

SIOB4/8-SYNC

Four Channel High Speed Serial I/O

**PCI-..., PMC-..., CPCI-..., PC104P-...
...-SIO4B/BX/BXR-SYNC
...-SIO8BX/BXS-SYNC**

Linux Device Driver User Manual

**Manual Revision: January 16, 2012
Driver Release Version 1.42.0**

**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

Copyright © 2004-2012, **General Standards Corporation**

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: <mailto: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.

ALL RIGHTS RESERVED.

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.

The information in this document is subject to change without notice. This document may be copied or reproduced provided it is in support of products from **General Standards Corporation**. For any other use, no part of this document may be copied or reproduced in any form or by any means without prior written consent of **General Standards Corporation**.

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

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

Table of Contents

1. Introduction.....	7
1.1. Purpose	7
1.2. Acronyms	7
1.3. Definitions.....	7
1.4. Software Overview.....	7
1.5. Hardware Overview.....	7
1.6. Reference Material.....	8
2. Installation	9
2.1. CPU and Kernel Support	9
2.2. Compiler Support	9
2.3. The /proc File System	9
2.4. File List	10
2.5. Directory Structure.....	10
2.6. Installation.....	11
2.7. Removal	11
2.8. Overall Make Script	11
3. The Driver.....	13
3.1. Build	13
3.2. Startup	13
3.2.1. Manual Driver Startup Procedures	14
3.2.2. Automatic Driver Startup Procedures.....	14
3.3. Verification	15
3.4. Version	15
3.5. Shutdown	15
4. The SYNC Library	16
4.1. Build	16
4.2. Use	16
5. Document Source Code Examples.....	17
5.1. SIO4DocSrcLib.a	17
5.1.1. Build	17
5.1.2. Library Use.....	17
5.2. sio4_sync_doc_src_lib.a.....	18
5.2.1. Build	18
5.2.2. Library Use.....	18
6. Sample Applications	19

6.1. id - Identify Board.....	19
6.1.1. Build	19
6.1.2. Execute	19
6.2. regs - Register Access.....	20
6.2.1. Build	20
6.2.2. Execute	20
6.3. sbtest - Single Board Test	21
6.3.1. Build	21
6.3.2. Execute	21
6.4. irq - Interrupt Tests - NOT IN THIS RELEASE.....	22
6.4.1. Build	22
6.4.2. Execute	22
6.5. synctest- SYNC Test	23
6.5.1. Build	23
6.5.2. Execute	23
6.6. Syncc2c- SYNC Channel-to-Channel.....	24
6.6.1. Build	24
6.6.2. Execute	24
6.7. txrate - Transmit Rate.....	25
6.7.1. Build	25
6.7.2. Execute	25
7. Driver Interface.....	26
7.1. Macros.....	26
7.1.1. IOCTL	26
7.1.2. Registers	26
7.1.3. SIO4_SYNC_LIB_VER.....	27
7.2. Data Types.....	27
7.2.1. FIFO_STATUS	27
7.2.2. sio4_driver_info_t	28
7.2.3. SIO4_INTERRUPT_STATUS.....	28
7.2.4. sio4_mp_chip_t	28
7.2.5. sio4_mp_prot_t.....	29
7.2.6. sio4_mp_t.....	29
7.2.7. sio4_osc_chip_t.....	30
7.2.8. sio4_osc_t.....	30
7.2.9. sio4_reg_t.....	30
7.2.10. sio4_sync_rx_t.....	31
7.2.11. sio4_sync_t.....	32
7.2.12. sio4_sync_tx_t.....	33
7.2.13. TX_RX	34
7.3. Functions.....	35
7.3.1. close()	35
7.3.2. ioctl()	35
7.3.3. open().....	36
7.3.4. read()	37
7.3.5. sio4_sync_get().....	38
7.3.6. sio4_sync_gpio_rx()	39
7.3.7. sio4_sync_gpio_tx()	40
7.3.8. sio4_sync_rx_get()	41
7.3.9. sio4_sync_rx_set().....	41

7.3.10. sio4_sync_set()	42
7.3.11. sio4_sync_tx_get()	43
7.3.12. sio4_sync_tx_set()	44
7.3.13. sio4_sync_version()	45
7.3.14. write()	45
7.4. IOCTL Services	46
7.4.1. SIO4_BOARD_JUMPERS	47
7.4.2. SIO4_FEATURE_TEST	47
7.4.3. SIO4_GET_DRIVER_INFO	48
7.4.4. SIO4_INIT_BOARD	48
7.4.5. SIO4_INT_NOTIFY	48
7.4.6. SIO4_MOD_REGISTER	49
7.4.7. SIO4_MP_CONFIG	49
7.4.8. SIO4_MP_INFO	49
7.4.9. SIO4_MP_INIT	50
7.4.10. SIO4_MP_RESET	50
7.4.11. SIO4_MP_TEST	50
7.4.12. SIO4_OSC_INFO	50
7.4.13. SIO4_OSC_INIT	51
7.4.14. SIO4_OSC_MEASURE	51
7.4.15. SIO4_OSC_PROGRAM	51
7.4.16. SIO4_OSC_REFERENCE	52
7.4.17. SIO4_OSC_RESET	52
7.4.18. SIO4_OSC_TEST	52
7.4.19. SIO4_READ_INT_STATUS	52
7.4.20. SIO4_READ_REGISTER	53
7.4.21. SIO4_READ_REGISTER_RAW	53
7.4.22. SIO4_RESET_CHANNEL	53
7.4.23. SIO4_RESET_DEVICE	53
7.4.24. SIO4_RESET_FIFO	54
7.4.25. SIO4_RX_FIFO_AE_CONFIG	54
7.4.26. SIO4_RX_FIFO_AF_CONFIG	54
7.4.27. SIO4_RX_FIFO_COUNT	54
7.4.28. SIO4_RX_FIFO_SIZE	55
7.4.29. SIO4_RX_IO_ABORT	55
7.4.30. SIO4_RX_IO_MODE_CONFIG	55
7.4.31. SIO4_SET_READ_TIMEOUT	56
7.4.32. SIO4_SET_WRITE_TIMEOUT	56
7.4.33. SIO4_TX_FIFO_AE_CONFIG	56
7.4.34. SIO4_TX_FIFO_AF_CONFIG	56
7.4.35. SIO4_TX_FIFO_COUNT	57
7.4.36. SIO4_TX_FIFO_SIZE	57
7.4.37. SIO4_TX_IO_ABORT	57
7.4.38. SIO4_TX_IO_MODE_CONFIG	57
7.4.39. SIO4_WRITE_REGISTER	58
8. Operation	59
8.1. I/O Modes	59
8.1.1. DMDMA	59
8.1.2. DMA	59
8.1.3. PIO	59
8.2. Oscillator Programming	59
8.2.1. Cypress CY22393 (1x) Programmable Oscillator Support	60
8.2.2. Cypress CY22393 (4x) Programmable Oscillator Support	61

8.2.3. Cypress IDC2053B Programmable Oscillator Support	61
8.2.4. Fixed Oscillator Support.....	61
8.2.5. All Other Cases.....	61
8.3. Multi-Protocol Transceiver Programming	61
8.3.1. CipeX SP508 Multi-Protocol Transceiver Support	62
8.3.2. Fixed Protocol Support.....	62
8.3.3. All Other Cases.....	63
8.4. Interrupt Notification	63
Document History	65

1. Introduction

This user manual applies to the SYNC specific release of driver release version 1.42.0.

NOTE: The device models listed on the front cover are those that are specifically supported by this release of the driver. Other models may be supported, though the level of support may vary. The driver may work with other SIO4 models, but performance may be degraded due to device feature and implementation differences.

1.1. Purpose

The purpose of this document is to describe the interface to the SIO4 Linux Device Driver and SYNC library, as it relates to SYNC versions of the SIO4. This software provides the interface between “Application Software” and the SIO4 board. The interface to this board is at the device level.

1.2. Acronyms

The following is a list of commonly occurring acronyms used throughout this document.

Acronyms	Description
DMA	Direct Memory Access
DMDMA	Demand Mode DMA
GSC	General Standards Corporation
LSBF	Least Significant Bit First
MSBF	Most Significant Bit First
PCI	Peripheral Component Interconnect
PMC	PCI Mezzanine Card

1.3. Definitions

The following is a list of commonly occurring terms used throughout this document.

Term	Definition
Application	Application means the user mode process, which runs in the user space with user mode privileges.
Driver	Driver means the kernel mode device driver, which runs in the kernel space with kernel mode privileges.
SIO4	This is used as a general reference to any SYNC version board supported by this driver.

1.4. Software Overview

This release of the SIO4 driver includes an executable device driver and a library of SYNC specific services. The driver software executes under control of the Linux operating system and runs in Kernel Mode as a Kernel Mode device driver. The SIO4 device driver is implemented as a standard loadable Linux device driver written in the ‘C’ programming language. With the driver, user applications are able to open and close a channel and, while open, perform read, write and I/O control operations. SIO4-SYNC applications must link the SYNC library with their application in order to utilize the additional, library based services.

1.5. Hardware Overview

The SIO4 is a four channel high-speed serial interface I/O board. This board provides for bi-directional serial data transfers between two computers, or one computer and an external peripheral.

This board also can transfer data indefinitely without host intervention. Once the data link between the two computers is established, the desired transfers can be performed and will become transparent to the user.

The SIO4 board includes a DMA controller and comes with a maximum of 256K Bytes of FIFO storage, which is 32K per channel side ($32K * 2 * 4$). The FIFO configuration can vary greatly from one SIO4 version to another (i.e. $32K * 2 * 4$ to $1K * 2 * 1$ to none at all). The SIO4 includes configurable transceiver support that includes RS-232 and RS-485/422 transceiver options. The DMA controller is capable of transferring data to and from host memory; whereas the FIFO memory provides a means for continuous transfer of data without interrupting the DMA transfers or requiring intervention from the host CPU. The board also provides for interrupt generation for various states of the board like FIFO empty, FIFO full and DMA complete.

1.6. Reference Material

The following reference material may be of particular benefit in using the SIO4 and this driver. The specifications provide the information necessary for an in depth understanding of the specialized features implemented on this board.

- The applicable *SIO4/SIO8 User Manual* from General Standards Corporation.
- The *PCI Bus Master Interface Chip* data handbook for the PCI9056/9080 from PLX Technology, Inc.

PLX Technology Inc.
870 Maude Avenue
Sunnyvale, California 94085 USA
Phone: 1-800-759-3735
WEB: <http://www.plxtech.com/>

2. Installation

2.1. CPU and Kernel Support

The driver is designed to operate with Linux kernel versions 2.6, 2.4 and 2.2 running on a PC system with one or more x86 processors. This release of the driver was tested under the below listed kernels.

Kernel	Distribution	X86	
		32-bit	64-bit
2.6.38	Red Hat Fedora Core 15	Yes	Yes
2.6.35	Red Hat Fedora Core 14	Yes	Yes
2.6.33	Red Hat Fedora Core 13	Yes	Yes
2.6.31	Red Hat Fedora Core 12	Yes	Yes
2.6.29	Red Hat Fedora Core 11	Yes	Yes
2.6.27	Red Hat Fedora Core 10	Yes	Yes
2.6.25	Red Hat Fedora Core 9	Yes	Yes
2.6.23	Red Hat Fedora Core 8	Yes	Yes
2.6.21	Red Hat Fedora Core 7	Yes	Yes
2.6.18	Red Hat Fedora Core 6	Yes	Yes
2.6.15	Red Hat Fedora Core 5	Yes	Yes
2.6.11	Red Hat Fedora Core 4	Yes	Yes
2.6.9	Red Hat Fedora Core 3	Yes	Yes
2.4.21	Red Hat Enterprise Linux Workstation Release 3	Yes	
2.4.7	Red Hat Linux 7.2	Yes	
2.2.14	Red Hat Linux 6.2	Yes	

NOTE: The driver will have to be rebuilt before being used as the driver is provided in source form only.

NOTE: The driver has not been tested with a non-versioned kernel.

NOTE: The driver has not been tested on an SMP host.

2.2. Compiler Support

The 32-bit build for this driver relies on the use of the GCC compiler. This dependence is due only to the driver's use of the file `divdi3.c`, which is copied from GCC 2.95.1. The driver build process has been verified according to the above CPU and kernel support paragraph. The build process may fail under other build environments.

NOTE: The dependence on the GCC compiler is due to the driver's use of 64-bit integer division. This division is performed during configuration of the programmable oscillator present on some versions of the SIO4. Under the 2.2 and 2.4 kernels the needed library services are linked implicitly during the build process. Under the 2.6 kernel build process, the needed services must be linked explicitly.

2.3. The /proc File System

NOTE: All SIO8 model boards appear as two SIO4 model boards.

While the driver is installed, the `/proc/sio4` file can be read to obtain information about the driver. Each file entry includes an entry name followed immediately by a colon, a space character, and then the entry value. Below is an example of what appears in the file, followed by descriptions of each entry.

```

version: 1.42
built: Jan 16 2012, 13:09:42
boards: 1
models: SIO4BX-SYNC
ids: 0x3

```

Entry	Description
version	This gives the driver version number in the form x.xx.
built	This gives the driver build date and time as a string. It is given in the C form of <code>printf("%s", __DATE__, __TIME__)</code> .
boards	This identifies the total number of SIO4 boards the driver detected.
models	This is a list that identifies the basic model numbers of the board detected by the driver. The order in the list corresponds to the device node indexes in the <code>/dev</code> directory. If the driver cannot specifically identify the board's type it will be listed only as "SIO4".
ids	This is a list identifying the values read from the boards' user jumpers.

2.4. File List

This release consists of the below listed files. The archive is described in detail in following subsections.

File	Description
sio4.sync.tar.gz	This archive contains the entire driver with all associated files, including sources and make files.
sio4_sync_linux_um.pdf	This is a PDF version of this user manual.

NOTE: Driver versions 1.05 and earlier were shipped as multiple archive files that were decompressed separately. As of version 1.06 all driver files (except the user manual) are supplied as a single archive. All previous files, archives and directories should be removed before proceeding with installation of this driver.

2.5. Directory Structure

The following table describes the directory structure observed by the source archive.

NOTE: As of release 1.19.2 the directory structure changed to accommodate the asynchronous library code and any associated files and build targets. Only the library sources are included in the SYNC version of the release.

NOTE: As of version 1.06, the driver and all associated support files are installed under a single directory. All previous releases of the driver utilized a different directory structure based on where the user installed the separate archives. All previous files, archives and directories should be removed before proceeding with installation of this driver.

Directory	Content
sio4	This is the driver's root directory.
sio4/async/lib	This directory contains the Asynchronous library sources.
sio4/docsrc	This directory contains the C sources from this user manual.
sio4/driver	This directory contains the driver executable and its sources.
sio4/id	This directory contains the id sample application.
sio4/regs	This directory contains the regs sample application.
sio4/sbtest	This directory contains the sbtest sample application.
sio4/sync	This directory contains release components specific to SYNC versions of the SIO4.
sio4/sync/docsrc	This directory contains the SYNC specific C sources from this user manual.
sio4/sync/lib	This directory contains the SYNC library and its sources.

<code>sio4/sync/syncc2c</code>	This directory contains the SYNC specific <code>syncc2c</code> sample application.
<code>sio4/sync/synctest</code>	This directory contains the SYNC specific <code>synctest</code> sample application.
<code>sio4/sync/txrate</code>	This directory contains the SYNC specific <code>txrate</code> sample application.
<code>sio4/sync/utils</code>	This directory contains utility sources used by the sample SYNC applications.
<code>sio4/utils</code>	This directory contains utility sources used by the sample applications.

2.6. Installation

Install the driver and its related files following the below listed steps.

1. Change the current directory to `/usr/src/linux/drivers`.

NOTE: Depending on the version of the kernel being used and the distribution, the `.../linux/...` portion of the directory may exist as a soft link or it may not. If it doesn't exist, then it should be present with a suffix including the base version number of the kernel. The directory should then appear as something like `.../linux-2.4/...`, or something similar. Perform the substitution as appropriate or create a `.../linux` soft link.

2. Copy the archive file `sio4.sync.tar.gz` into the current directory.
3. Issue the following command to decompress and extract the files from the provided archive. This creates the directory structure described earlier and copies all of the archive files into the created directories.

```
tar -xzf sio4.sync.tar.gz
```

2.7. Removal

Follow the below steps to remove the driver and all associated files.

