪ Execution Screen



2. Explanation on the Detailed Function

▪ Analog input



▪ On/Off

Each channel can be activated, and if each channel is set as Off, the corresponding channel does not output the waveform because each channel does not collect the data.

▪ AC/DC/ ICP

Coupling, DC Coupling can be chosen as the item which can select the input coupling type. The interception frequency of AC coupling is 3.5Hz. The power supply can be chosen to drive the IPC type sensor. If ICP is chosen, DC 20V, 4mA electricity is output in the corresponding channel. Therefore, ICP type sensor can be used without external powe. If ICP On is chosen, it converts to AC coupling automatically.

▪ Cal. Facter

It is the method to change the unit to M, K, m, u on the y axis for use.

▪ Units

The unit can be set for the output value that suits several input type,by changing the unit of data value that is output at the user's discretion.

삭제됨: Scale

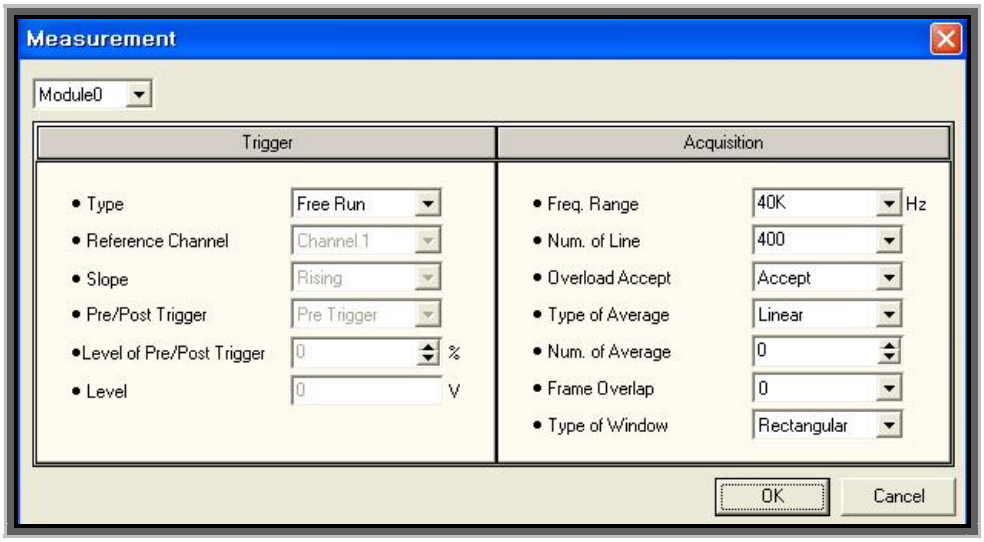
삭제됨: 출력 화면의 스케일 설정이 가능하며 외부에 감쇄기를 통해 아날로그 입력을 받았을 경우 스케일 조절을 통해 올바른 출력값을 얻어낼 수 있습니다.

▪ Offset

출력되는 파형의 Offset값 조절이 가능하여 필요시 사용자가 임의의 Offset값을 부여하여 파형 분석이 가능합니다.

삭제됨:

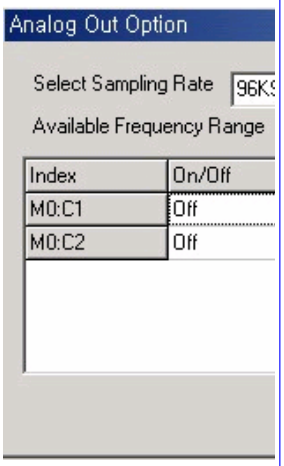
▪ Measurement



삭제됨:

● Analog Output Option창

Analog Output => Option창을 선택하면 다음 그림과 같이 Analog Out Option창이 활성화 됩니다.



▪ Select Sampling Rate

Analog Output의 출력 샘플링 속도를 결정하는 항목으로 최소 8KS/sec에서 최대 96KS/sec까지 속도 조절이 가능합니다.

1. Trigger

(1) Free run, internal and external types are provided. Free run is to output the constant inflow of the input waveform, and cannot choose others for the trigger mode. The internal mode enables the use of all functions of trigger. The external has the same internal as the mode.

(2) Reference Channel is the place to choose the desired location among the 4 channels for the setting.

(3) The slope has the trigger condition whwen the signal converts from the low level to high level which is input from the rising terminal. The trigger level happens when the signal shifts from the high level to the low level which is input from the falling terminal, and the trigger level is based on the TTL level.

(4) Pre/Post Trigger is provided with the Pre trigger, Post Trigger.

(5) Level of Pre/Post Trigger is the function to adjust and use the Pre trigger, Post Trigger 의 Level value.

(6) It starts collectigng the waveform following the level value as much as the figure input into the level.

2. Acquisition

▪ Freq. Range

It adjusts the desired width and can be set from 10 to 80Kz.

▪ Num. of Line

It inputs the coefficient of X axis, and if a lot of numbers are input, signal can be brought to the broad section. Reduce the coefficient and measure if you want to see the waveform.

▪ Overload Accept

It is the function to inquire if the frame overload is to be used.

▪ Type of Average

3 modes like Linear, Exponential and peak Hold are provided. You can chooe what you want and use. If use put in the number into the Nom. Of Average while using the Liner and Peak Hold, the data collection comes to a halt after computing the average by the input number.

▪ Num. of Average

Chose and input the average that you want. You can use this with the Type of Average above.

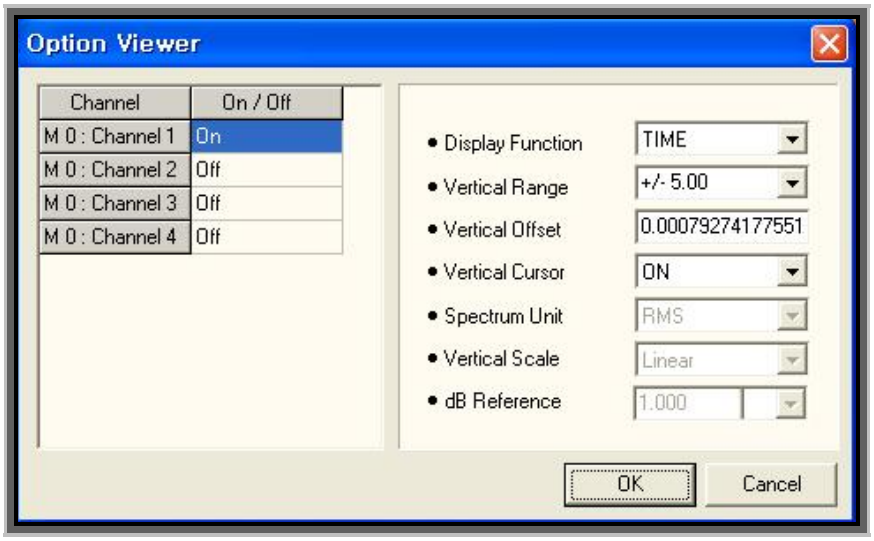
▪ Frame Overlap

You can adjust a 6 stages like 0, 25, 50, 66.7, 75, MAX.

▪ Type of Window

It provides 5 filters like Rectangular, Hanning, Hamming, Blackman, Flat-top.

3. Option Viewer



▪ On/Off

If it is set as ON and each channel is set as activation OFF, each channel is in the non-activation state. Therefore, corresponding channel does not output the waveform.

▪ Display Function

TIME is to analyze the current waveform in the time section. APS means the Auto Power Spectrum. FFT is a powerful tool to analyze the components of current waveform. Thouth the osciloscope anlayzes the waveform in the time section, it is used whwen the waveform needs to be analyzed in the frequency section.

▪ Vertical Range

It sets the range of vertical axis. If small signal comes in, you can fit the range of vertical axis for the measurement.

▪ Vertical Offset

One method is to use the one which has the offet determined, and the other method is to input number at discretion for use.

▪ Vertical Cursor

It is the method to switch on/off the cursor on the vertical axis.

▪ Spectrum Unit

Change each unit for RMS, PWR, PSD for use.

▪ Vertical Scale

It is the part to choose the unit on the vertical axis. 3 units like Linear, dB and log are provided.

삭제됨: -------

삭제됨: 사용자가 원하는 출력파형 형태를 선택 할 수 있으며 사인파형, 구형파형 등 여러 형태의 출력파형 선택이 가능합니다.

▪ **Frequency**

사용자가 선택한 파형의 주파수를 변화 시킬 수 있으며 해당 주파수의 선택 범위는 최소 96Hz에서 1kHz까지 가능합니다.

▪ **Amplitude**

사용자가 선택한 파형의 출력 전압을 가변 시킬 수 있으며 출력전압의 범위는 최소 1V에서 20V까지 가능합니다.

▪

삭제됨: 출력 파형과 함께 Offset을 선택하여

▪ dB Reference

This is whre the desired value can be set when dB is used. 4 units like M, K, m and u can be used.

▪ Run/Stop

This is where the sampling is made to start and stop.

▪ Windows

It indicates the method of arranging the display currently provided. 5 units are provided.

The difference between 2 and 3 in Cascade, Tile Horizontally Tile vertically Tile Horizontally2 Tile Horizontally3 is that 2 indicates only the arrangement and 3 enables the arrangement and sequence.

삭제됨:

▪ Sampling Rate

Sampling Rate는 최소 1S/sec 부터 최대 216KS/sec까지 가능합니다.

샘플링 속도가 증가 할수록 데이터 저장량이 많이 때문에 신호를 분석하는데 시간이 더 소요됩니다. 원 신호를 분석하기 위한 적당한 샘플링 속도를 선택하는 것이 효과적인 데이터 수집방법 입니다.

▪ Sampling Number

Sampling Number는 각 채널의 출력화면에 크기(샘플링 횟수)를 나타내며 Sampling Number가 클수록 더 많은 데이터를 화면에 출력해야 하기 때문에 시간이 더 소요됩니다.

Continuous Mode에서는 사용자가 샘플링 속도를 지정하지 않고 Continuous Mode 선택시 자동으로 Sampling Number가 정해지기 때문에 Continuous Mode는 별도로 Sampling Number를 설정 할 필요가 없습니다.

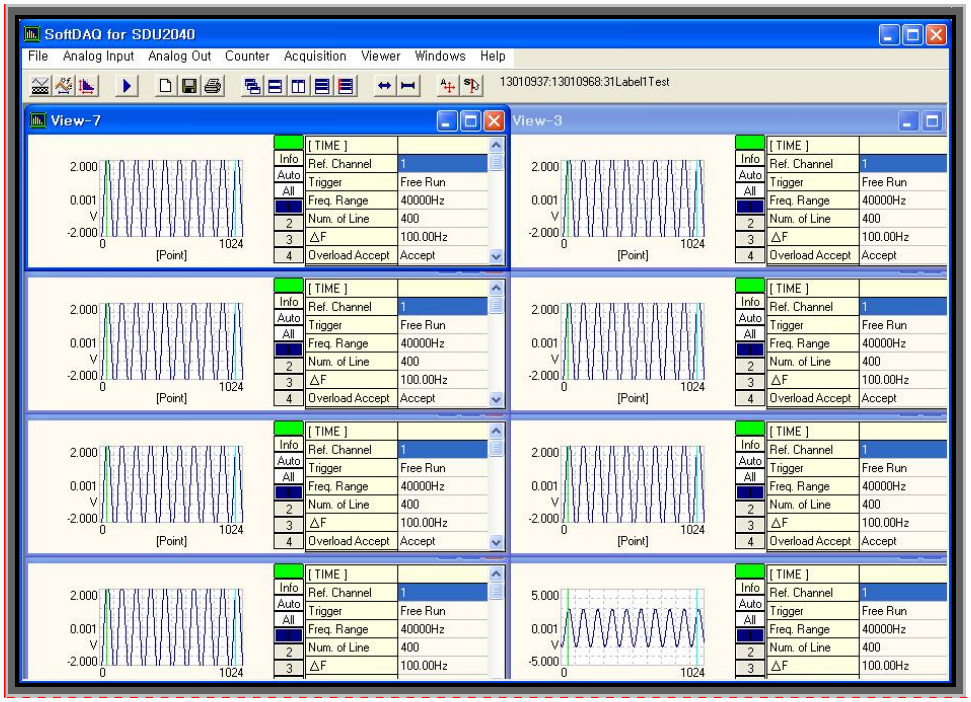
● Trigger Option

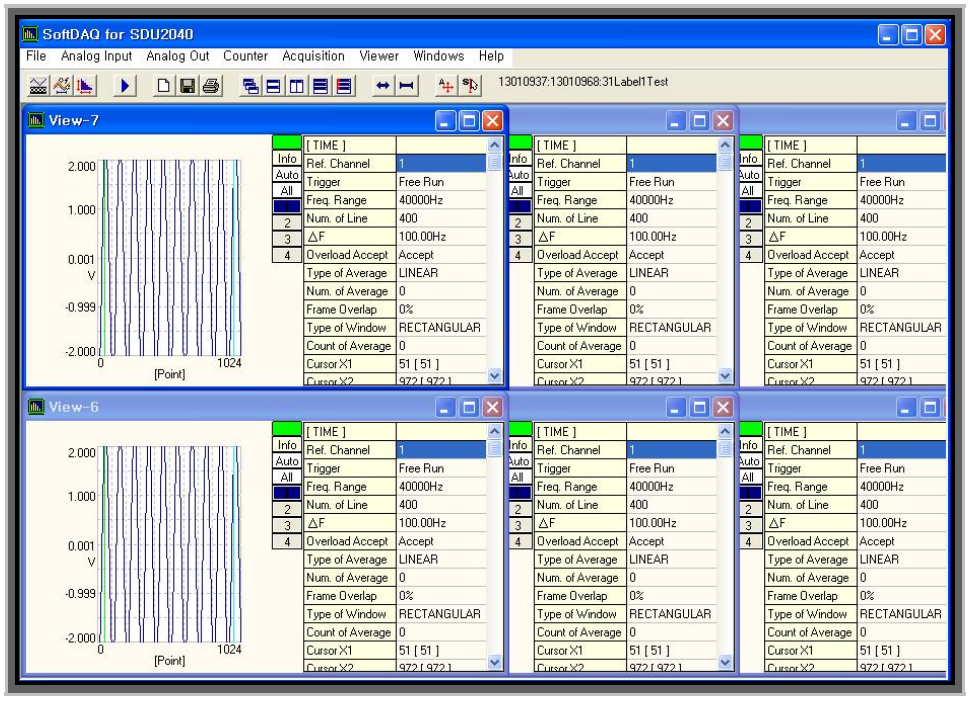
Acquisition => Trigger Option 창을 선택하면 다음과 같이

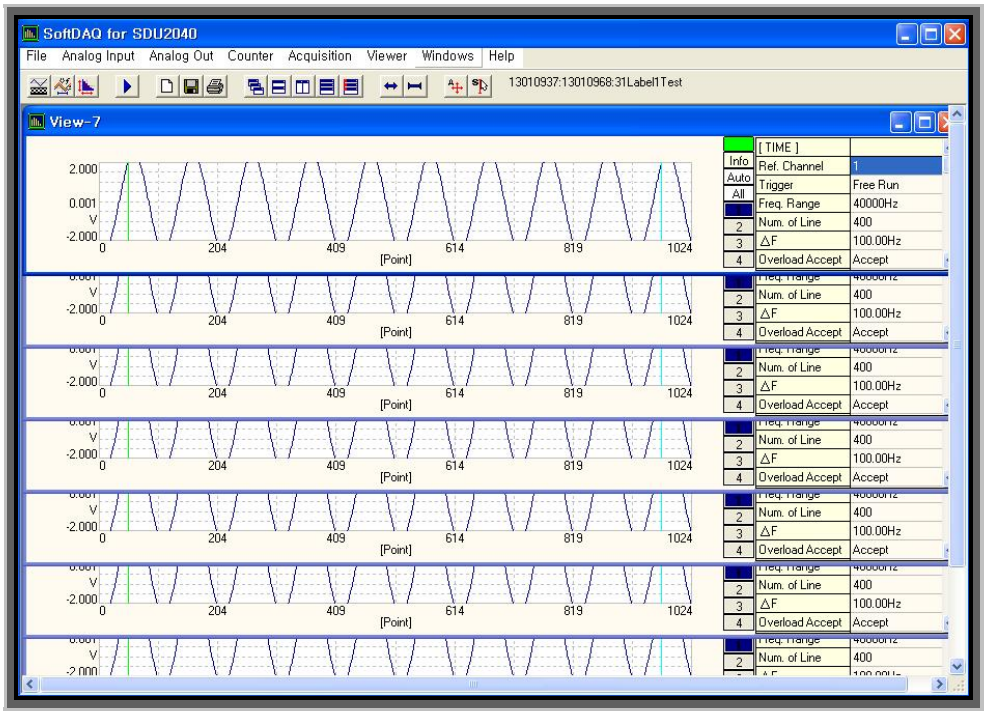
… [2]

