12AISS44AO4

12-Bit: 8 Ch Analog Input, 4 Ch Analog Output,
16-Bit Digital Input/Output

Windows XP/Win7 Driver User Manual

Manual Revision: June 17, 2015

General Standards Corporation
8302A Whitesburg Drive
Huntsville, AL 35802
Phone: (256) 880-8787
Fax: (256) 880-8788
URL: http://www.generalstandards.com
E-mail: sales@generalstandards.com
E-mail: support@generalstandards.com

Preface

Additional copies of this manual or other literature may be obtained from:

General Standards Corporation
8302A Whitesburg Dr.
Huntsville, Alabama 35802
Phone: (256) 880-8787
FAX: (256) 880-8788
URL: http://www.generalstandards.com
E-mail: sales@generalstandards.com

General Standards Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Although extensive editing and reviews are performed before release to ECO control, **General Standards Corporation** assumes no responsibility for any errors that may exist in this document. No commitment is made to update or keep current the information contained in this document.

General Standards Corporation does not assume any liability arising out of the application or use of any product or circuit described herein, nor is any license conveyed under any patent rights or any rights of others.

General Standards Corporation assumes no responsibility for any consequences resulting from omissions or errors in this manual or from the use of information contained herein.

General Standards Corporation reserves the right to make any changes, without notice, to this product to improve reliability, performance, function, or design.

The Purchaser of this software may use or modify in source form the subject software, but not to re-market or distribute it to outside agencies or separate internal company divisions. The software, however, may be embedded in the Purchaser's distributed software. In the event the Purchaser's customers require the software source code, then they would have to purchase their own copy of the software.

General Standards Corporation makes no warranty of any kind with regard to this software, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose and makes this software available solely on an "as-is" basis. **General Standards Corporation** reserves the right to make changes in this software without reservation and without notification to its users.

GSC is a trademark of **General Standards Corporation**.

PLX and PLX Technology are trademarks of PLX Technology, Inc.

Table of Contents

1. Scope

The Purpose of this document is to describe how to interface with the 12AISS44AO4 Windows Driver API developed by General Standards Corporation (GSC). This software provides the interface between the “Application Software” and the 12AISS44AO4 board.

The 12AISS44AO4 Driver API Software executes under control of the Windows Operating System. The 12AISS44AO4 is implemented as a standard Windows driver API written in “C” programming language. The 12AISS44AO4 Driver API Software is designed to operate on CPU boards containing x86 & x64 processors.

The 12AISS44AO4 Driver consists of a Windows driver with an interface layer (GSC API) to simplify the interface to the PLX Driver. While an application may interface directly to the PLX driver, interfacing to the GSC API layer, will simplify the application software development.

2. Hardware Overview

The PMC-12AISS44AO4 board is a single-width PCI mezzanine card (PMC) that provides 12-bit analog input/output capability for PMC applications. 8 analog input lines can be configured either as 8 single-ended input channels or as 8 differential channels, and are digitized at rates up to 2,000,000 conversions per second per channel. The voltage range for analog inputs are configured via hardware straps as ±100mV, ±1V or ±10V.
Four 12bit single-ended analog outputs provide software controlled ranges of ±2.5V, ±5V or ±10V. A 16-bit bidirectional digital I/O port also is provided. The board is functionally compatible with the IEEE PCI local bus specification Revision 2.3. A PCI interface adapter supports the "plug-n-play" initialization concept.
A PCI interface adapter provides the interface between the controlling PCI bus and the internal local controller through a 16-bit local bus. The local controller manages all input/output configuration and data manipulation functions, including auto calibration. Analog output levels are initialized to zero (midrange). Multiboard synchronization is supported.

Selftest networks permit all channels to be calibrated automatically to a single internal voltage reference. Offset and gain trimming of the output channels are performed by calibration DAC's that are loaded with channel correction values during initialization. The correction values are determined during auto calibration, and are stored in nonvolatile EEprom for subsequent transfer to the calibration DAC's. Either auto calibration or initialization can be invoked at any time by asserting a single control bit in the board control register.

The board is designed for minimum off-line maintenance, and includes internal monitoring features that eliminate the need for disconnecting or removing the module from the system for calibration. All analog input and output system connections are made through a single 80-pin, dual-ribbon front-access I/O connector. Power requirements consist of +5 VDC, in compliance with the PCI specification, and operation over the specified temperature range is achieved with conventional convection cooling.

