Multifunction-I/O-X3 series ADIOX3-API REFERENCE

Update 2019-3-23

SAYA Inc.

Table of contents

1. Overview

ADiox3-API is a simple API composed of 27 functions and 4 structures designed to quickly develop applications for Multifunction I / O-X3 series and Infrasound sensors. Corresponds to ADXIII42LE-***, ADXIII42FE-***, ADXIII-INF01LE, and ADXIII-INF04LE. All support Ethernet interface and do not support UART version of RS232C, LVTTL.

1. Driver function to access hardware
2. Save and retrieve function and signal adjustment settings
3. Survival function for multiple devices (find live hardware and continue measurement)

Access to nonexistent devices
In this case, the function is not executed and returns FALSE (= 0).

Development file

Header files and import libraries are provided for VC ++.

For the 32bit version	**CDROM¥MFIO_X3¥sdk¥VisuaCPP_X86**
For the 64bit version	**CDROM¥MFIO_X3¥sdk¥VisuaCPP_X64**

ADiox3.h／ADiox3.LIB [ADiox3.dll + ADioxScp.dll]
Main library header file, import library, dynamic link library.

Other development environment
As it is similar to Windows API, it also supports other development environments. However, helper files are not provided.

About CARD_ID
Up to 4 devices can be handled and identified by the CARD_ID (0 to 3) assigned to the IP address.

2. Open close(7)

bADioxOpen

Open the driver. It is not necessary to call this function when setting up open at once using bADioxSystemLoad.

BOOL bADioxOpen (BYTE bCardID,ADR_IP_CONF sADR_IP_CONF);

Argument	bCardID	CARD_ID to be opened.
	sADR_IP_CONF	ADR_IP_CONF structure is specified. In this structure, assign the IP address and port number to CARD_ID.
Return value	1 (TRUE) if successful, 0 (FALSE) if unsuccessful.	

bADioxClose

It performs driver closing, hardware termination processing, buffer release to system memory, etc.

BOOL bADioxClose(BYTE bCardID);

Argument	bCardID	Card ID of the target device.
Return value	1 (TRUE) if successful, 0 (FALSE) if unsuccessful.	

bADioxSystemLoad

Open drivers, initialize hardware (reflect settings), and secure memory buffers for multiple devices. This function opens the configuration file and executes bADioxOpen (open) → bADioxConfigration (setting) for the number of units. The contents set here are reflected in the "setting structure database" inside the DLL.

BOOL bADioxSystemLoad (char *szFile , ADR_IP_CONF sADR_IP_CONF);

Argument	*szFile	Specify the location of the configuration file.
	sADR_IP_CONF	ADR_IP_CONF structure will be will specified In this structure, assign the IP address and port number to CARD_ID.

Configuration file structure

Size	Contents	Value / Notes
DWORD (4Byte)	Header	0x954EF06B
DWORD (4Byte)	Version	0x00000001
ADIOX_SYSTEM structure	CARD_ID Setting of 0	CardID = 0 is required.
ADIOX_SYSTEM structure	CARD_ID Setting of 1	Insert a dummy even if you do not need it
ADIOX_SYSTEM structure	CARD_ID Setting of 2	Insert a dummy even if you do not need it
ADIOX_SYSTEM structure	CARD_ID Setting of 3	Insert a dummy even if you do not need it
DWORD (4Byte)	Footer	0x954EF06B

If it is troublesome
If the footer or coding for creating a file is troublesome, please configure it with MultiLogger X3 of the attached software.
ConfMlt.scp created in the same folder after MultiLogger X3 is the file of the above format.

Return value 1 (TRUE) if successful, 0 (FALSE) if unsuccessful.

bADioxSystemStore

Closes the driver, terminates the hardware, releases the memory buffer for multiple devices, and saves the device settings (ADIOX_SYSTEM) in the setting file. In order to save the device settings with this function, it is necessary to reflect them in the "setting structure database" inside the DLL in advance using vDbWrite.

BOOL bADioxSystemStore (char *szFile , ADR_IP_CONF sADR_IP_CONF);

Argument	*szFile	Specify the location of the configuration file.
	sADR_IP_CONF	Specify an ADR_IP_CONF structure in which the IP address and port number are assigned to CARD_ID.

Configuration file structure it is same as above bADioxSystemLoad.

Hint If you do not need to change the settings,
call the same file as for bADioxSystemLoad or exit with bADioxClose.

