

SIO4/8

Four/Eight Channel High Speed Serial I/O

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

Asynchronous Protocol Library Reference Manual

**Manual Revision: September 7, 2015
Library Release Version: 1.1**

**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 © 2014-2015, **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 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.

Zilog and Z16C30 are trademarks of Zilog, Inc.

Table of Contents

1. Introduction.....	5
1.1. Purpose	5
1.2. Acronyms	5
1.3. Definitions.....	5
1.4. Software Overview.....	5
1.5. Hardware Overview.....	5
1.6. Reference Material.....	6
2. The Asynchronous Serial Protocol.....	7
2.1. Description.....	7
2.2. History.....	7
3. Library Interface	8
3.1. High Level Functions.....	8
3.1.1. sio4_async_close()	8
3.1.2. sio4_async_get().....	8
3.1.3. sio4_async_init()	9
3.1.4. sio4_async_ioctl().....	9
3.1.5. sio4_async_lib_init()	10
3.1.6. sio4_async_open().....	10
3.1.7. sio4_async_read().....	10
3.1.8. sio4_async_set()	11
3.1.9. sio4_async_show()	11
3.1.10. sio4_async_version()	12
3.1.11. sio4_async_write()	12
3.2. Low Level Functions.....	13
3.3. Data Structures	14
3.3.1. sio4_async_t.....	14
4. Operation	35
4.1. Basic Illustration	35
4.2. Clocking Configurations.....	35
4.3. Error and Status Detection	36
4.3.1. Interrupt Events	37
4.3.2. Rx Status Word.....	37
4.4. Debugging Aids	37
4.4.1. sio4_async_show()	37
4.4.2. sio4_reg_list().....	38
4.5. Exclusions	38
4.5.1. Global Rx FIFO Full Configuration	38
Document History	39

Table of Figures

Figure 1 A depiction of an Asynchronous data stream.7

Figure 2 A functional illustration of an SIO4B or later model board.35

Figure 3 This illustrates the default Asynchronous clock routing on SIO4B and later model boards.36

1. Introduction

This document provides information on the Asynchronous Protocol Library, which is a library designed to facilitate use of the Asynchronous serial protocol with an SIO4 or SIO8.

1.1. Purpose

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

1.2. Acronyms

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

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

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 user space with user mode privileges.
ASYNC	This refers to the Asynchronous serial protocol.
Driver	Driver means the executable programming providing the direct access to the SIO4 hardware.
SIO4	This is used as a general reference to any Zilog based 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.

1.4. Software Overview

The Asynchronous Protocol Library is a statically linked library providing an Asynchronous 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 device driver. The interface provided by the library is Asynchronous 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. Applications are free to use the library interface and the driver interface at will. The only requirements are that applications must use the library's `sio4_async_lib_init()`, `sio4_async_open()` and `sio4_async_close()` functions.

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 the SIO4 and remote devices. The SIO4 board includes two DMA controllers and comes with a maximum of 256K Bytes of FIFO storage, which is 32K per channel per direction (32K * 2 * 4). The FIFO configuration can vary greatly from one SIO4 version to another (i.e. 32K * 2 * 4 to 512 * 2 * 2 to none at all). The SIO4 can be configured with a number of different cable transceivers. The transceivers can be fixed as RS232 or RS422, or they can be changed programmatically to a number of different options. The Zilog version of the SIO4

supports a number of different serial protocols by use of a pair of dual Universal Serial Controllers (the Zilog Z16C30). This includes Asynchronous, Monosync, Bisync, HDLC, Isochronous and others.

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 applicable *SIO4/SIO8 Driver User Manual* from General Standards Corporation.
- The *PCI Bus Master Interface Chip* data handbook 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.

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

2. The Asynchronous Serial Protocol

2.1. Description

The Asynchronous serial communications protocol is a byte oriented serial transmission protocol consisting of a Start Bit, a number of data bits, an optional Parity Bit, and up to two Stop Bits. For successful data transfer the provider and recipient must agree upon the transmission rate, the number of data bits, the use and type of parity, and the minimum length of the Stop Bit period. Refer to Figure 1 below. When no data is being transmitted the line is idle. When idle the line is held in the high or Mark state (a 1 bit). On the receiving side, a transition from high to low signals the beginning of a Start Bit. From this point the line level is sampled at about the center of each bit period. At the center of the Start Bit period, the line level is examined. If it is high, then the high to low transition is ignored as noise, and the decoding logic returns to waiting for another high-to-low transition. If the line is still low (the Space state, which is a 0 bit) then decoding continues per the byte size, use of parity and at least $\frac{1}{2}$ Stop Bit. If parity is used then the Parity Bit is evaluated with the data. If the evaluation fails, then the result is a Parity Error. After the last data bit and optional Parity Bit, the Stop Bit period begins. If the line is low when the Stop Bit is sampled, then the result is a Framing Error.

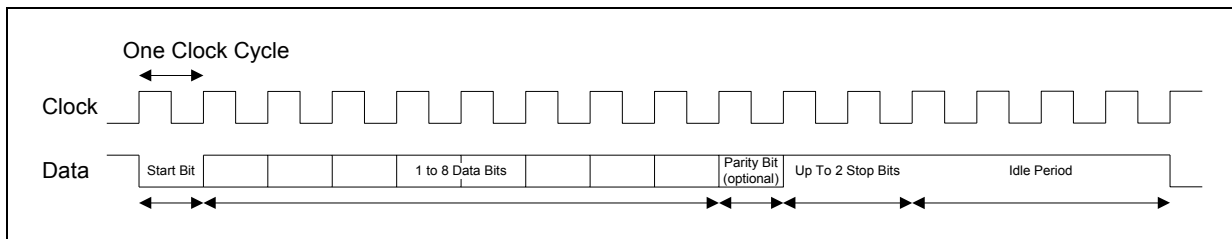


Figure 1 A depiction of an Asynchronous data stream.

The minimum signals needed for full-duplex communication are TxD and RxD (*Transmit Data* and *Receive Data*). Hardware flow control requires additional signals. The RTS/CTS flow control mechanism requires two additional signals. (RTS refers to *Request To Send* and CTS refers to *Clear To Send*.) The DTR/DSR flow control mechanism also requires two signals. (DTR refers to *Data Terminal Ready* and DSR refers to *Data Set Ready*.) Communications with a modem may also use the DCD signal (DCD refers to *Data Carrier Detect*).

Equipment using the Asynchronous protocol usually uses the RS-232 line protocol. The number of signals varies with individual implementations and the set of supported signals. The arrangement of the supported signals is typically DCE or DTE, or some other custom implementation. (DCE refers to *Data Communications Equipment* and DTE refers to *Data Terminal Equipment*.) The typical upper baud rate limit is 115,200 bits-per-second. Higher rates are possible with special RS-232 transceivers or by using other than RS-232.

2.2. History

The Asynchronous serial communications protocol has its origins prior to 1920, back in the days of the early electromechanical teletypewriter. At that time byte sizes were typically five-bits and the Stop Bit period was typically 1.5 times the bit period. Since then byte size support typically ranges from five to eight bits, and the Stop Bit period may be configurable in fractional increments from just over $\frac{1}{2}$ bit period to two bit periods.

3. Library Interface

The library header file is `sio4_async.h`. Including this header in a source file gives the source the full library and driver interface as the library header file includes the driver header files `sio4.h` and `sio4_usc.h`. The library header is located in the `.../sio4/async/lib/` directory. The library and all associated files are included as part of the driver archive.

3.1. High Level Functions

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

3.1.1. `sio4_async_close()`

This function is the entry point to close a connection previously opened to an SIO4 for Asynchronous operation. All resources allocated by the library for the opened device are released as part of the close operation. This includes freeing allocated memory and closing access to the SIO4 serial channel.

Prototype

```
int sio4_async_close(int fd);
```

Argument	Description
<code>fd</code>	This is a file descriptor obtained by calling <code>sio4_async_open()</code> (section 3.1.6, page 10).

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

3.1.2. `sio4_async_get()`

This function retrieves the current settings from the SIO4 for each of the referenced structure's fields. This function operates mostly by calling the low level functions corresponding to each field of the `sio4_async_t` structure.

Prototype

```
int sio4_async_get(int fd, sio4_async_t* async, const char** err);
```

Argument	Description
<code>fd</code>	This is a file descriptor obtained by calling <code>sio4_async_open()</code> (section 3.1.6, page 10).
<code>async</code>	This is the structure where the current settings are recorded (section 3.3.1, page 14). Any field pertaining to an unsupported feature will be set to <code>-1</code> . The value <code>-2</code> indicates a hardware setting that is invalid.
<code>err</code>	In the event of an error this will be set to identify the source of the error. The caller may provide a NULL pointer.

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

3.1.3. sio4_async_init()

This function initializes an `sio4_async_t` structure according to the capabilities of the accessed device and some basic caller preferences. This function operates mostly by calling the low level functions corresponding to each field of the `sio4_async_t` structure.

Prototype

```
int sio4_async_init(
    int          fd,
    const sio4_async_init_t*  init,
    sio4_async_t*  async,
    const char**   err);
```

Argument	Description
fd	This is a file descriptor obtained by calling <code>sio4_async_open()</code> (section 3.1.6, page 10).
init	This structure provides the basic information needed prior to initializing the next argument. See below for more information.
async	This is the structure that the call will initialize (section 3.3.1, page 14).
err	In the event of an error this will be set to identify the source of the error. The caller may provide a NULL pointer.

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

Data Type

This structure contains information used to configure Asynchronous clocking for the USC transmitter and receiver.

```
typedef struct
{
    s32 tx_bit_rate;
    s32 rx_bit_rate;
    s32 osc_prog;
} sio4_async_init_t;
```

Field	Description
tx_bit_rate	This is the desired bit rate for the transmitter. This value must be greater than or equal to one, and less than or equal to 1,250,000.
rx_bit_rate	This is the desired bit rate for the receiver. This value must be greater than or equal to one, and less than or equal to 1,250,000.
osc_prog	This is the frequency to which the channel's on-board oscillator is to be programmed. This should be the highest possible multiple of the Tx and Rx bit rates. If the on-board oscillator is not programmable, then this should be set to the frequency of the fixed on-board oscillator. The default rate is 20,000,000.

3.1.4. sio4_async_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_async_ioctl(int fd, int cmd, void* arg);
```

Argument	Description
fd	This is a file descriptor obtained by calling <code>sio4_async_open()</code> (section 3.1.6, page 10).
cmd	This is an IOCTL service macro from the driver header files.
arg	This is the argument required for the above referenced IOCTL service.

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

3.1.5. sio4_async_lib_init()

This function is required in order to prepare the Asynchronous Protocol Library for operation. This must be the first call to the library.

Prototype

```
int sio4_async_lib_init(void);
```

Argument	Description
None	The function has no arguments.

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

3.1.6. sio4_async_open()

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

Prototype

```
int sio4_async_open(int index);
```

Argument	Description
index	This is the zero based index of the SIO4 serial channel to access.

Return Value	Description
< 0	An error occurred. This is a negative <code>errno.h</code> value.
>= 0	A valid file access handle.

3.1.7. sio4_async_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.

NOTE: A read request returns an error if a read is already active.

Prototype

```
int sio4_async_read(int fd, void* buf, size_t size);
```

Argument	Description
fd	This is a file descriptor obtained by calling <code>sio4_async_open()</code> (section 3.1.6, page 10).
buf	The data read is placed here.
size	Read up to this number of bytes.

Return Value	Description
< 0	An error occurred. This is a negative <code>errno.h</code> value.
>= 0	This is the number of bytes placed in the buffer.

3.1.8. sio4_async_set()

This function configures an SIO4 channel according to the settings of the referenced `sio4_async_t` structure. All fields are validated before any settings are applied. This function operates mostly by calling the low level functions corresponding to each field of the `sio4_async_t` structure.

NOTE: Before calling this function the structure should, at minimum, be initialized by calling the `sio4_async_init()` function (section 3.1.3, page 9).

Prototype

```
int sio4_async_set(
    int          fd,
    const sio4_async_t* async,
    const char** err);
```

Argument	Description
fd	This is a file descriptor obtained by calling <code>sio4_async_open()</code> (section 3.1.6, page 10).
async	This is the structure containing the settings to be applied (section 3.3.1, page 14).
err	In the event of an error this will be set to identify the source of the error. The caller may provide a NULL pointer.

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

3.1.9. sio4_async_show()

This function displays the content of the referenced `sio4_async_t` structure to the screen. This is provided to assist debugging efforts. This function operates mostly by calling the low level functions corresponding to each field of the `sio4_async_t` structure.

Prototype

```
int sio4_async_show(
    int          fd,
    const sio4_async_t* async,
    const char** err);
```

Argument	Description
fd	This is a file descriptor obtained by calling <code>sio4_async_open()</code> (section 3.1.6, page 10).
async	This is the structure whose content will be displayed (section 3.3.1, page 14).
err	In the event of an error this will be set to identify the source of the error. The caller may provide a NULL pointer.

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

3.1.10. `sio4_async_version()`

This function retrieves the library's version number and build date and time.

Prototype

```
int sio4_async_version(const char** version, const char** built);
```

Argument	Description
fd	This is a file descriptor obtained by calling <code>sio4_async_open()</code> (section 3.1.6, page 10).
version	The library version number is returned here. The caller may provide a NULL pointer. The pointer returned refers to a static buffer.
built	The library build date and time are returned here. The string is formatted as if produced by the the C statement <code>printf("__DATE__ ", " __TIME__);</code> . The caller may provide a NULL pointer. The pointer returned refers to a static buffer.

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

3.1.11. `sio4_async_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 is first.

Prototype

```
int sio4_async_write(int fd, const void* buf, size_t size);
```

Argument	Description
fd	This is a file descriptor obtained by calling <code>sio4_async_open()</code> (section 3.1.6, page 10).
buf	This is the source for the data to write.
size	Write at most this number of bytes.

Return Value	Description
< 0	An error occurred. This is a negative <code>errno.h</code> value.
>= 0	This is the number of bytes written from the buffer.

3.2. Low Level Functions

The low level functions provide access to the board features reflected by the individual fields of the `sio4_async_t` structure (section 3.3.1, page 14). This structure is used to access all of the board features that are part of the Asynchronous Protocol Library. For each structure field there is a corresponding board feature and an associated low level function. When calling the high level functions that use the `sio4_async_t` structure, the high level functions perform their work by calling the low level functions for each structure field. This is especially useful for activities such as structure initialization and board configuration. Following high level configuration of the board though, there are times when an application may need to access features represented by only a subset of the `sio4_async_t` structure fields. This is where an application can make use of the low level function. All of the low level functions follow the prototype pattern shown below. The function naming convention includes the prefix “`sio4_async_t_`” followed by the C style field name, but with the periods (“.”) replaced by underscores (“_”).

NOTE: The low level functions do not accept NULL pointers.

Prototype

```
void sio4_async_t_field_name(int fd, s32* arg,
    sio4_async_action_t action, const char** err);
```

Argument	Description
fd	This is a file descriptor obtained by calling <code>sio4_async_open()</code> (section 3.1.6, page 10).
arg	This refers to the feature specific value being passed in to the function. This pointer must not be NULL.
action	This identifies the specific action to be carried out in regards to the above feature specific value. See the <code>sio4_async_action_t</code> data type below.
err	If there is an error, then this field will be set to point to a string localizing the error. The caller may pass in a NULL pointer.

Data Type

This enumeration identifies the specific action requested when a low level function is called.

```
typedef enum
{
    SIO4_ASYNC_ACTION_GET,
    SIO4_ASYNC_ACTION_INIT,
    SIO4_ASYNC_ACTION_SET,
    SIO4_ASYNC_ACTION_SHOW,
    SIO4_ASYNC_ACTION_VERIFY
} sio4_async_action_t;
```

Value	Description
<code>SIO4_ASYNC_ACTION_GET</code>	This requests the current setting from the driver.
<code>SIO4_ASYNC_ACTION_INIT</code>	This requests the initialization value from the library.
<code>SIO4_ASYNC_ACTION_SET</code>	This requests that the supplied value be applied by the driver.
<code>SIO4_ASYNC_ACTION_SHOW</code>	This requests that the supplied value be displayed to the screen.
<code>SIO4_ASYNC_ACTION_VERIFY</code>	This requests that the supplied value be verified.

Examples

For simplicity sake a low level function name can easily be derived given any field name, as shown in the below examples. The individual low level function names are identified with the corresponding structure fields beginning in section 3.3.1, page 14.

Field	Function
sio4_async_t.cable.loopback.mode	sio4_async_t_cable_loopback_mode()
sio4_async_t.rx.encoding	sio4_async_t_rx_encoding()
sio4_async_t.tx.parity.enable	sio4_async_t_tx_parity_enable()

3.3. Data Structures

The library header file is `sio4_async.h`. Including this header in a source file gives that source the full library and driver interface. The library header defines the complete Asynchronous interface offered by the library, which includes the following data structures and their associated macros and low level function declarations.

3.3.1. sio4_async_t

This structure contains all of the parameters used to configure an SIO4 channel for Asynchronous operation. The structure is initialized with default values by calling the `sio4_async_init()` function (section 3.1.3, page 9). Following this call, applications make changes to this structure's content according to their own requirements. Afterwards, the structure is passed to the `sio4_async_set()` function (section 3.1.7, page 10) where the settings are applied to the board.

