SIO4/8

Four/Eight Channel High Speed Serial I/O

All SIO4 and SIO8 Models
All Form Factors
All Standard Zilog Versions
All Standard SYNC Versions
All Standard FASYNC Versions

API Library
Reference Manual

Manual Revision: March 19, 2024
Driver Release Version 3.20.109.x.x

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

Preface

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

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

General Standards Corporation makes no warranty of any kind with regard to this documentation and/or software, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Although extensive editing and reviews are performed before release, **General Standards Corporation** assumes no responsibility for any errors, inaccuracies or omissions herein. This documentation, information and software are made available solely on an "as-is" basis. Nor is there any commitment to update or keep current this documentation.

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

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

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

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

Table of Figures

1. Introduction

This release of the SIO4 API Library and device driver is intended for all versions of the SIO4 and SIO8 that contain standard firmware for either Zilog Z16C30 or SYNC based boards.

> **NOTE:** The device models listed on the front cover are those that are specifically supported by the API given in this document. Other models may be supported, though the level of support may vary. The interface may work with other SIO4 models, but performance may be degraded due to device feature and implementation differences.
>
> **NOTE**: The SIO4 contains four independent serial channels. Each OS specific driver and API Library present each channel as a separate logical device and give separate access to each such device. SIO4 model boards essentially contain two independent SIO4 model boards of the same model.

1.1. Purpose

The purpose of this document is to describe the interface to the SIO4 API Library and, to a lesser extent, the underlying device driver. The API Library software provides the interface between "Application Software" and the device driver. The driver software provides the interface between the API Library and the actual SIO4 hardware. The API Library and driver interfaces are based on the board’s functionality and are primarily IOCTL based.

1.2. Acronyms

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

Acronyms	Description
API	Application Programming Interface
BMDMA	Block Mode DMA
DIL	Driver Interface Library
DMA	Direct Memory Access
DLL	Dynamic Link Library
DMDMA	Demand Mode DMA
DPLL	Digital Phase Lock Loop
GSC	General Standards Corporation
HDLC	High-level Data Link Control
PC104P	This refers to the PC/104+ form factor.
PCI	Peripheral Component Interconnect
PCIe	PCI Express
PIO	Programmed I/O
PMC	PCI Mezzanine Card
USC	Universal Serial Controller

1.3. Definitions

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

Term	Definition
…	This is a shortcut representation of the SIO4 installation directory or any of its subdirectories.
API Library	This is a library that provides application-level access to SIO4 hardware.
Application	This is a user mode process, which runs in user space with user mode privileges.

Driver	This refers to the device driver. Depending on the OS, the driver may be a user space application, a kernel mode process, or something in between. The term Driver and Device Driver are often used interchangeably.
INtime	This refers to the "INtime for Windows" real-time extension for Microsoft Windows. Refer to the *SIO4 INtime for Windows Driver User Manual*.
Library	This is usually a general reference to the API Library.
Linux	This refers to the Linux operating system. Refer to the *SIO4 Linux Driver User Manual*.
Protocol Library	This is any of several libraries that provide a protocol specific interface to the API Library.
RTX	This refers to the real-time extension for Windows known as RTX and RTX64. Refer to the *SIO4 RTX Driver User Manual*.
SIO4	This is used as a general reference to any SIO4 or SIO8 supported by the API Library and device driver.

1.4. Software Overview

1.4.1. Basic Software Architecture

This section describes the general architecture for the basic components that comprise SIO4 applications. The overall architecture is illustrated in Figure 1 below.

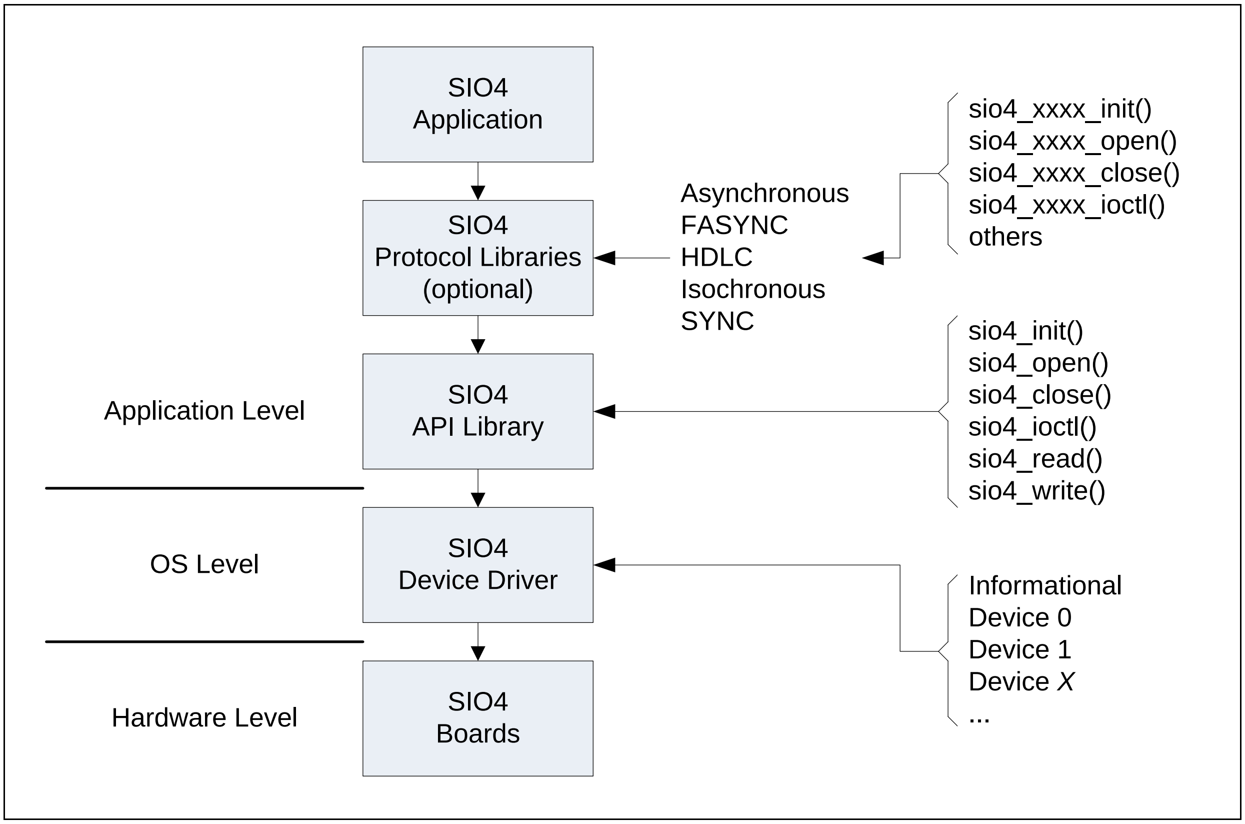

Figure 1 Basic architectural representation.

1.4.2. API Library

The primary means of accessing SIO4 boards is via the SIO4 API Library. This library forms a layer between the application and the driver. Additional information is given in section 4 (page 19). With the library, applications are able to open and close a device and, while open, perform I/O control and read and write operations.

1.4.3. Protocol Libraries

The Protocol Libraries are libraries tailored to specific serial communications protocols. The purpose of the libraries is to simplify one's use of the SIO4 for applications using one of the respective protocols. Each library is an application-level layer that sits immediately above the API Library. Each such Protocol Library implements an interface designed for the chosen serial protocol and thus hides the intricacies of the SIO4's implementation. Each library includes its own documentation. For additional information refer to section 10 (page 143).

1.4.4. Device Driver

The device driver is the host software that provides a means of communicating directly with SIO4 hardware. Depending on the OS, the driver may be a user space application, a kernel mode process, or something in between. The software interface to the device driver is analogous to that of the API Library.

1.5. Hardware Overview

The SIO4 is a four-channel high-speed serial interface I/O board. This board provides for bi-directional serial data transfers between two computers, or one computer and an external peripheral. Once the data link between the two devices is established, the desired transfers can be performed and will become transparent to the user. The SIO4 board includes two DMA controllers and comes with a maximum of 256K Bytes of FIFO storage, which is 32K per channel direction (32K * 2 * 4). Each DMA controller is capable of transferring data to and from host memory; whereas the FIFO help maintain continuous data transfer at the cable interface. The FIFO configuration can vary greatly from one SIO4 version to another (i.e., 32K * 2 * 4 to 1K * 2 *1 to none at all). The SIO4 comes with transceivers that are fixed as RS232 or RS485/422, or with transceivers that are configurable. The SIO4 comes in two basic varieties; SYNC models or Zilog models, which are based on two Z16C30 dual USC chips. Later model SIO4 boards support both models with the mode being software controlled on a per channel basis. The SIO4 also provides for interrupt generation for various states of the board like Sync Character detection, FIFO empty, FIFO full and DMA complete.

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

> **NOTE**: The SIO4 contains four identical, independent serial channels in which each channel contains its own serial controller and its own input and output FIFOs. The driver presents each channel as a separate logical device and gives access to each logical device independently.

1.6. Reference Material

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

- The applicable *SIO4 Driver User Manual* for your operating system from General Standards Corporation.

- The applicable *SIO4/SIO8 User Manual* from General Standards Corporation.

- The *PCI Bus Master Interface Chip* data handbook for the PCI9056/9080 from PLX Technology, Inc.

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

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

* The Zilog material is available from:

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

1.7. Licensing

For licensing information please refer to the text file `LICENSE.txt` in the root installation directory.

2. Installation

For information on driver installation refer to this same section number in the OS specific SIO4 driver user manual.

2.1. Host and Environment Support

For information on host and environment support refer to this same section number in the OS specific SIO4 driver user manual.

2.2. Driver and Device Information

Each driver implements an OS specific means of obtaining generic, high-level information about the driver and the installed boards. The information is given in ASCII text format. Each entry includes an entry name followed immediately by a colon, a space character, and the entry value. Below is an example of what is provided, followed by descriptions of each entry. This information is accessed by passing a device index value of `-1` to the API open service (section 4.6.4, page 24).

```
version: 3.20.109.50
32-bit support: yes
boards: 1
models: SIO4BX
ids: 0x3
```

Entry	Description
`version`	This gives the driver version number in the form `x.x.x.x`.
`32-bit support`	This reports the driver's support for 32-bit applications. This will be either "`yes`" or "`no`" for 64-bit driver builds and "`yes (native)`" for 32-bit builds.
`boards`	This identifies the total number of boards the driver detected.
`models`	This gives a comma separated list of the basic model number for each board the driver detected. The model numbers are listed in the same order that the boards are accessed via the API Library's open function. *
`ids`	This is a list identifying the values read from each board's user jumpers. *

* SIO8 model boards appear as two SIO4 model boards, though they are listed with the SIO8 model numbers. They are distinguishable by a suffix of `.0` or `.1` based on the order the devices are detected by the Plug-N-Play initialization process, respectively.

The API's source for the text provided is as follows.

OS	Source
Linux	The file "`/proc/sio4`".
INtime	The Driver Mailbox "`sio4`".
RTX	The Driver Root Device.

2.3. File List

For the list of primary files included with each release refer to this same section number in the OS specific SIO4 driver user manual.

2.4. Directory Structure

The following table describes the directory structure utilized by the installed files. During installation the directory structure is created and populated with the respective files.

NOTE: Additional or alternate directories may be installed, depending on the OS. For additional information refer to this same section number in the OS specific SIO4 driver user manual.

Directory	**Description**
`sio4/`	This is the driver root directory. It contains the documentation, the Overall Make Script (section 2.7, page 17) and the below listed subdirectories.
…`/api/`	This directory contains the API Library source files (section 4, page 19).
…`/async/`	This directory contains the Asynchronous Protocol Library and related files.
…`/docsrc/`	This directory contains the source files for the code samples given in this document (section 6, page 133).
…`/driver/`	This directory contains the device driver source files (section 5, page 132).
…`/hdlc/`	This directory contains the HDLC Protocol Library and related files.
…`/include/`	This directory contains the header files for the various libraries.
…`/isoc/`	This directory contains the Isochronous Protocol Library and related files.
…`/lib/`	This directory contains all of the libraries built from the driver archive.
…`/samples/`	This directory contains the sample application subdirectories and all of their corresponding source files (section 9, page 142).
…`/sync/`	This directory contains the SYNC Protocol Library and related files.
…`/utils/`	This directory contains the source files for the utility libraries used by the sample applications (section 7, page 134).

2.5. Installation

For installation instructions refer to this same section number in the OS specific SIO4 driver user manual.

2.6. Removal

For removal instructions refer to this same section number in the OS specific SIO4 driver user manual.

2.7. Overall Make Script

Each SIO4 installation includes an OS specific means of building all of the build targets included in the installation. For additional information refer to this same section number in the OS specific SIO4 driver user manual.

2.8. Environment Variables

For environment variable information refer to this same section number in the OS specific SIO4 driver user manual.

3. Main Interface Files

This section gives general information on the suggested device interface files to use when developing SIO4 based applications.

3.1. Main Header File

Throughout the remainder of this document references are made to various header files included as part of the SIO4 driver archive. For ease of use it is suggested that applications include only the single header file shown below rather than individually including those headers identified separately later in this document. Including this header file pulls in all other pertinent SIO4 specific header files. Therefore, sources may include only this one SIO4 header and make files may reference only this one SIO4 include directory.

Description	File	Location	OS
Header File	`sio4_main.h`	`…/include/`	All

3.2. Main Library File

Throughout the remainder of this document references are made to various statically linkable libraries included with the driver. For ease of use it is suggested that applications link only the single library file shown below rather than individually linking those libraries identified separately later in this document. Linking this library file pulls in all other static libraries included with the driver. Therefore, make files may reference only this one SIO4 static library and only this one SIO4 library directory.

Description	File	Location	OS
Library File	`sio4_main.a` `sio4_multi.a`	`…/lib/`	Linux
	`sio4_main.lib` `sio4_multi.lib`	`…\lib\`	INtime
	`sio4_main.lib` `sio4_multi.lib`	`…\lib\`	RTX

NOTE: For applications using the SIO4 and no other GSC devices, link the `sio4_main.a` library. For applications using multiple GSC device types, link the `xxxx_main.a` library for one of the devices and the `xxxx_multi.a` library for the others. Linking multiple `xxxx_main.a` libraries may likely produce link errors due to duplicate symbols being defined. While it may make little or no difference, it is recommended that one choose the `xxxx_main.a` library from the driver with the largest number in positions three (x.x.X.x.x) and/or four (x.x.x.X.x) in the driver release version number.

NOTE: The SIO4 API Library is implemented as a shared or dynamically liked library and is thus not linked with the SIO4 Main Library. Refer to the OS specific driver user manual for information on linking with the respective OS specific SIO4 Main Library.

3.2.1. Build

For information on building the Main Library refer to this same section number in the OS specific SIO4 driver user manual.

3.2.2. Additional Libraries

For information on any additional required libraries refer to this same section number in the OS specific SIO4 driver user manual.

4. API Library

The SIO4 API Library is the software interface between user applications and the SIO4 device driver. The interface is accessed by including the header file `sio4_api.h`.

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

4.1. Files

The library files are summarized in the table below.

Description	Files	Location	OS
Source Files	`*.c`, `*.h` …	…/api/	All
Header File	sio4_api.h	…/include/	All
Library File	libsio4_api.so *	…/lib/ /usr/lib/	Linux
	sio4_api.lib sio4_api.rtl *	…\lib\	INtime
	sio4_lib.lib sio4_lib.dll sio4_lib.rtdll *	…\lib\	RTX

* The Linux run time executable is implemented as a shared object file.
The INtime run time executable is implemented as an INtime DLL.
The RTX run time executable is implemented as an RTX DLL.