Return value 1 (TRUE) if successful, 0 (FALSE) if unsuccessful.

bADioxRetryOpen

In operation, devices that return an error should be disconnected with bADioxClose. This function retries (reconnects and initializes) the disconnected device. By periodically retrying the disconnected device, automatic recovery can be achieved. If retry fails, it will wait for response and lock, so retry frequency should be appropriate.

BOOL bADioxRetryOpen (ADR_IP_CONF sADR_IP_CONF, BYTE bCardID);

Argument	bCardID	Card ID of the target device.
	sADR_IP_CONF	Specify an ADR_IP_CONF structure in which the IP address and port number are assigned to CARD_ID.IP

Return value 1 (TRUE) if successful, 0 (FALSE) if unsuccessful.

bADioxBootStatus

Check if the device of specified CARD_ID can be started. In the case of data collection with multiple devices, it is used to identify the currently active CARD_ID and the disconnected (retry target) CARD_ID. Even a single device is used to understand the status.

BOOL bADioxBootStatus(BYTE bCardID, LPBYTE lpbErrorType);

Argument	bCardID	CARD_ID you want to check for activation
	lpbErrorType	A pointer containing the status of whether or not it could be activated. Value has no error cause (0) There was some error in the configuration file (1), connection failed, or even if it could connect, an error (2) occurred.

Return value 1 (TRUE) if successful, 0 (FALSE) if unsuccessful.

vADioxErrorMessageStop

Make sure that no error message is displayed. An error dialog pops up to prevent blocking data collection. This is because it is not desirable to give an error message if you introduce a mechanism that disconnects due to an error and recovers automatically based on a retry.

void vADioxErrorMessageStop(BOOL bStop);

Argument	bStop	Suppress error messages with TRUE (= 1). Enable error message with FALSE (= 0).

3. Setting (5)

bADioxConfigration

The device settings (ADIOX_SYSTEM) specified by CARD_ID are reflected in the hardware. It does not affect the "setting structure database" inside the DLL.

BOOL bADioxConfigration (struct ADIOX_SYSTEM *lpsTBufSetup,BYTE bCardID);

Argument	*lpsTBufSetup	Specify the ADIOX_SYSTEM structure that contains the device settings.
	bCardID	Card ID of the target device.

Return value 1 (TRUE) if successful, 0 (FALSE) if unsuccessful.

vDbWrite

The device settings (ADIOX_SYSTEM) specified by CARD_ID are reflected in the "setting structure database" inside the DLL. However, it does not access the hardware. Call this function in advance to save the settings on bADioxSystemStore.

void vDbWrite (struct ADIOX_SYSTEM sTBufSetup,BYTE bCardID);

Argument	sTBufSetup	Specify the ADIOX_SYSTEM structure that contains the device settings.
	bCardID	Card ID of the target device.

vDbRead

From the "configuration structure database" inside the DLL,
Reads the device setting (ADIOX_SYSTEM) specified by CARD_ID.

void vDbRead (struct ADIOX_SYSTEM *lpsTBufSetup,BYTE bCardID);

Argument	sTBufSetup	Specify the ADIOX_SYSTEM structure that contains the device settings.
	bCardID	Card ID of the target device.

bADioxScpSetup

From the device settings (ADIOX_SYSTEM), only the settings related to signal adjustment (sensor mode, calibration position, calibration factor, scaling, alarm) are reflected in the "setting structure database" in the hardware and DLL.

BOOL bADioxScpSetup (BOOL bExtentionAPI , struct ADIOX_SYSTEM * lpsTBufSetup , BYTE bCardID);

Argument	bExtentionAPI	Always specify FALSE.
	*lpsTBufSetup	Specify the ADIOX_SYSTEM structure that contains the device settings (including signal adjustment).
	bCardID	Card ID of the target device.
Return value	1 (TRUE) if successful, 0 (FALSE) if unsuccessful.	

bADioxScpDefault

For the specified CARD_ID, channel number, and signal type, generates default values for signal adjustment and reflects them in the "setting structure database" (device setting (ADIOX_SYSTEM)) inside the DLL.

BOOL bADioxScpDefault(DWORD dwSensorMode, BYTE bCardID, BYTE Bch);

Argument	dwSensorMode	Sensor type
	bCardID	Card ID of the target device.
	dwCH	Input channel number (0-7 for analog, 8-11 for counter / infrasound)
Return value	1 (TRUE) if successful, 0 (FALSE) if unsuccessful.	

4. Data acquisition, Signal condition(6)

bADioxIrqStatus

In ring buffer data acquisition, the interrupt status is acquired.