1. Shutdown the driver as described in previous paragraphs.
2. Change to the directory where the driver archive was installed. This should be `/usr/src/linux/drivers/`.
3. Issue the below command to remove the driver archive and all of the installed driver files.

```
rm -rf sio4.sync.tar.gz sio4
```

4. Issue the below command to remove all of the installed device nodes.

```
rm -f /dev/sio4*
```

5. If the automated startup procedure was adopted, then edit the system startup script `rc.local` and remove the line that invokes the `start` script. The file `rc.local` should be located in the `/etc/rc.d` directory.

2.8. Overall Make Script

A make script is included in the root installation directory. Executing this script will perform a make for all build targets included in the release. The script is named `make_all`. Follow the below steps to perform an overall make and to load the driver.

1. Change to the device's root directory, which may be `/usr/src/linux/drivers/sio4/`.
2. Issue the below command to make all archive targets and load the driver.

```
./make_all
```

3. The Driver

The paragraphs that follow give instructions on building, starting and verifying startup of the driver. These files are installed into the `/usr/src/linux/drivers/sio4/driver/` directory.

NOTE: This driver works with both SYNC and non-SYNC versions of the SIO4. The driver used here is the same exact driver provided with the non-SYNC driver release.

File	Description
*.c	These sources implement the driver interface and its functionality. Some functionality has been modularized based on individual source file base names.
*.h	These are driver header files. Others are listed below.
Makefile	This is the driver make file.
makefile.dep	This is a make dependency file. This is updated automatically.
sio4.h	This is the driver interface header file. It should be included by SIO4 applications.
start	This is a shell script to install the driver module and device nodes.

3.1. Build

Follow the below steps to build the driver.

1. Change to the directory where the driver and its sources were installed. This should be `/usr/src/linux/drivers/sio4/driver`.
2. Remove all existing build targets by issuing the below command.

```
make clean
```

3. Build the driver by issuing the below command.

```
make all
```

NOTE: Building the SIO4 driver requires installation of the kernel header sources. If they are not present the build will fail.

NOTE: Due to the differences between the many Linux distributions some build errors may occur. These errors may include system header location differences and should be easily correctable. Other errors may also appear as some distributors port newer kernel changes into older kernel distributions.

NOTE: Debug messages already present in the driver sources are enabled only when the macro `SIO4_DEBUG` is enabled. This macro is undefined by default and can be found in `sio4.h`.

3.2. Startup

NOTE: The driver may have to be rebuilt before being used due to kernel version differences between the GSC build host and the customer target host.

The startup script used in this procedure is designed to insure that the driver module in the install directory is the module that is loaded. This is accomplished by making sure that an already loaded module is first unloaded before attempting to load the module from the disk drive. In addition, the script also deletes and recreates the device nodes. This is done to insure that the device nodes in use have the same major number as assigned dynamically to the driver by the kernel, and so that the number of device nodes correspond to the number of boards identified by the driver.

3.2.1. Manual Driver Startup Procedures

Start the driver manually by following the below listed steps.

1. Login as root user, as some of the steps require root privileges.
2. Change to the directory where the driver was installed. This should be `/usr/src/linux/drivers/sio4/driver/`.
3. Install the driver module and create the device nodes by executing the below command. If any errors are encountered then an appropriate error message will be displayed.

```
./start
```

NOTE: While loading the amount of time taken for driver initialization will vary depending on the number of boards and each board's type. For those boards with programmable oscillators, additional initialization time may be needed for programming of each channel.

NOTE: The script's default specifies that the driver is installed in the same directory as the script. The script will fail if this is not so.

NOTE: The above step must be repeated each time the host is rebooted.

NOTE: The SIO4 device node major number is assigned dynamically by the kernel. The minor numbers and the device node suffix numbers are index numbers beginning with one, and increase by one for each additional serial channel.

4. Verify that the device module has been loaded by issuing the below command and examining the output. The module name `sio4` should be included in the output.

```
lsmod
```

5. Verify that the device nodes have been created by issuing the below command and examining the output. The output should include four nodes for each installed board.

```
ls -l /dev/sio4*
```

3.2.2. Automatic Driver Startup Procedures

Start the driver automatically with each system reboot by following the below listed steps.

1. Locate and edit the system startup script `rc.local`, which should be in the `/etc/rc.d` directory. Modify the file by adding the below line so that it is executed with every reboot.

```
/usr/src/linux/drivers/sio4/driver/start
```

NOTE: The script's default specifies that the driver is installed in the same directory as the script. The startup script will fail if this is not so.

2. Load the driver and create the required device nodes by rebooting the system.
3. Verify that the driver is loaded and that the device nodes have been created by following the verification steps given in the manual startup procedures.

3.3. Verification

Follow the below steps to verify that the driver has been properly installed and started.

1. Issue the below command to view the content of the driver's `/proc` file system text file.

```
cat /proc/sio4
```

2. If the file exists then the driver is installed and running.

3.4. Version

The driver version number can be obtained in a variety of ways. It is reported by the driver both when the driver is loaded and when it is unloaded (depending on kernel configuration options, this may be visible only in `/var/log/messages`). It is recorded in the file `/proc/sio4`. It can also be read by an application via the `SIO4_GET_DRIVER_INFO` IOCTL services.

3.5. Shutdown

Shutdown the driver following the below listed steps.

1. Login as root user, as some of the steps require root privileges.
 2. If the driver is currently loaded then issue the below command to unload the driver.
- ```
rmmod sio4
```
3. Verify that the driver module has been unloaded by issuing the below command. The module name `sio4` should not be in the list.

```
lsmod
```

## 4. The SYNC Library

This library provides applications access to services and functionality specific to SIO4-SYNC boards. The driver archive includes all of the library's sources and make files. The library must be linked with SYNC applications in order to exercise the library's SYNC specific functionality. The source files are delivered undocumented but may be used to assist in application development and to help ease the learning curve. The files are described here only briefly, though library use is described in the following paragraph. The files are installed into the `/usr/src/linux/drivers/sio4/sync/lib/` directory. The files included are listed below.

| File                            | Description                                                                                                          |
|---------------------------------|----------------------------------------------------------------------------------------------------------------------|
| <code>*.c</code>                | These are the sources that implement the library's functionality.                                                    |
| <code>sio4_sync_driver.h</code> | This is a header file used by the library.                                                                           |
| <code>makefile</code>           | This is the library make file.                                                                                       |
| <code>makefile.dep</code>       | This is a make dependency file. This is updated automatically.                                                       |
| <code>sio4_sync_lib.h</code>    | This is a header file that exports the library's interface. This must be included by applications using the library. |

### 4.1. Build

Follow the below steps to compile the example files.

1. Change to the directory where the library source files were installed. This should be `/usr/src/linux/drivers/sio4/sync/lib/`.

2. Remove all existing build targets by issuing the below command.

```
make clean
```

3. Build the library issuing the below command.

```
make all
```

### 4.2. Use

Use of the library has compile and link time requirements. Compile time use requires inclusion of the library's header file and expansion of the include search path. The header file is named `sio4_sync_lib.h` and the path to include is `/usr/src/linux/drivers/sio4/sync/lib/`. Link time use requires linking two additional libraries and expansion of the library search path. The first library is the SYNC library, which is named `sio4_sync_lib.a` and is located in `/usr/src/linux/drivers/sio4/sync/docsrc/`. The second library is the primary document source code example library, which is named `SIO4DocSrcLib.a` and is located in `/usr/src/linux/drivers/sio4/docsrc/`. This library is described below.



## 5. Document Source Code Examples

The driver archive contains all of the source code examples included in this document. They are provided in library form appropriate for linking with SIO4 command line applications. The files are delivered undocumented and unsupported but may be used to assist in application development and to help ease the learning curve. The files are described here only briefly, though library use is described in the following paragraph. The purpose of the files is to provide the user with the actual source code given in the manual, to provide example code that can be used in customer applications, and to provide a means of insuring that the examples cited will compile. The sources are provided in the form of two libraries. The primary library contains the bulk of the code samples. Some of these sources are applicable to both SYNC and non-SYNC versions of the SIO4, while some are applicable only to non-SYNC versions. Only those sources cited in this user manual are applicable to SIO4-SYNC boards. (The others are included in the release because the same library is included with all driver releases.) Unless otherwise noted, code samples are a part of this library. The secondary library is SYNC specific and comprises only a small number of code samples. The corresponding code samples are specifically identified as belonging to this library.

### 5.1. SIO4DocSrcLib.a

The library comprises the majority of the code samples in this document along with some that do not apply to SYNC boards.

**NOTE:** This source code library is common across all SIO4 models supported by the driver and may include some code samples intended for SIO4 models not listed on the cover page of this user manual. The sample code may work with other unlisted models, but may not function as expected since they may not necessarily be intended for these models. Library code not referenced in this user manual should generally not be used with those models listed on the cover page. For other SIO4 models, refer to the applicable driver user manual.

| File            | Description                                                                |
|-----------------|----------------------------------------------------------------------------|
| *.c             | These are the sources which correspond to the samples in this user manual. |
| makefile        | This is the library make file.                                             |
| makefile.dep    | This is a make dependency file. This is updated automatically.             |
| SIO4DocSrcLib.h | This is a header file that gives the prototypes for all library functions. |

#### 5.1.1. Build

Follow the below steps to compile the example files.

1. Change to the directory where the source code example files were installed. This should be `/usr/src/linux/drivers/sio4/docsrc/`.
2. Remove all existing build targets by issuing the below command.  
  
`make clean`
3. Compile the sample files by issuing the below command.  
  
`make all`

#### 5.1.2. Library Use

Use of the library has requirements applicable at the time the application is being built. C sources using the library must include the header file, which is named `SIO4DocSrcLib.h`. Applications using the library must link the file `SIO4DocSrcLib.a`. Both the include search path and the library search path must be expanded to include the directory `/usr/src/linux/drivers/sio4/docsrc/`.

## 5.2. sio4\_sync\_doc\_src\_lib.a

This library comprises the SYNC specific code samples in this document.

| File                    | Description                                                                |
|-------------------------|----------------------------------------------------------------------------|
| *.c                     | These are the sources which correspond to the samples in this user manual. |
| makefile                | This is the library make file.                                             |
| makefile.dep            | This is a make dependency file. This is updated automatically.             |
| sio4_sync_doc_src_lib.h | This is a header file that gives the prototypes for all library functions. |

### 5.2.1. Build

Follow the below steps to compile the example files.

1. Change to the directory where the source code example files were installed. This should be `/usr/src/linux/drivers/sio4/sync/docsrc/`.
2. Remove all existing build targets by issuing the below command.

```
make clean
```

3. Compile the sample files by issuing the below command.

```
make all
```

### 5.2.2. Library Use

Use of the library has requirements applicable at the time the application is being built. C sources using the library must include the header file, which is named `sio4_sync_doc_src_lib.h`. Applications using the library must link the file `sio4_sync_doc_src_lib.a`. Both the include search path and the library search path must be expanded to include the directory `/usr/src/linux/drivers/sio4/sync/docsrc/`.

## 6. Sample Applications

**NOTE:** The sample applications are unsupported and are provided without documentation.

**NOTE:** These sample applications are designed to function with the SIO4 models listed on the cover of this user manual. The sample applications may work with other models, but may not function as expected since they are not necessarily intended for those models. Refer to the driver user manual and sample applications supplied with the SIO4 model in question, as applicable.

### 6.1. id - Identify Board

This sample console application provides a command line driven Linux application that provides detailed board identification information. This can be used with tech support to help identify as much technical information about the board as possible from software. The application's sources are summarized in the below table.

| File            | Description                                              |
|-----------------|----------------------------------------------------------|
| id/*.c          | These are the application's source files.                |
| id/main.h       | This is the application's header file.                   |
| id/makefile     | This is the application make file.                       |
| id/makefile.dep | This is an automatically generated make dependency file. |
| docsrc/*        | These are utility sources used by the application.       |
| utils/*         | These are utility sources used by the application.       |

#### 6.1.1. Build

Follow the below steps to build the sample application.

1. Change to the directory where the sample application sources are installed (.../id).
2. Remove all existing build targets by issuing the below command.

```
make clean
```

3. Build the application by issuing the below command.

```
make all
```

#### 6.1.2. Execute

Follow the below steps to execute the sample application.

1. Change to the directory where the sample application sources are installed (.../id).
2. Start the sample application by issuing the command given below. Once started the application will automatically output identification information. A single iteration may take a few second to complete. The command line arguments are described in the table below.

```
./id <index>
```

| Argument | Description                                            |
|----------|--------------------------------------------------------|
| index    | This is the zero based index of the channel to access. |

## 6.2. regs - Register Access

This sample console application provides a menu based command line Linux application that permits interactive access to the board's registers, including write access to the GSC specific registers. The application's sources are summarized in the below table.

| File              | Description                                              |
|-------------------|----------------------------------------------------------|
| regs/*.c          | These are the application's source files.                |
| regs/main.h       | This is the application's header file.                   |
| regs/makefile     | This is the application make file.                       |
| regs/makefile.dep | This is an automatically generated make dependency file. |
| docsrc/*          | These are utility sources used by the application.       |
| utils/*           | These are utility sources used by the application.       |

### 6.2.1. Build

Follow the below steps to build the sample application.

1. Change to the directory where the sample application sources are installed (.../regs).
2. Remove all existing build targets by issuing the below command.

```
make clean
```

3. Build the application by issuing the below command.

```
make all
```

### 6.2.2. Execute

Follow the below steps to execute the sample application.

1. Change to the directory where the sample application sources are installed (.../regs).
2. Start the sample application by issuing the command given below. The command line argument is described in the table below.

```
./regs <index>
```

| Argument | Description                                          |
|----------|------------------------------------------------------|
| index    | This is the zero based index of the board to access. |

3. Select the desired options according to the menus presented. Select the main menu exit option when finished.

### 6.3. sbtest - Single Board Test

This sample console application provides a command line driven Linux application that tests the functionality of the driver and a specified board. The application's sources are summarized in the below table.

| File                | Description                                              |
|---------------------|----------------------------------------------------------|
| sbtest/*.c          | These are the application's source files.                |
| sbtest/main.h       | This is the application's header file.                   |
| sbtest/makefile     | This is the application make file.                       |
| sbtest/makefile.dep | This is an automatically generated make dependency file. |
| docsrc/*            | These are utility sources used by the application.       |
| utils/*             | These are utility sources used by the application.       |

#### 6.3.1. Build

Follow the below steps to build the sample application.

1. Change to the directory where the sample application sources are installed (.../sbtest).

2. Remove all existing build targets by issuing the below command.

```
make clean
```

3. Build the application by issuing the below command.

```
make all
```

#### 6.3.2. Execute

**NOTE:** This application should be run with no cable attached.

Follow the below steps to execute the sample application.

1. Change to the directory where the sample application sources are installed (.../sbtest).
2. Start the sample application by issuing the command given below. Once started the application will automatically performs a series of test operations. A single iteration should complete in less than one minute. The command line arguments are described in the table below.

```
./sbtest <-c> <-C> <-m#> <-n#> <index>
```

| Argument | Description                                                                             |
|----------|-----------------------------------------------------------------------------------------|
| -c       | Repeat the operation until an error is encountered.                                     |
| -C       | Repeat the operation, but continue even if errors are encountered.                      |
| -m#      | When repeating the operation, stop after “#” minutes, where “#” is a decimal number.    |
| -n#      | When repeating the operation, stop after “#” iterations, where “#” is a decimal number. |
| index    | This is the zero based index of the board to access.                                    |

## 6.4. irq - Interrupt Tests - NOT IN THIS RELEASE

This sample application performs an automated test of the board's interrupts, including those of the USC. All of the respective interrupts are addressed in individual modules. The application verifies the operation of each interrupt and provides demonstration code that illustrates what is needed to setup, initiate and recover from the respective interrupts. The application's sources are summarized in the below table.

| File         | Description                                                                            |
|--------------|----------------------------------------------------------------------------------------|
| *.c          | These are the sources, most of which implement the code for the respective interrupts. |
| main.h       | This is a header used by the application.                                              |
| makefile     | This is the driver make file.                                                          |
| makefile.dep | This is a make dependency file. This is updated automatically.                         |

### 6.4.1. Build

Follow the below steps to build/rebuild the sample application.

