

SIO4/8

Four/Eight Channel High Speed Serial I/O

**All SIO4 and SIO8 Models
All Form Factors
All Standard SYNC Versions**

SYNC Protocol Library Reference Manual

Manual Revision: March 19, 2024

**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 © 2012-2024, **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: sales@generalstandards.com

General Standards Corporation makes no warranty of any kind with regard to this documentation and/or software, 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, **General Standards Corporation** assumes no responsibility for any errors, inaccuracies or omissions herein. This documentation, information and software are made available solely on an “as-is” basis. Nor is there any commitment to update or keep current this documentation.

General Standards Corporation does not assume any liability arising out of the application or use of documentation, software, 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 documentation, software or product, to improve accuracy, clarity, reliability, performance, function, or design.

ALL RIGHTS RESERVED.

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

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

Zilog and Z16C30 are trademarks of Zilog, Inc.

Table of Contents

1. Introduction.....	6
1.1. Purpose	6
1.2. Acronyms	6
1.3. Definitions.....	6
1.4. Software Overview	6
1.5. Hardware Overview.....	6
1.6. Reference Material.....	7
2. The SYNC Serial Protocol.....	8
2.1. Description.....	8
3. Library Interface Files	9
3.1. Header File	9
3.2. Static Library Files	9
4. Library Interface	10
4.1. Function Calls	10
4.1.1. sio4_sync_close()	10
4.1.2. sio4_sync_get().....	10
4.1.3. sio4_sync_gpio_rx()	10
4.1.4. sio4_sync_gpio_tx()	11
4.1.5. sio4_sync_init()	11
4.1.6. sio4_sync_ioctl()	11
4.1.7. sio4_sync_open().....	12
4.1.8. sio4_sync_read().....	12
4.1.9. sio4_sync_rx_get()	13
4.1.10. sio4_sync_rx_set().....	13
4.1.11. sio4_sync_set()	13
4.1.12. sio4_sync_tx_get()	14
4.1.13. sio4_sync_tx_set().....	14
4.1.14. sio4_sync_write()	14
4.2. Data Structures	15
4.2.1. sio4_sync_t.....	15
4.2.2. sio4_sync_rx_t.....	15
4.2.3. sio4_sync_tx_t.....	17
5. Operating Information	21
5.1. Basic Illustration	21
5.2. Getting Started.....	22
5.2.1. Bit Rates vs Oscillator Frequency	22
5.2.2. Cable Validation.....	22
5.2.3. Customizing the Configuration.....	22
5.3. Debugging Aids	23
5.3.1. Device Identification	23
5.3.2. sio4_reg_list().....	23

5.4. Cable Configuration Modes	24
5.4.1. DCE/DTE Mode	24
5.4.2. Legacy Mode	24
Document History	25

Table of Figures

Figure 1 A depiction of a SYNC data stream using a three signal configuration.	8
Figure 2 A depiction of a SYNC data stream using a two signal (Gated Clock) configuration.	8
Figure 3 A functional illustration of an SIO4B-SYNC or later model board.	21

1. Introduction

This document provides information on the SYNC Protocol Library, which is a library designed to facilitate use of the -SYNC model SIO4 and SIO8 boards.

1.1. Purpose

The purpose of this document is twofold. First, it is intended to give a basic description of the SIO4 SYNC protocol. Second, it is intended to give a complete description of the SYNC Protocol Library interface.

1.2. Acronyms

The following is a list of commonly occurring acronyms which may appear throughout this document.

Acronyms	Description
DMA	Direct Memory Access
GSC	General Standards Corporation
PCI	Peripheral Component Interconnect
PMC	PCI Mezzanine Card

1.3. Definitions

The following is a list of commonly occurring terms which may appear throughout this document.

Term	Definition
Application	Application means the user mode process, which runs in user space with user mode privileges.
Driver	Driver means the executable providing the direct access to the SIO4 hardware.
Library	Depending on context, this is a general reference to the SYNC Protocol Library.
SIO4	This is used as a general reference to any -SYNC model board supported by this driver. This includes both SIO4 and SIO8 model boards. It is also used to refer to revisions of the board that do not include a suffix following the '4', such as SIO4A or SIO4B.
SYNC	This is a general reference to the SYNC model SIO4/SIO8 boards, the serial protocol used by those boards or the library itself.

1.4. Software Overview

The SYNC Protocol Library is a statically linked library providing a SYNC centric interface to the SIO4 device driver. The library is provided in source form and must be built before being used. The library is a thin software layer that sits between an SIO4 application and the SIO4 API Library. The interface provided by the library is SYNC specific and is a simplified rendition of the IOCTL services that are part of the overall driver interface. The library exists in parallel with the driver interface.

1.5. Hardware Overview

NOTE: The SIO8 boards appear to the system as two SIO4 boards.

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. Once the data link between the two devices is established, the desired transfers can be performed and will become transparent to the user. The SIO4 board includes two DMA controllers and comes with a maximum of 256K Bytes of FIFO storage, which is 32K per channel direction (32K * 2 * 4). Each DMA controller is capable of transferring data to and from host memory; whereas the FIFO help maintain continuous data transfer at the cable interface. 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 comes with

transceivers that are fixed as RS232 or RS485/422, or with transceivers that are configurable. The SIO4 comes in two basic varieties; SYNC models or Zilog models, which are based on two Z16C30 dual USC chips. Later model SIO4 boards support both models with the mode being software controlled on a per channel basis. The SIO4 also provides for interrupt generation for various states of the board like Sync Character detection, FIFO empty, FIFO full and DMA complete.

NOTE: Software selection of SYNC or Zilog mode of operation is not at this time explicitly supported by the Protocol Libraries, the SIO4 API Library or the device driver. The operating mode is controlled by the model ordered.

1.6. Reference Material

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

- The *SIO4/SIO8 Driver Reference Manual* from General Standards Corporation.
- 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/>

- The *Z16C30 USC User's Manual* from Zilog. *
- The *Z16C30 Electronic Programmer's Manual* from Zilog (Zilog part number ZEPMDC00001). *

* The Zilog material is available from:

Zilog, Inc.
910 E Hamilton Ave
CAMPBELL, CA 95008 USA
Phone: 1-408-558-8500
WEB: <http://www.zilog.com/>

2. The SYNC Serial Protocol

2.1. Description

The SYNC serial communications protocol is a byte oriented serial transmission protocol consisting of data grouped into bytes whose bits are transmitted with the NRZ encoding. For successful data transfer the provider and recipient must only agree on the number of bits per byte and on the use of either two or three signals, Envelope, Clock and Data. In the three-wire configuration, Figure 1 below, the Envelope signal is asserted to indicate that the Clock and Data signals are valid. During this period data is clocked out on the Clock signal's rising edge and is clocked in on the Clock signal's falling edge. While the Envelope signal is negated both the clock and data signals ignored. The two-wire configuration, Figure 2 below, is also referred to as the Gated Clock configuration. Here, the Clock signal is active when the Data signal is valid and the Clock signal is idle when the Data signal is to be ignored. While the Clock signal is active data is clock out and in just as in the three-wire configuration. The SYNC protocol does not employ a mechanism that identifies boundaries between data bytes or message packets. It does not have any built-in error detection. And at has no built-in hardware flow control.