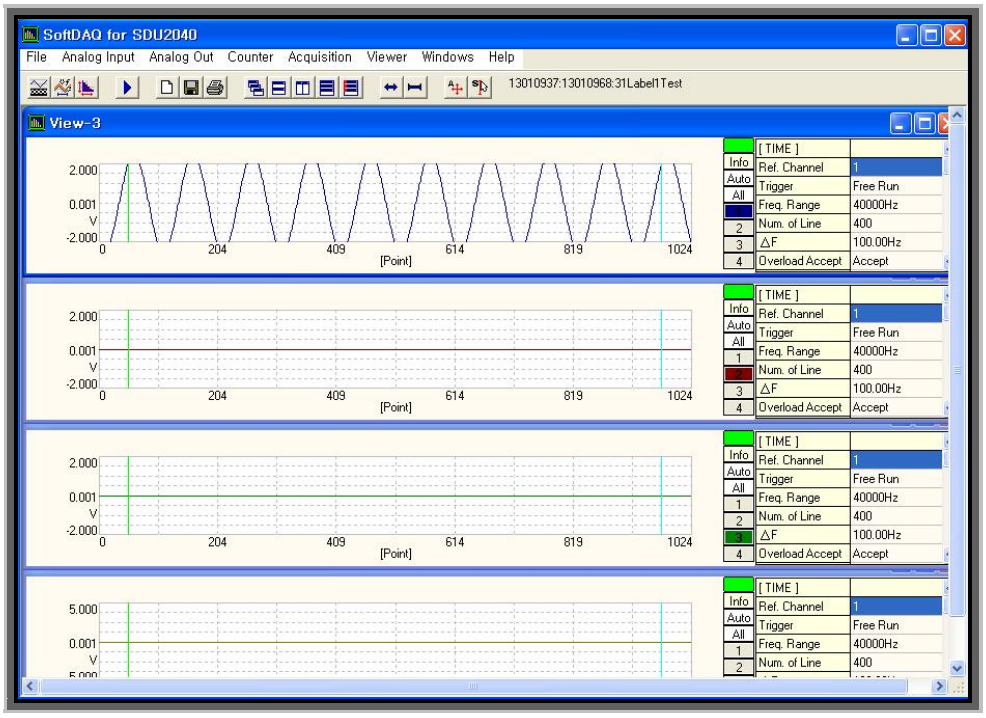
▪ **Type of Display**

메모 [C1]:






SoftDAQ for SDU2040
File Analog Input Analog Out Counter Acquisition Viewer Windows Help
13010937:13010968:31Label1Test
View-7
[TIME]
Ref. Channel
Trigger
Free Run
Freq. Range
40000Hz
Num. of Line
400
ΔF
100.00Hz
Overload Accept
Accept
[Point]


SoftDAQ for SDU2040
File Analog Input Analog Out Counter Acquisition Viewer Windows Help
13010937:13010968:31Label1Test
View-3
[TIME]
Ref. Channel
Trigger
Free Run
Freq. Range
40000Hz
Num. of Line
400
ΔF
100.00Hz
Overload Accept
Accept
[Point]

▪ **Extend Horizontally & reduce Horizontally**

After the vertical axis is extended by the range that you want and reduced, the original signal can be output. You can ajust and check, using the mouse.

▪ **Auto Scale all & Synchronize cursor**

2 types of method are being provided, which make the line on the left and right side on the screen shift to the same location as on other screen.

Other Function

▪ **Counter**

It enables the counter function for the pulse signal of the external TTL level and allows the counted result to be used for the trigger source for the analogue input.

▪ **Screen**

What you see on the right side in case of the screen output indicates the value that the user set. And there are Info, Auto, All, 1,2,3,4 on the side for your selection. If you click the Info, the informatioin window disappears. In case of using the Auto, it makes the waveform convenient to see automatically. All plays the role of switching on all channels of 1, 2, 3 and 4. If you want to delete the waveform that you do not want to see on the screen, double click the number.

You can see  on the upper left side on the screen. This shows the overflow, and if signal higher than the hardware recognizes, it changes to  .

SDU2040 has the phenomenon described in the above in case of over DC +/-10V.

삭제됨: Level

동기화 외부 카운터 숫자에 아날로그 파형 입력을 동기화 시킬 수 있으며 Level에 입력한 수치 만큼의 카운터 후에 파형 수집을 시작합니다.

▪ **Analog**

Rising Mode

트리거 조건이 설정된 해당 채널의 아날로그 입력이 Level전압기준으로 낮은 전압에서 높은 전압으로 변할 때 트리거 조건을 발생시켜 파형을 동기화 합니다.

Falling Mode

트리거 조건이 설정된 해당 채널의 아날로그 입력이 Level전압기준으로 높은 전압에서 낮은 전압으로 변할 때 트리거 조건을 발생시켜 파형을 동기화 합니다.

Level

트리거 조건이 설정된 해당 채널의 트리거 Level을 사용자가 원하는 전압으로 설정이 가능하며 단위는 mV 단위입니다.

Channel

사용자가 트리거 조건을 4 채널 중 특정 채널을 선택

… 3

14. API Function Reference

(1) sdDaqInitialize

- **Description:**

  It searches and initialize SDU 2040, and sets to make the initial communication possible.

- **Format :**

  C++

```
SD_DAQ_ERROR sdDaqInitialize(
    int* deviceNum_
);
```

  Visual Basic

```
sdDaqInitialize (ByVal deviceNum_ As Long) As Long
```

- **Parameters:**

Parameter	Value	Mean
deviceNum_	0 ~ MAX_DAQ	It is the variable that gets back the SDU 2040 device which is currently connected.

- **Usage :**

  It figures out the number of SDU 2040 device that is currently connected. If there is no device, it outputs the Error and finishes.

C++

```
SD_DAQ_ERROR res;
int m_deviceCount;
res = sdDaqInitialize(&m_deviceCount);
if (res != _ERROR_NONE || m_deviceCount <= 0) {
    printf("Error : Initialize");
}
```

Visual Basic

```
Dim Res   As Long
Dim deviceNum_   As Long
Res = sdDaqInitialize(deviceNum_)
If (Res = 0) Then
    MsgBox "Error : Initialize "
    End
End If
```

Returns :

Success : _ERROR_NONE

Failure : _ERROR_DEVICE_NOT_FOUND

- Comments:

  This is the commanding lanugate that always starts at the first time, and subsequent commanding language can be used only after this commanding language is successful.

(2) sdDaqFinalize

- Description:

  It revokes the communication connected to the SDU 2040 that is currently connected and returns various memories.

- Format :

C++

```
SD_DAQ_ERROR sdDaqFinalize();
```

Visual Basic

```
sdDaqFinalize () As Long
```

- Parameters:

  NONE

- Usage :

```
SD_DAQ_ERROR res;
res = sdDaqFinalize ();
if (res != _ERROR_NONE ) {
    printf("Error : Finalize");
}
```

Visual Basic

```
Dim Res   As Long
Dim deviceNum_   As Long
Res = sdDaqFinalize ()
If (Res < = 0) Then
    MsgBox "Error : Finalize"
    End
End If
```

- Returns :

  Sucdess : _ERROR_NONE

  Failulure : _ERROR_USB_DRIVER

- Comments:

  This is the commanding lanugate that is always called last, and all SDU work can be performed through this commanding.

(3) sdDaqOpenDevice

- Description:

  It opens the device that is currently connected. It should be excecuted only after the sdDaqInitialize function is executed. This function should be called for every device that is to be opened.

- Format :

  C++

```
SD_DAQ_ERROR sdDaqOpenDevice(
        int indexModule_
);
```

  Visual Basic

```
sdDaqOpenDevice (ByVal indexModule_ As Long) As Long
```

- Parameters:

Parameter	Value	Mean
indexModule_	0 ~ MAX_DAQ	The device number of SDU 2040 that is currently connected SeeIt can be set in the deviceNum obtained through the sdDaqInitialize function.

- Usage :

  C++

```
SD_DAQ_ERROR res;
int m_deviceCount;
res = sdDaqInitialize(&m_deviceCount);
if (res != _ERROR_NONE || m_deviceCount <= 0) {
    printf("Error : Initialize");
}
for (int i = 0; i < m_deviceCount; i++) {
    res = sdDaqOpenDevice(i);
    if (res != _ERROR_NONE) {
        printf("Error : Open device");
    }
}
```

Visual Basic

```
Dim G, G2 As Long
Dim deviceNum As Long
    For G = 0 To deviceNum_ - 1
        Res = sdDaqOpenDevice(G)
        If (Res <= 0) Then
            MsgBox "Error : Open Device "
            End
        End If
    Next G
```

- Returns :

  In case of success : _ERROR_NONE

  In case of failure : See the error type of "usbDaqErrorType.h"

- Comments:

  The unique number of the device after the success of this function is indexModule_number.

(4) sdDaqCloseDevice

- Description:

  It closes the device that is currently connected. It shold be executed before the sdDaqFinalize function is called all the time.

- Format :

  C++

```
SD_DAQ_ERROR sdDaqCloseDevice(
        int indexModule_
);
```

  Visual Basic

```
sdDaqCloseDevice (ByVal indexModule_ As Long) As Long
```

- Parameters:

Parameter	Value	Mean
indexModule_	0 ~ MAX_DAQ	SDU 2040 device number

- Usage :

  C++

```
SD_DAQ_ERROR res;
for (int i = 0; i < m_deviceCount; i++) {
    res = sdDaqCloseDevice(i);
    if (res != _ERROR_NONE) {
        printf("Error : Close device");
    }
}
```

  Visual Basic

```
Dim Res   As Long
Dim indexModule_ As Long
    Res = sdDaqCloseDevice(indexModule_)
    If (Res = 0) Then
        MsgBox "Error : Close device "
        End
    End If
```

- **Returns :**

  In case of success : _ERROR_NONE

  In case of failure : See the error type of "usbDaqErrorType.h" .

- **Comments:**

  sdDaqCloseDevice must be called for the successful device through the sdDaqOpenDevice function.

(5) sdDaqAdcChannelConfig

- Description:

  It sets the Analog Input channel of device that is currently connected.

- Format :

  C++

```
SD_DAQ_ERROR sdDaqAdcChannelConfig(
    SD_CHANNEL channel_,
        SD_ON_OFF channelOnOff_,
        float scale_,
    float offset_,
    char* units_,
    SD_ADC_COUPLING coupling_,
    SD_ON_OFF icpOnOff_,
    int indexModule_
);
```

  Visual Basic

```
sdDaqAdcChannelConfig (ByVal channel_ As Long,
                       ByVal channelOnOff_ As Long,
                       ByVal scale_ As Single,
                       ByVal offset_ As Single,
                       ByRef units_ As String,
                       ByVal coupling_ As Long,
                       ByVal icpOnOff_ As Long,
                       ByVal indexModule_ As Long) As Long
```

- **Parameters:**

Parameter	Value	Mean
channel_	_CHANNEL_1 or 0	Channel 1
	_CHANNEL_2 or 1	Channel 2
	_CHANNEL_3 or 2	Channel 3
	_CHANNEL_4 or 3	Channel 4
channelOnOff_	_ OFF or 0	Channel Off
	_ ON or 1	Channel On
scale_	Float	Output by multiplying the original data value of selected data by the scale_value.
offset_	Float	Output by multiplying the original data value of selected data by the offset_value.
units_	Char	The unit that the user wants can be expressed through the 15 characters. This is indicted by the unit when the data is stored and retrived.(이것은 데이터 저장 및 불러오기 시에(?) 단위로 표시 됩니다.)
coupling_	_AC or 0	AC coupling
	_DC or 1	DC coupling
icpOnOff_	_ OFF or 0	ICP Off
	_ ON or 1	If ICP is switched on. The coupling changes to AC coupling internally unconditionally.
indexModule_	0 ~ MAX_DAQ	SDU 2040 device number

- **Usage :**

  It swithes on the Analog Input Channel 1 of Module Number 0, and brings the scale to the original size through DC. It sets the offset as 0 and units as "V".

  C++

```
SD_DAQ_ERROR res;
res=sdDaqAdcChannelConfig(_CHANNEL_1, _ON, 1.0, 0.0, "V", _DC, _OFF, 0);
```

```
if (res != _ERROR_NONE) {
    printf("Error : Analog Input Channel Configuration");
}
```

Visual Basic

```
Dim Res  As Long
Res = sdDaqAdcChannelConfig(SD_CHANNEL_1, SD_ON, 1#, 0#, "V", SD_DC,
      SD_OFF, 0)
If (Res = 0) Then
    MsgBox "Error : Analog Input Channel Configuration"
    End
End If
```

- Returns :

  In case of success : _ERROR_NONE

  In case of failure : See the error type of "usbDaqErrorType.h" .

- Comments:

(6) sdDaqAdcSamplingConfig

- Description:

  It sets the Sampling Rate, Sampling Size, Sampling Mode of the device that is currently set.

- Format :

  C++

```
double sdDaqAdcSamplingConfig(
    double samplingRate_,
    int samplingNumber_,
    SD_SAMPLING_MODE samplingMode_,
    SD_TRIGGER_MODE trigSamplingmode_,
    SD_ON_OFF highPassFilter_,
    int indexModule_
);
```

  Visual Basic

```
sdDaqAdcSamplingConfig (ByVal samplingRate_ As Double,
                        ByVal samplingNumber_ As Long,
                        ByVal samplingMode_ As Long,
                        ByVal trigSamplingmode_ As Long,
                        ByVal highPassFilter_ As Long,
                        ByVal indexModule_ As Long)  As Long
```

- **Parameters:**

Parameter	Value	Mean
samplingRate_	Double (216000, 204800, 190000, 108000, 102400, 96000, 54000, 51200, 48000, 36000, 34133, 32000, 27000, 25600, 24000, 21600, 20480, 19200, 18000, 17066, 16000, 15429, 14628, 13714, 13500, 12800, 12000, 11377, 10800, 10666, 10240, 9818, 9600, 9309, 9000, 8727, 8533, 8308, 8000, 4000, 2000, 1000, 500, 100, 50, 10, 1)	Sampling Rate The input sampling rate is the most approximate value and is set as one of the value.
samplingNumber_	Integer (1 ~)	Sampling Number
samplingMode_	_SAMPLING_CONTINUOUS_MODE or 0	It obtains the data consecutively.
	_SAMPLING_N_SAMPLE_MODE or 1	It obtains the data by N.
	_SAMPLING_TRIGGER_MODE or 2	If the trigger happens, the data is obtained according to the trigger sampling mode.
trigSamplingmode_	_TRIG_CONTINUOUS_MODE or 0	It the trigger happens, the data is obtained from that point consecutively.
	_TRIG_SINGLE_N_SAMPLING_MODE or 1	If the trigger happens, the data is obtained,
	_TRIG_EVERY_N_SAMPLING_MODE or 2	If the trigger happens, the data is obtained by N.

highPassFilter_	_ OFF or 0	High Pass Filter Off
	_ ON or 1	High Pass Filter On
indexModule_	0 ~ MAX_DAQ	SDU 2040 device number

- **Usage :**

The device of Module 0 is set as 8KS/s, continuous sampling mode

In fact, the sampling size 1024, Trigger Sampling mode has no meaning in _SAMPLING_CONTINUOUS_MODE

C++

```
double res = sdDaqAdcSamplingConfig( 8000,
                                     1024,
                                     _SAMPLING_CONTINUOUS_MODE,
                                     _TRIG_CONTINUOUS_MODE,
                                     _OFF,
                                     0 );
if (res <= 0) {
    printf("Error : Sampling Configuration");
}
```

Visual Basic

```
Dim Res   As Double
Res = sdDaqAdcSamplingConfig( 8000,
                              1024,
                              SD_SAMPLING_CONTINUOUS_MODE,
                              SD_TRIG_CONTINUOUS_MODE,
                              SD_OFF,
                              0 );
If (Res = 0) Then
        MsgBox   "Error : Sampling Configuration"
        End
End If
```

- **Returns :**

In case of success :The samplingRate value that is set

In case of failure : If it is -1, samplingNumber_ is set. If it is 0, samplingRate_

results in error.

- Comments:

  In case of _SAMPLING_CONTINUOUS_MODE or _TRIG_CONTINUOUS_MODE, it is set internally. So, the user cannnot set it.

(7) sdDaqArm

- Description:

  Data Acquisition start finish command sdDaqArm(_START) should be necessarily executed before the start of sdDaqStart command, and sdDaqArm (_STOP) command should be executed before sdDaqStart(_STOP).

- The setting of Analog Input, Counter, Trigger should be completely set before that. This function should be necessarily called when only the counter is used.

- Format :

  C++

```
SD_DAQ_ERROR sdDaqArm(
    SD_START_STOP startStop_,
    int indexModule_
);
```

  Visual Basic

```
sdDaqArm   (ByVal startStop_ As Long, ByVal indexModule_ As Long) As Long
```

- Parameters:

Parameter	Value	Mean
startStop_	_STOP or 0	Completion of data acquisition
	_START or 1	Start of data acquisition
indexModule_	0 ~ MAX_DAQ	SDU 2040 device number

- Usage :

  Start of module 0 data acquisition

  C++

```
res = sdDaqArm(_START, 0);
if (res != _ERROR_NONE) {
    printf("ERROR : DAQ Arm");
}
```

Visual Basic

```
Dim Res   As Long
Res = sdDaqArm(SD_START, 0)
If (Res = 0) Then
        MsgBox " ERROR : DAQ Arm"
        End
End If
```

- Returns :

  In case of success : _ERROR_NONE

  In case of failure : See the error type of "usbDaqErrorType.h".