BOOL bADioxIrqStatus (struct ADIOX_IRQ * lpsTIrqPack , BYTE bCardID);

Argument	lpsTIrqPack	Specify a pointer to the structure ADIOX_IRQ that stores the interrupt cause.
	bCardID	Card ID of the target device.
Return value	1 (TRUE) if successful, 0 (FALSE) if unsuccessful.	

bADioxBufferRead

Reads 128 samples of block data from the ring buffer. If you receive a ring buffer interrupt message, please access this function to acquire buffer data.

**BOOL bADioxBufferRead (LPDWORD lpdwAdData , LPDWORD lpdwCtcData, LPDWORD lpdwDI,
GPS_DATA_EX *lpsGPS_DATA, int * lpiBatteryLevel, double * lpdTemperature,
BYTE bCardID, BOOL bInit);**

Argument	lpdwAdData	Analog input data channels 0 to 7 (AI 0 to 7) are stored sequentially from the top, and this is repeated 128 times. The size of analog data is 16 bits, and the upper 16 bits are always 0. The 16 bit straight binary data should be scaled by the programmer on the code or by dADioxLinCoef. Please secure the array of 8ch X 128 samples = 1024. The input format of each channel is specified by dwSensorMode in the ADIOX_SYSTEM structure.
	lpdwCtcData	Counter input data channels 0 to 3 (CTC 0 to 3) are stored sequentially from the top, and this is repeated 128 times. The size of one analog data is 32 bits. The programmer should scale the 32 bit straight binary data in code or with dADioxLinCoef. Please secure the array of 4ch X 128 samples = 512. The input format of each channel is specified by dwSensorMode in the ADIOX_SYSTEM structure.
	lpdwDI	Pointer containing digital input. This is only one sample. Only the lower 16 bits are valid, and the upper 16 bits are always 0. In the Infra Sound Sensor, the arguments are stored meaningless numbers.
	*lpsGPS_DATA	Pointer to the GPS_DATA_EX structure containing GPS data
	*lpiBatteryLevel	Pointer that stores battery level 0 to 100% indication. It is meaningless with the infrastructure sound sensor.
	*lpdTemperature	A pointer that stores the device temperature. The signal and the scaled value that is converted into ℃ is stored.

	bCardID	Card ID of the target device.
	bInit	If you specify TRUE, the Infrasound sensor calibrates the first sample value of this data string to zero. Set TRUE only at the beginning of data collection, and FALSE after that. Otherwise, zero calibration will be performed each time.
Return value	1 (TRUE) if successful, 0 (FALSE) if unsuccessful.	

* Infrasound data may be assigned to analog input data channels or counter input data channels.

bADioxLastBankCtrl

Makes the last bank readable in ring buffer data collection.

BOOL bADioxLastBankCtrl (BOOL bBankLock , DWORD dwStopAddr , BYTE bCardID);

Argument	bBankLock	When TRUE, the data collection side bank at stop is fixed to read. (Bank will change otherwise) In the case of FALSE, bank fixed is returned to free. In other words, in the code flow,
		bADioxLastBankCtrl(TRUE ,stopaddr ,)
		bADioxBufferRead()
		bADioxLastBankCtrl(FALSE , dummy,)
		it handles up to dwStopAddr of the ADIOX_IRQ structure as the valid range of data.

	dwStopAddr	Specify the stopped address (0-127). Specify dwStopAddr in the ADIOX_IRQ structure.
	bCardID	Card ID of the target device.
Return value	1 (TRUE) if successful, 0 (FALSE) if unsuccessful.	

bADioxBlockRead

Read various data at once. Access to this function should be judged by software. (In fact, please proceed polling)

**BOOL bADioxBlockRead (LPDWORD lpdwTmp , GPS_DATA_EX *lpsGPS_DATA, double *lpdTemp,
int * lpiBatteryLevel, BYTE bCardID, BOOL bInit);**

Argument	lpdwTmp	Analog input data channels 0 to 7 (AI 0 to 7), counter input data channels 0 to 3 (CTC 0 to 3), and digital input data are stored sequentially from the top. The analog data size is 16 bits and the upper 16 bits are always 0. Straight binary data should be scaled by the programmer in the code or by dADioxLinCoef. Secure 13 arrays with 8 analog inputs + 4 counters / Infrasound 4 channels / digital input. The input format of each channel is specified by dwSensorMode in the ADIOX_SYSTEM structure.
	*lpsGPS_DATA	A pointer to the GPS_DATA_EX structure containing GPS data.
	*lpdTemp	A pointer that stores the device temperature. The signal is converted and scaled to ° C is stored.
	*lpiBatteryLevel	Pointer that stores battery level 0 to 100% indication. It is meaningless with the infrastructure sound sensor.
	bCardID	Card ID of the target device.
	bInit	If you specify TRUE, the Infrasound sensor will calibrate this sample value to zero. Set TRUE only at the beginning of data collection, and FALSE after that. Otherwise, zero-calibrated will be done each time.
Return value	1 (TRUE) if successful, 0 (FALSE) if unsuccessful.	