3. Referenced Documents

The following documents provide reference material for the 12AISS44AO4 board:

- PMC-12AISS44AO4 User's Manual – GSC
- PLX Technology, Inc. PCI 9080 PCI Bus Master Interface Chip data sheet.

4. General Standards API

This section describes the interface to the 12AISS44AO4 GSC API. The 12AISS44AO4 GSC API isolates the user from operating system specific requirements, allowing the API to be used with all Windows operating systems (XP/Win7).

The 12AISS44AO4 Win Driver provides an interface to a 12AISS44AO4 card and a Windows application, which run on a x86 or x64 target processor. The driver is installed and devices are created when the driver is started during boot up. The functions of the driver can then be used to access the board. Devices are created with the name “board x” where “x” is the device number. Device numbers start at 1 and for each board found the device number will increment.

Included in the board driver software is a menu driven board application program. This program is delivered undocumented and unsupported but may be used to exercise the card and the device driver. It can also be used as an example for programming the 12AISS44AO4 device.

The user interfaces to the GSC API at the basic level with the following functions:

- Find Boards() - Detects all PLX Devices connected via the PCI Bus.
- Get Handle() - Opens a driver interface to one 12AISS44AO4 card.
- Readlocal32() - Reads local registers from one 12AISS44AO4 card.
- Writelocal32() - Writes to local Registers of one 12AISS44AO4 card.
- Close Handle() - Closes a driver interface to one 12AISS44AO4 card.

The user MUST call Find Boards to determine what PLX devices are installed in the system, and get the associated board number. The user then calls the Get Handle function with each board number to be used. This function obtains a handle to the device and initializes the device parameters within the API / driver. The user is then free (assuming no errors) to write / read the registers as desired. The user should always call Close Handle when done to free resources prior to exiting.

The function definitions and parameters are defined in the following paragraphs of this section.

4.1 AISS44AO4_FindBoards()

Detects all PLX Devices connected via the PCI Bus.

Prototype:

```
U32  AISS44AO4_FindBoards   (char          *pDeviceInfo,
                             U32           *ulError);
```

Returns – Total number of PLX boards found in the system or –1L if error or no boards found.

Where:

pDeviceInfo – Contains “Board #: Bus: Slot: Type: Ser#” info for PLX boards found.

ulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

4.2 AISS44AO4_Get_Handle

Initializes Handle for the passed board number IN THE DRIVER.

Prototype:

```
U32  AISS44AO4_Get_Handle  (U32      *ulError,
                            U32      BoardNumber);
```

Returns – Error code if invalid board number passed (0, >10), else # boards.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

4.3 AISS44AO4_Read_Local32

Read a value from the board local register.

Prototype:

```
U32  AISS44AO4_Read_Local32     (U32     BoardNumber,
                                 U32     *ulError,
                                 U32     ulRegister);
```

Returns – Value read from the register.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

ulRegister – Register to read. Values defined in AISS44AO4eintface.h

```
BCR                 0x00
DIO_IO              0x04
AOUT_00             0x08
AOUT_01             0x0C
AOUT_02             0x10
AOUT_03             0x14
AIN_DATA            0x18
RATE_A              0x1C
RATE_B              0x20
AIN_CONFIG          0x24
AIN_BUFF_SIZE       0x28
AIN_BUFF_THRSHLD    0x2C
ICR                 0x30
BRD_CONFIG          0x34
AUTOCAL             0x38
AUX_REG             0x3C
```

4.4 AISS44AO4_Write_Local32

Write a value to the board local register.

Prototype:

```
void  AISS44AO4_Write_Local32     (U32      BoardNumber,
                                  U32      *ulError,
                                  U32      ulRegister
                                  U32      uiValue);
```

Returns – N/A

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

iRegister – Register to write. Values defined in AISS44AO4eintface.h

BCR	0x00
DIO_IO	0x04
AOUT_00	0x08
AOUT_01	0x0C
AOUT_02	0x10
AOUT_03	0x14
AIN_DATA	0x18
RATE_A	0x1C
RATE_B	0x20
AIN_CONFIG	0x24
AIN_BUFF_SIZE	0x28
AIN_BUFF_THRSHLD	0x2C
ICR	0x30
BRD_CONFIG	0x34
AUTOCAL	0x38
AUX_REG	0x3C

uiValue – Value to write to the selected register.
Refer to the 12AISS44AO4 user manual for all register / bit definitions.