- Comments:

(8) sdDaqCounterConfig

- Description:

  The data aquisitio of counter setting channel on/off is same as sdDaqArm.

- Format :

  C++

```
SD_DAQ_ERROR sdDaqCounterConfig(
    SD_ON_OFF onOff_,
    int indexModule_
);
```

  Visual Basic

```
sdDaqCounterConfig (ByVal onOff_ As Long, ByVal indexModule_ As Long) As Long
```

- Parameters:

Parameter	Value	Mean
onOff_,	_ OFF or 0	Counter Off
	_ ON or 1	Counter On
indexModule_	0 ~ MAX_DAQ	SDU 2040 device number

- Usage :

  It switches on the device 0 number counter.

  C++

```
SD_DAQ_ERROR res = sdDaqCounterConfig(_ON, 0);
if (res != _ERROR_NONE) {
    printf("ERROR : Counter Configuration");
}
```

  Visual Basic

```
Dim Res   As Long
Res = sdDaqCounterConfig(SD_ON, 0);
If (Res< = 0) Then
        MsgBox   "ERROR : Counter Configuration"
        End
End If
```

- **Returns :**

  In case of success : _ERROR_NONE

  In case of failure : See the error type of "usbDaqErrorType.h".

- **Comments:**

(9) sdDaqTriggerConfig

- Description:

  trigger mode, trigger option(trigger source, level, up/down edge), pre/post trigger

- Format :

  C++

```
SD_DAQ_ERROR sdDaqTriggerConfig(
    SD_TRIGGER_SOURCE source_,
    int level_,
    SD_TRIGGER_EDGE edge_,
    SD_CHANNEL analogSourceChannel_,
    SD_TRIGGER_PREPOST prePost_,
    int prePostNum_,
    int indexModule_
);
```

  Visual Basic

```
sdDaqTriggerConfig (ByVal source_ As Long,
                    ByVal level_ As Long,
                    ByVal edge_ As Long,
                    ByVal analogSourceChannel_ As Long,
                    ByVal prePost_ As Long,
                    ByVal prePostNum_ As Long,
                    ByVal indexModule_ As Long) As Long
```

- **Parameters:**

Parameter	Value	Mean
source_	_DIGITAL or 1	Trigger Source : Digital trigger
	_ANALOG or 2	Trigger Source : Analog trigger
	_COUNTER or 4	Trigger Source : Counter trigger
level_	Integer	It sets the trigger level. If the trigger source is analog, it becomes the input value of mV, and if it is the counter, it becomes the number of counter.
edge_	_RISING_EDGE or 0	It sets the trigger edge. It sets in case that the source is analog or digital.
	_FALLING_EDGE or 1	It sets the trigger eduge. It sets in case that the trigger source is analog or digital.
analogSourceChannel_	_CHANNEL_1 or 0	Analog Source : Channel 1
	_CHANNEL_2 or 1	Analog Source : Channel 2
	_CHANNEL_3 or 2	Analog Source : Channel 3
	_CHANNEL_4 or 3	Analog Source : Channel 4
prePost_	_PREPOST_NONE or 0	No Pre/Post Sampling
	_POST_SAMPLING or 2	Post Sampling Mode
	_PRE_SAMPLING or 4	Pre Sampling Mode
prePostNum_	Integer	The number in case of Pre/Post Sampling Mode
indexModule_	0 ~ MAX_DAQ	SDU 2040 device number

- **Usage :**

  It sets the device 0 as Analog Trigger Mode, sets the Rising Edge Mode, Analog Trigger Source as channel 11, and sets the Analog Trigger Level as the Pre Sampling Mode of 50 data.

C++

```
SD_DAQ_ERROR res = sdDaqTriggerConfig(_ANALOG, 100, _RISING_EDGE,
_CHANNEL_1, _PRE_SAMPLING, 50, 0);
if (res != _ERROR_NONE) {
    printf("ERROR : Trigger Configuration");
}
```

Visual Basic

```
Dim Res  As Long
Res = sdDaqTriggerConfig(SD_ANALOG, 100, SD_RISING_EDGE, SD_CHANNEL_1,
        SD_PRE_SAMPLING, 50, 0)
If (Res< = 0) Then
        MsgBox "ERROR : Trigger Configuration"
        End
End If
```

- Returns :

  In case of success : _ERROR_NONE

  In case of failure : See the error type of "usbDaqErrorType.h".

- Comments:

(10) sdDaqDacConfig

- Description:

  channel on/off, dacOutMode, DAO sampling rate

- Format :

  C++

```
SD_DAQ_ERROR sdDaqDacConfig(
    SD_ON_OFF onOffCh1_,
    SD_ON_OFF onOffCh2_,
    SD_DAC_MODE modeCh1_,
    SD_DAC_MODE modeCh2_,
    SD_DAC_SAMP_RATE sampRate_,
    int indexModule_
);
```

  Visual Basic

```
sdDaqDacConfig (ByVal onOffCh1_ As Long,
                ByVal onOffCh2_ As Long,
                ByVal modeCh1_ As Double,
                ByVal modeCh2_ As Long,
                ByVal sampRate_ As Long,
                ByVal indexModule_ As Long) As Long
```

- **Parameters:**

Parameter	Value	Mean
onOffCh1_	_ OFF or 0	Analog Output Channel 1 Off
	_ ON or 1	Analog Output Channel 1 On
onOffCh2_	_ OFF or 0	Analog Output Channel 2 Off
	_ ON or 1	Analog Output Channel 2 On
modeCh1_,	_TRIGGER_OUT or 0	It sets the Analog Output Channel 1 as the Trigger Out mode. This is used to fit the synchronization at the multi device.
	_TABLE_OUT or 1	It sets the Analog Output Channel 1 as the Table Out mode. If the user lowers the certain table. The output signal is determined by that table.
modeCh2_	_TRIGGER_OUT or 0	It sets the Analog Output Channel 2 as the Trigger Out mode. This is used to fit the synchronization at the multi device.
	_TABLE_OUT or 1	It sets the Analog Output Channel 2 as the Table Out mode. If the user lowers the certain table. The output signal is determined by that table.
sampRate_	_96KSPS or 0x00,	It sets the Sampling Frequency of Analog Out. The maximum frequency that is output is determined by this setting.
	_48KSPS or 0x01,	
	_32KSPS or 0x02,	
	_24KSPS or 0x03,	
	_16KSPS or 0x04,	
	_12KSPS or 0x05,	
	_8KSPS or 0x06	
indexModule_	0 ~ MAX_DAQ	SDU 2040 device number

- **Usage :**

  The 2 Analog Out channels of device number 0 are switched on, and set as Table Out Mode. The Sampling Frequency is set as 96KS/s.

  C++

```
SD_DAQ_ERROR res = sdDaqDacConfig(_ON, _ON, _TABLE_OUT, _TABLE_OUT,
_96KSPS, 0);
if (res != _ERROR_NONE) {
      printf("ERROR : Analog out Configuration");
}
```

  Visual Basic

```
Dim Res   As Long
Res = sdDaqDacConfig(SD_ON,SD _ON, SD_TABLE_OUT, SD_TABLE_OUT,
        SD _96KSPS, 0)
If (Res <= 0) Then
          MsgBox "ERROR : Analog out Configuration"
          End
End If
```

- **Returns :**

  In case of success : _ERROR_NONE

  In case of failure : See the error type of "usbDaqErrorType.h".

- **Comments:**

(11) sdDaqDacStart

- Description:

  Analog Out Start / Stop

- Format :

  C++

```
SD_DAQ_ERROR sdDaqDacStart(
    SD_ON_OFF startOnOff_,
    int indexModule_
);
```

  Visual Basic

```
sdDaqDacStart  (ByVal startOnOff_ As Long, ByVal indexModule_ As Long) As Long
```

- Parameters:

Parameter	Value	Mean
startOnOff_	_ OFF or 0	Analog Output Stop
	_ ON or 1	Analog Output Start
indexModule_	0 ~ MAX_DAQ	SDU 2040 device number

- Usage :

  C++

```
SD_DAQ_ERROR res = sdDaqDacStart(_ON, 0);
if (res != _ERROR_NONE) {
    printf("ERROR : Analog out Start");
}
```

  Visual Basic

```
Dim Res  As Long
Res = sdDaqDacStart(SD_ON, 0)
If (Res< = 0) Then
        MsgBox  "ERROR : Analog out Start"
        End
End If
```

- **Returns :**

  In case of success : _ERROR_NONE

  In case of failure : See the error type of "usbDaqErrorType.h".

- **Comments:**

(12) sdDaqDacSetFormTable

- Description:

  It is the function to create the table via the internal setting. A total of 6 functions can be set. (DC , SIN, SQU, TRI, RAMP, NOISE)

- Format :

  C++

```
int sdDaqDacSetFormTable(
    SD_CHANNEL channel_,
    SD_DAC_FUNCTION function_,
    int frequency_,
    int amplitude_,
    int offset_,
    int indexModule_
);
```

  Visual Basic

```
sdDaqDacSetFormTable (ByVal channel_ As Long,
                      ByVal function_ As Long,
                      ByVal frequency_ As Long,
                      ByVal amplitude_ As Long,
                      ByVal offset_ As Long,
                      ByVal indexModule_ As Long) As Long
```

- **Parameters:**

Parameter	Value	Mean
channel_	_CHANNEL_1 or 0	Analog Out: Channel 1
	_CHANNEL_2 or 1	Analog Out : Channel 2
function_	_DAC_DC or 0	DC
	_DAC_SIN or 1	SINE Wave
	_DAC_SQU or 2	SQUE Wave
	_DAC_TRI or 3	TRIANGLE Wave
	_DAC_RAMP or 4	RAMP Wave
	_DAC_NOISE or 5	Random Noise
frequency_	Integer	According to the sampling frequency of analo. Out, it is valid as in the following. _96KSPS : 48KHz ~ 96Hz _48KSPS : 24KHz ~ 48Hz _32KSPS : 16KHz ~ 32Hz _24KSPS : 12KHz ~ 24Hz _16KSPS : 8KHz ~ 16Hz _12KSPS : 6KHz ~ 12Hz _8KSPS : 4KHz ~ 8Hz
amplitude_	Integer	The effective value is 20V(20000) at the maximum, and fluid according to the Offset.
Offset_	Integer	It posseses the +/- 10V(+/-10000) value.
indexModule_	0 ~ MAX_DAQ	SDU 2040 device number

- **Usage :**

  It sets the Analog Out Channel 1 of device number to output the 100Hz, SQUE wave with 0V~5V.

  C++

```
int resInt = sdDaqDacSetFormTable (_CHANNEL_1, _DAC_SQU, 100, 5000, 2500, 0);
if (resInt <= 0) {
```

```
        printf("Error : Analog Out Making Table");
    }
```

Visual Basic

```
Dim Res   As Long
Res = sdDaqDacSetFormTable (SD_CHANNEL_1, SD_DAC_SQU, 100, 5000, 2500,
0)        If (Res <= 0) Then
          MsgBox   "Error : Analog Out Making Table"
          End
End If
```

- Returns :

  In case of success : the value of revised frequency

  In case of failure : The value equal to or smaller than 0(See the error type of ("usbDaqErrorType.h").

- Comments:

  This function can be used in the sdDaqDacConfig when the model of channel setting is in the TABEL_OUT mode.

(13) sdDaqDacSetUserTable

- Description:

  It is the function to take the external user table out from the SDU 2040 in order to output in the form of Analog Out.

- Format :

  C++

```
SD_DAQ_ERROR sdDaqDacSetUserTable(
    SD_CHANNEL channel_,
    int bufferSize_,
    unsigned char* tableBuffer_,
    int indexModule_
);
```

  Visual Basic

```
sdDaqDacSetUserTable (ByVal channel_ As Long,
                      ByVal bufferSize_ As Long,
                      ByVal tableBuffer_ (3072)As String,
                      ByVal indexModule_ As Long) As Long
```

- Parameters:

Parameter	Value	Mean
channel_	_CHANNEL_1 or 0	Analog Out: Channel 1
	_CHANNEL_2 or 1	Analog Out : Channel 2
BufferSize_	Integer(max 1024Sample:24bit)	It is the size of the table that the use intends to take down, and can be set as 1024 sample at the maximum. (The 1Sample is 24bit)
tableBuffer_,	Unsigned Char *	The start address of user table
indexModule_	0 ~ MAX_DAQ	SDU 2040 device number

- **Usage :**

  C++

```
unsigned char userDefineTable[3072];
// ~
//Fill the table with data
// ~
SD_DAQ_ERROR res;
res = sdDaqDacSetUserTable (_CHANNEL_1, 1024, &userDefineTable, 0);
if (res != _ERROR_NONE) {
    printf("ERROR : Analog out User Table");
}
```

  Visual Basic

```
Dim Res   As Long
Dim userDefineTable(3072) As String
Res = sdDaqDacSetUserTable (SD_CHANNEL_1, 1024, userDefineTable(3072), 0)
If (Res <= 0) Then
        MsgBox "ERROR : Analog out User Table"
        End
End If
```

- **Returns :**

  In case of success : _ERROR_NONE

  In case of failure : See the error type of "usbDaqErrorType.h".

- **Comments:**

  This function can be used in the sdDaqDacConfig functioin when the mode setting of each channel is in the TABEL_OUT mode.

(14) sdDaqDacTriggerLevel

- Description:

  It is the function to set the trigger level when it is Analog Out Trigger mode.

- Format :

  C++

```
SD_DAQ_ERROR sdDaqDacTriggerLevel(
    SD_CHANNEL channel_,
    int trigLevel_,
    int indexModule_
);
```

  Visual Basic

```
sdDaqDacTriggerLevel (ByVal channel_ As Long,
                      ByVal trigLevel_ As Long,
                      ByVal indexModule_ As Long) As Long
```

- Parameters:

Parameter	Value	Mean
channel_	_CHANNEL_1 or 0	Analog Out: Channel 1
	_CHANNEL_2 or 1	Analog Out : Channel 2
trigLevel_	Integer	It is input by the mV unit that determines the trigger level of Analog Out. (At present, it is automatically set in case of Multi Device).
indexModule_	0 ~ MAX_DAQ	SDU 2040 device number

- Usage:

  C++

```
SD_DAQ_ERROR res;
res = sdDaqDacTriggerLevel (_CHANNEL_1, 5000, 0);
if (res != _ERROR_NONE) {
    printf("ERROR : Analog out Trigger Level");
}
```

Visual Basic

```
Dim Res   As Long
Res = sdDaqDacTriggerLevel (SD_CHANNEL_1, 5000, 0)
If (Res = 0) Then
        MsgBox " ERROR : Analog out Trigger Level"
        End
End If
```

- Returns :

  In case of success : _ERROR_NONE

  In case of failure : See the error type of "usbDaqErrorType.h".

- Comments:

  This function can be used in the sdDaqDacConfig function when the mode setting of each channel is in the TRIGGER_OUT mode.

  At present, this setting is valid only in the Multi Device, and the trigger level is automatically set in case that multi device is operating.

15. sdDaqTransferData

- Description:

  It reads the data through USB.

  The form of data depends on the channel that is currently activated.

  The data form in case that the 4 analog input channels and counter are activated at the same time, is such as followings:

  The analog input is 24 bit data, and the counter is 32 bit data. But all are converted into 32 bit and occupy the data space of 4 byte.

Channel 1 (4byte)	Channel 2 (4byte)	Channel 3 (4byte)	Channel 4 (4byte)	Counter (4byte)	Channel 1 (4byte)	Channel 2 (4byte)	•••	Counter (4byte)

  The method to convert the obtained analog data into the real data is such as followings:

  #define _LOWMAX 8388607

  Value of real data = 10.0 * (Data value of Analog Input) / _LOWMAX

  Reference : Example No. 4

- Format :

  C++

```
int sdDaqTransferData(
    int** dataPoint_,
    int indexModule_
);
```

  Visual Basic

```
sdDaqTransferData (ByRef dataPoint_ As Long, ByVal indexModule_ As Long) As Long
```

- Parameters:

Parameter	Value	Mean
dataPoint_	Integer **	It receives the data address of the data stored in the form of integer internally, and reads the data. The data butter is allocated automatically internally. So, no work like the butter allocation is required.

indexModule_	0 ~ MAX_DAQ	SDU 2040 device number

- **Usage :**

  C++

```
int* buffer;
int readDataCounter = sdDaqTransferData(&buffer, 0);
if (readDataCounter <= 0) {
    printf("Error : Get Data");
}
```

  Visual Basic

```
Dim Res   As Long
Dim buffer As Long
Res = sdDaqTransferData( buffer, 0)
If (Res< = 0) Then
        MsgBox "Error : Get Data"
        End
End If
```

- **Returns :**

  In case of success : The number of read data

  In case of failure : The number equal to or smaller than 0 (See the error type of "usbDaqErrorType.h")

- **Comments:**

16. sdDaqTransferPreDataFile

- Description:

  It is the function to create the file and make the header information before the storage of data file and the data.

