IRU151-I

Software Development Kit

Reference Manual

Revision History

Version	Revised Date	Author	Description
1.0	2018/12/05	Hank	- 1st release

How to Use This Document

This document is written to provide the information of the SDK and helps users implement their own applications

- Chapter 1, “Basic Structure” introduces the fundamental information for applying the functions in this document.
- Chapter 2, “Function Description” introduces the detailed information of the functions provided by the SDK.
- Appendix A, ‘‘Error Code’’ describes the meaning of the error code returned by the functions.
- Appendix B, ‘‘Function Table’’ shows all IRU151 function.

Table of Contents

This page is intentionally left blank.

CHAPTER 1
Introduction

This chapter introduces the information of the SDK.

1.1 The Basic of the Programming

To create an application with SDK, please follow the steps below.

Step 1 : Include IRU head file.

When user program a IRU application to call IRU API, user has to include "libiru.h" and "axiomtek_err.h" files.

"libiru.h" files is for API and control IRU device.

and "axiomtek_err.h" files. Is error code respectively.

It can help user to analyze what problem is.

```
#include "libiru.h"
#include "axiomtek_err.h"
```

Step 2 : Create a device

```
int          ret;
ret = Open_Device("IRU151");

if(ret != AXIO_OK)
   printf("Create Device Failure\n");
```

(Continue)

Step 3 : Get information from device via API

```
int         i=0;
char        *pDevInfo;
short       length;
(Continue)
int         modelidx = 1;

ret = GetDevInfo(modelidx, &pDevInfo, &length);

if(ret == AXIO_OK){
      for(i=0;i<length;i++)
                printf("%c", toascii( *(pDevInfo +i)));
       printf("\n");
} else
    printf("Get Device info failure\n");
```