1. Change to the directory where the sample application sources are installed (.../irq).
2. Build the sample application by issuing the below command.

```
make clean all
```

### 6.4.2. Execute

Follow the below steps to execute and exercise the test application.

1. Change to the directory where the sample application sources are installed (.../irq).
2. Start the test application by issuing the command given below. The arguments are described in the table below. All arguments are optional.

```
./irq <-c> <-C> <-m#> <-n#> <index>
```

| Argument | Description                                                                             |
|----------|-----------------------------------------------------------------------------------------|
| -c       | Repeat the operation until an error is encountered.                                     |
| -C       | Repeat the operation, but continue even if errors are encountered.                      |
| -m#      | When repeating the operation, stop after “#” minutes, where “#” is a decimal number.    |
| -n#      | When repeating the operation, stop after “#” iterations, where “#” is a decimal number. |
| index    | This is the zero based index of the channel to access.                                  |

3. The application will report board identification information then perform automated interrupt tests. Each test cycle should complete in less than one minute.

## 6.5. synctest- SYNC Test

This sample application performs an automated test of SYNC library services and the SIO4's corresponding functionality. The application's sources are summarized in the below table.

| File         | Description                                                                                                          |
|--------------|----------------------------------------------------------------------------------------------------------------------|
| *.c          | These are the sources which implement the IOCTL test code. Most contain functionality based on the file's base name. |
| main.h       | This is a header used by application.                                                                                |
| makefile     | This is the driver make file.                                                                                        |
| makefile.dep | This is a make dependency file. This is updated automatically.                                                       |

### 6.5.1. Build

Follow the below steps to build/rebuild the sample application.

1. Change to the directory where the sample application sources are installed (.../sync/synctest).
2. Build the sample application by issuing the below command.

```
make clean all
```

### 6.5.2. Execute

Follow the below steps to execute and exercise the test application.

1. Change to the directory where the sample application sources are installed (.../sync/synctest).
2. Start the test application by issuing the command given below. The arguments are described in the table below. All arguments are optional.

```
./synctest <-c> <-C> <-m#> <-n#> <index>
```

| Argument | Description                                                                             |
|----------|-----------------------------------------------------------------------------------------|
| -c       | Repeat the operation until an error is encountered.                                     |
| -C       | Repeat the operation, but continue even if errors are encountered.                      |
| -m#      | When repeating the operation, stop after “#” minutes, where “#” is a decimal number.    |
| -n#      | When repeating the operation, stop after “#” iterations, where “#” is a decimal number. |
| index    | This is the zero based index of the channel to access.                                  |

3. The application will report board identification information then perform a series of automated tests. Each test cycle should complete within just a few seconds.

## 6.6. Syncc2c- SYNC Channel-to-Channel

This sample application performs an automated test of SYNC transmit and receive functionality. The user specifies the transmit and receive channel indexes and the application transfers data between the channels specified. As appropriate, loop back mode is used. Otherwise an appropriate cable must be attached to the boards and channels specified. The application's sources are summarized in the below table.

| File         | Description                                                                                                          |
|--------------|----------------------------------------------------------------------------------------------------------------------|
| *.c          | These are the sources which implement the IOCTL test code. Most contain functionality based on the file's base name. |
| main.h       | This is a header used by application.                                                                                |
| makefile     | This is the driver make file.                                                                                        |
| makefile.dep | This is a make dependency file. This is updated automatically.                                                       |

### 6.6.1. Build

Follow the below steps to build/rebuild the sample application.

1. Change to the directory where the sample application sources are installed (.../sync/syncc2c).
2. Build the sample application by issuing the below command.

```
make clean all
```

### 6.6.2. Execute

Follow the below steps to execute and exercise the test application.

1. Change to the directory where the sample application sources are installed (.../sync/syncc2c).
2. Start the test application by issuing the command given below. The arguments are described in the table below. All but the last two arguments are optional.

```
./syncc2c <-c> <-C> <-m#> <-n#> tx# rx#
```

| Argument | Description                                                                             |
|----------|-----------------------------------------------------------------------------------------|
| -c       | Repeat the operation until an error is encountered.                                     |
| -C       | Repeat the operation, but continue even if errors are encountered.                      |
| -m#      | When repeating the operation, stop after “#” minutes, where “#” is a decimal number.    |
| -n#      | When repeating the operation, stop after “#” iterations, where “#” is a decimal number. |
| tx#      | This is the zero based index of the channel to transmit data over.                      |
| rx#      | This is the zero based index of the channel to receive data over.                       |

3. The application will report board identification information then perform transmit and receive data tests. Each test cycle should complete in about 30 seconds.



## 6.7. txrate - Transmit Rate

**NOTE:** This application is intended for SIO4BX-SYNC boards.

This sample application reports the expected transmit clock rate based upon the rates requested on the command line. The rates reported are those based on calculations and those reported by the driver. Alternatively the application will scan a range and report the expected transmit rate for each rate in the range. The application's sources are summarized in the below table.

| File         | Description                                                    |
|--------------|----------------------------------------------------------------|
| *.c          | These are the applications primary source files.               |
| main.h       | This is a header used by application.                          |
| makefile     | This is the driver make file.                                  |
| makefile.dep | This is a make dependency file. This is updated automatically. |

### 6.7.1. Build

Follow the below steps to build/rebuild the sample application.

1. Change to the directory where the sample application sources are installed (.../sync/txrate).
2. Build the sample application by issuing the below command.

```
make clean all
```

### 6.7.2. Execute

Follow the below steps to execute and exercise the test application.

1. Change to the directory where the sample application sources are installed (.../sync/txrate).
2. Start the test application by issuing the command given below. The arguments are described in the table below. All arguments are optional.

```
./txrate <-B#> <-b#> <-I> <-R> <-r#> <-sb#> <-se#>
 <ch0#> <ch1#> <ch2#> <ch3#>
```

| Argument | Description                                                                                                                                                |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -B#      | This is the zero based index of the board to access.                                                                                                       |
| -b#      | This is the number of bits in the firmware post divider that the rate calculations should use. The options are 11-bits or 15-bits. The default is 15-bits. |
| -I       | Initialize the channel if a rate is not specified.                                                                                                         |
| -R       | Reset the channel if a rate is not specified.                                                                                                              |
| -r#      | This is the reference oscillator frequency to use for calculations and for board setup.                                                                    |
| -sb#     | Perform a rate calculation scan beginning at this frequency.                                                                                               |
| -se#     | Perform a rate calculation scan ending at this frequency.                                                                                                  |
| ch0#     | This is the transmit rate to request for channel 0.                                                                                                        |
| ch1#     | This is the transmit rate to request for channel 1.                                                                                                        |
| ch2#     | This is the transmit rate to request for channel 2.                                                                                                        |
| ch3#     | This is the transmit rate to request for channel 3.                                                                                                        |

## 7. Driver Interface

The SIO4 driver conforms to the device driver standards required by the Linux Operating System and contains the standard driver entry points. The device driver provides a standard driver interface to the GSC SIO4 board for Linux applications. The interface includes various macros, data types and functions, all of which are described in the following paragraphs. The SIO4 specific portion of the driver interface is defined in the header files `sio4.h` and `sio4_sync_lib.h`, portions of which are described in this section. The headers define numerous items in addition to those described here.

**NOTE:** Contact General Standards Corporation if additional driver functionality is required.

**NOTE:** The driver included with this release is designed to work with the SIO4 models listed on the cover page of this user manual, as well as other models not listed. The driver interface may therefore include IOCTL services and support components intended for use with other models. Services and support components not documented in this manual should therefore not be used with the models listed on the cover. For other SIO4 models, refer to the applicable driver user manual.

### 7.1. Macros

The driver interface includes the following macros which are defined in `sio4.h` and `sio4_sync_lib.h`. These headers contain numerous additional utility type macros in addition to those described here.

#### 7.1.1. IOCTL

The IOCTL macros are documented following the function call descriptions.

#### 7.1.2. Registers

The following table gives the complete set of SIO4BX-SYNC registers. The tables are divided by register categories.

##### 7.1.2.1. GSC Registers

The following table gives the complete set of GSC specific SIO4BX-SYNC registers. For detailed definitions of these registers refer to the *SIO4 User Manual*.

| Macros                       | Description                                     |
|------------------------------|-------------------------------------------------|
| <code>SIO4_GSC_BCR</code>    | Board Control Register                          |
| <code>SIO4_GSC_BSR</code>    | Board Status Register                           |
| <code>SIO4_GSC_CSR</code>    | Control/Status Register                         |
| <code>SIO4_GSC_FCR</code>    | FIFO Count Register                             |
| <code>SIO4_GSC_FDR</code>    | FIFO Data Register                              |
| <code>SIO4_GSC_FR</code>     | Features Register                               |
| <code>SIO4_GSC_FRR</code>    | Firmware Revision Register                      |
| <code>SIO4_GSC_FSR</code>    | FIFO Size Register                              |
| <code>SIO4_GSC_ICR</code>    | Interrupt Control Register                      |
| <code>SIO4_GSC_ISR</code>    | Interrupt Status Register                       |
| <code>SIO4_GSC_IELR</code>   | Interrupt Edge/Level Register                   |
| <code>SIO4_GSC_IHLR</code>   | Interrupt Hi/low Register                       |
| <code>SIO4_GSC_POCSR</code>  | Programmable Oscillator Control/Status Register |
| <code>SIO4_GSC_PORAR</code>  | Programmable Oscillator RAM Address Register    |
| <code>SIO4_GSC_PORA2R</code> | Programmable Oscillator RAM Address 2 Register  |
| <code>SIO4_GSC_PORDR</code>  | Programmable Oscillator RAM Data Register       |

|                |                                        |
|----------------|----------------------------------------|
| SIO4_GSC_PSRCR | Pin Source Register                    |
| SIO4_GSC_PSTSR | Pin Status Register                    |
| SIO4_GSC_RAR   | Receiver Almost Empty/Full Register    |
| SIO4_GSC_RCR   | Rx Count Register                      |
| SIO4_GSC_TAR   | Transmitter Almost Empty/Full Register |
| SIO4_GSC_TCR   | Tx Count Register                      |

### 7.1.2.2. PCI Configuration Registers

The PCI registers are not listed as they are seldom required. For the file `sio4.h` for the applicable set of PCI9080 and PCI9056 register definitions.

### 7.1.2.3. PLX Feature Set Registers

The PLX Feature Set registers are not listed as they are seldom required. For the file `sio4.h` for the applicable set of PCI9080 and PCI9056 register definitions.

### 7.1.3. SIO4\_SYNC\_LIB\_VER

This macro gives the version number of the SYNC library. The major and minor version numbers are encoded in this value. The encoding formula is `major * 100 + minor`. This macro provides compile time access to the version number. The version number is also available at run time via the function `sio4_sync_version()`.

**NOTE:** This macro is defined in `sio4_sync_lib.h`.

| Macros            | Description                                  |
|-------------------|----------------------------------------------|
| SIO4_SYNC_LIB_VER | This macro gives the library version number. |

## 7.2. Data Types

This driver interface includes the following data types which are defined in `sio4.h`.

### 7.2.1. FIFO\_STATUS

This enumeration defines various possible values that may be received when reading a FIFO's status.

**NOTE:** Other values are possible but are not seen in normal use.

**NOTE:** The Almost Empty status becomes active when the FIFO contains *ALMOST EMPTY* or fewer bytes. Here, *ALMOST EMPTY* refers to the value programmed into the lower 16 bits of the Tx and Rx Almost Registers.

**NOTE:** The Almost Full status becomes active when the FIFO can receive *ALMOST FULL* or fewer additional bytes before being full. Here, *ALMOST FULL* refers to the value programmed into the upper 16 bits of the Tx and Rx Almost Registers.

#### Definition

```
typedef enum FIFOStatus
{
 ...
} FIFO_STATUS;
```

| Values                                  | Description                                                            |
|-----------------------------------------|------------------------------------------------------------------------|
| ALMOST_EMPTY_STATUS                     | The FIFO is almost full.                                               |
| ALMOST_FULL_STATUS                      | The FIFO is almost full.                                               |
| EMPTY_STATUS                            | The FIFO is empty.                                                     |
| FULL_STATUS                             | The FIFO is full.                                                      |
| INVALID_STATUS                          | The FIFO's current status is invalid.                                  |
| NOT_ALMOST_EMPTY_NOR_ALMOST_FULL_STATUS | The FIFO level is between the almost full and the almost empty states. |

### 7.2.2. sio4\_driver\_info\_t

This structure defines the data fields for the information returned by the SIO4\_GET\_DRIVER\_INFO IOCTL service.

Definition

```
typedef struct SIO4DriverInfo
{
 __u8 version[8];
 __u8 built[32];
} sio4_driver_info_t;
```

| Fields  | Description                                                                                                                     |
|---------|---------------------------------------------------------------------------------------------------------------------------------|
| version | This field gives the driver version number as a string in the form of X.XX.                                                     |
| built   | This field gives the driver build date and time as a string. It is given in the C form of printf("%s, %s", __DATE__, __TIME__). |

### 7.2.3. SIO4\_INTERRUPT\_STATUS

This structure records the interrupt status bits from the SIO4 Interrupt Status Register for the current channel. The bits reflect the accumulated status since the last interrupt notification or status request.

Definition

```
typedef struct IntStatus
{
 __u8 u8SIO4Status;
} SIO4_INTERRUPT_STATUS;
```

| Fields       | Description                                                                                                                                              |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| u8SIO4Status | The channel's interrupt status from the Interrupt Status Register. This may consist of either four or eight bits, depending on the board's capabilities. |

### 7.2.4. sio4\_mp\_chip\_t

This enumeration identifies the supported options for identifying the Multi-Protocol transceiver feature on the SIO4. The values are used in the chip field of the sio4\_mp\_t data structure, which is used with the Multi-Protocol transceiver based IOCTL services. Refers to the specific service for information on how this structure is used.

Definition

```
typedef enum
{
 ...
} sio4_mp_chip_t;
```

| Values               | Description                                                                                                        |
|----------------------|--------------------------------------------------------------------------------------------------------------------|
| SIO4_MP_CHIP_FIXED   | This refers to a fixed protocol implementation. The driver may not know which protocol is implemented on the SIO4. |
| SIO4_MP_CHIP_SP508   | This refers to the Sipex SP508 Multi-Protocol transceiver chip.                                                    |
| SIO4_MP_CHIP_UNKNOWN | The chip type is unknown.                                                                                          |

### 7.2.5. sio4\_mp\_prot\_t

This enumeration identifies the protocol options supported by the Multi-Protocol transceiver driver. The values are used in the want and got fields of the `sio4_mp_t` data structure, which is used with the Multi-Protocol transceiver based IOCTL services. Refer to the specific service for information on how this structure is used. Refer to the hardware user manual for detailed explanations of each protocol options.

#### Definition

```
typedef enum
{
 ...
} sio4_mp_prot_t;
```

| Values                  | Description                                                                              |
|-------------------------|------------------------------------------------------------------------------------------|
| SIO4_MP_PROT_DISABLE    | This refers to the disabled or tri-stated condition.                                     |
| SIO4_MP_PROT_INVALID    | This is returned by the driver when a requested protocol is unsupported or unrecognized. |
| SIO4_MP_PROT_READ       | This requests that the driver report the current protocol.                               |
| SIO4_MP_PROT_RS_232     | This refers to the RS-232 protocol.                                                      |
| SIO4_MP_PROT_RS_422_485 | This refers to the RS-422/RS-485 protocols.                                              |
| SIO4_MP_PROT_RS_423     | This refers to the RS-423 protocol.                                                      |
| SIO4_MP_PROT_UNKNOWN    | This is returned by the driver when the protocol is unknown.                             |

### 7.2.6. sio4\_mp\_t

This data structure is used to exchange information and requests about the board's Multi-Protocol transceiver feature between applications and the driver. This structure is used with the Multi-Protocol transceiver based IOCTL services. Refer to the specific service for information on how this structure is used.

#### Definition

```
typedef struct
{
 __s32 chip;
 __s32 prot_want;
 __s32 prot_got;
} sio4_mp_t;
```

| Field     | Description                                                                                                                                                                          |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| chip      | The driver will fill this field in with the Multi-Protocol transceiver chip identifier. Refer to the <code>sio4_mp_chip_t</code> data type documentation elsewhere in this document. |
| prot_want | This refers to the protocol desired by the application.                                                                                                                              |
| prot_got  | This refers to the protocol reported by the device.                                                                                                                                  |

### 7.2.7. sio4\_osc\_chip\_t

This enumeration identifies the supported options for identifying the programmable oscillator feature on the SIO4. The values are used in the `chip` field of the `sio4_osc_t` data structure, which is used with the programmable oscillator based IOCTL services. Refers to the specific service for information on how this structure is used.