  The information of head file is composed such as following:

  - 0: module index(1byte)
  - 1: Channel 1
    - 1 : onOff(1byte : char)
    - 2: minmax(8byte : double)
    - 10 : scale(8byte : double)
    - 18 : offset(8byte : double)
    - 26 : unit(15byte : char)
    - 41: calibAmp(4byte : float) – Not used
    - 35: calibOffset(4byte : int) – Not used
    - 49 : reserve(8byte)
  - 100 : Channel 2 – The composition is idential to the channle 1(starting from 101).
  - 200 : Channel 3 – The composition is idential to the channle 1(starting from 201).
  - 300 : Channel 4 – The composition is idential to the channle 1(starting from 31).
  - It sets the Analog Output Channel 1 as the Table Out mode. If the user lowers the certain table. The output signal is determined by that table.
  - 400 : Counter – The composition is idential to the channle 1(starting from 401).
  - 500 : It is the mark that informs that the current file is the Uproject file.
    - "uproject"(8byte : 500)
  - 508 : Sampling Rate(8byte(double))
  - 516 : Other information etc reserve(84byte : 234)
  - The data is stored from 600.

- Format :

  C++

```
SD_DAQ_ERROR sdDaqTransferPreDataFile(
    char* fPath_,
    int indexModule_
```

);

Visual Basic

sdDaqTransferPreDataFile (ByVal fPath_ As String, ByVal indexModule_ As Long) As Long

- Parameters:

Parameter	Value	Mean
fPath_,	char*	path of file for the data storage
indexModule_	0 ~ MAX_DAQ	SDU 2040 device number

- Usage :

C++

```
char filePath[] = "C:₩₩test.bin";
SD_DAQ_ERROR res = sdDaqTransferPreDataFile(filePath, 0);
if (res != _ERROR_NONE) {
    printf("Error : Making Header File");
}
```

Visual Basic

```
Dim Res   As Long
Dim filePath   As String
FilePath = "test.bin"
Res = sdDaqTransferPreDataFile(filePath, 0)
If (Res <= 0) Then
        MsgBox "Error : Making Header File"
        End
End If
```

- Returns :

In case of success : _ERROR_NONE

In case of failure : See the error type of "usbDaqErrorType.h".

- Comments:

This function should be necessarily called first to store the data to ensure other subsequent functions can be used.

17. sdDaqTransferDataFile

- Description:

  It stores the data after changing the data according to the channel information. The real data is input from the 600th address of file, and the form of data input is such as followings.(When the 4 analogue channles and counter are all being activated)

Channel 1 (4byte)	Channel 2 (4byte)	Channel 3 (4byte)	Channel 4 (4byte)	Counter (4byte)	Channel 1 (4byte)	Channel 2 (4byte)	•••	Counter (4byte)

- Format :

  C++

```
SD_DAQ_ERROR sdDaqTransferDataFile(
    int indexModule_
);
```

  Visual Basic

```
sdDaqTransferDataFile (ByVal indexModule_ As Long) As Long
```

- Parameters:

Parameter	Value	Mean
indexModule_	0 ~ MAX_DAQ	SDU 2040 device number

- Usage :

  C++

```
SD_DAQ_ERROR res = sdDaqTransferDataFile (0);
if (res != _ERROR_NONE) {
    printf("Error : Data Transfer to File");
}
```

  Visual Basic

```
Dim Res   As Long
Res = sdDaqTransferDataFile (0)
If (Res < = 0) Then
        MsgBox   "Error : Data Transfer to File"
        End
End If
```

- **Returns :**

  In case of success : _ERROR_NONE

  In case of failure : See the erro type of "usbDaqErrorType.h".

- **Comments:**

  The sdDaqTransferDataFile function can be used only after the path of file is set for the data storage by necessarily calling the sdDaqTransferPreDataFile function.

18. sdDaqTransferEndDataFile

- **Description:**

  It is the function to finish the file after the data storage is over.

- **Format :**

  C++

```
SD_DAQ_ERROR sdDaqTransferEndDataFile(
    int indexModule_
);
```

  Visual Basic

```
sdDaqTransferEndDataFile (ByVal indexModule_ As Long) As Long
```

- **Parameters:**

Parameter	Value	Mean
indexModule_	0 ~ MAX_DAQ	SDU 2040 device number

- **Usage :**

  C++

```
SD_DAQ_ERROR res = sdDaqTransferEndDataFile (0);
if (res != _ERROR_NONE) {
    printf("Error : Close File");
}
```

  Visual Basic

```
Dim Res   As Long
Res = sdDaqTransferEndDataFile (0)
If (Res < = 0) Then
        MsgBox   "Error : Close File"
        End
End If
```

- **Returns :**

  In case of success : _ERROR_NONE

  In case of failure : See the error type of "usbDaqErrorType.h".

- **Comments:**

  The sdDaqTransferEndDataFile function can be used only after the path of file is set for the data storage by necessarily calling the sdDaqTransferPreDataFile function.

19. sdDaqStart

- Description:

  It starts and finishes Data acquisition

- Format :

  C++

```
SD_DAQ_ERROR sdDaqStart(
    SD_ON_OFF startOnOff_ ,
    int indexModule_
);
```

  Visual Basic

```
sdDaqStart (ByVal startOnOff_ As Long, ByVal indexModule_ As Long) As Long
```

- Parameters:

Parameter	Value	Mean
startOnOff_	_ OFF or 0	Data Acquisition Start
	_ ON or 1	Data Acquisition Stop
indexModule_	0 ~ MAX_DAQ	SDU 2040 device number

- Usage :

  C++

```
SD_DAQ_ERROR res = sdDaqStart(_ON, 0);
if (res != _ERROR_NONE) {
    printf("Error : Data Acquisition Start");
}
```

  Visual Basic

```
Dim Res   As Long
Res = sdDaqStart(SD_ON, 0)
If (Res <= 0) Then
        MsgBox   "Error : Data Acquisition Start"
        End
End If
```

- Returns :

  In case of success : _ERROR_NONE

  In case of failure : See the error type of "usbDaqErrorType.h".

- Comments:

  All settings should be completed at the start and it should be executed only after sdDaqArm starts.If the current data is being obtained at the finishing point, the sdDaqTransferDataStop function is called in the first place and the data transmission comes to a halt before finishing.

20. sdDaqWriteUserData

- Description:

  It is the function that stores the data that user wants on the EEPROM inside the SDU 2040.

  .

- Format :

  C++

```
SD_DAQ_ERROR sdDaqWriteUserData(
    char* data_,
    WORD address_,
    int indexModule_
);
```

  Visual Basic

```
sdDaqWriteUserData (ByRef data_(32) As String,
                    ByVal address_ As Long,
                    ByVal indexModule_ As Long) As Long
```

- Parameters:

Parameter	Value	Mean
data_,	char	The data to be read
address_,	WORD(0, 32, 64, 96, 128, 160, 192, 224)	It uses the consecutive 32 bytes from the start address. The start address must be always the mulitple of 32 and has the value from 0 to 256.
indexModule_	0 ~ MAX_DAQ	SDU 2040 device number

- Usage :

  C++

```
char userData[32];
//~
//Fill the userData
//~
SD_DAQ_ERROR   res = sdDaqWriteUserData (userData, 0, 0);
if (res != _ERROR_NONE) {
    printf("Error : Writing User Data");
```

```
}
```

Visual Basic

```
Dim Res   As Long
Dim userData(32)   As String
Res = sdDaqWriteUserData (userData(32), 0, 0);
If (Res <= 0) Then
        MsgBox "Error : Initialize "
        End
End If
```

- Return :

  In case of success : _ERROR_NONE

  In case of failure : See the error type of "usbDaqErrorType.h".

- Comments:

21. sdDaqReadUserData

- Description:

  It is the function that reads the data that the user stored on EEPROM of SDU 2040.

- Format :

  C++

  SD_DAQ_ERROR sdDaqReadUserData(

  char* data_,

  WORD address_,

  int indexModule_

  );

  Visual Basic

  sdDaqReadUserData (ByRef data_(32) As String, ByVal address_ As Long, ByVal indexModule_ As Long) As Long

- Parameters:

Parameter	Value	Mean
data_,	char	The data to be read
address_,	WORD(0, 32, 64, 96, 128, 160, 192, 224)	It reads the consecutive 32 bytes from the start address. The start address must be always the mulitple of 32 and has the value from 0 to 256.
indexModule_	0 ~ MAX_DAQ	SDU 2040 device number

- Usage :

  C++

```
char userData[32];
SD_DAQ_ERROR   res = sdDaqReadUserData (userData, 0, 0);
if (res != _ERROR_NONE) {
    printf("Error : Reading User Data");
}
```

Visual Basic

```
Dim Res   As Long
Dim userData(32) As String
Res = sdDaqReadUserData (userData(32), 0, 0)
If (Res< = 0) Then
        MsgBox   "Error : Reading User Data"
        End
End If
```

- Return :

  In case of success : _ERROR_NONE

  In case of failure : See the error type of "usbDaqErrorType.h".

- Comments:

22. sdDaqStackConfig

- Description:

  The function that sets Multi Device

- Format :

  C++

```
SD_DAQ_ERROR sdDaqStackConfig(
    SD_ON_OFF onOff_,
    SD_MULTI_MODE multiMode_,
    int indexModule_
);
```

  Visual Basic

```
sdDaqStackConfig   (ByVal onOff_ As Long,
                    ByVal multiMode_ As Long,
                    ByVal indexModule_ As Long) As Long
```

- Parameters:

Parameter	Value	Mean
onOff_,	_ OFF or 0	Multi Device Mode Off
	_ ON or 1	Multi Device Mode On
multiMode_,	_SLAVE_MODE or 0x00	Slave mode로 동작
	_MASTER_MODE or 0x01	Master mode로 동작
indexModule_	0 ~ MAX_DAQ	SDU 2040 device number

- Usage :

  C++

```
SD_DAQ_ERROR   res = sdDaqStackConfig (userData, 0, 0);
if (res != _ERROR_NONE) {
    printf("Error : Multi Device Configuration");
}
```

  Visual Basic

```
Dim Res   As Long
Res = sdDaqStackConfig (userData, 0, 0)
If (Res = 0) Then
        MsgBox " Error : Multi Device Configuration"
        End
End If
```

- **Return :**

  In case of success : _ERROR_NONE

  In case of failure : See the error type of "usbDaqErrorType.h"

- **Comments:**

  Only one device should be always set as _MASTER_MODE in setting the multi device, and the remaining one should be set as _SLAVE_MODE to ensure the normal operation.

<sdDaqApi.h>

```
#ifndef _SDDAQAPI_H_
#define _SDDAQAPI_H_
/*
 * SDU 2040 API Version 0.9
 * 1998-2005 softDSP CO., LTD..  all rights reserved.
*/
#include <windows.h>
#include "usbDaqType.h"
#include "usbDaqErrorType.h"

#define DLL_DECL __declspec(dllexport)
#define _WINAPI WINAPI

extern "C" DLL_DECL SD_DAQ_ERROR _WINAPI sdDaqInitialize(int* deviceNum_);
extern "C" DLL_DECL SD_DAQ_ERROR _WINAPI sdDaqFinalize();
extern "C" DLL_DECL SD_DAQ_ERROR _WINAPI sdDaqOpenDevice(int indexModule_);
extern "C" DLL_DECL SD_DAQ_ERROR _WINAPI sdDaqCloseDevice(int indexModule_);
extern "C" DLL_DECL SD_DAQ_ERROR _WINAPI sdDaqAdcChannelConfig(SD_CHANNEL
channel_, SD_ON_OFF channelOnOff_, float scale_, float offset_, char* units_,
SD_ADC_COUPLING coupling_, SD_ON_OFF icpOnOff_, int indexModule_);
extern "C" DLL_DECL double _WINAPI sdDaqAdcSamplingConfig(double samplingRate_,
int samplingNumber_, SD_SAMPLING_MODE samplingMode_, SD_TRIGGER_MODE
trigSamplingmode_, SD_ON_OFF highPassFilter_, int indexModule_);
extern "C" DLL_DECL SD_DAQ_ERROR _WINAPI sdDaqArm(SD_START_STOP startStop_,
int indexModule_);
extern "C" DLL_DECL SD_DAQ_ERROR _WINAPI sdDaqCounterConfig(SD_ON_OFF onOff_,
int indexModule_);
extern "C" DLL_DECL SD_DAQ_ERROR _WINAPI
sdDaqTriggerConfig(SD_TRIGGER_SOURCE source_, int level_, SD_TRIGGER_EDGE
edge_, SD_CHANNEL analogSourceChannel_, SD_TRIGGER_PREPOST prePost_, int
prePostNum_, int indexModule_);
extern "C" DLL_DECL SD_DAQ_ERROR _WINAPI sdDaqDacConfig(SD_ON_OFF onOffCh1_,
SD_ON_OFF onOffCh2_, SD_DAC_MODE modeCh1_, SD_DAC_MODE modeCh2_,
```

```
SD_DAC_SAMP_RATE sampRate_, int indexModule_);
extern "C" DLL_DECL SD_DAQ_ERROR _WINAPI sdDaqDacStart(SD_ON_OFF startOnOff_,
int indexModule_);
extern "C" DLL_DECL int _WINAPI sdDaqDacSetFormTable(SD_CHANNEL channel_,
SD_DAC_FUNCTION function_, int frequency_, int amplitude_, int offset_, int
indexModule_);
extern "C" DLL_DECL SD_DAQ_ERROR _WINAPI sdDaqDacSetUserTable(SD_CHANNEL
channel_, int bufferSize_, unsigned char* tableBuffer_, int indexModule_);
extern "C" DLL_DECL SD_DAQ_ERROR _WINAPI sdDaqDacTriggerLevel(SD_CHANNEL
channel_, int trigLevel_, int indexModule_);
extern "C" DLL_DECL int _WINAPI sdDaqTransferData(int** dataPoint_, int indexModule_);
extern "C" DLL_DECL SD_DAQ_ERROR _WINAPI sdDaqTransferPreDataFile(char* fPath_,
int indexModule_);
extern "C" DLL_DECL SD_DAQ_ERROR _WINAPI sdDaqTransferDataFile(int indexModule_);
extern "C" DLL_DECL SD_DAQ_ERROR _WINAPI sdDaqTransferEndDataFile(int
indexModule_);
extern "C" DLL_DECL SD_DAQ_ERROR _WINAPI sdDaqStart(SD_ON_OFF startOnOff_ , int
indexModule_);
extern "C" DLL_DECL SD_DAQ_ERROR _WINAPI sdDaqWriteUserData(char* data_, WORD
address_, int indexModule_);
extern "C" DLL_DECL SD_DAQ_ERROR _WINAPI sdDaqReadUserData(char* data_, WORD
address_, int indexModule_);
extern "C" DLL_DECL SD_DAQ_ERROR _WINAPI sdDaqStackConfig(SD_ON_OFF onOff_,
SD_MULTI_MODE multiMode_, int indexModule_);

#endif
```

<usbDaqType.h>

```
#ifndef _USBDAQTYPE_H_
#define _USBDAQTYPE_H_

enum {MAX_CHANNL_NUM = 5};
enum {MAX_AI_NUM = 4};
enum {MAX_AO_NUM = 2};

typedef enum {
    _SAMPLING_CONTINUOUS_MODE      = 0x00,
    _SAMPLING_N_SAMPLE_MODE        = 0x01,
    _SAMPLING_TRIGGER_MODE         = 0x02
}SD_SAMPLING_MODE;

typedef enum {
    _TRIG_CONTINUOUS_MODE          = 0x00,
    _TRIG_SINGLE_N_SAMPLING_MODE   = 0x01,
    _TRIG_EVERY_N_SAMPLING_MODE    = 0x02
}SD_TRIGGER_MODE;

typedef enum {
    _STOP                  = 0x00,
    _START                 = 0x01
}SD_START_STOP;

typedef enum {
    _OFF                   = 0x00,
    _ON                    = 0x01
}SD_ON_OFF;

typedef enum {
    _AC                    = 0x00,
    _DC                    = 0x01
}SD_ADC_COUPLING;
```

```
typedef enum {
    _DISABLE                = 0x00,
    _ENABLE                 = 0x01
}SD_ENABLE_DISABLE;

typedef enum {
    _TRIGGER_OUT            = 0x00,
    _TABLE_OUT              = 0x01
}SD_DAC_MODE;

typedef enum {
    _96KSPS = 0x00,
    _48KSPS = 0x01,
    _32KSPS = 0x02,
    _24KSPS = 0x03,
    _16KSPS = 0x04,
    _12KSPS = 0x05,
    _8KSPS = 0x06
}SD_DAC_SAMP_RATE;

typedef enum {
    _DAC_DC = 0,
    _DAC_SIN = 1,
    _DAC_SQU = 2,
    _DAC_TRI = 3,
    _DAC_RAMP = 4,
    _DAC_NOISE = 5
}SD_DAC_FUNCTION;

typedef enum {
    //_NONE                 = 0x00,
    _DIGITAL                = 0x01,
    _ANALOG                 = 0x02,
    _COUNTER                = 0x04
}SD_TRIGGER_SOURCE;
```

```
typedef enum {
    _RISING_EDGE            = 0x00,
    _FALLING_EDGE           = 0x01,
}SD_TRIGGER_EDGE;

typedef enum {
    _PREPOST_NONE           = 0x00,
    _POST_SAMPLING          = 0x02,
    _PRE_SAMPLING           = 0x04
}SD_TRIGGER_PREPOST;

typedef enum {
    _COUNTER_OFF = 0x00,
    _COUNTER_ON = 0x08,
}SD_COUNTER;

typedef enum {
    _CHANNEL_1              = 0x00,
    _CHANNEL_2              = 0x01,
    _CHANNEL_3              = 0x02,
    _CHANNEL_4              = 0x03,
    _CHANNEL_COUNTER        = 0x04,
}SD_CHANNEL;

typedef enum {
    _HIGH_SPEED             = 0x20,
    _FULL_SPEED             = 0x11
}SD_SPEED_MODE;

typedef enum {
    _SLAVE_MODE             = 0x00,
    _MASTER_MODE            = 0x01
}SD_MULTI_MODE;

#endif
```