Step 4 : Close a device.

```
Close_Device();
```

Step 5 : The application can be closed

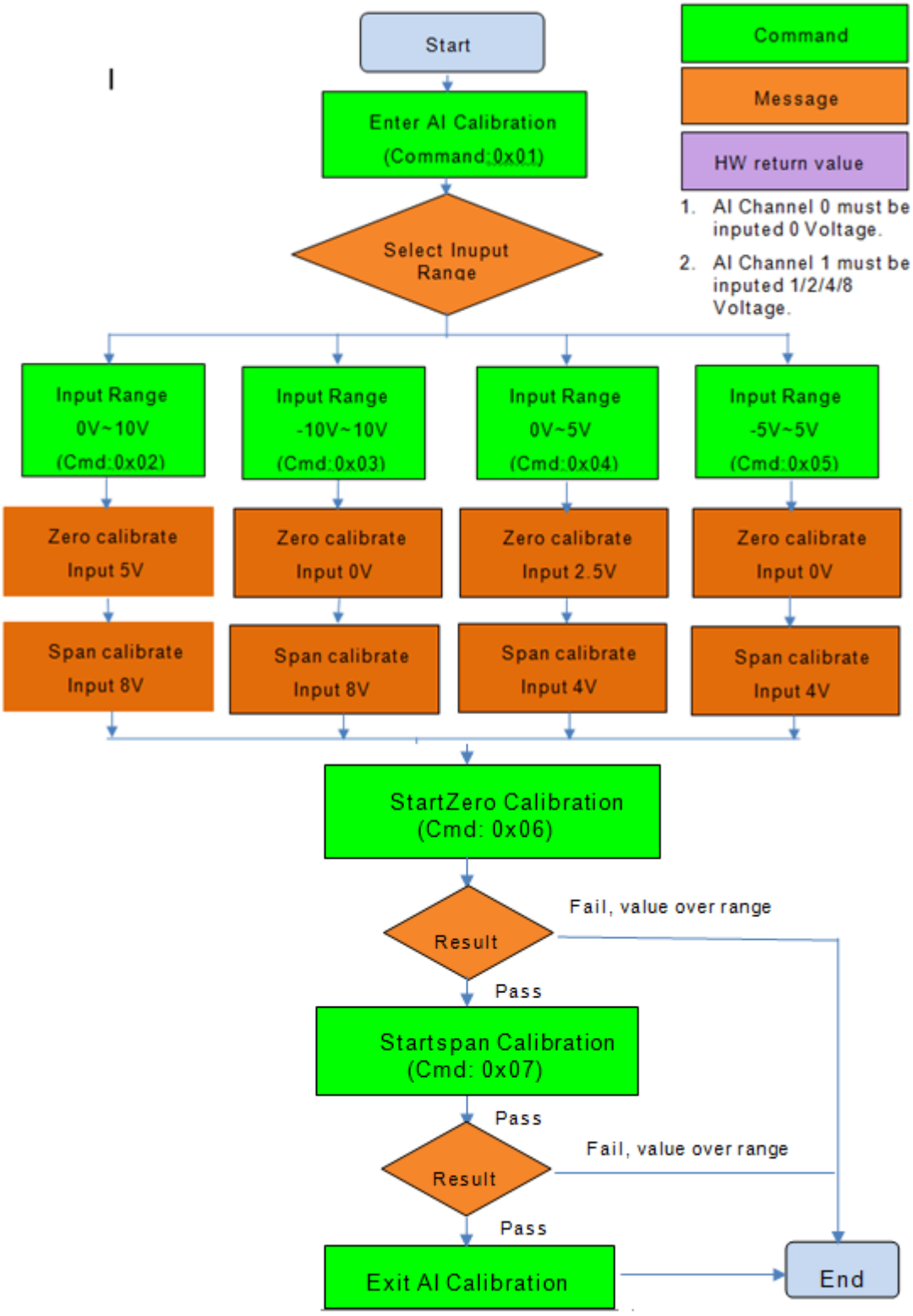
1.2 Create a Receiver Data Example

This example shows how to receive DI data. The sample is for a digital I/O receiver.

```
#define AXIO_OK 0x00

int         channel = 1;
int         filter = 0
int         trig = 2;
int         stop;
int         ret;

ret = Open_Device("IRU151");
if(ret != AXIO_OK){
            printf("open iru151 failure ... \n");
            return;
}

while(1)
{
    switch(stop){
            case 0:
                        ret = EnDIReceiver(channel, filter, trig);
                        if(ret != AXIO_OK)
                                    printf("Enable DI Receiver failure \n");
                        break;
            case 1:
                        ret = DisDIReceiver(channel);
                        if(ret != AXIO_OK)
                                    printf("Enable DI Receiver failure \n");
                        break;
            case 2:
                        printf("Exit \n");
                        break;
    }
```

```
        if(stop == 2){
                break;
        }
    }
    Close_Device();
```

1.3 The Calibration Flow Chart



This page is intentionally left blank.

CHAPTER 2
Function Description

This chapter describes the detail information of the SDK functions.

2.1 Open_Device

- **Description**

  Open the communication channel of I/O module.

- **Definition**

```
int Open_Device (
            char        *devName
    );
```

- **Parameters**

  *devName [in]: The name of target module. (IRU151/IR152)

- **Return value**

  AXIO_OK if successful. Other value represent the error. (See Error Code)

- **Example**

```
int             ret;

ret = Open_Device( “IRU151”);

if(ret != AXIO_OK)
    printf (“Open Device failure\n”);
```

2.2 CloseDevice

- **Description**

  Close the communication channel of I/O module.

- **Definition**

  int Close_Device ();

- **Parameters**

  NULL

- **Return value**

  AXIO_OK if successful. Other value represent the error. (See Error Code)

- **Example**

  Int ret = 0;

  ret = Close_Device ();

2.3 GetDevInfoEx

- **Description**

  Get the device information.

- **Definition**

```
int GetDevInfoEx(
        int     infoIndex,
        char    *info,
        int     *infoLen
    );
```

- **Parameters**

infoIndex	[in]:	This information index. 0x00 : Model Name 0x01 : Manufacture 0x02 : Firmware version
*info	[out]:	The information string.
*infoLen	[out]:	The length of information string.

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int     ret;
int     infoLen = 0;
char    info[128] = "";

ret = GetDevInfo(0x00, info, &infoLen);
```

(Continue)

```
if (ret == AXIO_OK)
{
    printf(“%s\n”, info);
}
else
{
    printf("Get Device info failure\n");
}
```

2.4 EnAIReceiver

- **Description**

  Enable the receiver of the selected AI receiver.

- **Definition**

  int EnAIReceiver (void);

- **Parameters**

  NULL

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int             ret;

ret = EnAIReceiver ();

if(ret != AXIO_OK)
    printf(“Enable AI Receiver failure\n”);
```

2.5 DisAIReceiver

- **Description**

  Disable the receiver of the selected AI channel.