*In some cases, infrasound data is assigned to the analog input data channel or counter input data channel.

dADioxLinCoef

When DWORD type analog counter / infrasound input value acquired by bADioxBufferRead or bADioxBlockRead is passed to this function, processing such as reflection of zero / span calibration, linearizer, scaling, alarm, burnout detection etc. will be returned.

**double dADioxLinCoef (DWORD dwANALOG, double dTemp, LPBOOL lpbBurnOut,
LPDWORD lpdwAlarm,BYTE bCardID,BYTE bCH);**

Argument	dwANALOG	Set the input value such as analog counter infrasound
	dTemp	Set the temperature(°C). Used for zero junction compensation such as thermocouples.
	lpbBurnOut	It is a pointer of BOOL type with burnout detection flag. Burnout occurs when TRUE (= 1).
	lpdwAlarm	It is a pointer of BOOL type with the alarm detection flag. An alarm occurs when TRUE (= 1).
	bCardID	Card ID of the target device.
	bCH	Input channel number (0-7 for analog, 8-11 for counter / infrasound)
Return value	Converted value	

vADioxErrorReport

Report an error.

**void vADioxErrorReport (LPDWORD lpdwSetError , LPDWORD lpdwWriteRetry,
LPDWORD lpdwReadRetry,LPDWORD lpdwReadFlameError,LPDWORD lpdwReadAddressError);**

Argument	lpdwSetError	A pointer that stores the number of final errors such as no function response.
	lpdwWriteRetry	Pointer that stores the number of retries for data transmission
	lpdwReadRetry	Pointer that stores the number of retries for data reception
	lpdwReadFlameError	A pointer that stores the number of collapses of the frame structure of data transmission / reception packets.
	lpdwReadAddressError	Address verification error of data transmitted and received.
Return value	0 is returned.	

5. Calibration(4)

bADioxZeroAdj

Performs zero calibration (= offset calibration) for the specified CARD_ID and input channel number. The zero calibration position is set by bADioxScpSetup. Before performing this function, provide a zero reference value (for hardware) to the target input. Repeat zero calibration and span calibration for more accurate calibration.

BOOL bADioxZeroAdj (BYTE bCardID, BYTE bChannel);

Argument	bCardID	Card ID of the target device.
	bChannel	Analog input channel number
Return value	1 (TRUE) if successful, 0 (FALSE) if unsuccessful.	

bADioxSpanAdj

Performs span calibration (= gain calibration) for the specified CARD_ID and channel number. The span calibration position is set by bADioxScpSetup. Before executing this function, provide a span reference value (in hardware) to the target analog input. Repeat zero calibration and spon calibration for more accurate calibration.

BOOL bADioxSpanAdj (BYTE bCardID,BYTE bChannel);

Argument	bCardID	Card ID of the target device.
	bChannel	Analog input channel number
Return value	1 (TRUE) if successful, 0 (FALSE) if unsuccessful.	

vADioxFreeAdj

The specified CARD_ID and channel number are not calibrated, that is, the zero calibration amount is 0 and the span calibration amount is 1.

void vADioxFreeAdj (BYTE bCardID,BYTE bChannel);

Argument	bCardID	Card ID of the target device.
	bChannel	Analog input channel number

bADioxAutoZero

Perform zero calibration on the amplifier and A / D converter on the board. Do not use this function frequently. When using polling and ring buffer, do not use as there is a step to the zero point. It is not necessary to call this function unless you require high accurate one.

BOOL bADioxAutoZero(BYTE bCardID);

Argument	bCardID	Card ID of the target device.
Return value	1 (TRUE) if successful, 0 (FALSE) if unsuccessful.	