```
typedef struct
{
    // All fields are filled in by the library when calling
    // sio4_async_init(). Application mods must be made before calling
    // sio4_async_set().

    struct
    {
        s32    ref;
        s32    prog;
    } osc;

    Struct    // cable
    {
        s32    enable;        // PSRCR D31
        s32    mode;          // PSRCR D28, DCE or DTE
        s32    protocol;      // PSRSR D24-D27
        s32    txc;           // PSRCR D6-D8
        s32    txd;           // PSRCR D19-D20
        s32    txaux;         // PSRCR D17-D18
        s32    dcd;           // PSRCR D15-D16
        s32    dtr_dsr;       // PSRCR D21-D22
        s32    rts;           // PSRCR D13-D14

        struct
        {
            s32 mode;          // PSRCR D23, D29
        } loopback;
    }
};
```

```

    struct
    {
        s32 enable;        // PSRCR D30
    } term;

    struct
    {
        s32 txc;           // CCR 0x3333
        s32 txd_cts;       // CSR D2-D3
        s32 rxc;           // CCR 0xCCCC
        s32 rxd_dcd;       // CSR D4-D5
    } legacy;

} cable;

Struct // tx
{
    s32 mode;              // USC CMR D8-D11
    s32 enable;            // USC TMR D0-D1
    s32 char_len;          // USC TMR D2-D4
    s32 encoding;          // USC TMR D13-D15
    s32 bit_rate;          // reflects sio4_async_init_t.tx_bit_rate
    s32 idle_cond;         // USC TCSR D8-D10

    struct
    {
        s32 enable;        // USC TMR D5
        s32 type;          // USC TMR D6-D7
    } parity;

    struct
    {
        s32 size;          // FSR D0-D15, read-only
        s32 ae;            // TAR D0-D15
        s32 af;            // TAR D16-D31
        s32 empty_cfg;     // CSR D18, D26
        s32 space_cfg;     // CSR D4-D5
    } fifo;

    struct
    {
        s32 mode;
        s32 pio_thresh;
        s32 timeout;
        s32 overrun;
    } io;

} tx;

Struct // rx
{
    s32 mode;              // USC CMR D0-D3
    s32 enable;            // USC RMR D0-D1
    s32 char_len;          // USC RMR D2-D4
    s32 encoding;          // USC RMR D13-D15
    s32 bit_rate;          // reflects sio4_async_init_t.rx_bit_rate
    s32 sync_byte;         // SBR D0-D7

```

```

s32      status_word; // CSR D3

struct
{
    s32 enable;        // USC RMR D5
    s32 type;          // USC RMR D6-D7
} parity;

struct
{
    s32 size;          // FSR D16-D21, read-only
    s32 ae;            // RAR D0-D15
    s32 af;            // RAR D16-D31
    s32 full_cfg;      // BCR D8
} fifo;

struct
{
    s32 mode;
    s32 pio_thresh;
    s32 timeout;
    s32 overrun;
    s32 underrun;
} io;

struct
{
    s32 enable;        // CSR D2
    s32 clk_src;       // BCR D22
} time_stamp;

} rx;

Struct   // usc
{
    s32 mode;          // USC CCAR D8-D9
    s32 txd;           // USC IOCR D6-D7
    s32 cts;           // PSRCR D9-D10, USC IOCR D14-D15
    s32 cts_legacy;    // USC IOCR D14-D15
    s32 dcd;           // PSRCR D11-D12, USC IOCR D12-D13
    s32 dcd_legacy;    // USC IOCR D12-D13

    // All of the folling USC fields are initialized
    // by sio4_async_init() based on the content of the
    // sio4_async_init_t structure.

    struct
    {
        s32 clk_rate;   // USC CMR D4-D5
        s32 clk_src;    // USC CMCR D3-D5
        s32 txc;        // PSRCR D0-D2, USC IOCR D3-D5
        s32 txc_legacy; // USC IOCR D3-D5
        s32 stop_bits;  // USC CMR D14
    } tx;

    struct
    {

```



```

        s32 clk_rate;    // USC CMR D12-D13
        s32 clk_src;    // USC CMCR D0-D2
        s32 rxc;        // PSRCR D3-D5, USC IOCR D0-D2
        s32 rxc_legacy; // USC IOCR D0-D2
    } rx;

    struct
    {
        s32 enable;      // USC HCR D0
        s32 clk_src;     // USC CMCR D8-D9
        s32 divider;     // USC TC1R D0-D15
        s32 mode;        // USC HCR D1
    } brg0;

    struct
    {
        s32 enable;      // USC HCR D4
        s32 clk_src;     // USC CMCR D10-D11
        s32 divider;     // USC TC0R D0-D15
        s32 mode;        // USC HCR D5
    } brg1;

    struct
    {
        s32 clk_src;     // USC CMCR D12-D13
        s32 rate;        // USC HCR D14-D15
    } ctr0;

    struct
    {
        s32 clk_src;     // USC CMCR D14-D15
        s32 rate_src;    // USC HCR D13, ...
    } ctrl;

    struct
    {
        s32 rate;        // USC HCR D10-D11
    } dp11;

    } usc;