- **Definition**

  int DisAIReceiver (void);

- **Parameters**

  NULL

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int             ret;

ret = DisAIReceiver ();

if(ret != AXIO_OK)
    printf("Disable AI Receiver failure\n");
```

2.6 ReadAIDataEx

- **Description**

  Read AI data from SDK

- **Definition**

```
void        ReadAIDataEx(
            int             size,
            char            *data,
            int             *length
    );
```

- **Parameters**

size	[in]: The buffer size.
*data	[out]: The AI data. (For example: Setting AI channel are 0, 1, 2, the input data will put buffer by order. The data of channel 0 is stored in data[0] and and data[1], the data of channe 1 is stroed in data[2] and and data[3] , the data of channel 2 is stored in data[4] and and data[5] and so on.
* length	[out]: The length of AI data.

- **Return value**

  NULL

- **Example**

```
int         ret = 0;
int         I = 0;
char        buffer[1024 * 32] = “”;
int         length = 0;

ret = ReadAIDataEx(1024*32, buffer, &length);
```

(Continue)

```
if (length > 0)
{

    for(i=0;i< length;i++)
            printf("0x%02x ", data[i]);
    printf("\n");
}
```

2.7 ReadAIVoltEx

- **Description**

  Read voltage data from SDK

- **Definition**

```
void        ReadAIVoltEx (
            int             size,
            double          *data,
            int             * length
    );
```

- **Parameters**

size	[in]: The buffer size.
*data	[out]: The voltage data (For example: Setting AI channel are 0, 1, 3, the input data will put buffer by order. The data of channel 0 is stored in data[0], the data of channel 1 is stroed in data[1], the data of channel 3 is stored in data[2] and so on.
*length	[out]: The length of voltage data.

- **Return value**

  NULL

- **Example**

```
int         ret = 0;
int         i = 0;
double      data[1024];
int         length = 0;

ret = ReadAIVoltEx (1024, data, &length);



(Continue)


if (length > 0)
{
    for(i=0;i< DataLength;i++)
            printf("%lf \n", data[i]);
    printf("\n");
}
```

2.8 SetAITrigConfEx

- **Description**

  Set up the AI trigger configure for channel 0.

- **Definition**

```
int SetAITrigConfEx(
        int         mode,
        int         source,
        int         condition,
        int         level
    );
```

- **Parameters**

mode	Trigger mode
	0x00 : Auto run.
	0x01 : Post trigger
source	Triger source
	0x00 : AI channel 0 (Analog trigger)
	0x01 : DI channel 0 (Digital trigger)
condition	Trigger condition
	0x00 : Rising edge
	0x01 : Falling edge
level:	Trigger voltage. (4500 => 4.5V)
	(The value is dependent on input range and the unit is 1mV.)

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

(Continue)

- **Example**

```
int         ret = 0;
int         mode = 0x1;
int         source = 0x0;
int         condition = 0x0;
int         level = 2500;       /* 2.5 voltage */

ret = SetAITrigConfEx(mode, source, condition, levelV);

if(ret != AXIO_OK)
    printf("Set AI Trigger configure failure\n");
```

2.9 SetAIChannel

- **Description**

  Select the input channel.

- **Definition**

```
int SetAIChannel (
            short       channel
    );
```

- **Parameters**

  channel [in]: Bit0-7 indicates channel0-7.

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int         ret;
short       channel = 0x01;

ret  = SetAIChannel (channel);

if(ret != AXIO_OK)
    printf("Set AI channel failure\n");
```

2.10 SetAISampleRate

- **Description**

  Set AI sample rate.

  (This API is not support on IRU151)

- **Definition**

```
int SetAISampleRate (
            Int             divisor
    );
```

- **Parameters**

  divisor [in]: The divisor of sample rate..

  The internal clock source is 88.67M.

  Sample rate = 88.67M / divisor (24 bit)

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int         ret;
int         nDivisor = 8867;

ret = SetAISampleRate (nDivisor);

 if(ret != AXIO_OK){
    printf("Set AI sample rate failure\n");
```

2.11 SetAInputRange

- **Description**

  Set AI input range. There are 4 types to be chosen.

  (IRU151 only supports -5~+5V, -10~+10V, 0~20 mA)

- **Definition**

```
int SetAInputRange (
            short           range
    );
```

- **Parameters**

range	Input voltage range
	0 : 0V ~ +5V
	1 : 0V ~ +10V
	2 : -5V ~ +5V
	3 : -10V ~ +10V
	14: 0~+20 mA (0~+5V)
	15: -20~+20 mA(-5V~+5V)

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int         ret;
short       range = 0;

ret = SetAInputRange (range);

if(ret != AXIO_OK){
    printf("Set AI Input range failure\n");
```

2.12 FactoryCalibratedRestore

- **Description**

  Reset the AI calibration value to default value.