6. High-speed timer (3)

vStartTimerIRQ

Start timer interrupt. This timer is faster and more accurate than the standard Windows timer.

void vStartTimerIRQ (UINT uiPacerDelay,HWND hmWnd);

Argument	uiPacerDelay	Specify the timer interrupt interval in units of 1 msec.
	hmWnd	Window handle to transmit timer interrupt (message). When a timer interrupt occurs, The message WM_USER + 2020 will be sent to this window handle.

vStopMtimerIRQ

Stops the timer interrupt started by vStartTimerIRQ.

void vStopMtimerIRQ ( );

vReserMtimerIRQ

In timer interrupt started by vStartTimerIRQ, it notifies that interrupt processing is completed. The next timer interrupt will not resume until this function is called.

void vReserMtimerIRQ ( );

7. Structure(4)

ADIOX_SYSTEM

Configuration structure that covers all settings of hardware and signal conditioning.

```
struct ADIOX_SYSTEM
        {
        DWORD    dwClockScall;

        // AI TRIG
        DWORD    dwStartTrigDelay;
        DWORD    dwStartTrigLevel1;
        DWORD    dwStartTrigLevel2;
        DWORD    dwStopTrigDelay;
        DWORD    dwStopTrigLevel1;
        DWORD    dwStopTrigLevel2;

        // DI  TRIG
        DWORD    dwStartMask;
        DWORD    dwStartDiPattern;
        DWORD    dwStartTrigSourceDI_ch;
        DWORD    dwStopMask;
        DWORD    dwStopDiPattern;
        DWORD    dwStopTrigSourceDI_ch;

        // TRIG MODE
        DWORD    dwTrigStopMode;
        DWORD    dwTrigStartMode;

        // TRIG MISC
        DWORD    dwIntrruptMode;
        DWORD    dwDeadTime;

        // STOP COUNTER
        DWORD    dwStopCounterValue;

        // INFRA SOUND
        double   dSetCoef;
        DWORD    dwInfraSoundMode;

        // COUNTER
        DWORD    dwChatCan;
        DWORD    dwLatchMode_A;
        DWORD    dwLatchMode_B;
        DWORD    dwLatchMode_C;
        DWORD    dwLatchMode_D;
        DWORD    dwZ_CENTER_A;
        DWORD    dwZ_CENTER_B;
        DWORD    dwZ_CENTER_C;
        DWORD    dwZ_CENTER_D;
        DWORD    dwSoftwareClear_A;
        DWORD    dwSoftwareClear_B;
        DWORD    dwSoftwareClear_C;
        DWORD    dwSoftwareClear_D;

        // AO,DO
        DWORD    dwAo0;
        DWORD    dwAo1;
        DWORD    dwDOS;

        // AI MODE
        DWORD    dwInputShort;
        DWORD    dwFilterEnable;
        DWORD    dwPeakholdCh;

        // 信号調節
        DWORD    dwSensorMode[MAX_AI_CH];
        double   doZeroPos[MAX_AI_CH];
        double   doSpanPos[MAX_AI_CH];
        double   doZero_Coefficient[MAX_AI_CH];
        double   doSpan_Coefficient[MAX_AI_CH];
        BOOL     bScalling[MAX_AI_CH];
        double   dOutTopScall[MAX_AI_CH];
        double   dOutBottomScall[MAX_AI_CH];
        double   dInTopScall[MAX_AI_CH];
        double   dInBottomScall[MAX_AI_CH];
        DWORD    bAlarmMode[MAX_AI_CH];
        double   dAlarmUpper[MAX_AI_CH];
        double   dAlarmLower[MAX_AI_CH];
        double   dScallingRatio[MAX_AI_CH];
        };
```

Member variable

[Sampling frequency]

dwClockScall	Specify the sampling frequency of A / D converter. Always sample with 8 channels of multiplex. Sampling frequency = (460.8 KHz ÷ dwClockScall), valid range 0x17 to 0x1FFFFFF.

[Trigger mode]

dwTrigStopMode	Specify a stop trigger. Please select from the following seven.
dwTrigStartMode	Specify a start trigger. Please select from the following seven.

RESET	(0x0)	Trigger condition not satisfied
BURST	(0x1)	Unconditional trigger
DI_POSEDGE	(0x2)	Trigger on rising edge of digital input
DI_NEGEDGE	(0x3)	Trigger on falling edge of digital input
DI_PATTERN	(0x4)	Trigger when digital input has specified pattern
AI_LEVEL	(0xB)	Level (edge) trigger of analog input
AI_AREA	(0xC)	Area trigger for analog input

If there is no trigger, set dwTrigStartMode = BURST and dwTrigStopMode = RESET. Now the arguments related to the trigger become meaningless except for other trigger delays.