4.5 AISS44AO4_Close_Handle

Closes the device handle and frees the resources.

Prototype:

```
void  AISS44AO4_Close_Handle     (U32     BoardNumber,
                                  U32     *ulError);
```

Returns – N/A

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

4.6 Interface Functions

These functions allow the user to perform certain operations on the board, without having to keep track of individual register values and bit definitions.

4.6.1 AISS44AO4_Initialize

Perform a reset on the board. All register values are set to defaults. This Function does NOT wait for Initialization to complete, such that multiple boards can be initialized without waiting for each to finish.

Prototype:

```
void  AISS44AO4_Initialize     (U32      BoardNumber,
                                U32      *ulError);
```

Returns – N/A

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

4.6.2 AISS44AO4_Autocal

Perform an auto calibration on the board. This operation generates new calibration correction values which are stored in nonvolatile EEprom. This Function waits for Autocal to complete, and returns status.

Prototype:

```
U32  AISS44AO4_Autocal      (U32      BoardNumber,
                             U32      *ulError);
```

Returns – 0xAA for Interrupt timeout, or autocal status (0=FAIL or 1=PASS).

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

4.6.3 AISS44AO4_Set_Input_Mode

Sets the input mode of the board: Differential, Single-Ended, Selftest (zero or Vref) or output channel (0-3) monitor.

Prototype:

```
void  AISS44AO4_Set_Input_Mode   (U32      BoardNumber,
                                  U32      *ulError
                                  U32      ulInputMode);
```

Returns – N/A

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

ulInputMode – Valid values: 0 – 7
- 0 = Differential
- 1 = Single-Ended
- 2 = Zero Selftest
- 3 = +Vref Selftest
- 4 = Monitor Ch0 Output
- 5 = Monitor Ch1 Output
- 6 = Monitor Ch2 Output
- 7 = Monitor Ch3 Output

4.6.4 AISS44AO4_Set_Output_Mode

Sets the Output mode of the board: Immediate or Simultaneous.

Prototype:

```
void  AISS44AO4_Set_Output_Mode (U32     BoardNumber,
                                 U32     *ulError
                                 U32     ulOutputMode);
```

Returns – N/A

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

ulOutputMode – Valid values: 0 = Immediate, 1 = Simultaneous

4.6.5 AISS44AO4_Clear_Input_Buffer

Clears all data from the input buffer.

Prototype:

```
void  AISS44AO4_Clear_Input_Buffer     (U32     BoardNumber,
                                        U32     *ulError);
```

Returns – N/A

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

4.6.6 AISS44AO4_Enable_Input_Buffer

Enables data collection to the input buffer.

Prototype:

```
void  AISS44AO4_Enable_Input_Buffer    (U32    BoardNumber,
                                        U32    *ulError);
```

Returns – N/A

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

4.6.7 AISS44AO4_Disable_Input_Buffer

Disables data collection to the input buffer.

Prototype:

```
void  AISS44AO4_Disable_Input_Buffer    (U32    BoardNumber,
                                        U32    *ulError);
```

Returns – N/A

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

4.6.8 AISS44AO4_EnableInterrupt

Enables the desired interrupt in the local register, and for the PCI bus. See 12AISS44AO4 User manual for interrupt sources.

Prototype:

```
U32  AISS44AO4_EnableInterrupt    (U32    BoardNumber,
                                   U32    ulValue,
                                   U32    *ulError);
```

Returns – Interrupt value set.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulValue – The desired interrupt value to set, valid for 0 – 0x7FF.

ulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

4.6.9 AISS44AO4_DisableInterrupt

Disables all interrupts in the local register, and for the PCI bus.

Prototype:

```
void  AISS44AO4_DisableInterrupt   (U32      BoardNumber,
                                    U32      ulValue,
                                    U32      *ulError);
```

Returns – N/A.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulValue – The desired interrupt value to clear, valid for 0 – 0x7FF.

ulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

4.6.10 AISS44AO4_Open_DMA_Channel

Opens the desired DMA channel for transferring data from the input FIFO.

Prototype:

```
void  AISS44AO4_Open_DMA_Channel(U32     BoardNumber,
                                 U32     ulChannel,
                                 U32     ulMode,
                                 U32     *ulError);
```

Returns – N/A.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulChannel – The desired channel to open, valid for channel 0 or 1.

ulMode – Enables / Disables Demand Mode, 0 = Disable / 1 = Enable, normally enabled.

ulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

4.6.11 AISS44AO4_DMA_FROM_Buffer

Transfers the desired number of WORDS from the board input buffer.