- **Definition**

  int FactoryCalibratedRestore(void);

- **Parameters**

  NULL

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int             ret;

ret = FactoryCalibratedRestore ();

if(ret != AXIO_OK){
    printf("Reset AI Calibration factor failure\n");
```

2.13 ReadCaliFactors

- **Description**

  Read clalibrated value from flash.

- **Definition**

```
int ReadCaliFactory (
        int             mode,
        int             size,
        CALI_FACTORS    *factors,
        Int             *length
    );
```

- **Parameters**

mode	[in]: 0: User mode, 1: Factory mode
*size	[in]: The buffer size of factors.
*factors	[out]: The factors buffer.
*length	[out]: The length of factors.

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
CALI_FACTORS factors[4];
int          length = 0;
int          ret = 0;
int          I = 0;

ret = ReadCaliFactors (0, 4, factors, &length);
```

(Continue)

```
if(ret == AXIO_OK){
    for(i=0;i<length;i++)
    {
        printf("Input range: %d\n", factors[i].inputRange);
        printf("Offset: %d\n", factors[i].offset);
        printf("Gain: %lf\n", factors[i].gain);
        printf("\n");
    }
    printf("\n");
} else
    printf("Read calibration factor failure\n");
```

2.14 GetAISingleValueEx

- **Description**

  Get the AI single voltage with the selected channels and input range.

- **Definition**

```
int GetAISingleValue (
            int         size,
            double      *data,
            int         *length
    );
```

- **Parameters**

size	[in]: The buffer size.
*data	[out]: The voltage data.
*length	[out]: The length of voltage data.

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
double      data[16]
int         length = 0;
int         ret = 0, I = 0;
```

```
ret = SetAIChannel (15);
if(ret != AXIO_OK)
    printf("Set AI channel failure\n");

ret = SetAInputRange (0);
if(ret != AXIO_OK)
    printf("Set AI input range failure\n");

ret = GetAISingleValueEx (16, data, &length);
```

(Continue)

```
if (ret == AXIO_OK)
{

    for(i=0;i<length;i++)
    {
            printf("%lf", data[i]);
    }
    printf("\n");
}
else
{
    printf("Get AI single voltage failure\n");
}
```

2.15 GetAIinitConfEx

- **Description**

  Get AI initial setting from flash memory.

- **Definition**

```
int GetAIinitConfEx (
        int             configSize,
        AI_CONIFG       *config,
        Int             *length
    );
```

- **Parameters**

configSize	[in]: The buffer size of AI configuration.
*config	[out]: The AI configuration buffer.
*length	[out]: The length of AI configuration.

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int i = 0;
int ret = 0;
AI_CONFIG config[4];
int length = 0;

ret = GetAIinitConfEx(4, config, &length);
if(ret == AXIO_OK)
{
    for(i = 0; i < length; i++)
   {
      printf("AI%d\n", i);
      printf("Enabled: %d\n", config[i].enabled);
      printf("Input range: %d\n", config[i].inputRange);
(Continue)
```

```
            printf("Sample rate: %d\n", config[i].sampleRate);
            printf("=============================================\n");
        }
    }
```

2.16 SetAIinitConf

- **Description**

  Store AI initial configuration to flash memory.