} sio4_async_t;

```

3.3.1.1. sio4_async_t.osc

This section describes the structure's oscillator configuration fields.

Field	Description
osc	This structure configures the oscillator interface.
osc.ref	This field specifies the frequency of the fixed onboard reference oscillator. The default is 20MHz. The feature's low level function is <code>sio4_async_t_osc_ref()</code> . This value is provided for informational purposes only.
osc.prog	This field specifies the desired programmable oscillator frequency. This is essentially the clock frequency provided by the onboard programmable oscillator to the USC. The default is 20MHz. The feature's low level function is <code>sio4_async_t_osc_prog()</code> . The value provided is not recorded for later retrieval. Retrieval requests return one if oscillator programming is supported and zero if it isn't.

3.3.1.2. `sio4_async_t.cable`

This section describes the structure's cable configuration fields.

Field	Description																						
<code>cable</code>	This structure configures the cable interface.																						
<code>cable.enable</code>	<p>This field either enables or disables the cable transceivers. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_cable_enable()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td><code>SIO4_ASYNC_CABLE_ENABLE_NO</code></td><td>Disable the cable transceivers.</td></tr> <tr> <td><code>SIO4_ASYNC_CABLE_ENABLE_YES</code></td><td>Enable the cable transceivers. This is the default.</td></tr> </table>	Value	Description	<code>SIO4_ASYNC_CABLE_ENABLE_NO</code>	Disable the cable transceivers.	<code>SIO4_ASYNC_CABLE_ENABLE_YES</code>	Enable the cable transceivers. This is the default.																
Value	Description																						
<code>SIO4_ASYNC_CABLE_ENABLE_NO</code>	Disable the cable transceivers.																						
<code>SIO4_ASYNC_CABLE_ENABLE_YES</code>	Enable the cable transceivers. This is the default.																						
<code>cable.mode</code>	<p>This field specifies the arrangement of the signals on the cable interface. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_cable_mode()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td><code>SIO4_ASYNC_CABLE_MODE_DCE</code></td><td>Select the DCE cable signal configuration.</td></tr> <tr> <td><code>SIO4_ASYNC_CABLE_MODE_DTE</code></td><td>Select the DTE cable signal configuration. This is the default.</td></tr> </table>	Value	Description	<code>SIO4_ASYNC_CABLE_MODE_DCE</code>	Select the DCE cable signal configuration.	<code>SIO4_ASYNC_CABLE_MODE_DTE</code>	Select the DTE cable signal configuration. This is the default.																
Value	Description																						
<code>SIO4_ASYNC_CABLE_MODE_DCE</code>	Select the DCE cable signal configuration.																						
<code>SIO4_ASYNC_CABLE_MODE_DTE</code>	Select the DTE cable signal configuration. This is the default.																						
<code>cable.Protocol</code>	<p>This field specifies the cable transceiver configuration. The options available depend on the board's transceiver capabilities. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_cable_protocol()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td><code>SIO4_ASYNC_CABLE_PROTOCOL_DISABLE</code></td><td>This disables the cable transceivers.</td></tr> <tr> <td><code>SIO4_ASYNC_CABLE_PROTOCOL_RS232</code></td><td>This selects the RS232 protocol. This is the default.</td></tr> <tr> <td><code>SIO4_ASYNC_CABLE_PROTOCOL_RS422_423_1</code></td><td>This selects the RS422/RS423 mixed protocol version 1.</td></tr> <tr> <td><code>SIO4_ASYNC_CABLE_PROTOCOL_RS422_423_2</code></td><td>This selects the RS422/RS423 mixed protocol version 2.</td></tr> <tr> <td><code>SIO4_ASYNC_CABLE_PROTOCOL_RS422_RS485</code></td><td>This selects the RS422/RS485 mixed protocol.</td></tr> <tr> <td><code>SIO4_ASYNC_CABLE_PROTOCOL_RS423</code></td><td>This selects the RS423 protocol.</td></tr> <tr> <td><code>SIO4_ASYNC_CABLE_PROTOCOL_RS530</code></td><td>This selects the RS530 protocol, version 1.</td></tr> <tr> <td><code>SIO4_ASYNC_CABLE_PROTOCOL_RS530A</code></td><td>This selects the RS530 protocol, version 2.</td></tr> <tr> <td><code>SIO4_ASYNC_CABLE_PROTOCOL_V35</code></td><td>This selects the V.35 protocol, version 1.</td></tr> <tr> <td><code>SIO4_ASYNC_CABLE_PROTOCOL_V35A</code></td><td>This selects the V.35 protocol, version 2.</td></tr> </table>	Value	Description	<code>SIO4_ASYNC_CABLE_PROTOCOL_DISABLE</code>	This disables the cable transceivers.	<code>SIO4_ASYNC_CABLE_PROTOCOL_RS232</code>	This selects the RS232 protocol. This is the default.	<code>SIO4_ASYNC_CABLE_PROTOCOL_RS422_423_1</code>	This selects the RS422/RS423 mixed protocol version 1.	<code>SIO4_ASYNC_CABLE_PROTOCOL_RS422_423_2</code>	This selects the RS422/RS423 mixed protocol version 2.	<code>SIO4_ASYNC_CABLE_PROTOCOL_RS422_RS485</code>	This selects the RS422/RS485 mixed protocol.	<code>SIO4_ASYNC_CABLE_PROTOCOL_RS423</code>	This selects the RS423 protocol.	<code>SIO4_ASYNC_CABLE_PROTOCOL_RS530</code>	This selects the RS530 protocol, version 1.	<code>SIO4_ASYNC_CABLE_PROTOCOL_RS530A</code>	This selects the RS530 protocol, version 2.	<code>SIO4_ASYNC_CABLE_PROTOCOL_V35</code>	This selects the V.35 protocol, version 1.	<code>SIO4_ASYNC_CABLE_PROTOCOL_V35A</code>	This selects the V.35 protocol, version 2.
Value	Description																						
<code>SIO4_ASYNC_CABLE_PROTOCOL_DISABLE</code>	This disables the cable transceivers.																						
<code>SIO4_ASYNC_CABLE_PROTOCOL_RS232</code>	This selects the RS232 protocol. This is the default.																						
<code>SIO4_ASYNC_CABLE_PROTOCOL_RS422_423_1</code>	This selects the RS422/RS423 mixed protocol version 1.																						
<code>SIO4_ASYNC_CABLE_PROTOCOL_RS422_423_2</code>	This selects the RS422/RS423 mixed protocol version 2.																						
<code>SIO4_ASYNC_CABLE_PROTOCOL_RS422_RS485</code>	This selects the RS422/RS485 mixed protocol.																						
<code>SIO4_ASYNC_CABLE_PROTOCOL_RS423</code>	This selects the RS423 protocol.																						
<code>SIO4_ASYNC_CABLE_PROTOCOL_RS530</code>	This selects the RS530 protocol, version 1.																						
<code>SIO4_ASYNC_CABLE_PROTOCOL_RS530A</code>	This selects the RS530 protocol, version 2.																						
<code>SIO4_ASYNC_CABLE_PROTOCOL_V35</code>	This selects the V.35 protocol, version 1.																						
<code>SIO4_ASYNC_CABLE_PROTOCOL_V35A</code>	This selects the V.35 protocol, version 2.																						
<code>cable.txc</code>	<p>This field specifies the configuration of the cable's Tx Clock signal. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_cable_txc()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td><code>SIO4_ASYNC_CABLE_TXC_OUT_0</code></td><td>This drives the signal low.</td></tr> <tr> <td><code>SIO4_ASYNC_CABLE_TXC_OUT_1</code></td><td>This drives the signal high.</td></tr> <tr> <td><code>SIO4_ASYNC_CABLE_TXC_OUT_CBL_RXA</code></td><td>This drives the signal from what appears at the cable's Rx Aux signal.</td></tr> <tr> <td><code>SIO4_ASYNC_CABLE_TXC_OUT_CBL_RXC</code></td><td>This drives the signal from what appears at the cable's Rx Clock signal.</td></tr> <tr> <td><code>SIO4_ASYNC_CABLE_TXC_OUT_OSC</code></td><td>This drives the signal from the onboard oscillator.</td></tr> <tr> <td><code>SIO4_ASYNC_CABLE_TXC_OUT_OSC_INV</code></td><td>This drives the signal from the inverted form</td></tr> </table>	Value	Description	<code>SIO4_ASYNC_CABLE_TXC_OUT_0</code>	This drives the signal low.	<code>SIO4_ASYNC_CABLE_TXC_OUT_1</code>	This drives the signal high.	<code>SIO4_ASYNC_CABLE_TXC_OUT_CBL_RXA</code>	This drives the signal from what appears at the cable's Rx Aux signal.	<code>SIO4_ASYNC_CABLE_TXC_OUT_CBL_RXC</code>	This drives the signal from what appears at the cable's Rx Clock signal.	<code>SIO4_ASYNC_CABLE_TXC_OUT_OSC</code>	This drives the signal from the onboard oscillator.	<code>SIO4_ASYNC_CABLE_TXC_OUT_OSC_INV</code>	This drives the signal from the inverted form								
Value	Description																						
<code>SIO4_ASYNC_CABLE_TXC_OUT_0</code>	This drives the signal low.																						
<code>SIO4_ASYNC_CABLE_TXC_OUT_1</code>	This drives the signal high.																						
<code>SIO4_ASYNC_CABLE_TXC_OUT_CBL_RXA</code>	This drives the signal from what appears at the cable's Rx Aux signal.																						
<code>SIO4_ASYNC_CABLE_TXC_OUT_CBL_RXC</code>	This drives the signal from what appears at the cable's Rx Clock signal.																						
<code>SIO4_ASYNC_CABLE_TXC_OUT_OSC</code>	This drives the signal from the onboard oscillator.																						
<code>SIO4_ASYNC_CABLE_TXC_OUT_OSC_INV</code>	This drives the signal from the inverted form																						

	<table> <tr> <td></td><td>of the onboard oscillator.</td></tr> <tr> <td>SIO4_ASYNC_CABLE_TXC_OUT_USC_RXC</td><td>This drives the signal from what appears at the USC's Rx Clock pin.</td></tr> <tr> <td>SIO4_ASYNC_CABLE_TXC_OUT_USC_TXC</td><td>This drives the signal from what appears at the USC's Tx Clock pin. This is the default.</td></tr> </table>		of the onboard oscillator.	SIO4_ASYNC_CABLE_TXC_OUT_USC_RXC	This drives the signal from what appears at the USC's Rx Clock pin.	SIO4_ASYNC_CABLE_TXC_OUT_USC_TXC	This drives the signal from what appears at the USC's Tx Clock pin. This is the default.				
	of the onboard oscillator.										
SIO4_ASYNC_CABLE_TXC_OUT_USC_RXC	This drives the signal from what appears at the USC's Rx Clock pin.										
SIO4_ASYNC_CABLE_TXC_OUT_USC_TXC	This drives the signal from what appears at the USC's Tx Clock pin. This is the default.										
cable.txd	<p>This field specifies the configuration of the cable's Tx Data signal. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_cable_txd()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_CABLE_TXD_OUT_0</td><td>This drives the signal low.</td></tr> <tr> <td>SIO4_ASYNC_CABLE_TXD_OUT_1</td><td>This drives the signal high.</td></tr> <tr> <td>SIO4_ASYNC_CABLE_TXD_OUT_USC_TXD</td><td>This drives the signal from what appears at the USC's Tx Data pin. This is the default.</td></tr> </table>	Value	Description	SIO4_ASYNC_CABLE_TXD_OUT_0	This drives the signal low.	SIO4_ASYNC_CABLE_TXD_OUT_1	This drives the signal high.	SIO4_ASYNC_CABLE_TXD_OUT_USC_TXD	This drives the signal from what appears at the USC's Tx Data pin. This is the default.		
Value	Description										
SIO4_ASYNC_CABLE_TXD_OUT_0	This drives the signal low.										
SIO4_ASYNC_CABLE_TXD_OUT_1	This drives the signal high.										
SIO4_ASYNC_CABLE_TXD_OUT_USC_TXD	This drives the signal from what appears at the USC's Tx Data pin. This is the default.										
cable.txaux	<p>This field specifies the configuration of the cable's Tx Aux signal. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_cable_txaux()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_CABLE_TXAUX_OUT_0</td><td>This drives the signal low.</td></tr> <tr> <td>SIO4_ASYNC_CABLE_TXAUX_OUT_1</td><td>This drives the signal high.</td></tr> <tr> <td>SIO4_ASYNC_CABLE_TXAUX_OUT_OSC</td><td>This drives the signal from the onboard oscillator.</td></tr> <tr> <td>SIO4_ASYNC_CABLE_TXAUX_TRI</td><td>This tri-states the drive segment of the transceivers. This is the default.</td></tr> </table>	Value	Description	SIO4_ASYNC_CABLE_TXAUX_OUT_0	This drives the signal low.	SIO4_ASYNC_CABLE_TXAUX_OUT_1	This drives the signal high.	SIO4_ASYNC_CABLE_TXAUX_OUT_OSC	This drives the signal from the onboard oscillator.	SIO4_ASYNC_CABLE_TXAUX_TRI	This tri-states the drive segment of the transceivers. This is the default.
Value	Description										
SIO4_ASYNC_CABLE_TXAUX_OUT_0	This drives the signal low.										
SIO4_ASYNC_CABLE_TXAUX_OUT_1	This drives the signal high.										
SIO4_ASYNC_CABLE_TXAUX_OUT_OSC	This drives the signal from the onboard oscillator.										
SIO4_ASYNC_CABLE_TXAUX_TRI	This tri-states the drive segment of the transceivers. This is the default.										
cable.dcd	<p>This field specifies the cable DCD signal source when the cable signal is driven. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_cable_dcd()</code>.</p> <p>NOTE: Refer to the <code>usc.dcd</code> field (section 3.3.1.5, page 27) for affecting the cable signal's driven state.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_CABLE_DCD_OUT_0</td><td>This drives the signal low.</td></tr> <tr> <td>SIO4_ASYNC_CABLE_DCD_OUT_1</td><td>This drives the signal high.</td></tr> <tr> <td>SIO4_ASYNC_CABLE_DCD_OUT_RTS</td><td>This drives the signal from the Rx FIFO Almost Full status.</td></tr> <tr> <td>SIO4_ASYNC_CABLE_DCD_OUT_USC_DCD</td><td>This drives the signal from what appears at the USC's DCD pin. This is the default.</td></tr> </table>	Value	Description	SIO4_ASYNC_CABLE_DCD_OUT_0	This drives the signal low.	SIO4_ASYNC_CABLE_DCD_OUT_1	This drives the signal high.	SIO4_ASYNC_CABLE_DCD_OUT_RTS	This drives the signal from the Rx FIFO Almost Full status.	SIO4_ASYNC_CABLE_DCD_OUT_USC_DCD	This drives the signal from what appears at the USC's DCD pin. This is the default.
Value	Description										
SIO4_ASYNC_CABLE_DCD_OUT_0	This drives the signal low.										
SIO4_ASYNC_CABLE_DCD_OUT_1	This drives the signal high.										
SIO4_ASYNC_CABLE_DCD_OUT_RTS	This drives the signal from the Rx FIFO Almost Full status.										
SIO4_ASYNC_CABLE_DCD_OUT_USC_DCD	This drives the signal from what appears at the USC's DCD pin. This is the default.										
cable.dtr_dsr	<p>This field specifies the configuration of the cable's DTR/DSR signal. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_cable_dtr_dsr()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_CABLE_DTR_DSR_OUT_0</td><td>This drives the signal low.</td></tr> <tr> <td>SIO4_ASYNC_CABLE_DTR_DSR_OUT_1</td><td>This drives the signal high.</td></tr> <tr> <td>SIO4_ASYNC_CABLE_DTR_DSR_IN</td><td>This configures the signal as an input.</td></tr> <tr> <td>SIO4_ASYNC_CABLE_DTR_DSR_TRI</td><td>This tri-states the drive segment of the transceivers. This is the default.</td></tr> </table>	Value	Description	SIO4_ASYNC_CABLE_DTR_DSR_OUT_0	This drives the signal low.	SIO4_ASYNC_CABLE_DTR_DSR_OUT_1	This drives the signal high.	SIO4_ASYNC_CABLE_DTR_DSR_IN	This configures the signal as an input.	SIO4_ASYNC_CABLE_DTR_DSR_TRI	This tri-states the drive segment of the transceivers. This is the default.
Value	Description										
SIO4_ASYNC_CABLE_DTR_DSR_OUT_0	This drives the signal low.										
SIO4_ASYNC_CABLE_DTR_DSR_OUT_1	This drives the signal high.										
SIO4_ASYNC_CABLE_DTR_DSR_IN	This configures the signal as an input.										
SIO4_ASYNC_CABLE_DTR_DSR_TRI	This tri-states the drive segment of the transceivers. This is the default.										
cable.rts	<p>This field specifies the configuration of the cable's RTS signal. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_cable_rts()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_CABLE_RTS_OUT_0</td><td>This drives the signal low.</td></tr> <tr> <td>SIO4_ASYNC_CABLE_RTS_OUT_1</td><td>This drives the signal high.</td></tr> <tr> <td>SIO4_ASYNC_CABLE_RTS_OUT_CTS</td><td>This drives the signal from what appears at the USC's CTS pin.</td></tr> <tr> <td>SIO4_ASYNC_CABLE_RTS_OUT_RTS</td><td>This drives the signal from the Rx FIFO Almost Full status. This is the default.</td></tr> </table>	Value	Description	SIO4_ASYNC_CABLE_RTS_OUT_0	This drives the signal low.	SIO4_ASYNC_CABLE_RTS_OUT_1	This drives the signal high.	SIO4_ASYNC_CABLE_RTS_OUT_CTS	This drives the signal from what appears at the USC's CTS pin.	SIO4_ASYNC_CABLE_RTS_OUT_RTS	This drives the signal from the Rx FIFO Almost Full status. This is the default.
Value	Description										
SIO4_ASYNC_CABLE_RTS_OUT_0	This drives the signal low.										
SIO4_ASYNC_CABLE_RTS_OUT_1	This drives the signal high.										
SIO4_ASYNC_CABLE_RTS_OUT_CTS	This drives the signal from what appears at the USC's CTS pin.										
SIO4_ASYNC_CABLE_RTS_OUT_RTS	This drives the signal from the Rx FIFO Almost Full status. This is the default.										

cable. loopback	This structure configures the cable’s loopback feature.	
cable. loopback. mode	This field specifies the loopback mode. Valid values are given in the table below. The feature’s low level function is <code>sio4_async_t_cable_loopback_mode()</code> .	
	Value	Description
	SIO4_ASYNC_LOOPBACK_MODE_DISABLE	This disables loopback operation. This is the default.
	SIO4_ASYNC_LOOPBACK_MODE_EXTERNAL	This selects the external loopback mode. *
	SIO4_ASYNC_LOOPBACK_MODE_INTERNAL	This selects the internal loopback mode.
* If external loopback mode is requested but not available, then the internal loopback mode is selected.		
cable. term	This structure configures the cable’s termination feature. The operation of this feature depends on the active cable transceiver type.	
cable. term. enable	This field specifies the configuration of the transceiver’s built-in termination feature. Valid values are given in the table below. The feature’s low level function is <code>sio4_async_t_cable_term_enable()</code> .	
	Value	Description
	SIO4_ASYNC_CABLE_TERM_ENABLE_NO	The built-in termination is disabled. This is the default.
	SIO4_ASYNC_CABLE_TERM_ENABLE_YES	The built-in termination is enabled.
cable. legacy	This structure configures the cable’s legacy interface feature. These fields are utilized if the board DCE/DTE cable configuration feature is unsupported by the board or is unused.	
cable. legacy. txc	This field specifies the legacy configuration of the cable’s Tx Clock signal. Valid values are given in the table below. The feature’s low level function is <code>sio4_async_t_cable_legacy_txc()</code> .	
	Value	Description
	SIO4_ASYNC_CABLE_LEGACY_TXC_DISABLE	This disables the Tx Clock signal.
	SIO4_ASYNC_CABLE_LEGACY_TXC_BOTH	This drives the Tx Clock signal on both the upper and lower group of pins.
	SIO4_ASYNC_CABLE_LEGACY_TXC_LOW	This drives the Tx Clock signal on the lower group of pins.
SIO4_ASYNC_CABLE_LEGACY_TXC_UP	This drives the Tx Clock signal on the upper group of pins. This is the default.	
cable. legacy. txd_cts	This field specifies the legacy configuration of the cable’s Tx Data and CTS signals. Valid values are given in the table below. The feature’s low level function is <code>sio4_async_t_cable_legacy_txd_cts()</code> .	
	Value	Description
	SIO4_ASYNC_CABLE_LEGACY_TXD_CTS_BOTH	This drives the signals on both the upper and lower group of pins.
	SIO4_ASYNC_CABLE_LEGACY_TXD_CTS_LOW	This drives the signals on the lower group of pins.
	SIO4_ASYNC_CABLE_LEGACY_TXD_CTS_TRI	This tri-states the signals.
SIO4_ASYNC_CABLE_LEGACY_TXD_CTS_UP	This drives the signals on the upper group of pins. This is the default.	

cable. legacy. rxr	This field specifies the legacy configuration of the cable's Rx Clock signal. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_cable_legacy_rxc()</code> .	
	Value	Description
	SIO4_ASYNC_CABLE_LEGACY_RXC_DISABLE	This disables the Rx Clock signal.
	SIO4_ASYNC_CABLE_LEGACY_RXC_LOW	This drives the Rx Clock signal on the lower group of pins. This is the default.
cable. legacy. rxd_dcd	This field specifies the legacy configuration of the cable's Rx Data and DCD signals. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_cable_legacy_rxd_dcd()</code> .	
	Value	Description
	SIO4_ASYNC_CABLE_LEGACY_RXD_DCD_DISABLE	This disables the signals.
	SIO4_ASYNC_CABLE_LEGACY_RXD_DCD_LOW	This uses the signals as inputs from the lower group of pins. This is the default.
	SIO4_ASYNC_CABLE_LEGACY_RXD_DCD_UP	This uses the signals as inputs from the upper group of pins.

3.3.1.3. sio4_async_t.tx

This section describes the structure's transmitter configuration fields.

Field	Description	
tx	This structure configures the transmitter portion of the channel.	
tx. mode	This field specifies the transmitter's operating mode. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_tx_mode()</code> .	
	Value	Description
tx. enable	SIO4_ASYNC_TX_MODE_ASYNC	This selects the Asynchronous operating mode. This is the default and the only valid option for this library.
	This field specifies if the transmitter is to be enabled. When configuration is begun (see <code>sio4_async_set()</code> , section 3.1.7, page 10) the transmitter is initialized and disabled. The option in this field is applied towards the end of the configuration process. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_tx_enable()</code> .	
	Value	Description
	SIO4_ASYNC_TX_ENABLE_NO_AFTER	This disables the transmitter after it has finished the transmission in progress.
	SIO4_ASYNC_TX_ENABLE_NO_NOW	This disables the transmitter immediately.
	SIO4_ASYNC_TX_ENABLE_YES_NOW	This enables the transmitter immediately. This is the default.
tx. char_len	SIO4_ASYNC_TX_ENABLE_YES_W_AE	This enables the transmitter according to the state of any hardware flow control lines.