4.2. Build

For build instructions refer to this same section number in the OS specific SIO4 driver user manual.

4.3. Library Use

For Library usage information refer to this same section number in the OS specific SIO4 driver user manual.

4.4. Macros

The Library interface includes the following macros, which are defined in `sio4.h`.

4.4.1. IOCTL Codes

The IOCTL macros are documented in section 4.8 (page 27).

4.4.2. Registers

The following gives the complete set of SIO4 registers.

4.4.2.1. GSC Registers

The following table gives the complete set of GSC specific SIO4 registers. For detailed definitions of these registers refer to the relevant SIO4 User Manual. Please note that the set of registers supported by any given board may vary according to model and firmware version. For the set of supported registers and detailed definitions of these registers please refer to the appropriate *SIO4 User Manual*.

NOTE: Refer to the output of the "id" sample application (…`/id/`) for a complete list of the registers supported by the device being accessed.

Macros	**Description**	**Supported Models**
`SIO4_GSC_BCR`	Board Control Register	All models
`SIO4_GSC_BSR`	Board Status Register	SIO4A/B/BX/BXR, SIO8BXS
`SIO4_GSC_CCR`	Clock Control Register	SIO4/A/B/BX w/ USC (legacy support only)
`SIO4_GSC_CSR`	Command/Status Register	All models
`SIO4_GSC_FCR`	FIFO Count Reg	SIO4B/BX/BXR, SIO8BXS, PCI-SIO4A
`SIO4_GSC_FDR`	FIFO Data Register	All models
`SIO4_GSC_FR`	Features Register	SIO4A/B/BX/BXR, SIO8BXS
`SIO4_GSC_FRR`	Firmware Revision Register	All models
`SIO4_GSC_FSR`	FIFO Size Reg	SIO4B/BX/BXR, SIO8BXS, PCI-SIO4A
`SIO4_GSC_FTR`	Firmware Type Register	SIO4BXR, SIO8BXS, some SIO4BX
`SIO4_GSC_GPIOSR`	GPIO Source Register	PCI-SIO4A only
`SIO4_GSC_ICR`	Interrupt Control	All models
`SIO4_GSC_IELR`	Interrupt Edge/Level Register	All but original SIO4
`SIO4_GSC_IHLR`	Interrupt High/Low Register	All but original SIO4
`SIO4_GSC_IOCR`	I/O Control Register	PMC-SIO4AR-SYNC only
`SIO4_GSC_ISR`	Interrupt Status	All models
`SIO4_GSC_PCR`	Prog Clock Reg	SIO4A only
`SIO4_GSC_POCSR`	Prog Osc Ctrl/Stat	SIO4B/BX/BXR, SIO8BXS
`SIO4_GSC_PORAR`	Prog Osc RAM Adrs	SIO4B/BX/BXR, SIO8BXS
`SIO4_GSC_PORDR`	Prog Osc RAM Data	SIO4B/BX/BXR, SIO8BXS
`SIO4_GSC_PORD2R`	Prog Osc RAM Data 2	Some SIO8BXS
`SIO4_GSC_PSRCR`	Pin Source Reg	All but SIO4A-SYNC and original SIO4
`SIO4_GSC_PSTSR`	Pin Ststus Reg	SIO4B/BX/BXR, SIO8BXS
`SIO4_GSC_RAR`	Rx Almost Register	All models
`SIO4_GSC_RCR`	Rx Count Reg	All -SYNC models
`SIO4_GSC_SBR`	Sync Byte Register	All models w/ USC
`SIO4_GSC_TAR`	Tx Almost Register	All models
`SIO4_GSC_TCR`	Tx Count Reg	All -SYNC models
`SIO4_GSC_TSR`	Time Stamp Register	SIO4BXR w/ USC

4.4.2.2. PCI Configuration Registers

Access to the PCI registers is seldom required so these registers are not listed here. For the complete list of the PCI register identifiers refer to header files `gsc_pci9056.h` and `gsc_pci9080.h`, which are automatically included via `sio4.h`.

4.4.2.3. PLX Feature Set Registers

Access to the PLX registers is seldom required so these registers are not listed here. For the complete list of the PLX register identifiers refer to header files `gsc_pci9056.h` and `gsc_pci9080.h`, which are automatically included via `sio4.h`.

4.4.2.4. Zilog Z16C30 USC Registers

The following table gives the complete set of Zilog USC registers. These registers appear only on SIO4 models which include the Zilog Z16C30 USC chips. This essentially refers to all boards without the SYNC suffix in the model number.

Macros	Description
`SIO4_USC_CCAR`	Channel Command/Address Register
`SIO4_USC_CCR`	Channel Control Register
`SIO4_USC_CCSR`	Channel Command/Status Register
`SIO4_USC_CMCR`	Clock Mode Control Register
`SIO4_USC_CMR`	Channel Mode Register
`SIO4_USC_DCCR`	Daisy-Chain Control Register
`SIO4_USC_HCR`	Hardware Configuration Register
`SIO4_USC_ICR`	Interrupt Control Register
`SIO4_USC_IOCR`	I/O Control Register
`SIO4_USC_IVR`	Interrupt Vector Register
`SIO4_USC_MISR`	Miscellaneous Interrupt Status Register
`SIO4_USC_PRR`	Primary Reserved Register
`SIO4_USC_RCCR`	Receive Character Count Register
`SIO4_USC_RCLR`	Receive Count Limit Register
`SIO4_USC_RCSR`	Receive Command Status Register
`SIO4_USC_RDR`	Receive Data Register
`SIO4_USC_RICR`	Receive Interrupt Control Register
`SIO4_USC_RMR`	Receive Mode Register
`SIO4_USC_RSR`	Receive Sync Register
`SIO4_USC_SICR`	Status Interrupt Control Register
`SIO4_USC_SRR`	Secondary Reserved Register
`SIO4_USC_TC0R`	Time Constant 0 Register
`SIO4_USC_TC1R`	Time Constant 1 Register
`SIO4_USC_TCCR`	Transmit Character Count Register
`SIO4_USC_TCLR`	Transmit Count Limit Register
`SIO4_USC_TCSR`	Transmit Command/Status Register
`SIO4_USC_TDR`	Transmit Data Register
`SIO4_USC_TICR`	Transmit Interrupt Control Register
`SIO4_USC_TMCR`	Test Mode Control Register
`SIO4_USC_TMDR`	Test Mode Data Register
`SIO4_USC_TMR`	Transmit Mode Register
`SIO4_USC_TSR`	Transmit Sync Register

4.5. Data Types

The data types used by the API Library are described with the IOCTL services with which they are used. For additional information refer to section 4.7 (page 27).

4.6. Functions

The interface includes the following functions. The return values reflect the completion status of the requested operation. A return value less than zero always reflects an error condition. The table below summarizes the error status values. For the I/O function, read and write, non-negative return values reflect the number of bytes transferred between the application and the interface. A value equal to the requested transfer size indicates complete success. Return values less than the requested transfer size indicate that the I/O timeout expired. For the other API function calls a return value of zero indicates success.

Return Value	Description	OS
`-1` to `-499`	This is the value "`(-errno)`" (see `errno.h`).	All †
`-500` to `-999`	This is the value returned from the Driver Interface Library. *	INtime
`<= -1000`	This is "`(int)(GetLastRtError()+1000)`" forced to a negative value.	

* Applicable error codes, if any, are defined in the header `os_common.h`.

† The RTX error return values are limited to this range.

4.6.1. sio4_close()

This function is the entry point to close a connection made via the API's open call (section 4.6.4, page 24). The device is put in an initialized state before this call returns. The programmable oscillator, if present, is not modified.

Prototype

```
int sio4_close(int fd);
```

Argument	Description
`fd`	This is the file descriptor obtained from the open service (section 4.6.4, page 24).

Return Value	Description
`0`	The operation succeeded.
`< 0`	An error occurred. See error value description above.

Example

```
#include <stdio.h>

#include "sio4_dsl.h"

int sio4_close_dsl(int fd)
{
    int errs;
    int ret;

    ret = sio4_close(fd);

    if (ret)
        printf("ERROR: sio4_close() returned %d\n", ret);

    errs    = ret ? 1 : 0;
    return(errs);
}
```

4.6.2. sio4_init()

This function is the entry point to initializing the SIO4 API Library and must be the first call into the Library. This function may be called more than once, but only the first successful call actually initializes the library. Subsequent calls perform no operation at all. All other API calls return a failure status when the API Library is not initialized.

Prototype

```
int sio4_init(void);
```

Return Value	Description
`0`	The operation succeeded.
`< 0`	An error occurred. See error value description above.

Example

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

```
#include "sio4_dsl.h"

int sio4_init_dsl(void)
{
    int errs;
    int ret;

    ret = sio4_init();

    if (ret)
        printf("ERROR: sio4_init() returned %d\n", ret);

    errs    = ret ? 1 : 0;
    return(errs);
}
```

4.6.3. sio4_ioctl()

This function is the entry point to performing setup and control operations on a SIO4 board. The specific operation performed varies according to the `request` argument. The `request` argument also governs the use and interpretation of the `arg` argument. The set of supported options for the `request` argument consists of the IOCTL services supported by the driver, which are defined in section 4.7 (page 27).

NOTE: IOCTL operations are not supported for an open on device index `-1`.

Prototype

```
int sio4_ioctl(int fd, int request, void* arg);
```

Argument	Description
`fd`	This is the file descriptor obtained from the open service (section 4.6.4, page 24).
`request`	This specifies the desired operation to be performed (section 4.7, page 27).
`arg`	This is specific to the IOCTL operation specified by the `request` argument.

Return Value	Description
`0`	The operation succeeded.
`< 0`	An error occurred. See error value description above.

Example

```
#include <stdio.h>

#include "sio4_dsl.h"

int sio4_ioctl_dsl(int fd, int request, void* arg)
{
    int errs;
    int ret;

    ret = sio4_ioctl(fd, request, arg);

    if (ret)
        printf("ERROR: sio4_ioctl() returned %d\n", ret);
    errs    = ret ? 1 : 0;
    return(errs);
}
```

4.6.4. sio4_open()

This function is the entry point to open a connection to an SIO4 serial channel. Before returning, the initialize IOCTL service is called to reset all hardware and software settings to their defaults. The programmable oscillator, if present, is not modified.

Prototype

```
int sio4_open(int device, int share, int* fd);
```

Argument	Description
`device`	This is the zero-based index of the SIO4 channel to access. * †
`share`	Open the device in Shared Access Mode? If non-zero, the device is opened in Shared Access Mode (see below). If zero, the device is opened in Exclusive Access Mode (see below).
`fd`	The device handle is returned here. The pointer cannot be NULL. Values returned are as follows. **Value** — **Description** `>= 0` — This is the handle to use to access the device in subsequent calls. `-1` — There was an error. The device is not accessible.

* The index value `-1` can also be given to acquire driver information (see section 2.2, page 16).

† Each SIO4 contains four independent serial channels accessed by four sequential index numbers. That is, 0 to 3 for the first SIO4, 4 to 7 for the second, and so on.

Return Value	Description
`0`	The operation succeeded.
`< 0`	An error occurred. See error value description above.

Example

```
#include <stdio.h>

#include "sio4_dsl.h"

int sio4_open_dsl(int device, int share, int* fd)
{
    int errs;
    int ret;

    ret = sio4_open(device, share, fd);

    if (ret)
        printf("ERROR: sio4_open() returned %d\n", ret);

    errs    = ret ? 1 : 0;
    return(errs);
}
```

4.6.4.1. Access Modes

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

Shared Access Mode:

Shared Access Mode allows multiple applications to access the same device simultaneously. In this mode, the first successful open request returns with the device in an initialized state. Subsequent successful Shared Access Mode open requests do not affect the state of the device. Once opened in Shared Access Mode, the device access remains in this mode until all Shared Access Mode accesses release the device with a close request.

Exclusive Access Mode:

Exclusive Access Mode allows a single application to acquire exclusive access to a device. In this mode, a successful open request returns with the device in an initialized state. While open in this mode all subsequent open requests will fail regardless of the access mode requested. Once opened in Exclusive Access Mode, the device access remains in this mode until released by the application with a close request.

4.6.5. sio4_read()

This function is the entry point to read data from a serial channel. The function reads up to `bytes` bytes from a serial channel. The return value is the number of bytes actually read.

> **NOTE:** If an open was performed using index of `-1`, then read requests will acquire driver information (section 2.2, page 16).
>
> **NOTE:** For additional information refer to the Data Transfer Modes section (section 8.2, page 135).

Prototype

```
int sio4_read(int fd, void* dst, size_t bytes);
```

Argument	Description
`fd`	This is the file descriptor obtained from the open service (section 4.6.4, page 24).
`dst`	This is the destination for the data read from the serial channel.
`bytes`	This is the desired number of bytes to read.

Return Value	Description
`0` to `bytes`	The operation succeeded. A value less than `bytes` indicates that the I/O timeout period (section 4.8.3.6, page 38) lapsed before the entire request could be satisfied.
`< 0`	An error occurred. See error value description above.

Example

```
#include <stdio.h>

#include "sio4_dsl.h"

int sio4_read_dsl(int fd, void* dst, size_t bytes, size_t* qty)
{
    int errs;
    int ret;

    ret = sio4_read(fd, dst, bytes);

    if (ret < 0)
        printf("ERROR: sio4_read() returned %d\n", ret);
    if (qty)
        qty[0]  = (ret < 0) ? 0 : (size_t) ret;

    errs    = (ret < 0) ? 1 : 0;
    return(errs);
}
```

4.6.6. sio4_write()

This function is the entry point to write data to a serial channel. The function writes up to `bytes` bytes to a serial channel. The return value is the number of bytes actually written.

Prototype

```
int sio4_write(int fd, const void* src, size_t bytes);
```

Argument	Description
`fd`	This is the file descriptor obtained from the open service (section 4.6.4, page 24).
`src`	This is the source for the data written to the serial channel.
`bytes`	This is the desired number of bytes to write.

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

Example

```
#include <stdio.h>

#include "sio4_dsl.h"

int sio4_write_dsl(int fd, const void* src, size_t bytes, size_t* qty)
{
    int errs;
    int ret;

    ret = sio4_write(fd, src, bytes);

    if (ret < 0)
        printf("ERROR: sio4_write() returned %d\n", ret);

    if (qty)
        qty[0]  = (ret < 0) ? 0 : (size_t) ret;

    errs    = (ret < 0) ? 1 : 0;
    return(errs);
}
```

4.7. IOCTL Services

The SIO4 API Library and device driver implement the following IOCTL services. Each service is described along with the applicable `sio4_ioctl()` function arguments. The IOCTL services are numerous and are therefore organized into the following basic categories. Within some categories the services are further subdivided.

- Services that are common to both Zilog based and SYNC style boards (section 4.8, page 27)
- Services specific to Zilog based boards (section 4.9, page 60)
- Services specific to the USC on the Zilog based boards (section 4.10, page 66)
- Services specific to SYNC boards (section 4.11, page 117)
- Services specific to SIO4A boards (section 4.12, page 128)

All services whose argument data type is `s32*` accept the value of `-1` being passed to the driver. If there is a setting associated with the service, then passing in the value `-1` is a request for the current setting. If the service is supported by the attached SIO4, then the current setting is returned. If the service is not supported, then the value returned by the driver will be `-1`. If there isn't a setting associated with the service, then passing in the value `-1` is a request to determine if the service is supported. If the service is supported, then the value one is returned. If the service is not supported, then the value `-1` is returned.

> **NOTE**: The set of supported services can also be determined by looking at the output of the *services* sample application (under the `…/services/` directory). The application output specifically identifies the services that are supported and will conditionally list those that are not.