<usbDaqErrorType.h>

```
#ifndef _USBDAQERRORTYPE_H_
#define _USBDAQERRORTYPE_H_

/* Error Code Definitions */
typedef enum {
    _ERROR_NONE                 =  1,    /* NO Error */
    _ERROR_USB_DRIVER           = -1,   /* USB Driver Error */
    _ERROR_DEVICE_NOT_FOUND     = -2,   /* Device not Found */
    _ERROR_USB_DRIVER_READ      = -3,   /* USB Driver Read Error */
    _ERROR_USB_DRIVER_WRITE     = -4,   /* USB Driver Write Error */
    _ERROR_DEVICE_OPEN          = -5,   /* Device Open Error */
    _ERROR_DEVICE_CLOSE         = -6,   /* Device Close Error */

    _ERROR_DAQ_START_STOP       = -10,   /* Sampling Start/stop Error */
    _ERROR_ADC_START_STOP       = -11,   /* ADC Start/stop  Error */
    _ERROR_ADC_SET              = -12,   /* ADC SET(Sample Rate, High Pass
                                            Filter, Zero Calibration, Pre Sample)
                                            Error */
    _ERROR_ADC_SET_READ         = -13,   /* ADC SET(Sample Rate, High Pass
                                            Filter, Zero Calibration, Pre Sample)
                                            READ Error */
    _ERROR_ADC_CHANNEL_SET      = -14,   /* ADC Channel Select Error */
    _ERROR_ADC_ICP_SET          = -15,   /* ADC ICP Select Error */
    _ERROR_ADC_LP_SET           = -16,   /* ADC Low Pass Filter Error */

    _ERROR_SAMPLING_SET         = -20,   /* Sampling Time (trigger, interval,
                                            number) Error */
    _ERROR_SAMPLING_SIZE        = -21,   /* Sampling Size Error */

    _ERROR_DAC_START_STOP       = -30,   /* Analog Out Start/Stop Error */
    _ERROR_DAC_PULSE_SET        = -31,   /* Analog Out Pulse Error */
    _ERROR_DAC_TABLE_NUM        = -32,   /* Analog Out Table Number Error */
    _ERROR_DAC_WRITE_DATA       = -33,
```

```
    _ERROR_DAC_SET                      = -34,

    _ERROR_TRIG_SET                     = -40,   /* Trigger On/Off */
    _ERROR_TRIG_DIGITAL                 = -41,   /* Digital Trigger Set Error */
    _ERROR_TRIG_ANALOG                  = -42,   /* Analog Trigger Set Error */
    _ERROR_TRIG_ANALOG_LEVEL            = -43,   /* Analog Trigger Level Error */
    _ERROR_TRIG_COUNTER_LEVEL           = -44,   /* Counter Trigger Level Error */
    _ERROR_TRIG_PRE_SAMPLING_NUM        = -45,   /* Pre Sampling Number Error */

    _ERROR_READ_CONFIG_DATA             = -50,   /* Configuration Data Read Error */
    _ERROR_READ_CALIB_DATA_AMP          = -51,   /* AMP Calibration Data Read Error */
    _ERROR_WRITE_CALIB_DATA_AMP         = -52,   /* AMP Calibration Data Write Error */
    _ERROR_READ_CALIB_DATA_OFFSET       = -53,   /* Offset Calibration Data Read Error
*/
    _ERROR_WRITE_CALIB_DATA_OFFSET      = -54,   /* Offset Calibration Data Write Error
*/
    _ERROR_READ_ADC_RESET_STATUS        = -55,

    _ERROR_READ_MEMORY_DATA_NUM         = -60,   /* Read Memory Data Error */
    _ERROR_READ_COUNTER_VALUE           = -61,   /* Read Counter Value Error */

    _ERROR_FPGA_CONFIG_START            = -70,   /* FPGA Configuration Error */
    _ERROR_FPGA_WRITE_CONFIG            = -71,   /* Write FPGA Configuration Error */
    _ERROR_FPGA_CHECK_CONFIG            = -72,   /* Check FPGA Configuration Error */

    _ERROR_READ_VERSION                 = -80,   /* Read Version Error */

    _ERROR_WRITE_USER_MULTI_DATA        = -90,   /* User Specific Mulit Data Write Error
*/
    _ERROR_READ_USER_MULTI_DATA         = -91,   /* User Specific Multi Data Read
Error */
    _ERROR_WRITE_USER_DATA              = -92,   /* User Specific Data Write Error */
    _ERROR_READ_USER_DATA               = -93,   /* User Specific Data Read Error */
    _ERROR_WRITE_EEPROM                 = -94,   /* Write EEPROM Error */
    _ERROR_READ_EEPROM                  = -95,   /* Read EEPROM Error */
```

```
    _ERROR_READ_CURRENT_SPEED          = -96,   /* Read Current Speed Error */
    _ERROR_ADC_CONSTRAINT_STOP         = -97,    /* Read ADC Constraint Stop Error
*/

    _ERROR_RESET_FX2_FIFO               = -110,
    _ERROR_CHANGE_FX2_MODE             = -111,

    _ERROR_FILE_OPEN                   = -150,    /* FILE OPEN Error */

    _ERROR_MULTI_SET                   = -160     /* Multi Stack Error*/
}SD_DAQ_ERROR;

#endif
```

<Simple Example Program>

Stroring after obtaining the data through the Analog Input channel 1

<C++>

```
// Test_1.cpp
// 2005/09/06
// softDSP Co., Ltd.
// info@softdsp.com
// Stroring after obtaining the data through the Analog Input channel 1


#include <stdio.h>
#include "sdDaqApi.h"

int main(void)
{
    SD_DAQ_ERROR res;
    int i;

    //Initialize
    int m_deviceCount;
    res = sdDaqInitialize(&m_deviceCount);
    if (res != _ERROR_NONE || m_deviceCount <= 0) {
        printf("Error : Initialize");
        return -1;
    }
    else printf("Success : The number of connected device : %d ₩n",
m_deviceCount);

    //Open device 0
    res = sdDaqOpenDevice(0);
    if (res != _ERROR_NONE) {
        printf("Error : Open device");
        return -1;
    }
    else printf("Success : Open device ₩n");
```

```
	//Configuration : Analog Input
	//Turn On Analog Input Channel 1
	res=sdDaqAdcChannelConfig(_CHANNEL_1, _ON, 1.0, 0.0, "V", _DC, _OFF, 0);
	if (res != _ERROR_NONE) {
		printf("Error : Analog Input Channel Configuration");
		return -1;
	}
	else printf("Success : Analog Input Channel 1 Configuration ₩n");
	//Turn Off Analog Input Channel 2 ~ 4
	for (i = _CHANNEL_2; i < MAX_AI_NUM; i++) {
		res=sdDaqAdcChannelConfig((SD_CHANNEL)i, _OFF, 1.0, 0.0, "V", _DC,
_OFF, 0);
		if (res != _ERROR_NONE) {
			printf("Error : Analog Input Channel Configuration");
			return -1;
		}
		else printf("Success : Analog Input Channel %d Configuration ₩n", i + 1);
	}

	//Configuration : Counter
	//Turn Off Counter
	res = sdDaqCounterConfig(_OFF, 0);
	if (res != _ERROR_NONE) {
		printf("ERROR : Counter Configuration");
		return -1;
	}
	else printf("Success : Counter Configuration ₩n");

	//Configuration : Trigger
	res = sdDaqTriggerConfig(_ANALOG, 100, _RISING_EDGE, _CHANNEL_1,
_PRE_SAMPLING, 50, 0);
	if (res != _ERROR_NONE) {
		printf("ERROR : Trigger Configuration");
		return -1;
```

```
    }
    else printf("Success : Trigger Configuration ₩n");

    //Configuration : Sampling Rate
    double resInt = sdDaqAdcSamplingConfig(8000, 4096,
_SAMPLING_CONTINUOUS_MODE, _TRIG_CONTINUOUS_MODE, _OFF, 0);
    if (resInt <= 0) {
        printf("Error : Sampling Configuration");
        return -1;
    }
    else printf("Success : Sampling Configuration ₩n");

    //Save File, make head file
    char filePath[] = "test_1.bin";
    res = sdDaqTransferPreDataFile(filePath, 0);
    if (res != _ERROR_NONE) {
        printf("Error : Making Header File");
    }
    else printf("Success : Making Header File ₩n");

    //Start Arm
    res = sdDaqArm(_START, 0);
    if (res != _ERROR_NONE) {
        printf("ERROR : DAQ Arm Start");
        return -1;
    }
    else printf("Success : DAQ Arm Start ₩n");

    //Delay 100ms for Arm
    Sleep(100);

    //Sampling Start
    res = sdDaqStart(_ON, 0);
    if (res != _ERROR_NONE) {
        printf("Error : Data Acquisition Start");
```

```
        return -1;
}
else printf("Success : Data Acquisition Start ₩n");

//Data Transfer
int* buffer;
int readDataCounter = sdDaqTransferData(&buffer, 0);
if (readDataCounter <= 0) {
        printf("Error : Get Data");
        return -1;
}
else printf("Success : The Number of Data : %d ₩n", readDataCounter);

//Save File, Data transfer to the file
res = sdDaqTransferDataFile (0);
if (res != _ERROR_NONE) {
        printf("Error : Data Transfer to File");
}
else printf("Success : Data Transfer to File ₩n");

//Stop Sampling
res = sdDaqStart(_OFF, 0);
if (res != _ERROR_NONE) {
        printf("Error : Data Acquisition Stop");
        return -1;
}
else printf("Success : Data Acquisition Stop ₩n");

//Stop Arm
res = sdDaqArm(_STOP, 0);
if (res != _ERROR_NONE) {
        printf("ERROR : DAQ Arm Stop");
        return -1;
}
else printf("Success : DAQ Arm Stop ₩n");
```

```
        //Save File, Data transfer to the file
        res = sdDaqTransferEndDataFile (0);
        if (res != _ERROR_NONE) {
            printf("Error : Close File");
        }
        else printf("Success : Close File ₩n");

        //Close device 0
        res = sdDaqCloseDevice(0);
        if (res != _ERROR_NONE) {
            printf("Error : Close device");
            return -1;
        }
        else printf("Success : Close device ₩n");

        //Finalize
        res = sdDaqFinalize ();
        if (res != _ERROR_NONE ) {
            printf("Error : Finalize");
            return -1;
        }
        else printf("Success : Finalize ₩n");

        return 0;
}
```

1. Setting the Analog Out Channel 1, 2 to check the output

<C++>

```
// Test_2.cpp
// 2005/09/06
// softDSP Co., Ltd.
// info@softdsp.com
// Analog Out channel 1, 2 output

#include <stdio.h>
#include <conio.h>
#include "sdDaqApi.h"

int main(void)
{
    SD_DAQ_ERROR res;
    int resInt;

    //Initialize
    int m_deviceCount;
    res = sdDaqInitialize(&m_deviceCount);
    if (res != _ERROR_NONE || m_deviceCount <= 0) {
        printf("Error : Initialize");
        return -1;
    }
    else printf("Success : The number of connected device : %d ₩n",
m_deviceCount);

    //Open device 0
    res = sdDaqOpenDevice(0);
    if (res != _ERROR_NONE) {
        printf("Error : Open device");
        return -1;
    }
    else printf("Success : Open device ₩n");
```

```
//Analog Out Configuration
res = sdDaqDacConfig(_ON, _ON, _TABLE_OUT, _TABLE_OUT, _96KSPS, 0);
if (res != _ERROR_NONE) {
      printf("ERROR : Analog out Configuration");
}
else printf("Success : Analog out Configuration");

//Analog Out Table Configuration Channel 1
resInt = sdDaqDacSetFormTable (_CHANNEL_1, _DAC_SQU, 100, 5000, 2500, 0);
if (resInt <= 0) {
      printf("Error : Analog Out Making Table");
}
else printf("Success : Analog out Making Table ₩n");

//Analog Out Table Configuration Channel 2
resInt = sdDaqDacSetFormTable (_CHANNEL_2, _DAC_SIN, 100, 5000, 0, 0);
if (resInt <= 0) {
      printf("Error : Analog Out Making Table");
}
else printf("Success : Analog out Making Table ₩n");

//Start Analog Out
res = sdDaqDacStart(_ON, 0);
if (res != _ERROR_NONE) {
      printf("ERROR : Analog out Start");
}
else printf("Success : Analog out Start ₩n");

//waiting :
printf("Click any key to stop Analog Out ₩n");
getch();

//Stop Analog Out
res = sdDaqDacStart(_OFF, 0);
```

```
        if (res != _ERROR_NONE) {
            printf("ERROR : Analog out Stop");
        }
        else printf("Success : Analog out Stop ₩n");

        //Close device 0
        res = sdDaqCloseDevice(0);
        if (res != _ERROR_NONE) {
            printf("Error : Close device");
            return -1;
        }
        else printf("Success : Close device ₩n");

        //Finalize
        res = sdDaqFinalize ();
        if (res != _ERROR_NONE ) {
            printf("Error : Finalize");
            return -1;
        }
        else printf("Success : Finalize ₩n");

        return 0;
    }
```

2. Obtaining the data by usinng the Counter

<C++>

```
// Test_3.cpp
// 2005/09/06
// softDSP Co., Ltd.
// info@softdsp.com
// Storing after obtaining the data through the Analogu input channel 1


#include <stdio.h>
#include "sdDaqApi.h"

int main(void)
{
    SD_DAQ_ERROR res;
    int i;

    //Initialize
    int m_deviceCount;
    res = sdDaqInitialize(&m_deviceCount);
    if (res != _ERROR_NONE || m_deviceCount <= 0) {
        printf("Error : Initialize");
        return -1;
    }
    else printf("Success : The number of connected device : %d ₩n",
m_deviceCount);

    //Open device 0
    res = sdDaqOpenDevice(0);
    if (res != _ERROR_NONE) {
        printf("Error : Open device");
        return -1;
    }
    else printf("Success : Open device ₩n");
```

```
    //Configuration : Analog Input
    //Turn Off Analog Input Channel 1 ~ 4
    for (i = _CHANNEL_1; i < MAX_AI_NUM; i++) {
        res=sdDaqAdcChannelConfig((SD_CHANNEL)i, _OFF, 1.0, 0.0, "V", _DC,
_OFF, 0);
        if (res != _ERROR_NONE) {
            printf("Error : Analog Input Channel Configuration");
            return -1;
        }
        else printf("Success : Analog Input Channel %d Configuration ₩n", i + 1);
    }

    //Configuration : Counter
    //Turn Off Counter
    res = sdDaqCounterConfig(_ON, 0);
    if (res != _ERROR_NONE) {
        printf("ERROR : Counter Configuration");
        return -1;
    }
    else printf("Success : Counter Configuration ₩n");

    //Configuration : Trigger
    res = sdDaqTriggerConfig(_ANALOG, 100, _RISING_EDGE, _CHANNEL_1,
_PRE_SAMPLING, 50, 0);
    if (res != _ERROR_NONE) {
        printf("ERROR : Trigger Configuration");
        return -1;
    }
    else printf("Success : Trigger Configuration ₩n");

    //Configuration : Sampling Rate
    double resInt = sdDaqAdcSamplingConfig(8000, 4096,
_SAMPLING_CONTINUOUS_MODE, _TRIG_CONTINUOUS_MODE, _OFF, 0);
    if (resInt <= 0) {
```

```
        printf("Error : Sampling Configuration");
        return -1;
    }
    else printf("Success : Sampling Configuration ₩n");

    //Save File, make head file
    char filePath[] = "test_3.bin";
    res = sdDaqTransferPreDataFile(filePath, 0);
    if (res != _ERROR_NONE) {
        printf("Error : Making Header File");
    }
    else printf("Success : Making Header File ₩n");

    //Start Arm
    res = sdDaqArm(_START, 0);
    if (res != _ERROR_NONE) {
        printf("ERROR : DAQ Arm Start");
        return -1;
    }
    else printf("Success : DAQ Arm Start ₩n");

    //Delay 100ms for Arm
    Sleep(100);

    //Sampling Start
    res = sdDaqStart(_ON, 0);
    if (res != _ERROR_NONE) {
        printf("Error : Data Acquisition Start");
        return -1;
    }
    else printf("Success : Data Acquisition Start ₩n");

    //Data Transfer
    int* buffer;
    int readDataCounter = sdDaqTransferData(&buffer, 0);
```

```
if (readDataCounter <= 0) {
      printf("Error : Get Data");
      return -1;
}
else printf("Success : The Number of Data : %d ₩n", readDataCounter);

//Save File, Data transfer to the file
res = sdDaqTransferDataFile (0);
if (res != _ERROR_NONE) {
      printf("Error : Data Transfer to File");
}
else printf("Success : Data Transfer to File ₩n");

//Stop Sampling
res = sdDaqStart(_OFF, 0);
if (res != _ERROR_NONE) {
      printf("Error : Data Acquisition Stop");
      return -1;
}
else printf("Success : Data Acquisition Stop ₩n");

//Stop Arm
res = sdDaqArm(_STOP, 0);
if (res != _ERROR_NONE) {
      printf("ERROR : DAQ Arm Stop");
      return -1;
}
else printf("Success : DAQ Arm Stop ₩n");

//Save File, Data transfer to the file
res = sdDaqTransferEndDataFile (0);
if (res != _ERROR_NONE) {
      printf("Error : Close File");
}
else printf("Success : Close File ₩n");
```

```
	//Close device 0
	res = sdDaqCloseDevice(0);
	if (res != _ERROR_NONE) {
		printf("Error : Close device");
		return -1;
	}
	else printf("Success : Close device ₩n");

	//Finalize
	res = sdDaqFinalize ();
	if (res != _ERROR_NONE ) {
		printf("Error : Finalize");
		return -1;
	}
	else printf("Success : Finalize ₩n");

	return 0;
}
```