#### Definition

```
typedef enum
{
 ...
} sio4_osc_chip_t;
```

| Values                   | Description                                                                                                             |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------|
| SIO4_OSC_CHIP_CY22393    | This refers to a single Cypress CY22393, which provides each SIO4 channel with its own programmable oscillator.         |
| SIO4_OSC_CHIP_CY22393_2  | This refers to two Cypress CY22393s, which provides each SIO4 channel with its own programmable oscillator.             |
| SIO4_OSC_CHIP_FIXED      | This refers to a fixed frequency, non-programmable oscillator that is shared by all SIO4 channels.                      |
| SIO4_OSC_CHIP_IDC2053B   | This refers to a single Cypress IDC2053B, which provides all SIO4 channel with the same programmable oscillator output. |
| SIO4_OSC_CHIP_IDC2053B_4 | This refers to four Cypress IDC2053B programmable oscillators, which provides each SIO4 channel with its own output.    |
| SIO4_OSC_CHIP_UNKNOWN    | The oscillator is unknown.                                                                                              |

### 7.2.8. sio4\_osc\_t

This data structure is used to exchange information and requests about the board's programmable oscillator between applications and the driver. This structure is used with the programmable oscillator based IOCTL services. Refers to the specific service for information on how this structure is used.

#### Definition

```
typedef struct
{
 __u32 chip;
 __s32 freq_ref;
 __s32 freq_want;
 __s32 freq_got;
} sio4_osc_t;
```

| Field     | Description                                                                                                                                                           |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| chip      | The driver will fill this field in with the oscillator chip identifier. Refer to the <code>sio4_osc_chip_t</code> data type documentation elsewhere in this document. |
| freq_ref  | This refers to the frequency of the oscillator's reference source.                                                                                                    |
| freq_want | This refers to the clock output frequency desired by the application.                                                                                                 |
| freq_got  | This refers to the clock output frequency produced by the device.                                                                                                     |

### 7.2.9. sio4\_reg\_t

This structure defines the data fields applicable to performing register read, write, and read-modify-write operations with the register access IOCTL services.

## Definition

```
typedef struct
{
 __u32 reg;
 __u32 value;
 __u32 mask;
} sio4_reg_t;
```

| Fields | Description                                                                                                                                                                                       |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| reg    | This identifies the register to access.                                                                                                                                                           |
| value  | The register value is placed here. This is either the value read from the register, the value to write to the register, or the bits to apply for modifications.                                   |
| mask   | This is the set of bits to modify for a read-modify-write access. If a bit is set here, then the corresponding “value” bit is applied to the register. Otherwise, the register bit is unmodified. |

## 7.2.10. sio4\_sync\_rx\_t

This structure defines SYNC specific receiver data fields. It is used both for retrieving configuration settings and applying configuration settings.

**NOTE:** This data type is defined in `sio4_sync_lib.h`.

## Definition

```
typedef struct
{
 int enable;
 int lsbf;
 int reset;
 __u16 word_size;

 struct
 {
 int high;
 } clock;

 struct
 {
 int enable;
 int high;
 } env;

 struct
 {
 int high;
 } data;
} sio4_sync_rx_t;
```

| Fields | Description                                                                                                                                                                                                                    |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| enable | This refers to the enabled/disabled state of the receiver. If zero, then the receiver is disabled. If non-zero, then the receiver is enabled.                                                                                  |
| lsbf   | This refers to the bit order that data is received in. If zero, then data is clocked in Most Significant Bit First. If non-zero, then data is clocked in Least Significant Bit First, hence the field name <code>lsbf</code> . |

|            |                                                                                                                                                                                                                                                           |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| reset      | This refers to Rx Count errors. If this is reported as zero, then no errors has occurred. If non-zero an error has occurred and needs to be reset. If zero when settings are being applied, then no action is taken. If non-zero then the error is reset. |
| word_size  | This refers to the number of bits in the last received word.                                                                                                                                                                                              |
| clock      | This refers to the Rx Clock configuration.                                                                                                                                                                                                                |
| clock.high | If zero, then the Rx Clock is active low (falling edge). If non-zero, then it is active high (rising edge).                                                                                                                                               |
| env        | This refers to the Rx Envelope configuration.                                                                                                                                                                                                             |
| env.enable | This refers to enabling or disabling the use of the Rx Envelope. If zero, then it is disabled. This is applicable for two-wire configurations. If non-zero, then it is enabled. This is applicable to three-wire configurations.                          |
| env.high   | If zero, then Rx Envelope is active low. If non-zero, then it is active high. This is ignored when Rx Envelope is disabled.                                                                                                                               |
| data       | This refers to the Rx Data configuration.                                                                                                                                                                                                                 |
| data.high  | If zero, then Rx Data is active low. If non-zero, then it is active high. This is ignored when Rx Data is disabled.                                                                                                                                       |

### 7.2.11. sio4\_sync\_t

This structure defines SYNC specific data fields that are independent of the board's transmitter and receiver. It is used both for retrieving configuration settings and applying configuration settings.

**NOTE:** This data type is defined in `sio4_sync_lib.h`.

#### Definition

```
typedef struct
{
 int dce_enable;
 int dte_enable;

 struct
 {
 int enable;
 int internal;
 } lb;
} sio4_sync_t;
```

| Fields     | Description                                                                                                                                                                                                 |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dce_enable | This refers to the board's DCE configuration. If zero, then the DCE mode is disabled. If non-zero, then the DCE mode is enabled. *                                                                          |
| dte_enable | This refers to the board's DTE configuration. If zero, then the DTE mode is disabled. If non-zero, then the DTE mode is enabled. The SYNC library ignores this field if the dce_enable field is non-zero. * |
| lb         | This substructure refers to the board's loopback configuration.                                                                                                                                             |
| enable     | This refers to the board's signal loopback capability. If zero, then the loopback feature is disabled. If non-zero, then the feature is enabled.                                                            |
| internal   | This refers to the use of internal or external loopback. If zero, then external loopback is utilized. If non-zero, then internal loopback is used. This field is ignored when loopback is disabled.         |

\* Either the DCE mode or the DTE mode must be enabled for data to be transferred across the cable interface. If both are disabled, then no data can be transferred.



**7.2.12. sio4\_sync\_tx\_t**

This structure defines SYNC specific transmitter data fields. It is used both for retrieving configuration settings and applying configuration settings.

**NOTE:** This data type is defined in `sio4_sync_lib.h`.

**Definition**

```
typedef struct
{
 int enable;
 int auto_dis;
 int lsbfi;
 __u16 word_size;
 __u16 gap_size;

 struct
 {
 int clock;
 int ext;
 int high;
 int idle;
 } clock;

 struct
 {
 int env;
 int high;
 } env;

 struct
 {
 int data;
 int high;
 } data;

 struct
 {
 int enable;
 int clock;
 int high;
 } aux_clock;

 struct
 {
 int enable;
 int high;
 } spare;
} sio4_sync_tx_t;
```

| Fields   | Description                                                                                                                                                |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| enable   | This refers to the enabled/disabled state of the transmitter. If zero, then the transmitter is disabled. If non-zero, then the transmitter is enabled.     |
| auto_dis | This refers to automatic disabling of the transmitter when the Tx FIFO runs empty. If zero, then the transmitter is not automatically disabled when the Tx |

|                  |                                                                                                                                                                                                                                                                                                                                                                    |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                  | FIFO runs empty. If non-zero, then the transmitter is automatically disabled when the Tx FIFO runs empty.                                                                                                                                                                                                                                                          |
| lsbf             | This refers to the bit order that data is transmitted. If zero, then data is clocked out Most Significant Bit First. If non-zero, then data is clocked out Least Significant Bit First, hence the field name <code>lsbf</code> . This refers to each byte within each transmit word. Multi-byte words are always sent out in the order that appear in the Tx FIFO. |
| word_size        | This refers to the size of each transmit word in bits.                                                                                                                                                                                                                                                                                                             |
| gap_size         | This refers to the number of bits transmitter at the completion of each transmit word.                                                                                                                                                                                                                                                                             |
| clock            | This refers to the Tx Clock configuration.                                                                                                                                                                                                                                                                                                                         |
| clock.clock      | This refers to the functionality of the Tx Clock signal. If zero, then the signal is a GPIO Output. If non-zero, then it is the transmit clock.                                                                                                                                                                                                                    |
| clock.ext        | This refers to the transmit clock's input source. If zero, then it is the onboard oscillator divided by two. If non-zero, then it is the Rx Auxiliary Clock.                                                                                                                                                                                                       |
| clock.high       | If Tx Clock is functioning as the transmit clock, then it is active low (falling edge) when zero, and active high (rising edge) when non-zero. For the GPIO functionality, it is low when zero and high when non-zero.                                                                                                                                             |
| clock.idle       | This refers to the transmit clock's operation when the transmitter is idle. When zero, the signal is not idle. It is driven with the transmit clock. When non-zero the signal is idle and is driven low.                                                                                                                                                           |
| env              | This refers to the Tx Envelope configuration.                                                                                                                                                                                                                                                                                                                      |
| env.env          | This refers to the functionality of the Tx Envelope signal. If zero, then the signal is a GPIO Output. If non-zero, then it is the transmit envelope.                                                                                                                                                                                                              |
| env.high         | If Tx Envelope is functioning as the transmit envelope, then it is active low when zero, and active high when non-zero. For the GPIO functionality, it is low when zero and high when non-zero.                                                                                                                                                                    |
| data             | This refers to the Tx Data configuration.                                                                                                                                                                                                                                                                                                                          |
| data.data        | This refers to the functionality of the Tx Data signal. If zero, then the signal is a GPIO Output. If non-zero, then it is the transmit data.                                                                                                                                                                                                                      |
| data.high        | If Tx Data is functioning as the transmit data, then it is active low when zero, and active high when non-zero. For the GPIO functionality, it is low when zero and high when non-zero.                                                                                                                                                                            |
| aux_clock        | This refers to the Tx Auxiliary Clock configuration.                                                                                                                                                                                                                                                                                                               |
| aux_clock.enable | This refers to the enable/disable state of the signal. If zero, then it is disabled and tri-stated. If non-zero, then it operates according to the below settings.                                                                                                                                                                                                 |
| aux_clock.clock  | This refers to the functionality of the Tx Auxiliary Clock signal. If zero, then the signal is a GPIO Output. If non-zero, then it is driven with the onboard oscillator divided by two.                                                                                                                                                                           |
| aux_clock.high   | If Tx Auxiliary Clock is functioning as a clock output, then it is active low (falling edge) when zero, and active high (rising edge) when non-zero. For the GPIO functionality, it is low when zero and high when non-zero.                                                                                                                                       |
| spare            | This refers to the Tx Spare configuration.                                                                                                                                                                                                                                                                                                                         |
| spare.enable     | This refers to the enable/disable state of the signal. If zero, then it is disabled and tri-stated. If non-zero, then it is a GPIO output.                                                                                                                                                                                                                         |
| spare.high       | If Tx Spare is enabled, then it is low when zero and high when non-zero.                                                                                                                                                                                                                                                                                           |

### 7.2.13. TX\_RX

This enumeration defines the possible external FIFO reset selection options.

Definition

```
typedef enum TxRx
{
```

```

 ...
} TX_RX;

```

| Fields         | Description              |
|----------------|--------------------------|
| RX_FIFO        | Reset the receive FIFO.  |
| TX_FIFO        | Reset the transmit FIFO. |
| TX_AND_RX_FIFO | Reset both FIFOs.        |

## 7.3. Functions

This driver interface includes the following functions.

### 7.3.1. close()

This function is the entry point to close a connection to an open SIO4 serial channel. This function should only be called after a successful open of the respective device. Upon closing the channel, all settings and configurations are put in a reset state. The programmable oscillator reference frequency is unaffected.

Prototype

```
int close(int fd);
```

| Argument | Description                                             |
|----------|---------------------------------------------------------|
| fd       | This is the file descriptor of the device to be closed. |

| Return Value | Description                                     |
|--------------|-------------------------------------------------|
| -1           | An error occurred. Consult <code>errno</code> . |
| 0            | The operation succeeded.                        |

Example

```

#include <errno.h>
#include <stdio.h>
#include <unistd.h>

#include "SIO4DocSrcLib.h"

int sio4_close(int fd, int verbose)
{
 int status;

 status = close(fd);

 if ((verbose) && (status == -1))
 printf("close() failure, errno = %d\n", errno);

 return(status);
}

```

### 7.3.2. ioctl()

This function is the entry point to performing setup and control operations on an open SIO4 serial channel. This function should only be called after a successful open of the respective device. The specific operation performed varies according to the `request` argument. The `request` argument also governs the use and interpretation of any additional arguments. The set of supported IOCTL services is defined in a following section.

### Prototype

```
int ioctl(int fd, int request, ...);
```

| Argument | Description                                                                                                                                                                                                                            |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| fd       | This is the file descriptor of the device to access.                                                                                                                                                                                   |
| request  | This specifies the desired operation to be performed.                                                                                                                                                                                  |
| ...      | This is any additional arguments. If request does not call for any additional arguments, then any additional arguments provided are ignored. The SIO4 IOCTL services use at most one argument, which is represented by a 32 bit value. |

| Return Value | Description                       |
|--------------|-----------------------------------|
| -1           | An error occurred. Consult errno. |
| 0            | The operation succeeded.          |

### Example

```
#include <errno.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/ioctl.h>

#include "sio4_dsl.h"

int sio4_ioctl(int fd, int request, unsigned long arg, int verbose)
{
 int status;

 status = ioctl(fd, request, arg);

 if ((verbose) && (status == -1))
 printf("ioctl() failure, errno = %d\n", errno);

 return(status);
}
```

### 7.3.3. open()

This function is the entry point to open a connection to an SIO4 serial channel. Upon opening the channel, all settings and configurations are put in an initialized state. The programmable oscillator reference frequency is unaffected.

### Prototype

```
int open(const char* pathname, int flags);
```

| Argument | Description                                        |
|----------|----------------------------------------------------|
| pathname | This is the name of the device to open.            |
| flags    | This is the desired read/write access. Use O_RDWR. |

**NOTE:** Another form of the open( ) function has a mode argument. This form is not displayed here as the mode argument is ignored when opening an existing file/device.

| Return Value | Description                       |
|--------------|-----------------------------------|
| -1           | An error occurred. Consult errno. |
| else         | A valid file descriptor.          |

**Example**

```

#include <assert.h>
#include <errno.h>
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>

#include "SIO4DocSrcLib.h"

int sio4_open(int board, int channel, int verbose)
{
 int fd;
 int index;
 char name[80];

 assert(board >= 0);
 assert((channel >= 0) && (channel <= 3));

 index = (board * 4) + channel + 1;
 sprintf(name, "/dev/sio4%d", index);
 fd = open(name, O_RDWR);

 if ((verbose) && (fd == -1))
 {
 printf("open() failure on %s, errno = %d\n",
 name,
 errno);
 }

 return(fd);
}

```

**7.3.4. read()**

This function is the entry point to reading received data from an open SIO4 serial channel. This function should only be called after a successful open of the respective device. The function reads up to `count` bytes from the receive FIFO. If the number of bytes requested is not available within the configured time limit, the read operation times out.

**NOTE:** Refer to the SIO4\_RX\_IO\_MODE\_CONFIG IOCTL services to configure this call for use of PIO, DMA or Demand Mode DMA data transfer.

**NOTE:** While performing the data transfer the driver may use the Rx FIFO Almost Full interrupt to wait for the arrival of additional data. This can occur only when performing reads using PIO or non-Demand Mode DMA. It does not occur when using Demand Mode DMA.