For device configuration it is recommended that applications rely upon the provided Protocol Libraries. In these cases, the Protocol Libraries perform complete configuration of the SIO4 serial channel. When a provided Protocol Library cannot be used, it is recommended that end users consider adapting a provided library to their own needs. If not using a provided Protocol Library or if interacting with an SIO4 in a more dynamic fashion a comprehensive understanding of the IOCTL services becomes critical.

4.8. Common IOCTL Services

This section describes the IOCTL services that are common across SYNC and Zilog based versions of the SIO4. The services are organized into groups according to their functionality and applicability.

4.8.1. Cable Interface Specific IOCTL Services

These IOCTL services relate to operation of the board's cable interface.

4.8.1.1. SIO4_IOCTL_CBL_MODE

This service configures the routing of the signals at the cable interface when the transceivers are enabled via the `SIO4_IOCTL_XCVR_ENABLE` service (section 4.8.1.4, page 28).

Usage

Argument	Description
`request`	`SIO4_IOCTL_CBL_MODE`
`arg`	`s32*`

The table below lists the options used with this service.

NOTE: Use care when enabling interrupts driven by clock signals. If the clock is present, then the influx of interrupts could make the system unresponsive.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_CBL_MODE_DCE`	This refers to routing the signals for the DCE configuration.	
`SIO4_CBL_MODE_DTE`	This refers to routing the signals for the DTE configuration.	

4.8.1.2. SIO4_IOCTL_CBL_PIN_STATUS

This service returns the state of the signals at the cable interface. The state is provided by reading the Pin Status Register. Bits that do not reflect cable pin status are masked off and returned as zero.

Usage

Argument	Description
`request`	`SIO4_IOCTL_CBL_PIN_STATUS`
`arg`	`s32*`

The table below lists the options used with this service. The valid set of bits varies with different models and different firmware versions.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` or above	N/A	The status requested.

4.8.1.3. SIO4_IOCTL_LOOP_BACK

This service controls the LEDs that correspond to the channel being accessed.

Usage

Argument	Description
`request`	`SIO4_IOCTL_LOOP_BACK`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_LOOP_BACK_DISABLE`	This refers to loopback operation being disabled.	
`SIO4_LOOP_BACK_EXTERNAL`	This refers to use of external loopback operation.	
`SIO4_LOOP_BACK_INTERNAL`	This refers to use of internal loopback operation.	

4.8.1.4. SIO4_IOCTL_XCVR_ENABLE

This service enables and disables the transceivers.

Usage

Argument	Description
`request`	`SIO4_IOCTL_XCVR_ENABLE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_XCVR_ENABLE_NO`	This refers to the transceivers being disabled. When disabled, the legacy mode functionality is active, if it is supported.	
`SIO4_XCVR_ENABLE_YES`	This refers to the transceivers being enabled. When enabled, the legacy mode functionality is disabled, if it is supported.	

4.8.1.5. SIO4_IOCTL_XCVR_PROTOCOL

This service enables and disables the transceivers.

Usage

Argument	Description
`request`	`SIO4_IOCTL_XCVR_PROTOCOL`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_XCVR_PROTOCOL_RS232`	This refers to RS-232.	
`SIO4_XCVR_PROTOCOL_RS422_RS423_1`	This refers to RS-422/RS-423 (mode 1).	
`SIO4_XCVR_PROTOCOL_RS422_RS423_2`	This refers to RS-422/RS-423 (mode 2).	
`SIO4_XCVR_PROTOCOL_RS422_RS485`	This refers to RS-422/RS-485.	
`SIO4_XCVR_PROTOCOL_RS423`	This refers to RS-423.	
`SIO4_XCVR_PROTOCOL_RS530`	This refers to RS-530 (mode 1).	
`SIO4_XCVR_PROTOCOL_RS530A`	This refers to RS-530 (mode 2).	
`SIO4_XCVR_PROTOCOL_V35`	This refers to V.35 (mode 1).	
`SIO4_XCVR_PROTOCOL_V35A`	This refers to V.35 (mode 2).	
`SIO4_XCVR_PROTOCOL_UNKNOWN`	N/A	This indicates that the transceiver type is unknown.

4.8.1.6. SIO4_IOCTL_XCVR_TERM

This service enables and disables the termination resisters that are part of the transceivers. This feature is not applicable to all transceiver protocols. When not applicable, the setting has no affect.

Usage

Argument	Description
`request`	`SIO4_IOCTL_XCVR_TERM`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_XCVR_TERM_DISABLE`	This refers to disabling the termination resisters.	
`SIO4_XCVR_TERM_ENABLE`	This refers to enabling the termination resisters.	

4.8.2. FIFO Specific IOCTL Services

These IOCTL services relate to the operation of the Rx FIFO and the Tx FIFO.

4.8.2.1. SIO4_IOCTL_FIFO_SPACE_CFG

This service configures the relative distribution of FIFO space between the receiver and transmitter.

Usage

Argument	Description
`request`	`SIO4_IOCTL_FIFO_SPACE_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_FIFO_SPACE_CFG_RX_2X`	This refers to the Rx FIFO being twice the size of the Tx FIFO.	
`SIO4_FIFO_SPACE_CFG_TX_2X`	This refers to the Tx FIFO being twice the size of the Rx FIFO.	

4.8.2.2. SIO4_IOCTL_RX_FIFO_AE

This service configures the Rx FIFO Almost Empty threshold level. When applying a setting the FIFO is reset and the current content is lost. The Rx FIFO Almost Empty status is asserted when the Rx FIFO contains *Almost Empty* or fewer data values.

Usage

Argument	Description
`request`	`SIO4_IOCTL_RX_FIFO_AE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFFFF`	This is the valid range for this service.	

4.8.2.3. SIO4_IOCTL_RX_FIFO_AF

This service configures the Rx FIFO Almost Full threshold level. When applying a setting the Rx FIFO is reset and the current content is lost. The Rx FIFO Almost Full status is asserted when the Rx FIFO can receive *Almost Full* or fewer data values before becoming full.

Usage

Argument	Description
`request`	See table above.
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFFFF`	This is the valid range for this service.	

4.8.2.4. SIO4_IOCTL_RX_FIFO_FILL_LEVEL

This service retrieves the current Rx FIFO fill level. If the fill level cannot be determined exactly, then it is approximated based on the FIFO status.

Usage

Argument	Description
`request`	`SIO4_IOCTL_RX_FIFO_FILL_LEVEL`
`arg`	`s32*`

The table below lists the options used with this service. Values returned are from zero to `0xFFFF`, but will not exceed the size of the FIFO.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFFFF`	This is the valid range for this service.	

4.8.2.5. SIO4_IOCTL_RX_FIFO_FULL_CFG

This service configures how a receiver responds to an Rx FIFO Full condition.

> **NOTE:** This service refers to the channel specific settings, not to the global setting. The channel specific settings are ignored of the global setting is set to the *disable* option.
>
> **NOTE:** If the *disable* setting is selected and the feature becomes activated, then this may indicate that an Rx FIFO overflow has also occurred.
>
> **NOTE:** If the *disable* setting is selected and the feature causes the receiver to become disabled, then the application must implement a recovery procedure. For –SYNC boards this means reading data from the Rx FIFO and re-enabling the receiver. For this, refer to the `SIO4_IOCTL_SYNC_RX_ENABLE` service (section 4.11.3.5, page 120). For Z16C30 boards this means reading data from the Rx FIFO, flushing the USC's Rx FIFO and re-enabling the USC receiver. For this, refer to the `SIO4_IOCTL_USC_SEND_COMMAND` service (section 4.10.15.6, page 98, use the `SIO4_USC_SEND_CMD_RX_FIFO_PURGE` command), and the `SIO4_IOCTL_USC_RX_ENABLE` service (section 4.10.18.7, page 104).

Usage

Argument	Description
`request`	`SIO4_IOCTL_RX_FIFO_FULL_CFG`
`arg`	`s32*`

The table below lists the options used by this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_RX_FIFO_FULL_CFG_DISABLE`	This refers to disabling the receiver. Once the receiver is disabled, it must be manually enabled.	
`SIO4_RX_FIFO_FULL_CFG_OVER`	This refers to letting the FIFO overrun.	

4.8.2.6. SIO4_IOCTL_RX_FIFO_FULL_CFG_GLB

This service configures how the receivers respond to an Rx FIFO Full condition.

> **NOTE:** This is a global setting that affects all four channels.
>
> **NOTE:** If the *disable* option is selected, then it overrides the channel specific setting.
>
> **NOTE:** If the *disable* setting is selected and the feature becomes activated, then this may indicate that an Rx FIFO overflow has also occurred.
>
> **NOTE:** If the *disable* setting is selected and the feature causes the receiver to become disabled, then the application must implement a recovery procedure. For –SYNC boards this means reading data from the Rx FIFO and re-enabling the receiver. For this, refer to the `SIO4_IOCTL_SYNC_RX_ENABLE` service (section 4.11.3.5, page 120). For Z16C30 boards this means reading data from the Rx FIFO, flushing the USC's Rx FIFO and re-enabling the USC receiver. For this, refer to the `SIO4_IOCTL_USC_SEND_COMMAND` service (section 4.10.15.6, page 98, use the `SIO4_USC_SEND_CMD_RX_FIFO_PURGE` command), and the `SIO4_IOCTL_USC_RX_ENABLE` service (section 4.10.18.7, page 104).

Usage

Argument	Description
`request`	`SIO4_IOCTL_RX_FIFO_FULL_CFG_GLB`
`arg`	`s32*`

The table below lists the options used by this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_RX_FIFO_FULL_CFG_GLB_DISABLE`	This refers to disabling the receiver. Once a receiver is disabled, it must be manually enabled. If selected, then this overrides the channel specific settings.	
`SIO4_RX_FIFO_FULL_CFG_GLB_OVER`	This refers to letting the FIFO overrun.	

4.8.2.7. SIO4_IOCTL_RX_FIFO_OVERRUN

This service operates on the Rx FIFO Overrun condition.

Usage

Argument	Description
`request`	`SIO4_IOCTL_RX_FIFO_OVERRUN`
`arg`	`s32*`

The table below lists the options used by this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_FIFO_OVERRUN_CLEAR`	This clears the overrun status.	N/A
`SIO4_FIFO_OVERRUN_TEST`	This checks the overrun status.	N/A

`SIO4_FIFO_OVERRUN_NO`	N/A	This indicates that an overrun has not occurred.
`SIO4_FIFO_OVERRUN_YES`	N/A	This indicates that an overrun has occurred.

4.8.2.8. SIO4_IOCTL_RX_FIFO_RESET

This service resets the Rx FIFO. The FIFO content is lost when it is reset. It also clears the associated overrun and underrun status bits.

Usage

Argument	Description
`request`	`SIO4_IOCTL_RX_FIFO_RESET`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests support status.	The service is unsupported.
`1`	Reset the FIFO.	The service is supported or the FIFO was reset.

4.8.2.9. SIO4_IOCTL_RX_FIFO_STATUS

This service reports the Rx FIFO Fill Level relative to the Rx FIFO Threshold Level.

Usage

Argument	Description
`request`	`SIO4_IOCTL_RX_FIFO_STATUS`
`arg`	`s32*`

The table below lists the options used by this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_FIFO_STATUS_EMPTY`	N/A	The FIFO is empty.
`SIO4_FIFO_STATUS_ALMOST_EMPTY`	N/A	The FIFO is Almost Empty. The FIFO contains *Almost Empty* or fewer data values (section 4.8.2.2, page 30).
`SIO4_FIFO_STATUS_MEDIUM`	N/A	The FIFO is between Almost Empty and Almost Full.
`SIO4_FIFO_STATUS_ALMOST_FULL`	N/A	The FIFO is Almost Full. The FIFO can receive *Almost Full* or fewer data value before becoming full (section 4.8.2.3, page 30).
`SIO4_FIFO_STATUS_FULL`	N/A	The FIFO is full.

4.8.2.10. SIO4_IOCTL_RX_FIFO_UNDERRUN

This service operates on the Rx FIFO Underrun condition.

Usage

Argument	Description
`request`	`SIO4_IOCTL_RX_FIFO_UNDERRUN`
`arg`	`s32*`

The table below lists the options used by this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_FIFO_UNDERRUN_CLEAR`	This clears the underrun status.	N/A
`SIO4_FIFO_UNDERRUN_TEST`	This checks the underrun status.	N/A
`SIO4_FIFO_UNDERRUN_NO`	N/A	Indicates that an underrun has not occurred.
`SIO4_FIFO_UNDERRUN_YES`	N/A	Indicates that an underrun has occurred.

4.8.2.11. SIO4_IOCTL_TX_FIFO_AE

This service configures the Tx FIFO Almost Empty threshold level. When applying a setting the FIFO is reset and the current content is lost. The Tx Almost Empty status is asserted when the Tx FIFO contains *Almost Empty* or fewer data values.

Usage

Argument	Description
`request`	`SIO4_IOCTL_TX_FIFO_AE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFFFF`	This is the valid range for this service.	

4.8.2.12. SIO4_IOCTL_TX_FIFO_AF

This service configures the Tx FIFO Almost Full threshold level. When applying a setting the Tx FIFO is reset and the current content is lost. The Tx Almost Full status is asserted when the Tx FIFO can receive *Almost Full* or fewer data values before becoming full.

Usage

Argument	Description
`request`	`SIO4_IOCTL_TX_FIFO_AF`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFFFF`	This is the valid range for this service.	

4.8.2.13. SIO4_IOCTL_TX_FIFO_EMPTY_CFG

This service configures how the transmitter responds to a Tx FIFO Empty condition.

Usage

Argument	Description
`request`	`SIO4_IOCTL_TX_FIFO_EMPTY_CFG`
`arg`	`s32*`

The table below lists the options used by this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_TX_FIFO_EMPTY_CFG_IGNORE`	This refers to ignoring the condition.	
`SIO4_TX_FIFO_EMPTY_CFG_TX_OFF`	This refers to disabling the transmitter.	

4.8.2.14. SIO4_IOCTL_TX_FIFO_FILL_LEVEL

This service retrieves the current Tx FIFO fill level. If the fill level cannot be determined exactly, then it is approximated based on the FIFO status.

Usage

Argument	Description
`request`	`SIO4_IOCTL_TX_FIFO_FILL_LEVEL`
`arg`	`s32*`

The table below lists the options used with this service. Values returned will not exceed the size of the FIFO.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFFFF`	N/A	This is the valid range for this service.

4.8.2.15. SIO4_IOCTL_TX_FIFO_OVERRUN

This service operates on the Tx FIFO Overrun condition.

Usage

Argument	Description
`request`	`SIO4_IOCTL_TX_FIFO_OVERRUN`
`arg`	`s32*`

The table below lists the options used by this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_FIFO_OVERRUN_CLEAR`	This clears the overrun status.	N/A
`SIO4_FIFO_OVERRUN_TEST`	This checks the overrun status.	N/A
`SIO4_FIFO_OVERRUN_NO`	N/A	Indicates that an overrun has not occurred.

`SIO4_FIFO_OVERRUN_YES`	N/A	Indicates that an overrun has occurred.

4.8.2.16. SIO4_IOCTL_TX_FIFO_RESET

This service resets the Tx FIFO. The FIFO content is lost when it is reset. It also clears the associated overrun status bit.

Usage

Argument	Description
`request`	`SIO4_IOCTL_TX_FIFO_RESET`
`arg`	`s32*`

The table below lists the options used by this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests support status.	The service is unsupported.
`1`	Reset the FIFO.	The service is supported or the FIFO was reset.

4.8.2.17. SIO4_IOCTL_TX_FIFO_STATUS

This service reports the Tx FIFO Fill Level relative to the Tx FIFO Threshold Level.

