General Standards Corporation
SIO4 Application Interface
Users Manual

v1.2.9

March 30, 2005

Table of Contents

Introduction

This document describes the Application Programmers Interface (API) for the General Standards Corporation I/O Interface boards. Some API functions apply only to certain hardware. Each function contains the list of boards that it supports. For examples of how to use the API functions, refer to the source code included in the API examples. These examples are located in the sub-directories named samples/SIO4B_Test.

This API was written using Microsoft Visual C++ 6.0 and is compatible with both Win32 and MFC applications.

Installation

The API support files are installed during the standard installation of the driver. The API support files are placed, by default, into the C:\Program Files\General Standards Corporation\GscApi\ directory and subdirectories and consist of the following files:

GscApi.h – This is the header file that should be included in any source files that utilize the API. This file contains the function prototypes and constant definitions needed to access the API.

GscApi.lib – This is the import library file that should be included in your project so that the linker can find the API functions.

GscApi.dll – This is the dynamically linked library file that contains the actual API code. It should be located in the same directory as your executable or in your system path so that your application can access the API functions. This file will also be installed to your system32 directory during installation.

Below is a list of files copied by the installer during the installation process:

Critical System Files:

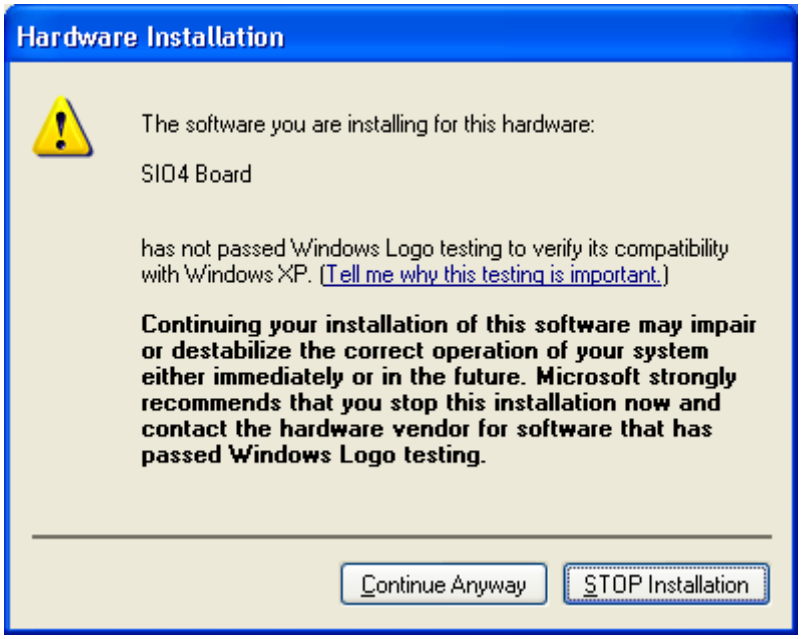
%WINDIR%\system32\GscApi.dll
%WINDIR%\system32\PlxApi.dll
%WINDIR%\inf\SIO4Cdriver.inf
%WINDIR%\system32\drivers\PciSIO4.sys

Examples/Documentation:

%PROGRAMFILES% \General Standards Corporation\GscApi\

It is recommended that you install the driver/API before installing the SIO4B card. After the installation completes, shut the system down and install the SIO4B card.

Under Windows XP, you may get the following warning during the Hardware Wizard's installation of the card. You can safely choose Continue Anyway to install the driver.



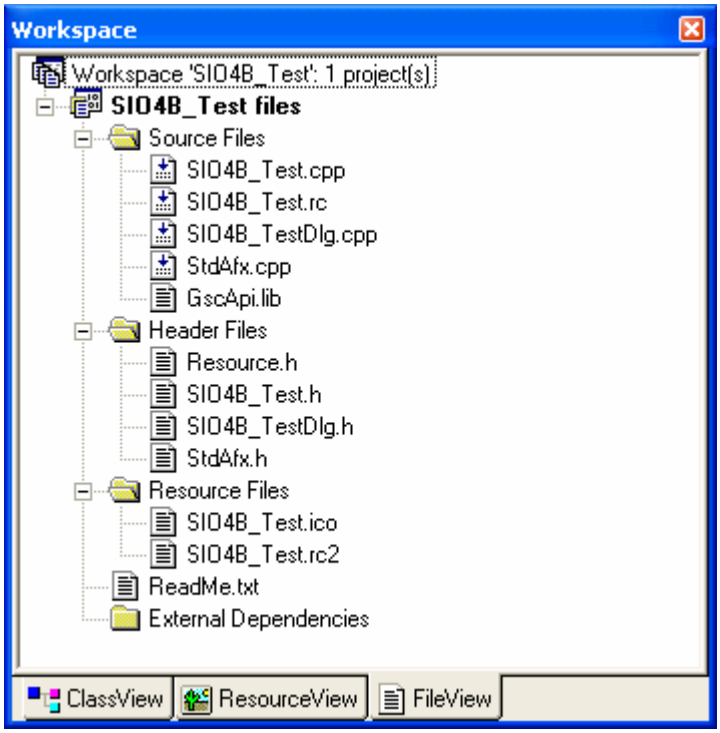
Project Setup

To utilize the SIO4B-API in your software application, you should include the GscApi.h header file and the GscApi.lib static library file in your project. The details of adding these files to your project will differ from compiler to compiler. We will concentrate on the Microsoft® Visual C++ IDE. Support for other compilers may be added in future releases of the SIO4B-API.

First, be sure that your source code file that will make use of the SIO4B-API has the GscApi.h header file included as follows:

```
#include "GscApi.h"
```

Next, be sure that your project has the GscApi.lib static library file included to be compiled as part of your project as follows (here is a sample of the workspace for the SIO4B_Test included with the SIO4B-API in the Samples directory):



Lastly, the GscApi.dll file should be made available to your final executable program – either by having this file in the same directory or making it available via your system's path. By default, the installer copies the GscApi.dll file to the system32 directory so it should be available to all applications.

System Level Routines

The System Level Routines perform functions that either apply to all SIO4 boards in the system, or are not board specific. These routines are used to gather information about the current system setup. All of these functions return zero if successful or a non-zero error code if a failure occurs.

GscSio4FindBoards

GscSio4FindBoards(…) is used to report the number of GSC SIO4 boards in the system as well as some board specific information. An application may call this function at any time.