**Prototype**

```
int read(int fd, void *buf, size_t count);
```

| Argument | Description                                          |
|----------|------------------------------------------------------|
| fd       | This is the file descriptor of the device to access. |
| buf      | The data read will be put here.                      |
| count    | This is the desired number of bytes to read.         |

| Return Value | Description                                                                                                |
|--------------|------------------------------------------------------------------------------------------------------------|
| -1           | An error occurred. Consult <code>errno</code> .                                                            |
| 0 to count   | The operation succeeded. If the return value is less than <code>count</code> , then the request timed out. |

#### Example

```
#include <errno.h>
#include <stddef.h>
#include <stdio.h>
#include <unistd.h>

#include "SIO4DocSrcLib.h"

int sio4_read(int fd, void *buf, size_t count, int verbose)
{
 int status;

 status = read(fd, buf, count);

 if ((verbose) && (status == -1))
 printf("read() failure, errno = %d\n", errno);

 return(status);
}
```

#### 7.3.5. sio4\_sync\_get()

This function is the entry point to retrieving SYNC configuration settings that are independent of the transmitter and receiver. This function should only be called after a successful open of the respective device.

**NOTE:** The prototype for this function is in `sio4_sync_lib.h`.

#### Prototype

```
int sio4_sync_get(int fd, sio4_sync_t* sync);
```

| Argument | Description                                                                                                |
|----------|------------------------------------------------------------------------------------------------------------|
| fd       | This is the file descriptor of the device to access.                                                       |
| sync     | This points to the structure that is to receive the current configuration state. This pointer may be NULL. |

| Return Value | Description                                                                   |
|--------------|-------------------------------------------------------------------------------|
| 0            | No errors were encountered.                                                   |
| > 0          | An appropriate error value, which is the applicable <code>errno</code> value. |

#### Example

```
#include <errno.h>
#include <stdio.h>
```

```

#include "sio4_sync_doc_src_lib.h"

int sio4_sync_dce_set(int fd, int enable, int verbose)
{
 sio4_sync_t sync;
 int status;
 const char* str = "sio4_sync_get";

 status = sio4_sync_get(fd, &sync);

 if (status == 0)
 {
 sync.dce_enable = enable;
 status = sio4_sync_set(fd, &sync);
 str = "sio4_sync_set";
 }

 if ((verbose) && (status == -1))
 printf("%s() failure, errno = %d\n", str, errno);

 return(status);
}

```

### 7.3.6. sio4\_sync\_gpio\_rx()

This function is the entry point to reading the SYNC board's GPIO inputs. The value read represents the current value of all signals included in the Pin Source Register, even those not configured as GPIO. Bits undefined in the register are returned as zero. This function should only be called after a successful open of the respective device.

**NOTE:** The prototype for this function is in `sio4_sync_lib.h`.

#### Prototype

```
int sio4_sync_gpio_rx(int fd, __u32* value);
```

| Argument | Description                                                                      |
|----------|----------------------------------------------------------------------------------|
| fd       | This is the file descriptor of the device to access.                             |
| value    | This points to the variable to receive the value read. This pointer may be NULL. |

| Return Value | Description                                                      |
|--------------|------------------------------------------------------------------|
| 0            | No errors were encountered.                                      |
| > 0          | An appropriate error value, which is the applicable errno value. |

#### Example

```

#include <errno.h>
#include <stdio.h>

#include "sio4_sync_doc_src_lib.h"

int sio4_sync_gpio_read_tx(int fd, __u32* value, int verbose)
{
 int status;

 status = sio4_sync_gpio_rx(fd, value);
}

```

```

value[0] &= 0x2F0;

if ((verbose) && (status == -1))
 printf("sio4_sync_gpio_read() failure, errno = %d\n", errno);

return(status);
}

```

### 7.3.7. sio4\_sync\_gpio\_tx()

This function is the entry point to writing to the SYNC board's GPIO outputs. Only those signals currently configured as GPIO outputs are affected. All bits that do not correspond to GPIO outputs are quietly ignored. This function should only be called after a successful open of the respective device.

**NOTE:** The prototype for this function is in `sio4_sync_lib.h`.

#### Prototype

```
int sio4_sync_gpio_tx(int fd, __u32 value);
```

| Argument | Description                                          |
|----------|------------------------------------------------------|
| fd       | This is the file descriptor of the device to access. |
| value    | This is the value to write to the GPIO outputs.      |

| Return Value | Description                                                      |
|--------------|------------------------------------------------------------------|
| 0            | No errors were encountered.                                      |
| > 0          | An appropriate error value, which is the applicable errno value. |

#### Example

```

#include <errno.h>
#include <stdio.h>

#include "sio4_sync_doc_src_lib.h"

int sio4_sync_gpio_mod(int fd, __u32 value, __u32 mask, int verbose)
{
 int status;
 const char* str = "sio4_sync_gpio_rx";
 __u32 v;

 status = sio4_sync_gpio_rx(fd, &v);

 if (status == 0)
 {
 value = (value & mask) | (v & ~mask);
 status = sio4_sync_gpio_tx(fd, value);
 str = "sio4_sync_gpio_tx";
 }

 if ((verbose) && (status == -1))
 printf("%s() failure, errno = %d\n", str, errno);

 return(status);
}

```



### 7.3.8. sio4\_sync\_rx\_get()

This function is the entry point to retrieving receiver specific SYNC configuration settings. This function should only be called after a successful open of the respective device.

**NOTE:** The prototype for this function is in `sio4_sync_lib.h`.

#### Prototype

```
int sio4_sync_rx_get(int fd, sio4_sync_rx_t* rx);
```

| Argument | Description                                                                                                |
|----------|------------------------------------------------------------------------------------------------------------|
| fd       | This is the file descriptor of the device to access.                                                       |
| rx       | This points to the structure that is to receive the current configuration state. This pointer may be NULL. |

| Return Value | Description                                                      |
|--------------|------------------------------------------------------------------|
| 0            | No errors were encountered.                                      |
| > 0          | An appropriate error value, which is the applicable errno value. |

#### Example

```
#include <errno.h>
#include <stdio.h>

#include "sio4_sync_doc_src_lib.h"

int sio4_sync_rx_data_get(int fd, int* high, int verbose)
{
 sio4_sync_rx_t rx;
 int status;

 status = sio4_sync_rx_get(fd, &rx);
 high[0] = rx.data.high;

 if ((verbose) && (status == -1))
 printf("sio4_sync_rx_get() failure, errno = %d\n", errno);

 return(status);
}
```

### 7.3.9. sio4\_sync\_rx\_set()

This function is the entry point to applying receiver specific SYNC configuration settings. This function should only be called after a successful open of the respective device.

**NOTE:** The prototype for this function is in `sio4_sync_lib.h`.

#### Prototype

```
int sio4_sync_rx_set(int fd, const sio4_sync_rx_t* rx);
```

| Argument | Description                                                                                 |
|----------|---------------------------------------------------------------------------------------------|
| fd       | This is the file descriptor of the device to access.                                        |
| rx       | This points to the structure that contains the settings to apply. This pointer may be NULL. |

| Return Value | Description                                                      |
|--------------|------------------------------------------------------------------|
| 0            | No errors were encountered.                                      |
| > 0          | An appropriate error value, which is the applicable errno value. |

**Example**

```
#include <errno.h>
#include <stdio.h>

#include "sio4_sync_doc_src_lib.h"

int sio4_sync_rx_env_set(int fd, int enable, int high, int verbose)
{
 sio4_sync_rx_t rx;
 int status;
 const char* str = "sio4_sync_rx_get";

 status = sio4_sync_rx_get(fd, &rx);

 if (status == 0)
 {
 rx.env.enable = enable;
 rx.env.high = high;
 status = sio4_sync_rx_set(fd, &rx);
 str = "sio4_sync_rx_set";
 }

 if ((verbose) && (status == -1))
 printf("%s() failure, errno = %d\n", str, errno);

 return(status);
}
```

**7.3.10. sio4\_sync\_set()**

This function is the entry point to applying SYNC configuration settings that are independent of the transmitter and receiver. This function should only be called after a successful open of the respective device.

**NOTE:** The prototype for this function is in `sio4_sync_lib.h`.

**Prototype**

```
int sio4_sync_set(int fd, const sio4_sync_t* sync);
```

| Argument | Description                                                                             |
|----------|-----------------------------------------------------------------------------------------|
| fd       | This is the file descriptor of the device to access.                                    |
| sync     | This points to the structure containing the setting to apply. This pointer may be NULL. |

| Return Value | Description                                                      |
|--------------|------------------------------------------------------------------|
| 0            | No errors were encountered.                                      |
| > 0          | An appropriate error value, which is the applicable errno value. |

**Example**

```

#include <errno.h>
#include <stdio.h>

#include "sio4_sync_doc_src_lib.h"

int sio4_sync_dte_get(int fd, int* enable, int verbose)
{
 sio4_sync_t sync;
 int status;

 status = sio4_sync_get(fd, &sync);
 enable[0] = sync.dte_enable;

 if ((verbose) && (status == -1))
 printf("sio4_sync_get() failure, errno = %d\n", errno);

 return(status);
}

```

**7.3.11. sio4\_sync\_tx\_get()**

This function is the entry point to retrieving transmitter specific SYNC configuration settings. This function should only be called after a successful open of the respective device.

**NOTE:** The prototype for this function is in `sio4_sync_lib.h`.

**Prototype**

```
int sio4_sync_tx_get(int fd, sio4_sync_tx_t* tx);
```

| Argument | Description                                                                                                |
|----------|------------------------------------------------------------------------------------------------------------|
| fd       | This is the file descriptor of the device to access.                                                       |
| tx       | This points to the structure that is to receive the current configuration state. This pointer may be NULL. |

| Return Value | Description                                                                   |
|--------------|-------------------------------------------------------------------------------|
| 0            | No errors were encountered.                                                   |
| > 0          | An appropriate error value, which is the applicable <code>errno</code> value. |

**Example**

```

#include <errno.h>
#include <stdio.h>

#include "sio4_sync_doc_src_lib.h"

int sio4_sync_tx_env_get(int fd, int* env, int* high, int verbose)
{
 sio4_sync_tx_t tx;
 int status;

 status = sio4_sync_tx_get(fd, &tx);
 env[0] = tx.env.env;
}

```

```

 high[0] = tx.env.high;

 if ((verbose) && (status == -1))
 printf("sio4_sync_tx_get() failure, errno = %d\n", errno);

 return(status);
}

```

### 7.3.12. sio4\_sync\_tx\_set()

This function is the entry point to applying transmitter specific SYNC configuration settings. This function should only be called after a successful open of the respective device.

**NOTE:** The prototype for this function is in `sio4_sync_lib.h`.

#### Prototype

```
int sio4_sync_tx_set(int fd, const sio4_sync_tx_t* tx);
```

| Argument | Description                                                                                 |
|----------|---------------------------------------------------------------------------------------------|
| fd       | This is the file descriptor of the device to access.                                        |
| tx       | This points to the structure that contains the settings to apply. This pointer may be NULL. |

| Return Value | Description                                                      |
|--------------|------------------------------------------------------------------|
| 0            | No errors were encountered.                                      |
| > 0          | An appropriate error value, which is the applicable errno value. |

#### Example

```

#include <errno.h>
#include <stdio.h>

#include "sio4_sync_doc_src_lib.h"

int sio4_sync_tx_data_set(int fd, int data, int high, int verbose)
{
 sio4_sync_tx_t tx;
 int status;
 const char* str = "sio4_sync_tx_get";

 status = sio4_sync_tx_get(fd, &tx);

 if (status == 0)
 {
 tx.data.data = data;
 tx.data.high = high;
 status = sio4_sync_tx_set(fd, &tx);
 str = "sio4_sync_tx_set";
 }

 if ((verbose) && (status == -1))
 printf("%s() failure, errno = %d\n", str, errno);

 return(status);
}

```

### 7.3.13. sio4\_sync\_version()

This function is the entry point to retrieving the SYNC library's version information. This function may be called at any time and does not require access to an SIO4.

**NOTE:** The prototype for this function is in `sio4_sync_lib.h`.

#### Prototype

```
void sio4_sync_version(int* version, const char** built);
```

| Argument | Description                                                                                                                                                               |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| version  | This returns the version number. The value returned is the value of the macro <code>SIO4_SYNC_LIB_VER</code> at the time the library was built. This pointer may be NULL. |
| built    | This gives the library build date and time as a string. It is given in the C form of <code>printf("%s, %s", __DATE__, __TIME__)</code> . This pointer may be NULL.        |

#### Example

```
#include <errno.h>
#include <stdio.h>

#include "sio4_sync_doc_src_lib.h"

void sio4_sync_version_get(
 int* version,
 const char** built,
 int verbose)
{
 const char* blt;
 int ver;

 sio4_sync_version(&ver, &blt);

 if (verbose)
 {
 printf("SIO4-SYNC Library Version:\n");
 printf(" Version: %d.%02d\n", ver / 100, ver % 100);
 printf(" Built: %s\n", blt);
 }

 if (version)
 version[0] = ver;

 if (built)
 built[0] = blt;
}
```

### 7.3.14. write()

This function is the entry point to writing data for transmission to an open SIO4 serial channel. This function should only be called after a successful open of the respective device. The function writes up to `count` bytes to the transmit FIFO. If the number of bytes requested cannot be sent within the configured time limit, the write operation times out.

**NOTE:** Refer to the `SIO4_TX_IO_MODE_CONFIG` IOCTL services to configure this call for use of PIO, DMA or Demand Mode DMA data transfer.

**NOTE:** While performing the data transfer the driver may use the Tx FIFO Almost Empty interrupt to wait for additional space to become available in the Tx FIFO as data is being pulled from the FIFO and processed by the transmitter. This can occur only when performing writes using PIO or non-Demand Mode DMA. It does not occur when using Demand Mode DMA.

### Prototype

```
int write(int fd, const void *buf, size_t count);
```

| Argument | Description                                          |
|----------|------------------------------------------------------|
| fd       | This is the file descriptor of the device to access. |
| buf      | The data written comes from here.                    |
| count    | This is the desired number of bytes to write.        |

| Return Value | Description                                                                                                |
|--------------|------------------------------------------------------------------------------------------------------------|
| -1           | An error occurred. Consult <code>errno</code> .                                                            |
| 0 to count   | The operation succeeded. If the return value is less than <code>count</code> , then the request timed out. |

### Example

```
#include <errno.h>
#include <stddef.h>
#include <stdio.h>
#include <unistd.h>

#include "SIO4DocSrcLib.h"

int sio4_write(int fd, const void *buf, size_t count, int verbose)
{
 int status;

 status = write(fd, buf, count);

 if ((verbose) && (status == -1))
 printf("write() failure, errno = %d\n", errno);

 return(status);
}
```

## 7.4. IOCTL Services

The SIO4 driver implements the following IOCTL services. Each service is described along with the applicable `ioctl()` function arguments. In the definitions given the optional argument is identified as `arg` and is an unsigned 32-bit data type. Unless otherwise stated the return value definitions are those defined for the `ioctl()` function call.

**NOTE:** Many of the IOCTL services alter the state of the channel's operation and can adversely affect the channel's proper operation if data transfer is in progress. Exercise care when using these services to insure that data integrity is maintained.

### 7.4.1. SIO4\_BOARD\_JUMPERS

This service reads the jumper information for the user jumpers. If the jumpers are not supported on the board in use, then the returned value is the XXX\_UNKNOWN macro. If the jumpers are supported, then the value returned will be from 0x0 to 0x3 for boards with two jumpers and from 0x0 to 0xF for boards with four jumpers.

#### Usage

| <b>ioctl() Argument</b> | <b>Description</b> |
|-------------------------|--------------------|
| request                 | SIO4_BOARD_JUMPERS |
| arg                     | __s32*             |

The table below lists the predefined macros used with this service.

| <b>Macros</b>                  | <b>Description</b>                 |
|--------------------------------|------------------------------------|
| SIO4_BOARD_JUMPERS_UNSUPPORTED | The board jumpers are unsupported. |

### 7.4.2. SIO4\_FEATURE\_TEST

This service provides information on an SIO4's feature set. To gain support information on a specific feature the corresponding macro is supplied. The value returned will be the corresponding support information, which is the XXX\_YES or XXX\_NO macro in most cases. If the XXX\_COUNT macro is supplied, the value returned is the number of feature options supported by the service, and should be one more than the service's XXX\_LAST\_INDEX macro.