- **Definition**

  int SetAIinitConf (void)

- **Parameters**

  NULL

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int             ret;

ret = SetAIinitConf ();

if(ret != AXIO_OK){
    printf("Save AI Initial configure failure\n");
```

2.17 AICalibrationEx

- **Description**

  Start the AI calibration mode.

  (IRU151 only supports -5~+5V and -10~+10V)

- **Definition**

```
int AICalibrationEx(
        int     mode,
        int     size,
        char    *data,
        int     *length
    );
```

- **Parameters**

  mode [in]: The calibration mode.

  0x01 : Enter the user calibrate mode.

  0x02 : Zero input 5V – Span input 8V (For input range 0V ~ +10V)

  0x03 : Zero input 0V – Span input 8V (For input range -10V ~ +10V)

  0x04 : Zero input 2.5V – Span input 4V (For input range 0V ~ +5V)

  0x05 : Zero input 0V – Span input 4V (For input range -5V ~ +5V)

  0x06 : Start zero calibrate & return the information.

  0x07 : Start span calibrate & return the information.

  0x08 : Save calibration value & exit the user calibration mode.

  size [in]: The buffer size.

  *data [out]: The return calibration data.

  *length [out]: The length of calibration data.

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

  (Continue)

```
char        * pCalibratedInfo;
short       length;
int         ret;
int         i;

ret = AICalibration(1, & pCalibratedInfo, &length);

if(ret == AXIO_OK)
{
        for(i=0;i<length;i++)
            printf("0x%02x ", *( pCalibratedInfo +i));
        printf("\n");
}
else
{
    printf("Calibration Fail \n");
}
```

2.18 RestoreAIConf

- **Description**

  Reset AI intial configuration to default in flash memory.

- **Definition**

  int RestoreAIConf (void)

- **Parameters**

  NULL

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int ret;

ret = RestoreAIConf ();

if(ret != AXIO_OK){
    printf("Restore AI factory configure failure\n");
```

2.19 GetAIDataLength

- **Description**

  Get remaining AI data length in SDK.

- **Definition**

  int GetAIDataLength (int *length);

- **Parameters**

  *length [out]: The length of remaining data.

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int     ret = 0;
int     length = 0;

ret = GetAIDataLength(&length);

if(ret != AXIO_OK){
    printf("Get AI data length failure.\n");
else
    printf("%d\n", length);
```

2.20 GetDITrigConfEx

- **Description**

  Get DI trigger configuration.

- **Definition**

```
int GetDITrigConf (
            DI_TRIGGER_CONFIG       *config
    );
```

- **Parameters**

  *config [out]: The DI trigger configuration.

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int                     i=0;
DI_TRIGGER_CONFIG       config;
int                     length = 0;
int                     ret = 0;

ret = GetDITrigConfEx(&config);
if(ret == AXIO_OK)
{
    for(i = 0; i < 2; i++)
    {
        printf("DI%d: Filter=%d, Trigger Condition: %s\n", i, config.filter[i],
        config.condition[i] == 0 ? "Rising" : config.condition[i] == 1 ? "Falling" : "Both");
    }
}
```

2.21 ClearDITrigConf

- **Description**

  Restore the default settings in the flash memory.

- **Definition**

  int ClearDITrigConf (void)

- **Parameters**

  NULL.

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int             ret;

ret = ClearDITrigConf();

if(ret != AXIO_OK)
    printf("Clear DI Configure failure\n");
```

2.22 SetDITrigConf

- **Description**

  Save the DI trigger setting in the flash memory.

- **Definition**

  int SetDITrigConf (void)

- **Parameters**

  NULL

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int             ret = 0;

ret = SetDITrigConf();

if(ret != AXIO_OK)
   printf("Save DI Configuration failure\n");
```

2.23 GetDILevelEx

- **Description**

  Get current DI level.

- **Definition**

  int GetDILevel (int *level)

- **Parameters**

  *level [out]: The current DI level.

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int ret = 0;
int level = 0;

ret = GetDILevelEx(&level);
if(ret == AXIO_OK)
{
    printf("%d\n", level);
}
else
{
    printf("Get DI level failure.");
}
```

2.24 EnDIReceiver

- **Description**

  Enable the DI receiver.

- **Definition**

```
int EnDIReceiver(
        short       channel,
        short       filter,
        short       condition
    );
```

- **Parameters**

channel	[in]: Channel. (Bit0-7 indicates channel 0-7)
filter	[in]: DI filter time. (0-250 ms)
condition	[in]: DI trigger condition 0x00 : Raising edge. 0x01 : Falling edge. 0x02 : Both.

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
short       channel = 0x01;
short       filter = 0x00;
short       condition = 0x02;
int         ret = 0x00;

ret = EnDIReceiver(channel, filter, condition);

if(ret != AXIO_OK)
    printf("Enable DI Receiver failure\n");
```

2.25 DisDIReceiver

- **Description**

  Disable the DI receiver.