The SYNC transmitter can be configured to insert clock cycles in between consecutive bytes, but would otherwise omit such boundary indications. Any idle period between bytes appears incidentally based upon the available of the data in the channel's Tx Buffer. The line condition will go idle either when no more data is to be transmitted or when the SIO4 is waiting for additional transmit data. When the data bytes are eight bits or less the bits of each byte are transmitted consecutively without any idle period between successive bits. When the data bytes are nine bits or more the bits are transmitted consecutively without any idle period between successive bits unless there is a delay in providing data to the transmitter. When such delays arise, idle periods may appear at the cable interface between bits of a data byte. When such idle periods appear, they will be at eight-bit boundaries. (For example, only after the eighth bit, the 16th bit, the 24th bit and so on.) This can be minimized by preloading the Tx FIFO before starting transmission.

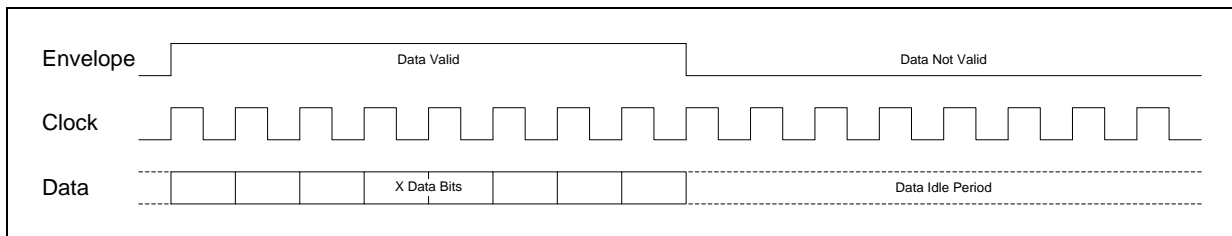


Figure 1 A depiction of a SYNC data stream using a three signal configuration.

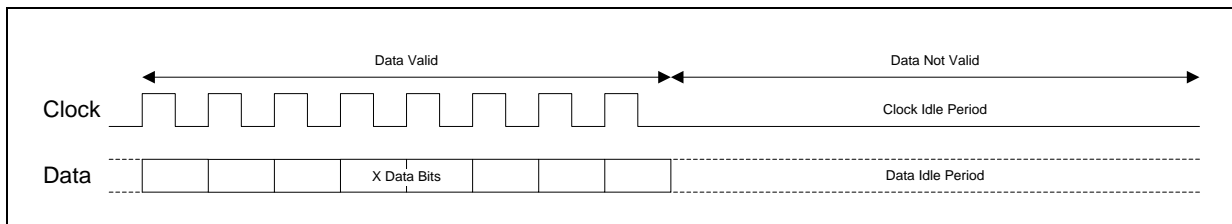


Figure 2 A depiction of a SYNC data stream using a two signal (Gated Clock) configuration.

3. Library Interface Files

This section gives general information on the SYNC Protocol Library interface files.

3.1. Header File

The library's interface is defined via the header file shown below. To use the SYNC Protocol Library applications must include this header file in their sources. Including this header file pulls in all other pertinent SIO4 specific header files. Therefore, sources may include only this one SIO4 header and make files may reference only this one SIO4 include directory.

File	Location
sio4_sync.h	.../sio4/include

3.2. Static Library Files

The executable code for the API defined for the SYNC Protocol Library is contained in the static library file `sio4_sync.a`, which is identified below. Using this library however, requires linking in other SIO4 specific static libraries. For this reason, and for ease of use, it is recommended that application make files link in the SIO4 Main Library instead of the SYNC static library along with all of its dependencies. The result is that application make files reference only a single SIO4 static library and only a single SIO4 static library path.

Library	File	Location
SYNC Protocol Library	sio4_sync.a	.../sio4/lib
SIO4 Main Library	sio4_main.a *	.../sio4/lib

* Refer to the SIO4 API Library Reference Manual for clarification when using multiple GSC product types in the same application.

4. Library Interface

The library interface is defined via the header file `sio4_sync.h`, which is located in the `.../sio4/include/` directory.

4.1. Function Calls

The library header defines the complete SYNC interface offered by the library, which includes the following high level function declarations.

4.1.1. `sio4_sync_close()`

This function is the entry point to close a connection previously opened to an SIO4 for SYNC operation.

Prototype

```
int sio4_sync_close(int fd);
```

Argument	Description
<code>fd</code>	This is a file descriptor obtained from <code>sio4_sync_open()</code> (section 4.1.7, page 12).

Return Value	Description
0	The operation completed successfully.
< 0	An error occurred. This is a negative <code>errno.h</code> value.

4.1.2. `sio4_sync_get()`

This function retrieves the SYNC settings that are independent of both the transmitter and the receiver.

Prototype

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

Argument	Description
<code>fd</code>	This is a file descriptor obtained from <code>sio4_sync_open()</code> (section 4.1.7, page 12).
<code>sync</code>	The settings requested are recorded here (section 4.2.1, page 15).

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

4.1.3. `sio4_sync_gpio_rx()`

This function retrieves the cable signal levels for those signals configured for GPIO operation.

Prototype

```
int sio4_sync_gpio_rx(int fd, u32* value);
```

Argument	Description
<code>fd</code>	This is a file descriptor obtained from <code>sio4_sync_open()</code> (section 4.1.7, page 12).
<code>value</code>	The value read is recorded here.

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

4.1.4. `sio4_sync_gpio_tx()`

This function applies an update to the cable signal levels for those signals configured for GPIO operation.

Prototype

```
int sio4_sync_gpio_tx(int fd, u32 value);
```

Argument	Description
<code>fd</code>	This is a file descriptor obtained from <code>sio4_sync_open()</code> (section 4.1.7, page 12).
<code>value</code>	This is the value to apply.

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

4.1.5. `sio4_sync_init()`

This function initializes the SYNC Protocol Library and must be the first call into the library.

NOTE: This service initializes the SYNC Protocol Library as well as the SIO4 API Library.

NOTE: This function may be called more than once, but only the first successful call initializes the library. Any subsequent call has no effect.

Prototype

```
int sio4_sync_init(void);
```

Return Value	Description
0	The operation completed successfully.
< 0	An error occurred. This is a negative <code>errno.h</code> value.

4.1.6. `sio4_sync_ioctl()`

This function is the entry point to performing IOCTL operations on the device. Refer to the driver reference manual for complete information on the driver's set of IOCTL services.

Prototype

```
int sio4_sync_ioctl(int fd, int request, void* arg);
```

Argument	Description
<code>fd</code>	This is a file descriptor obtained from <code>sio4_sync_open()</code> (section 4.1.7, page 12).
<code>request</code>	This is an IOCTL macro contained in <code>sio4.h</code> or <code>sio4 usc.h</code> .
<code>arg</code>	This is the argument type required for the above referenced IOCTL service.