	This field specifies if the size of transmitted characters. The length specified does not include the Parity Bit, if Parity is enabled. The data bits are the lower significant bits of the byte. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_tx_char_len()</code> .	
	Value	Description
	SIO4_ASYNC_TX_CHAR_LEN_1	Characters are 1-bit in length.
	SIO4_ASYNC_TX_CHAR_LEN_2	Characters are 2-bits in length.

	<table><tr><td>SIO4_ASYNC_TX_CHAR_LEN_3</td><td>Characters are 3-bits in length.</td></tr><tr><td>SIO4_ASYNC_TX_CHAR_LEN_4</td><td>Characters are 4-bits in length.</td></tr><tr><td>SIO4_ASYNC_TX_CHAR_LEN_5</td><td>Characters are 5-bits in length.</td></tr><tr><td>SIO4_ASYNC_TX_CHAR_LEN_6</td><td>Characters are 6-bits in length.</td></tr><tr><td>SIO4_ASYNC_TX_CHAR_LEN_7</td><td>Characters are 7-bits in length.</td></tr><tr><td>SIO4_ASYNC_TX_CHAR_LEN_8</td><td>Characters are 8-bits in length. This is the default.</td></tr></table>	SIO4_ASYNC_TX_CHAR_LEN_3	Characters are 3-bits in length.	SIO4_ASYNC_TX_CHAR_LEN_4	Characters are 4-bits in length.	SIO4_ASYNC_TX_CHAR_LEN_5	Characters are 5-bits in length.	SIO4_ASYNC_TX_CHAR_LEN_6	Characters are 6-bits in length.	SIO4_ASYNC_TX_CHAR_LEN_7	Characters are 7-bits in length.	SIO4_ASYNC_TX_CHAR_LEN_8	Characters are 8-bits in length. This is the default.						
SIO4_ASYNC_TX_CHAR_LEN_3	Characters are 3-bits in length.																		
SIO4_ASYNC_TX_CHAR_LEN_4	Characters are 4-bits in length.																		
SIO4_ASYNC_TX_CHAR_LEN_5	Characters are 5-bits in length.																		
SIO4_ASYNC_TX_CHAR_LEN_6	Characters are 6-bits in length.																		
SIO4_ASYNC_TX_CHAR_LEN_7	Characters are 7-bits in length.																		
SIO4_ASYNC_TX_CHAR_LEN_8	Characters are 8-bits in length. This is the default.																		
tx. encoding	<p>This field specifies the encoding of the transmitted data. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_tx_encoding()</code>.</p> <table><tr><th>Value</th><th>Description</th></tr><tr><td>SIO4_ASYNC_TX_ENCODING_BI_MARK</td><td>This refers to Biphasic Mark encoding.</td></tr><tr><td>SIO4_ASYNC_TX_ENCODING_BI_LEVEL</td><td>This refers to Biphasic Level encoding.</td></tr><tr><td>SIO4_ASYNC_TX_ENCODING_BI_SPACE</td><td>This refers to Biphasic Space encoding.</td></tr><tr><td>SIO4_ASYNC_TX_ENCODING_D_BI_LEVEL</td><td>This refers to Differential Biphasic Level encoding.</td></tr><tr><td>SIO4_ASYNC_TX_ENCODING_NRZ</td><td>This refers to NRZ encoding. This is the default.</td></tr><tr><td>SIO4_ASYNC_TX_ENCODING_NRZB</td><td>This refers to NRZB encoding.</td></tr><tr><td>SIO4_ASYNC_TX_ENCODING_NRZI_MARK</td><td>This refers to NRZI-Mark encoding.</td></tr><tr><td>SIO4_ASYNC_TX_ENCODING_NRZI_SPACE</td><td>This refers to NRZI-Space encoding.</td></tr></table>	Value	Description	SIO4_ASYNC_TX_ENCODING_BI_MARK	This refers to Biphasic Mark encoding.	SIO4_ASYNC_TX_ENCODING_BI_LEVEL	This refers to Biphasic Level encoding.	SIO4_ASYNC_TX_ENCODING_BI_SPACE	This refers to Biphasic Space encoding.	SIO4_ASYNC_TX_ENCODING_D_BI_LEVEL	This refers to Differential Biphasic Level encoding.	SIO4_ASYNC_TX_ENCODING_NRZ	This refers to NRZ encoding. This is the default.	SIO4_ASYNC_TX_ENCODING_NRZB	This refers to NRZB encoding.	SIO4_ASYNC_TX_ENCODING_NRZI_MARK	This refers to NRZI-Mark encoding.	SIO4_ASYNC_TX_ENCODING_NRZI_SPACE	This refers to NRZI-Space encoding.
Value	Description																		
SIO4_ASYNC_TX_ENCODING_BI_MARK	This refers to Biphasic Mark encoding.																		
SIO4_ASYNC_TX_ENCODING_BI_LEVEL	This refers to Biphasic Level encoding.																		
SIO4_ASYNC_TX_ENCODING_BI_SPACE	This refers to Biphasic Space encoding.																		
SIO4_ASYNC_TX_ENCODING_D_BI_LEVEL	This refers to Differential Biphasic Level encoding.																		
SIO4_ASYNC_TX_ENCODING_NRZ	This refers to NRZ encoding. This is the default.																		
SIO4_ASYNC_TX_ENCODING_NRZB	This refers to NRZB encoding.																		
SIO4_ASYNC_TX_ENCODING_NRZI_MARK	This refers to NRZI-Mark encoding.																		
SIO4_ASYNC_TX_ENCODING_NRZI_SPACE	This refers to NRZI-Space encoding.																		
tx. bit_rate	<p>This specifies the desired transmission bit rate. During the <code>sio4_async_init()</code> call (section 3.1.3, page 9) this is set to the <code>sio4_async_init_t.tx_bit_rate</code> field value. During the call, the library will automatically determine the USC configuration needed to best produce the requested transmit bit rate. Before returning, this field is set to the bit rate that configuration will produce. The feature's low level function is <code>sio4_async_t_tx_bit_rate()</code>. The Tx Bit Rate is used when the device is being configured, but the value is not recorded for later retrieval.</p>																		
tx. idle_cond	<p>This field specifies what appears on the Tx Data cable signal while no data is being transmitted. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_tx_idle_cond()</code>.</p> <table><tr><th>Value</th><th>Description</th></tr><tr><td>SIO4_ASYNC_TX_IDLE_COND_0</td><td>The Tx Data signal is driven low.</td></tr><tr><td>SIO4_ASYNC_TX_IDLE_COND_0_1</td><td>The Tx Data signal is alternately driven low then high.</td></tr><tr><td>SIO4_ASYNC_TX_IDLE_COND_1</td><td>The Tx Data signal is driven high.</td></tr><tr><td>SIO4_ASYNC_TX_IDLE_COND_DEFAULT</td><td>The Tx Data signal is driven with the pattern that is the default for the selected serial protocol. This is the default.</td></tr><tr><td>SIO4_ASYNC_TX_IDLE_COND_MARK</td><td>The Tx Data signal is driven with the Mark state.</td></tr><tr><td>SIO4_ASYNC_TX_IDLE_COND_MARK_SPACE</td><td>The Tx Data signal is alternately driven with the Mark and Space states.</td></tr><tr><td>SIO4_ASYNC_TX_IDLE_COND_SPACE</td><td>The Tx Data signal is driven with the Space state.</td></tr></table>	Value	Description	SIO4_ASYNC_TX_IDLE_COND_0	The Tx Data signal is driven low.	SIO4_ASYNC_TX_IDLE_COND_0_1	The Tx Data signal is alternately driven low then high.	SIO4_ASYNC_TX_IDLE_COND_1	The Tx Data signal is driven high.	SIO4_ASYNC_TX_IDLE_COND_DEFAULT	The Tx Data signal is driven with the pattern that is the default for the selected serial protocol. This is the default.	SIO4_ASYNC_TX_IDLE_COND_MARK	The Tx Data signal is driven with the Mark state.	SIO4_ASYNC_TX_IDLE_COND_MARK_SPACE	The Tx Data signal is alternately driven with the Mark and Space states.	SIO4_ASYNC_TX_IDLE_COND_SPACE	The Tx Data signal is driven with the Space state.		
Value	Description																		
SIO4_ASYNC_TX_IDLE_COND_0	The Tx Data signal is driven low.																		
SIO4_ASYNC_TX_IDLE_COND_0_1	The Tx Data signal is alternately driven low then high.																		
SIO4_ASYNC_TX_IDLE_COND_1	The Tx Data signal is driven high.																		
SIO4_ASYNC_TX_IDLE_COND_DEFAULT	The Tx Data signal is driven with the pattern that is the default for the selected serial protocol. This is the default.																		
SIO4_ASYNC_TX_IDLE_COND_MARK	The Tx Data signal is driven with the Mark state.																		
SIO4_ASYNC_TX_IDLE_COND_MARK_SPACE	The Tx Data signal is alternately driven with the Mark and Space states.																		
SIO4_ASYNC_TX_IDLE_COND_SPACE	The Tx Data signal is driven with the Space state.																		
tx. parity	<p>This structure configures the transmitter's use of Parity checking.</p>																		
tx. parity. enable	<p>This field enables or disables the use of Parity. When used, the Parity Bit appears to the immediate left of the most significant data bit. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_tx_parity_enable()</code>.</p> <table><tr><th>Value</th><th>Description</th></tr><tr><td>SIO4_ASYNC_TX_PARITY_ENABLE_NO</td><td>Do <u>not</u> generate a Parity bit. This is the default.</td></tr><tr><td>SIO4_ASYNC_TX_PARITY_ENABLE_YES</td><td><u>Do</u> generate a Parity bit.</td></tr></table>	Value	Description	SIO4_ASYNC_TX_PARITY_ENABLE_NO	Do <u>not</u> generate a Parity bit. This is the default.	SIO4_ASYNC_TX_PARITY_ENABLE_YES	<u>Do</u> generate a Parity bit.												
Value	Description																		
SIO4_ASYNC_TX_PARITY_ENABLE_NO	Do <u>not</u> generate a Parity bit. This is the default.																		
SIO4_ASYNC_TX_PARITY_ENABLE_YES	<u>Do</u> generate a Parity bit.																		

tx. parity. type	<p>This field specifies the type of Parity to use, when its use is enabled. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_tx_parity_type()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_TX_PARITY_TYPE_EVEN</td><td>This specifies Even Parity. This is the default.</td></tr> <tr> <td>SIO4_ASYNC_TX_PARITY_TYPE_ODD</td><td>This specifies Odd Parity.</td></tr> <tr> <td>SIO4_ASYNC_TX_PARITY_TYPE_ONE</td><td>This specifies One Parity (the parity bit is always set).</td></tr> <tr> <td>SIO4_ASYNC_TX_PARITY_TYPE_ZERO</td><td>This specifies Zero Parity (the parity bit is always clear).</td></tr> </table>	Value	Description	SIO4_ASYNC_TX_PARITY_TYPE_EVEN	This specifies Even Parity. This is the default.	SIO4_ASYNC_TX_PARITY_TYPE_ODD	This specifies Odd Parity.	SIO4_ASYNC_TX_PARITY_TYPE_ONE	This specifies One Parity (the parity bit is always set).	SIO4_ASYNC_TX_PARITY_TYPE_ZERO	This specifies Zero Parity (the parity bit is always clear).
Value	Description										
SIO4_ASYNC_TX_PARITY_TYPE_EVEN	This specifies Even Parity. This is the default.										
SIO4_ASYNC_TX_PARITY_TYPE_ODD	This specifies Odd Parity.										
SIO4_ASYNC_TX_PARITY_TYPE_ONE	This specifies One Parity (the parity bit is always set).										
SIO4_ASYNC_TX_PARITY_TYPE_ZERO	This specifies Zero Parity (the parity bit is always clear).										
tx. fifo	This structure configures the transmitter's FIFO parameters.										
tx. fifo. size	This field is filled in by the <code>sio4_async_init()</code> call (section 3.1.3, page 9) with the size of the channel's Tx FIFO. This is offered for informational purposes only. The feature's low level function is <code>sio4_async_t_tx_fifo_size()</code> .										
tx. fifo. ae	This field specifies the Tx FIFO Almost Empty setting. The Tx FIFO Almost Empty status is asserted (goes low) when the Tx FIFO contains this number of values, or fewer. The valid value range is from zero to 0xFFFF. The default is 0x7. The feature's low level function is <code>sio4_async_t_tx_fifo_ae()</code> .										
tx. fifo. af	This field specifies the Tx FIFO Almost Full setting. The Tx FIFO Almost Full status is asserted (goes low) when the Tx FIFO contains this number of free spaces, or fewer. The valid value range is from zero to 0xFFFF. The default is 0x7. The feature's low level function is <code>sio4_async_t_tx_fifo_af()</code> .										
tx. fifo. empty_cfg	<p>This field configures the transmitter's reaction to the Tx FIFO becoming empty. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_tx_fifo_empty_cfg()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_TX_FIFO_EMPTY_CFG_IGNORE</td><td>This specifies that the condition is to be ignored. This is the default.</td></tr> <tr> <td>SIO4_ASYNC_TX_FIFO_EMPTY_CFG_TX_OFF</td><td>This specifies that the transmitter be disabled when the condition occurs.</td></tr> </table>	Value	Description	SIO4_ASYNC_TX_FIFO_EMPTY_CFG_IGNORE	This specifies that the condition is to be ignored. This is the default.	SIO4_ASYNC_TX_FIFO_EMPTY_CFG_TX_OFF	This specifies that the transmitter be disabled when the condition occurs.				
Value	Description										
SIO4_ASYNC_TX_FIFO_EMPTY_CFG_IGNORE	This specifies that the condition is to be ignored. This is the default.										
SIO4_ASYNC_TX_FIFO_EMPTY_CFG_TX_OFF	This specifies that the transmitter be disabled when the condition occurs.										
tx. fifo. space_cfg	<p>This field configures the FIFO space allocation between the transmitter and the receiver when the Tx FIFO and Rx FIFO are of different sizes. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_tx_fifo_space_cfg()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_TX_FIFO_SPACE_CFG_RX_2X</td><td>This specifies that the Rx FIFO be twice as large as the Tx FIFO. This is the default.</td></tr> <tr> <td>SIO4_ASYNC_TX_FIFO_SPACE_CFG_TX_2X</td><td>This specifies that the Tx FIFO be twice as large as the Rx FIFO.</td></tr> </table>	Value	Description	SIO4_ASYNC_TX_FIFO_SPACE_CFG_RX_2X	This specifies that the Rx FIFO be twice as large as the Tx FIFO. This is the default.	SIO4_ASYNC_TX_FIFO_SPACE_CFG_TX_2X	This specifies that the Tx FIFO be twice as large as the Rx FIFO.				
Value	Description										
SIO4_ASYNC_TX_FIFO_SPACE_CFG_RX_2X	This specifies that the Rx FIFO be twice as large as the Tx FIFO. This is the default.										
SIO4_ASYNC_TX_FIFO_SPACE_CFG_TX_2X	This specifies that the Tx FIFO be twice as large as the Rx FIFO.										
tx. io	This structure configures the transmitter's software settings. These settings are used during <code>sio4_async_write()</code> calls (see section 3.1.11, page 12).										
tx. io. mode	<p>This field selects the mechanism used to transfer data from host memory to the channel's Tx FIFO. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_tx_io_mode()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_TX_IO_MODE_DMA</td><td>This selects DMA mode transfers. *</td></tr> <tr> <td>SIO4_ASYNC_TX_IO_MODE_DMDMA</td><td>This selects Demand Mode DMA mode transfers. *</td></tr> <tr> <td>SIO4_ASYNC_TX_IO_MODE_PIO</td><td>This selects PIO mode transfers. This is the default.</td></tr> </table> <p>* The SIO4 has only two DMA engines. A DMA or DMDMA transfer request will fail if both DMA engines are already in use by other SIO4 channels.</p>	Value	Description	SIO4_ASYNC_TX_IO_MODE_DMA	This selects DMA mode transfers. *	SIO4_ASYNC_TX_IO_MODE_DMDMA	This selects Demand Mode DMA mode transfers. *	SIO4_ASYNC_TX_IO_MODE_PIO	This selects PIO mode transfers. This is the default.		
Value	Description										
SIO4_ASYNC_TX_IO_MODE_DMA	This selects DMA mode transfers. *										
SIO4_ASYNC_TX_IO_MODE_DMDMA	This selects Demand Mode DMA mode transfers. *										
SIO4_ASYNC_TX_IO_MODE_PIO	This selects PIO mode transfers. This is the default.										
tx.	This field specifies the threshold for write request sizes that force the use of PIO mode. If a write										

io. pio_thresh	request is this size or smaller, then the transfer will automatically use PIO. The valid range is any non-negative value. The default is 64. The feature's low level function is <code>sio4_async_t_tx_io_pio_thresh()</code> .						
tx. io. timeout	This field specifies the maximum duration of write requests to the driver. The valid range is from zero to 3600. The units are seconds. The default is 10 seconds. The value zero should be used with PIO mode only as it tells the driver to write as much data as possible to the Tx FIFO, but not to wait for additional free space. DMA and DMDMA requests always require a sleep to wait for the hardware to complete the transfer. The feature's low level function is <code>sio4_async_t_tx_io_timeout()</code> .						
tx. io. overrun	<p>This field tells the driver if it is to check for Tx FIFO overrun conditions before proceeding with write requests. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_tx_io_overrun()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_TX_IO_OVERRUN_CHECK</td><td>This specifies that the driver <u>should</u> check for overrun conditions. This is the default.</td></tr> <tr> <td>SIO4_ASYNC_TX_IO_OVERRUN_IGNORE</td><td>This specifies that the driver should <u>not</u> check for overrun conditions.</td></tr> </table>	Value	Description	SIO4_ASYNC_TX_IO_OVERRUN_CHECK	This specifies that the driver <u>should</u> check for overrun conditions. This is the default.	SIO4_ASYNC_TX_IO_OVERRUN_IGNORE	This specifies that the driver should <u>not</u> check for overrun conditions.
Value	Description						
SIO4_ASYNC_TX_IO_OVERRUN_CHECK	This specifies that the driver <u>should</u> check for overrun conditions. This is the default.						
SIO4_ASYNC_TX_IO_OVERRUN_IGNORE	This specifies that the driver should <u>not</u> check for overrun conditions.						

3.3.1.4. sio4_async_t.rx

This section describes the structure's receiver configuration fields.

Field	Description										
rx	This structure configures the receiver portion of the channel.										
rx. mode	<p>This field specifies the receiver's operating mode. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_rx_mode()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_RX_MODE_ASYNC</td><td>This selects the Asynchronous operating mode. This is the default and the only valid option for this library.</td></tr> </table>	Value	Description	SIO4_ASYNC_RX_MODE_ASYNC	This selects the Asynchronous operating mode. This is the default and the only valid option for this library.						
Value	Description										
SIO4_ASYNC_RX_MODE_ASYNC	This selects the Asynchronous operating mode. This is the default and the only valid option for this library.										
rx. enable	<p>This field specifies if the receiver is to be enabled. When configuration is begun (see <code>sio4_async_set()</code>, section 3.1.7, page 10) the receiver is initialized and disabled. The option in this field is applied towards the end of the configuration process. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_rx_enable()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_RX_ENABLE_NO_AFTER</td><td>This disables the receiver after it has finished the reception in progress.</td></tr> <tr> <td>SIO4_ASYNC_RX_ENABLE_NO_NOW</td><td>This disables the receiver immediately.</td></tr> <tr> <td>SIO4_ASYNC_RX_ENABLE_YES_NOW</td><td>This enables the receiver immediately. This is the default.</td></tr> <tr> <td>SIO4_ASYNC_RX_ENABLE_YES_W_AE</td><td>This enables the receiver according to the state of any hardware flow control lines.</td></tr> </table>	Value	Description	SIO4_ASYNC_RX_ENABLE_NO_AFTER	This disables the receiver after it has finished the reception in progress.	SIO4_ASYNC_RX_ENABLE_NO_NOW	This disables the receiver immediately.	SIO4_ASYNC_RX_ENABLE_YES_NOW	This enables the receiver immediately. This is the default.	SIO4_ASYNC_RX_ENABLE_YES_W_AE	This enables the receiver according to the state of any hardware flow control lines.
Value	Description										
SIO4_ASYNC_RX_ENABLE_NO_AFTER	This disables the receiver after it has finished the reception in progress.										
SIO4_ASYNC_RX_ENABLE_NO_NOW	This disables the receiver immediately.										
SIO4_ASYNC_RX_ENABLE_YES_NOW	This enables the receiver immediately. This is the default.										
SIO4_ASYNC_RX_ENABLE_YES_W_AE	This enables the receiver according to the state of any hardware flow control lines.										
rx. char_len	<p>This field specifies the size of received characters. The length specified does not include the Parity Bit, if Parity is enabled. The data bits are the lower significant bits of the byte. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_rx_char_len()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_RX_CHAR_LEN 1</td><td>Characters are 1-bit in length.</td></tr> <tr> <td>SIO4_ASYNC_RX_CHAR_LEN 2</td><td>Characters are 2-bits in length.</td></tr> <tr> <td>SIO4_ASYNC_RX_CHAR_LEN 3</td><td>Characters are 3-bits in length.</td></tr> <tr> <td>SIO4_ASYNC_RX_CHAR_LEN 4</td><td>Characters are 4-bits in length.</td></tr> </table>	Value	Description	SIO4_ASYNC_RX_CHAR_LEN 1	Characters are 1-bit in length.	SIO4_ASYNC_RX_CHAR_LEN 2	Characters are 2-bits in length.	SIO4_ASYNC_RX_CHAR_LEN 3	Characters are 3-bits in length.	SIO4_ASYNC_RX_CHAR_LEN 4	Characters are 4-bits in length.
Value	Description										
SIO4_ASYNC_RX_CHAR_LEN 1	Characters are 1-bit in length.										
SIO4_ASYNC_RX_CHAR_LEN 2	Characters are 2-bits in length.										
SIO4_ASYNC_RX_CHAR_LEN 3	Characters are 3-bits in length.										
SIO4_ASYNC_RX_CHAR_LEN 4	Characters are 4-bits in length.										

	SIO4_ASYNC_RX_CHAR_LEN_5	Characters are 5-bits in length.
	SIO4_ASYNC_RX_CHAR_LEN_6	Characters are 6-bits in length.
	SIO4_ASYNC_RX_CHAR_LEN_7	Characters are 7-bits in length.
	SIO4_ASYNC_RX_CHAR_LEN_8	Characters are 8-bits in length. This is the default.
rx. encoding	This field specifies the encoding of the received data. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_rx_encoding()</code> .	
	Value	Description
	SIO4_ASYNC_RX_ENCODING_BI_MARK	This refers to Biphase Mark encoding.
	SIO4_ASYNC_RX_ENCODING_BI_LEVEL	This refers to Biphase Level encoding.
	SIO4_ASYNC_RX_ENCODING_BI_SPACE	This refers to Biphase Space encoding.
	SIO4_ASYNC_RX_ENCODING_D_BI_LEVEL	This refers to Differential Biphase Level encoding.
	SIO4_ASYNC_RX_ENCODING_NRZ	This refers to NRZ encoding. This is the default.
	SIO4_ASYNC_RX_ENCODING_NRZB	This refers to NRZB encoding.
	SIO4_ASYNC_RX_ENCODING_NRZI_MARK	This refers to NRZI-Mark encoding.
	SIO4_ASYNC_RX_ENCODING_NRZI_SPACE	This refers to NRZI-Space encoding.