- **Definition**

```
int DisDIReceiver(
        short       channel
    );
```

- **Parameters**

channel	[in]: Channel. (Bit0-7 indicates channel 0-7)

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
short   channel;
int     ret;

channel = 0x1;

ret = DisDIReceiver (channel);

if(ret != AXIO_OK)
    printf("Disable DI Receiver failure\n");
```

2.26 EnDICounterEx

- **Description**

  Enable the DI channe.

  (Only channel 0 support this mode on IRU device.)

- **Definition**

```
int EnDICounterEx(
            int         channel
            int         condition,
            int         counter
    );
```

- **Parameters**

  channel [in]: Channel (Bit0-7 indicates channel 0-7)

  condition [in]: Set the trigger condition

  0x00 : Raising edge.

  0x01 : Falling edge.

  0x02 : Both.

  counter [in]: The number of counts. (1 ~ 65535)

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
short       condition = 2;
int         count = 100;
int         ret = 0;

ret = EnDICounterEx(0x01, condition, count);

if(ret != AXIO_OK)
    printf("Enable DI Counter failure\n");
```

2.27 DisDICounterEx

- **Description**

  Disable the DI counter mode.

- **Definition**

```
int DisDICounterEx(
    int     channel,
    char    *data,
    int     *length
);
```

- **Parameters**

  channel [in]: Channel (Bit0-7 indicates channel 0-7)

  data [out]: The data of DI counter.
  Byte0: Channel index.
  Byte1: Number of counts. (HI)
  Byte2: Number of counts. (LO)
  …

  length [out]: The length of data.

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
char    buffer[64];
int     length = 0;
int     ret = 0;

ret = DisDICounterEx(0x01, buffer, &length);

if(ret != AXIO_OK)
    printf("Disable DI Counter failure\n");
```

2.28 GetDICounterEx

- **Description**

  Get the DI counter number.

- **Definition**

  int GetDICounterEx (int channel, int *data)

- **Parameters**

  channel [in]: The channel index. (Bit0-7 indicates channel 0-7)

  data [out]: The current data of DI counter.

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int     ret = 0;
int     counts = 0;

ret = GetDICounterEx(0x01, &counts);
if(ret == AXIO_OK)
{
    printf("Current counts = %d\n", counts);
}
else
{
    printf("Get DI counter failure.");
}
```

2.29 SetDICounterCompletedCallback

- **Description**

  Set the DI counter finish callback function.

- **Definition**

```
int SetDICounterCompletedCallback(
        DICounterCompletedCallback      callback
    );
```

- **Parameters**

  callback The callback function for receiving counter finished value.

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
void DICounterCompleted_Callback(int counter)
{
        printf("DI counter was completed. counter=%d\n", counter);
}

// Set DI counter completed callback
SetDICounterCompletedCallback(DICounterCompleted_Callback);
```

2.30 SetDIStatusChangedCallback

- **Description**

  Set the DI status change callback function for the notification while receiving the DI Data.

- **Definition**

```
int SetDIStatusChangedCallback(
    DIStatusChanged_Callback callback
    );
```

- **Parameters**

  callback [in]: The callback function for receiving DI Data notification..

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
void DIStatusChanged_Callback()
{
        printf("DI status was changed.\n");
}

// Set DI status changed callback
SetDIStatusChangedCallback(DIStatusChanged_Callback);
```

2.31 GetDOInitConfEx

- **Description**

  Get the DO setting from the flash memory.

- **Definition**

```
int GetDOInitConfEx (
        char        *config,
        int         *length
    );
```

- **Parameters**

*config [out]: The data of the gotten information.

[Byte 0 : Channel 0]
0x00 : Low
0x01 : High
0xFF : User not been set
[Byte 1 : Channel 1]
0x00 : Low
0x01 : High
0xFF : User not been set

*length [out]: The data length of DO initial configuration.

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int     i=0;
char    config[16];
int     length = 0;

ret = GetDOInitConfEx (config, &length);
```

(Continue)

```
if(ret == AXIO_OK){
        for(i = 0; i < length; i++)
        {
            printf("DO%d: %s\n", i, (config[i] == 0 ? "Low" : config[i] == 1 ? "High" :
"Default"));
        }
        printf("\n");
} else
        printf("Get DO setting failure\n");
```

2.32 ClearDOInitConf

- **Description**

  Clear the DO setting.

- **Definition**

  int ClearDOInitConf (void)

- **Parameters**

  NULL

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int             ret;

ret = ClearDOInitConf ();

if(ret != AXIO_OK)
      printf("Clear DO setting failure\n");
```

2.33 SetDOInitConf

- **Description**

  Save DO settings.