Supported Hardware:

All

Prototype:

```
int GscSio4FindBoards(
                        int *boardCount,
                        GSC_DEVICES_STRUCT *results);
```

Parameters:

boardCount – a pointer to the location to save the number of boards detected. This value will be zero if no boards are found.

results – a pointer to the devices structure that will be filled in with the information from the boards found. If this parameter is NULL, no board specific information will be returned. The boardCount will, however, still be returned. The devices structure is defined as follows:

```
typedef struct
{
        int     busNumber;          // Identifies the bus that contains the board
        int     slotNumber;         // Identifies the slot that contains the board
        int     vendorId;           // Identifies the board Vendor
        int     deviceId;           // Identifies the device
        char    serialNumber[25];   // A unique board serial number
} GSC_DEVICES_STRUCT;
```

GscSio4GetErrorString

GscSio4GetErrorString(…) is used to translate the error codes that are returned by the various API functions into meaningful null-terminated strings. The strings returned by this function are guaranteed to be less than 80 characters in length.

Supported Hardware:

All

Prototype:

```
int GscSio4GetErrorString(
                                int errorCode,
                                char *errorString);
```

Parameters:

errorCode – the error code returned by an API function.

errorString – a pointer to a character string that will be filled with the text that corresponds to the errorCode.

Board Level Routines

The Board Level Routines perform functions that apply to a single SIO4 board. These functions affect all channels of the SIO4 board. Each of these routines requires the board number (*boardNumber*) as the first argument. The board numbers run from 1 up to the number that is returned from the call to GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

These routines can be called at any time. All of these functions return zero if successful or a non-zero error code if a failure occurs.

GscSio4Open

GscSio4Open(…) is used to "open" the SIO4 board for operation. It should be called before any other Board or Channel Level routines and should only be called once. In the process of opening a board, all four channels are reset and the clock outputs are disabled.

Supported Hardware:

All

Prototype:

int GscSio4Open(int boardNumber);

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

GscSio4Close

GscSio4Close(…) is used to "close" the SIO4 board. It should be the last API function called before the application terminates. This function releases the resources that are used by the API and driver.

Supported Hardware:

All

Prototype:

int GscSio4Close(int boardNumber);

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

GscSio4GetVersions

GscSio4GetVersions(…) returns the various version numbers associated with the API, the low level driver, and the SIO4 board's FPGA. The Library and Driver version numbers are returned in the form: 0x00MMmmee where MM is the major release number, mm is the minor release number, and ee is the engineering release number. The entire version is defined as MM.mm.ee for example 1.02.05 is returned as 0x00010205. The FPGA version number has several encoded fields. The low byte contains the actual version number. Refer to the hardware users manual for details on the other encoded fields.

Supported Hardware:

All

Prototype:

```
int GscSio4GetVersions(
                        int boardNumber,
                        int *libVersion,
                        int *driverVersion,
                        int *fpgaVersion);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

libVersion – A pointer to the location that will receive the library (API) version number. If this value is NULL, no value will be returned.

driverVersion – A pointer to the location that will receive the low level driver version number. If this value is NULL, no value will be returned.

fpgaVersion – A pointer to the location that will receive the FPGA firmware version number. If this value is NULL, no value will be returned.

GscSio4LocalRegisterRead

GscSio4LocalRegisterRead(…) is used to read the local board registers. These registers reside within the board's FPGA. It is not recommended that a user application directly access these registers. This function is included for diagnostic purposes only.

Supported Hardware:

All

Prototype:

```
int GscSio4LocalRegisterRead(
                                    int boardNumber,
                                    int reg,
                                    int *result);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

reg – The address of the register to be read. Macros for these addresses are described in the section titled "Local Register Definitions".

result – A pointer to the location that will receive the results of the read operation.

GscSio4LocalRegisterWrite

GscSio4LocalRegisterWrite(…) is used to write to the local board registers. These registers reside within the board's FPGA. It is not recommended that a user application directly access these registers. This function is included for diagnostic purposes only.

Supported Hardware:

All

Prototype:

```
int GscSio4LocalRegisterWrite(
                                    int boardNumber,
                                    int reg,
                                    int value);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

reg – The address of the register to be written. Macros for these addresses are described in the section titled "Local Register Definitions".

value – The value that is to be written to the local register.

Channel Level Routines

The Channel Level Routines perform functions that apply to a single channel on an SIO4 board. Each of these routines requires the board number (boardNumber) as the first parameter and the channel number (channel) as the second parameter. The board number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system. The channel number will always be 1, 2, 3, or 4.

These routines can be called at any time. All of these functions return zero if successful or a non-zero error code if a failure occurs.

GscSio4ChannelReset

GscSio4ChannelReset(…) resets a single channel on the SIO4 board. In addition to disabling the serial channel, this function sets the “Almost Empty” and “Almost Full” FIFO flags to 16.

Supported Hardware:

All

Prototype:

```
int GscSio4ChannelReset(
                        int boardNumber,
                        int channel);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

GscSio4ChannelResetRxFifo

GscSio4ChannelResetRxFifo(…) resets the Rx FIFO for a single channel. After the reset, the FIFO will contain no data.

Supported Hardware:

All

Prototype:

```
int GscSio4ChannelResetRxFifo (
                                    int boardNumber,
                                    int channel);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

GscSio4ChannelResetTxFifo

GscSio4ChannelResetTxFifo(…) resets the Tx FIFO for a single channel. After the reset, the FIFO will contain no data.

Supported Hardware:

All

Prototype:

```
int GscSio4ChannelResetTxFifo (
                                    int boardNumber,
                                    int channel);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

GscSio4ChannelRegisterRead

GscSio4ChannelRegisterRead(…) is used to read the registers in the Universal Serial Chip that controls the specified channel. It is not recommended that a user application directly access these registers. This function is included for diagnostic purposes only.

Supported Hardware:

PCI-SIO4B

Prototype:

```
int GscSio4ChannelRegisterRead(
                                    int boardNumber,
                                    int channel,
                                    int reg,
                                    int *value);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

reg – The address of the register to be read. Macros for these addresses are described in the section titled “Channel Register Definitions”.

value – A pointer to the location that will receive the results of the read operation.

GscSio4ChannelRegisterWrite

GscSio4ChannelRegisterWrite(…) is used to write to the registers in the Universal Serial Chip that controls the specified channel. It is not recommended that a user application directly access these registers. This function is included for diagnostic purposes only.

Supported Hardware:

PCI-SIO4B

Prototype:

```
int GscSio4ChannelRegisterWrite(
                                    int boardNumber,
                                    int channel,
                                    int reg,
                                    int value);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

reg – The address of the register to be written. Macros for these addresses are described in the section titled “Channel Register Definitions”.

value – The value that is to be written to the register.