#### Usage

| <b>ioctl() Argument</b> | <b>Description</b> |
|-------------------------|--------------------|
| request                 | SIO4_FEATURE_TEST  |
| arg                     | __s32*             |

The table below lists the options used with this service.

| <b>Macros</b>                  | <b>Description</b>                                                                                       |
|--------------------------------|----------------------------------------------------------------------------------------------------------|
| SIO4_FEATURE_BCR_BOARD_RESET   | Does the Board Control Register support the Board Reset bit?                                             |
| SIO4_FEATURE_BCR_RX_FFC        | Does the Board Control Register support the Rx FIFO Full Configuration bit?                              |
| SIO4_FEATURE_BCR_SCD           | Does the Board Control Register support the Single Cycle Disable bit?                                    |
| SIO4_FEATURE_BSR               | Is the Board Status Register supported?                                                                  |
| SIO4_FEATURE_BOARD_CHANNELS    | This refers to the number of channels on the entire board and should be either four or eight.            |
| SIO4_FEATURE_COUNT             | This reports the number of features supported by the service.                                            |
| SIO4_FEATURE_FIFO_COUNT        | Are the FIFO Count registers supported?                                                                  |
| SIO4_FEATURE_FIFO_SIZE         | Are the FIFO Size registers supported?                                                                   |
| SIO4_FEATURE_FR                | Is the Features Register supported?                                                                      |
| SIO4_FEATURE_FW_PD_BITS        | This is the number of bits in the Firmware Post Divider. This is zero if the dividers are not supported. |
| SIO4_FEATURE_IRQ_32            | Are all 32-bits of the interrupt configuration registers significant?                                    |
| SIO4_FEATURE_LEGACY_CABLE_CTRL | Does the firmware include the legacy cable interface control?                                            |
| SIO4_FEATURE_MP                | Is the Multi-Protocol transceiver feature in firmware?                                                   |

|                           |                                                           |
|---------------------------|-----------------------------------------------------------|
| SIO4_FEATURE_MP_CHIP      | Which Multi-Protocol transceiver chip is present?         |
| SIO4_FEATURE_MP_PROGRAM   | Can a transceiver selection be reprogrammed?              |
| SIO4_FEATURE_OSC_CHIP     | Which programmable oscillator chip is present?            |
| SIO4_FEATURE_OSC_MEASURE  | Is the driver able to measure the oscillator's frequency? |
| SIO4_FEATURE_OSC_PER_CHAN | Is each channel separately and individually programmable? |
| SIO4_FEATURE_OSC_PROGRAM  | Is the driver able to program the oscillator?             |
| SIO4_FEATURE_PORD2R       | Is the PORD2R register supported?                         |
| SIO4_FEATURE_PSRCR        | Are the Pin Source Registers supported?                   |
| SIO4_FEATURE_PSTSR        | Are the Pin Status Registers supported?                   |
| SIO4_FEATURE_USER_JUMPERS | Returns zero, two or four, depending on support.          |

The table below lists common response values for most the feature options.

| Macros               | Description                                                          |
|----------------------|----------------------------------------------------------------------|
| SIO4_FEATURE_NO      | The feature is not supported.                                        |
| SIO4_FEATURE_UNKNOWN | Either the feature is unknown or support for the feature is unknown. |
| SIO4_FEATURE_YES     | The feature is supported.                                            |

#### 7.4.3. SIO4\_GET\_DRIVER\_INFO

This service retrieves information about the driver itself. At this time this includes only a driver version string.

Usage

| <b>ioctl()</b> Argument | Description          |
|-------------------------|----------------------|
| request                 | SIO4_GET_DRIVER_INFO |
| arg                     | Sio4_driver_info_t*  |

#### 7.4.4. SIO4\_INIT\_BOARD

This service initializes all of the board's hardware. This includes the transmitter, the receiver, the FIFOs, the cable configurations, the transceivers and the programmable oscillators. For boards with programmable oscillators and programmable transceivers, these features are initialized in preparation for use.

Usage

| <b>ioctl()</b> Argument | Description     |
|-------------------------|-----------------|
| request                 | SIO4_INIT_BOARD |
| arg                     | Not used.       |

#### 7.4.5. SIO4\_INT\_NOTIFY

This service requests that the application be notified of one or more interrupts on the given serial channel. The parameter value is the bit wise or-ing of the possible notification bits. (The bits are defined in a previous section of this document.) Notification is given only for those bits which are set. Passing in a value of zero (0) cancels all notification requests. Once a specified interrupt occurs the driver clears and disables the interrupt, then notifies the application via a SIGIO (from `signal.h`) signal. To receive any subsequent notifications the application must make another notification request. The referenced interrupts are enabled. Unreferenced interrupts are disabled.

**NOTE:** The application will not receive notification of any interrupt that the driver itself is waiting on. The driver may use the Rx FIFO Almost Full interrupt during read requests and the Tx FIFO Almost Empty interrupt during writes. Refer to the `read()` service (page 37) and the `write()` service (page 45) for additional information.



**NOTE:** Interrupt options referenced but unsupported by the current hardware are quietly ignored.

#### Usage

| <b>ioctl( ) Argument</b> | <b>Description</b> |
|--------------------------|--------------------|
| request                  | SIO4_INT_NOTIFY    |
| arg                      | unsigned char      |

The table below lists the options used with this service. These options may be used in any bitwise combination.

| <b>Macros</b>              | <b>Description</b>                  |
|----------------------------|-------------------------------------|
| SIO4_INT_NOTIFY_RX_FIFO_AF | The Rx FIFO Almost Full interrupt.  |
| SIO4_INT_NOTIFY_RX_FIFO_E  | The Rx FIFO Empty interrupt.        |
| SIO4_INT_NOTIFY_RX_FIFO_F  | The Rx FIFO Full interrupt.         |
| SIO4_INT_NOTIFY_TX_FIFO_AE | The Tx FIFO Almost Empty interrupt. |
| SIO4_INT_NOTIFY_TX_FIFO_E  | The Tx FIFO Empty interrupt.        |
| SIO4_INT_NOTIFY_TX_FIFO_F  | The Tx FIFO Full interrupt.         |

#### 7.4.6. SIO4\_MOD\_REGISTER

This service performs a read-modify-write operation on an SIO4 register. This includes only the GSC firmware registers and USC registers. The PCI registers and the PLX feature set registers are read-only. Refer to the SIO4 User Manual and to `sio4.h` for a complete list of the available registers.

#### Usage

| <b>ioctl( ) Argument</b> | <b>Description</b> |
|--------------------------|--------------------|
| request                  | SIO4_MOD_REGISTER  |
| arg                      | Sio4_reg_t*        |

#### 7.4.7. SIO4\_MP\_CONFIG

This service is used to select and/or report on the current transceiver protocol. The driver uses the `prot_want` field and ignores all others. The results are recorded in the data structure's `prot_got` field. Refer to the Multi-Protocol transceiver programming information later in this document for more information.

**NOTE:** The driver will fulfill the request based on the SIO4's capabilities. When the protocol can be changed and that requested is available, the requested change will be selected. Requests will otherwise fail and the protocol will be unchanged.

#### Usage

| <b>ioctl( ) Argument</b> | <b>Description</b> |
|--------------------------|--------------------|
| request                  | SIO4_MP_CONFIG     |
| arg                      | sio4_mp_t*         |

#### 7.4.8. SIO4\_MP\_INFO

This service returns information about the current Multi-Protocol transceiver configuration. All field contents are ignored and are set by the driver according to the current configuration. Refer to the Multi-Protocol transceiver programming information later in this document for more information.

## Usage

| <b>ioctl() Argument</b> | <b>Description</b> |
|-------------------------|--------------------|
| request                 | SIO4_MP_INFO       |
| arg                     | sio4_mp_t*         |

**7.4.9. SIO4\_MP\_INIT**

This service initializes the board's Multi-Protocol transceiver feature. This returns the Multi-Protocol transceivers to their initial power up state. The results are recorded in the data structure's `prot_got` field. Refer to the Multi-Protocol transceiver programming information later in this document for more information.

## Usage

| <b>ioctl() Argument</b> | <b>Description</b> |
|-------------------------|--------------------|
| request                 | SIO4_MP_INIT       |
| arg                     | sio4_mp_t*         |

**7.4.10. SIO4\_MP\_RESET**

This service resets the board's Multi-Protocol transceiver feature. This disables the transceivers by tri-stating the outputs. The results are recorded in the data structure's `prot_got` field. Refer to the Multi-Protocol transceiver programming information later in this document for more information.

## Usage

| <b>ioctl() Argument</b> | <b>Description</b> |
|-------------------------|--------------------|
| request                 | SIO4_MP_RESET      |
| arg                     | sio4_mp_t*         |

**7.4.11. SIO4\_MP\_TEST**

This service is used to determine if the board's Multi-Protocol transceiver feature supports a given protocol. The protocol to be tested is recorded in the structure's `prot_want` field. The results are recorded in the data structure's `prot_got` field. The reported value will be `SIO4_MP_PROT_INVALID` if the requested protocol value is unrecognized or unsupported. It will be `SIO4_MP_PROT_UNKNOWN` when support for the specified protocol is unknown. This is applicable when the SIO4 doesn't support the feature or when the chip used is unsupported by the driver. The reported value will equal the requested protocol when that protocol is supported. Refer to the Multi-Protocol transceiver programming information later in this document for more information.

## Usage

| <b>ioctl() Argument</b> | <b>Description</b> |
|-------------------------|--------------------|
| request                 | SIO4_MP_TEST       |
| arg                     | sio4_mp_t*         |

**7.4.12. SIO4\_OSC\_INFO**

This service returns current configuration information about the channel's oscillator. The driver ignores the structure's current content and fills in all fields according to the channel's current configuration. Refer to the oscillator programming information later in this document for more information.

## Usage

| <b>ioctl() Argument</b> | <b>Description</b> |
|-------------------------|--------------------|
| request                 | SIO4_OSC_INFO      |
| arg                     | sio4_osc_t*        |

**7.4.13. SIO4\_OSC\_INIT**

This service initializes the channel's programmable oscillator hardware. The channel's input clock will be reprogrammed to output the reference frequency as a result of this service, depending on the device's capabilities. The driver ignores the structure's current content and fills in all fields according to the channel's post-initialization configuration. The reference frequency is unaltered, the desired frequency is set to the reference frequency, and the frequency obtained is reported. Refer to the oscillator programming information later in this document for more information.

## Usage

| <b>ioctl() Argument</b> | <b>Description</b> |
|-------------------------|--------------------|
| request                 | SIO4_OSC_INIT      |
| arg                     | sio4_osc_t*        |

**7.4.14. SIO4\_OSC\_MEASURE**

This service is used to measure the frequency produced by the current oscillator hardware configuration. The driver ignores all structure field values and fills them in according to the test results and the channel's current configuration. The test results are recorded in the data structure's `freq_got` field. A value of -1 is reported when the frequency can't be measured. Refer to the oscillator programming information later in this document for more information.

**NOTE:** The driver will perform a measurement test based on the SIO4's capabilities. When a measurement can be made, the test duration and the accuracy of the results are dependent on the board's capabilities. Refer to the hardware manual for additional details.

## Usage

| <b>ioctl() Argument</b> | <b>Description</b> |
|-------------------------|--------------------|
| request                 | SIO4_OSC_MEASURE   |
| arg                     | sio4_osc_t*        |

**7.4.15. SIO4\_OSC\_PROGRAM**

This service is used to update and report on the programmed frequency produced by the channel's oscillator hardware. This service will reprogram the channel's oscillator hardware to produce the requested frequency, or one as near as possible to that requested. The resulting frequency will depend on the capability of the hardware and how its resources are being used, as applicable. If the requested value is -1, then the service will report the channel's current configuration without making any changes. The driver ignores all other fields and fills them in according to the channel's post-programming configuration. Refer to the oscillator programming information later in this document for more information.

## Usage

| <b>ioctl() Argument</b> | <b>Description</b> |
|-------------------------|--------------------|
| request                 | SIO4_OSC_PROGRAM   |
| arg                     | sio4_osc_t*        |

### 7.4.16. SIO4\_OSC\_REFERENCE

This service is used to update and report on the recorded frequency for the channel's reference source. Changing this setting does not alter any existing programming results. New settings apply to subsequent calculations only! The only argument field used by the driver is the `freq_ref` field. If its value is `-1`, then the driver will report the current recorded reference frequency. The value supplied will otherwise be qualified per the requirements of the channel's oscillator and recorded for subsequent use. An error will be reported if it is invalid. The driver ignores all other fields and fills them in according to the channel's current configuration. This service does not alter any other oscillator related parameter. Refer to the oscillator programming information later in this document for more information.

**CAUTION:** Setting the reference frequency to an incorrect value may have an adverse affect on the programmable oscillator. The results depend on the oscillator and the incorrect value specified.

#### Usage

| <b>ioctl()</b> Argument | Description                     |
|-------------------------|---------------------------------|
| <code>request</code>    | <code>SIO4_OSC_REFERENCE</code> |
| <code>arg</code>        | <code>sio4_osc_t*</code>        |

### 7.4.17. SIO4\_OSC\_RESET

This service resets the channel's oscillator hardware. The channel's input clock will be set to the lowest possible frequency as a result of this service, depending on the device's capabilities. The driver ignores the structure's current content and fills in all fields according to the channel's post-reset configuration. The reference frequency is unaltered, the desired frequency is set to zero, and the frequency obtained is reported. Refer to the oscillator programming information later in this document for more information.

#### Usage

| <b>ioctl()</b> Argument | Description                 |
|-------------------------|-----------------------------|
| <code>request</code>    | <code>SIO4_OSC_RESET</code> |
| <code>arg</code>        | <code>sio4_osc_t*</code>    |

### 7.4.18. SIO4\_OSC\_TEST

This service reports the frequency that should be produced were the programming service requested for the desired frequency. The channel's input clock will be set to the lowest possible frequency as a result of this service, depending on the device's capabilities. The driver ignores the structure's current content and fills in all fields according to the channel's post-reset configuration. The reference frequency is unaltered, the desired frequency is set to zero, and the frequency obtained is reported. Refer to the oscillator programming information later in this document for more information.

#### Usage

| <b>ioctl()</b> Argument | Description                |
|-------------------------|----------------------------|
| <code>request</code>    | <code>SIO4_OSC_TEST</code> |
| <code>arg</code>        | <code>sio4_osc_t*</code>   |

### 7.4.19. SIO4\_READ\_INT\_STATUS

This service requests the interrupt status information following interrupt notification. The status reported reflects all of the interrupts for the channel. The recorded status represents the accumulated status of all interrupts since the status was last read or notification requested. Once read, the recorded status is cleared.

**NOTE:** The application will not receive notification of any interrupt that the driver itself is waiting on.

**NOTE:** Due to the timeliness of various interacting events it is possible for multiple interrupts to occur before the status is read. This can result in one SIGIO prompted status read reporting multiple interrupts and the next SIGIO prompted status read reporting no interrupts.

#### Usage

| <b>ioctl()</b> Argument | Description            |
|-------------------------|------------------------|
| request                 | SIO4_READ_INT_STATUS   |
| arg                     | SIO4_INTERRUPT_STATUS* |

#### 7.4.20. SIO4\_READ\_REGISTER

This service reads the value of an SIO4 register. This includes all PCI registers, all PLX feature set registers, and all GSC firmware registers for the referenced channel. Refer to the SIO4 User Manual and to `sio4.h` for a complete list of the available registers.

#### Usage

| <b>ioctl()</b> Argument | Description        |
|-------------------------|--------------------|
| request                 | SIO4_READ_REGISTER |
| arg                     | Sio4_reg_t*        |

#### 7.4.21. SIO4\_READ\_REGISTER\_RAW

This service reads the value of an SIO4 firmware register without respect to the channel being accessed. This applies to firmware registers only. Permissible values are from zero to 63. All other values result in failure. Refer to the SIO4 User Manual and to `sio4.h` for a complete list of the predefined register identifiers.

#### Usage

| <b>ioctl()</b> Argument | Description            |
|-------------------------|------------------------|
| request                 | SIO4_READ_REGISTER_RAW |
| arg                     | Sio4_reg_t*            |

#### 7.4.22. SIO4\_RESET\_CHANNEL

This service performs a reset of the entire channel. This includes the transmitter, the receiver, the FIFOs, the cable configuration, the transceivers and the programmable oscillator. (The programmable oscillator is reset only if the SIO4 supports a different programmable source for each channel.)

#### Usage

| <b>ioctl()</b> Argument | Description        |
|-------------------------|--------------------|
| request                 | SIO4_RESET_CHANNEL |
| arg                     | Not used.          |

#### 7.4.23. SIO4\_RESET\_DEVICE

This service resets all of the board's hardware. This includes the transmitter, the receiver, the FIFOs, the cable configurations, the transceivers and the programmable oscillators. The programmable transceivers and programmable oscillators are disabled, if supported in hardware.