Return Value	Description
0	The operation completed successfully.
< 0	An error occurred. This is a negative <code>errno.h</code> value.

4.1.7. sio4_sync_open()

This function is the entry point to open a connection to an SIO4 serial channel for SYNC operation. The handle returned by this call is used for all subsequent access to the specified channel. The file descriptor returned can be used for access the library's high-level functions, the library's low-level functions, and the driver interface.

NOTE: If the value of the `index` argument is specified as `-1`, then the function opens the file `/proc/sio4` for reading. In this case the `share` argument is ignored.

NOTE: If the value of the `index` argument is specified as `-1`, then the file descriptor returned can be used only with `sio4_sync_read()` (section 4.1.8, page 12) and `sio4_sync_close()` (section 4.1.1, page 10). The file descriptor cannot be used with any other services.

Prototype

```
int sio4_sync_open(int index, int share, int* fd);
```

Argument	Description						
<code>index</code>	This is the zero-based index of the SIO4 serial channel to access.						
<code>share</code>	Open the device in Shared Access Mode? If non-zero the device is opened in Shared Access Mode (see below). If zero the device is opened in Exclusive Access Mode (see below).						
<code>fd</code>	The device handle is returned here. The pointer cannot be NULL. Values returned are as follows. <table border="1"> <tr> <th>Value</th><th>Description</th></tr> <tr> <td><code>>= 0</code></td><td>This is the handle to use to access the device in subsequent calls.</td></tr> <tr> <td><code>-1</code></td><td>There was an error. The device is not accessible.</td></tr> </table>	Value	Description	<code>>= 0</code>	This is the handle to use to access the device in subsequent calls.	<code>-1</code>	There was an error. The device is not accessible.
Value	Description						
<code>>= 0</code>	This is the handle to use to access the device in subsequent calls.						
<code>-1</code>	There was an error. The device is not accessible.						

Return Value	Description
<code>0</code>	The operation completed successfully.
<code>< 0</code>	An error occurred. This is a negative <code>errno.h</code> value.

4.1.7.1. Access Modes

The value of the `share` argument determines the device access mode, as follows.

Shared Access Mode:

If the `share` argument is non-zero the device is opened in Shared Access Mode. The first such open request will succeed and return with the device in an initialized state. Subsequent such open requests will also succeed, but will not alter the device state. Once opened in Shared Access Mode, device access remains in this mode until all Shared Access Mode open requests release the device with a corresponding close request.

Exclusive Access Mode:

If the `share` argument is zero the device is opened in Exclusive Access Mode. In this mode, only one application at a time can access the device. The first such open request will succeed and return with the device in an initialized state. Subsequent open requests, regardless of the `share` argument value, will fail until the device is released with a corresponding close request.

4.1.8. sio4_sync_read()

This function requests that a buffer of data be read from the serial channel. The request will return either when it has been fulfilled or the read timeout expires, whichever occurs first. This is a blocking call.

NOTE: All read requests are serialized.

Prototype

```
int sio4_sync_read(int fd, void* buf, size_t bytes);
```

Argument	Description
fd	This is a file descriptor obtained from <code>sio4_sync_open()</code> (section 4.1.7, page 12).
buf	The data read is placed here.
bytes	Read up to this number of bytes.

Return Value	Description
0 to bytes	The operation succeeded. This is the number of bytes placed in the buffer. A value less than <code>bytes</code> generally indicates that the I/O timeout period lapsed before the entire request could be satisfied.
< 0	An error occurred. This is a negative <code>errno.h</code> value.

4.1.9. sio4_sync_rx_get()

This function retrieves the receiver specific SYNC setting.

Prototype

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

Argument	Description
fd	This is a file descriptor obtained from <code>sio4_sync_open()</code> (section 4.1.7, page 12).
rx	The settings are retrieved and are recorded here (section 4.2.2, page 15).

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

4.1.10. sio4_sync_rx_set()

This function applies receiver specific SYNC setting.

Prototype

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

Argument	Description
fd	This is a file descriptor obtained from <code>sio4_sync_open()</code> (section 4.1.7, page 12).
rx	The settings to apply are recorded here.

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

4.1.11. sio4_sync_set()

This function applies the SYNC settings that are independent of both the transmitter and the receiver.

Prototype

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

Argument	Description
fd	This is a file descriptor obtained from <code>sio4_sync_open()</code> (section 4.1.7, page 12).
sync	The settings to apply are recorded here (section 4.2.1, page 15).

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

4.1.12. `sio4_sync_tx_get()`

This function retrieves the transmitter specific SYNC setting.

Prototype

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

Argument	Description
fd	This is a file descriptor obtained from <code>sio4_sync_open()</code> (section 4.1.7, page 12).
tx	The settings are retrieved and are recorded here (section 4.2.3, page 17).

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

4.1.13. `sio4_sync_tx_set()`

This function applies transmitter specific SYNC setting.

Prototype

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

Argument	Description
fd	This is a file descriptor obtained from <code>sio4_sync_open()</code> (section 4.1.7, page 12).
tx	The settings to apply are recorded here (section 4.2.3, page 17).

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

4.1.14. `sio4_sync_write()`

This function requests that a buffer of data be written to the serial channel. The request will return either when it has been fulfilled or the write timeout expires, whichever occurs first. This is a blocking call.

NOTE: All write requests are serialized.

Prototype

```
int sio4_sync_write(int fd, const void* buf, size_t bytes);
```

Argument	Description
fd	This is a file descriptor obtained from <code>sio4_sync_open()</code> (section 4.1.7, page 12).
buf	This is the source for the data to write.
bytes	Write at most this number of bytes.

Return Value	Description
0 to bytes	The operation succeeded. This is the number of bytes written from the buffer. A value less than <code>bytes</code> generally indicates that the I/O timeout period lapsed before the entire request could be satisfied.
< 0	An error occurred. This is a negative <code>errno.h</code> value.

4.2. Data Structures

Including the library header (`sio4_sync.h`) in a source file gives that source the full library and driver interface. The library header defines the complete SYNC interface offered by the library, which includes the following data structures.

4.2.1. `sio4_sync_t`

This structure is used to define SYNC specific configuration items that are independent of the transmitter and the receiver.

Definition

```
typedef struct
{
    int      dce_enable;    // 0 = No, !0 = Yes (Precedence over DTE.)
    int      dte_enable;    // 0 = No, !0 = Yes (If DCE disabled.)

    struct
    {
        int  enable;        // 0 = No (disble), !0 = Yes (enable)
        int  internal;      // 0 = No (Extern), !0 = Yes (Intern)
    } lb;
} sio4_sync_t;
```

Fields	Description
<code>dce_enable</code>	If non-zero this enables DCE cable interface operation. *†‡
<code>dte_enable</code>	If non-zero this enables DTE cable interface operation. *†‡
<code>lb</code>	This structure contains loop back settings.
<code>lb.enable</code>	If non-zero this enables loop back operation.
<code>lb.internal</code>	If loopback is enabled this selects between internal and external loopback. If non-zero, the internal loopback is selected. If zero, then external loopback is selected.