rx. bit_rate	This specifies the desired receive bit rate. During the <code>sio4_async_init()</code> call (section 3.1.3, page 9) this is set to the <code>sio4_async_init_t.rx_bit_rate</code> field provided to the call. During the call, the library will automatically determine the USC configuration needed to best produce the requested receive bit rate. Before returning, this field is set to the bit rate that configuration will produce. The feature's low level function is <code>sio4_async_t_rx_bit_rate()</code> . The Tx Bit Rate is used when the device is being configured, but the value is not recorded for later retrieval.	
rx. sync_byte	This specifies the value to be compared to received data as the data enters the Rx FIFO (the one outside the USC). This comparison can be used for interrupt generation. Valid values are from zero to 0xFF. The default is zero. The feature's low level function is <code>sio4_async_t_rx_sync_byte()</code> .	
rx. status_word	This field controls whether the firmware will place the USC Receive Control/Status Register in the Rx FIFO along with the received data. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_rx_status_word()</code> .	
	Value	Description
	SIO4_ASYNC_RX_STATUS_WORD_DISABLE	The RCSR is <u>not</u> placed in the Rx FIFO. This is the default.
	SIO4_ASYNC_RX_STATUS_WORD_ENABLE	The RCSR <u>is</u> placed in the Rx FIFO.
rx. parity	This structure configures the receiver's use of Parity checking.	
rx. parity. enable	This field enables or disables the use of Parity. When enabled, the character size does not include the Parity Bit. When used, the Parity Bit appears to the immediate left of the most significant data bit. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_rx_parity_enable()</code> .	
	Value	Description
	SIO4_ASYNC_RX_PARITY_ENABLE_NO	Do <u>not</u> generate a Parity bit. This is the default.
	SIO4_ASYNC_RX_PARITY_ENABLE_YES	<u>Do</u> generate a Parity bit.
rx. parity. type	This field specifies the type of Parity to use, when its use is enabled. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_rx_parity_type()</code> .	
	Value	Description
	SIO4_ASYNC_RX_PARITY_TYPE_EVEN	This specifies Even Parity. This is the default.

	SIO4_ASYNC_RX_PARITY_TYPE_ODD	This specifies Odd Parity.
	SIO4_ASYNC_RX_PARITY_TYPE_ONE	This specifies One Parity (the parity bit is always set).
	SIO4_ASYNC_RX_PARITY_TYPE_ZERO	This specifies Zero Parity (the parity bit is always clear).
rx. fifo	This structure configures the receiver's FIFO parameters.	
rx. fifo. size	This field is filled in by the <code>sio4_async_init()</code> call (section 3.1.3, page 9) with the size of the channel's Rx FIFO. This is offered for informational purposes only. The feature's low level function is <code>sio4_async_t_rx_fifo_size()</code> .	
rx. fifo. ae	This field specifies the Rx FIFO Almost Empty setting. The Rx FIFO Almost Empty status is asserted (goes low) when the Rx FIFO contain this number of values, or fewer. The valid value range is from zero to 0xFFFF. The default is 0x7. The feature's low level function is <code>sio4_async_t_rx_fifo_ae()</code> .	
rx. fifo. af	This field specifies the Rx FIFO Almost Full setting. The Rx FIFO Almost Full status is asserted (goes low) when the Rx FIFO contain this number of values, or fewer. The valid value range is from zero to 0xFFFF. The default is 0x7. The feature's low level function is <code>sio4_async_t_rx_fifo_af()</code> .	
rx. fifo. full_cfg	This field configures the receiver's reaction to the Rx FIFO becoming full. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_rx_fifo_full_cfg()</code> . This field refers to the channel specific setting, when supported. The corresponding global setting is not supported by the Asynchronous Protocol Library because it affects channels other than the one being accessed. The global setting must be handled separately by the application.	
	Value	Description
	SIO4_ASYNC_RX_FIFO_FULL_CFG_DISABLE	This specifies that the receiver be disabled when the condition occurs.
	SIO4_ASYNC_RX_FIFO_FULL_CFG_OVER	This specifies that the condition produce an overflow. This is the default.
rx. io	This structure configures the receiver's software settings. These settings are used during <code>sio4_async_read()</code> calls (section 3.1.7, page 10).	
rx. io. mode	This field selects the mechanism used to transfer data from the channel's Rx FIFO to host memory. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_rx_io_mode()</code> .	
	Value	Description
	SIO4_ASYNC_RX_IO_MODE_DMA	This selects DMA mode transfers. *
	SIO4_ASYNC_RX_IO_MODE_DMDMA	This selects Demand Mode DMA mode transfers. *
	SIO4_ASYNC_RX_IO_MODE_PIO	This selects PIO mode transfers. This is the default.
* The SIO4 has only two DMA engines. A DMA or DMDMA transfer request will fail if both DMA engines are already in use by other SIO4 channels.		
rx. io. pio_thresh	This field specifies the threshold for read request sizes that force the use of PIO mode. If a read request is this size or smaller, then the transfer will automatically use PIO. The valid range is any non-negative value. The default is 64. The feature's low level function is <code>sio4_async_t_rx_io_pio_thresh()</code> .	
rx. io. timeout	This field specifies the maximum duration of write requests. The valid range is from zero to 3600. The units are seconds. The default is 10 seconds. The value zero should be used with PIO mode only as it tells the driver to read as much data as possible from the Rx FIFO, but not to wait for additional data. DMA and DMDMA requests always require a sleep to wait for the hardware to complete the transfer. The feature's low level function is <code>sio4_async_t_rx_io_timeout()</code> .	

rx. io. overrun	<p>This field tells the driver if it is to check for Rx FIFO overrun conditions before proceeding with read requests. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_rx_io_overrun()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_RX_IO_OVERRUN_CHECK</td><td>This specifies that the driver check for overrun conditions. This is the default.</td></tr> <tr> <td>SIO4_ASYNC_RX_IO_OVERRUN_IGNORE</td><td>This specifies that the driver <u>not</u> check for overrun conditions.</td></tr> </table>	Value	Description	SIO4_ASYNC_RX_IO_OVERRUN_CHECK	This specifies that the driver check for overrun conditions. This is the default.	SIO4_ASYNC_RX_IO_OVERRUN_IGNORE	This specifies that the driver <u>not</u> check for overrun conditions.
Value	Description						
SIO4_ASYNC_RX_IO_OVERRUN_CHECK	This specifies that the driver check for overrun conditions. This is the default.						
SIO4_ASYNC_RX_IO_OVERRUN_IGNORE	This specifies that the driver <u>not</u> check for overrun conditions.						
rx. io. underrun	<p>This field tells the driver if it is to check for Rx FIFO underrun conditions before proceeding with read requests. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_rx_io_underrun()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_RX_IO_UNDERRUN_CHECK</td><td>This specifies that the driver check for underrun conditions. This is the default.</td></tr> <tr> <td>SIO4_ASYNC_RX_IO_UNDERRUN_IGNORE</td><td>This specifies that the driver <u>not</u> check for underrun conditions.</td></tr> </table>	Value	Description	SIO4_ASYNC_RX_IO_UNDERRUN_CHECK	This specifies that the driver check for underrun conditions. This is the default.	SIO4_ASYNC_RX_IO_UNDERRUN_IGNORE	This specifies that the driver <u>not</u> check for underrun conditions.
Value	Description						
SIO4_ASYNC_RX_IO_UNDERRUN_CHECK	This specifies that the driver check for underrun conditions. This is the default.						
SIO4_ASYNC_RX_IO_UNDERRUN_IGNORE	This specifies that the driver <u>not</u> check for underrun conditions.						
rx. time_stamp	This structure configures the receiver's Time Stamp settings.						
rx. time_stamp. enable	<p>This field enables or disables the channel's use of the Time Stamp feature. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_rx_time_stamp_enable()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_RX_TIME_STAMP_ENABLE_NO</td><td>Do <u>not</u> use the Time Stamp feature. This is the default.</td></tr> <tr> <td>SIO4_ASYNC_RX_TIME_STAMP_ENABLE_YES</td><td><u>Do</u> use the Time Stamp feature.</td></tr> </table>	Value	Description	SIO4_ASYNC_RX_TIME_STAMP_ENABLE_NO	Do <u>not</u> use the Time Stamp feature. This is the default.	SIO4_ASYNC_RX_TIME_STAMP_ENABLE_YES	<u>Do</u> use the Time Stamp feature.
Value	Description						
SIO4_ASYNC_RX_TIME_STAMP_ENABLE_NO	Do <u>not</u> use the Time Stamp feature. This is the default.						
SIO4_ASYNC_RX_TIME_STAMP_ENABLE_YES	<u>Do</u> use the Time Stamp feature.						
rx. time_stamp. clk_src	<p>This field selects the Time Stamp clock source. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_rx_time_stamp_clk_src()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_RX_TIME_STAMP_CLK_SRC_EXT</td><td>Use the board's external TTL clock source. *</td></tr> <tr> <td>SIO4_ASYNC_RX_TIME_STAMP_CLK_SRC_INT</td><td>Use the board's internal 1us clock. This is the default. *</td></tr> </table> <p>* All four channels on the SIO4 use the same clock source.</p>	Value	Description	SIO4_ASYNC_RX_TIME_STAMP_CLK_SRC_EXT	Use the board's external TTL clock source. *	SIO4_ASYNC_RX_TIME_STAMP_CLK_SRC_INT	Use the board's internal 1us clock. This is the default. *
Value	Description						
SIO4_ASYNC_RX_TIME_STAMP_CLK_SRC_EXT	Use the board's external TTL clock source. *						
SIO4_ASYNC_RX_TIME_STAMP_CLK_SRC_INT	Use the board's internal 1us clock. This is the default. *						

3.3.1.5. sio4_async_t.usc

This section describes the structure's USC configuration fields.

Field	Description										
usc	This structure configures the remaining USC portion of the channel.										
usc. mode	<p>This field specifies the USC's overall operating mode. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_mode()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_USC_MODE_AUTO_ECHO</td><td>This is the USC's Auto Echo mode.</td></tr> <tr> <td>SIO4_ASYNC_USC_MODE_LOOPBACK_EXT</td><td>This is the USC's external loopback mode.</td></tr> <tr> <td>SIO4_ASYNC_USC_MODE_LOOPBACK_INT</td><td>This is the USC's internal loopback mode.</td></tr> <tr> <td>SIO4_ASYNC_USC_MODE_NORMAL</td><td>This is the USC's normal operating mode. This is the default.</td></tr> </table>	Value	Description	SIO4_ASYNC_USC_MODE_AUTO_ECHO	This is the USC's Auto Echo mode.	SIO4_ASYNC_USC_MODE_LOOPBACK_EXT	This is the USC's external loopback mode.	SIO4_ASYNC_USC_MODE_LOOPBACK_INT	This is the USC's internal loopback mode.	SIO4_ASYNC_USC_MODE_NORMAL	This is the USC's normal operating mode. This is the default.
Value	Description										
SIO4_ASYNC_USC_MODE_AUTO_ECHO	This is the USC's Auto Echo mode.										
SIO4_ASYNC_USC_MODE_LOOPBACK_EXT	This is the USC's external loopback mode.										
SIO4_ASYNC_USC_MODE_LOOPBACK_INT	This is the USC's internal loopback mode.										
SIO4_ASYNC_USC_MODE_NORMAL	This is the USC's normal operating mode. This is the default.										
usc. txd	This field configures the operation of the USC's Tx Data pin. Valid values are given in the table										

	<p>below. The feature's low level function is <code>sio4_async_t_usc_txd()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_USC_TXD_OUT_0</td><td>The pin is driven low.</td></tr> <tr> <td>SIO4_ASYNC_USC_TXD_OUT_1</td><td>The pin is driven high.</td></tr> <tr> <td>SIO4_ASYNC_USC_TXD_OUT_TXD</td><td>The pin is driven from the transmitter's Tx Data signal. This is the default.</td></tr> <tr> <td>SIO4_ASYNC_USC_TXD_TRI</td><td>The pin is tri-stated.</td></tr> </table>	Value	Description	SIO4_ASYNC_USC_TXD_OUT_0	The pin is driven low.	SIO4_ASYNC_USC_TXD_OUT_1	The pin is driven high.	SIO4_ASYNC_USC_TXD_OUT_TXD	The pin is driven from the transmitter's Tx Data signal. This is the default.	SIO4_ASYNC_USC_TXD_TRI	The pin is tri-stated.		
Value	Description												
SIO4_ASYNC_USC_TXD_OUT_0	The pin is driven low.												
SIO4_ASYNC_USC_TXD_OUT_1	The pin is driven high.												
SIO4_ASYNC_USC_TXD_OUT_TXD	The pin is driven from the transmitter's Tx Data signal. This is the default.												
SIO4_ASYNC_USC_TXD_TRI	The pin is tri-stated.												
usc. cts	<p>This field configures the operation of the USC's CTS pin. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_cts()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_USC_CTS_OUT_0</td><td>The pin is driven low.</td></tr> <tr> <td>SIO4_ASYNC_USC_CTS_OUT_1</td><td>The pin is driven high.</td></tr> <tr> <td>SIO4_ASYNC_USC_CTS_IN_CBL_CTS</td><td>The pin is an input driver from the cable's CTS signal.</td></tr> <tr> <td>SIO4_ASYNC_USC_CTS_TRI</td><td>The pin is tri-stated. This is the default.</td></tr> </table>	Value	Description	SIO4_ASYNC_USC_CTS_OUT_0	The pin is driven low.	SIO4_ASYNC_USC_CTS_OUT_1	The pin is driven high.	SIO4_ASYNC_USC_CTS_IN_CBL_CTS	The pin is an input driver from the cable's CTS signal.	SIO4_ASYNC_USC_CTS_TRI	The pin is tri-stated. This is the default.		
Value	Description												
SIO4_ASYNC_USC_CTS_OUT_0	The pin is driven low.												
SIO4_ASYNC_USC_CTS_OUT_1	The pin is driven high.												
SIO4_ASYNC_USC_CTS_IN_CBL_CTS	The pin is an input driver from the cable's CTS signal.												
SIO4_ASYNC_USC_CTS_TRI	The pin is tri-stated. This is the default.												
usc. cts_legacy	<p>This field configures the operation of the USC's CTS pin for legacy mode cable interface configurations. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_cts_legacy()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_USC_TX_CTS_LEG_IN</td><td>The pin operates as an input. This is the default.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_CTS_LEG_OUT_0</td><td>The pin operates as an output driven low.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_CTS_LEG_OUT_1</td><td>The pin operates as an output driven high.</td></tr> </table>	Value	Description	SIO4_ASYNC_USC_TX_CTS_LEG_IN	The pin operates as an input. This is the default.	SIO4_ASYNC_USC_TX_CTS_LEG_OUT_0	The pin operates as an output driven low.	SIO4_ASYNC_USC_TX_CTS_LEG_OUT_1	The pin operates as an output driven high.				
Value	Description												
SIO4_ASYNC_USC_TX_CTS_LEG_IN	The pin operates as an input. This is the default.												
SIO4_ASYNC_USC_TX_CTS_LEG_OUT_0	The pin operates as an output driven low.												
SIO4_ASYNC_USC_TX_CTS_LEG_OUT_1	The pin operates as an output driven high.												
usc. dcd	<p>This field configures the operation of the USC's DCD pin. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_dcd()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_USC_DCD_DISABLE</td><td>The pin is disabled. This is the default.</td></tr> <tr> <td>SIO4_ASYNC_USC_DCD_IN_DCD_CBL_DCD</td><td>The pin is an input for the receiver's DCD function and is driven from the cable's DCD signal.</td></tr> <tr> <td>SIO4_ASYNC_USC_DCD_IN_SYNC_CBL_DCD</td><td>The pin is an input for the receiver's SYNC function and is driven from the cable's DCD signal.</td></tr> <tr> <td>SIO4_ASYNC_USC_DCD_OUT_0</td><td>The pin is driven low. *</td></tr> <tr> <td>SIO4_ASYNC_USC_DCD_OUT_1</td><td>The pin is driven high. *</td></tr> </table> <p>* This option enables the cable DCD signal to be driven, though the <code>cable.dcd</code> field (section 3.3.1.2, page 18) may configure the cable to output an alternate signal.</p>	Value	Description	SIO4_ASYNC_USC_DCD_DISABLE	The pin is disabled. This is the default.	SIO4_ASYNC_USC_DCD_IN_DCD_CBL_DCD	The pin is an input for the receiver's DCD function and is driven from the cable's DCD signal.	SIO4_ASYNC_USC_DCD_IN_SYNC_CBL_DCD	The pin is an input for the receiver's SYNC function and is driven from the cable's DCD signal.	SIO4_ASYNC_USC_DCD_OUT_0	The pin is driven low. *	SIO4_ASYNC_USC_DCD_OUT_1	The pin is driven high. *
Value	Description												
SIO4_ASYNC_USC_DCD_DISABLE	The pin is disabled. This is the default.												
SIO4_ASYNC_USC_DCD_IN_DCD_CBL_DCD	The pin is an input for the receiver's DCD function and is driven from the cable's DCD signal.												
SIO4_ASYNC_USC_DCD_IN_SYNC_CBL_DCD	The pin is an input for the receiver's SYNC function and is driven from the cable's DCD signal.												
SIO4_ASYNC_USC_DCD_OUT_0	The pin is driven low. *												
SIO4_ASYNC_USC_DCD_OUT_1	The pin is driven high. *												
usc. dcd_legacy	<p>This field configures the operation of the USC's DCD pin for legacy mode cable interface configurations. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_dcd_legacy()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_USC_DCD_LEG_IN_DCD</td><td>The pin operates as a DCD input. This is the default.</td></tr> <tr> <td>SIO4_ASYNC_USC_DCD_LEG_IN_SYNC</td><td>The pin operates as a SYNC input.</td></tr> <tr> <td>SIO4_ASYNC_USC_DCD_LEG_OUT_0</td><td>The pin operates as an output driven low.</td></tr> <tr> <td>SIO4_ASYNC_USC_DCD_LEG_OUT_1</td><td>The pin operates as an output driven high.</td></tr> </table>	Value	Description	SIO4_ASYNC_USC_DCD_LEG_IN_DCD	The pin operates as a DCD input. This is the default.	SIO4_ASYNC_USC_DCD_LEG_IN_SYNC	The pin operates as a SYNC input.	SIO4_ASYNC_USC_DCD_LEG_OUT_0	The pin operates as an output driven low.	SIO4_ASYNC_USC_DCD_LEG_OUT_1	The pin operates as an output driven high.		
Value	Description												
SIO4_ASYNC_USC_DCD_LEG_IN_DCD	The pin operates as a DCD input. This is the default.												
SIO4_ASYNC_USC_DCD_LEG_IN_SYNC	The pin operates as a SYNC input.												
SIO4_ASYNC_USC_DCD_LEG_OUT_0	The pin operates as an output driven low.												
SIO4_ASYNC_USC_DCD_LEG_OUT_1	The pin operates as an output driven high.												
usc. tx	This structure configures the remaining USC transmitter settings.												
usc. tx.	This field configures the clock divider rate for the USC transmitter. This corresponds to the oversampling rate used by the receiver. Valid values are given in the table below. The feature's												

clk_rate	<p>low level function is <code>sio4_async_t_usc_tx_clk_rate()</code>.</p> <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>SIO4_ASYNC_USC_TX_CLK_RATE_16X</td><td>Divide the source by 16.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_CLK_RATE_32X</td><td>Divide the source by 32.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_CLK_RATE_64X</td><td>Divide the source by 64. *</td></tr> </tbody> </table> <p>* This is the hard coded initialization default, which may subsequently be modified as needed to produce the user specified Tx bit rate.</p>	Value	Description	SIO4_ASYNC_USC_TX_CLK_RATE_16X	Divide the source by 16.	SIO4_ASYNC_USC_TX_CLK_RATE_32X	Divide the source by 32.	SIO4_ASYNC_USC_TX_CLK_RATE_64X	Divide the source by 64. *																				
Value	Description																												
SIO4_ASYNC_USC_TX_CLK_RATE_16X	Divide the source by 16.																												
SIO4_ASYNC_USC_TX_CLK_RATE_32X	Divide the source by 32.																												
SIO4_ASYNC_USC_TX_CLK_RATE_64X	Divide the source by 64. *																												
usc. tx. clk_src	<p>This field configures the source for the USC transmitter clock. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_tx_clk_src()</code>.</p> <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>SIO4_ASYNC_USC_TX_CLK_SRC_BRG0</td><td>Select Baud Rate Generator 0.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_CLK_SRC_BRG1</td><td>Select Baud Rate Generator 1.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_CLK_SRC_CTR0</td><td>Select Counter 0.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_CLK_SRC_CTR1</td><td>Select Counter 1.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_CLK_SRC_DISABLE</td><td>Disable the transmitter.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_CLK_SRC_DPLL</td><td>Select the DPLL.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_CLK_SRC_RXC_PIN</td><td>Select the Rx Clock pin. *</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_CLK_SRC_TXC_PIN</td><td>Select the Tx Clock pin.</td></tr> </tbody> </table> <p>* This is the hard coded initialization default, which may subsequently be modified as needed to produce the user specified Tx bit rate.</p>	Value	Description	SIO4_ASYNC_USC_TX_CLK_SRC_BRG0	Select Baud Rate Generator 0.	SIO4_ASYNC_USC_TX_CLK_SRC_BRG1	Select Baud Rate Generator 1.	SIO4_ASYNC_USC_TX_CLK_SRC_CTR0	Select Counter 0.	SIO4_ASYNC_USC_TX_CLK_SRC_CTR1	Select Counter 1.	SIO4_ASYNC_USC_TX_CLK_SRC_DISABLE	Disable the transmitter.	SIO4_ASYNC_USC_TX_CLK_SRC_DPLL	Select the DPLL.	SIO4_ASYNC_USC_TX_CLK_SRC_RXC_PIN	Select the Rx Clock pin. *	SIO4_ASYNC_USC_TX_CLK_SRC_TXC_PIN	Select the Tx Clock pin.										
Value	Description																												
SIO4_ASYNC_USC_TX_CLK_SRC_BRG0	Select Baud Rate Generator 0.																												
SIO4_ASYNC_USC_TX_CLK_SRC_BRG1	Select Baud Rate Generator 1.																												
SIO4_ASYNC_USC_TX_CLK_SRC_CTR0	Select Counter 0.																												
SIO4_ASYNC_USC_TX_CLK_SRC_CTR1	Select Counter 1.																												
SIO4_ASYNC_USC_TX_CLK_SRC_DISABLE	Disable the transmitter.																												