**WARNING:** This service affects all channels on the board and should be used with care.

Usage

| <b>ioctl( ) Argument</b> | <b>Description</b> |
|--------------------------|--------------------|
| request                  | SIO4_RESET_DEVICE  |
| arg                      | Not used.          |

#### 7.4.24. SIO4\_RESET\_FIFO

This service resets either or both of the channel FIFOs.

Usage

| <b>ioctl( ) Argument</b> | <b>Description</b> |
|--------------------------|--------------------|
| request                  | SIO4_RESET_FIFO    |
| arg                      | TX_RX*             |

#### 7.4.25. SIO4\_RX\_FIFO\_AE\_CONFIG

This service configures the Rx FIFO Almost Empty level and reports the current level. When applying a setting, the Rx FIFO is reset and the current content is lost. If the XXX\_READ macro is supplied then no change is applied. Before returning the current programmed level is obtained and supplied to the caller.

Usage

| <b>ioctl( ) Argument</b> | <b>Description</b>     |
|--------------------------|------------------------|
| request                  | SIO4_RX_FIFO_AE_CONFIG |
| arg                      | __s32*                 |

#### 7.4.26. SIO4\_RX\_FIFO\_AF\_CONFIG

This service configures the Rx FIFO Almost Full level and reports the current level. When applying a setting, the Rx FIFO is reset and the current content is lost. If the XXX\_READ macro is supplied then no change is applied. Before returning the current programmed level is obtained and supplied to the caller.

Usage

| <b>ioctl( ) Argument</b> | <b>Description</b>     |
|--------------------------|------------------------|
| request                  | SIO4_RX_FIFO_AF_CONFIG |
| arg                      | __s32*                 |

#### 7.4.27. SIO4\_RX\_FIFO\_COUNT

This service retrieves the current Rx FIFO fill level. The value obtained is either the number of bytes of data in the Rx FIFO or the XXX\_UNKNOWN macro if the Rx FIFO Count Register is unsupported.

Usage

| <b>ioctl( ) Argument</b> | <b>Description</b> |
|--------------------------|--------------------|
| request                  | SIO4_RX_FIFO_COUNT |
| arg                      | __s32*             |

The value returned is from zero to the size of the FIFO or the value given below.

| Macros                  | Description                     |
|-------------------------|---------------------------------|
| SIO4_FIFO_COUNT_UNKNOWN | The FIFO fill level is unknown. |

#### 7.4.28. SIO4\_RX\_FIFO\_SIZE

This service retrieves the size of the Rx FIFO. The value obtained is either the capacity of the Rx FIFO in bytes or the XXX\_UNKNOWN macro if the Rx FIFO Size Register is unsupported.

Usage

| ioctl() Argument | Description       |
|------------------|-------------------|
| request          | SIO4_RX_FIFO_SIZE |
| arg              | __s32*            |

The table below lists the predefined macros used with this service.

| Macros                 | Description               |
|------------------------|---------------------------|
| SIO4_FIFO_SIZE_UNKNOWN | The FIFO size is unknown. |

#### 7.4.29. SIO4\_RX\_IO\_ABORT

This service aborts a read() operation. This service waits for up to 10 seconds to abort either a currently active read() operation or one that is initiated during the abort waiting period.

Usage

| ioctl() Argument | Description      |
|------------------|------------------|
| request          | SIO4_RX_IO_ABORT |
| arg              | __s32*           |

The table below lists the options used with this service.

| Macros | Description                  |
|--------|------------------------------|
| 0      | An abort did not take place. |
| 1      | An abort did take place.     |

#### 7.4.30. SIO4\_RX\_IO\_MODE\_CONFIG

This service updates and reports the mode used by the driver for data read operations. This refers to how data is moved from the SIO4 to host memory when the read() function is called.

Usage

| ioctl() Argument | Description            |
|------------------|------------------------|
| request          | SIO4_RX_IO_MODE_CONFIG |
| arg              | __s32*                 |

The table below lists the options used with this service.

| Macros               | Description                                        |
|----------------------|----------------------------------------------------|
| SIO4_IO_MODE_DEFAULT | This refers to the default I/O mode, which is PIO. |

|                    |                                                                                        |
|--------------------|----------------------------------------------------------------------------------------|
| SIO4_IO_MODE_DMA   | This refers to DMA, which is generally performed without regard to the FIFO's content. |
| SIO4_IO_MODE_DMDMA | This refers to Demand Mode DMA, which transfers data as it becomes available.          |
| SIO4_IO_MODE_PIO   | This refers to PIO, which uses repetitive register accesses.                           |
| SIO4_IO_MODE_READ  | This is used to retrieve the current configuration.                                    |

#### 7.4.31. SIO4\_SET\_READ\_TIMEOUT

This service sets the timeout limit for read requests, and is the maximum amount of time the driver will wait for a blocking `read()` request to complete. The timeout period is specified in seconds. Timeout values of zero (0) or less mean do not wait.

Usage

| <b>ioctl() Argument</b> | <b>Description</b>    |
|-------------------------|-----------------------|
| request                 | SIO4_SET_READ_TIMEOUT |
| arg                     | __u32                 |

#### 7.4.32. SIO4\_SET\_WRITE\_TIMEOUT

This service sets the timeout limit for write requests, and is the maximum amount of time the driver will wait for a blocking `write()` request to complete. The timeout period is specified in seconds. Timeout values of zero (0) or less mean do not wait.

Usage

| <b>ioctl() Argument</b> | <b>Description</b>     |
|-------------------------|------------------------|
| request                 | SIO4_SET_WRITE_TIMEOUT |
| arg                     | __u32                  |

#### 7.4.33. SIO4\_TX\_FIFO\_AE\_CONFIG

This service configures the Tx FIFO Almost Empty level and reports the current level. When applying a setting, the Tx FIFO is reset and the current content is lost. If the `XXX_READ` macro is supplied then no change is applied. Before returning the current programmed level is obtained and supplied to the caller.

Usage

| <b>ioctl() Argument</b> | <b>Description</b>     |
|-------------------------|------------------------|
| request                 | SIO4_TX_FIFO_AE_CONFIG |
| arg                     | __s32*                 |

#### 7.4.34. SIO4\_TX\_FIFO\_AF\_CONFIG

This service configures the Tx FIFO Almost Full level and reports the current level. When applying a setting, the Tx FIFO is reset and the current content is lost. If the `XXX_READ` macro is supplied then no change is applied. Before returning the current programmed level is obtained and supplied to the caller.

Usage

| <b>ioctl() Argument</b> | <b>Description</b>     |
|-------------------------|------------------------|
| request                 | SIO4_TX_FIFO_AF_CONFIG |
| arg                     | __s32*                 |



**7.4.35. SIO4\_TX\_FIFO\_COUNT**

This service retrieves the current Tx FIFO fill level. The value obtained is either the number of bytes of data in the Tx FIFO or the XXX\_UNKNOWN macro if the Tx FIFO Count Register is unsupported.

**Usage**

| <b>ioctl() Argument</b> | <b>Description</b> |
|-------------------------|--------------------|
| request                 | SIO4_TX_FIFO_COUNT |
| arg                     | __s32*             |

The value returned is from zero to the size of the FIFO or the value given below.

| <b>Macros</b>           | <b>Description</b>              |
|-------------------------|---------------------------------|
| SIO4_FIFO_COUNT_UNKNOWN | The FIFO fill level is unknown. |

**7.4.36. SIO4\_TX\_FIFO\_SIZE**

This service retrieves the size of the Tx FIFO. The value obtained is either the capacity of the Tx FIFO in bytes or the XXX\_UNKNOWN macro if the Tx FIFO Size Register is unsupported.

**Usage**

| <b>ioctl() Argument</b> | <b>Description</b> |
|-------------------------|--------------------|
| request                 | SIO4_TX_FIFO_SIZE  |
| arg                     | __s32*             |

The table below lists the predefined macros used with this service.

| <b>Macros</b>          | <b>Description</b>        |
|------------------------|---------------------------|
| SIO4_FIFO_SIZE_UNKNOWN | The FIFO size is unknown. |

**7.4.37. SIO4\_TX\_IO\_ABORT**

This service aborts a write() operation. This service waits for up to 10 seconds to abort either a currently active write() operation or one that is initiated during the abort waiting period.

**Usage**

| <b>ioctl() Argument</b> | <b>Description</b> |
|-------------------------|--------------------|
| request                 | SIO4_TX_IO_ABORT   |
| arg                     | __s32*             |

The table below lists the options used with this service.

| <b>Macros</b> | <b>Description</b>           |
|---------------|------------------------------|
| 0             | An abort did not take place. |
| 1             | An abort did take place.     |

**7.4.38. SIO4\_TX\_IO\_MODE\_CONFIG**

This service updates and reports the mode used by the driver for data write operations. This refers to how data is moved from host memory to the SIO4 when the write() function is called.

## Usage

| <b>ioctl( ) Argument</b> | <b>Description</b>     |
|--------------------------|------------------------|
| request                  | SIO4_TX_IO_MODE_CONFIG |
| arg                      | __s32*                 |

The table below lists the options used with this service.

| <b>Macros</b>        | <b>Description</b>                                                                     |
|----------------------|----------------------------------------------------------------------------------------|
| SIO4_IO_MODE_DEFAULT | This refers to the default I/O mode, which is PIO.                                     |
| SIO4_IO_MODE_DMA     | This refers to DMA, which is generally performed without regard to the FIFO's content. |
| SIO4_IO_MODE_DMDMA   | This refers to Demand Mode DMA, which transfers data as space becomes available.       |
| SIO4_IO_MODE_PIO     | This refers to PIO, which uses repetitive register accesses.                           |
| SIO4_IO_MODE_READ    | This is used to retrieve the current configuration.                                    |

**7.4.39. SIO4\_WRITE\_REGISTER**

This service writes a value to an SIO4 register. This includes GSC firmware registers and the USC registers only. All PCI and PLX feature set registers are read-only. Refer to the SIO4 User Manual and to `sio4.h` for a complete list of available registers. Applications should exercise care in writing to some of these registers. This is because some are used by the driver for interrupt and DMA purposes. Writing to these registers may interfere with proper SIO4 and driver operation and may disrupt the stability of the operating system. The registers of concern are those listed below.

- The GSC Board Control Register
- The GSC Interrupt Control Register (and the interrupt configuration registers)
- The GSC Interrupt Status Register

## Usage

| <b>ioctl( ) Argument</b> | <b>Description</b>  |
|--------------------------|---------------------|
| request                  | SIO4_WRITE_REGISTER |
| arg                      | Sio4_reg_t*         |

## 8. Operation

This section explains some operational procedures on using the driver. This is in no way intended to be a comprehensive guide on using the SIO4. This is simply to address a very few issues relating to GSC specific features of the SIO4.

### 8.1. I/O Modes

The following describes the three supported I/O modes used for data transfer between the host and the SIO4. All three modes are available using the C library routines `read()` and `write()`. Applications select the desired mode using IOCTL services. Use the `SIO4_TX_IO_MODE_CONFIG` IOCTL service to configure the `write()` data transfer mode and use the `SIO4_RX_IO_MODE_CONFIG` IOCTL service to configure the `read()` data transfer mode.

#### 8.1.1. DMDMA

This refers to Demand Mode DMA. This mode transfers data with the least amount of CPU overhead. It accommodates transfers that exceed the size of the installed FIFOs and uses the FIFO fill level to throttle data movement over the PCI bus. This permits efficient data movement over the PCI bus and also permits the transfer to remain active while data is being transferred over the cable interface. Since the SIO4 can have up to eight data streams (4 Rx and 4 Tx) and only two DMA engines are available, applications must make selective use of DMA and non-DMA I/O requests. Applications can make DMDMA mode I/O requests without having to monitor FIFO fill levels.

#### 8.1.2. DMA

This refers to Non-Demand Mode DMA. This mode transfers data with little CPU overhead, but is suitable only for requests that do not exceed the size of the installed FIFOs. Using this mode, applications must monitor a FIFO's fill level to insure that it can accommodate desired requests. Calling `read()` when the Rx FIFO contains insufficient data will result in indeterminate data at the point where the FIFO runs empty. Calling `write()` when the Tx FIFO contains insufficient free space will result in data loss at the point the FIFO becomes full. Since the SIO4 can have up to eight data streams (4 Rx and 4 Tx) and only two DMA engines are available, applications must make selective use of DMA and non-DMA I/O requests.

#### 8.1.3. PIO

This mode uses repetitive register accesses. While it is the least efficient method it accommodates simultaneous transfers on any number of channels and in both directions. Applications can make PIO mode I/O requests without having to monitor FIFO fill levels.

## 8.2. Oscillator Programming

The ability to program the SIO4's onboard oscillators depends on the board's hardware capabilities and on support included in the driver. The driver can identify the oscillator chip for all SIO4 implementations up to and including those using the Cypress CY22393 Programmable Oscillator. At present however, the driver includes built-in programming support only for those SIO4s using a single CY22393. The driver will return an error status when exercising the programmable oscillator features for all other programmable oscillator types. The general procedure to follow when using the programmable oscillator features are as follows.

**NOTE:** The driver measures the SIO4's reference frequency when the driver is first loaded. If it cannot be measured, then it is initialized to 20MHz. Thereafter, the reference frequency is changed only when done explicitly by application requests using the `SIO4_OSC_REFERENCE` IOCTL service.

1. Determine if the driver is able to perform oscillator programming for the device. This can be done using the `SIO4_FEATURE_TEST` IOCTL service on the `SIO4_FEATURE_OSC_PROGRAM` feature. If the feature is unsupported, then do not attempt programming. Attempting to use the driver's built-in programming features will be unsuccessful when this feature is unsupported. If the feature is supported, then continue with the following steps.
2. Tell the driver the SIO4's reference frequency. This is done using the `SIO4_OSC_REFERENCE` IOCTL service. The specified reference frequency is applicable to all channels since the SIO4 has only a single reference oscillator. The specified reference frequency is used for subsequent operations only.
3. Reset the channel's clock. This is done using the `SIO4_OSC_RESET` IOCTL service. Depending on the oscillator, this may disable the channel's clock. Depending on the SIO4, this effort may affect all channels.
4. Initialize the channel's clock. This is done using the `SIO4_OSC_INIT` IOCTL service. Depending on the oscillator, this should configure the channel to output the reference frequency. Depending on the SIO4, this effort may affect all channels.
5. Request that the oscillator be reprogrammed for the desired frequency. This is done using the `SIO4_OSC_PROGRAM` IOCTL service. The resulting frequency will be as close as possible to the requested frequency. How close this actually is depends on the oscillator's capabilities, its current resource usage and the reference frequency. Check the `sio4_osc_t` structure's `freq_got` field after programming to verify that the resulting frequency is sufficient. Depending on the SIO4, the programming effort may affect all channels.

**NOTE:** On occasion, the oscillator programming effort may not take full affect even though the operation completes successfully. Applications should therefore measure the oscillator frequency following programming requests. If the measured results differ significantly from what the programming request indicated would be produced, then repeat the programming and measurement steps until the results are satisfactory.

6. If desired, the channel's current frequency can be measured at any time using the `SIO4_OSC_MEASURE` IOCTL service. However, this should only be done if the frequency can be measured. This capability depends on the SIO4's feature set. Support for this feature can be determined by using the `SIO4_FEATURE_TEST` IOCTL service with the `SIO4_FEATURE_OSC_MEASURE` feature argument.
7. If desired, the current configuration may be determined at any time using the `SIO4_OSC_INFO` IOCTL service. The information returned will be based on the driver's recorded state information.