GscSio4ChannelSetMode / GscSio4ChannelGetMode

GscSio4ChannelSetMode(…) sets a single channel of the SIO4 board to the desired serial format and bit rate.

Each mode has its own defaults, as described below, which can be altered by calling the appropriate Channel Level Routines after this function returns.

Supported Hardware:

All

Prototype:

```
int GscSio4ChannelSetMode(
                                int boardNumber,
                                int channel,
                                int mode,
                                int bitRate);

int GscSio4ChannelGetMode(
                                int boardNumber,
                                int channel,
                                int *mode,
                                int *bitRate);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

mode – The desired/current serial mode for this channel. The value should be one of the following:

GSC_MODE_ASYNC – Sets the channel to standard asynchronous mode. The channel defaults to 8 data bits, no Parity, and one stop bit. It also uses a 16x sampling clock.

GSC_MODE_ISO – Sets the channel to isochronous mode. Uses the same defaults as GSC_MODE_ASYNC except the sampling clock, which is set to 1x.

GSC_MODE_HDLC – Sets the channel to HDLC mode. The Transmit clock is derived from the on-board source at the rate specified (*bitRate*) and is also driven onto the cable for use by the receiving end. The receiver clock is

connected to the cable and should be supplied by the transmitter at the other end.

GSC_MODE_SYNC -
GSC_MODE_SYNC_ENV – (SIO4-SYNC boards only)
GSC_MODE_ASYNC_CV -
GSC_MODE_MONOSYNC -
GSC_MODE_BISYNC -
GSC_MODE_TRANS_BISYNC –

GSC_MODE_NBIF -
GSC_MODE_802_3 -

bitRate – The desired/current serial bit (baud) rate for this channel. This value can range from 250 to 10,000,000 for synchronous modes and 50 to 1,000,000 for asynchronous modes.

GscSio4ChannelSetDataSize / GscSio4ChannelGetDataSize

GscSio4ChannelSetDataSize(…) sets the size of the transmitted and received data for a single channel of the SIO4 board. The data size can be set to any value between 1 and 8 inclusive on the standard SIO4 boards. The data size can be set to any value between 1 and 65535 on the –SYNC boards.

Supported Hardware:

ALL

Prototype:

```
int GscSio4ChannelSetDataSize(
                                    int boardNumber,
                                    int channel,
                                    int dataSize);

int GscSio4ChannelGetDataSize(
                                    int boardNumber,
                                    int channel,
                                    int *dataSize);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

dataSize – The desired/current data size for this channel. The value should be between 1 and 8 on the standard boards and between 1 and 65535 on the -SYNC boards.

GscSio4ChannelSetGapSize / GscSio4ChannelGetGapSize

GscSio4ChannelSetGapSize(…) sets the size of the gap between transmitted data words for a single channel of the SIO4 board. The gap size can be set to any value between 0 and 65535.

Supported Hardware:

SIO4-SYNC

Prototype:

```
int GscSio4ChannelSetGapSize(
                                int boardNumber,
                                int channel,
                                int gapSize);

int GscSio4ChannelGetGapSize(
                                int boardNumber,
                                int channel,
                                int *gapSize);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

gapSize – The desired/current gap size for this channel. The value should be between 1 and 65535.

GscSio4ChannelSetMsbLsbOrder / GscSio4ChannelGetMsbLsbOrder

GscSio4ChannelSetMsbLsbOrder(…) sets the order that the bits are shifted out onto the “wire”.

Supported Hardware:

SIO4-SYNC

Prototype:

```
int GscSio4ChannelSetGapSize(
                                    int boardNumber,
                                    int channel,
                                    int txOrder,
                                    int rxOrder);

int GscSio4ChannelGetGapSize(
                                    int boardNumber,
                                    int channel,
                                    int *txOrder,
                                    int *rxOrder);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

txOrder – The desired/current MSB/LSB ordering for the transmitter. The valid values are either GSC_MSB_FIRST or GSC_LSB_FIRST.

rxOrder – The desired/current MSB/LSB ordering for the receiver. The valid values are either GSC_MSB_FIRST or GSC_LSB_FIRST.

GscSio4ChannelSetParity / GscSio4ChannelGetParity

GscSio4ChannelSetParity(…) sets the type of parity that will be used on a single channel of the SIO4 board. The parity can be set to None, Even Odd, Space, or Mark.

Supported Hardware:

PCI-SIO4B

Prototype:

```
int GscSio4ChannelSetParity(
                                int boardNumber,
                                int channel,
                                int parity);

int GscSio4ChannelGetParity(
                                int boardNumber,
                                int channel,
                                int *parity);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

parity – The desired/current parity type for this channel. The value should be one of the following:

GSC_PARITY_NONE – Sets the channel to no parity.

GSC_PARITY_EVEN – Sets the channel to Even parity.

GSC_PARITY_ODD – Sets the channel to Odd parity.

GSC_PARITY_MARK - Sets the channel to Mark parity.

GSC_PARITY_SPACE - Sets the channel to Space parity.

GscSio4ChannelSetStopBits / GscSio4ChannelGetStopBits

GscSio4ChannelSetStopBits(…) sets the number of stop bits to use for a single channel of the SIO4 board. The number of stop bits can be set to 0, 1, 1 ½, or 2 bits.

Supported Hardware:

PCI-SIO4B

Prototype:

```
int GscSio4ChannelSetStopBits(
                                    int boardNumber,
                                    int channel,
                                    int stopBits);

int GscSio4ChannelGetStopBits(
                                    int boardNumber,
                                    int channel,
                                    int *stopBits);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

stopBits – The desired/current number of stop bits for this channel. The value should be one of the following:

GSC_STOP_BITS_0 – Sets the channel to no stop bits. This setting can only be used when the channel is set to Async with Code Violation (MIL STD 1553) mode.

GSC_STOP_BITS_1 – Sets the channel to 1 stop bit.

GSC_STOP_BITS_1_5 – Sets the channel to 1 ½ stop bits.

GSC_STOP_BITS_2 - Sets the channel to 2 stop bits.