[Analog trigger]

dwStartTrigLevel1	Specifies start trigger level1 for analog level trigger and analog area trigger.(*1)
dwStartTrigLevel2	Specifies start trigger level2 for analog level trigger and analog area trigger.(*1)
dwStopTrigLevel1	Specifies stop trigger level1 for analog level trigger and analog area trigger.(*1)
dwStopTrigLevel2	Specifies stop trigger level2 for analog level trigger and analog area trigger.(*1)

[Trigger delay]

dwStartTrigDelay	Start trigger delay. Value 1 = 1/8 sampling time. For such time, its start delays a bit than trigger and then collect data. You can specify from 0x0 to 0xFFFF.
dwStopTrigDelay	Stop trigger delay. Value 1 = 1/8 sampling time. For such time, its start delays a bit than trigger and then collect data. You can specify from 0x0 to 0xFFFF.

[Digital edge trigger]

dwStartTrigSourceDI_ch	Channel designation for digital edge start trigger. Specify the number of channels, ie 0-15.
dwStopTrigSourceDI_ch	Channel specification for digital edge / stop trigger. Specify the number of channels, ie 0-15.

[Digital pattern trigger]

dwStartMask	Mask for digital pattern start trigger. The bit field of this variable corresponds to the channel of the digital input. (* 3) The corresponding bit is 1 for mask and 0 for unmask.
dwStopMask	Mask for digital pattern stop trigger. The bit field of this variable corresponds to the channel of the digital input. (* 3) The corresponding bit is 1 for mask and 0 for unmask.
dwStartDiPattern	Specifies the trigger pattern for digital pattern start trigger. The trigger comes into effect when this value matches the digital input value.
dwStopDiPattern	Specifies the trigger pattern for digital pattern stop trigger. The trigger comes into effect when this value matches the digital input value.

[Trigger ・ Ring buffer start stop ・ Automatic stop]

dwIntrruptMode	Ring buffer start / stop. It starts with DMA_INT (1) and stops with NOT_INT (0).
dwStopCounterValue	Stops automatically when collecting data for the number of banks specified by this variable.If 0 is specified, this automatic stop function becomes invalid and data collection will be performed indefinitely until stop trigger or stop command is executed.
dwDeadTime	Specify the time until stop trigger detection becomes valid after start trigger is valid. This is to prevent the stop trigger from being applied suddenly. The value is 1/8 sampling time and valid up to 0-0xFFFFFFFF.

[Infrasound]

dSetCoef	Infrastructure sound calibration factor (specifies the sensitivity), for example, 2017 is 2.0.
dwInfraSoundMode	Specify 1 for the infrasound sensor. Other than that is 0.

[Encoder counter]

dwChatCan	Turn chattering canceller on with dwChatCan. 1 and off with 0.
dwLatchMode_A	Specifies the counter 0 latch mode from the following three.
dwLatchMode_B	Specifies the counter 1 latch mode from the following three.
dwLatchMode_C	Specifies the counter 2 latch mode from the following three.
dwLatchMode_D	Specifies the counter 3 latch mode from the following three.

SOFT	(0x0)	software latch
Z_PHASE	(0x1)	Latch when Z phase condition satisfied
DI_SEL	(0x2)	Latch on Y phase rising edge

dwZ_CENTER_A	Specify counter 0, Z phase condition satisfied mode (counter reset)

	by any of the following 0-1.
dwZ_CENTER_B	Specify counter 1, Z phase condition satisfied mode (counter reset) by any of the following 0-1.
dwZ_CENTER_C	Specify counter 2, Z phase condition satisfied mode (counter reset) by any of the following 0-1.
dwZ_CENTER_D	Specify counter 3, Z phase condition satisfied mode (counter reset) by any of the following 0-1.

1: CCW direction: B phase falling when AZ phase is 1,
CW direction: A phase falling when BZ phase is 1
0: Counter reset on Z phase rising condition

dwSoftwareClear_A	When dwLatchMode_A is set to SOFT, this variable is 1 and the counter is reset, 0 is non-reset.
dwSoftwareClear_B	When dwLatchMode_B is set to SOFT, this variable is 1 and the counter is reset, 0 is non-reset.
dwSoftwareClear_C	When dwLatchMode_C is set to SOFT, this variable is 1 and the counter is reset, 0 is non-reset.
dwSoftwareClear_D	When dwLatchMode_D is set to SOFT, this variable is 1 and the counter is reset, 0 is non-reset.

[output]

dwAo0	Analog output channel 0. Specify 0x0 to 0xFFFF.
dwAo1	Analog output channel 1. Specify 0x0 to 0xFFFF.
dwDOS	Digital output channel output value. The bit field of this variable corresponds to the channel of the digital input. (* 3) Since DO of up to 16 channels is implemented, specify 0 to 0xFFFF.