* If both are selected, then DCE is selected.

† If neither is selected, then legacy cable mode operation becomes active, if it is supported.

‡ If either option is enabled, and supported, then legacy cable mode operation is disabled, if supported.

4.2.2. `sio4_sync_rx_t`

This structure is used to define receiver specific SYNC configuration items.

Definition

```

typedef struct
{
    int      enable;          // SIO4_IOCTL_SYNC_RX_ENABLE
    int      bit_order;       // SIO4_IOCTL_SYNC_RX_BIT_ORDER
    int      reset;           // 0 = no error/reset, !0 = error/reset
    ul6      word_size;       // SIO4_IOCTL_SYNC_RX_BIT_COUNT, READ-ONLY

    struct
    {
        s32   cfg;            // SIO4_IOCTL_SYNC_RXC_CFG
        s32   pol;            // SIO4_IOCTL_SYNC_RXC_POL
    } clock;

    struct
    {
        s32   cfg;            // SIO4_IOCTL_SYNC_RXE_CFG
        s32   pol;            // SIO4_IOCTL_SYNC_RXE_POL
    } env;

    struct
    {
        s32   cfg;            // SIO4_IOCTL_SYNC_RXD_CFG
        s32   idle;           // SIO4_IOCTL_SYNC_TXD_IDLE_CFG
        s32   legacy;         // SIO4_IOCTL_SYNC_LEG_RXD_CFG
    } data;
} sio4_sync_rx_t;

```

Fields	Description						
enable	<div>This field specifies the receiver enable state. Valid options are as given below.</div> <table><tr><th>Options</th><th>Description</th></tr><tr><td>SIO4_SYNC_ENABLE_NO</td><td>The receiver is disabled.</td></tr><tr><td>SIO4_SYNC_ENABLE_YES</td><td>The receiver is enabled.</td></tr></table>	Options	Description	SIO4_SYNC_ENABLE_NO	The receiver is disabled.	SIO4_SYNC_ENABLE_YES	The receiver is enabled.
Options	Description						
SIO4_SYNC_ENABLE_NO	The receiver is disabled.						
SIO4_SYNC_ENABLE_YES	The receiver is enabled.						
bit_order	<div>This field specifies the bit receive order. Valid options are as given below.</div> <table><tr><th>Options</th><th>Description</th></tr><tr><td>SIO4_SYNC_BIT_ORDER_LSB</td><td>Receive the least significant bit first.</td></tr><tr><td>SIO4_SYNC_BIT_ORDER_MSB</td><td>Receive the most significant bit first.</td></tr></table>	Options	Description	SIO4_SYNC_BIT_ORDER_LSB	Receive the least significant bit first.	SIO4_SYNC_BIT_ORDER_MSB	Receive the most significant bit first.
Options	Description						
SIO4_SYNC_BIT_ORDER_LSB	Receive the least significant bit first.						
SIO4_SYNC_BIT_ORDER_MSB	Receive the most significant bit first.						
reset	<div>SET: If non-zero, then any receive bit count error is cleared. If zero, then any receiver bit error is not cleared.</div> <div>GET: If zero, then no bit error occurred. If non-zero, then a bit error occurred.</div>						
word_size	<div>When the settings are retrieved, this is the number of bits in the most recently received data. The value will be from zero to 0xFFFF.</div>						
clock	<div>This structure contains Rx Clock settings.</div>						
clock. cfg	<div>This field specifies the basic clock signal configuration. Valid options are as given below.</div> <table><tr><th>Options</th><th>Description</th></tr><tr><td>SIO4_SYNC_RXC_CFG_FALL_EDGE</td><td>The receiver clocks data in on the clock's falling edge.</td></tr><tr><td>SIO4_SYNC_RXC_CFG_RISE_EDGE</td><td>The receiver clocks data in on the clock's rising edge.</td></tr></table>	Options	Description	SIO4_SYNC_RXC_CFG_FALL_EDGE	The receiver clocks data in on the clock's falling edge.	SIO4_SYNC_RXC_CFG_RISE_EDGE	The receiver clocks data in on the clock's rising edge.
Options	Description						
SIO4_SYNC_RXC_CFG_FALL_EDGE	The receiver clocks data in on the clock's falling edge.						
SIO4_SYNC_RXC_CFG_RISE_EDGE	The receiver clocks data in on the clock's rising edge.						

clock. pol	This field specifies the clock's polarity. Valid options are as given below.	
	Options	Description
	SIO4_SYNC_CLOCK_POL_FALL	The receiver clocks data in on the clock's falling edge (default).
	SIO4_SYNC_CLOCK_POL_RISE	The receiver clocks data in on the clock's rising edge.
env	This structure contains Rx Envelope settings.	
env. cfg	This field specifies the basic envelope signal configuration. Valid options are as given below.	
	Options	Description
	SIO4_SYNC_RXE_CFG_ACTIVE_HI	The envelope is high when asserted.
	SIO4_SYNC_RXE_CFG_ACTIVE_LO	The envelope is low when asserted.
	SIO4_SYNC_RXE_CFG_DISABLE	The envelope is disabled.
env. pol	This field specifies the polarity of the envelope signal. Valid options are as given below.	
	Options	Description
	SIO4_SYNC_ENV_POL_ACTIVE_HI	The envelope is high when asserted.
	SIO4_SYNC_ENV_POL_ACTIVE_LO	The envelope is low when asserted.
data	This structure contains Rx Data settings.	
data. cfg	This field specifies the basic data signal configuration. Valid options are as given below.	
	Options	Description
	SIO4_SYNC_RXD_CFG_ACTIVE_HI	The data signal uses active high logic.
	SIO4_SYNC_RXD_CFG_ACTIVE_LO	The data signal uses active low logic.
data. idle	This field specifies the state of the data line while transmission is idle. Valid options are as given below.	
	Options	Description
	SIO4_SYNC_TXD_IDLE_CFG_OUT_0	The data signal is driven low.
	SIO4_SYNC_TXD_IDLE_CFG_OUT_1	The data signal is driven high.
data. legacy	This field specifies the basic data signal configuration for legacy cable configurations. Valid options are as given below.	
	Options	Description
	SIO4_SYNC_LEG_RXD_CFG_LOW	The data signal is routed to the lower cable signal pins.
	SIO4_SYNC_LEG_RXD_CFG_TRI	The data signal is disabled.
	SIO4_SYNC_LEG_RXD_CFG_UP	The data signal is routed to the upper cable signal pins.

4.2.3. sio4_sync_tx_t

This structure is used to define transmitter specific SYNC configuration items.

NOTE: This structure does not contain any fields pertaining to the desired transmit bit rate.