### 8.2.1. Cypress CY22393 (1x) Programmable Oscillator Support

The SIO4's support for this device includes a fixed reference oscillator, a Cypress CY22393 (with four programmable oscillators), and four firmware based post dividers. The driver defaults the reference frequency to the measured frequency at startup and initializes the programmable oscillators to their off state. The driver manages the firmware post dividers and the CY22393, with its oscillators and Digital Phase Lock Loop Generators, as best as possible to fulfill application requests. When a programming request is made the driver applies the appropriate changes, measures the results, and reprograms the changes as necessary. The measurement and reprogramming steps occur when a channel is opened and closed, and when operations are requested by an application. The driver responds to the services according to the following table.

| Service                       | Response                                                                                             |
|-------------------------------|------------------------------------------------------------------------------------------------------|
| <code>SIO4_OSC_INFO</code>    | The current settings are reported.                                                                   |
| <code>SIO4_OSC_INIT</code>    | The desired frequency is set to the reference frequency and the channel is reconfigured accordingly. |
| <code>SIO4_OSC_MEASURE</code> | The output frequency is measured using SIO4 firmware resources. The measured                         |

|                                 |                                                                                                                                                                                                                                |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                 | value is reported in the <code>freq_got</code> field.                                                                                                                                                                          |
| <code>SIO4_OSC_PROGRAM</code>   | If the requested frequency is non-negative and 20MHz or less, then the driver programs in that configuration that will most closely match the request. This is done based on the CY22393's resources available at that moment. |
| <code>SIO4_OSC_REFERENCE</code> | The requested value is recorded if it is 8MHz or higher and 30MHz or lower.                                                                                                                                                    |
| <code>SIO4_OSC_RESET</code>     | The desired frequency is set to zero and the channel is reconfigured accordingly.                                                                                                                                              |

### 8.2.2. Cypress CY22393 (4x) Programmable Oscillator Support

The driver does not include support for this device configuration. The driver returns EIO for all programmable oscillator requests when the SIO4 uses this chip configuration.

### 8.2.3. Cypress IDC2053B Programmable Oscillator Support

The driver does not include support for this device. The driver returns EIO for all programmable oscillator requests when the SIO4 uses this chip.

### 8.2.4. Fixed Oscillator Support

When the SIO4 has a fixed oscillator, no programming can be performed. Rather than return errors though, the driver treats the hardware as a programmable oscillator capable only of supply the reference frequency. The driver responds to the IOCTL services according to the following table.

| Service                         | Response                                                                         |
|---------------------------------|----------------------------------------------------------------------------------|
| <code>SIO4_OSC_INFO</code>      | The current settings are reported.                                               |
| <code>SIO4_OSC_INIT</code>      | The <code>freq_got</code> value is updated to the reference frequency.           |
| <code>SIO4_OSC_MEASURE</code>   | The <code>freq_got</code> value is reported as -1 (due to firmware limitations). |
| <code>SIO4_OSC_PROGRAM</code>   | The requested value is recorded if it is non-zero and 20MHz or lower.            |
| <code>SIO4_OSC_REFERENCE</code> | The requested value is recorded if it is 1MHz or higher and 20MHz or lower.      |
| <code>SIO4_OSC_RESET</code>     | The <code>freq_got</code> value is updated to the reference frequency.           |

### 8.2.5. All Other Cases

This applies when the SIO4 includes no programmable oscillator support and when the SIO4 uses a programmable oscillator unrecognized by the driver. The driver responds to the IOCTL services according to the following table.

| Service                         | Response                                                                                                                                  |
|---------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>SIO4_OSC_INFO</code>      | The current recorded settings are reported.                                                                                               |
| <code>SIO4_OSC_INIT</code>      | The recorded <code>freq_want</code> and <code>freq_got</code> values are set to the reference frequency.                                  |
| <code>SIO4_OSC_MEASURE</code>   | The <code>freq_got</code> value is reported as zero.                                                                                      |
| <code>SIO4_OSC_PROGRAM</code>   | The recorded <code>freq_want</code> and <code>freq_got</code> values are set to the requested value if it is non-zero and 20MHz or lower. |
| <code>SIO4_OSC_REFERENCE</code> | The requested value is recorded if it is 1MHz or higher and 20MHz or lower.                                                               |
| <code>SIO4_OSC_RESET</code>     | The recorded <code>freq_want</code> and <code>freq_got</code> values are set to zero.                                                     |

## 8.3. Multi-Protocol Transceiver Programming

This feature includes boards with varying capabilities. Some boards are able to change the transceiver protocol under software control. Some have fixed transceiver protocols and can report the protocol via firmware. Others have fixed transceiver protocols, but are not able to report the protocol. The general procedure to follow when using this feature is as follows.

1. Determine if the SIO4 supports this feature. This can be done using the SIO4\_FEATURE\_TEST IOCTL service on the SIO4\_FEATURE\_MP feature. If this feature is unsupported, then do not attempt to exercise the board's Multi-Protocol transceiver feature. Attempting to do so will be unsuccessful when this feature is unsupported. If the feature is supported, then continue with the following steps.
2. Determine if the SIO4's transceiver protocol can be changed. This can be done using the SIO4\_FEATURE\_TEST IOCTL service on the SIO4\_FEATURE\_MP\_CHANGE feature. If this feature is unsupported, then do not attempt to exercise the board's Multi-Protocol transceiver feature. Attempting to do so will be unsuccessful when this feature is unsupported. If the feature is supported, then continue with the following steps.
3. Determine if the transceiver protocol desired is supported. This can be done using the SIO4\_MP\_TEST IOCTL. If a suitable protocol cannot be selected, then do not attempt to further exercise the board's Multi-Protocol transceiver feature. If a suitable protocol is available, then continue with the following steps.
4. Select a suitable transceiver protocol. This can be done using the SIO4\_MP\_CONFIG IOCTL.
5. If desired, the current configuration can be determined at any time using the SIO4\_OSC\_INFO IOCTL service.

### 8.3.1. Cipex SP508 Multi-Protocol Transceiver Support

When the SIO4 includes these transceiver chips, the driver responds to the services according to the following table.

| Service        | Response                                                                                                                                                                              |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SIO4_MP_CONFIG | The chip will be given as the SP508 option. The resulting protocol will equal the requested protocol if it is supported. The resulting protocol will otherwise be the invalid option. |
| SIO4_MP_INFO   | The chip will be given as the SP508 option. The desired protocol will be the read option. The resulting protocol will reflect the board's current configuration.                      |
| SIO4_MP_INIT   | The chip will be given as the SP508 option. The desired and resulting protocol will both be the RS-422/485 option.                                                                    |
| SIO4_MP_RESET  | The chip will be given as the SP508 option. The desired and resulting protocol will both be the disable option.                                                                       |
| SIO4_MP_TEST   | The chip will be given as the SP508 option. The resulting protocol will be the requested protocol if it is supported. The resulting protocol will otherwise be the invalid option.    |

### 8.3.2. Fixed Protocol Support

Some SIO4s include Multi-Protocol support in firmware but not in hardware. This applies when the SIO4 has fixed transceivers whose type is reported by firmware. Under these circumstances the driver responds to the IOCTL services according to the following table.

| Service        | Response                                                                                                                                                                                           |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SIO4_MP_CONFIG | The chip will be given as the fixed option. The resulting protocol will reflect the board's hardwired protocol.                                                                                    |
| SIO4_MP_INFO   | The chip will be given as the fixed option. The desired protocol will be the read option and the resulting protocol will reflect the board's hardwired protocol.                                   |
| SIO4_MP_INIT   | The chip will be given as the fixed option. The desired and resulting protocols will reflect the board's hardwired protocol option.                                                                |
| SIO4_MP_RESET  | The chip will be given as the fixed option. The desired and resulting protocols will reflect the board's hardwired protocol.                                                                       |
| SIO4_MP_TEST   | The chip will be given as the fixed option. The resulting protocol will be the test protocol if it is the board's hardwired protocol. The resulting protocol will otherwise be the invalid option. |

### 8.3.3. All Other Cases

This applies when the firmware includes no Multi-Transceiver protocol support and when support is present but the protocol is fixed. In these cases the driver responds to the IOCTL services according to the following table.

| Service        | Response                                                                                                                                 |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------|
| SIO4_MP_CONFIG | The chip and resulting protocol will each be given as their respective unknown options.                                                  |
| SIO4_MP_INFO   | The desired protocol will be the read option. The chip and resulting protocol will each be given as their respective unknown options.    |
| SIO4_MP_INIT   | The chip, the desired protocol and resulting protocol will all be given as their respective unknown options.                             |
| SIO4_MP_RESET  | The desired protocol will be the disable option. The chip and resulting protocol will each be given as their respective unknown options. |
| SIO4_MP_TEST   | The chip and resulting protocol will each be given as their respective unknown options.                                                  |

## 8.4. Interrupt Notification

Applications can make indirect use of SIO4 interrupts by using the Interrupt Notification IOCTL services. This requires the following basic steps. These steps are illustrated in the source code sample that follows.

1. Use the `fcntl` interface to register the application's signal handler.
2. Issue the `SIO4_INT_NOTIFY` IOCTL service to request notification.
3. When the `SIGIO` signal is received, issue the `SIO4_READ_INT_STATUS` IOCTL service to determine which interrupt occurred.
4. Perform any application required actions.
5. If additional notification is required for an interrupt that was reported then repeat steps two through five as required.
6. When finished issue the `SIO4_INT_NOTIFY` IOCTL service with an argument value of zero (0) to specify that notification be terminated.

### Example

```
#include <errno.h>
#include <fcntl.h>
#include <signal.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/ioctl.h>

#include "SIO4DocSrcLib.h"

static int _fd;

static void handle_sigio(int signo)
{
 SIO4_INTERRUPT_STATUS int_stat;
 int status;

 status = ioctl(_fd, SIO4_READ_INT_STATUS, &int_stat);
```

```

 if (status == -1)
 {
 /* The request failed. */
 }
 else if (int_stat.u8SIO4Status & SIO4_INT_NOTIFY_TX_FIFO_AE)
 {
 /* Handle the Tx FIFO Almost Empty condition. */
 }
}

int sio4_async_setup(int fd)
{
 int flags;
 unsigned char notify;
 pid_t pid;
 int status;

 ioctl(fd, SIO4_INT_NOTIFY, 0);
 _fd = fd;
 signal(SIGIO, handle_sigio);
 pid = getpid();
 fcntl(fd, F_SETOWN, pid);
 flags = fcntl(fd, F_GETFL);
 flags |= FASYNC;
 fcntl(fd, F_SETFL, flags);
 notify = SIO4_INT_NOTIFY_TX_FIFO_AE;
 status = ioctl(fd, SIO4_INT_NOTIFY, notify);
 return(status);
}

```



## Document History

| Revision           | Description                                                                                                                                                                                                                                                                                                                           |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| January 16, 2012   | Updated to release version 1.42.0. Modified the title page model number. Updated the kernel support table. Updated the compiler support information.                                                                                                                                                                                  |
| August 19, 2011    | Updated to release version 1.41.0.                                                                                                                                                                                                                                                                                                    |
| August 11, 2011    | Updated to release version 1.40.1.                                                                                                                                                                                                                                                                                                    |
| June 17, 2011      | Updated to release version 1.40.0.                                                                                                                                                                                                                                                                                                    |
| March 2, 2011      | Updated to release version 1.39.0.                                                                                                                                                                                                                                                                                                    |
| March 1, 2011      | Updated to release version 1.38.1. Removed app4. Updated some version notes. Renamed the app1 application to <code>sync_test</code> . Renamed the app2 application to <code>sync2c</code> .                                                                                                                                           |
| December 11, 2010  | Updated to release version 1.38.0. Various editorial changes.                                                                                                                                                                                                                                                                         |
| November 22, 2010  | Updated to release version 1.37.0. Removed termination configuration from the <code>sio4_sync_t</code> structure. Removed all items and services relating to a FIFO's type. Added several Feature Test IOCTL options. Removed the Read FIFO Status IOCTL service. Removed the app3 sample application and added <code>sbtest</code> . |
| July 27, 2010      | Updated to release version 1.36.0. Updated the CPU and Kernel Support information.                                                                                                                                                                                                                                                    |
| June 10, 2010      | Updated to release version 1.35.0.                                                                                                                                                                                                                                                                                                    |
| March 18, 2010     | Updated to release version 1.33.0.                                                                                                                                                                                                                                                                                                    |
| February 18, 2010  | Updated to release version 1.32.1.                                                                                                                                                                                                                                                                                                    |
| February 13, 2010  | Updated to release version 1.32.0.                                                                                                                                                                                                                                                                                                    |
| January 25, 2010   | Updated to release version 1.31.0. Added the <code>id</code> sample application.                                                                                                                                                                                                                                                      |
| December 18, 2009  | Updated to release version 1.30.0. Added information regarding the SIO4BXR programmable oscillator feature.                                                                                                                                                                                                                           |
| November 12, 2009  | Updated to release version 1.29.0.                                                                                                                                                                                                                                                                                                    |
| September 19, 2009 | Updated to release version 1.28.0. Updated kernel support list.                                                                                                                                                                                                                                                                       |
| September 11, 2009 | Updated to release version 1.27.0. Updated kernel support list.                                                                                                                                                                                                                                                                       |
| August 23, 2009    | Updated to release version 1.26.0. Updated kernel support list. Renamed overall make script. Renamed the driver startup script.                                                                                                                                                                                                       |
| June 2, 2009       | Updated to release version 1.25.1.                                                                                                                                                                                                                                                                                                    |
| March 7, 2009      | Updated to release version 1.25.0.                                                                                                                                                                                                                                                                                                    |
| February 21, 2009  | Updated to release version 1.24.0. Added the sample application <code>sync/txrate</code> . Reorganized the installed files sections. Added the <code>SIO4_FEATURE_FW_PD_BITS</code> feature test option.                                                                                                                              |
| June 25, 2008      | Updated to release version 1.23.0. Corrected the names of some IOCTL macros. The accumulated interrupt status is no longer cleared when a new notification request is made. Added information on I/O interrupt usage. Additional kernel porting.                                                                                      |
| March 29, 2007     | Updated to release version 1.22.0. Notes were added for oscillator programming changes applicable to programmable oscillator models.                                                                                                                                                                                                  |
| August 25, 2006    | Updated to release version 1.21.0. List specific 2.2, 2.4, 2.6 and 32/64-bit kernels tested.                                                                                                                                                                                                                                          |
| August 8, 2006     | Updated to release version 1.20.0. Added driver updates.                                                                                                                                                                                                                                                                              |
| January 30, 2006   | Updated to release version 1.19.2. Added an overall make script. Altered the directory structure.                                                                                                                                                                                                                                     |
| January 25, 2006   | Updated to release version 1.19.1.                                                                                                                                                                                                                                                                                                    |
| December 19, 2005  | Updated to release version 1.19.0.                                                                                                                                                                                                                                                                                                    |
| October 4, 2005    | Updated to release version 1.18.3.                                                                                                                                                                                                                                                                                                    |
| September 30, 2005 | Updated to release version 1.18.2.                                                                                                                                                                                                                                                                                                    |
| September 26, 2005 | Updated to release version 1.18.1.                                                                                                                                                                                                                                                                                                    |
| July 15, 2005      | Updated to release version 1.18.0. Removed feature definitions that are no longer supported.                                                                                                                                                                                                                                          |
| May 24, 2005       | Updated to release version 1.17.1.                                                                                                                                                                                                                                                                                                    |
| May 19, 2005       | Updated to release version 1.17.0.                                                                                                                                                                                                                                                                                                    |
| May 10, 2005       | Updated to release version 1.16.0. Corrected timeout information. Corrected remarks about                                                                                                                                                                                                                                             |

|                  |                                                                                           |
|------------------|-------------------------------------------------------------------------------------------|
|                  | the <code>sio4_sync_tx_t.clock.idle</code> bit. Added new feature options.                |
| April 5, 2005    | Updated to release version 1.15.1.                                                        |
| March 23, 2005   | Updated to release version 1.15.0.                                                        |
| January 25, 2005 | Updated to release version 1.14.0.                                                        |
| January 24, 2005 | Updated to the driver to support the 2.6 kernel.                                          |
| November 3, 2004 | Updated to release version 1.12.1.                                                        |
| November 2, 2004 | Updated to release version 1.12.0.                                                        |
| October 18, 2004 | Updated to release version 1.11.0.                                                        |
| August 30, 2004  | Updated to release version 1.10.0.                                                        |
| August 18, 2004  | Updated to release version 1.09.0. Updated documentation on some init and reset services. |
| August 17, 2004  | Updated to release version 1.08.0. Fixed driver <code>SIO4_INIT_CHANNEL</code> bug.       |
| August 11, 2004  | Updated to release version 1.07.2. Changed UART references to USC.                        |
| August 10, 2004  | Updated to release version 1.07.1. Added information on supported devices.                |
| August 9, 2004   | Updated to release version 1.07.0. Initial driver release.                                |