SIO4_ASYNC_USC_TX_CLK_SRC_DPLL	Select the DPLL.																												
SIO4_ASYNC_USC_TX_CLK_SRC_RXC_PIN	Select the Rx Clock pin. *																												
SIO4_ASYNC_USC_TX_CLK_SRC_TXC_PIN	Select the Tx Clock pin.																												
usc. tx. txc	<p>This field configures the operation of the USC's Tx Clock pin. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_tx_txc()</code>.</p> <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>SIO4_ASYNC_USC_TX_TXC_IN_0</td><td>The pin is an input driven low.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_TXC_IN_1</td><td>The pin is an input driven high.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_TXC_IN_CBL_RXAUX</td><td>The pin is an input driven from the cable's Rx Aux signal.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_TXC_IN_CBL_RXC</td><td>The pin is an input driven from the cable's Rx Clock signal.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_TXC_IN_OSC</td><td>The pin is an input driven from the onboard oscillator.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_TXC_IN_OSC_INV</td><td>The pin is an input driven from the inverted onboard oscillator.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_TXC_OUT_BRG0</td><td>The pin is an output driven from Baud Rate Generator 0.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_TXC_OUT_BRG1</td><td>The pin is an output driven from Baud Rate Generator 1.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_TXC_OUT_BYTE_CLK</td><td>The pin is an output driven from the transmitter's Byte Clock.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_TXC_OUT_CLK</td><td>The pin is an output driven from the transmit clock. This is the default.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_TXC_OUT_COMP</td><td>The pin is an output driven from the transmit complete signal.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_TXC_OUT_CTR1</td><td>The pin is an output driven from Counter 1.</td></tr> <tr> <td>SIO4_ASYNC_USC_TX_TXC_OUT_DPLL_TX</td><td>The pin is an output driven from the transmit clock from the DPLL.</td></tr> </tbody> </table>	Value	Description	SIO4_ASYNC_USC_TX_TXC_IN_0	The pin is an input driven low.	SIO4_ASYNC_USC_TX_TXC_IN_1	The pin is an input driven high.	SIO4_ASYNC_USC_TX_TXC_IN_CBL_RXAUX	The pin is an input driven from the cable's Rx Aux signal.	SIO4_ASYNC_USC_TX_TXC_IN_CBL_RXC	The pin is an input driven from the cable's Rx Clock signal.	SIO4_ASYNC_USC_TX_TXC_IN_OSC	The pin is an input driven from the onboard oscillator.	SIO4_ASYNC_USC_TX_TXC_IN_OSC_INV	The pin is an input driven from the inverted onboard oscillator.	SIO4_ASYNC_USC_TX_TXC_OUT_BRG0	The pin is an output driven from Baud Rate Generator 0.	SIO4_ASYNC_USC_TX_TXC_OUT_BRG1	The pin is an output driven from Baud Rate Generator 1.	SIO4_ASYNC_USC_TX_TXC_OUT_BYTE_CLK	The pin is an output driven from the transmitter's Byte Clock.	SIO4_ASYNC_USC_TX_TXC_OUT_CLK	The pin is an output driven from the transmit clock. This is the default.	SIO4_ASYNC_USC_TX_TXC_OUT_COMP	The pin is an output driven from the transmit complete signal.	SIO4_ASYNC_USC_TX_TXC_OUT_CTR1	The pin is an output driven from Counter 1.	SIO4_ASYNC_USC_TX_TXC_OUT_DPLL_TX	The pin is an output driven from the transmit clock from the DPLL.
Value	Description																												
SIO4_ASYNC_USC_TX_TXC_IN_0	The pin is an input driven low.																												
SIO4_ASYNC_USC_TX_TXC_IN_1	The pin is an input driven high.																												
SIO4_ASYNC_USC_TX_TXC_IN_CBL_RXAUX	The pin is an input driven from the cable's Rx Aux signal.																												
SIO4_ASYNC_USC_TX_TXC_IN_CBL_RXC	The pin is an input driven from the cable's Rx Clock signal.																												
SIO4_ASYNC_USC_TX_TXC_IN_OSC	The pin is an input driven from the onboard oscillator.																												
SIO4_ASYNC_USC_TX_TXC_IN_OSC_INV	The pin is an input driven from the inverted onboard oscillator.																												
SIO4_ASYNC_USC_TX_TXC_OUT_BRG0	The pin is an output driven from Baud Rate Generator 0.																												
SIO4_ASYNC_USC_TX_TXC_OUT_BRG1	The pin is an output driven from Baud Rate Generator 1.																												
SIO4_ASYNC_USC_TX_TXC_OUT_BYTE_CLK	The pin is an output driven from the transmitter's Byte Clock.																												
SIO4_ASYNC_USC_TX_TXC_OUT_CLK	The pin is an output driven from the transmit clock. This is the default.																												
SIO4_ASYNC_USC_TX_TXC_OUT_COMP	The pin is an output driven from the transmit complete signal.																												
SIO4_ASYNC_USC_TX_TXC_OUT_CTR1	The pin is an output driven from Counter 1.																												
SIO4_ASYNC_USC_TX_TXC_OUT_DPLL_TX	The pin is an output driven from the transmit clock from the DPLL.																												
usc. tx. txc_legacy	<p>This field configures the operation of the USC's Tx Clock pin for legacy mode cable interface configurations. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_tx_txc_legacy()</code>.</p> <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> </tbody> </table>	Value	Description																										
Value	Description																												

	SIO4_ASYNC_USC_TX_TXC_LEG_IN	The pin operates as an input.																																																		
	SIO4_ASYNC_USC_TX_TXC_LEG_OUT_BRG0	The pin is an output driven from Baud Rate Generator 0.																																																		
	SIO4_ASYNC_USC_TX_TXC_LEG_OUT_BRG1	The pin is an output driven from Baud Rate Generator 1.																																																		
	SIO4_ASYNC_USC_TX_TXC_LEG_OUT_BYTE_CLK	The pin is an output driven from the transmitter's Byte Clock.																																																		
	SIO4_ASYNC_USC_TX_TXC_LEG_OUT_CLK	The pin is an output driven from the transmit clock. This is the default.																																																		
	SIO4_ASYNC_USC_TX_TXC_LEG_OUT_COMP	The pin is an output driven from the transmit complete signal.																																																		
	SIO4_ASYNC_USC_TX_TXC_LEG_OUT_CTR1	The pin is an output driven from Counter 1.																																																		
	SIO4_ASYNC_USC_TX_TXC_LEG_OUT_DPLL_TX	The pin is an output driven from the transmit clock from the DPLL.																																																		
usc. tx. stop_bits	This field configures the number of Stop Bits injected by the transmitter after each data byte. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_tx_stop_bits()</code> .																																																			
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_1</td><td>This refers to a single stop bit. This is the default.</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_2</td><td>This refers to two stop bits.</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_0_9_16</td><td>This refers to 9/16th of a stop bit.</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_0_10_16</td><td>This refers to 10/16th of a stop bit (5/8th).</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_0_11_16</td><td>This refers to 11/16th of a stop bit.</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_0_12_16</td><td>This refers to 12/16th of a stop bit (3/4th).</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_0_13_16</td><td>This refers to 13/16th of a stop bit.</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_0_14_16</td><td>This refers to 14/16th of a stop bit (7/8th).</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_0_15_16</td><td>This refers to 15/16th of a stop bit.</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_1_1_16</td><td>This refers to 1 and 1/16th stop bits.</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_1_2_16</td><td>This refers to 1 and 2/16th stop bits (1+1/8th).</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_1_3_16</td><td>This refers to 1 and 3/16th stop bits.</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_1_4_16</td><td>This refers to 1 and 4/16th stop bits (1+1/4th).</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_1_5_16</td><td>This refers to 1 and 5/16th stop bits.</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_1_6_16</td><td>This refers to 1 and 6/16th stop bits (1+3/8th).</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_1_7_16</td><td>This refers to 1 and 7/16th stop bits.</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_1_8_16</td><td>This refers to 1 and 8/16th stop bits (1+1/2).</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_1_9_16</td><td>This refers to 1 and 9/16th stop bits.</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_1_10_16</td><td>This refers to 1 and 10/16th stop bits (1+5/8th).</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_1_11_16</td><td>This refers to 1 and 11/16th stop bits.</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_1_12_16</td><td>This refers to 1 and 12/16th stop bits (1+3/4th).</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_1_13_16</td><td>This refers to 1 and 13/16th stop bits.</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_1_14_16</td><td>This refers to 1 and 14/16th stop bits (1+7/8th).</td></tr><tr><td>SIO4_ASYNC_USC_TX_STOP_BIT_1_15_16</td><td>This refers to 1 and 15/16th stop bits.</td></tr></table>	Value	Description	SIO4_ASYNC_USC_TX_STOP_BIT_1	This refers to a single stop bit. This is the default.	SIO4_ASYNC_USC_TX_STOP_BIT_2	This refers to two stop bits.	SIO4_ASYNC_USC_TX_STOP_BIT_0_9_16	This refers to 9/16 th of a stop bit.	SIO4_ASYNC_USC_TX_STOP_BIT_0_10_16	This refers to 10/16 th of a stop bit (5/8 th).	SIO4_ASYNC_USC_TX_STOP_BIT_0_11_16	This refers to 11/16 th of a stop bit.	SIO4_ASYNC_USC_TX_STOP_BIT_0_12_16	This refers to 12/16 th of a stop bit (3/4 th).	SIO4_ASYNC_USC_TX_STOP_BIT_0_13_16	This refers to 13/16 th of a stop bit.	SIO4_ASYNC_USC_TX_STOP_BIT_0_14_16	This refers to 14/16 th of a stop bit (7/8 th).	SIO4_ASYNC_USC_TX_STOP_BIT_0_15_16	This refers to 15/16 th of a stop bit.	SIO4_ASYNC_USC_TX_STOP_BIT_1_1_16	This refers to 1 and 1/16 th stop bits.	SIO4_ASYNC_USC_TX_STOP_BIT_1_2_16	This refers to 1 and 2/16 th stop bits (1+1/8 th).	SIO4_ASYNC_USC_TX_STOP_BIT_1_3_16	This refers to 1 and 3/16 th stop bits.	SIO4_ASYNC_USC_TX_STOP_BIT_1_4_16	This refers to 1 and 4/16 th stop bits (1+1/4 th).	SIO4_ASYNC_USC_TX_STOP_BIT_1_5_16	This refers to 1 and 5/16 th stop bits.	SIO4_ASYNC_USC_TX_STOP_BIT_1_6_16	This refers to 1 and 6/16 th stop bits (1+3/8 th).	SIO4_ASYNC_USC_TX_STOP_BIT_1_7_16	This refers to 1 and 7/16 th stop bits.	SIO4_ASYNC_USC_TX_STOP_BIT_1_8_16	This refers to 1 and 8/16 th stop bits (1+1/2).	SIO4_ASYNC_USC_TX_STOP_BIT_1_9_16	This refers to 1 and 9/16 th stop bits.	SIO4_ASYNC_USC_TX_STOP_BIT_1_10_16	This refers to 1 and 10/16 th stop bits (1+5/8 th).	SIO4_ASYNC_USC_TX_STOP_BIT_1_11_16	This refers to 1 and 11/16 th stop bits.	SIO4_ASYNC_USC_TX_STOP_BIT_1_12_16	This refers to 1 and 12/16 th stop bits (1+3/4 th).	SIO4_ASYNC_USC_TX_STOP_BIT_1_13_16	This refers to 1 and 13/16 th stop bits.	SIO4_ASYNC_USC_TX_STOP_BIT_1_14_16	This refers to 1 and 14/16 th stop bits (1+7/8 th).	SIO4_ASYNC_USC_TX_STOP_BIT_1_15_16	This refers to 1 and 15/16 th stop bits.	
Value	Description																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_1	This refers to a single stop bit. This is the default.																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_2	This refers to two stop bits.																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_0_9_16	This refers to 9/16 th of a stop bit.																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_0_10_16	This refers to 10/16 th of a stop bit (5/8 th).																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_0_11_16	This refers to 11/16 th of a stop bit.																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_0_12_16	This refers to 12/16 th of a stop bit (3/4 th).																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_0_13_16	This refers to 13/16 th of a stop bit.																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_0_14_16	This refers to 14/16 th of a stop bit (7/8 th).																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_0_15_16	This refers to 15/16 th of a stop bit.																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_1_1_16	This refers to 1 and 1/16 th stop bits.																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_1_2_16	This refers to 1 and 2/16 th stop bits (1+1/8 th).																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_1_3_16	This refers to 1 and 3/16 th stop bits.																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_1_4_16	This refers to 1 and 4/16 th stop bits (1+1/4 th).																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_1_5_16	This refers to 1 and 5/16 th stop bits.																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_1_6_16	This refers to 1 and 6/16 th stop bits (1+3/8 th).																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_1_7_16	This refers to 1 and 7/16 th stop bits.																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_1_8_16	This refers to 1 and 8/16 th stop bits (1+1/2).																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_1_9_16	This refers to 1 and 9/16 th stop bits.																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_1_10_16	This refers to 1 and 10/16 th stop bits (1+5/8 th).																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_1_11_16	This refers to 1 and 11/16 th stop bits.																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_1_12_16	This refers to 1 and 12/16 th stop bits (1+3/4 th).																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_1_13_16	This refers to 1 and 13/16 th stop bits.																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_1_14_16	This refers to 1 and 14/16 th stop bits (1+7/8 th).																																																			
SIO4_ASYNC_USC_TX_STOP_BIT_1_15_16	This refers to 1 and 15/16 th stop bits.																																																			
usc.	This structure configures the remaining USC receiver settings.																																																			

rx																													
usc. rx. clk_rate	<p>This field configures the clock divider rate for the USC receiver. This is the receiver oversampling rate. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_rx_clk_rate()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_USC_RX_CLK_RATE_16X</td><td>Divide the source by 16.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_CLK_RATE_32X</td><td>Divide the source by 32.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_CLK_RATE_64X</td><td>Divide the source by 64. *</td></tr> </table> <p>* This is the hard coded initialization default, which may subsequently be modified as needed to produce the user specified Rx bit rate.</p>	Value	Description	SIO4_ASYNC_USC_RX_CLK_RATE_16X	Divide the source by 16.	SIO4_ASYNC_USC_RX_CLK_RATE_32X	Divide the source by 32.	SIO4_ASYNC_USC_RX_CLK_RATE_64X	Divide the source by 64. *																				
Value	Description																												
SIO4_ASYNC_USC_RX_CLK_RATE_16X	Divide the source by 16.																												
SIO4_ASYNC_USC_RX_CLK_RATE_32X	Divide the source by 32.																												
SIO4_ASYNC_USC_RX_CLK_RATE_64X	Divide the source by 64. *																												
usc. rx. clk_src	<p>This field configures the source for the USC receiver clock. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_rx_clk_src()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_USC_RX_CLK_SRC_BRG0</td><td>Select Baud Rate Generator 0.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_CLK_SRC_BRG1</td><td>Select Baud Rate Generator 1.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_CLK_SRC_CTR0</td><td>Select Counter 0.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_CLK_SRC_CTR1</td><td>Select Counter 1.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_CLK_SRC_DISABLE</td><td>Disable the receiver.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_CLK_SRC_DPLL</td><td>Select the DPLL.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_CLK_SRC_RXC_PIN</td><td>Select the Rx Clock pin. *</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_CLK_SRC_TXC_PIN</td><td>Select the Tx Clock pin.</td></tr> </table> <p>* This is the hard coded initialization default, which may subsequently be modified as needed to produce the user specified Rx bit rate.</p>	Value	Description	SIO4_ASYNC_USC_RX_CLK_SRC_BRG0	Select Baud Rate Generator 0.	SIO4_ASYNC_USC_RX_CLK_SRC_BRG1	Select Baud Rate Generator 1.	SIO4_ASYNC_USC_RX_CLK_SRC_CTR0	Select Counter 0.	SIO4_ASYNC_USC_RX_CLK_SRC_CTR1	Select Counter 1.	SIO4_ASYNC_USC_RX_CLK_SRC_DISABLE	Disable the receiver.	SIO4_ASYNC_USC_RX_CLK_SRC_DPLL	Select the DPLL.	SIO4_ASYNC_USC_RX_CLK_SRC_RXC_PIN	Select the Rx Clock pin. *	SIO4_ASYNC_USC_RX_CLK_SRC_TXC_PIN	Select the Tx Clock pin.										
Value	Description																												
SIO4_ASYNC_USC_RX_CLK_SRC_BRG0	Select Baud Rate Generator 0.																												
SIO4_ASYNC_USC_RX_CLK_SRC_BRG1	Select Baud Rate Generator 1.																												
SIO4_ASYNC_USC_RX_CLK_SRC_CTR0	Select Counter 0.																												
SIO4_ASYNC_USC_RX_CLK_SRC_CTR1	Select Counter 1.																												
SIO4_ASYNC_USC_RX_CLK_SRC_DISABLE	Disable the receiver.																												
SIO4_ASYNC_USC_RX_CLK_SRC_DPLL	Select the DPLL.																												
SIO4_ASYNC_USC_RX_CLK_SRC_RXC_PIN	Select the Rx Clock pin. *																												
SIO4_ASYNC_USC_RX_CLK_SRC_TXC_PIN	Select the Tx Clock pin.																												
usc. rx. rxc	<p>This field configures the operation of the USC's Rx Clock pin. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_rx_rxc()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_IN_0</td><td>The pin is an input driven low.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_IN_1</td><td>The pin is an input driven high.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_IN_CBL_RXAUX</td><td>The pin is an input driven from the cable's Rx Aux signal.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_IN_CBL_RXC</td><td>The pin is an input driven from the cable's Rx Clock signal. This is the default.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_IN_OSC</td><td>The pin is an input driven from the onboard oscillator.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_IN_OSC_INV</td><td>The pin is an input driven from the inverted onboard oscillator.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_OUT_BRG0</td><td>The pin is an output driven from Baud Rate Generator 0.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_OUT_BRG1</td><td>The pin is an output driven from Baud Rate Generator 1.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_OUT_BYTE_CLK</td><td>The pin is an output driven from the receiver's Byte Clock.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_OUT_CLK</td><td>The pin is an output driven from the receiver clock.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_OUT_CTR0</td><td>The pin is an output driven from Counter 0.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_OUT_DPLL_RX</td><td>The pin is an output driven from the DPLL receiver clock.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_OUT_SYNC</td><td>The pin is an output driven from the receiver's SYNC signal.</td></tr> </table>	Value	Description	SIO4_ASYNC_USC_RX_RXC_IN_0	The pin is an input driven low.	SIO4_ASYNC_USC_RX_RXC_IN_1	The pin is an input driven high.	SIO4_ASYNC_USC_RX_RXC_IN_CBL_RXAUX	The pin is an input driven from the cable's Rx Aux signal.	SIO4_ASYNC_USC_RX_RXC_IN_CBL_RXC	The pin is an input driven from the cable's Rx Clock signal. This is the default.	SIO4_ASYNC_USC_RX_RXC_IN_OSC	The pin is an input driven from the onboard oscillator.	SIO4_ASYNC_USC_RX_RXC_IN_OSC_INV	The pin is an input driven from the inverted onboard oscillator.	SIO4_ASYNC_USC_RX_RXC_OUT_BRG0	The pin is an output driven from Baud Rate Generator 0.	SIO4_ASYNC_USC_RX_RXC_OUT_BRG1	The pin is an output driven from Baud Rate Generator 1.	SIO4_ASYNC_USC_RX_RXC_OUT_BYTE_CLK	The pin is an output driven from the receiver's Byte Clock.	SIO4_ASYNC_USC_RX_RXC_OUT_CLK	The pin is an output driven from the receiver clock.	SIO4_ASYNC_USC_RX_RXC_OUT_CTR0	The pin is an output driven from Counter 0.	SIO4_ASYNC_USC_RX_RXC_OUT_DPLL_RX	The pin is an output driven from the DPLL receiver clock.	SIO4_ASYNC_USC_RX_RXC_OUT_SYNC	The pin is an output driven from the receiver's SYNC signal.
Value	Description																												
SIO4_ASYNC_USC_RX_RXC_IN_0	The pin is an input driven low.																												
SIO4_ASYNC_USC_RX_RXC_IN_1	The pin is an input driven high.																												