GscSio4ChannelSetEncoding / GscSio4ChannelGetEncoding

GscSio4ChannelSetEncoding(…) sets the encoding type for a single channel of the SIO4 board. The encoding can be set to NRZ, NRZB, NRZI Mark, NRZI Space, Biphase Mark, Biphase Space, Biphase Level, or Differential Biphase Level on standard boards. The encoding can be set to only NRZ or NRZB on the –SYNC boards.

Supported Hardware:

All

Prototype:

```
int GscSio4ChannelSetEncoding(
                                    int boardNumber,
                                    int channel,
                                    int encoding);

int GscSio4ChannelGetEncoding(
                                    int boardNumber,
                                    int channel,
                                    int *encoding);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

encoding – The desired/current encoding type for this channel. This value should be one of the macros defined in the section “Channel Encoding Definitions”.

GscSio4ChannelSetProtocol / GscSio4ChannelGetProtocol

GscSio4ChannelSetProtocol(…) sets the physical interface protocol and termination options. The protocol on the standard SIO4B card is fixed at RS422/RS485 or RS232 depending on the configuration set at the factory. Only the –BX cards allow this value to be changed.

Supported Hardware:

PCI-SIO4B-BX

Prototype:

```
int GscSio4ChannelSetProtocol(
                                    int boardNumber,
                                    int channel,
                                    int protocol,
                                    int termination);

int GscSio4ChannelGetProtocol(
                                    int boardNumber,
                                    int channel,
                                    int *protocol,
                                    int *termination);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

protocol – The desired/current physical interface protocol. The value can be any of the following:

GSC_PROTOCOL_RS422_RS485 – Sets
GSC_PROTOCOL_RS423 – Sets
GSC_PROTOCOL_RS232 – Sets
GSC_PROTOCOL_RS530_1 – Sets
GSC_PROTOCOL_RS530_2 – Sets
GSC_PROTOCOL_V35_1 – Sets
GSC_PROTOCOL_V35_2 – Sets
GSC_PROTOCOL_RS422_RS423_1 – Sets
GSC_PROTOCOL_RS422_RS423_2 – Sets

termination – The desired/current termination setting. The value can be any of the following:

GSC_TERMINATION_ENABLED – Sets

GSC_TERMINATION_DISABLED – Sets

GscSio4ChannelSetDteDce / GscSio4ChannelGetDteDce

GscSio4ChannelSetDteDce(…) sets a single channel of the SIO4 board to either DTE or DCE mode. Each channel defaults to DTE mode when it is configured. Calling this routine is only necessary if DCE mode is required, or to switch back to DTE mode after a previous change to DCE mode.

The pin-outs for both DTE and DCE modes are available in the Hardware Users Manual.

Supported Hardware:

All

Prototype:

```
int GscSio4ChannelSetDteDce(
                                    int boardNumber,
                                    int channel,
                                    int mode);

int GscSio4ChannelGetDteDce(
                                    int boardNumber,
                                    int channel,
                                    int *mode);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

mode – The desired/current DTE/DCE mode for this channel. The value should be one of the following:

GSC_PIN_DTE – Sets the channel to DTE mode. See the hardware manual for the cable pin-out.

GSC_PIN_DCE - Sets the channel to DCE mode. See the hardware manual for the cable pin-out.

GscSio4ChannelSetLoopBack / GscSio4ChannelGetLoopBack

This function

Supported Hardware:

All

Prototype:

```
int GscSio4ChannelSetLoopBack(
                                    int boardNumber,
                                    int channel,
                                    int loopMode);

int GscSio4ChannelGetLoopBack(
                                    int boardNumber,
                                    int channel,
                                    int *loopMode);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

loopMode – The desired/current loopback mode.

GscSio4ChannelSetPinMode / GscSio4ChannelGetPinMode

GscSio4ChannelSetPinMode(…)

Supported Hardware:

All

Prototype:

```
int GscSio4ChannelSetPinMode (
                                    int boardNumber,
                                    int channel,
                                    int pinName,
                                    int mode);

int GscSio4ChannelGetPinMode (
                                    int boardNumber,
                                    int channel,
                                    int pinName,
                                    int *mode);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

GscSio4ChannelSetPinValue / GscSio4ChannelGetPinValue

GscSio4ChannelSetPinValue(…)

Supported Hardware:

All

Prototype:

```
int GscSio4ChannelSetPinValue (
                                    int boardNumber,
                                    int channel,
                                    int pinName,
                                    int value);

int GscSio4ChannelGetPinValue (
                                    int boardNumber,
                                    int channel,
                                    int pinName,
                                    int *value);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

GscSio4ChannelFifoSizes

GscSio4ChannelFifoSizes(…) returns the size, in bytes, of the channel's Transmit and Receive FIFOs. The size of the Transmit FIFO is returned in the upper 16 bits and the size of the Receive FIFO is returned in the lower 16 bits of the result (*sizes*).

Supported Hardware:

PCI-SIO4B

Prototype:

```
int GscSio4ChannelFifoSizes(
                                    int boardNumber,
                                    int channel,
                                    int *sizes);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

sizes – A pointer to the location that will receive the size (in bytes) of the Transmit (upper 16 bits) and the Receive (lower 16 bits) FIFOs

GscSio4ChannelFifoCounts

GscSio4ChannelFifoCounts(…) returns the current number of bytes in the channel's Transmit and Receive FIFOs. The number of bytes in the Transmit FIFO are returned in the upper 16 bits and the number of bytes in the Receive FIFO are returned in the lower 16 bits of the result (*counts*).

Supported Hardware:

PCI-SIO4B

Prototype:

```
int GscSio4ChannelFifoCounts(
                                    int boardNumber,
                                    int channel,
                                    int *counts);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

counts – A pointer to the location that will receive the number of bytes currently in the Transmit (upper 16 bits) and the Receive (lower 16 bits) FIFOs.

GscSio4ChannelSetTxAlmost / GscSio4ChannelGetTxAlmost

GscSio4ChannelSetTxAlmost(…) programs the "Almost Full" and "Almost Empty" registers in the Transmit FIFO for a single channel. Once the values are programmed, the FIFO will be reset to force the change to take effect. This will also clear the contents of the FIFO, so this command should be done before any data transfers occur.