3. Storing after obtaining Analog Input Channel 1 and Counter data by setting the Analog Trigger(.bin, .csv)

```
// Test_4.cpp
// 2005/09/06
// softDSP Co., Ltd.
// info@softdsp.com
// storing after obtaining Analog Input Channel 1 data by setting the Analog Trigger

#include <stdio.h>
#include "sdDaqApi.h"

#define _LOWMAX 8388607
#define _LOWMIN -8388608

//Save As .CSV File
```

```
void fileSave(int* buffer_, int sizeOfBuffer_, char* fileName_)
{
    int* tempBuffer;
    FILE* fp;
    int i;

    //file open
    fp = fopen(fileName_, "wb+");
    if (fp == NULL) return;

    //memory allocation
    tempBuffer = new int[sizeOfBuffer_];

    //data transfer to the buffer
    for (i = 0; i < sizeOfBuffer_; i++) {
        memcpy(tempBuffer + i, buffer_ + i, sizeof(int));
    }

    //cacurate and write result to the file
    for (i = 0; i < sizeOfBuffer_; i++) {
        fprintf(fp, "%5.3f, ₩n", 10.0 * tempBuffer[i] / _LOWMAX);
    }

    //memory free
    delete [] tempBuffer;

    //file close
    fclose(fp);
}

//Main
int main(void)
{
    SD_DAQ_ERROR res;
    int i;
```

```
	//Initialize
	int m_deviceCount;
	res = sdDaqInitialize(&m_deviceCount);
	if (res != _ERROR_NONE || m_deviceCount <= 0) {
		printf("Error : Initialize");
		return -1;
	}
	else printf("Success : The number of connected device : %d ₩n",
m_deviceCount);

	//Open device 0
	res = sdDaqOpenDevice(0);
	if (res != _ERROR_NONE) {
		printf("Error : Open device");
		return -1;
	}
	else printf("Success : Open device ₩n");

	//Configuration : Analog Input
	//Turn On Analog Input Channel 1
	res=sdDaqAdcChannelConfig(_CHANNEL_1, _ON, 1.0, 0.0, "V", _DC, _OFF, 0);
	if (res != _ERROR_NONE) {
		printf("Error : Analog Input Channel Configuration");
		return -1;
	}
	else printf("Success : Analog Input Channel 1 Configuration ₩n");
	//Turn Off Analog Input Channel 2 ~ 4
	for (i = _CHANNEL_2; i < MAX_AI_NUM; i++) {
		res=sdDaqAdcChannelConfig((SD_CHANNEL)i, _OFF, 1.0, 0.0, "V", _DC,
_OFF, 0);
		if (res != _ERROR_NONE) {
			printf("Error : Analog Input Channel Configuration");
			return -1;
		}
```

```
            else printf("Success : Analog Input Channel %d Configuration ₩n", i + 1);
        }

        //Configuration : Counter
        //Turn Off Counter
        res = sdDaqCounterConfig(_OFF, 0);
        if (res != _ERROR_NONE) {
            printf("ERROR : Counter Configuration");
            return -1;
        }
        else printf("Success : Counter Configuration ₩n");

        //Configuration : Trigger
        res = sdDaqTriggerConfig(_ANALOG, 100, _RISING_EDGE, _CHANNEL_1,
_PRE_SAMPLING, 100, 0);
        if (res != _ERROR_NONE) {
            printf("ERROR : Trigger Configuration");
            return -1;
        }
        else printf("Success : Trigger Configuration ₩n");

        //Configuration : Sampling Rate
        double resInt = sdDaqAdcSamplingConfig(8000, 1024,
_SAMPLING_TRIGGER_MODE, _TRIG_SINGLE_N_SAMPLING_MODE, _OFF, 0);
        if (resInt <= 0) {
            printf("Error : Sampling Configuration");
            return -1;
        }
        else printf("Success : Sampling Configuration ₩n");

        //Save File, make head file
        char filePath[] = "test_4.bin";
        res = sdDaqTransferPreDataFile(filePath, 0);
        if (res != _ERROR_NONE) {
            printf("Error : Making Header File");
```

```
}
else printf("Success : Making Header File ₩n");

//Start Arm
res = sdDaqArm(_START, 0);
if (res != _ERROR_NONE) {
      printf("ERROR : DAQ Arm Start");
      return -1;
}
else printf("Success : DAQ Arm Start ₩n");

//Delay 100ms for Arm
Sleep(100);

//Sampling Start
res = sdDaqStart(_ON, 0);
if (res != _ERROR_NONE) {
      printf("Error : Data Acquisition Start");
      return -1;
}
else printf("Success : Data Acquisition Start ₩n");

//Data Transfer
int* buffer;
int readDataCounter = sdDaqTransferData(&buffer, 0);
if (readDataCounter <= 0) {
      printf("Error : Get Data");
      return -1;
}
else printf("Success : The Number of Data : %d ₩n", readDataCounter);

//Save File, Data transfer to the file
res = sdDaqTransferDataFile (0);
if (res != _ERROR_NONE) {
      printf("Error : Data Transfer to File");
```

```
}
else printf("Success : Data Transfer to File ₩n");

//Save As .CSV File Format
fileSave(buffer, readDataCounter,"Test_4.csv");

//Stop Sampling
res = sdDaqStart(_OFF, 0);
if (res != _ERROR_NONE) {
      printf("Error : Data Acquisition Stop");
      return -1;
}
else printf("Success : Data Acquisition Stop ₩n");

//Stop Arm
res = sdDaqArm(_STOP, 0);
if (res != _ERROR_NONE) {
      printf("ERROR : DAQ Arm Stop");
      return -1;
}
else printf("Success : DAQ Arm Stop ₩n");

//Save File, Data transfer to the file
res = sdDaqTransferEndDataFile (0);
if (res != _ERROR_NONE) {
      printf("Error : Close File");
}
else printf("Success : Close File ₩n");

//Close device 0
res = sdDaqCloseDevice(0);
if (res != _ERROR_NONE) {
      printf("Error : Close device");
      return -1;
}
```

```
        else printf("Success : Close device ₩n");

        //Finalize
        res = sdDaqFinalize ();
        if (res != _ERROR_NONE ) {
            printf("Error : Finalize");
            return -1;
        }
        else printf("Success : Finalize ₩n");

        return 0;
}
```

4. Storing after obtaining Analog Input Channel 1 and Counter data by setting the Analog Trigger(.bin, .csv)

<C++>

```
// Test_5.cpp
// 2005/09/06
// softDSP Co., Ltd.
// info@softdsp.com
5. // Storing after obtaining Analog Input Channel 1 and Counter data by setting the
   Analog Trigger(.bin, .csv)

#include <stdio.h>
#include "sdDaqApi.h"

#define _LOWMAX 8388607
#define _LOWMIN -8388608

//Save As .CSV File
void fileSave(int* buffer_, int sizeOfBuffer_, char* fileName_)
{
    int* tempBufferAnalogInput;
    int* tempBufferCounter;
    FILE* fp;
    int i;

    //file open
    fp = fopen(fileName_, "wb+");
    if (fp == NULL) return;

    //memory allocation
    tempBufferAnalogInput = new int[sizeOfBuffer_];
    tempBufferCounter = new int[sizeOfBuffer_];

    //data transfer to the buffer
    for (i = 0; i < sizeOfBuffer_; i++) {
```

```
            memcpy(tempBufferAnalogInput + i, buffer_ + i * 2, sizeof(int));
            memcpy(tempBufferCounter + i, buffer_ + i * 2 + 1, sizeof(int));
        }

        //cacurate and write result to the file
        for (i = 0; i < sizeOfBuffer_; i++) {
            fprintf(fp, "%5.3f, ", 10.0 * tempBufferAnalogInput[i] / _LOWMAX);
            fprintf(fp, "%5d, ₩n", tempBufferCounter[i]);
        }

        //memory free
        delete [] tempBufferAnalogInput;
        delete [] tempBufferCounter;

        //file close
        fclose(fp);
}

//Main
int main(void)
{
        SD_DAQ_ERROR res;
        int i;

        //Initialize
        int m_deviceCount;
        res = sdDaqInitialize(&m_deviceCount);
        if (res != _ERROR_NONE || m_deviceCount <= 0) {
            printf("Error : Initialize");
            return -1;
        }
        else printf("Success : The number of connected device : %d ₩n",
m_deviceCount);

        //Open device 0
```

```
        res = sdDaqOpenDevice(0);
        if (res != _ERROR_NONE) {
            printf("Error : Open device");
            return -1;
        }
        else printf("Success : Open device ₩n");

        //Configuration : Analog Input
        //Turn On Analog Input Channel 1
        res=sdDaqAdcChannelConfig(_CHANNEL_1, _ON, 1.0, 0.0, "V", _DC, _OFF, 0);
        if (res != _ERROR_NONE) {
            printf("Error : Analog Input Channel Configuration");
            return -1;
        }
        else printf("Success : Analog Input Channel 1 Configuration ₩n");
        //Turn Off Analog Input Channel 2 ~ 4
        for (i = _CHANNEL_2; i < MAX_AI_NUM; i++) {
            res=sdDaqAdcChannelConfig((SD_CHANNEL)i,  _OFF,  1.0,  0.0,  "V",  _DC,
_OFF, 0);
            if (res != _ERROR_NONE) {
                printf("Error : Analog Input Channel Configuration");
                return -1;
            }
            else printf("Success : Analog Input Channel %d Configuration ₩n", i + 1);
        }

        //Configuration : Counter
        //Turn Off Counter
        res = sdDaqCounterConfig(_ON, 0);
        if (res != _ERROR_NONE) {
            printf("ERROR : Counter Configuration");
            return -1;
        }
        else printf("Success : Counter Configuration ₩n");
```

```
	//Configuration : Trigger
	res = sdDaqTriggerConfig(_ANALOG, 100, _RISING_EDGE, _CHANNEL_1,
_PREPOST_NONE, 0, 0);
	if (res != _ERROR_NONE) {
		printf("ERROR : Trigger Configuration");
		return -1;
	}
	else printf("Success : Trigger Configuration ₩n");

	//Configuration : Sampling Rate
	double resInt = sdDaqAdcSamplingConfig(8000, 1024,
_SAMPLING_TRIGGER_MODE, _TRIG_SINGLE_N_SAMPLING_MODE, _OFF, 0);
	if (resInt <= 0) {
		printf("Error : Sampling Configuration");
		return -1;
	}
	else printf("Success : Sampling Configuration ₩n");

	//Save File, make head file
	char filePath[] = "test_5.bin";
	res = sdDaqTransferPreDataFile(filePath, 0);
	if (res != _ERROR_NONE) {
		printf("Error : Making Header File");
	}
	else printf("Success : Making Header File ₩n");

	//Start Arm
	res = sdDaqArm(_START, 0);
	if (res != _ERROR_NONE) {
		printf("ERROR : DAQ Arm Start");
		return -1;
	}
	else printf("Success : DAQ Arm Start ₩n");

	//Delay 100ms for Arm
```

```
Sleep(100);

//Sampling Start
res = sdDaqStart(_ON, 0);
if (res != _ERROR_NONE) {
	printf("Error : Data Acquisition Start");
	return -1;
}
else printf("Success : Data Acquisition Start ₩n");

//Data Transfer
int* buffer;
int readDataCounter = sdDaqTransferData(&buffer, 0);
if (readDataCounter <= 0) {
	printf("Error : Get Data");
	return -1;
}
else printf("Success : The Number of Data : %d ₩n", readDataCounter);

//Save File, Data transfer to the file
res = sdDaqTransferDataFile (0);
if (res != _ERROR_NONE) {
	printf("Error : Data Transfer to File");
}
else printf("Success : Data Transfer to File ₩n");

//Save As .CSV File Format
fileSave(buffer, readDataCounter,"Test_5.csv");

//Stop Sampling
res = sdDaqStart(_OFF, 0);
if (res != _ERROR_NONE) {
	printf("Error : Data Acquisition Stop");
	return -1;
}
```

```
        else printf("Success : Data Acquisition Stop ₩n");

        //Stop Arm
        res = sdDaqArm(_STOP, 0);
        if (res != _ERROR_NONE) {
            printf("ERROR : DAQ Arm Stop");
            return -1;
        }
        else printf("Success : DAQ Arm Stop ₩n");

        //Save File, Data transfer to the file
        res = sdDaqTransferEndDataFile (0);
        if (res != _ERROR_NONE) {
            printf("Error : Close File");
        }
        else printf("Success : Close File ₩n");

        //Close device 0
        res = sdDaqCloseDevice(0);
        if (res != _ERROR_NONE) {
            printf("Error : Close device");
            return -1;
        }
        else printf("Success : Close device ₩n");

        //Finalize
        res = sdDaqFinalize ();
        if (res != _ERROR_NONE ) {
            printf("Error : Finalize");
            return -1;
        }
        else printf("Success : Finalize ₩n");

        return 0;
    }
```

<sdDaqApi.bas>

```
Declare Function GetProcessHeap Lib "kernel32" () As Long
Declare Function HeapAlloc Lib "kernel32" _
    (ByVal hHeap As Long, ByVal dwFlags As Long, ByVal dwBytes As Long) As Long
Declare Function HeapFree Lib "kernel32" _
    (ByVal hHeap As Long, ByVal dwFlags As Long, lpMem As Any) As Long

Declare Sub CopyMemoryRead Lib "kernel32" Alias _
    "RtlMoveMemory" (Destination As Any, _
    ByVal Source As Long, ByVal Length As Long)

Declare Function sdDaqInitialize Lib "sdDaqApi.dll" Alias _
   "_sdDaqInitialize@4" (ByRef deviceNum_ As Long) As Long
Declare Function sdDaqFinalize Lib "sdDaqApi.dll" Alias _
    "_sdDaqFinalize@0" () As Long
Declare Function sdDaqOpenDevice Lib "sdDaqApi.dll" Alias _
    "_sdDaqOpenDevice@4" (ByVal indexModule_ As Long) As Long
Declare Function sdDaqCloseDevice Lib "sdDaqApi.dll" Alias _
    "_sdDaqCloseDevice@4" (ByVal indexModule_ As Long) As Long
Declare Function sdDaqAdcChannelConfig Lib "sdDaqApi.dll" Alias _
    "_sdDaqAdcChannelConfig@32" (ByVal channel_ As Long, ByVal channelOnOff_
As Long, ByVal scale_ As Single, ByVal offset_ As Single, ByRef units_ As String,
ByVal coupling_ As Long, ByVal icpOnOff_ As Long, ByVal indexModule_ As Long) As
Long

Declare Function sdDaqAdcSamplingConfig Lib "sdDaqApi.dll" Alias _
    "_sdDaqAdcSamplingConfig@28" (ByVal samplingRate_ As Double, ByVal
samplingNumber_ As Long, ByVal samplingMode_ As Long, ByVal trigSamplingmode_
As Long, ByVal highPassFilter_ As Long, ByVal indexModule_ As Long) As Long

Declare Function sdDaqArm Lib "sdDaqApi.dll" Alias _
    "_sdDaqArm@8" (ByVal startStop_ As Long, ByVal indexModule_ As Long) As
Long
Declare Function sdDaqCounterConfig Lib "sdDaqApi.dll" Alias _
    "_sdDaqCounterConfig@8" (ByVal onOff_ As Long, ByVal indexModule_ As Long)
```

```
As Long
Declare Function sdDaqTriggerConfig Lib "sdDaqApi.dll" Alias _
    "_sdDaqTriggerConfig@28" (ByVal source_ As Long, ByVal level_ As Long, ByVal
edge_ As Long, ByVal analogSourceChannel_ As Long, ByVal prePost_ As Long,
ByVal prePostNum_ As Long, ByVal indexModule_ As Long) As Long

Declare Function sdDaqDacConfig Lib "sdDaqApi.dll" Alias _
    "_sdDaqDacConfig@24" (ByVal onOffCh1_ As Long, ByVal onOffCh2_ As Long,
ByVal modeCh1_ As Double, ByVal modeCh2_ As Long, ByVal sampRate_ As Long,
ByVal indexModule_ As Long) As Long

Declare Function sdDaqDacStart Lib "sdDaqApi.dll" Alias _
    "_sdDaqDacStart@8" (ByVal startOnOff_ As Long, ByVal indexModule_ As Long)
As Long
Declare Function sdDaqDacSetFormTable Lib "sdDaqApi.dll" Alias _
    "_sdDaqDacSetFormTable@24" (ByVal channel_ As Long, ByVal function_ As
Long, ByVal frequency_ As Long, ByVal amplitude_ As Long, ByVal offset_ As Long,
ByVal indexModule_ As Long) As Long

Declare Function sdDaqDacSetUserTable Lib "sdDaqApi.dll" Alias _
    "_sdDaqDacSetUserTable@16" (ByVal channel_ As Long, ByVal bufferSize_ As
Long, ByRef tableBuffer_ As String, ByVal indexModule_ As Long) As Long

Declare Function sdDaqDacTriggerLevel Lib "sdDaqApi.dll" Alias _
    "_sdDaqDacTriggerLevel@12" (ByVal channel_ As Long, ByVal trigLevel_ As Long,
ByVal indexModule_ As Long) As Long

Declare Function sdDaqTransferData Lib "sdDaqApi.dll" Alias _
    "_sdDaqTransferData@8" (ByRef dataPoint_ As Long, ByVal indexModule_ As
Long) As Long

Declare Function sdDaqTransferPreDataFile Lib "sdDaqApi.dll" Alias _
    "_sdDaqTransferPreDataFile@8" (ByVal fPath_ As String, ByVal indexModule_ As
Long) As Long
```

```
Declare Function sdDaqTransferDataFile Lib "sdDaqApi.dll" Alias _
    "_sdDaqTransferDataFile@4" (ByVal indexModule_ As Long) As Long

Declare Function sdDaqTransferEndDataFile Lib "sdDaqApi.dll" Alias _
    "_sdDaqTransferEndDataFile@4" (ByVal indexModule_ As Long) As Long

Declare Function sdDaqTransferDataStop Lib "sdDaqApi.dll" Alias _
    "_sdDaqTransferDataStop@4" (ByVal indexModule_ As Long) As Long

Declare Function sdDaqStart Lib "sdDaqApi.dll" Alias _
    "_sdDaqStart@8" (ByVal startOnOff_ As Long, ByVal indexModule_ As Long) As
Long

Declare Function sdDaqWriteUserData Lib "sdDaqApi.dll" Alias _
    "_sdDaqWriteUserData@12" (ByRef data_ As String, ByVal address_ As Long,
ByVal indexModule_ As Long) As Long

Declare Function sdDaqReadUserData Lib "sdDaqApi.dll" Alias _
    "_sdDaqReadUserData@12" (ByRef data_ As String, ByVal address_ As Long,
ByVal indexModule_ As Long) As Long

Declare Function sdDaqStackConfig Lib "sdDaqApi.dll" Alias _
    "_sdDaqStackConfig@12" (ByVal onOff_ As Long, ByVal multiMode_ As Long,
ByVal indexModule_ As Long) As Long
```