Definition

```
typedef struct
{
    s32    enable;           // SIO4_IOCTL_SYNC_TX_ENABLE
    s32    empty_cfg;       // SIO4_IOCTL_TX_FIFO_EMPTY_CFG
```

```

s32      bit_order;      // SIO4_IOCTL_SYNC_TX_BIT_ORDER
u16      word_size;      // SIO4_IOCTL_SYNC_TX_WORD_SIZE
u16      gap_size;       // SIO4_IOCTL_SYNC_TX_GAP_SIZE

struct
{
    s32    cfg;           // SIO4_IOCTL_SYNC_TXC_CFG
    s32    pol;           // SIO4_IOCTL_SYNC_TXC_POL
    s32    src;           // SIO4_IOCTL_SYNC_TXC_SRC
    s32    idle;          // SIO4_IOCTL_SYNC_TXC_IDLE
    s32    idle_cfg;      // SIO4_IOCTL_SYNC_TXC_IDLE_CFG
} clock;

struct
{
    s32    cfg;           // SIO4_IOCTL_SYNC_TXE_CFG
    s32    pol;           // SIO4_IOCTL_SYNC_TXE_POL
} env;

struct
{
    s32    cfg;           // SIO4_IOCTL_SYNC_TXD_CFG
    s32    idle;          // SIO4_IOCTL_SYNC_TXD_IDLE_CFG
    s32    legacy;        // SIO4_IOCTL_SYNC_LEG_TXD_CFG
} data;

s32      aux_clock;      // SIO4_IOCTL_SYNC_TXAUXC_CFG
s32      spare;          // SIO4_IOCTL_SYNC_TXSP_CFG
} sio4_sync_tx_t;

```

Fields	Description						
enable	This field specifies the transmitter enable state. Valid options are as given below.						
	<table><tr><th>Options</th><th>Description</th></tr><tr><td>SIO4_SYNC_ENABLE_NO</td><td>The transmitter is disabled.</td></tr><tr><td>SIO4_SYNC_ENABLE_YES</td><td>The transmitter is enabled.</td></tr></table>	Options	Description	SIO4_SYNC_ENABLE_NO	The transmitter is disabled.	SIO4_SYNC_ENABLE_YES	The transmitter is enabled.
	Options	Description					
	SIO4_SYNC_ENABLE_NO	The transmitter is disabled.					
SIO4_SYNC_ENABLE_YES	The transmitter is enabled.						
empty_cfg	This field specifies the transmitter response to the Tx FIFO going empty. Valid options are as given below.						
	<table><tr><th>Options</th><th>Description</th></tr><tr><td>SIO4_TX_FIFO_EMPTY_CFG_IGNORE</td><td>No action takes place</td></tr><tr><td>SIO4_TX_FIFO_EMPTY_CFG_TX_OFF</td><td>The transmitter is disabled.</td></tr></table>	Options	Description	SIO4_TX_FIFO_EMPTY_CFG_IGNORE	No action takes place	SIO4_TX_FIFO_EMPTY_CFG_TX_OFF	The transmitter is disabled.
	Options	Description					
	SIO4_TX_FIFO_EMPTY_CFG_IGNORE	No action takes place					
SIO4_TX_FIFO_EMPTY_CFG_TX_OFF	The transmitter is disabled.						
bit_order	This field specifies the bit transmit order. Valid options are as given below.						
	<table><tr><th>Options</th><th>Description</th></tr><tr><td>SIO4_SYNC_BIT_ORDER_LSB</td><td>Transmit the least significant bit first.</td></tr><tr><td>SIO4_SYNC_BIT_ORDER_MSB</td><td>Transmit the most significant bit first.</td></tr></table>	Options	Description	SIO4_SYNC_BIT_ORDER_LSB	Transmit the least significant bit first.	SIO4_SYNC_BIT_ORDER_MSB	Transmit the most significant bit first.
	Options	Description					
	SIO4_SYNC_BIT_ORDER_LSB	Transmit the least significant bit first.					
SIO4_SYNC_BIT_ORDER_MSB	Transmit the most significant bit first.						
word_size	This specifies the size, in bits, of each transmitted data word. The value eight (8) is most common. Valid values are from one to 0xFFFF.						
gap_size	This specifies the size of the gap, in single bit periods, that is inserted between consecutive data words. Valid values are from zero to 0xFFFF.						
clock	This structure contains Tx Clock settings.						