Supported Hardware:

PCI-SIO4B

Prototype:

```
int GscSio4ChannelSetTxAlmost(
                                    int boardNumber,
                                    int channel,
                                    int almostValue);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

almostValue – The 32bit value that will be programmed into the Transmitter FIFO's Almost Full (upper 16 bits) and Almost Empty (lower 16 bits) registers.

GscSio4ChannelSetRxAlmost / GscSio4ChannelGetRxAlmost

GscSio4ChannelSetRxAlmost(…) programs the "Almost Full" and "Almost Empty" registers in the Receive FIFO for a single channel. Once the values are programmed, the FIFO will be reset to force the change to take effect. This will also clear the contents of the FIFO, so this command should be done before any data transfers occur.

Supported Hardware:

PCI-SIO4B

Prototype:

```
int GscSio4ChannelSetRxAlmost(
                                    int boardNumber,
                                    int channel,
                                    int almostValue);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

almostValue – The 32bit value that will be programmed into the Receiver FIFO's Almost Full (upper 16 bits) and Almost Empty (lower 16 bits) registers.

GscSio4ChannelCheckForData

GscSio4ChannelCheckForData(…) checks for the reception of a complete packet on the Channel's Receiver. If no complete packets have been received, the routine returns immediately with a count of 0x00. If a packet is ready, the routine returns with count set to the length of the data packet and the packet contents are copied into the supplied buffer.

This routine is only available when packet framing is enabled for the channel. Packet framing is currently supported only when the channel is configured for HDLC.

Supported Hardware:

PCI-SIO4B

Prototype:

```
int GscSio4ChannelCheckForData(
                                    int boardNumber,
                                    int channel,
                                    char *buffer,
                                    int *count);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

buffer – A pointer to the start of the data buffer that will receive the data. The buffer should be long enough to hold the longest possible packet.

count – The number of bytes that were contained in the packet and transferred to *buffer*. *Count* will be 0x00 if no packets were available.

GscSio4ChannelReceiveData

GscSio4ChannelReceiveData(…) starts the reception of data on the specified channel. The data received on the channel is transferred into the memory buffer pointed to by *buffer*. A total of *count* bytes will be transferred. This function may return before the transfer completes. When this function returns, the value pointed to by *id* will contain a unique identifier that can be used to determine the progress of the transfer.

Supported Hardware:

All

Prototype:

```
int GscSio4ChannelReceiveData(
                                    int boardNumber,
                                    int channel,
                                    char *buffer,
                                    int count,
                                    int *id);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

buffer – A pointer to the start of the data buffer that will receive the data. The buffer should be at least *count* bytes long.

count – The number of bytes to transfer.

id – A pointer to the location that will hold the unique transfer identifier that is assigned to this transfer. This value can be used to determine when the transfer has completed.

GscSio4ChannelReceiveDataAndWait

GscSio4ChannelReceiveDataAndWait(…) starts the reception of data on the specified channel. The data received on the channel is transferred into the memory buffer pointed to by *buffer*. A total of *count* bytes will be transferred. This function will not return until the entire transfer has completed or the timeout period has expired. If a timeout occurs, the value in *bytesTransferred* will specify the number of bytes that were actually received. (Note that if no timeout occurs, the *bytesTransferred* value is undefined.)

Supported Hardware:

All

Prototype:

```
int GscSio4ChannelReceiveDataAndWait(
                                    int boardNumber,
                                    int channel,
                                    char *buffer,
                                    int count,
                                    int timeout,
                                    int *bytesTransferred);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

buffer – A pointer to the start of the data buffer that will receive the data. The buffer should be at least *count* bytes long.

count – The number of bytes to transfer.

timeout – The desired timeout period (in milliseconds) for the transfer.

bytesTransferred – If a timeout occurs, this value will specify the total number of bytes that were actually received. If no timeout occurs, this value is undefined.

GscSio4ChannelTransmitData

GscSio4ChannelTransmitData(…) starts the transmission of data on the specified channel. The data to be transmitted on the channel is transferred from the memory buffer pointed to by *buffer*. A total of *count* bytes will be transferred. This function may return before the transfer completes. When this function returns, the value pointed to by *id* will contain a unique identifier that can be used to determine the progress of the transfer.

Supported Hardware:

All

Prototype:

```
int GscSio4ChannelTransmitData(
                                    int boardNumber,
                                    int channel,
                                    char *buffer,
                                    int count,
                                    int *id);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

buffer – A pointer to the start of the data buffer that will be transmitted. The buffer should be at least *count* bytes long.

count – The number of bytes to transfer.

id – A pointer to the location that will hold the unique transfer identifier that is assigned to this transfer. This value can be used to determine when the transfer has completed.

GscSio4ChannelTransmitDataAndWait

GscSio4ChannelTransmitDataAndWait(…) starts the transmission of data on the specified channel. The data to be transmitted on the channel is transferred from the memory buffer pointed to by *buffer*. A total of *count* bytes will be transferred. This function will not return until the entire transfer has completed or the timeout period has expired. If a timeout occurs, the value in *bytesTransferred* will specify the number of bytes that were actually transmitted. (Note that if no timeout occurs, the *bytesTransferred* value is undefined.)

Supported Hardware:

All

Prototype:

```
int GscSio4ChannelTransmitData(
                                    int boardNumber,
                                    int channel,
                                    char *buffer,
                                    int count,
                                    int timeout
                                    int *bytesTransferred);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

buffer – A pointer to the start of the data buffer that will be transmitted. The buffer should be at least *count* bytes long.

count – The number of bytes to transfer.

timeout – The desired timeout period (in milliseconds) for the transfer.

bytesTransferred – If a timeout occurs, this value will specify the total number of bytes that were actually transmitted. If no timeout occurs, this value is undefined.

GscSio4ChannelQueryTransfer

GscSio4ChannelQueryTransfer(…) is used to determine the status of a transfer that was initiated by a call to either GscSio4ChannelReceiveData (…) or GscSio4ChannelTransmitData (…). The result is returned in *stat* and will be 0 if the transfer has completed or non-zero if it has not completed.

Supported Hardware:

All

Prototype:

```
int GscSio4ChannelQueryTransfer(
                                    int boardNumber,
                                    int channel,
                                    int *stat,
                                    int id);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

stat – A pointer to the location that will hold the returned status of the transfer. The *stat* will be 0 if the transfer has completed. Otherwise, it will hold the number of bytes left to transfer.

id – The unique ID that was assigned to the transfer by the call to either GscSio4ChannelReceiveData(…) or GscSio4ChannelTransmitData(…)

GscSio4ChannelWaitForTransfer

GscSio4ChannelWaitForTransfer (…) is used to wait for the completion of a transfer that was initiated by a call to either GscSio4ChannelReceiveData (…) or GscSio4ChannelTransmitData (…). The routine will return when either the transfer completes or the timeout period expires. If the timeout period expires, the *bytesTransferred* parameter will be updated with the number of bytes that were successfully transferred. If the transfer completes, or another type of error occurs, the *bytesTransferred* parameter will be -1.