[Analog hardware function]

dwInputShort	0 is a normal analog input, 1 is short out the input to ground.
dwFilterEnable	Switch digital filter to analog input signal. 0 for Off, 1 for On.
dwPeakholdCh	Specifies the peak hold. The bit field of this variable corresponds to the channel of the analog input. 1 is on, 0 is off. Be sure to turn it off for ring buffer data collection. In polling, it gets data, records the highest value until the next acquisition, and makes it data.

[Signal conditioning]

dwSensorMode[*]	Specify the signal type for each input channel. Amplifier gain, scaling, and linearizer are set automatically. For example, if it is CA_K, data acquired by bADioxBufferRead or bADioxBlockRead is automatically converted to ° C simply by passing it through dADioxLinCoef. The definition that can be set is shown below. Note that channel assignments are fixed for items with a red mark at the end.

[Signal sources that can be set to [Channel 0 to 7]

Definition	**Value (decimal base)**	**Content**
NOT_USE	0	Signal condition not used
CA_K	1	Thermocouple K 1400 to -240 °C
CA_Kb	2	Thermocouple K 250 to 0 ° C low noise
CA_J	3	Thermocouple J 1200 to -200 °C
CA_Jb	4	Thermocouple J 190 to 0 ° C low noise
CA_E	5	Thermocouple E 1000 to -240 °C
CA_Eb	6	Thermocouple E 155 to 0 ° C low noise
CA_T	7	Thermocouple T 400 to -200 °C
CA_Tb	8	Thermocouple T 215 to 0 ° C low noise
CA_R	9	Thermocouple R 1760 to -50 °C
CA_Rb	10	Thermocouple R 955 to 0 ° C low noise
CA_S	11	Thermocouple S 1300 to -200 °C
CA_Sb	12	Thermocouple S 1040 to 0 ° C low noise
CA_N	13	Thermocouple N 1400 to -240 °C
CA_Nb	14	Thermocouple N 320 to 0 ° C low noise
CA_B	15	Thermocouple B 1800 to 200 °C
CA_Bb	16	Thermocouple B 1495 to 0 ° C low noise
PT100	17	Platinum RTD Pt100 800 to 0 °C
JPT100	18	Platinum RTD JPt100 500 to 0 ° C
VBP_10mV	19	Voltage ± 10mV range
VBP_100mV	20	Voltage ± 100mV range
VBP_1V	21	Voltage ± 1V range
VBP_10V	22	Voltage ± 10V range
I_4_20	23	Current 4-20 mA / 500 Ω termination
I_4_20EX	24	Current 4-20 mA / 350 Ω termination
I_4_20EX2	25	current 4-20mA / 47Ω onboard termination
D16BIT	26	0-65535

Below is the Infrasound Sensor **ADXIII-INF01LE** function.

VIB	46	accelerometer XYZ (earthquake) **Always AI0-2 (Gal)**
SPL	47	Sound level meter (Z characteristics) **Always AI3 (dB/SPL)**

APL	48	barometer **Always AI4(KPa)**

Below is the Infrasound Sensor **ADXIII-INF04LE** function.

VIB	46	accelerometer XYZ (earthquake) **Always AI0-2 (Gal)**
INF04_HF	49	Infrasound HF(0.001Hz-1000Hz) **Always AI3 (mPa)**

Signal sources that can be set to [Channels 8 to 11]

Definition	Value	Content
EC_4X	27	4x encoder counter (Z not used)
EC_4XZ	28	4x encoder counter (Z used)
EC_2X	29	2x encoder counter (Z not used)
EC_2XZ	30	2x encoder counter (Z used)
EC_1X	31	1x encoder counter (Z not used)
EC_1XZ	32	1x encoder counter (Z used)
UPC	33	up / down counter (Z not used)
UPC_Z	34	up / down counter (Z used)

Below is the Infrasound Sensor **ADXIII-INF01LE** function.

INFRS_TA	37	Temperature sensor **Always CTC2 (°C)**
INFRS_FB	36	Infrasound DC **Always CTC0 (mPa)**
INFRS_DIF	41	Infrasound AC **Always CTC1 (mPa)**

Below is the Infrasound Sensor **ADXIII-INF04LE** function.

INF04_LF	50	Infrasound LF (pressure) **Always CTC0 (hPa)**
INF04-TMP	51	Temperature sensor **Always CTC1 (°C)**

[Signal conditioning / Calibration]

doZeroPos[*]	Stores "MAX_AI_CH" (* 2) as zero calibration positions.
doSpanPos[*]	Stores "MAX_AI_CH" (* 2) as span calibration positions.
doZero_Coefficient[*]	Stores "MAX_AI_CH" (* 2) as zero calibration coefficients.
doSpan_Coefficient[*]	Stores "MAX_AI_CH" (* 2) as span calibration coefficients.