clock. cfg	<p>This field specifies the basic clock signal configuration. Valid options are as given below.</p> <table border="1"> <thead> <tr> <th>Options</th><th>Description</th></tr> </thead> <tbody> <tr> <td>SIO4_SYNC_TXC_CFG_EXT</td><td>The transmitter is configured to use an external clock.</td></tr> <tr> <td>SIO4_SYNC_TXC_CFG_INT</td><td>The transmitter is configured to use an internal clock.</td></tr> </tbody> </table>	Options	Description	SIO4_SYNC_TXC_CFG_EXT	The transmitter is configured to use an external clock.	SIO4_SYNC_TXC_CFG_INT	The transmitter is configured to use an internal clock.								
Options	Description														
SIO4_SYNC_TXC_CFG_EXT	The transmitter is configured to use an external clock.														
SIO4_SYNC_TXC_CFG_INT	The transmitter is configured to use an internal clock.														
clock. pol	<p>This field specifies the clock's polarity. Valid options are as given below.</p> <table border="1"> <thead> <tr> <th>Options</th><th>Description</th></tr> </thead> <tbody> <tr> <td>SIO4_SYNC_CLOCK_POL_FALL</td><td>The transmitter clocks data out on the clock's falling edge.</td></tr> <tr> <td>SIO4_SYNC_CLOCK_POL_RISE</td><td>The transmitter clocks data out on the clock's rising edge.</td></tr> </tbody> </table>	Options	Description	SIO4_SYNC_CLOCK_POL_FALL	The transmitter clocks data out on the clock's falling edge.	SIO4_SYNC_CLOCK_POL_RISE	The transmitter clocks data out on the clock's rising edge.								
Options	Description														
SIO4_SYNC_CLOCK_POL_FALL	The transmitter clocks data out on the clock's falling edge.														
SIO4_SYNC_CLOCK_POL_RISE	The transmitter clocks data out on the clock's rising edge.														
clock. src	<p>This field specifies the clock's source. Valid options are as given below.</p> <table border="1"> <thead> <tr> <th>Options</th><th>Description</th></tr> </thead> <tbody> <tr> <td>SIO4_SYNC_TXC_SRC_0</td><td>The clock input is a logic low.</td></tr> <tr> <td>SIO4_SYNC_TXC_SRC_1</td><td>The clock input is a logic high.</td></tr> <tr> <td>SIO4_SYNC_TXC_SRC_EXT_FALL</td><td>Data is transmitted on the falling edge of the external clock.</td></tr> <tr> <td>SIO4_SYNC_TXC_SRC_EXT_RISE</td><td>Data is transmitted on the rising edge of the external clock.</td></tr> <tr> <td>SIO4_SYNC_TXC_SRC_OSC_HALF_FALL</td><td>Data is transmitted on the falling edge of the internal clock, which is divided by two.</td></tr> <tr> <td>SIO4_SYNC_TXC_SRC_OSC_HALF_RISE</td><td>Data is transmitted on the rising edge of the internal clock, which is divided by two.</td></tr> </tbody> </table>	Options	Description	SIO4_SYNC_TXC_SRC_0	The clock input is a logic low.	SIO4_SYNC_TXC_SRC_1	The clock input is a logic high.	SIO4_SYNC_TXC_SRC_EXT_FALL	Data is transmitted on the falling edge of the external clock.	SIO4_SYNC_TXC_SRC_EXT_RISE	Data is transmitted on the rising edge of the external clock.	SIO4_SYNC_TXC_SRC_OSC_HALF_FALL	Data is transmitted on the falling edge of the internal clock, which is divided by two.	SIO4_SYNC_TXC_SRC_OSC_HALF_RISE	Data is transmitted on the rising edge of the internal clock, which is divided by two.
Options	Description														
SIO4_SYNC_TXC_SRC_0	The clock input is a logic low.														
SIO4_SYNC_TXC_SRC_1	The clock input is a logic high.														
SIO4_SYNC_TXC_SRC_EXT_FALL	Data is transmitted on the falling edge of the external clock.														
SIO4_SYNC_TXC_SRC_EXT_RISE	Data is transmitted on the rising edge of the external clock.														
SIO4_SYNC_TXC_SRC_OSC_HALF_FALL	Data is transmitted on the falling edge of the internal clock, which is divided by two.														
SIO4_SYNC_TXC_SRC_OSC_HALF_RISE	Data is transmitted on the rising edge of the internal clock, which is divided by two.														
clock. idle	<p>This field specifies the clock's operation when data is not be transmitted. Valid options are as given below.</p> <table border="1"> <thead> <tr> <th>Options</th><th>Description</th></tr> </thead> <tbody> <tr> <td>SIO4_SYNC_TXC_IDLE_NO</td><td>The clock is idle.</td></tr> <tr> <td>SIO4_SYNC_TXC_IDLE_YES</td><td>The clock continues to run.</td></tr> </tbody> </table>	Options	Description	SIO4_SYNC_TXC_IDLE_NO	The clock is idle.	SIO4_SYNC_TXC_IDLE_YES	The clock continues to run.								
Options	Description														
SIO4_SYNC_TXC_IDLE_NO	The clock is idle.														
SIO4_SYNC_TXC_IDLE_YES	The clock continues to run.														
clock. idle_cfg	<p>This field specifies the clock's operation when data is not be transmitted. Valid options are as given below.</p> <table border="1"> <thead> <tr> <th>Options</th><th>Description</th></tr> </thead> <tbody> <tr> <td>SIO4_SYNC_TXC_IDLE_CFG_ACTIVE</td><td>The clock continues to run.</td></tr> <tr> <td>SIO4_SYNC_TXC_IDLE_CFG_IDLE_0</td><td>The signal is held low.</td></tr> <tr> <td>SIO4_SYNC_TXC_IDLE_CFG_IDLE_1</td><td>The signal is held high.</td></tr> </tbody> </table>	Options	Description	SIO4_SYNC_TXC_IDLE_CFG_ACTIVE	The clock continues to run.	SIO4_SYNC_TXC_IDLE_CFG_IDLE_0	The signal is held low.	SIO4_SYNC_TXC_IDLE_CFG_IDLE_1	The signal is held high.						
Options	Description														
SIO4_SYNC_TXC_IDLE_CFG_ACTIVE	The clock continues to run.														
SIO4_SYNC_TXC_IDLE_CFG_IDLE_0	The signal is held low.														
SIO4_SYNC_TXC_IDLE_CFG_IDLE_1	The signal is held high.														
env	This structure contains Tx Envelope settings.														
env. cfg	<p>This field specifies the basic envelope signal configuration. Valid options are as given below.</p> <table border="1"> <thead> <tr> <th>Options</th><th>Description</th></tr> </thead> <tbody> <tr> <td>SIO4_SYNC_TXE_CFG_ACTIVE_LO</td><td>The envelope is low when asserted.</td></tr> <tr> <td>SIO4_SYNC_TXE_CFG_ACTIVE_HI</td><td>The envelope is high when asserted.</td></tr> <tr> <td>SIO4_SYNC_TXE_CFG_OUT_0</td><td>The signal is driven low.</td></tr> <tr> <td>SIO4_SYNC_TXE_CFG_OUT_1</td><td>The signal is driven high.</td></tr> </tbody> </table>	Options	Description	SIO4_SYNC_TXE_CFG_ACTIVE_LO	The envelope is low when asserted.	SIO4_SYNC_TXE_CFG_ACTIVE_HI	The envelope is high when asserted.	SIO4_SYNC_TXE_CFG_OUT_0	The signal is driven low.	SIO4_SYNC_TXE_CFG_OUT_1	The signal is driven high.				
Options	Description														
SIO4_SYNC_TXE_CFG_ACTIVE_LO	The envelope is low when asserted.														
SIO4_SYNC_TXE_CFG_ACTIVE_HI	The envelope is high when asserted.														
SIO4_SYNC_TXE_CFG_OUT_0	The signal is driven low.														
SIO4_SYNC_TXE_CFG_OUT_1	The signal is driven high.														

env. pol	This field specifies the polarity of the envelope signal. Valid options are as given below.	
	Options	Description
	SIO4_SYNC_ENV_POL_ACTIVE_HI	The envelope is high when asserted.
	SIO4_SYNC_ENV_POL_ACTIVE_LO	The envelope is low when asserted.
data	This structure contains Tx Data settings.	
data. cfg	This field specifies the basic data signal configuration. Valid options are as given below.	
	Options	Description
	SIO4_SYNC_TXD_CFG_ACTIVE_HI	The data signal uses active high logic.
	SIO4_SYNC_TXD_CFG_ACTIVE_LO	The data signal uses active low logic.
	SIO4_SYNC_TXD_CFG_OUT_0	The data signal is driven low.
	SIO4_SYNC_TXD_CFG_OUT_1	The data signal is driven high.
data. legacy	This field specifies the basic data signal configuration for legacy cable configurations. Valid options are as given below.	
	Options	Description
	SIO4_SYNC_LEG_TXD_CFG_LOW	The data signal is routed to the lower cable signal pins.
	SIO4_SYNC_LEG_TXD_CFG_TRI	The data signal is tri-stated.
	SIO4_SYNC_LEG_TXD_CFG_UP	The data signal is routed to the upper cable signal pins.
	SIO4_SYNC_LEG_TXD_CFG_UP_LOW	The data signal is routed to both the upper and the upper cable signal pins.
aux_clock	This field specifies the basic data signal configuration for Tx Auxiliary Clock output signal. Valid options are as given below.	
	Options	Description
	SIO4_SYNC_TXAUXC_CFG_TRI	The signal is tri-stated.
	SIO4_SYNC_TXAUXC_CFG_OSC_HALF	The data signal is driven by the on-board oscillator, which is divided in half.
	SIO4_SYNC_TXAUXC_CFG_OUT_0	The output signal is driven low.
	SIO4_SYNC_TXAUXC_CFG_OUT_1	The output signal is driven high.
spare	This field specifies the basic data signal configuration for Tx Auxiliary Clock output signal. Valid options are as given below.	
	Options	Description
	SIO4_SYNC_TXSP_CFG_DISABLE	The signal is disabled.
	SIO4_SYNC_TXSP_CFG_INPUT	The signal is configured as an input.
	SIO4_SYNC_TXSP_CFG_OUT_0	The output signal is driven low.
	SIO4_SYNC_TXSP_CFG_OUT_1	The output signal is driven high.