Usage

Argument	Description
`request`	`SIO4_IOCTL_TX_FIFO_STATUS`
`arg`	`s32*`

The table below lists the options used by this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_FIFO_STATUS_EMPTY`	N/A	The FIFO is empty.
`SIO4_FIFO_STATUS_ALMOST_EMPTY`	N/A	The FIFO is Almost Empty. The FIFO contains *Almost Empty* or fewer data values (section 4.8.2.11, page 34).
`SIO4_FIFO_STATUS_MEDIUM`	N/A	The FIFO is between Almost Empty and Almost Full. Both the Almost Full and the Almost Empty status bits are set.
`SIO4_FIFO_STATUS_ALMOST_FULL`	N/A	The FIFO is Almost Full. The FIFO can receive *Almost Full* or fewer data values before becoming full (section 4.8.2.12, page 34).
`SIO4_FIFO_STATUS_FULL`	N/A	The FIFO is full.

4.8.3. I/O Specific IOCTL Services

These IOCTL services relate to the I/O operation when transferring data either to or from the SIO4.

4.8.3.1. SIO4_IOCTL_RX_IO_ABORT

This service aborts an active read request, if one is active.

Usage

Argument	Description
`request`	`SIO4_IOCTL_RX_IO_ABORT`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_IO_ABORT_NO`	This refers to the I/O request not being aborted.	
`SIO4_IO_ABORT_YES`	This refers to the I/O request being aborted, or the service is supported.	

4.8.3.2. SIO4_IOCTL_RX_IO_DMA_THRESHOLD

This service configures the minimum size of DMA transfers to be performed by the read service. If the Rx FIFO contains less than the specified amount of data, then the driver will wait a single system timer interval before trying again. However, if the Rx FIFO can fulfill what remains of the current request, or if the input stream appears to be idle, then the driver will perform a transfer rather than wait for more data.

> **NOTE:** The Rx I/O DMA Threshold applies to all Block Mode DMA transfers. The threshold only applies to Demand Mode DMA transfers when the Rx I/O Timeout is zero.

Usage

Argument	Description
`request`	`SIO4_IOCTL_RX_IO_DMA_THRESHOLD`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver
`-1`	Requests current setting.
`0` to FIFO Size	This is the valid range for this service. A value of zero disables the feature. The default is 45.

4.8.3.3. SIO4_IOCTL_RX_IO_MODE

This service configures the I/O mode used to transfer data from the channel during read requests.

Usage

Argument	Description
`request`	`SIO4_IOCTL_RX_IO_MODE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.

`GSC_IO_MODE_BMDMA`	This refers to Block Mode DMA.
`GSC_IO_MODE_DMDMA`	This refers to Demand Mode DMA.
`GSC_IO_MODE_PIO`	This refers to PIO. This is the default.

4.8.3.4. SIO4_IOCTL_RX_IO_OVERRUN

This service configures how the read service responds to overrun conditions.

Usage

Argument	Description
`request`	`SIO4_IOCTL_RX_IO_OVERRUN`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_IO_OVERRUN_CHECK`	This refers to checking for overruns before performing I/O.	
`SIO4_IO_OVERRUN_IGNORE`	This refers to ignoring the overrun status.	

4.8.3.5. SIO4_IOCTL_RX_IO_PIO_THRESHOLD

This service configures the read request size at or below which requests are automatically performed using PIO. The default threshold setting is 44 bytes. (For more information see section 8.3, page 137.)

Usage

Argument	Description
`request`	`SIO4_IOCTL_RX_IO_PIO_THRESHOLD`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFFFFFFF`	This is the valid range for this service.	

4.8.3.6. SIO4_IOCTL_RX_IO_TIMEOUT

This service configures the read timeout period. When the duration of an I/O request takes this number of seconds or longer, the request will end.

Usage

Argument	Description
`request`	`SIO4_IOCTL_RX_IO_TIMEOUT`
`arg`	`s32*`

Valid argument values are in the range from zero to `3600`, `-1`, and `SIO4_IO_TIMEOUT_INFINITE`. The value zero tells the driver to read as much data as possible from the Rx FIFO, but not to wait for additional data when none is available. A value of `-1` is used to retrieve the current setting. If the option `SIO4_IO_TIMEOUT_INFINITE` is used, then the driver will wait indefinitely rather than timing out. The default is 10 seconds.

4.8.3.7. SIO4_IOCTL_RX_IO_UNDERRUN

This service configures how read requests will respond to Rx FIFO Underrun conditions.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_RX_IO_UNDERRUN`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_IO_UNDERRUN_CHECK`	This refers to check for underruns before performing I/O.	
`SIO4_IO_UNDERRUN_IGNORE`	This refers to ignoring the underrun status.	

4.8.3.8. SIO4_IOCTL_TX_IO_ABORT

This service aborts an active write request, if one is active.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_TX_IO_ABORT`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_IO_ABORT_NO`	This refers to the I/O request not being aborted.	
`SIO4_IO_ABORT_YES`	This refers to the I/O request being aborted, or the service is supported.	

4.8.3.9. SIO4_IOCTL_TX_IO_DMA_THRESHOLD

This service configures the minimum size of DMA transfers to be performed by the write service. If the Tx FIFO has less than the specified amount of available space, then the driver will wait a single system timer interval before trying again. However, if the Tx FIFO can accommodate what remains of the current request, or if the output stream appears to be idle, then the driver will perform a transfer rather than wait for more space.

> **NOTE:** The Tx I/O DMA Threshold applies to all Block Mode DMA transfers. The threshold only applies to Demand Mode DMA transfers when the Tx I/O Timeout is zero.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_TX_IO_DMA_THRESHOLD`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver
`-1`	Requests current setting.
`0` to FIFO Size	This is the valid range for this service. A value of zero disables the feature. The default is 45.

4.8.3.10. SIO4_IOCTL_TX_IO_MODE

This service configures the I/O mode used to transfer data to the channel during write requests.

Usage

Argument	Description
`request`	`SIO4_IOCTL_TX_IO_MODE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`GSC_IO_MODE_BMDMA`	This refers to Block Mode DMA.	
`GSC_IO_MODE_DMDMA`	This refers to Demand Mode DMA.	
`GSC_IO_MODE_PIO`	This refers to PIO. This is the default.	

4.8.3.11. SIO4_IOCTL_TX_IO_OVERRUN

This service configures how the write service responds to overrun conditions.

Usage

Argument	Description
`request`	`SIO4_IOCTL_TX_IO_OVERRUN`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_IO_OVERRUN_CHECK`	This refers to check for overruns before performing I/O.	
`SIO4_IO_OVERRUN_IGNORE`	This refers to ignoring the overrun status.	

4.8.3.12. SIO4_IOCTL_TX_IO_PIO_THRESHOLD

This service configures the write request size at or below which requests are automatically performed using PIO. The default threshold setting is 44 bytes. (For more information see section 8.3, page 137.)

Usage

Argument	Description
`request`	`SIO4_IOCTL_TX_IO_PIO_THRESHOLD`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFFFFFFF`	This is the valid range for this service.	

4.8.3.13. SIO4_IOCTL_TX_IO_TIMEOUT

This service configures the maximum duration of write requests to the driver. The units are seconds.

Usage

Argument	Description
`request`	`SIO4_IOCTL_TX_IO_TIMEOUT`
`arg`	`s32*`

Valid argument values are in the range from zero to `3600`, `-1`, and `SIO4_IO_TIMEOUT_INFINITE`. The value zero tells the driver to write as much data as possible to the Tx FIFO, but not to wait for additional free space when none is available. A value of `-1` is used to retrieve the current setting. If the option `SIO4_IO_TIMEOUT_INFINITE` is used, then the driver will wait indefinitely rather than timing out. The default is 10 seconds.

4.8.4. Interrupt Specific IOCTL Services

These IOCTL services relate to the interrupt operation and detection.

4.8.4.1. SIO4_IOCTL_IRQ_GSC_CFG_HIGH

This service configures the triggering polarity of the firmware specific interrupts.

Usage

Argument	Description
`request`	`SIO4_IOCTL_IRQ_GSC_CFG_HIGH`
`arg`	`s32*`

The table below lists the options used with this service. Valid arguments values may include any combination of the below macros, or none at all. If a bit is set, then the corresponding interrupt is configured to trigger on a level-high or high going state. If a bit is clear, then the corresponding interrupt is configured to trigger on a level-low or low going state.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_IRQ_RX_ENV`	This refers to the interrupt that responds to the Rx Envelope signal. (`SYNC` model boards only)	
`SIO4_IRQ_RX_FIFO_AF`	This refers to the interrupt that responds to the Rx FIFO Almost Full state.	
`SIO4_IRQ_RX_FIFO_E`	This refers to the interrupt that response to the Rx FIFO Empty state.	
`SIO4_IRQ_RX_FIFO_F`	This refers to the interrupt that response to the Rx FIFO Full state.	
`SIO4_IRQ_RX_SPARE`	This refers to the interrupt that response to the Rx Spare signal. (`SYNC` model boards only)	
`SIO4_IRQ_SYNC_BYTE`	This refers to the interrupt that responds to received characters matching the configured SYNC byte. (`Z16C30` model boards only)	
`SIO4_IRQ_TX_FIFO_AE`	This refers to the interrupt that responds to the Tx FIFO Almost Empty state.	
`SIO4_IRQ_TX_FIFO_E`	This refers to the interrupt that responds to the Tx FIFO Empty state.	
`SIO4_IRQ_TX_FIFO_F`	This refers to the interrupt that responds to the Tx FIFO Full state.	

`SIO4_IRQ_USC`	This refers to the interrupt that response to USC generated interrupts. (`Z16C30` model boards only)

4.8.4.2. SIO4_IOCTL_IRQ_GSC_ENABLE

This service enables or disables the firmware specific interrupts.

Usage

Argument	Description
`request`	`SIO4_IOCTL_IRQ_GSC_ENABLE`
`arg`	`s32*`

The table below lists the options used with this service. Valid arguments values may include any combination of the below macros, or none at all. If a bit is set, then the corresponding interrupt is enabled. If a bit is clear, then the corresponding interrupt is disabled.

> **NOTE**: Level triggered interrupts are disabled when they occur (except for the USC interrupt). Edge triggered interrupts are not disabled when they occur.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_IRQ_RX_ENV`	This refers to the interrupt that responds to the Rx Envelope signal. (`SYNC` model boards only)	
`SIO4_IRQ_RX_FIFO_AF`	This refers to the interrupt that responds to the Rx FIFO Almost Full state.	
`SIO4_IRQ_RX_FIFO_E`	This refers to the interrupt that response to the Rx FIFO Empty state.	
`SIO4_IRQ_RX_FIFO_F`	This refers to the interrupt that response to the Rx FIFO Full state.	
`SIO4_IRQ_RX_SPARE`	This refers to the interrupt that response to the Rx Spare signal. (`SYNC` model boards only)	
`SIO4_IRQ_SYNC_BYTE`	This refers to the interrupt that responds to received characters matching the configured SYNC byte. (`Z16C30` model boards only)	
`SIO4_IRQ_TX_FIFO_AE`	This refers to the interrupt that responds to the Tx FIFO Almost Empty state.	
`SIO4_IRQ_TX_FIFO_E`	This refers to the interrupt that responds to the Tx FIFO Empty state.	
`SIO4_IRQ_TX_FIFO_F`	This refers to the interrupt that responds to the Tx FIFO Full state.	
`SIO4_IRQ_USC`	This refers to the interrupt that response to USC generated interrupts. (`Z16C30` model boards only)	

4.8.4.3. SIO4_IOCTL_IRQ_USC_ENABLE

This service enables or disables USC specific interrupts.

Usage

Argument	Description
`request`	`SIO4_IOCTL_IRQ_USC_ENABLE`
`arg`	`s32*`

The table below lists the options used with this service. Valid arguments values may include any combination of the below macros, or none at all. If a bit is set, then the corresponding interrupt is enabled. If a bit is clear, then the corresponding interrupt is disabled.

> **NOTE**: Use care when enabling interrupts driven by clock signals. If the clock is present, then the influx of interrupts could make the system unresponsive.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_IRQ_USC_IOP_CTS_FALL`	This refers to the interrupt that responds to a high-to-low transition on the USC's CTS pin.	
`SIO4_IRQ_USC_IOP_CTS_RISE`	This refers to the interrupt that responds to a low-to-high transition on the USC's CTS pin.	
`SIO4_IRQ_USC_IOP_DCD_FALL`	This refers to the interrupt that responds to a high-to-low transition on the USC's DCD pin.	
`SIO4_IRQ_USC_IOP_DCD_RISE`	This refers to the interrupt that responds to a low-to-high transition on the USC's DCD pin.	
`SIO4_IRQ_USC_IOP_RXC_FALL`	This refers to the interrupt that responds to a high-to-low transition on the USC's RxC pin. *	
`SIO4_IRQ_USC_IOP_RXC_RISE`	This refers to the interrupt that responds to a low-to-high transition on the USC's RxC pin. *	
`SIO4_IRQ_USC_IOP_RXREQ_FALL`	This refers to the interrupt that responds to a high-to-low transition on the USC's RxReq pin. *	
`SIO4_IRQ_USC_IOP_RXREQ_RISE`	This refers to the interrupt that responds to a low-to-high transition on the USC's RxReq pin. *	
`SIO4_IRQ_USC_IOP_TXC_FALL`	This refers to the interrupt that responds to a high-to-low transition on the USC's TxC pin. *	
`SIO4_IRQ_USC_IOP_TXC_RISE`	This refers to the interrupt that responds to a low-to-high transition on the USC's TxC pin. *	
`SIO4_IRQ_USC_IOP_TXREQ_FALL`	This refers to the interrupt that responds to a high-to-low transition on the USC's TxReq pin. *	
`SIO4_IRQ_USC_IOP_TXREQ_RISE`	This refers to the interrupt that responds to a low-to-high transition on the USC's TxReq pin. *	
`SIO4_IRQ_USC_MISC_BRG0_ZERO`	This refers to the interrupt that responds to the BRG0 clocking signal.	
`SIO4_IRQ_USC_MISC_BRG1_ZERO`	This refers to the interrupt that responds to the BRG1 clocking signal.	
`SIO4_IRQ_USC_MISC_DPLL_DESYNC`	This refers to the interrupt that responds to the DPLL becoming unsynchronized.	
`SIO4_IRQ_USC_MISC_RCC_UNDERRUN`	This refers to the interrupt that responds to an underrun of the RCC FIFO.	
`SIO4_IRQ_USC_RX_ABORT_PAR_ERROR`	This refers to the interrupt that responds to the condition where the next character to be taken from the USC FIFO has the Rx Abort status or the Parity Error status.	
`SIO4_IRQ_USC_RX_BOUND`	This refers to the interrupt that responds to the condition where the next character to be taken from the USC FIFO has the Rx Bound status.	
`SIO4_IRQ_USC_RX_BREAK_ABORT`	This refers to the interrupt that responds to the condition where the next character to be taken from the USC FIFO has the Rx Break or Rx Abort status.	
`SIO4_IRQ_USC_RX_DATA`	This refers to the interrupt that responds to the receipt of a character. *	
`SIO4_IRQ_USC_RX_EXITED_HUNT`	This refers to the interrupt that responds to the condition where the receiver has exited the Hunt mode.	
`SIO4_IRQ_USC_RX_IDLE_RECEIVED`	This refers to the interrupt that responds to the condition where the next character to be taken from the USC FIFO has the Rx Idle status.	
`SIO4_IRQ_USC_RX_OVERRUN`	This refers to the interrupt that responds to the condition where a character has been received for placement in the USC's Rx FIFO, but the FIFO is full.	
`SIO4_IRQ_USC_TX_ABORT_SENT`	This refers to the interrupt that responds to the condition where the transmitter has sent out Abort.	