Supported Hardware:

All

Prototype:

```
int GscSio4ChannelWaitForTransfer(
                                    int boardNumber,
                                    int channel,
                                    int timeout,
                                    int id.
                                    int *bytesTransferred);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

timeout – The desired timeout value in milliseconds that the routine will wait for the transfer to complete.

id – The unique ID that was assigned to the transfer by the call to either GscSio4ChannelReceiveData(…) or GscSio4ChannelTransmitData(…)

bytesTransferred - A pointer to the location that will hold the number of bytes that were actually transferred if the timeout period expires. This value will be -1 if the transfer completes, or an error occurs.

GscSio4ChannelFlushTransfer

GscSio4ChannelFlushTransfer (…) is used to force any data that is in the Rx FIFO to be transferred via DMA to memory. For a Tx channel, data is transferred to the Tx FIFO until it is full. Calling this routine is only necessary when a transfer did not complete on its own, or when aborting a transfer that has not completed.

Supported Hardware:

All

Prototype:

```
int GscSio4ChannelFlushTransfer(
                                    int boardNumber,
                                    int channel,
                                    int id);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

id – The unique ID that was assigned to the transfer by the call to either GscSio4ChannelReceiveData(…) or GscSio4ChannelTransmitData(…)

GscSio4ChannelRemoveTransfer

GscSio4ChannelRemoveTransfer (…) is used to remove a pending transfer from the transfer queue. Calling this routine is only necessary when a transfer did not complete on its own, or to abort a transfer that has not completed. If a transfer ID of -1 is passed to the routine, all pending transfers will be removed.

Supported Hardware:

All

Prototype:

```
int GscSio4ChannelRemoveTransfer(
                                        int boardNumber,
                                        int channel,
                                        int id,
                                        int *bytesTransferred);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

id – The unique ID that was assigned to the transfer by the call to either GscSio4ChannelReceiveData(…) or GscSio4ChannelTransmitData(…)

bytesTransferred - A pointer to the location that will hold the number of bytes that were actually transferred before the call to GscSio4ChannelRemoveTransfer (). This value will be -1 if the transfer had already completed, or an error occurs.

GscSio4ChannelRegisterInterrupt

GscSio4ChannelRegisterInterrupt (…) is used register a callback routine with the interrupt handler. There are several interrupt sources associated with each interrupt. This routine allows any or all of the interrupt sources to be associated with a callback function. The callback function can be shared between interrupt sources or a different callback can be used for each source. This routine also determines whether the interrupt occurs on the Rising Edge (High True) or Falling Edge (Low True).

Supported Hardware:

All

Prototype:

```
int GscSio4ChannelRegisterInterrupt(
                                    int boardNumber,
                                    int channel,
                                    int interrupt,
                                    int type,
                                    GSC_CB_FUNCTION function);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

interrupt – This value determines which interrupts are associated with the provided callback function. This value should be the logical OR of one or more of the following:

GSC_INTR_SYNC_DETECT – Triggers an interrupt when the SYNC byte is received on the channel. (This source is not available on the –Sync boards)

GSC_INTR_USC – Triggers an interrupt when the on board USC has an interrupt pending. Refer to the USC data sheet for details of its possible interrupt sources. (This source is not available on the –Sync boards)

GSC_INTR_TX_FIFO_EMPTY – Triggers an interrupt when the Transmit FIFO for the channel is empty.

GSC_INTR_TX_FIFO_FULL – Triggers an interrupt when the Transmit FIFO for the channel is full.

GSC_INTR_TX_FIFO_ALMOST_EMPTY – Triggers an interrupt when the Transmit FIFO for the channel is almost empty. The level at which this interrupt will occur is set by calling the GscSio4ChannelSetTxAlmost(…) routine.

GSC_INTR_RX_FIFO_EMPTY – Triggers an interrupt when the Receive FIFO for the channel is empty.

GSC_INTR_RX_FIFO_FULL – Triggers an interrupt when the Receive FIFO for the channel is full.

GSC_INTR_RX_FIFO_ALMOST_FULL – Triggers an interrupt when the Receive FIFO for the channel is almost full. The level at which this interrupt will occur is set by calling the GscSio4ChannelSetRxAlmost(…) routine.

GSC_INTR_RX_ENVELOPE – Triggers an interrupt when the RX Envelope signal changes. (This source is only available on the –Sync boards)

type – This value determines whether the interrupt occurs on the rising of falling edge. It should be one of the following:

GSC_INTR_RISING_EDGE – The interrupt will occur on the rising edge of the interrupt signal (i.e. when the condition becomes true.)

GSC_INTR_FALLING_EDGE – The interrupt will occur on the falling edge of the interrupt signal (i.e. when the condition becomes not true.)

function – This is the address of the interrupt callback function. If this value is set to NULL, the callback for the current “interrupt” parameter will be cleared, otherwise this routine will be called for each of the conditions specified in the “interrupt” parameter. The prototype for the callback function is:

```
void callback_function(
                        int boardNumber,
                        int channel,
                        int interrupt);
```

The parameters to the callback specify the board and channel number on which the interrupt occurred as well as the source of the interrupt (as defined above.) If multiple interrupt sources are mapped to the same callback routine, the “interrupt” value can be used to determine the source of the interrupt.

GscSio4ChannelSetClock

GscSio4ChannelSetClock(…) is used to set the serial bit rate (baud rate) for a specific channel. Under normal conditions, this routine will not be used since the GscSio4ChannelSetMode(…) function sets the channels bit rate when the channel's mode is set. This function is provided to allow the bit rate to be changed without re-configuring the channel.

Supported Hardware:

All

Prototype:

```
int GscSio4ChannelSetClock(
                                    int boardNumber,
                                    int channel,
                                    int frequency);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

frequency – The desired bit rate for this channel. This value is specified in Hz and can range from 100 to 10000000 (1000000 for async channels).

GscSio4ChannelSetClockSource

GscSio4ChannelSetClockSource(…) is used to set the clock source for the Transmitter and Receiver of a channel.

Supported Hardware:

All

Prototype:

```
int GscSio4ChannelSetClock(
                                    int boardNumber,
                                    int channel,
                                    int txSource,
                                    int rxSource);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

txSource – The desired clock source for the Transmitter. The valid values are defined below.

rxSource – The desired clock source for the Receiver. The valid values are defined below.