5. Operating Information

This section explains some basic operational procedures for using the SIO4 with the SYNC Protocol Library. This is in no way intended to be a comprehensive guide. This is simply to address a very few issues relating to their use.

5.1. Basic Illustration

The below figure is included to assist individuals in the configuration of the SIO4-SYNC. The figure illustrates boards with more recent firmware. The DMA references are handled automatically by the driver to facilitate movement of data between the USC and the on-board FIFOs.

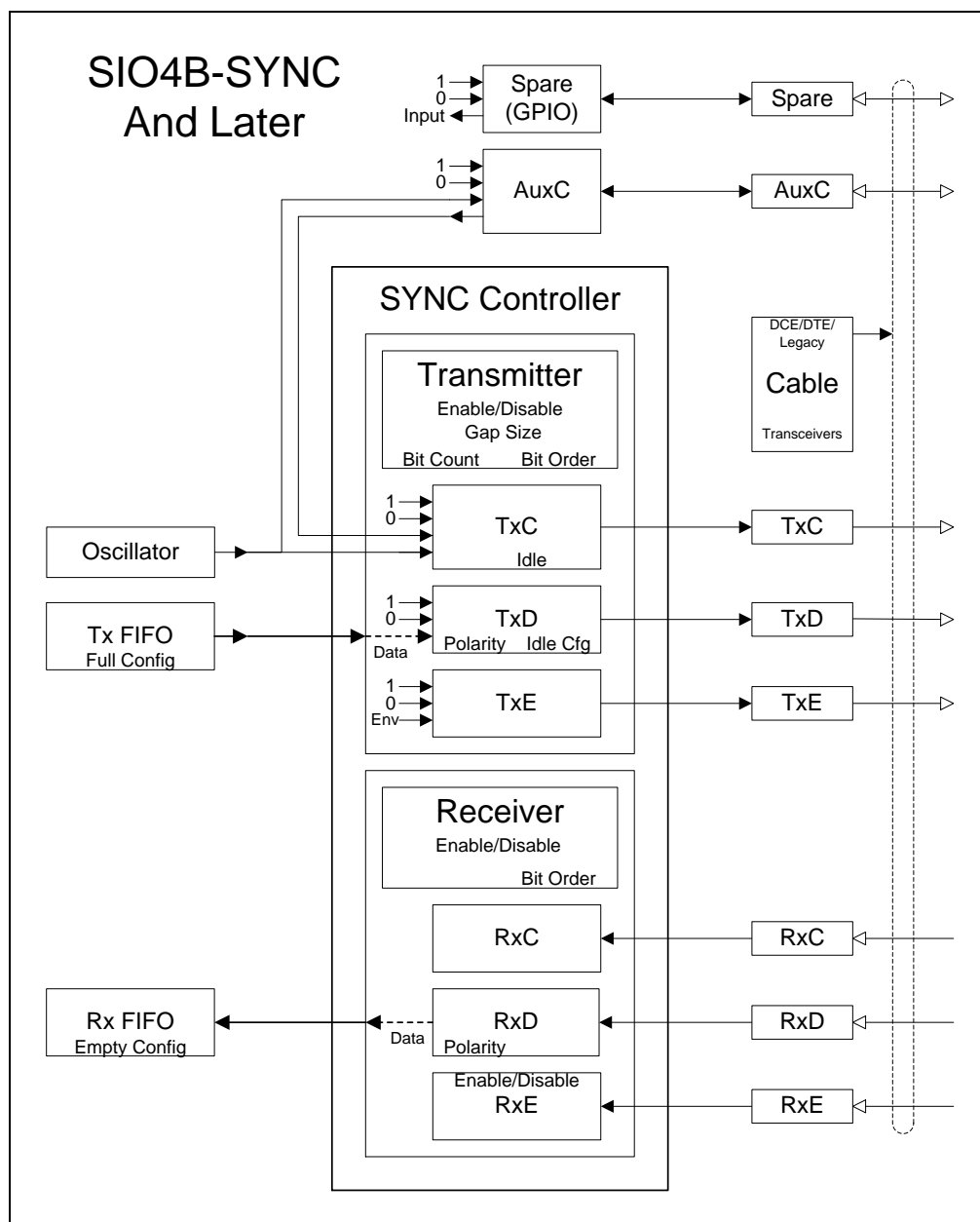


Figure 3 A functional illustration of an SIO4B-SYNC or later model board.

5.2. Getting Started

To configure the SIO4 for SYNC operation meeting specific requirements, the recommended starting point is a local, customizable copy of the `syncc2c` sample application included with the driver. This application is designed to transfer bulk data between an SIO4 transmitter and an SIO4 receiver. The transmitter and receiver can be from two channels on different boards, two channels on the same board, or by loopback mode using the transmitter and receiver from the very same channel. There are two loopback configurations. Internal Loopback performs the transfer by routing signals totally onboard the SIO4 without driving the cable transceivers. External Loopback performs the transfer by routing the signals through the cable transceivers.

NOTE: When using Loopback operation, it is recommended that cabling and any remote equipment be disconnected from the SIO4. This is because External Loopback is the default when Internal Loopback is not supported by firmware.

5.2.1. Bit Rates vs Oscillator Frequency

The Tx bit rate is tied directly to the frequency of the onboard programmable oscillator. Specifically, the Tx bit rate is exactly one half of the oscillator frequency. The default reference oscillator installed on the SIO4 is 20MHz. Therefore, the default maximum Tx bit rate is 10MHz. For whatever Tx bit rate is required, up to 10MHz, applications should program the oscillator to twice that rate. The Rx bit rate is totally up to the connected transmission source and is not affected in any way by the frequency of the onboard programmable oscillator.

5.2.2. Cable Validation

The first step in deriving a customized configuration is to verify cabling. This should be done using the application's default settings. It is recommended that one channel be used for loopback testing and that two other channels on the same board be connected by the cabling to be tested. A script with the below commands is a convenient means of repeating these tests until cabling has been verified successfully.

```
./syncc2c 0 0 -i
./syncc2c 0 0 -e
./syncc2c 1 2
```

5.2.3. Customizing the Configuration

The second step in deriving a customized configuration is to methodically modify the application code, one parameter at a time, until all necessary parameter changes have been accommodated. That is, choose a parameter from the documentation that must be changed from its default, modify the application so the required setting can be specified from the command line, then test the resulting changes. A script with the below commands is a convenient means of repeating these tests. In this example the “-X” represents the new command line argument for the parameter being altered from its default. This script tests all three cabling setups both without the change and with the change. This is done to verify that the default operation remains functional after the code modifications. In some cases, the script may need to be expanded to test each parameter addition to ensure prior changes remain functional. It is suggested that parameter changes be accommodated one at a time to ease the development and testing process.

```
./syncc2c 0 0 -I
./syncc2c 0 0 -e
./syncc2c 1 2 -I
./syncc2c 0 0 -I -X
./syncc2c 0 0 -e -X
./syncc2c 1 2 -X
```

NOTE: At times modifications for a parameter may need to be implemented on the transmitter or receiver first in order to facilitate validation. At other times the transmitter and receiver may temporarily be configured differently in order to verify that a change is implemented properly.