Prototype:

```
U32  AISS44AO4_DMA_FROM_Buffer  (U32     BoardNumber,
                                 U32     ulChannel,
                                 U32     ulWords,
                                 U32*    uData,
                                 U32     *ulError);
```

Returns – WORDS transferred if no error.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulChannel – The DMA channel previously opened, valid for channel 0 or 1.

ulWords – Number of WORDS to transfer. (BYTES = ulWords*4).

ulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

4.6.12 AISS44AO4_Close_DMA_Channel

Closes the desired DMA channel.

Prototype:

```
void  AISS44AO4_Close_DMA_Channel(U32     BoardNumber,
                                  U32     ulChannel,
                                  U32     *ulError);
```

Returns – N/A.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

uiValue – The desired channel to close, valid for channel 0 or 1.

ulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

4.6.13 AISS44AO4_Register_Interrupt_Notify

Registers a user event for interrupt notification.
NOTE: This function DOES NOT enable the interrupts the notification is based on.

Prototype:

```
void  AISS44AO4_Register_Interrupt_Notify      (U32                BoardNumber,
                                               GS_NOTIFY_OBJECT    *event,
                                               U32                 ulIntr,
                                               U32                 *ulError);
```

Returns – N/A.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

event – User creates an event and stores the handle in a GS_NOTIFY_OBJECT structure member hEvent. The user should not set or change the other members of the structure.

ulIntr – Interrupt value to associate with the event.

ulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

4.6.14 AISS44AO4_Cancel_Interrupt_Notify

Cancels interrupt notification for a user event.

Prototype:

```
void  AISS44AO4_Cancel_Interrupt_Notify  (U32                  BoardNumber,
                                         GS_NOTIFY_OBJECT     *event,
                                         U32                  *ulError);
```

Returns – N/A.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

event – A valid GS_NOTIFY_OBJECT structure which has been initialized by a call to AISS44AO4_Register_Interrupt_Notify.

ulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

5. Driver Installation

This section details driver installation on the target system. Any current driver previously installed for the 12AISS44AO4 must be uninstalled prior to this installation to avoid interference.

To install the driver, API, and associated example files, insert the CD ROM into the drive and close the bay. The installation should commence automatically and display user prompts. Follow the onscreen instructions to complete the installation.

Should the installation fail to automatically start, Select ***Start → Run → Browse*** on the Windows toolbar/popup and browse to find ***Setup.exe*** on the CD ROM. Click on ***OK*** to commence the installation.

The following files are installed on the target system:

OS dependent\…\GS12AISS.sys
OS dependent\…\GS12AISSM.sys
OS dependent\…\GSebApi.dll
OS dependent\…\12AISS44edriver.inf
Program Files\General Standards\12AISS44\eExample.exe
Program Files\General Standards\12AISS44\AISS44AO4 eDriver C.dll
Program Files\General Standards\12AISS44\AISS44AO4 eDriver C.lib
Program Files\General Standards\12AISS44\AISS44AO4eintface.h
Program Files\General Standards\12AISS44\AISS44AO4_Example.c
Program Files\General Standards\12AISS44\Tools.c
Program Files\General Standards\12AISS44\Tools.h
Program Files\General Standards\12AISS44\CioColor.h
Program Files\General Standards\12AISS44\12AISS44ebdriver.inf

6. Example Program

This section describes the example program, and the files required to develop an application.

The complied example program allows the user to exercise the installed device, while observing the outputs. To execute, double click on 'Example.exe'. Refer to the Driver Installation section for file location.

The source is provided to educate the user with the GSC API function calls and provide a working example to aid the user with application development. To build the example program using MS Visual C++, create a project and add the following files:

Source Files	→ ***AISS44AO4_Example.c***
	→ ***Tools.c***
Header Files	→ ***AISS44AO4eintface.h***
	→ ***CioColor.h***
	→ ***Tools.h***
Resource Files	→ ***AISS44AO4 eDriver C.lib***

Select ***Build → [ProjectName].exe*** on the toolbar.
NOTE: AISS44AO4 eDriver C.dll must be in the project directory to run the example.

Contact GSC for example programs (drivers) for other development environments (i.e LabVIEW™, LabWindows/CVI™, etc.)