GSC_CLOCK_INTERNAL – The clock is driven by the on board programmable oscillator. The bit rate is set by either the GscSio4ChannelSetMode() or the GscSio4ChannelSetClock() functions.

GSC_CLOCK_EXTERNAL – The clock is driven by the RxClk pin of the cable. The bit rated is determined by the external clock, but should also be set by either the GscSio4ChannelSetMode() or the GscSio4ChannelSetClock() functions to ensure optimal performance.

Protocol Level Routines

The Protocol Level Routines perform functions that apply to a specific protocol on a single channel on an SIO4 board. Each of these routines requires the board number (boardNumber) as the first parameter and the channel number (channel) as the second parameter. The board number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system. The channel number will always be 1, 2, 3, or 4.

These routines can be called at any time. All of these functions return zero if successful or a non-zero error code if a failure occurs.

GscSio4ChannelSetHdlcCrcMode / GscSio4ChannelGetHdlcCrcMode

GscSio4ChannelSetHdlcCrcMode(…) sets the mode for CRC generation/detection for a single channel of the SIO4 board. The CRC mode can be set to None, 16 bit, 32 bit, or CCITT. This routine is also used to set the initial value of the CRC register. This value can be set to either all 0 or all 1.

Supported Hardware:

PCI-SIO4B

Prototype:

```
int GscSio4ChannelSetHdlcCrcMode(
                                    int boardNumber,
                                    int channel,
                                    int crcMode,
                                    int initialValue);

int GscSio4ChannelGetHdlcCrcMode(
                                    int boardNumber,
                                    int channel,
                                    int *crcMode,
                                    int *initialValue);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

crcMode – The desired/current CRC type for this channel. The value should be one of the following:

GSC_CRC_NONE – Sets the channel to use no CRC.
GSC_CRC_16 – Sets the channel to use a 16 bit CRC.
GSC_CRC_32 – Sets the channel to use a 32 bit CRC.
GSC_CRC_CCITT - Sets the channel to use a CCITT CRC.

initialValue – The desired/current initial value for the CRC register. The value should be one of the following:

GSC_CRC_INIT_0 – Initializes the CRC register to all 0's.
GSC_CRC_INIT_1 – Initializes the CRC register to all 1's.

GscSio4ChannelSetBiSyncPattern / GscSio4ChannelGetBiSyncPattern

GscSio4ChannelSetBiSyncPattern(…) sets the two bytes that are used for the Sync Pattern when the channel is configured for BiSync mode.

Supported Hardware:

PCI-SIO4B

Prototype:

```
int GscSio4ChannelSetBiSyncPattern(
                                    int boardNumber,
                                    int channel,
                                    int txSync,
                                    int rxSync);

int GscSio4ChannelGetBiSyncPattern(
                                    int boardNumber,
                                    int channel,
                                    int *txSync,
                                    int *rxSync);
```

Parameters:

boardNumber – The number of the desired board. This number corresponds to the results of the GscSio4FindBoards(…) function. Note that this number will always be 1 in a single board system.

channel – The desired channel number. This number will be 1, 2, 3, or 4.

txSync – The desired/current value for the sync pattern for the transmitter.

rxSync – The desired/current value for the sync pattern for the receiver.

Structures and Macro Definitions

This section contains the descriptions of the various structures and macro definitions available to users of the API.

Devices Structure

Interrupt Callback Prototype

Channel Mode Definitions

The Channel Mode Definitions are used to set the current operating protocol for each channel of the SIO4 board. These definitions are passed as a parameter of the GscSio4ChannelSetMode(…) command.

Macro	Protocol	Defaults
GSC_MODE_ASYNC	Asynchronous Mode	8 Data Bits No Parity 1 Stop Bit 16x Clock NRZ Encoding
GSC_MODE_HDLC	HDLC/SDLC Mode	8 Data Bits NRZ Encoding
GSC_MODE_SYNC	Synchronous Mode*	8 Data Bits 0 Gap Bits NRZ Encoding
GSC_MODE_SYNC_ENV	Synchronous Mode w/ Envelope*	8 Data Bits 0 Gap Bits NRZ Encoding
GSC_MODE_ISO	Isochronous Mode	8 Data Bits NRZ Encoding
GSC_MODE_MONOSYNC	Monosync Mode	8 Data Bits NRZ Encoding
GSC_MODE_BISYNC	BiSync Mode	8 Data Bits NRZ Encoding
GSC_MODE_TRANS_BISYNC	Transparent BiSync Mode	8 Data Bits NRZ Encoding
GSC_MODE_802_3	IEEE 802.3 Ethernet Mode	8 Data Bits NRZ Encoding

* These are the only modes that are available on the –SYNC card. They are not available on the standard card.

Channel Encoding Definitions

The Channel Encoding Definitions are used to set the desired channel encoding for each channel of the SIO4 board. These definitions are passed as a parameter of the GscSio4ChannelSetEncoding(…) command.

Macro	**Description**
GSC_ENCODING_NRZ	
GSC_ENCODING_NRZB	
GSC_ENCODING_NRZI_MARK	
GSC_ENCODING_NRZI_SPACE	
GSC_ENCODING_BIPHASE_MARK	
GSC_ENCODING_BIPHASE_SPACE	
GSC_ENCODING_BIPHASE_LEVEL	
GSC_ENCODING_DIFF_BIPHASE_LEVEL	

Channel Protocol and Termination Definitions

GSC_PROTOCOL_RS422_RS485,
GSC_PROTOCOL_RS423,
GSC_PROTOCOL_RS232,
GSC_PROTOCOL_RS530_1,
GSC_PROTOCOL_RS530_2,
GSC_PROTOCOL_V35_1,
GSC_PROTOCOL_V35_2,
GSC_PROTOCOL_RS422_RS423_1,
GSC_PROTOCOL_RS422_RS423_2,

GSC_TERMINATION_ENABLED,
GSC_TERMINATION_DISABLED,

Channel Interrupt Definitions