`SIO4_IRQ_USC_TX_CRC_SENT`	This refers to the interrupt that responds to the condition where the transmitter has sent out a CRC.
`SIO4_IRQ_USC_TX_DATA`	This refers to the interrupt generated when the transmitter has placed data in the USC Tx FIFO. *
`SIO4_IRQ_USC_TX_END_SENT`	This refers to the interrupt that responds to the condition where the transmitter has sent out the end of a message or frame.
`SIO4_IRQ_USC_TX_IDLE_SENT`	This refers to the interrupt that responds to the condition where the transmitter has sent out an Idle condition.
`SIO4_IRQ_USC_TX_PREAMBLE_SENT`	This refers to the interrupt that responds to the condition where the transmitter has sent out the Preamble.
`SIO4_IRQ_USC_TX_UNDERRUN`	This refers to the interrupt that responds to the condition where the transmitter is ready for data, but the USC Tx FIFO is empty.

* These interrupts are disabled when they occur due to the high rate expected for these interrupt source.

4.8.5. Miscellaneous IOCTL Services

4.8.5.1. SIO4_IOCTL_INITIALIZE

This service initializes a channel to the state it was in when the channel was first opened. All channel hardware and software settings are initialized. The programmable oscillator is unaffected.

Usage

Argument	Description
`request`	`SIO4_IOCTL_INITIALIZE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests support status.	The service is unsupported.
`1`	Perform initialization.	The service is supported or initialization was performed.

4.8.5.2. SIO4_IOCTL_LED_CHANNEL

This service controls the LEDs for the channel being accessed. The specific functionality accessed depends on the channel LED implementation as described below.

Usage

Argument	Description
`request`	`SIO4_IOCTL_LED_CHANNEL`
`arg`	`s32*`

The table below lists the valid options for those boards which don’t support any channel LEDs.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.

The table below lists the valid options for those boards which support a single bi-color LED for each channel. This configuration supports the colors RED and GREEN, when can each be turned on or off. In addition, both can be turned on at the same time producing a slightly ORANGE color.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_LED_CHAN_1_LED_BOTH`	This option turns both the RED and the GREEN on, producing a slightly ORANGE color.	
`SIO4_LED_CHAN_1_LED_GREEN`	This option turns the GREEN on and the RED off.	
`SIO4_LED_CHAN_1_LED_OFF`	This option turns the LED off.	
`SIO4_LED_CHAN_1_LED_RED`	This option turns the RED on and the GREEN off.	

The table below lists the valid options for those boards which support two independent, single-color LEDs for each channel. All board LEDs are numbered sequentially on the PCB. This configuration permits each LED pair to individually represent numbers from zero through three.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_LED_CHAN_2_LED_OFF`	This option turns both LEDs off.	
`SIO4_LED_CHAN_2_LED_LOW_ON`	This option turns the lower numbered LED on and the higher numbered LED off.	
`SIO4_LED_CHAN_2_LED_UP_ON`	This option turns the higher numbered LED on and the lower numbered LED off.	
`SIO4_LED_CHAN_2_LED_BOTH_ON`	This option turns both channel LED on.	

4.8.5.3. SIO4_IOCTL_LED_MAIN

This service controls the board's main LEDs, which operate independently of the Channel LEDs. The specific functionality accessed depends on the main LED implementation as described below.

Usage

Argument	Description
`request`	`SIO4_IOCTL_LED_MAIN`
`arg`	`s32*`

The table below lists the valid options for those boards which don't support any main LEDs.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.

The table below lists the valid options for those boards which support a single bi-color main LED. This configuration supports the colors RED and GREEN, when can each be turned on or off. Turning on both is not supported as it does not produce an alternate color.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_LED_MAIN_1_LED_GREEN`	This option turns the GREEN on and the RED off.	
`SIO4_LED_MAIN_1_LED_OFF`	This option turns the LED off.	
`SIO4_LED_MAIN_1_LED_RED`	This option turns the RED on and the GREEN off.	

The table below lists the valid options for those boards which support three independent, single-color main LEDs. All board LEDs are numbered sequentially on the PCB. This configuration permits the main LEDs to represent numbers from zero through seven. The main LEDs are enabled via a mask. If a mask bit is set, then the corresponding LED is turned on. If a mask bit is clear, then the corresponding LED is turned off.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0x7`	This refers to the specified set of LEDs being on or off.	

4.8.5.4. SIO4_IOCTL_QUERY

This service provides information on an SIO4's feature set.

Usage

Argument	Description
`request`	`SIO4_IOCTL_QUERY`
`arg`	`s32*`

The table below lists the options used with this service. The feature in question is passed to the service. The results are returned by the service. Most feature queries return zero if the feature is not supported and non-zero if the feature is supported. Other features return a value specific to feature in question.

Values	Description
`-1`	This requests support information for this service. The value returned is zero.
`SIO4_QUERY_BOARD_RESET`	This refers to the Board Reset feature.
`SIO4_QUERY_BUS_SPEED`	This refers to the PCI Bus speed of the board. The options returned are 33 (for 33MHz) or 66 (for 66MHz).
`SIO4_QUERY_BUS_WIDTH`	This refers to the PCI Bus width of the board. The options returned are 32 (for 32-bits) or 64 (for 64-bits).
`SIO4_QUERY_CHANNEL_QTY`	This refers to the total number of channels for the board. This should be either four (for SIO4 boards) or eight (for SIO8 boards).
`SIO4_QUERY_COUNT`	This refers to the total number of query options supported by the driver.
`SIO4_QUERY_DEVICE_QTY`	This refers to the number of SIO4 devices on the board. This should be either one (for SIO4 boards) or two (for SIO8 boards).
`SIO4_QUERY_DEVICE_TYPE`	This refers to the basic board series type. This should be `GSC_DEV_TYPE_SIO4`.
`SIO4_QUERY_DMDMA_SCD`	This indicates if the board supports Single Cycle Disable for Demand Mode DMA operations.
`SIO4_QUERY_FIFO_SIZE_RX`	This refers to the size of the channel's Rx FIFO.
`SIO4_QUERY_FIFO_SIZE_TOTAL`	This refers to the total size for all four Tx and Rx FIFOs.
`SIO4_QUERY_FIFO_SIZE_TX`	This refers to the size of the channel's Tx FIFO.
`SIO4_QUERY_FIFO_SPACE_CFG`	This refers to the firmware having support for the FIFO Space Configuration option.
`SIO4_QUERY_FORM_FACTOR`	This refers to the board's basic form factor. See `sio4_form_factor_t` below.
`SIO4_QUERY_FW_TYPE`	This refers to the firmware's basic type, according to the board's model number. Valid options are listed in section 4.8.5.4.1 on page 48.
`SIO4_QUERY_INDEX_CHANNEL`	This refers to the index of the channel being accessed. This should be from zero to three.
`SIO4_QUERY_INDEX_DEVICE`	This refers to the zero-based index of the SIO4 being accessed.

`SIO4_QUERY_INDEX_SUBDEVICE`	This refers to the sub-device index for the board being accessed. This should be zero for all SIO4 boards and zero or one for all SIO8 boards. *
`SIO4_QUERY_IRQ_32`	This refers to the number of basic firmware interrupts supported by the firmware. This should be 32 or 16.
`SIO4_QUERY_LEGACY_CABLE`	This refers to firmware support for the legacy cable interface (i.e., upper and lower references rather than DTE or DCE).
`SIO4_QUERY_LED_CHANNEL`	This requests the number of channel specific LEDs.
`SIO4_QUERY_LED_MAIN`	This requests the number of main LEDs.
`SIO4_QUERY_MODEL_BASE`	This refers to the base model number. See `sio4_model_t` below.
`SIO4_QUERY_MODEL_FASYNC`	This refers to the board being of the –FASYNC type.
`SIO4_QUERY_MODEL_SYNC`	This refers to the board being of the –SYNC type.
`SIO4_QUERY_MODEL_Z16C30`	This refers to the board being of the USC type.
`SIO4_QUERY_MP`	This refers to the board's support for the Multi-Protocol transceiver firmware feature.
`SIO4_QUERY_MP_CHIP`	This indicates the Multi-Protocol transceiver chip type on the board. See `sio4_mp_chip_t` below.
`SIO4_QUERY_MP_PROGRAM`	This refers to the Multi-Protocol feature being programmable. If supported, then the transceiver type is software selectable.
`SIO4_QUERY_OSC_CHIP`	This refers to the master oscillator chip type. See `sio4_osc_chip_t` below.
`SIO4_QUERY_OSC_MEASURE`	This refers to the firmware's support for measuring the oscillator frequency.
`SIO4_QUERY_OSC_PD_MAX`	This refers to the firmware's Maximum Post Divider value for the oscillator programming feature.
`SIO4_QUERY_OSC_PER_CHANNEL`	This indicates if the board supports an oscillator per channel.
`SIO4_QUERY_OSC_PROGRAM`	This indicates if the on-board oscillator is programmable.
`SIO4_QUERY_REG_ACSR`	This indicates support for the Async Control/Status Register.
`SIO4_QUERY_REG_BSR`	This indicates support for the Board Status Register.
`SIO4_QUERY_REG_CCR`	This indicates support for the Clock Control Register.
`SIO4_QUERY_REG_FCR`	This indicates support for the FIFO Count Register.
`SIO4_QUERY_REG_FR`	This indicates support for the Features Register.
`SIO4_QUERY_REG_FSR`	This indicates support for the FIFO Size Register.
`SIO4_QUERY_REG_FTR`	This indicates support for the Firmware Type Register.
`SIO4_QUERY_REG_GPIOSR`	This indicates support for the GPIO Source Register.
`SIO4_QUERY_REG_IELR`	This indicates support for the Interrupt Edge/Level Register.
`SIO4_QUERY_REG_IHLR`	This indicates support for the Interrupt High/Low Register.
`SIO4_QUERY_REG_IOCR`	This indicates support for the I/O Control Register.
`SIO4_QUERY_REG_PCR`	This indicates support for the Programmable Clock Register.
`SIO4_QUERY_REG_POCSR`	This indicates support for the Programmable Oscillator Control/Status Register.
`SIO4_QUERY_REG_PORAR`	This indicates support for the Programmable Oscillator RAM Address Register.
`SIO4_QUERY_REG_PORDR`	This indicates support for the Programmable Oscillator RAM Data Register.

`SIO4_QUERY_REG_PORD2R`	This indicates support for the Programmable Oscillator RAM Data 2 Register.
`SIO4_QUERY_REG_PSRCR`	This indicates support for the Pin Source Register.
`SIO4_QUERY_REG_PSRCR_BITS`	This indicates the set of bits that are supported by the Pin Source Register. This is zero if the register is unsupported.
`SIO4_QUERY_REG_PSTSR`	This indicates support for the Pin Status Register.
`SIO4_QUERY_REG_PSTSR_BITS`	This indicates the set of bits that are supported by the Pin Status Register. This is zero if the register is unsupported.
`SIO4_QUERY_REG_RCR`	This indicates support for the Rx Count Register.
`SIO4_QUERY_REG_SBR`	This indicates support for the Sync Byte Register.
`SIO4_QUERY_REG_TCR`	This indicates support for the Tx Count Register.
`SIO4_QUERY_REG_TGR`	This indicates support for the Tx Gap Register.
`SIO4_QUERY_REG_TSR`	This indicates support for the Time Stamp Register.
`SIO4_QUERY_RX_FIFO_FULL_CFG`	This is the channel specific setting that indicates if the firmware's response to an Rx FIFO Full status is configurable.
`SIO4_QUERY_RX_FIFO_FULL_CFG_GLB`	This is the global setting that indicates if the firmware's response to an Rx FIFO Full status is configurable.
`SIO4_QUERY_RX_FIFO_OVERRUN`	This indicates if the Rx FIFO Overrun feature is supported.
`SIO4_QUERY_RX_FIFO_UNDERRUN`	This indicates if the Rx FIFO Underrun feature is supported.
`SIO4_QUERY_RX_STATUS_WORD`	This indicates if the firmware can put the USC's Rx Status Register in the Rx FIFO with each character pulled from the USC's Rx FIFO.
`SIO4_QUERY_SIO4_TYPE`	This indicates the board's basic SIO4 type See `sio4_type_t` below.
`SIO4_QUERY_TIME_STAMP`	This indicates if firmware supports the Time Stamp feature.
`SIO4_QUERY_TX_FIFO_EMPTY_CFG`	This indicates if the firmware's response to a Tx FIFO Empty status is configurable.
`SIO4_QUERY_TX_FIFO_OVERRUN`	This indicates if the Tx FIFO Overrun feature is supported.
`SIO4_QUERY_USER_JUMPER_QTY`	This indicates the number of user jumpers supported on the board. *
`SIO4_QUERY_USER_JUMPER_SENSE`	This indicates the bit value associated with the "jumper present" condition.
`SIO4_QUERY_USER_JUMPER_VAL`	This indicates the value read from the user jumpers. Zero is returned if the jumpers are not supported.

* To get the board's overall user id use the Jumper Value and the sub-device index (`SIO4_QUERY_USER_JUMPER_VAL` and `SIO4_QUERY_INDEX_SUBDEVICE`, respectively).

4.8.5.4.1. SIO4_QUERY_FW_TYPE and sio4_fw_type_t

Using the `SIO4_QUERY_FW_TYPE` query option returns the below values from the `sio4_fw_type_t` enumeration.

Value	**Description**
`SIO4_FW_TYPE_FASYNC`	The firmware is for a –FASYNC board.
`SIO4_FW_TYPE_SYNC`	The firmware is for a –SYNC board.
`SIO4_FW_TYPE_Z16C30`	The firmware is for a Z16C30 based board.

4.8.5.4.2. SIO4_QUERY_FORM_FACTOR and sio4_form_factor_t

Using the `SIO4_QUERY_FORM_FACTOR` query option returns the below values from the `sio4_form_factor_t` enumeration.

Value	Description
`SIO4_FORM_FACTOR_CPCI`	The board is of the Compact PCI form factor. *
`SIO4_FORM_FACTOR_PC104P`	The board is of the PC/104+ form factor. *
`SIO4_FORM_FACTOR_PCI`	The board is of the PCI form factor. *
`SIO4_FORM_FACTOR_PCI66`	The board is of the PCI form factor. †
`SIO4_FORM_FACTOR_PCIE`	This refers to the single lane PCI Express form factor.
`SIO4_FORM_FACTOR_PCIE4`	This refers to the four lane PCI Express form factor.
`SIO4_FORM_FACTOR_PMC`	The board is of the PMC form factor. *
`SIO4_FORM_FACTOR_PMC66`	The board is of the PMC form factor. †
`SIO4_FORM_FACTOR_UNKNOWN`	The form factor could not be determined.
`SIO4_FORM_FACTOR_XMC`	The board is of the XMC form factor.

* The PCI bus is 32-bits wide and runs at up to 33MHz.
† The PCI bus is 32-bits wide and runs at up to 66MHz.

4.8.5.4.3. SIO4_QUERY_MODEL_BASE and sio4_model_t

Using the `SIO4_QUERY_MODEL_BASE` query option returns the below values from the `sio4_model_t` enumeration.