SIO4_ASYNC_USC_RX_RXC_IN_CBL_RXAUX	The pin is an input driven from the cable's Rx Aux signal.																												
SIO4_ASYNC_USC_RX_RXC_IN_CBL_RXC	The pin is an input driven from the cable's Rx Clock signal. This is the default.																												
SIO4_ASYNC_USC_RX_RXC_IN_OSC	The pin is an input driven from the onboard oscillator.																												
SIO4_ASYNC_USC_RX_RXC_IN_OSC_INV	The pin is an input driven from the inverted onboard oscillator.																												
SIO4_ASYNC_USC_RX_RXC_OUT_BRG0	The pin is an output driven from Baud Rate Generator 0.																												
SIO4_ASYNC_USC_RX_RXC_OUT_BRG1	The pin is an output driven from Baud Rate Generator 1.																												
SIO4_ASYNC_USC_RX_RXC_OUT_BYTE_CLK	The pin is an output driven from the receiver's Byte Clock.																												
SIO4_ASYNC_USC_RX_RXC_OUT_CLK	The pin is an output driven from the receiver clock.																												
SIO4_ASYNC_USC_RX_RXC_OUT_CTR0	The pin is an output driven from Counter 0.																												
SIO4_ASYNC_USC_RX_RXC_OUT_DPLL_RX	The pin is an output driven from the DPLL receiver clock.																												
SIO4_ASYNC_USC_RX_RXC_OUT_SYNC	The pin is an output driven from the receiver's SYNC signal.																												
usc.	This field configures the operation of the USC's Rx Clock pin for legacy mode cable interface																												

rx. rx_legacy	<p>configurations. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_rx_rxc_legacy()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_LEG_IN</td><td>The pin is an input. This is the default.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_LEG_OUT_BRG0</td><td>The pin is an output driven from Baud Rate Generator 0.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_LEG_OUT_BRG1</td><td>The pin is an output driven from Baud Rate Generator 1.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_LEG_OUT_BYTE_CLK</td><td>The pin is an output driven from the receiver's Byte Clock.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_LEG_OUT_CLK</td><td>The pin is an output driven from the receiver clock.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_LEG_OUT_CTR0</td><td>The pin is an output driven from Counter 0.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_LEG_OUT_DPLL_RX</td><td>The pin is an output driven from the DPLL receiver clock.</td></tr> <tr> <td>SIO4_ASYNC_USC_RX_RXC_LEG_OUT_SYNC</td><td>The pin is an output driven from the receiver's SYNC signal.</td></tr> </table>	Value	Description	SIO4_ASYNC_USC_RX_RXC_LEG_IN	The pin is an input. This is the default.	SIO4_ASYNC_USC_RX_RXC_LEG_OUT_BRG0	The pin is an output driven from Baud Rate Generator 0.	SIO4_ASYNC_USC_RX_RXC_LEG_OUT_BRG1	The pin is an output driven from Baud Rate Generator 1.	SIO4_ASYNC_USC_RX_RXC_LEG_OUT_BYTE_CLK	The pin is an output driven from the receiver's Byte Clock.	SIO4_ASYNC_USC_RX_RXC_LEG_OUT_CLK	The pin is an output driven from the receiver clock.	SIO4_ASYNC_USC_RX_RXC_LEG_OUT_CTR0	The pin is an output driven from Counter 0.	SIO4_ASYNC_USC_RX_RXC_LEG_OUT_DPLL_RX	The pin is an output driven from the DPLL receiver clock.	SIO4_ASYNC_USC_RX_RXC_LEG_OUT_SYNC	The pin is an output driven from the receiver's SYNC signal.
Value	Description																		
SIO4_ASYNC_USC_RX_RXC_LEG_IN	The pin is an input. This is the default.																		
SIO4_ASYNC_USC_RX_RXC_LEG_OUT_BRG0	The pin is an output driven from Baud Rate Generator 0.																		
SIO4_ASYNC_USC_RX_RXC_LEG_OUT_BRG1	The pin is an output driven from Baud Rate Generator 1.																		
SIO4_ASYNC_USC_RX_RXC_LEG_OUT_BYTE_CLK	The pin is an output driven from the receiver's Byte Clock.																		
SIO4_ASYNC_USC_RX_RXC_LEG_OUT_CLK	The pin is an output driven from the receiver clock.																		
SIO4_ASYNC_USC_RX_RXC_LEG_OUT_CTR0	The pin is an output driven from Counter 0.																		
SIO4_ASYNC_USC_RX_RXC_LEG_OUT_DPLL_RX	The pin is an output driven from the DPLL receiver clock.																		
SIO4_ASYNC_USC_RX_RXC_LEG_OUT_SYNC	The pin is an output driven from the receiver's SYNC signal.																		
usc. brg0	This structure configures settings for Baud Rate Generator 0 (BRG0).																		
usc. brg0. enable	<p>This field enables or disabled Baud Rate Generator 0. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_brg0_enable()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_USC_BRG0_ENABLE_NO</td><td>This disables BRG0.*</td></tr> <tr> <td>SIO4_ASYNC_USC_BRG0_ENABLE_YES</td><td>This enables BRG0.</td></tr> </table> <p>* This is the hard coded initialization default, which may subsequently be modified as needed to produce the user specified Tx bit rate.</p>	Value	Description	SIO4_ASYNC_USC_BRG0_ENABLE_NO	This disables BRG0.*	SIO4_ASYNC_USC_BRG0_ENABLE_YES	This enables BRG0.												
Value	Description																		
SIO4_ASYNC_USC_BRG0_ENABLE_NO	This disables BRG0.*																		
SIO4_ASYNC_USC_BRG0_ENABLE_YES	This enables BRG0.																		
usc. brg0. clk_src	<p>This field selects the clock source for Baud Rate Generator 0. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_brg0_clk_src()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_USC_BRG0_CLK_SRC_CTR0</td><td>This selects the Counter 0 output. *</td></tr> <tr> <td>SIO4_ASYNC_USC_BRG0_CLK_SRC_CTR1</td><td>This selects the Counter 1 output.</td></tr> <tr> <td>SIO4_ASYNC_USC_BRG0_CLK_SRC_RXC_PIN</td><td>This selects the signal present at the USC's Rx Clock pin.</td></tr> <tr> <td>SIO4_ASYNC_USC_BRG0_CLK_SRC_TXC_PIN</td><td>This selects the signal present at the USC's Tx Clock pin.</td></tr> </table> <p>* This is the hard coded initialization default, which may subsequently be modified as needed to produce the user specified Tx bit rate.</p>	Value	Description	SIO4_ASYNC_USC_BRG0_CLK_SRC_CTR0	This selects the Counter 0 output. *	SIO4_ASYNC_USC_BRG0_CLK_SRC_CTR1	This selects the Counter 1 output.	SIO4_ASYNC_USC_BRG0_CLK_SRC_RXC_PIN	This selects the signal present at the USC's Rx Clock pin.	SIO4_ASYNC_USC_BRG0_CLK_SRC_TXC_PIN	This selects the signal present at the USC's Tx Clock pin.								
Value	Description																		
SIO4_ASYNC_USC_BRG0_CLK_SRC_CTR0	This selects the Counter 0 output. *																		
SIO4_ASYNC_USC_BRG0_CLK_SRC_CTR1	This selects the Counter 1 output.																		
SIO4_ASYNC_USC_BRG0_CLK_SRC_RXC_PIN	This selects the signal present at the USC's Rx Clock pin.																		
SIO4_ASYNC_USC_BRG0_CLK_SRC_TXC_PIN	This selects the signal present at the USC's Tx Clock pin.																		
usc. brg0. divider	<p>This field specifies the clock divider value for Baud Rate Generator 0. The valid value range is from zero to 0xFFFF. The hard coded initialization default is zero, which may subsequently be modified as needed to produce the user specified Tx bit rate. The feature's low level function is <code>sio4_async_t_usc_brg0_divider()</code>.</p>																		
usc. brg0. mode	<p>This field specifies the Baud Rate Generator 0 operating mode. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_brg0_mode()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_USC_BRG0_MODE_CONT</td><td>This selects continuous operation. This is the default.</td></tr> <tr> <td>SIO4_ASYNC_USC_BRG0_MODE_SINGLE</td><td>This selects single shot mode, in which clocking stops when the counter value reaches zero.</td></tr> </table>	Value	Description	SIO4_ASYNC_USC_BRG0_MODE_CONT	This selects continuous operation. This is the default.	SIO4_ASYNC_USC_BRG0_MODE_SINGLE	This selects single shot mode, in which clocking stops when the counter value reaches zero.												
Value	Description																		
SIO4_ASYNC_USC_BRG0_MODE_CONT	This selects continuous operation. This is the default.																		
SIO4_ASYNC_USC_BRG0_MODE_SINGLE	This selects single shot mode, in which clocking stops when the counter value reaches zero.																		

usc. brg1	This structure configures settings for Baud Rate Generator 1 (BRG1).										
usc. brg1. enable	<p>This field enables or disabled Baud Rate Generator 1. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_brg1_enable()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_USC_BRG1_ENABLE_NO</td><td>This disables BRG1. *</td></tr> <tr> <td>SIO4_ASYNC_USC_BRG1_ENABLE_YES</td><td>This enables BRG1.</td></tr> </table> <p>* This is the hard coded initialization default, which may subsequently be modified as needed to produce the user specified Rx bit rate.</p>	Value	Description	SIO4_ASYNC_USC_BRG1_ENABLE_NO	This disables BRG1. *	SIO4_ASYNC_USC_BRG1_ENABLE_YES	This enables BRG1.				
Value	Description										
SIO4_ASYNC_USC_BRG1_ENABLE_NO	This disables BRG1. *										
SIO4_ASYNC_USC_BRG1_ENABLE_YES	This enables BRG1.										
usc. brg1. clk_src	<p>This field selects the clock source for Baud Rate Generator 1. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_brg1_clk_src()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_USC_BRG1_CLK_SRC_CTR0</td><td>This selects the output from Counter 0.</td></tr> <tr> <td>SIO4_ASYNC_USC_BRG1_CLK_SRC_CTR1</td><td>This selects the output from Counter 1. This is the default.</td></tr> <tr> <td>SIO4_ASYNC_USC_BRG1_CLK_SRC_RXC_PIN</td><td>This selects the signal present at the USC's Rx Clock pin.</td></tr> <tr> <td>SIO4_ASYNC_USC_BRG1_CLK_SRC_TXC_PIN</td><td>This selects the signal present at the USC's Tx Clock pin.</td></tr> </table>	Value	Description	SIO4_ASYNC_USC_BRG1_CLK_SRC_CTR0	This selects the output from Counter 0.	SIO4_ASYNC_USC_BRG1_CLK_SRC_CTR1	This selects the output from Counter 1. This is the default.	SIO4_ASYNC_USC_BRG1_CLK_SRC_RXC_PIN	This selects the signal present at the USC's Rx Clock pin.	SIO4_ASYNC_USC_BRG1_CLK_SRC_TXC_PIN	This selects the signal present at the USC's Tx Clock pin.
Value	Description										
SIO4_ASYNC_USC_BRG1_CLK_SRC_CTR0	This selects the output from Counter 0.										
SIO4_ASYNC_USC_BRG1_CLK_SRC_CTR1	This selects the output from Counter 1. This is the default.										
SIO4_ASYNC_USC_BRG1_CLK_SRC_RXC_PIN	This selects the signal present at the USC's Rx Clock pin.										
SIO4_ASYNC_USC_BRG1_CLK_SRC_TXC_PIN	This selects the signal present at the USC's Tx Clock pin.										
usc. brg1. divider	This field specifies the clock divider value for Baud Rate Generator 1. The valid value range is from zero to 0xFFFF. The hard coded initialization default is zero, which may subsequently be modified as needed to produce the user specified Rx bit rate. The feature's low level function is <code>sio4_async_t_usc_brg1_divider()</code> .										
usc. brg1. mode	<p>This field specifies the Baud Rate Generator 1 operating mode. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_brg1_mode()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_USC_BRG1_MODE_CONT</td><td>This selects continuous operation. This is the default.</td></tr> <tr> <td>SIO4_ASYNC_USC_BRG1_MODE_SINGLE</td><td>This selects single shot mode, in which clocking stops when the counter value reaches zero.</td></tr> </table>	Value	Description	SIO4_ASYNC_USC_BRG1_MODE_CONT	This selects continuous operation. This is the default.	SIO4_ASYNC_USC_BRG1_MODE_SINGLE	This selects single shot mode, in which clocking stops when the counter value reaches zero.				
Value	Description										
SIO4_ASYNC_USC_BRG1_MODE_CONT	This selects continuous operation. This is the default.										
SIO4_ASYNC_USC_BRG1_MODE_SINGLE	This selects single shot mode, in which clocking stops when the counter value reaches zero.										
usc. ctr0	This structure configures settings for Counter 0 (CTR0).										
usc. ctr0. clk_src	<p>This field selects the clock source for Counter 0. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_ctr0_clk_src()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_USC_CTR0_CLK_SRC_DISABLE</td><td>This disables Counter 0. *</td></tr> <tr> <td>SIO4_ASYNC_USC_CTR0_CLK_SRC_RXC_PIN</td><td>This selects the signal present at the USC's Rx Clock pin.</td></tr> <tr> <td>SIO4_ASYNC_USC_CTR0_CLK_SRC_TXC_PIN</td><td>This selects the signal present at the USC's Tx Clock pin.</td></tr> </table> <p>* This is the hard coded initialization default, which may subsequently be modified as needed to produce the user specified Tx bit rate.</p>	Value	Description	SIO4_ASYNC_USC_CTR0_CLK_SRC_DISABLE	This disables Counter 0. *	SIO4_ASYNC_USC_CTR0_CLK_SRC_RXC_PIN	This selects the signal present at the USC's Rx Clock pin.	SIO4_ASYNC_USC_CTR0_CLK_SRC_TXC_PIN	This selects the signal present at the USC's Tx Clock pin.		
Value	Description										
SIO4_ASYNC_USC_CTR0_CLK_SRC_DISABLE	This disables Counter 0. *										
SIO4_ASYNC_USC_CTR0_CLK_SRC_RXC_PIN	This selects the signal present at the USC's Rx Clock pin.										
SIO4_ASYNC_USC_CTR0_CLK_SRC_TXC_PIN	This selects the signal present at the USC's Tx Clock pin.										
usc. ctr0. rate	<p>This field selects the divider rate for Counter 0. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_ctr0_rate()</code>.</p> <table> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>SIO4_ASYNC_USC_CTR0_RATE_4X</td><td>This sets the output as the input divided by four.</td></tr> <tr> <td>SIO4_ASYNC_USC_CTR0_RATE_8X</td><td>This sets the output as the input divided by eight.</td></tr> <tr> <td>SIO4_ASYNC_USC_CTR0_RATE_16X</td><td>This sets the output as the input divided by 16.</td></tr> <tr> <td>SIO4_ASYNC_USC_CTR0_RATE_32X</td><td>This sets the output as the input divided by 32. *</td></tr> </table>	Value	Description	SIO4_ASYNC_USC_CTR0_RATE_4X	This sets the output as the input divided by four.	SIO4_ASYNC_USC_CTR0_RATE_8X	This sets the output as the input divided by eight.	SIO4_ASYNC_USC_CTR0_RATE_16X	This sets the output as the input divided by 16.	SIO4_ASYNC_USC_CTR0_RATE_32X	This sets the output as the input divided by 32. *
Value	Description										
SIO4_ASYNC_USC_CTR0_RATE_4X	This sets the output as the input divided by four.										
SIO4_ASYNC_USC_CTR0_RATE_8X	This sets the output as the input divided by eight.										
SIO4_ASYNC_USC_CTR0_RATE_16X	This sets the output as the input divided by 16.										
SIO4_ASYNC_USC_CTR0_RATE_32X	This sets the output as the input divided by 32. *										

	* This is the hard coded initialization default, which may subsequently be modified as needed to produce the user specified Tx bit rate.										
usc. ctrl	This structure configures settings for Counter 1 (CTR1).										
usc. ctrl. clk_src	<p>This field selects the clock source for Counter 1. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_ctrl_clk_src()</code>.</p> <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>SIO4_ASYNC_USC_CTRL_CLK_SRC_DISABLE</td><td>This disables Counter 1. *</td></tr> <tr> <td>SIO4_ASYNC_USC_CTRL_CLK_SRC_RXC_PIN</td><td>This selects the signal present at the USC's Rx Clock pin.</td></tr> <tr> <td>SIO4_ASYNC_USC_CTRL_CLK_SRC_TXC_PIN</td><td>This selects the signal present at the USC's Tx Clock pin.</td></tr> </tbody> </table> <p>* This is the hard coded initialization default, which may subsequently be modified as needed to produce the user specified Rx bit rate.</p>	Value	Description	SIO4_ASYNC_USC_CTRL_CLK_SRC_DISABLE	This disables Counter 1. *	SIO4_ASYNC_USC_CTRL_CLK_SRC_RXC_PIN	This selects the signal present at the USC's Rx Clock pin.	SIO4_ASYNC_USC_CTRL_CLK_SRC_TXC_PIN	This selects the signal present at the USC's Tx Clock pin.		
Value	Description										
SIO4_ASYNC_USC_CTRL_CLK_SRC_DISABLE	This disables Counter 1. *										
SIO4_ASYNC_USC_CTRL_CLK_SRC_RXC_PIN	This selects the signal present at the USC's Rx Clock pin.										
SIO4_ASYNC_USC_CTRL_CLK_SRC_TXC_PIN	This selects the signal present at the USC's Tx Clock pin.										
usc. ctrl. rate_src	<p>This field selects the source for the rate divider used by Counter 1. Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_ctrl_rate_src()</code>.</p> <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>SIO4_ASYNC_USC_CTRL_RATE_SRC_CTR0</td><td>This selects the rate divider used by CTR0.</td></tr> <tr> <td>SIO4_ASYNC_USC_CTRL_RATE_SRC_DPLL</td><td>This selects the rate divider used by the DPLL. This is the default.</td></tr> </tbody> </table>	Value	Description	SIO4_ASYNC_USC_CTRL_RATE_SRC_CTR0	This selects the rate divider used by CTR0.	SIO4_ASYNC_USC_CTRL_RATE_SRC_DPLL	This selects the rate divider used by the DPLL. This is the default.				
Value	Description										
SIO4_ASYNC_USC_CTRL_RATE_SRC_CTR0	This selects the rate divider used by CTR0.										
SIO4_ASYNC_USC_CTRL_RATE_SRC_DPLL	This selects the rate divider used by the DPLL. This is the default.										
usc. dpll	This structure configures settings for DPLL. The DPLL is unused by the Asynchronous Protocol Library except for the one field described below.										
usc. dpll. rate	<p>This field selects the divider rate for the DPLL, which is used as the divider rate for Counter 1 (CTR1). Valid values are given in the table below. The feature's low level function is <code>sio4_async_t_usc_dpll_rate()</code>.</p> <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>SIO4_ASYNC_USC_DPLL_RATE_4X</td><td>This sets the output as the input divided by four. This option is usable only when the DPLL divider rate is being used as the divider rate for CTR1.</td></tr> <tr> <td>SIO4_ASYNC_USC_DPLL_RATE_8X</td><td>This sets the output as the input divided by eight.</td></tr> <tr> <td>SIO4_ASYNC_USC_DPLL_RATE_16X</td><td>This sets the output as the input divided by 16.</td></tr> <tr> <td>SIO4_ASYNC_USC_DPLL_RATE_32X</td><td>This sets the output as the input divided by 32. *</td></tr> </tbody> </table> <p>* This is the hard coded initialization default, which may subsequently be modified as needed to produce the user specified Rx bit rate.</p>	Value	Description	SIO4_ASYNC_USC_DPLL_RATE_4X	This sets the output as the input divided by four. This option is usable only when the DPLL divider rate is being used as the divider rate for CTR1.	SIO4_ASYNC_USC_DPLL_RATE_8X	This sets the output as the input divided by eight.	SIO4_ASYNC_USC_DPLL_RATE_16X	This sets the output as the input divided by 16.	SIO4_ASYNC_USC_DPLL_RATE_32X	This sets the output as the input divided by 32. *
Value	Description										
SIO4_ASYNC_USC_DPLL_RATE_4X	This sets the output as the input divided by four. This option is usable only when the DPLL divider rate is being used as the divider rate for CTR1.										
SIO4_ASYNC_USC_DPLL_RATE_8X	This sets the output as the input divided by eight.										
SIO4_ASYNC_USC_DPLL_RATE_16X	This sets the output as the input divided by 16.										
SIO4_ASYNC_USC_DPLL_RATE_32X	This sets the output as the input divided by 32. *										

4. Operation

This section is intended to provide limited information on the operation of the board and/or the Asynchronous Protocol Library.

4.1. Basic Illustration

The below figure is included to assist individuals in the configuration of the SIO4. 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.

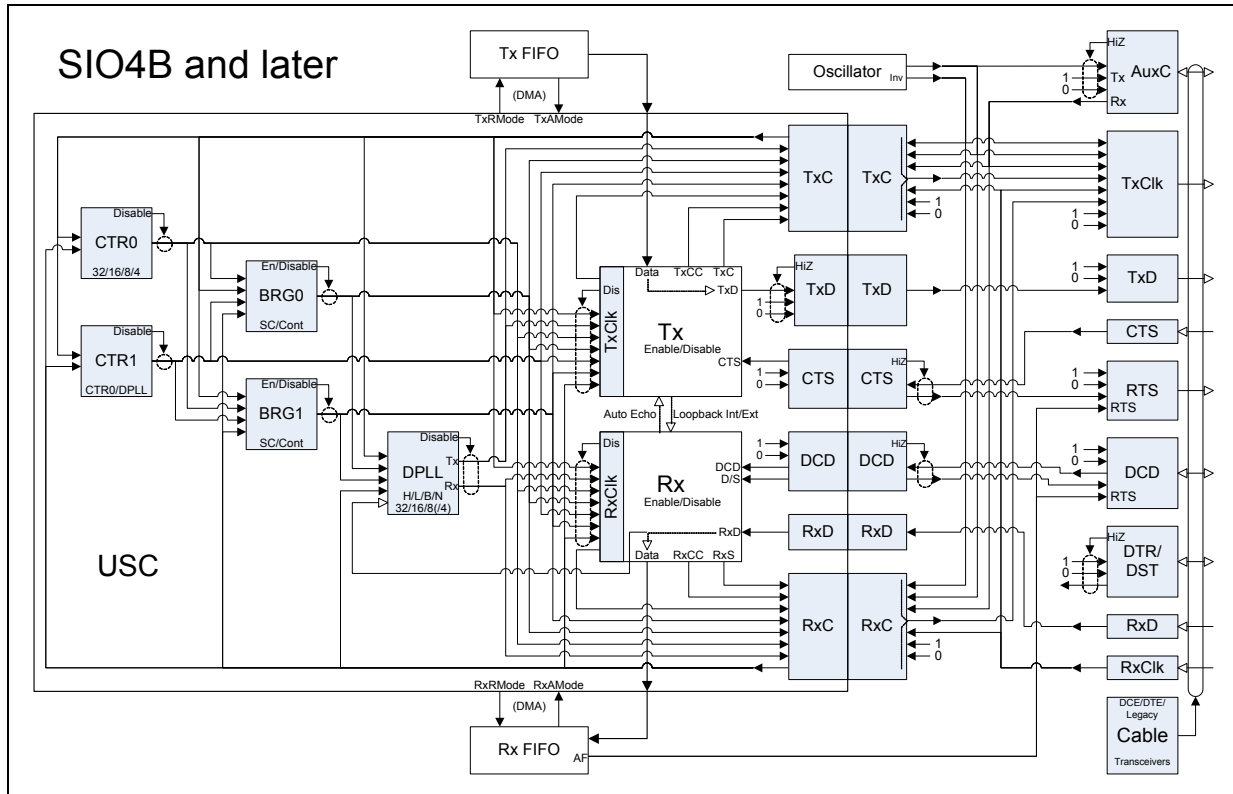


Figure 2 A functional illustration of an SIO4B or later model board.

4.2. Clocking Configurations

The Asynchronous serial protocol requires an Rx Clock signal for data reception. This limits the clocking configuration options when using the SIO4 for full duplex data transfer. The SIO4 receiver clocks in data from the cable's Tx Data signal using the cable's Rx Clock signal. In this case both signals are routed directly from the cable interface to the USC. The SIO4 transmitter clocks out data to the cable's Tx Data signal using the clock driven on the cable's Tx Clock signal. In this case the origin of the cable's Tx Clock signal and the USC's transmit clock is the SIO4's programmable oscillator, which is programmed to the desired Tx bit rate. The Tx Data is routed directly from the USC to the cable interface. Clock and data signal routing for SIO4B and later model boards is illustrated in Figure 3. This signal routing pictured is the configuration produced by the `sio4_async_init()` function, which initializes the `sio4_async_t` structure given as an argument.

NOTE: On SIO4A model boards, full duplex operation requires either that the driver be updated to support programming of the SIO4A's programmable oscillator or an alternate signal routing configuration which may require an Rx Clock cable signal that is active full time.

NOTE: On SIO4 model boards (with no 'A' or 'B' suffix) full duplex operation requires either that the board's fixed frequency oscillator be replaced to match the desired transmit bit rate or an alternate signal routing configuration which may require an Rx Clock cable signal that is active full time.

The below code sample illustrates the minimum required steps to configure an SIO4 for Asynchronous operation. Error checking is omitted for brevity.