```
GSC_INTR_RISING_EDGE,
GSC_INTR_FALLING_EDGE,
GSC_INTR_HIGH_TRUE,
GSC_INTR_LOW_TRUE,

GSC_INTR_SYNC_DETECT                = 0x0001,
GSC_INTR_USC                        = 0x0002,
GSC_INTR_TX_FIFO_EMPTY              = 0x0004,
GSC_INTR_TX_FIFO_FULL               = 0x0008,
GSC_INTR_TX_FIFO_ALMOST_EMPTY       = 0x0010,
GSC_INTR_RX_FIFO_EMPTY              = 0x0020,
GSC_INTR_RX_FIFO_FULL               = 0x0040,
GSC_INTR_RX_FIFO_ALMOST_FULL        = 0x0080,
GSC_INTR_TX_TRANSFER_COMPLETE       = 0x0100,
GSC_INTR_RX_TRANSFER_COMPLETE       = 0x0200,
GSC_INTR_RX_ENVELOPE                = GSC_INTR_SYNC_DETECT,
// -Sync card definition
```

Channel Pin Definitions

```
GSC_PIN_DTE,
GSC_PIN_DCE,
GSC_PIN_AUTO,
GSC_PIN_GPIO,
GSC_PIN_RX_CLOCK,       // Keep these enums in order
GSC_PIN_RX_DATA,        //
GSC_PIN_CTS,            //
GSC_PIN_DCD,            //
GSC_PIN_TX_CLOCK,       //
GSC_PIN_TX_DATA,        //
GSC_PIN_RTS,            //
GSC_PIN_AUXCLK,         // Keep these enums in order
GSC_PIN_RX_ENV,
GSC_PIN_TX_ENV,
```

Channel Parity Definitions

GSC_PARITY_NONE,
GSC_PARITY_EVEN,
GSC_PARITY_ODD,
GSC_PARITY_MARK,
GSC_PARITY_SPACE,

Channel Stop Bits Definition

GSC_STOP_BITS_0,
GSC_STOP_BITS_1,
GSC_STOP_BITS_1_5,
GSC_STOP_BITS_2,

Loopback Definitions

GSC_LOOP_NONE,
GSC_LOOP_INTERNAL,
GSC_LOOP_EXTERNAL,

HDLC CRC Defintions

GSC_CRC_NONE,
GSC_CRC_16,
GSC_CRC_32,
GSC_CRC_CCITT,
GSC_CRC_INIT_0,
GSC_CRC_INIT_1,

Local Register Definitions

The Local Register Definitions are used to access the various registers that are contained in the on board FPGA. These registers should not be accessed during normal operation and are included only for diagnostic purposes. For detailed descriptions of the registers, refer to the SIO4 hardware user's manual.

Macro	Value	Description
FW_REVISION_REG	0x0000	Firmware Revision Register
BOARD_CONTROL_REG	0x0004	Board Control Register
BOARD_STATUS_REG	0x0008	Board Status Register
CLOCK_CONTROL_REG	0x000c	Clock Control Register
TX_ALMOST_BASE_REG	0x0010	Base value for the Tx Almost registers
RX_ALMOST_BASE_REG	0x0014	Base value for the Rx Almost registers
DATA_FIFO_BASE_REG	0x0018	Base value for the Tx and Rx Data FIFOs
CONTROL_STATUS_BASE_REG	0x001c	Base value for the Control/Status registers
SYNC_CHARACTER_BASE_REG	0x0050	Base value for the Sync Byte Registers
INTERRUPT_CONTROL_REG	0x0060	Interrupt Control Register
INTERRUPT_STATUS_REG	0x0064	Interrupt Status/Clear Register
INTERRUPT_EDGE_LEVEL_REG	0x0068	Interrupt Edge/Level Register (RO)
INTERRUPT_HI_LO_REG	0x006c	Interrupt High/Low, Rising/Falling register
PIN_SOURCE_BASE_REG	0x0080	Base value for the Pin Source Registers
PIN_STATUS_BASE_REG	0x0090	Base value for the Pin Status Registers
POSC_RAM_ADDRESS_REG	0x00a0	Programmable OSC Address Register
POSC_RAM_DATA_REG	0x00a4	Programmable OSC Data Register
POSC_CONTROL_STATUS_REG	0x00a8	Programmable OSC Control/Status Register
TX_COUNT_BASE_REG	0x00b0	
FIFO_COUNT_BASE_REG	0x00d0	Base value for the FIFO Count Registers
FIFO_SIZE_BASE_REG	0x00e0	Base value for the FIFO Size Registers
FEATURES_REG	0x00fc	Features Register

Channel Register Definitions

The Channel Register Definitions are used to access the various registers that are contained in the Zilog USC chip for each channel. These registers should not be accessed during normal operation and are included only for diagnostic purposes. For detailed descriptions of the registers, refer to the Zilog USC hardware user's manual.

Macro	Value	Description
USC_CCAR	0x0000	Channel Command/Address Register
USC_CMR	0x0002	Channel Mode Register
USC_CCSR	0x0004	Channel Command/Status Register
USC_CCR	0x0006	Channel Control Register
USC_TMDR	0x000c	Test Mode Data Register
USC_TMCR	0x000e	Test Mode Control Register
USC_CMCR	0x0010	Clock Mode Control Register
USC_HCR	0x0012	Hardware Configuration Register
USC_IVR	0x0014	Interrupt Vector Register
USC_IOCR	0x0016	I/O Control Register
USC_ICR	0x0018	Interrupt Control Register
USC_DCCR	0x001a	Daisy Chain Control Register
USC_MISR	0x001c	Misc. Interrupt Status Register
USC_SICR	0x001e	Status Interrupt Control Register
USC_RDR	0x0020	Receive Data Register (RO)
USC_RMR	0x0022	Receive Mode Register
USC_RCSR	0x0024	Receive Command Status Register
USC_RICR	0x0026	Receive Interrupt Control Register
USC_RSR	0x0028	Receive Sync Register
USC_RCLR	0x002a	Receive Count Limit Register
USC_RCCR	0x002c	Receive Character Count Register
USC_TC0R	0x002e	Time Constant 0 Register
USC_TDR	0x0030	Transmit Data Register (WO)
USC_TMR	0x0032	Transmit Mode Register
USC_TCSR	0x0034	Transmit Command Status Register
USC_TICR	0x0036	Transmit Interrupt Control Register
USC_TSR	0x0038	Transmit Sync Register
USC_TCLR	0x003a	Transmit Count Limit Register
USC_TCCR	0x003c	Transmit Character Count Register
USC_TC1R	0x003e	Time Constant 1 Register

Miscellaneous Token Definitions

GSC_ENABLED,
GSC_DISABLED,

GSC_CLOCK_INTERNAL,
GSC_CLOCK_EXTERNAL,

GSC_LSB_FIRST,
GSC_MSB_FIRST,