Value	Description
`SIO4_MODEL_SIO4`	The basic model number is SIO4.
`SIO4_MODEL_SIO4A`	The basic model number is SIO4A.
`SIO4_MODEL_SIO4A_SYNC`	The basic model number is SIO4A-SYNC.
`SIO4_MODEL_SIO4AR`	The basic model number is SIO4AR.
`SIO4_MODEL_SIO4AR_SYNC`	The basic model number is SIO4ARx-SYNC.
`SIO4_MODEL_SIO4ARHM`	The basic model number is SIO4ARHM.
`SIO4_MODEL_SIO4ARHM_SYNC`	The basic model number is SIO4ARHM-SYNC.
`SIO4_MODEL_SIO4B`	The basic model number is SIO4B.
`SIO4_MODEL_SIO4B_SYNC`	The basic model number is SIO4B-SYNC.
`SIO4_MODEL_SIO4BX`	The basic model number is SIO4BX.
`SIO4_MODEL_SIO4BX_SYNC`	The basic model number is SIO4BX-SYNC.
`SIO4_MODEL_SIO4BX2`	The basic model number is SIO4BX2.
`SIO4_MODEL_SIO4BX2_SYNC`	The basic model number is SIO4BX2-SYNC.
`SIO4_MODEL_SIO4BXR`	The basic model number is SIO4BXR.
`SIO4_MODEL_SIO4BXR_FASYNC`	The basic model number is SIO4BXR-FASYNC.
`SIO4_MODEL_SIO4BXR_SYNC`	The basic model number is SIO4BXR-SYNC.
`SIO4_MODEL_SIO8BX2`	The basic model number is SIO48BX2.
`SIO4_MODEL_SIO8BX2_SYNC`	The basic model number is SIO48BX2-SYNC.
`SIO4_MODEL_SIO8BX2V_FASYNC`	The basic model number is SIO48BX2V-FASYNC.
`SIO4_MODEL_SIO8BXS`	The basic model number is SIO48BXS.
`SIO4_MODEL_SIO8BXS_SYNC`	The basic model number is SIO48BXS-SYNC.

4.8.5.4.4. SIO4_QUERY_MP_CHIP and sio4_mp_chip_t

Using the `SIO4_QUERY_MP_CHIP` query option returns the below values from the `sio4_mp_chip_t` enumeration.

Value	Description
`SIO4_MP_CHIP_FIXED`	The transceivers are not programmable

`SIO4_MP_CHIP_SP508`	The transceivers are SP508.

4.8.5.4.5. SIO4_QUERY_OSC_CHIP and sio4_osc_chip_t

Using the `SIO4_QUERY_OSC_CHIP` query option returns the below values from the `sio4_osc_chip_t` enumeration.

Value	**Description**
`SIO4_OSC_CHIP_FIXED`	The board has a fixed frequency oscillator.
`SIO4_OSC_CHIP_IDC2053B`	The board has a single IDC2053B oscillator.
`SIO4_OSC_CHIP_IDC2053B_4`	The board has four IDC2053B oscillators.
`SIO4_OSC_CHIP_CY22393`	The board has a single CY22393 oscillator.
`SIO4_OSC_CHIP_CY22393_2`	The board has two CY22393 oscillators.

4.8.5.4.6. SIO4_QUERY_SIO4_TYPE and sio4_type_t

Using the `SIO4_QUERY_SIO4_TYPE` query option returns the below values from the `sio4_type_t` enumeration.

Value	**Description**
`SIO4_TYPE_SIO4`	The board is a version of the original SIO4.
`SIO4_TYPE_SIO4A`	The board is a version of the SIO4A.
`SIO4_TYPE_SIO4AR`	The board is a version of the SIO4AR.
`SIO4_TYPE_SIO4AHRM`	The board is a version of the SIO4ARHM.
`SIO4_TYPE_SIO4B`	The board is a version of the SIO4B.
`SIO4_TYPE_SIO4BX`	The board is a version of the SIO4BX.
`SIO4_TYPE_SIO4BX2`	The board is a version of the SIO4BX2.
`SIO4_TYPE_SIO4BXR`	The board is a version of the SIO4BXR.
`SIO4_TYPE_SIO8BX2`	The board is a version of the SIO8BX2.
`SIO4_TYPE_SIO8BXS`	The board is a version of the SIO8BXS.

4.8.6. Oscillator Specific IOCTL Services

These IOCTL services relate to operation of the on-board oscillator(s).

4.8.6.1. SIO4_IOCTL_OSC_MEASURE

This service measures the master oscillator frequency. The results are reported in hertz.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_OSC_MEASURE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`0` or above	N/A	This refers to the measured results.
`1`	Request that the measurement be made.	N/A

4.8.6.2. SIO4_IOCTL_OSC_PROGRAM

This service programs the master oscillator for the specified frequency, in hertz.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_OSC_PROGRAM`
`arg`	`s32*`

The table below lists the options used with this service. The range for valid values depends on the type of oscillator on the board being accessed.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`> 0`	This refers to the requested frequency.	

The valid range for requests is as follows, though the frequency produced through programming may be more limited.

Oscillator Type	**`Fmin` (Hz)**	**`Fmax` (Hz)**	**Information**
CY22393	5	20,000,000	The minimum achievable frequency depends on the SIO4 model and the firmware version.
IDC2053G	1,000,000	20,000,000	Oscillator programming is not implemented.
Fixed	▨	▨	Oscillator programming is unsupported. *
Unknown	▨	▨	Oscillator programming is unsupported. *

* The on-board oscillator is not programmable, but it can be replaced with an oscillator from 1MHz to 20MHz.

4.8.6.3. SIO4_IOCTL_OSC_REFERENCE

This service specifies the master oscillator reference frequency, in hertz. The default is 20,000,000 Hz. The purpose of this service is to let the driver know the exact frequency as it cannot be determined precisely by examination.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_OSC_REFERENCE`
`arg`	`s32*`

The table below lists the options used with this service. The range for valid values depends on the board being accessed.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`> 0`	This refers to the default, specified or measured frequency.	

Valid argument values are as follows.

Oscillator Type	**Fmin (Hz)**	**Fmax (Hz)**	**Information**
CY22393	20,000,000	20,000,000	The reference frequency cannot be changed.
IDC2053G	1,000,000	20,000,000	Oscillator programming is not implemented.
Fixed	1,000,000	20,000,000	Oscillator programming is unsupported.
Unknown	1,000,000	20,000,000	Oscillator programming is unsupported.

4.8.7. Register Specific IOCTL Services

These IOCTL services relate to access of the board's register.

4.8.7.1. SIO4_IOCTL_REG_MOD

This service performs a read-modify-write of an SIO4 register. This includes only the GSC firmware registers and the USC registers, when supported. The PCI and PLX Feature Set Registers are read-only. Refer to `sio4.h` for a complete list of the GSC firmware registers and to `sio4_usc.h` for a complete list of the USC registers.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_REG_MOD`
`arg`	`gsc_reg_t*`

Definition

```
typedef struct
{
    u32 reg;
    u32 value;
    u32 mask;
} gsc_reg_t;
```

Fields	**Description**
`reg`	This is set to the identifier for the register to access.
`value`	This contains the value for the register bits to modify.
`mask`	This specifies the set of bits to modify. If a bit here is set, then the respective register bit is modified. If a bit here is zero, then the respective register bit is unmodified.

4.8.7.2. SIO4_IOCTL_REG_READ

This service reads the value of an SIO4 register. This includes the PCI registers, the PLX Feature Set Registers, GSC firmware registers and the USC registers, when supported. Refer to `sio4.h` for a complete list of the GSC firmware registers and to `sio4_usc.h` for a complete list of the USC registers.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_REG_READ`
`arg`	`gsc_reg_t*`

Definition

```
typedef struct
{
    u32 reg;
    u32 value;
    u32 mask;
} gsc_reg_t;
```

Fields	**Description**
`reg`	This is set to the identifier for the register to access.

`value`	This is the value read from the specified register.
`mask`	This is ignored for read request.

4.8.7.3. SIO4_IOCTL_REG_READ_RAW

This service reads the value of an SIO4 firmware register without regard to the register layout or channel owner. Refer to `sio4.h` for a complete list of the GSC firmware registers.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_REG_READ_RAW`
`arg`	`gsc_reg_t*`

Definition

```
typedef struct
{
    u32 reg;
    u32 value;
    u32 mask;
} gsc_reg_t;
```

Fields	**Description**
`reg`	This is set to the zero-based index of the 32-bit register to access. Valid values are from zero to the end of the BAR2 address region.
`value`	This is the value read from the specified register.
`mask`	This is ignored for read request.

4.8.7.4. SIO4_IOCTL_REG_WRITE

This service writes a value to an SIO4 register. This includes only the GSC firmware registers and the USC registers, when supported. The PCI and PLX Feature Set Registers are read-only. Refer to `sio4.h` for a complete list of the GSC firmware registers and to `sio4_usc.h` for a complete list of the USC registers.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_REG_WRITE`
`arg`	`gsc_reg_t*`

Definition

```
typedef struct
{
    u32 reg;
    u32 value;
    u32 mask;
} gsc_reg_t;
```

Fields	**Description**
`reg`	This is set to the identifier for the register to access.
`value`	This is the value to write to the specified register.
`mask`	This is ignored for write request.

4.8.7.5. SIO4_IOCTL_REG_WRITE_RAW

This service writes to an SIO4 firmware register without regard to the register layout or channel owner. Refer to `sio4.h` for a complete list of the GSC firmware registers.

Usage

Argument	Description
`request`	`SIO4_IOCTL_REG_WRITE_RAW`
`arg`	`gsc_reg_t*`

Definition

```
typedef struct
{
    u32 reg;
    u32 value;
    u32 mask;
} gsc_reg_t;
```

Fields	Description
`reg`	This is set to the zero-based index of the 32-bit register to access. Valid values are from zero to the end of the BAR2 address region.
`value`	This is the value to write to the referenced register.
`mask`	This is ignored for write request.

4.8.8. Time Stamp Specific IOCTL Services

These IOCTL services relate to use of the Time Stamping feature.

4.8.8.1. SIO4_IOCTL_TIME_STAMP_COUNT

This service operates on the Time Stamp counter.

Usage

Argument	Description
`request`	`SIO4_IOCTL_TIME_STAMP_COUNT`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_TIME_STAMP_COUNT_CLEAR`	This refers to count being cleared.	
`SIO4_TIME_STAMP_COUNT_CONTINUE`	This refers to no change at all.	

4.8.8.2. SIO4_IOCTL_TIME_STAMP_ENABLE

This service enables or disables the Time Stamp feature.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_TIME_STAMP_COUNT`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_TIME_STAMP_ENABLE_NO`	This refers to the feature being disabled.	
`SIO4_TIME_STAMP_ENABLE_YES`	This refers to the feature being enabled.	

4.8.8.3. SIO4_IOCTL_TIME_STAMP_SRC

This service configures the source selection for the Time Stamp counter.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_TIME_STAMP_SRC`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_TIME_STAMP_SRC_EXT`	This refers to the external cable interface source.	
`SIO4_TIME_STAMP_SRC_INT`	This refers to the internal source.	

4.8.8.4. SIO4_IOCTL_TIME_STAMP_VAL

This service configures the current Time Stamp value.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_TIME_STAMP_VAL`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFFFFFF`	This is the valid range for this service.	

4.8.9. Wait Event Specific IOCTL Services

These IOCTL services relate to the Wait Event operation.

4.8.9.1. SIO4_IOCTL_WAIT_CANCEL

This service resumes all threads blocked via `SIO4_IOCTL_WAIT_EVENT` IOCTL calls (section 4.8.9.2, page 56), according to the provided criteria. When a blocked thread is waiting for any event specified in the structure, then the thread is resumed.

Usage

Argument	Description
`request`	`SIO4_IOCTL_WAIT_CANCEL`
`arg`	`gsc_wait_t*`

Definition

```
typedef struct
{
    u32  flags;
    u32  main;
    u32  gsc;
    u32  alt;
    u32  io;
    u32  timeout_ms;
    u32  count;
} gsc_wait_t;
```

Fields	Description
`flags`	This is unused by wait cancel operations.
`main`	This specifies the set of `GSC_WAIT_MAIN_*` events whose wait requests are to be cancelled. Refer to section 4.8.9.2.2 on page 57.
`gsc`	This specifies the set of `SIO4_WAIT_GSC_*` events whose wait requests are to be cancelled. Refer to section 4.8.9.2.3 on page 58.
`alt`	This specifies the set of `SIO4_WAIT_USC_*` events whose wait requests are to be cancelled. Refer to section 4.8.9.2.4 on page 58.
`io`	This specifies the set of `SIO4_WAIT_IO_*` events whose wait requests are to be cancelled. Refer to section 4.8.9.2.5 on page 59.
`timeout_ms`	This is unused by wait cancel operations.
`count`	Upon return this indicates the number of waits that were cancelled.

4.8.9.2. SIO4_IOCTL_WAIT_EVENT

This service blocks a thread until any one of a specified set of events occurs, or until a timeout lapses, whichever occurs first. The set of possible events to wait for are specified in the structure's `main`, `gsc`, `alt` and `io` fields. All field values must be valid and at least one event must be specified. If the thread is resumed because one of the referenced events has occurred, then the bit for the respective event is the only event bit that will be set. All other event bits and fields will be zero. (Multiple event bits will be set only if the events occur simultaneously.)

NOTE: The service waits only for the first of the specified events, not for all specified events.

NOTE: A wait timeout is reported via the `gsc_wait_t` structure's `flags` field having the `GSC_WAIT_FLAG_TIMEOUT` flag set, rather than via an `ETIMEDOUT` error.

Usage

Argument	Description
`request`	`SIO4_IOCTL_WAIT_EVENT`
`arg`	`gsc_wait_t*`

Definition

```
typedef struct
```

```
{
    u32  flags;
    u32  main;
    u32  gsc;
    u32  alt;
    u32  io;
    u32  timeout_ms;
    u32  count;
} gsc_wait_t;
```

Fields	Description
`flags`	This must initially be zero. Upon return this indicates the reason that the thread was resumed. Refer to section 4.8.9.2.1 on page 57.
`main`	This specifies the set of `GSC_WAIT_MAIN_*` events whose wait requests are to be cancelled. Refer to section 4.8.9.2.2 on page 57.
`gsc`	This specifies the set of `SIO4_WAIT_GSC_*` events whose wait requests are to be cancelled. Refer to section 4.8.9.2.3 on page 58.
`alt`	This specifies the set of `SIO4_WAIT_USC_*` events whose wait requests are to be cancelled. Refer to section 4.8.9.2.4 on page 58.
`io`	This specifies the set of `SIO4_WAIT_IO_*` events whose wait requests are to be cancelled. Refer to section 4.8.9.2.5 on page 59.
`timeout_ms`	This specified the maximum amount of time, in milliseconds, that the thread is to wait for any of the referenced events. A value of zero means do not timeout at all. If non-zero, then upon return the value will be the approximate amount of time actually waited.
`count`	This is unused by wait event operations and must be zero.

4.8.9.2.1. `gsc_wait_t.flags` Options

Upon return from a wait request the wait structure's `flags` field will indicate the reason that the thread was resumed. Only one of the below options will be set.

Values	Description
`GSC_WAIT_FLAG_CANCEL`	The wait request was cancelled.
`GSC_WAIT_FLAG_DONE`	One of the referenced events occurred.
`GSC_WAIT_FLAG_TIMEOUT`	The timeout period lapsed before a referenced event occurred.

4.8.9.2.2. `gsc_wait_t.main` Options

The wait structure's `main` field may specify any of the below primary interrupt options. These interrupt options are supported by the SIO4 and other General Standards products.

Values	Description
`GSC_WAIT_MAIN_DMA0`	This refers to the DMA Done interrupt on DMA engine number zero.
`GSC_WAIT_MAIN_DMA1`	This refers to the DMA Done interrupt on DMA engine number one.
`GSC_WAIT_MAIN_GSC`	This refers to any of the Interrupt Control/Status Register interrupts.
`GSC_WAIT_MAIN_OTHER`	This generally refers to an interrupt generated by another device sharing the same interrupt as the SIO4.
`GSC_WAIT_MAIN_PCI`	This refers to any interrupt generated by the SIO4.
`GSC_WAIT_MAIN_SPURIOUS`	This refers to board interrupts which should never be generated.
`GSC_WAIT_MAIN_UNKNOWN`	This refers to board interrupts whose source could not be identified.