<sdDaqApiType.bas>

```
Option Explicit

'enum {MAX_CHANNL_NUM = 5};
Global Const MAX_CHANNL_NUM As Long = 5

'enum {MAX_AI_NUM = 4};
Global Const MAX_AI_NUM As Long = 4

'enum {MAX_AO_NUM = 2};
Global Const MAX_AO_NUM As Long = 2

' DAC STOP & START
Global Const SD_STOP As Long = 0
Global Const SD_START As Long = 1

' ON & OFF
Global Const SD_OFF As Long = 0
Global Const SD_ON As Long = 1

' COUNTER ON&OFF
Global Const SD_COUNTER_OFF As Long = 0
Global Const SD_COUNTER_ON As Long = 1

' CHANNEL
Global Const SD_CHANNEL_1 As Long = 0
Global Const SD_CHANNEL_2 As Long = 1
Global Const SD_CHANNEL_3 As Long = 2
Global Const SD_CHANNEL_4 As Long = 3
Global Const SD_CHANNEL_COUNTER As Long = 4

' DAC_FUNCTION
Global Const SD_DAC_DC As Long = 0
Global Const SD_DAC_SIN As Long = 1
Global Const SD_DAC_SQU As Long = 2
```

```
Global Const SD_DAC_TRI As Long = 3
Global Const SD_DAC_RAMP As Long = 4
Global Const SD_DAC_NOISE As Long = 5

' TRIGGER_SOURCE
Global Const SD_DIGITAL As Long = 1
Global Const SD_ANALOG As Long = 2
Global Const SD_COUNTER As Long = 4

' ADC_COUPLING
Global Const SD_AC As Long = 0
Global Const SD_DC As Long = 1

' SAMPLING_MODE
Global Const SD_SAMPLING_CONTINUOUS_MODE As Long = 0
Global Const SD_SAMPLING_N_SAMPLE_MODE As Long = 1
Global Const SD_SAMPLING_TRIGGER_MODE As Long = 2

' TRIGGER_MODE
Global Const SD_TRIG_CONTINUOUS_MODE As Long = 0
Global Const SD_TRIG_SINGLE_N_SAMPLING_MODE As Long = 1
Global Const SD_TRIG_EVERY_N_SAMPLING_MODE As Long = 2

' TRIGGER_PREPOST
Global Const SD_PREPOST_NONE As Long = 0
Global Const SD_POST_SAMPLING As Long = 2
Global Const SD_PRE_SAMPLING As Long = 4

' TRIGGER_EDGE
Global Const SD_RISING_EDGE As Long = 0
Global Const SD_FALLING_EDGE As Long = 1

'DEVICE_NUMBER
Global Const SD_DEVICE_NUMBER As Long = 0
```

```
'ANY CONSTANT
Global Const LOWMIN As Long = -8388608

Global Const LOWMAX As Long = 8388607


'Usb Daq Error Type
Global Const ERROR_NONE                    As Long = 1        ' NO Error
Global Const ERROR_USB_DRIVER              As Long = -1       ' USB Driver Error
Global Const ERROR_DEVICE_NOT_FOUND        As Long = -2       ' Device not
Found
Global Const ERROR_USB_DRIVER_READ         As Long = -3       ' USB Driver
Read Error
Global Const ERROR_USB_DRIVER_WRITE        As Long = -4       ' USB Driver
Write Error
Global Const ERROR_DEVICE_OPEN             As Long = -5       ' Device Open
Error
Global Const ERROR_DEVICE_CLOSE            As Long = -6       ' Device Close
Error
Global Const ERROR_DAQ_START_STOP          As Long = -10      ' Sampling
Start/stop Error
Global Const ERROR_ADC_START_STOP          As Long = -11      ' ADC
Start/stop  Error
Global Const ERROR_ADC_SET                 As Long = -12      ' ADC
SET(Sample Rate   High Pass Filter   Zero Calibration   Pre Sample) Error
Global Const ERROR_ADC_SET_READ            As Long = -13      ' ADC
SET(Sample Rate   High Pass Filter   Zero Calibration   Pre Sample) READ Error
Global Const ERROR_ADC_CHANNEL_SET         As Long = -14      ' ADC Channel
Select Error
Global Const ERROR_ADC_ICP_SET             As Long = -15      ' ADC ICP Select
Error
Global Const ERROR_ADC_LP_SET              As Long = -16      ' ADC Low Pass
Filter Error
```

```
Global Const ERROR_SAMPLING_SET              As Long = -20     ' Sampling Time
(trigger   interval   number) Error

Global Const ERROR_DAC_START_STOP            As Long = -30     ' Analog Out
Start/Stop Error
Global Const ERROR_DAC_PULSE_SET             As Long = -31     ' Analog Out
Pulse Error
Global Const ERROR_DAC_TABLE_NUM             As Long = -32     ' Analog Out
Table Number Error
Global Const ERROR_DAC_WRITE_DATA            As Long = -33
Global Const ERROR_DAC_SET                   As Long = -34

Global Const ERROR_TRIG_SET                  As Long = -40     ' Trigger On/Off
Global Const ERROR_TRIG_DIGITAL              As Long = -41     ' Digital Trigger
Set Error
Global Const ERROR_TRIG_ANALOG               As Long = -42     ' Analog Trigger
Set Error
Global Const ERROR_TRIG_ANALOG_LEVEL         As Long = -43     ' Analog Trigger
Level Error
Global Const ERROR_TRIG_COUNTER_LEVEL        As Long = -44     ' Counter
Trigger Level Error
Global Const ERROR_TRIG_PRE_SAMPLING_NUM     As Long = -45     ' Pre Sampling
Number Error

Global Const ERROR_READ_CONFIG_DATA          As Long = -50     ' Configuration
Data Read Error
Global Const ERROR_READ_CALIB_DATA_AMP       As Long = -51     ' AMP
Calibration Data Read Error
Global Const ERROR_WRITE_CALIB_DATA_AMP      As Long = -52     ' AMP
Calibration Data Write Error
Global Const ERROR_READ_CALIB_DATA_OFFSET    As Long = -53     ' Offset
Calibration Data Read Error
Global Const ERROR_WRITE_CALIB_DATA_OFFSET   As Long = -54     ' Offset
Calibration Data Write Error
Global Const ERROR_READ_ADC_RESET_STATUS     As Long = -55
```

```
Global Const ERROR_READ_MEMORY_DATA_NUM      As Long = -60      ' Read
Memory Data Error
Global Const ERROR_READ_COUNTER_VALUE        As Long = -61      ' Read
Counter Value Error

Global Const ERROR_FPGA_CONFIG_START         As Long = -70      ' FPGA
Configuration Error
Global Const ERROR_FPGA_WRITE_CONFIG         As Long = -71      ' Write FPGA
Configuration Error
Global Const ERROR_FPGA_CHECK_CONFIG         As Long = -72      ' Check FPGA
Configuration Error

Global Const ERROR_READ_VERSION              As Long = -80      ' Read Version
Error

Global Const ERROR_WRITE_USER_MULTI_DATA     As Long = -90      ' User Specific
Mulit Data Write Error
Global Const ERROR_READ_USER_MULTI_DATA      As Long = -91      ' User Specific
Multi Data Read Error
Global Const ERROR_WRITE_USER_DATA           As Long = -92      ' User Specific
Data Write Error
Global Const ERROR_READ_USER_DATA            As Long = -93      ' User Specific
Data Read Error
Global Const ERROR_WRITE_EEPROM              As Long = -94      ' Write EEPROM
Error
Global Const ERROR_READ_EEPROM               As Long = -95      ' Read EEPROM
Error
Global Const ERROR_READ_CURRENT_SPEED        As Long = -96      ' Read
Current Speed Error
Global Const ERROR_ADC_CONSTRAINT_STOP       As Long = -97      ' Read ADC
Constraint Stop Error

Global Const ERROR_RESET_FX2_FIFO            As Long = -110
Global Const ERROR_CHANGE_FX2_MODE           As Long = -111
```

```
Global Const ERROR_FILE_OPEN                As Long = -150        ' FILE OPEN
Error

Global Const ERROR_MULTI_SET                As Long = -160      ' Multi Stack Error
```

SDU2040Test_1.ftm

```
Private Sub Command1_Click()
Dim Res As Long

    ' Setting Variables
    ' CHANNEL
    Res = sdDaqAdcChannelConfig(SD_CHANNEL_1, SD_ON, 1#, 0#, "V", SD_DC,
SD_OFF, 0)
    If (Res <= 0) Then
        MsgBox "Error : Analog Input Channel Configuration"
        Call sdDaqFinalize
        End
    End If

    For G = SD_CHANNEL_2 To MAX_AI_NUM - 1
        Res = sdDaqAdcChannelConfig(G, SD_OFF, 1#, 0#, "V", SD_DC, SD_OFF, 0)
        If (Res <= 0) Then
            MsgBox "Error : Analog Input Channel Configuration"
            End
        End If
    Next G

    ' COUNTER
    ' Turn Off
    Res = sdDaqCounterConfig(SD_OFF, 0)
    If (Res <= 0) Then
        MsgBox "ERROR : Counter Configuration"
        Call sdDaqFinalize
        End
    End If

    ' TRIGGER
    Res = sdDaqTriggerConfig(SD_ANALOG, 100, SD_RISING_EDGE, SD_CHANNEL_1,
SD_PRE_SAMPLING, 50, 0)
```

```
If (Res <= 0) Then
    MsgBox "ERROR : Trigger Configuration"
    Call sdDaqFinalize
    End
End If

' SAMPLING
Dim ResInt As Long
ResInt = sdDaqAdcSamplingConfig(8000, 4096,
SD_SAMPLING_CONTINUOUS_MODE, SD_TRIG_CONTINUOUS_MODE, SD_OFF, 0)
If (ResInt <= 0) Then
    MsgBox "Error : Sampling Configuration"
    Call sdDaqFinalize
    End
End If

' Save File, make head file

Dim filePath As String
filePath = "test_1.bin"

Res = sdDaqTransferPreDataFile(filePath, 0)
If (Res <= 0) Then
    MsgBox "Error : Making Header File"
    Call sdDaqFinalize
    End
End If

' ARM
Res = sdDaqArm(SD_START, 0)
If (Res <= 0) Then
    MsgBox "ERROR : DAQ Arm Start"
    Call sdDaqFinalize
    End
End If
```

```
' Setting Variables
' DAQ START
Res = sdDaqStart(SD_ON, 0)
If (Res <= 0) Then
      MsgBox "Error : Data Acquisition Start"
      Call sdDaqFinalize
      End
End If

Dim dataPoint_ As Long
Res = sdDaqTransferData(dataPoint_, 0)
If (Res <= 0) Then
      MsgBox "Error : Get Data"
      Call sdDaqFinalize
      End
End If

Res = sdDaqTransferDataFile(0)
    If (Res <= 0) Then
      MsgBox "Error : Data Transfer to File"
      Call sdDaqFinalize
      End
End If

'Stop Sampling
Res = sdDaqStart(SD_OFF, 0)
If (Res <= 0) Then
      MsgBox "Error : Data Acquisition Stop"
      Call sdDaqFinalize
      End
End If

'Stop Arm
Res = sdDaqArm(SD_STOP, 0)
If (Res <= 0) Then
```

```
            MsgBox "ERROR : DAQ Arm Stop"
            Call sdDaqFinalize
            End
        End If

        'Save File, Data transfer to the file
        Res = sdDaqTransferEndDataFile(0)
        If (Res <= 0) Then
            MsgBox "Error : Close File"
            Call sdDaqFinalize
            End
        Else: MsgBox "Success"
        End If

            'Close device 0
        Res = sdDaqCloseDevice(0)
        If (Res <= 0) Then
            MsgBox "Error : Close device"
            Call sdDaqFinalize
            End
        End If
    End Sub

    Private Sub Form_Load()
        Dim Res As Long
        Dim deviceNum_ As Long
        Res = sdDaqInitialize(deviceNum_)
        If (Res <= 0) Then
            MsgBox "Error : Initialize "
            End
        End If


        Dim G2 As Long
        For G = 0 To deviceNum_ - 1
```

```
            Res = sdDaqOpenDevice(G)
            If (Res <= 0) Then
                MsgBox "Error : Open Device "
                End
            End If
        Next G
    End Sub

    Private Sub Form_Unload(Cancel As Integer)
    Call sdDaqFinalize
    End Sub
```

SDU2040Test_2.ftm

```
Private Sub Command1_Click()
Dim Res As Long

    ' Setting Variables
    ' Analog Out Configuration
    Res = sdDaqDacConfig(SD_ON, SD_ON, SD_TABLE_OUT, SD_TABLE_OUT,
SD_96KSPS, 0)
    If (Res <= 0) Then
        MsgBox "ERROR : Analog out Configuration"
        Call sdDaqFinalize
        End
    End If

    ' COUNTER
    ' Analog Out Table Configuration Channel 1
    Res = sdDaqDacSetFormTable(SD_CHANNEL_1, SD_DAC_SQU, 100, 5000, 2500,
0)
    If (Res <= 0) Then
        MsgBox "Error : Analog Out Making Table"
        Call sdDaqFinalize
        End
    End If

    ' Analog Out Table Configuration Channel 2
    Res = sdDaqDacSetFormTable(SD_CHANNEL_2, SD_DAC_SIN, 100, 5000, 0, 0)
    If (Res <= 0) Then
        MsgBox "Error : Analog Out Making Table"
        Call sdDaqFinalize
        End
    End If

    ' Start Analog Out
    Res = sdDaqDacStart(SD_ON, 0)
```

```
    If (Res <= 0) Then
        MsgBox "ERROR : Analog out Start"
        Call sdDaqFinalize
        End
    End If
End Sub

Private Sub Command2_Click()
'Stop Analog Out
    Res = sdDaqDacStart(SD_OFF, 0)
    If (Res <= 0) Then
        MsgBox "ERROR : Analog out Stop"
        Call sdDaqFinalize
        End
    End If

        'Close device 0
    Res = sdDaqCloseDevice(0)
    If (Res <= 0) Then
        MsgBox "Error : Close device"
        Call sdDaqFinalize
        End
    Else: MsgBox "Success"
    End If
End Sub

Private Sub Form_Load()
    Dim Res As Long
    Dim deviceNum_ As Long
    Res = sdDaqInitialize(deviceNum_)
    If (Res <= 0) Then
        MsgBox "Error : Initialize "
        End
    End If
```

```
    Dim G2 As Long
    For G = 0 To deviceNum_ - 1
        Res = sdDaqOpenDevice(G)
        If (Res <= 0) Then
            MsgBox "Error : Open Device "
            End
        End If
    Next G
End Sub

Private Sub Form_Unload(Cancel As Integer)
    Dim Res As Long
    Res = sdDaqFinalize()
    Call sdDaqFinalize
End Sub
```