[Signal conditioning / Scaling]

bScalling[*]	TRUE if scaling, FALSE if not. Stores "MAX_AI_CH" (* 2).
dOutTopScall[*]	Scaling reference value after conversion (upper). Store this "MAX_AI_CH"(*2).
dOutBottomScall[*]	Scaling reference value (bottom) after conversion. Store this "MAX_AI_CH"(*2).
dInTopScall[*]	Scaling reference value before conversion (upper). Store this "MAX_AI_CH"(*2).
dInBottomScall[*]	Scaling reference value before conversion (bottom).Store this "MAX_AI_CH"(*2).
dScallingRatio[*]	This variable is used internally. Do not operate.

For example, when 4 to 20 mA is set to 1000 to 10000 rpm,
dOutTopScall=10000、dOutBottomScall＝1000、dInTopScall＝20、dInBottomScall＝4、bScalling=TRUE

[Signal conditioning /alarm]

bAlarmMode[*]	Specifies the alarm mode. When 0 is specified: Off, when 1 is specified: Alarm (over) at dAlarmUpper or higher If 2 is specified, alarm (under) will occur at dAlarmLower or lower. When 3 is specified, an alarm (in range) within the range of dAlarmUpper to dAlarmLower, When 4 is specified, the alarm (out range) will be in the range of dAlarmUpper to dAlarmLower. Store this "MAX_AI_CH" (* 2).
dlarmUpper[*]	Alarm setting value (upper) Is stored "MAX_AI_CH" (* 2).
dAlarmLower[*]	Alarm setting value (bottom) Is stored "MAX_AI_CH" (* 2).

*1 Analog level minimum to maximum correspond to 0 to 0xFFFF.
*2 MAX_AI_CH is 12, 0-7 is analog input, 8-11 is counter and infrasound.
*3 For example, Bit 5 (0x20) corresponds to digital input / output channel 5.

ADIOX_IRQ

Stores interrupt status.

```
struct ADIOX_IRQ
        {
        DWORD   dwBankCount;
        DWORD   dwTrigSeq;
        DWORD   dwBankAddr;
        DWORD   dwIrq;
        DWORD   dwStopAddr;
        };
```

Member variable

dwBankCount	dwBankCount. Stores the number of ring buffer bank changes (number of interrupts). If this number and the number of interrupts are different, sampling is too fast due to buffer overrun.
dwTrigSeg	Trigger ring buffer engine status. TRIG_IDLE 0x0 Stopped or waiting for start trigger. TRIG_RUN 0x1 Collecting data. TRIG_TURN 0x2 Wait for stop trigger. TRIG_HIST0x3 During dead time.
dwBankAddr	Indicates a bank of ring buffers.
dwIrq	Interrupt occurrence status. Occurs at 1. → Obtain ring buffer data immediately.
dwStopAddr	Ring buffer stop address (where it stopped)

ADR_IP_CONF

Assign an IP address and port number to CARD_ID.

```
struct ADR_IP_CONF
        {
        int     iIP1[MAX_MFIO];
        int     iIP2[MAX_MFIO];
        int     iIP3[MAX_MFIO];
        int     iIP4[MAX_MFIO];
        int     iPORT[MAX_MFIO];
        BOOL    bEnable[MAX_MFIO];
        };
```

Member variable

iIP1[*]	IP address for [Array number = CARD_ID], first digit (192 for 192.168.1.50)
iIP2[*]	IP address for [Array number = CARD_ID], 2'nd digit (168 for 192.168.1.50)
iIP3[*]	IP address for [Array number = CARD_ID], 3'rd digit (1 for 192.168.1.50)
iIP4[*]	IP address for [Array number = CARD_ID], 4'th digit (50 for 192.168.1.50)
iPORT[*]	Port number for [Array number = CARD_ID]
bEnable[*]	Whether or not to enable the device with [Sequence number = CARD_ID], TRUE as valid, FALSE as invalid.

GPS_DATA_EX

Stores GPS data in the infrastructure sound sensor.

```
struct GPS_DATA_EX
        {
        DWORD   dwYear;
        DWORD   dwMonth;
        DWORD   dwDay;
        DWORD   dwHour;
        DWORD   dwMinute;
        double  dSecond;
        };
```

Member variable

dwYear	Year
dwMonth	Month
dwDay	Day
dwHour	Hour
dwMinute	Minute
dSecond	Second (3 decimal places)