4.8.9.2.3. `gsc_wait_t.gsc` Options

The wait structure's `gsc` field may specify any combination of the below interrupt options. These are the interrupt options referenced in the Interrupt Control Register. Applications are responsible for selecting the desired interrupt options. Refer to `SIO4_IOCTL_IRQ_GSC_ENABLE` (section 4.8.4.2, page 42).

Values	Description
`SIO4_WAIT_GSC_RX_ENV`	This refers to the interrupt that responds to the Rx Envelope signal. (`SYNC` model boards only)
`SIO4_WAIT_GSC_RX_FIFO_AF`	This refers to the interrupt that responds to the Rx FIFO Almost Full state.
`SIO4_WAIT_GSC_RX_FIFO_E`	This refers to the interrupt that response to the Rx FIFO Empty state.
`SIO4_WAIT_GSC_RX_FIFO_F`	This refers to the interrupt that response to the Rx FIFO Full state.
`SIO4_WAIT_GSC_RX_SPARE`	This refers to the interrupt that response to the Rx Spare signal. (`SYNC` model boards only)
`SIO4_WAIT_GSC_SYNC_BYTE`	This refers to the interrupt that responds to received characters matching the configured SYNC byte. (`Z16C30` model boards only)
`SIO4_WAIT_GSC_TX_FIFO_AE`	This refers to the interrupt that responds to the Tx FIFO Almost Empty state.
`SIO4_WAIT_GSC_TX_FIFO_E`	This refers to the interrupt that responds to the Tx FIFO Empty state.
`SIO4_WAIT_GSC_TX_FIFO_F`	This refers to the interrupt that responds to the Tx FIFO Full state.
`SIO4_WAIT_GSC_USC`	This refers to the interrupt that response to USC generated interrupts. (`Z16C30` model boards only)

4.8.9.2.4. `gsc_wait_t.alt` Options

The wait structure's `alt` field may specify any combination of the below interrupt options. These are the interrupt options generated by the USC of non-SYNC board only. Applications are responsible for selecting the desired interrupt options. Refer to `SIO4_IOCTL_IRQ_USC_ENABLE` (section 4.8.4.3, page 42).

Values	Description
`SIO4_WAIT_USC_IOP_CTS_FALL`	This refers to the interrupt generated when the USC CTS pin goes from a high to a low state.
`SIO4_WAIT_USC_IOP_CTS_RISE`	This refers to the interrupt generated when the USC CTS pin goes from a low to a high state.
`SIO4_WAIT_USC_IOP_DCD_FALL`	This refers to the interrupt generated when the USC DCD pin goes from a high to a low state.
`SIO4_WAIT_USC_IOP_DCD_RISE`	This refers to the interrupt generated when the USC DCD pin goes from a low to a high state.
`SIO4_WAIT_USC_IOP_RXC_FALL`	This refers to the interrupt generated when the USC Rx Clock pin goes from a high to a low state.
`SIO4_WAIT_USC_IOP_RXC_RISE`	This refers to the interrupt generated when the USC Rx Clock pin goes from a low to a high state.
`SIO4_WAIT_USC_IOP_RXREQ_FALL`	This refers to the interrupt generated when the USC Rx Request pin goes from a high to a low state. *
`SIO4_WAIT_USC_IOP_RXREQ_RISE`	This refers to the interrupt generated when the USC Rx Request pin goes from a low to a high state. *
`SIO4_WAIT_USC_IOP_TXC_FALL`	This refers to the interrupt generated when the USC Tx Clock pin goes from a high to a low state.
`SIO4_WAIT_USC_IOP_TXC_RISE`	This refers to the interrupt generated when the USC Tx Clock pin goes from a low to a high state.
`SIO4_WAIT_USC_IOP_TXREQ_FALL`	This refers to the interrupt generated when the USC Rx Request pin goes from a high to a low state. †
`SIO4_WAIT_USC_IOP_TXREQ_RISE`	This refers to the interrupt generated when the USC Rx Request pin goes from a low to a high state. †

`SIO4_WAIT_USC_MISC_BRG0_ZERO`	This refers to the interrupt generated when BRG0 counts down to zero.
`SIO4_WAIT_USC_MISC_BRG1_ZERO`	This refers to the interrupt generated when BRG1 counts down to zero.
`SIO4_WAIT_USC_MISC_DPLL_DESYNC`	This refers to the interrupt generated when the DPLL loses sync.
`SIO4_WAIT_USC_MISC_RCC_UNDERRUN`	This refers to the interrupt generated when the RCC value is read, but the RCC FIFO is empty.
`SIO4_WAIT_USC_RX_ABORT_PAR_ERROR`	This refers to the interrupt generated when the receiver detects an abort over the cable interface or when it detects a parity error in the received data.
`SIO4_WAIT_USC_RX_BOUND`	This refers to the interrupt generated when a character marked with the Rx Bound status is removed from the USC Rx FIFO.
`SIO4_WAIT_USC_RX_BREAK_ABORT`	This refers to the interrupt generated when the receiver detects Break (asynchronous modes) or Abort condition (HDLC).
`SIO4_WAIT_USC_RX_DATA`	This refers to the interrupt generated when a received character brings the Rx FIFO fill level above the specified threshold level.
`SIO4_WAIT_USC_RX_EXITED_HUNT`	This refers to the interrupt generated when the receiver exits Hunt mode.
`SIO4_WAIT_USC_RX_IDLE_RECEIVED`	This refers to the interrupt generated when the receiver detects 15 or 16 consecutive ones (HDLC).
`SIO4_WAIT_USC_RX_OVERRUN`	This refers to the interrupt generated when a character marked with the Rx Overrun status is removed from the USC Rx FIFO.
`SIO4_WAIT_USC_SPURIOUS`	This refers to the condition where a USC interrupt is generated, but its source cannot be determined.
`SIO4_WAIT_USC_TX_ABORT_SENT`	This refers to the interrupt generated when the transmitter has sent out an Abort character (HDLC).
`SIO4_WAIT_USC_TX_CRC_SENT`	This refers to the interrupt generated when the transmitter has sent the CRC that ends a frame or message.
`SIO4_WAIT_USC_TX_DATA`	This refers to the interrupt generated when the number of empty spaces in the USC Tx FIFO is above the specified threshold level.
`SIO4_WAIT_USC_TX_END_SENT`	This refers to the interrupt generated when the transmitter has sent out the closing flag or SYNC sequence that ends a frame or message.
`SIO4_WAIT_USC_TX_IDLE_SENT`	This refers to the interrupt generated when the transmitter has sent out the Idle condition or SYNC characters.
`SIO4_WAIT_USC_TX_PREAMBLE_SENT`	This refers to the interrupt generated when the transmitter has sent out the Preamble (asynchronous modes).
`SIO4_WAIT_USC_TX_UNDERRUN`	This refers the interrupt generated when the USC transmitter needs another character to send, but the USC Tx FIFO is empty.

* This pin is automatically configured by the driver to facilitate transfer of data from the USC receiver to the Rx FIFO.

† This pin is automatically configured by the driver to facilitate transfer of data from the Tx FIFO to the USC transmitter.

4.8.9.2.5. `gsc_wait_t.io` Options

The wait structure's `io` field may specify any of the below event options. These events are generated in response to application board data read requests.

Values	Description
`SIO4_WAIT_IO_RX_ABORT`	This refers to read requests which have been aborted.
`SIO4_WAIT_IO_RX_DONE`	This refers to read requests which have been satisfied.
`SIO4_WAIT_IO_RX_ERROR`	This refers to read requests which end due to an error.

`SIO4_WAIT_IO_RX_TIMEOUT`	This refers to read requests which end due to the timeout period lapse.
`SIO4_WAIT_IO_TX_ABORT`	This refers to write requests which have been aborted.
`SIO4_WAIT_IO_TX_DONE`	This refers to write requests which have been satisfied.
`SIO4_WAIT_IO_TX_ERROR`	This refers to write requests which end due to an error.
`SIO4_WAIT_IO_TX_TIMEOUT`	This refers to write requests which end due to the timeout period lapse.

4.8.9.3. SIO4_IOCTL_WAIT_STATUS

This service counts all threads blocked via the `SIO4_IOCTL_WAIT_EVENT` IOCTL service (section 4.8.9.2, page 56), according to the provided criteria. A match is made when a waiting thread's wait criteria matches any of the criteria specified in the structure passed to this service.

> **NOTE:** The driver itself makes use of the wait services for various internal operations. Driver initiated waits are ignored by application status requests.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_WAIT_STATUS`
`arg`	`gsc_wait_t*`

Definition

```
typedef struct
{
    u32  flags;
    u32  main;
    u32  gsc;
    u32  alt;
    u32  io;
    u32  timeout_ms;
    u32  count;
} gsc_wait_t;
```

Fields	**Description**
`flags`	This is unused by wait status operations.
`main`	This specifies the set of `GSC_WAIT_MAIN_*` events whose wait requests are to be counted. Refer to section 4.8.9.2.2 on page 57.
`gsc`	This specifies the set of `SIO4_WAIT_GSC_*` events whose wait requests are to be counted. Refer to section 4.8.9.2.3 on page 58.
`alt`	This specifies the set of `SIO4_WAIT_USC_*` events whose wait requests are to be counted. Refer to section 4.8.9.2.3 on page 58.
`io`	This specifies the set of `SIO4_WAIT_IO_*` events whose wait requests are to be counted. Refer to section 4.8.9.2.4 on page 58.
`timeout_ms`	This is unused by wait status operations.
`count`	Upon return this indicates the number of waits that met any of the specified criteria.

4.9. Zilog Model Specific IOCTL Services

These IOCTL services relate only to those SIO4 model boards which utilize the Zilog Z16C20 USC.

> **NOTE**: All services whose argument data type is `s32*` accept the value of `-1` being passed to the driver. If there is a setting associated with the service, then passing in the value `-1` is a request for the current setting. If the service is supported, then the current setting is returned. If the service is

not supported, then the value returned by the driver will be `-1`. If there isn't a setting associated with the service, then passing in the value `-1` is a request to determine if the service is supported. If the service is supported, then the value one is returned. If the service is not supported, then the value `-1` is returned.

4.9.1. Zilog Model Cable Specific IOCTL Services

These IOCTL services relate to the configuration of the cable interface.

4.9.1.1. SIO4_IOCTL_Z16_CBL_DCD_CFG

This service configures the cable DCD signal output source for USC based SIO4 boards.

NOTE: This service configures only the source for the DCD signal when configured as an output. The signal direction is configured via the `SIO4_IOCTL_USC_DCD_CFG` IOCTL service (section 4.10.19.3, page 107).

Usage

Argument	Description
`request`	`SIO4_IOCTL_Z16_CBL_DCD_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_Z16_CBL_DCD_CFG_OUT_0`	Drive the cable signal low.	
`SIO4_Z16_CBL_DCD_CFG_OUT_1`	Drive the cable signal high.	
`SIO4_Z16_CBL_DCD_CFG_OUT_RTS`	Drive the cable signal from the receiver's Request To Send source, which is the Rx FIFO Almost Full signal.	
`SIO4_Z16_CBL_DCD_CFG_OUT_USC_DCD`	Drive the cable signal from the USC DCD signal.	

4.9.1.2. SIO4_IOCTL_Z16_CBL_DTR_DSR_CFG

This service configures the cable DTR/DSR signal for USC based SIO4 boards. The DTR/DSR cable signal is provided for DTR/DSR flow control and is entirely under software control.

Usage

Argument	Description
`request`	`SIO4_IOCTL_Z16_CBL_DTR_DSR_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_Z16_CBL_DTR_DSR_CFG_OUT_0`	Drive the cable signal low.	
`SIO4_Z16_CBL_DTR_DSR_CFG_OUT_1`	Drive the cable signal high.	
`SIO4_Z16_CBL_DTR_DSR_CFG_IN`	Configure the signal as an input.	
`SIO4_Z16_CBL_DTR_DSR_CFG_TRI`	Tristate the signal.	

4.9.1.3. SIO4_IOCTL_Z16_CBL_RTS_CFG

This service configures the cable RTS signal for USC based SIO4 boards.

Usage

Argument	Description
`request`	`SIO4_IOCTL_Z16_CBL_RTS_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_Z16_CBL_RTS_CFG_OUT_0`	Drive the cable signal low.	
`SIO4_Z16_CBL_RTS_CFG_OUT_1`	Drive the cable signal high.	
`SIO4_Z16_CBL_RTS_CFG_OUT_RTS`	Drive the cable signal from the receiver’s Request To Send source, which is the Rx FIFO Almost Full signal.	
`SIO4_Z16_CBL_RTS_CFG_OUT_USC_CTS`	Drive the cable signal from the USC CTS signal.	

4.9.1.4. SIO4_IOCTL_Z16_CBL_TXAUXC_CFG

This service configures the cable Tx Auxiliary Clock signal for USC based SIO4 boards.

Usage

Argument	Description
`request`	`SIO4_IOCTL_Z16_CBL_TXAUXC_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_Z16_CBL_TXAUXC_CFG_OUT_0`	Drive the cable signal low.	
`SIO4_Z16_CBL_TXAUXC_CFG_OUT_1`	Drive the cable signal high.	
`SIO4_Z16_CBL_TXAUXC_CFG_OUT_OSC`	Drive the cable signal from the on-board oscillator.	
`SIO4_Z16_CBL_TXAUXC_CFG_TRI`	Tristate the signal.	

4.9.1.5. SIO4_IOCTL_Z16_CBL_TXC_CFG

This service configures the cable Tx Clock signal for USC based SIO4 boards.

Usage

Argument	Description
`request`	`SIO4_IOCTL_Z16_CBL_TXC_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_Z16_CBL_TXC_CFG_OUT_0`	Drive the cable signal low.	
`SIO4_Z16_CBL_TXC_CFG_OUT_1`	Drive the cable signal high.	
`SIO4_Z16_CBL_TXC_CFG_OUT_CBL_RXA`	Drive the cable signal from the Rx Auxiliary Clock signal on the cable interface.	
`SIO4_Z16_CBL_TXC_CFG_OUT_CBL_RXC`	Drive the cable signal from the Rx Clock signal on the cable interface.	
`SIO4_Z16_CBL_TXC_CFG_OUT_OSC`	Drive the cable signal from the on-board oscillator.	
`SIO4_Z16_CBL_TXC_CFG_OUT_OSC_INV`	Drive the cable signal from the on-board oscillator, but inverted.	
`SIO4_Z16_CBL_TXC_CFG_OUT_USC_RXC`	Drive the cable signal from the USC's Rx Clock pin.	
`SIO4_Z16_CBL_TXC_CFG_OUT_USC_TXC`	Drive the cable signal from the USC's Tx Clock pin.	

4.9.1.6. SIO4_IOCTL_Z16_CBL_TXD_CFG

This service configures the cable Tx Data signal for USC based SIO4 boards.

Usage

Argument	Description
`request`	`SIO4_IOCTL_Z16_CBL_TXD_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_Z16_CBL_TXD_CFG_OUT_0`	Drive the cable signal low.	
`SIO4_Z16_CBL_TXD_CFG_OUT_1`	Drive the cable signal high.	
`SIO4_Z16_CBL_TXD_CFG_OUT_USC_TXD`	Drive the cable signal from the USC's Tx Data signal.	

4.9.2. Zilog Model Legacy Cable Specific IOCTL Services

These IOCTL services relate to the configuration of the legacy cable interface.

> **NOTE**: These services are available only when the legacy cable configuration is supported by firmware. Please also read Cable Configuration Modes (section 8.6, page 137).