SDU2040Test_3.ftm

```
Private Sub Command1_Click()
Dim Res As Long

    ' Setting Variables
    ' CHANNEL
    For G = SD_CHANNEL_1 To MAX_AI_NUM - 1
        Res = sdDaqAdcChannelConfig(G, SD_OFF, 1#, 0#, "V", SD_DC, SD_OFF, 0)
        If (Res <= 0) Then
            MsgBox "Error : Analog Input Channel Configuration"
            End
        End If
    Next G

    ' COUNTER
    ' Turn Off
    Res = sdDaqCounterConfig(SD_ON, 0)
    If (Res <= 0) Then
        MsgBox "ERROR : Counter Configuration"
        Call sdDaqFinalize
        End
    End If

    ' TRIGGER
    Res = sdDaqTriggerConfig(SD_ANALOG, 100, SD_RISING_EDGE, SD_CHANNEL_1,
SD_PRE_SAMPLING, 50, 0)
    If (Res <= 0) Then
        MsgBox "ERROR : Trigger Configuration"
        Call sdDaqFinalize
        End
    End If

    ' SAMPLING
    Dim ResInt As Long
```

```
    ResInt = sdDaqAdcSamplingConfig(8000, 4096,
SD_SAMPLING_CONTINUOUS_MODE, SD_TRIG_CONTINUOUS_MODE, SD_OFF, 0)
    If (ResInt <= 0) Then
        MsgBox "Error : Sampling Configuration"
        Call sdDaqFinalize
        End
    End If

    ' Save File, make head file

    Dim filePath As String
    filePath = "test_3.bin"

    Res = sdDaqTransferPreDataFile(filePath, 0)
    If (Res <= 0) Then
        MsgBox "Error : Making Header File"
        Call sdDaqFinalize
        End
    End If

    ' ARM
    Res = sdDaqArm(SD_START, 0)
    If (Res <= 0) Then
        MsgBox "ERROR : DAQ Arm Start"
        Call sdDaqFinalize
        End
    End If

    ' Setting Variables
    ' DAQ START
    Res = sdDaqStart(SD_ON, 0)
    If (Res <= 0) Then
        MsgBox "Error : Data Acquisition Start"
        Call sdDaqFinalize
        End
```

```
End If

Dim dataPoint_ As Long
Res = sdDaqTransferData(dataPoint_, 0)
If (Res <= 0) Then
    MsgBox "Error : Get Data"
    Call sdDaqFinalize
    End
End If

Res = sdDaqTransferDataFile(0)
   If (Res <= 0) Then
    MsgBox "Error : Data Transfer to File"
    Call sdDaqFinalize
    End
End If

'Stop Sampling
Res = sdDaqStart(SD_OFF, 0)
If (Res <= 0) Then
    MsgBox "Error : Data Acquisition Stop"
    Call sdDaqFinalize
    End
End If

'Stop Arm
Res = sdDaqArm(SD_STOP, 0)
If (Res <= 0) Then
    MsgBox "ERROR : DAQ Arm Stop"
    Call sdDaqFinalize
    End
End If

'Save File, Data transfer to the file
Res = sdDaqTransferEndDataFile(0)
```

```
    If (Res <= 0) Then
        MsgBox "Error : Close File"
        Call sdDaqFinalize
        End
    Else: MsgBox "Success"
    End If

        'Close device 0
    Res = sdDaqCloseDevice(0)
    If (Res <= 0) Then
        MsgBox "Error : Close device"
        Call sdDaqFinalize
        End
    End If
End Sub

Private Sub Form_Load()
    Dim Res As Long
    Dim deviceNum_ As Long
    Res = sdDaqInitialize(deviceNum_)
    If (Res <= 0) Then
        MsgBox "Error : Initialize "
        End
    End If

    Dim G2 As Long
    For G = 0 To deviceNum_ - 1
        Res = sdDaqOpenDevice(G)
        If (Res <= 0) Then
            MsgBox "Error : Open Device "
            End
        End If
    Next G
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
Call sdDaqFinalize
End Sub
```

SDU2040Test_4.ftm

```
Private Sub Command1_Click()
Dim Res As Long

    ''''''''''''''''''''''
    ' Setting Variables
    ' CHANNEL
    Res = sdDaqAdcChannelConfig(SD_CHANNEL_1, SD_ON, 1#, 0#, "V", SD_DC,
SD_OFF, 0)
    If (Res <= 0) Then
        MsgBox "Error : Analog Input Channel Configuration"
        Call sdDaqFinalize
        End
    End If

    For G = SD_CHANNEL_2 To MAX_AI_NUM - 1
        Res = sdDaqAdcChannelConfig(G, SD_OFF, 1#, 0#, "V", SD_DC, SD_OFF, 0)
        If (Res <= 0) Then
            MsgBox "Error : Analog Input Channel Configuration"
            End
        End If
    Next G

    ' COUNTER
    ' Turn Off
    Res = sdDaqCounterConfig(SD_OFF, 0)
    If (Res <= 0) Then
        MsgBox "ERROR : Counter Configuration"
        Call sdDaqFinalize
        End
    End If

    ' TRIGGER
    Res = sdDaqTriggerConfig(SD_ANALOG, 100, SD_RISING_EDGE, SD_CHANNEL_1,
```

```
SD_PRE_SAMPLING, 50, 0)
    If (Res <= 0) Then
        MsgBox "ERROR : Trigger Configuration"
        Call sdDaqFinalize
        End
    End If

    ' SAMPLING
    Dim ResInt As Long
    ResInt = sdDaqAdcSamplingConfig(8000, 1024,
SD_SAMPLING_CONTINUOUS_MODE, SD_TRIG_CONTINUOUS_MODE, SD_OFF, 0)
    If (ResInt <= 0) Then
        MsgBox "Error : Sampling Configuration"
        Call sdDaqFinalize
        End
    End If

    ' Save File, make head file

    Dim filePath As String
    filePath = "test_4.bin"

    Res = sdDaqTransferPreDataFile(filePath, 0)
    If (Res <= 0) Then
        MsgBox "Error : Making Header File"
        Call sdDaqFinalize
        End
    End If

    ' ARM
    Res = sdDaqArm(SD_START, 0)
    If (Res <= 0) Then
        MsgBox "ERROR : DAQ Arm Start"
        Call sdDaqFinalize
        End
```

```
End If

'Sleep(100);

      ''''''''''''''''''''''

' Setting Variables
' DAQ START
Res = sdDaqStart(SD_ON, 0)
If (Res <= 0) Then
      MsgBox "Error : Data Acquisition Start"
      Call sdDaqFinalize
      End
End If

Dim readDataCounter As Long

readDataCounter = sdDaqTransferData(buffer, 0)
If (readDataCounter <= 0) Then
      MsgBox "Error : Get Data"
      Call sdDaqFinalize
      End
End If

'Dim n&
'readDataCounter = sdDaqTransferData(n&, 0)
'MsgBox n



' Save File, Data transfer to the file
Res = sdDaqTransferDataFile(0)
     If (Res <= 0) Then
      MsgBox "Error : Data Transfer to File"
      Call sdDaqFinalize
      End
End If
```

```
' Save As .CSV File Format
Dim tempBuffer() As Long

ReDim tempBuffer(readDataCounter)

Dim c As Long
Dim d As Integer

Dim i As Integer
Dim fp As Long

fp = FreeFile
Open "Test_4.csv" For Output As fp

'ReDim tempDbl(readDataCounter) As Double
'Dim tempVar As Double

For i = 0 To readDataCounter - 1
    CopyMemory tempBuffer(i), ByVal (VarPtr(buffer) + i), LenB(d)

Next i


ReDim tempDbl(readDataCounter) As Double
Dim tempVar As Double


For i = 0 To readDataCounter - 1
    tempDbl(i) = tempBuffer(i)
    tempVar = Format(10# * tempDbl(i) / LOWMAX, "00000.000")
    Print #fp, tempVar

Next i

Close fp
```

```
        'Stop Sampling
        Res = sdDaqStart(SD_OFF, 0)
        If (Res <= 0) Then
            MsgBox "Error : Data Acquisition Stop"
            Call sdDaqFinalize
            End
        End If

        'Stop Arm
        Res = sdDaqArm(SD_STOP, 0)
        If (Res <= 0) Then
            MsgBox "ERROR : DAQ Arm Stop"
            Call sdDaqFinalize
            End
        End If

        'Save File, Data transfer to the file
        Res = sdDaqTransferEndDataFile(0)
        If (Res <= 0) Then
            MsgBox "Error : Close File"
            Call sdDaqFinalize
            End
        Else: MsgBox "Success"
        End If

            'Close device 0
        Res = sdDaqCloseDevice(0)
        If (Res <= 0) Then
            MsgBox "Error : Close device"
            Call sdDaqFinalize
            End
        End If
    End Sub

    Private Sub Form_Load()
```

```
    Dim Res As Long
    Dim deviceNum_ As Long
    Res = sdDaqInitialize(deviceNum_)
    If (Res <= 0) Then
        MsgBox "Error : Initialize "
        End
    End If


    Dim G2 As Long
    For G = 0 To deviceNum_ - 1
        Res = sdDaqOpenDevice(G)
        If (Res <= 0) Then
            MsgBox "Error : Open Device "
            End
        End If
    Next G
End Sub

Private Sub Form_Unload(Cancel As Integer)
Call sdDaqFinalize
End Sub
```

SDU2040Test_5.ftm

```
Private Sub Command1_Click()
Dim Res As Long

    ' Setting Variables
    ' CHANNEL
    Res = sdDaqAdcChannelConfig(SD_CHANNEL_1, SD_ON, 1#, 0#, "V", SD_DC,
SD_OFF, 0)
    If (Res <= 0) Then
        MsgBox "Error : Analog Input Channel Configuration"
        Call sdDaqFinalize
        End
    End If

    For G = SD_CHANNEL_2 To MAX_AI_NUM - 1
        Res = sdDaqAdcChannelConfig(G, SD_OFF, 1#, 0#, "V", SD_DC, SD_OFF, 0)
        If (Res <= 0) Then
            MsgBox "Error : Analog Input Channel Configuration"
            End
        End If
    Next G

    ' COUNTER
    ' Turn Off
    Res = sdDaqCounterConfig(SD_OFF, 0)
    If (Res <= 0) Then
        MsgBox "ERROR : Counter Configuration"
        Call sdDaqFinalize
        End
    End If

    ' TRIGGER
    Res = sdDaqTriggerConfig(SD_ANALOG, 100, SD_RISING_EDGE, SD_CHANNEL_1,
```

```
SD_PRE_SAMPLING, 50, 0)
    If (Res <= 0) Then
        MsgBox "ERROR : Trigger Configuration"
        Call sdDaqFinalize
        End
    End If

    ' SAMPLING
    Dim ResInt As Long
    ResInt = sdDaqAdcSamplingConfig(8000, 1024,
SD_SAMPLING_CONTINUOUS_MODE, SD_TRIG_CONTINUOUS_MODE, SD_OFF, 0)
    If (ResInt <= 0) Then
        MsgBox "Error : Sampling Configuration"
        Call sdDaqFinalize
        End
    End If

    ' Save File, make head file

    Dim filePath As String
    filePath = "test_5.bin"

    Res = sdDaqTransferPreDataFile(filePath, 0)
    If (Res <= 0) Then
        MsgBox "Error : Making Header File"
        Call sdDaqFinalize
        End
    End If

    ' ARM
    Res = sdDaqArm(SD_START, 0)
    If (Res <= 0) Then
        MsgBox "ERROR : DAQ Arm Start"
        Call sdDaqFinalize
        End
```

```
End If

'Sleep(100);

' Setting Variables
' DAQ START
Res = sdDaqStart(SD_ON, 0)
If (Res <= 0) Then
      MsgBox "Error : Data Acquisition Start"
      Call sdDaqFinalize
      End
End If

'Dim buffer As Long
Dim readDataCounter As Long
readDataCounter = sdDaqTransferData(buffer, 0)
If (readDataCounter <= 0) Then
      MsgBox "Error : Get Data"
      Call sdDaqFinalize
      End
End If

Res = sdDaqTransferDataFile(0)
    If (Res <= 0) Then
      MsgBox "Error : Data Transfer to File"
      Call sdDaqFinalize
      End
End If


Dim tempBufferAnalogInput() As Long
ReDim tempBufferAnalogInput(readDataCounter)
Dim tempBufferCounter() As Long
ReDim tempBufferCounter(readDataCounter)
```

```
Dim c As Long
Dim I As Long
For I = 0 To readDataCounter - 1
    CopyMemory tempBufferAnalogInput(I), ByVal (VarPtr(buffer) + I * 2), LenB(c)
    CopyMemory tempBufferCounter(I), ByVal (VarPtr(buffer) + I * 2 + 1), LenB(c)
Next I

Dim fp As Long
fp = FreeFile
Open "Test_4.csv" For Output As fp

For I = 0 To readDataCounter - 1
    Print #fp, 10# * tempBufferAnalogInput(I) / LOWMAX
    Print #fp, tempBufferCounter(I)
Next I
Close fp

'Stop Sampling
Res = sdDaqStart(SD_OFF, 0)
If (Res <= 0) Then
    MsgBox "Error : Data Acquisition Stop"
    Call sdDaqFinalize
    End
End If

'Stop Arm
Res = sdDaqArm(SD_STOP, 0)
If (Res <= 0) Then
    MsgBox "ERROR : DAQ Arm Stop"
    Call sdDaqFinalize
    End
End If

'Save File, Data transfer to the file
Res = sdDaqTransferEndDataFile(0)
```

```
    If (Res <= 0) Then
        MsgBox "Error : Close File"
        Call sdDaqFinalize
        End
    Else: MsgBox "Success"
    End If

        'Close device 0
    Res = sdDaqCloseDevice(0)
    If (Res <= 0) Then
        MsgBox "Error : Close device"
        Call sdDaqFinalize
        End
    End If
End Sub

Private Sub Form_Load()
    Dim Res As Long
    Dim deviceNum_ As Long
    Res = sdDaqInitialize(deviceNum_)
    If (Res <= 0) Then
        MsgBox "Error : Initialize "
        End
    End If

    Dim G2 As Long
    For G = 0 To deviceNum_ - 1
        Res = sdDaqOpenDevice(G)
        If (Res <= 0) Then
            MsgBox "Error : Open Device "
            End
        End If
    Next G
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
Call sdDaqFinalize
End Sub
```

15. Specification

(1) General specification

Environment :

Operating : 0℃~50℃, 10% ~ 80% RH.

Input Power : USB Powered 5V DC

Computer Communication : USB Interface

Warm-up : 20 minute

(2) Analog Specifications

Analog inputs :

Channels : 4 Channels

Input Connector : BNC Connector

Input Configuration : Unbalanced Differential

Resolution : 24Bit

Over Voltage Protection : 42Vpeak

Offset Voltage : ±3Mv

Bandwidth : 50kHz

Type of ADC : Delta Sigma

Sampling Rate : Max 216KS/sec

AC Cutoff Frequency : 3.5Hz

Input Impedance : 1MΩ

Input Coupling : AC/DC Coupling

Input Range : Max ±10V

Low-Pass Filter :

Pass Band :

10S/sec ~ 4KS/sec :

8KS/sec ~ 216KS/sec :

Stop Band :

10S/sec ~ 4KS/sec :

8KS/sec ~ 216KS/sec :

Alias Rejection :

10S/sec ~ 4KS/sec :

8KS/sec ~ 216KS/sec :

Amplitude Accuracy : (Fin < Fc/2)

Total Harmonic Distortion :

ICP Bias Current : 4mA

Analog outputs :

Channel : 2 Channels

Signal Connection : BNC Connector

Frequency Range :

Amplitude Setting : Max ±10Vpp

Output Impedance : 50Ω

Waveform Mode : SINE, SQUA, TRIA, RAMP, DC

Counter

Channels : 1 Channel

Connector : BNC Connector

Input Level : TTL Compatible

External Trigger :

Channels : 1 Channel

Connector : BNC Connector

Input Level : TTL Compatible

FCC NOTICE

THIS DEVICE COMPLIES WITH PART 15 OF THE FCC RULES.
OPERATION IS SUBJECT TO THE FOLLOWING TWO CONDITION:
(1) THIS DEVICE MAY NOT CAUSE HARMFUL INTERFERENCE, AND
(2) THIS DEVICE MUST ACCEPT ANY INTERFERENCE RECEIVED, INCLUDING INTERFERENCE THAT MAY CAUSE UNDERSIRED OPERATION.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation.

This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communication. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures :

- Reorient or relocate the receiving antenna.

- Increase the separation between the equipment and receiver.

- Connect the equipment into an outlet on a circuit difference from that to which the receiver is connected.

- Consult the dealer of an experienced radio/TV technician for help.

NOTE : The manufacturer is not responsible for any radio or TV interference caused by unauthorized modifications to this equipment. Such modifications could void the user’s authority to operate the equipment.