- **Definition**

  int SetDOInitConf (void)

- **Parameters**

  NULL

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int ret;

ret = SetDOInitConf ();

if(ret != AXIO_OK)
   printf("Save DO setting failure\n");
```

2.34 SetDOStatus

- **Description**

  Set the DO status on the selected channel..

- **Definition**

```
int SetDOStatus(
        int         channel,
        int         level
    );
```

- **Parameters**

channel	[in]: Selected channel.
	Bit 0 to 1 indicates the channels 0 ~ 1.
level	[in]: The output status.
	Bit 0 to 3 indicates the status of channel 0 ~ 1.
	1 : High
	0 : Low.

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
short       channel = 0x03 /* selected channel : 0 and 1 */
int         level = 0x03    /* channel 0 and 1 are high */
int         ret;

ret = SetDOStatus(channel, level);

if(ret != AXIO_OK)
    printf("Set DO status failure\n");
```

2.35 GetDOStatus

- **Description**

  Get the DO status on the selected channels

- **Definition**

```
int GetDOStatus (
        short       channel,
        int         *level
    );
```

- **Parameters**

channel	[in]: The value of 1 to 3 indicates the channels 0 – 1.
*level	[out]: The current DO output status. (0x03: channel0&1 HI)

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int     status = 0;
short   channel = 3;        /* channel 0 & 1 */
int     ret;

ret = GetDOStatusEx(channel, &status);

if(ret == AXIO_OK)
{
    printf("DO status=%02X\n", status);
}
 else
{
   printf("Get DO status failure\n");
}
```

2.36 EnDOPWM

- **Description**

  Enable PWM mode on the channel 0 and set all parameters.

- **Definition**

```
int EnDOPWM (
            int         channel ,
            int         dutycycle,
            short       freq
    );
```

- **Parameters**

channel	Selected channel (Must be 0)
dutycycle	The range of the duty cycle is from 1(%) to 99(%).
freq	The range of the value is from 1 to 500(Hz).

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
short       freq = 100;
int         duty = 50;
short       channel = 1;
int         ret;

ret = EnDOPWM(channel, duty, freq);

if(ret != AXIO_OK){
    printf("Enable DO PWM failure\n");
```

2.37 DisDOPWM

- **Description**

  Disable PWM mode on the channel 0.

- **Definition**

```
int DisDOPWM (
        int         channel
    );
```

- **Parameters**

  channel Selected channel (Must be 0)

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
int ret;

ret = DisDOPWM();

if(ret != AXIO_OK){
    printf("Disable DO PWM failure\n");
```

2.38 GetDOPWMConf

- **Description**

  During DO PWM starting, User can call this function to get the DO PWM mode configuration.

- **Definition**

```
int GetDOPWMConf (
        char        *config,
        int         *length
    );
```

- **Parameters**

*config	The data of the gotten information. Byte 0 : Channel index. Byte 1 : Duty cycle Byte 2 : Frequency(H) Byte 3 : Frequency(L)
*length	The data length of the gotten information.

- **Return value**

  AXIO_OK if success, or other value represents the error. (See Error Code)

- **Example**

```
short   channel = 1;
int     ret;
char    config[16];
int     length = 0;

ret = GetDOPWMConf (config, &length);
```

(continue)

```
if(ret != AXIO_OK)
{
    printf("Get DO PWM Configure failure\n");
}
else
{
      for(int i=0;i<length ;i++)
      {
          printf("0x%02x", config[i]);
      }
      printf("\n");
}
```

APPENDIX A
Error Code

Error Code List

Error Code	Error Name	Description
0x00000000	AXIO_OK	Success
0xE0000001	AXIO_ERR_HANDLE	The invalid handle.
0xE0000502	AXIO_ERR_CMD	The command operation failure.
0xE0000003	AXIO_ERR_PARAMETERS	The input parameters are incorrect..
0xE0000004	AXIO_ERR_NOT_SUPPORTED	The feature is not support.
0xE0000005	AXIO_ERR_RESPN_TIMEOUT	The command response is timeout.
0xE0000006	AXIO_ERR_RESPN_MCU	The error response from MCU.