NOTE: It is best to initially test parameter additions separately, one at a time. Where there are parameter interactions, testing parameter combinations should be completed before moving on.

The general sequence for addition of a new parameter modification is as follows.

1. Add a field representing the parameter to the `args_t` structure defined in `main.h`.
2. Add command line support for the new parameter by updating the `_parse_args()` function at the top of `main.c`. At minimum, the `list[]` table must be updated to associate assignment of a setting with a command line argument. This may also mean assigning a default to the new field following the `memset()` function call.

NOTE: One may have to review multiple sample applications to get a feel for how to add specific command line argument types to the argument table.

3. Update the `_setup_common()`, `_setup_tx()` or `_setup_rx()` function in `setup.c` to apply the value from the new field to the `sio4_sync_t`, `sio4_sync_tx_t` or `sio4_sync_rx_t` structure prior to calling `sio4_sync_set()`, `sio4_sync_tx_set()` or `sio4_sync_rx_set()`, respectively.
4. Now update the script steps given above for the code changes just implemented. Continue adding support for other required parameter changes when testing is complete. Update the above script for each addition.

5.3. Debugging Aids

The SIO4 driver archive includes debugging aids appropriate for use with the SYNC Protocol Library. The aids are described below.

5.3.1. Device Identification

When communicating with technical support complete device identification is virtually always necessary. The *id* example application is provided for this specific purpose. This is a text only console application. The output can be piped to a file, which can then be emailed to GSC technical support when requested. Locate the application as follows.

Description	File	Location	OS
Application	<code>id</code>	<code>.../id/</code>	Linux
	<code>id.rta</code>	<code>...\\id\\</code>	INtime

5.3.2. `sio4_reg_list()`

The function `sio4_reg_list()` is included in the SIO4 utility library. The purpose of the function is to report the current content of registers for the referenced serial channel. The arguments control the set of registers included in the output and the detail with which the register content is reported. This function can be called at any time to report the device state, but it is most often called after completing board setup, or just before or after `sio4_sync_read()` or `sio4_sync_write()` calls in order to help explain the results of individual transfer requests.

Prototype

```
int sio4_reg_list(int fd, int gsc, int gsc_detail,
                 int usc, int usc_detail);
```

Argument	Description
<code>fd</code>	This is a file descriptor obtained by calling <code>sio4_sync_open()</code> (section 4.1.7, page 12).

<code>gsc</code>	If non-zero, then the output will include a dump of all <code>GSC_SIO4_xxx</code> registers. Refer to <code>sio4.h</code> for a complete list of these registers.
<code>gsc_detail</code>	If non-zero, then the dump of the GSC registers will include detailed information about all register fields, including the field value and the meaning of the value.
<code>usc</code>	This should be zero for SYNC boards.
<code>usc_detail</code>	This should be zero for SYNC boards.

Return Value	Description
<code>>= 0</code>	This is the number of errors encountered during execution of the function.

5.4. Cable Configuration Modes

The SIO4 supports two cable interfacing modes; DCE/DTE mode and Legacy mode. Older boards support only Legacy mode. More recent boards support only DCE/DTE mode. Intermediate boards support both modes. When both are available selection of the active mode is governed by enabling or disabling the transceivers. This is done through the `sio4_sync_t.cable.enable` field, which is described under section 4.2.1 beginning on page 15.

5.4.1. DCE/DTE Mode

The DCE/DTE Mode controls cable signaling according to the DCE or DTE selection, as described in the board user manual. When available in firmware this mode is enabled by enabling the cable transceivers (see paragraph above). The SYNC Protocol Library passes all DCE/DTE mode settings to the driver unconditionally. DCE/DTE mode settings received by the driver are applied only if this mode is supported by the board. The driver otherwise ignores these settings.

5.4.2. Legacy Mode

The Legacy Mode of operation controls signal routing for the cable interface, according to the Upper and Lower settings, and the USC, as described in the board user manual. When available in firmware this is the default mode of operation. When selectable this mode is activated by disabling the cable transceivers (see paragraph above). The SYNC Protocol Library's `sio4_sync_set()` function (section 4.1.11, page 13) applies Legacy mode settings only if the Legacy mode will be active when the function exits. The library otherwise ignores the Legacy settings. All other library functions process their settings unconditionally. Legacy mode settings received by the driver are applied only if this mode is supported by the board. The driver otherwise ignores these settings.

Document History

Revision	Description
March 19, 2024	Updated the release date. Added comment about use of the static libraries.
November 16, 2023	Updated the release date.
June 15, 2023	Updated the release date. Minor editorial changes.
December 13, 2022	Updated release date. Corrected Rx Clock Polarity options. Added information on the Tx bit rate and the programmable oscillator frequency. Minor editorial modifications.
June 2, 2022	Updated the release date.
February 8, 2022	Updated the release date. Minor typographic corrections. Added a Getting Started section (section 5.2, page 22). Updated the Debugging Aids section (section 5.3, page 23).
August 9, 2021	Updated the release date. Updated the descriptions of the I/O return values. Changed some argument and field names for consistency and clarity. Added a note about read requests being serialized. Added a note about write requests being serialized. Added statements about the read and write services being blocking calls.
May 5, 2021	Updated the release date.
February 26, 2021	Updated the release date.
February 18, 2021	Updated the release date.
October 12, 2020	Updated the release date.
July 30, 2019	Updated the release date.
March 24, 2019	Updated the release date.
March 15, 2019	Minor editorial changes. Corrected numerous function names missing “_sync_” in the middle of the names. Reordered functions alphabetically.
October 18, 2018	Updated the inside cover page.
October 31, 2017	Added information on the Cable Configuration Modes. Updated some legacy setting information.
October 17, 2017	Minor overhaul of reference manual. Removed library versioning. Added information on opening device index -1. Added section on library interface files. Updated return status information for high level services. Added support for device index -1. Numerous editorial changes throughout.
December 7, 2016	Added a section for general operating information. Made various miscellaneous updates.
April 13, 2016	Updated to version 1.10.
September 7, 2015	Updated to version 1.9.
December 9, 2014	Update the release date.
December 4, 2014	Update the release date.
May 17, 2014	Update the release date.
April 16, 2014	Updated to version 1.8.
October 22, 2013	Updated to version 1.7.
October 15, 2013	Updated to version 1.6.
August 27, 2013	Updated to version 1.5. Renamed library base name to <code>sio4_sync</code> . Rewrote library. Updated structure field names and valid value sets.
October 11, 2012	Updated to version 1.4. Renamed library base name to <code>sio4_sync</code> .
September 9, 2012	Initial release, version 1.3. This is for the 2.x series driver.