```
void config_sample(int fd)
{
    const char*      err      = NULL;
    sio4_async_init_t  init;
    sio4_async_t      async;

    init.tx_bit_rate   = 1000000L;
    init.rx_bit_rate   = 1000000L;
    sio4_async_init(fd, &init, &async, &err);
    sio4_async_set(fd, &async, &err);
}
```

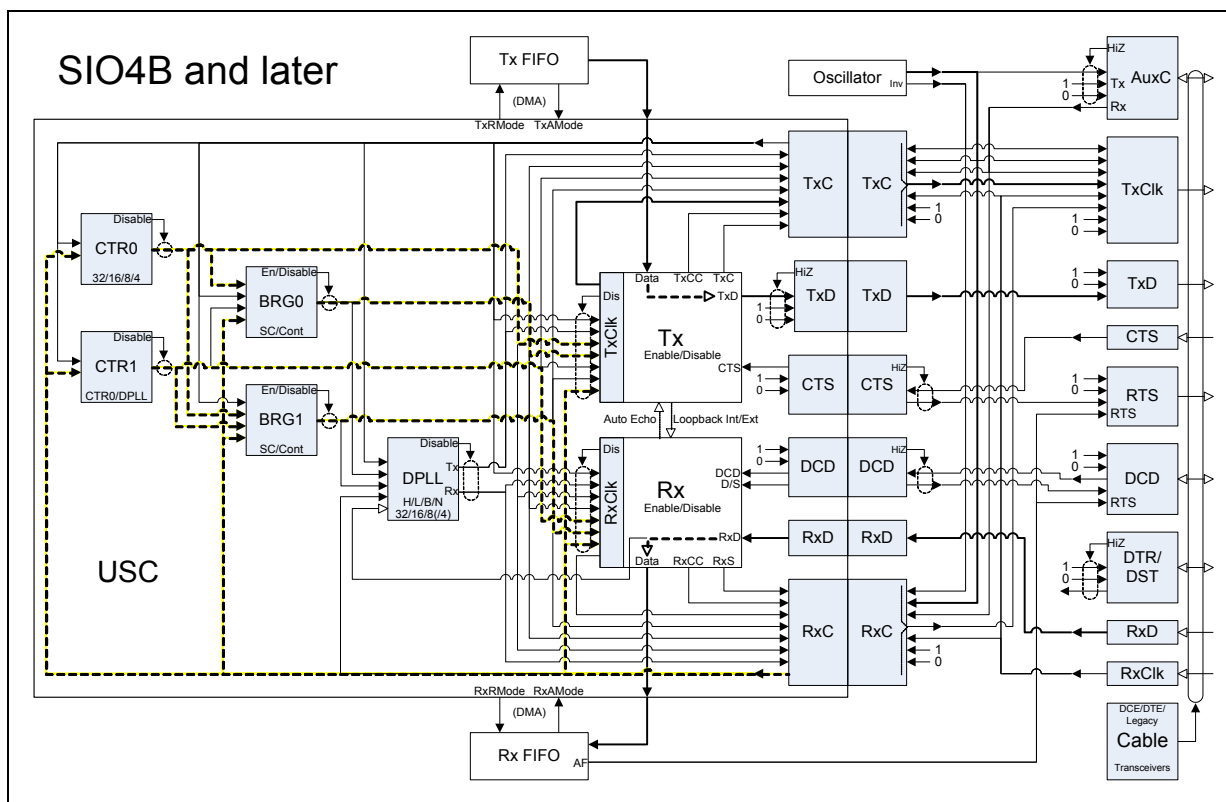


Figure 3 This illustrates the default Asynchronous clock routing on SIO4B and later model boards.

4.3. Error and Status Detection

The serial controller used on the SIO4 incorporates the ability to detect a number of error and other conditions for both the transmit and the receive data streams.

4.3.1. Interrupt Events

The most efficient means of detecting the various conditions, especially errors, is by use of interrupts. The basic steps for this are to enable the interrupts of interest then have a thread wait for a corresponding interrupt event. (See the Interrupt and the Wait Event services in the driver reference manual.) This is illustrated in the following code fragments.

Thread A	Thread B
<pre> For (;;) { ... read SIO4 data if (error recorded) { Error exists in 1) Read buffer, or 2) SIO4 Rx FIFO Resync data stream. } else { Read buffer is error free. } ... } </pre>	<pre> For (;;) { ... Enable desired interrupts. Wait for an interrupt. if (error interrupt occurred) { Record the error. } ... } </pre>

4.3.2. Rx Status Word

The SIO4 can also provide status in the Rx data stream on a per byte basis. This is done by enabling the Rx Status Word feature (`sio4_async_t.rx.status_word`, section 3.3.1.4, page 24). When enabled, the SIO4 places the lower eight bits of the USC's Receive Command/Status Register in the Rx FIFO immediately after the data itself. This allows an application to identify the precise location in the data stream where some Rx related conditions occur. The downside of this is that it doubles the volume of data going through the Rx FIFO and effectively reduces its size by 1/2. Refer to the *Z16C30 Data Handbook* for information on the USC's Receive Command/Status Register.

4.4. Debugging Aids

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

4.4.1. `sio4_async_show()`

The function `sio4_async_show()` (section 3.1.9, page 11) is part of the protocol library interface. The purpose of the function is to produce a human readable report of all fields included in the `sio4_async_t` structure (section 3.3.1, page 14) passed in as a function argument. The function is best used to report the structure's content before it is passed to `sio4_async_set()` (section 3.1.7, page 10) or after it is passed to `sio4_async_get()` (section 3.1.2, page 8). The output can be used with Figure 2 to help visualize the channel configuration reflected by the structure content. When used in conjunction with `sio4_async_set()`, the `sio4_async_show()` output indicates the state that `sio4_async_set()` is expected to produce. When used in conjunction with `sio4_async_get()`, the `sio4_async_show()` output indicates the channel's current state. This may be beneficial after calling `sio4_async_set()` in order to verify the results achieved. The pair of calls may also be used before or after `read()` or `write()` calls in order to help explain the results of individual transfer requests.

4.4.2. 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_async_read()` or `sio4_async_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
fd	This is a file descriptor obtained by calling <code>sio4_async_open()</code> (section 3.1.6, page 10).
gsc	If non-zero, then the output will include a dump of all GSC_SIO4_XXX registers. Refer to <code>sio4.h</code> for a complete list of these registers.
gsc_detail	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.
usc	If non-zero, then the output will include a dump of all GSC_USC_XXX registers. Refer to <code>sio4_usc.h</code> for a complete list of these registers.
usc_detail	If non-zero, then the dump of the USC registers will include detailed information about all register fields, including the field value and the meaning of the value.

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

4.5. Exclusions

4.5.1. Global Rx FIFO Full Configuration

The global Rx FIFO Full Configuration setting (see `SIO4_IOCTL_RX_FIFO_FULL_CFG_GLB` in `sio4.h`) is not included as part of the Asynchronous Protocol Library. It is excluded because the setting can override the channel specific settings for all four channels. If an application is to access this feature it must be done in parallel with use of the Asynchronous Protocol Library.

Document History

Revision	Description
September 7, 2015	Updated to release version 1.1.
December 9, 2014	Initial library release, version 1.0.