4.9.2.1. SIO4_IOCTL_Z16_LEG_RXC

This service configures the routing of the cable Rx Clock signal when using legacy cable configuration mode. Please also read Cable Configuration Modes (section 8.6, page 137).

Usage

Argument	Description
`request`	`SIO4_IOCTL_Z16_LEG_RXC`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The feature is unsupported or inaccessible.
`SIO4_Z16_LEG_RXC_DISABLE`	Disable the cable signal.	
`SIO4_Z16_LEG_RXC_LOWER`	Connect to the signal at the lower cable portion.	
`SIO4_Z16_LEG_RXC_UPPER`	Connect to the signal at the upper cable portion.	

4.9.2.2. SIO4_IOCTL_Z16_LEG_RXD_DCD_CFG

This service configures the cable Rx Data and DCD signals for USC based SIO4 boards supporting the legacy cable interface. Please also read Cable Configuration Modes (section 8.6, page 137).

Usage

Argument	Description
`request`	`SIO4_IOCTL_Z16_LEG_RXD_DCD_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The feature is unsupported or inaccessible.
`SIO4_Z16_LEG_RXD_DCD_CFG_LOW`	This configures the signals for the lower connector pins.	
`SIO4_Z16_LEG_RXD_DCD_CFG_TRI`	This tri-states the cable signals.	
`SIO4_Z16_LEG_RXD_DCD_CFG_UP`	This configures the signals for the upper connector pins.	

4.9.2.3. SIO4_IOCTL_Z16_LEG_TXC

This service configures the routing of the cable Tx Clock signal when using legacy cable configuration mode. Please also read Cable Configuration Modes (section 8.6, page 137).

Usage

Argument	Description
`request`	`SIO4_IOCTL_CBL_LEG_TXC`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The feature is unsupported or inaccessible.
`SIO4_Z16_LEG_TXC_DISABLE`	Disable the cable signal.	
`SIO4_Z16_LEG_TXC_BOTH`	Connect the signal to both the upper and lower cable portions.	
`SIO4_Z16_LEG_TXC_LOWER`	Connect the signal to the lower cable portion.	
`SIO4_Z16_LEG_TXC_UPPER`	Connect the signal to the upper cable portion.	

4.9.2.4. SIO4_IOCTL_Z16_LEG_TXD_CTS_CFG

This service configures the cable Tx Data and CTS signals for USC based SIO4 boards supporting the legacy cable interface. Please also read Cable Configuration Modes (section 8.6, page 137).

Usage

Argument	Description
`request`	`SIO4_IOCTL_Z16_LEG_TXD_CTS_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The feature is unsupported or inaccessible.
`SIO4_Z16_LEG_TXD_CTS_CFG_LOW`	This configures the signals for the lower connector pins.	
`SIO4_Z16_LEG_TXD_CTS_CFG_TRI`	This tri-states the cable signals.	
`SIO4_Z16_LEG_TXD_CTS_CFG_UP`	This configures the signals for the upper connector pins.	
`SIO4_Z16_LEG_TXD_CTS_CFG_UP_LOW`	This configures the signals for the upper and lower connector pins.	

4.9.3. Zilog Model Miscellaneous IOCTL Services

4.9.3.1. SIO4_IOCTL_Z16_RX_STS_WRD_ENABLE

This service configures the receiver’s option of including the USC’s Rx Status Register in the main FIFO with each data value extracted from the USC’s Rx FIFO.

Usage

Argument	Description
`request`	`SIO4_IOCTL_Z16_RX_STS_WRD_ENABLE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_Z16_RX_STS_WRD_ENABLE_NO`	Do not enable the feature.	
`SIO4_Z16_RX_STS_WRD_ENABLE_YES`	Enable the feature.	

4.9.3.2. SIO4_IOCTL_Z16_SYNC_BYTE

This service set the Sync Byte Value used for Sync Bytes Detection for USC based SIO4 boards.

Usage

Argument	Description
`request`	`SIO4_IOCTL_Z16_SYNC_BYTE`
`arg`	`s32*`

The table below lists the options used with this service. Valid argument values are from zero to `0xFF`.

Macros	Description
`-1`	Requests current setting.

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFF`	This is the valid range for this service.	

4.10. USC Specific Zilog based IOCTL Services

These IOCTL services are specific to the USC on Zilog based SIO4 boards.

> **NOTE**: All services whose argument data type is `s32*` accept the value of `-1` being passed to the driver. If there is a setting associated with the service, then passing in the value `-1` is a request for the current setting. If the service is supported, then the current setting is returned. If the service is not supported, then the value returned by the driver will be `-1`. If there isn't a setting associated with the service, then passing in the value `-1` is a request to determine if the service is supported. If the service is supported, then the value one is returned. If the service is not supported, then the value `-1` is returned.

4.10.1. USC 802.3 Protocol Specific IOCTL Services

These services are specific to the USC's 802.3 protocol support.

4.10.1.1. SIO4_IOCTL_USC_8023_RX_ADRS_SRCH

This service configures the Address Search feature for the receiver when using the 802.3 serial protocol.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_8023_RX_ADRS_SRCH`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_8023_RX_ADRS_SRCH_NO`	This disables use of the feature.	
`SIO4_USC_8023_RX_ADRS_SRCH_YES`	This enables use of the feature.	

4.10.1.2. SIO4_IOCTL_USC_8023_TX_UNDERRUN

This service configures the Underrun reaction of the transmitter when using the 802.3 serial protocol.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_8023_TX_UNDERRUN`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_8023_TX_UNDERRUN_CRC`	Send out the current CRC value.	
`SIO4_USC_8023_TX_UNDERRUN_NONE`	Take no additional action.	

4.10.2. USC Asynchronous Protocol Specific IOCTL Services

These services are specific to the USC's Asynchronous protocol support.

4.10.2.1. SIO4_IOCTL_USC_ASYNC_RX_CLK_RATE

This service configures the Clock Rate (the oversampling rate) for the receiver when using the Asynchronous protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_ASYNC_RX_CLK_RATE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_ASYNC_CLK_RATE_16X`	Divide the source clock by 16.	
`SIO4_USC_ASYNC_CLK_RATE_32X`	Divide the source clock by 32.	
`SIO4_USC_ASYNC_CLK_RATE_64X`	Divide the source clock by 64.	

4.10.2.2. SIO4_IOCTL_USC_ASYNC_TX_CLK_RATE

This service configures the Clock Rate for the transmitter when using the Asynchronous protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_ASYNC_TX_CLK_RATE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_ASYNC_CLK_RATE_16X`	Divide the source clock by 16.	
`SIO4_USC_ASYNC_CLK_RATE_32X`	Divide the source clock by 32.	
`SIO4_USC_ASYNC_CLK_RATE_64X`	Divide the source clock by 64.	

4.10.2.3. SIO4_IOCTL_USC_ASYNC_TX_STOP_BIT

This service configures the number of Stop Bits for the transmitter when using the Asynchronous protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_ASYNC_TX_STOP_BIT`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
-1	Requests current setting.	The service is unsupported.
SIO4_USC_ASYNC_TX_STOP_BIT_0__9_16	Set the Stop Bit period to 9/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_0_10_16	Set the Stop Bit period to 10/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_0_11_16	Set the Stop Bit period to 11/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_0_12_16	Set the Stop Bit period to 12/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_0_13_16	Set the Stop Bit period to 13/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_0_14_16	Set the Stop Bit period to 14/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_0_15_16	Set the Stop Bit period to 15/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_1	Set the Stop Bit period to one bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_1__1_16	Set the Stop Bit period to 1 + 1/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_1__2_16	Set the Stop Bit period to 1 + 2/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_1__3_16	Set the Stop Bit period to 1 + 3/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_1__4_16	Set the Stop Bit period to 1 + 4/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_1__5_16	Set the Stop Bit period to 1 + 5/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_1__6_16	Set the Stop Bit period to 1 + 6/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_1__7_16	Set the Stop Bit period to 1 + 7/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_1__8_16	Set the Stop Bit period to 1 + 8/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_1__9_16	Set the Stop Bit period to 1 + 9/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_1_10_16	Set the Stop Bit period to 1 + 10/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_1_11_16	Set the Stop Bit period to 1 + 11/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_1_12_16	Set the Stop Bit period to 1 + 12/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_1_13_16	Set the Stop Bit period to 1 + 13/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_1_14_16	Set the Stop Bit period to 1 + 14/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_1_15_16	Set the Stop Bit period to 1 + 15/16th of a bit period.	
SIO4_USC_ASYNC_TX_STOP_BIT_2	Set the Stop Bit period to two-bit periods.	

4.10.3. USC Asynchronous with Code Violations Protocol Specific IOCTL Services

These services are specific to the USC's Asynchronous with Code Violations protocol support.

4.10.3.1. SIO4_IOCTL_USC_ACV_RX_EXT_W

This service configures the Extended Word feature for the receiver when using the Asynchronous with Code Violation protocol.

Usage

Argument	Description
request	SIO4_IOCTL_USC_ACV_RX_EXT_W
arg	s32*

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
-1	Requests current setting.	The service is unsupported.
SIO4_USC_ACV_EXT_W_NO	Do not use Extended Words.	
SIO4_USC_ACV_EXT_W_YES	Use Extended Words.	

4.10.3.2. SIO4_IOCTL_USC_ACV_TX_CV_POL

This service configures the Code Violation Polarity for the transmitter when using the Asynchronous with Code Violation protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_ACV_TX_CV_POL`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_ACV_TX_CV_POL_0`	This selects the zero or low polarity.	
`SIO4_USC_ACV_TX_CV_POL_1`	This selects the one or high polarity.	

4.10.3.3. SIO4_IOCTL_USC_ACV_TX_EXT_W

This service configures the Extended Word feature for the transmitter when using the Asynchronous with Code Violation protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_ACV_TX_EXT_W`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_ACV_EXT_W_NO`	Do not use Extended Words.	
`SIO4_USC_ACV_EXT_W_YES`	Use Extended Words.	

4.10.3.4. SIO4_IOCTL_USC_ACV_TX_STOP_BIT

This service configures the number of Stop Bits for the transmitter when using the Asynchronous with Code Violation protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_ACV_TX_STOP_BIT`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_ACV_TX_STOP_BIT_1`	Use one Stop Bit.	
`SIO4_USC_ACV_TX_STOP_BIT_2`	Use two Stop Bits.	
`SIO4_USC_ACV_TX_STOP_BIT_NONE`	Use no Stop Bits.	

4.10.4. USC Binary Synchronous Communications (BSC) Protocol Specific IOCTL Services

These services are specific to the USC’s Binary Synchronous Communications (BiSync, or BSC) protocol support.

4.10.4.1. SIO4_IOCTL_USC_BSC_RX_SHORT

This service configures the Short Sync Character selection for the receiver when using the Bisynchronous Serial Communications protocol.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_BSC_RX_SHORT`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_BSC_SHORT_NO`	Do not use a Short Sync Character.	
`SIO4_USC_BSC_SHORT_YES`	Use a Short Sync Character.	

4.10.4.2. SIO4_IOCTL_USC_BSC_RX_STRIP

This service configures the receiver to strip detected sync characters from the input stream when using the Bisynchronous Serial Communications protocol.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_BSC_RX_STRIP`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_BSC_RX_STRIP_NO`	Do not Strip the Sync Character(s) from the input stream.	
`SIO4_USC_BSC_RX_STRIP_YES`	Strip the Sync Character(s) from the input stream.	

4.10.4.3. SIO4_IOCTL_USC_BSC_RX_SYN0

This service configures the SYN0 Sync Character value for the receiver when using the Bisynchronous Serial Communications protocol.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_BSC_RX_SYN0`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFF`	This is the valid range for this service.	

4.10.4.4. SIO4_IOCTL_USC_BSC_RX_SYN1

This service configures the SYN1 Sync Character value for the receiver when using the Bisynchronous Serial Communications protocol.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_BSC_RX_SYN1`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFF`	This is the valid range for this service.	

4.10.4.5. SIO4_IOCTL_USC_BSC_TX_PREAMBLE

This service configures the Preamble output selection for the transmitter when using the Bisynchronous Serial Communications protocol.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_BSC_TX_PREAMBLE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_BSC_TX_PREAMBLE_NO`	This refers to sending no Preamble.	
`SIO4_USC_BSC_TX_PREAMBLE_YES`	This refers to sending a Preamble.	

4.10.4.6. SIO4_IOCTL_USC_BSC_TX_SHORT

This service configures the Short Sync Character selection for the transmitter when using the Bisynchronous Serial Communications protocol.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_BSC_TX_SHORT`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_BSC_SHORT_NO`	Do not use a Short Sync Character.	
`SIO4_USC_BSC_SHORT_YES`	Use a Short Sync Character.	

4.10.4.7. SIO4_IOCTL_USC_BSC_TX_SYN0

This service configures the SYN0 Sync Character value for the transmitter when using the Bisynchronous Serial Communications protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_BSC_TX_SYN0`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFF`	This is the valid range for this service.	

4.10.4.8. SIO4_IOCTL_USC_BSC_TX_SYN1

This service configures the SYN1 Sync Character value for the transmitter when using the Bisynchronous Serial Communications protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_BSC_TX_SYN1`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFF`	This is the valid range for this service.	

4.10.4.9. SIO4_IOCTL_USC_BSC_TX_UNDERRUN

This service configures the Underrun reaction selection for the transmitter when using the Bisynchronous Serial Communications protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_BSC_TX_UNDERRUN`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_BSC_TX_UNDERRUN_CRC_S01`	Send the CRC, then the SYN0 and SYN1 characters.	
`SIO4_USC_BSC_TX_UNDERRUN_CRC_S1`	Send the CRC, then the SYN1 character.	
`SIO4_USC_BSC_TX_UNDERRUN_S01`	Send the SYN0 and SYN1 characters.	
`SIO4_USC_BSC_TX_UNDERRUN_S1`	Send the SYN1 character.	

4.10.5. USC HDLC Protocol Specific IOCTL Services

These services are specific to the USC's HDLC protocol support. HDLC stands for High-Level Data Link Control. For detailed information on the HDLC protocol refer to ISO/IEC 13239.

4.10.5.1. SIO4_IOCTL_HDLC_TX_FRAME_INIT

This service enables and disables driver processing for transmission of HDLC frames. When being enabled, the service performs initialization in preparation for subsequent frame transmission.

> **NOTE**: This service was implemented specifically for support of the HDLC Protocol Library and may not be suitable for any other use.
>
> **NOTE**: This service is serialized with write requests and the other …`_HDLC_TX_FRAME_`… services to prevent interference with HDLC frame transmission operations.
>
> **NOTE**: This service clears all data from the Tx FIFO and from the USC transmitter. It also clears all conditions reportable via the …`HDLC_TX_FRAME_STATUS` service (section 4.10.5.3, page 74).

Usage

Argument	Description
`request`	`SIO4_IOCTL_HDLC_TX_FRAME_INIT`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_HDLC_TX_FRAME_INIT_DISABLE`	Disable HDLC Tx frame processing. *	
`SIO4_HDLC_TX_FRAME_INIT_ENABLE`	Enable HDLC Tx frame processing. *	

* The *disable* option is returned if HDLC Tx frame processing is not enabled.

4.10.5.2. SIO4_IOCTL_HDLC_TX_FRAME_SETUP

This service configures the write service for transmission of an HDLC frame. While HDLC Tx frame processing is enabled this service must be called prior to every write request.

> **NOTE**: This service was implemented specifically for support of the HDLC Protocol Library and may not be suitable for any other use.
>
> **NOTE**: This service is serialized with write requests and the other …`_HDLC_TX_FRAME_`… services to prevent interference with HDLC frame transmission operations.

Usage

Argument	Description
`request`	`SIO4_IOCTL_HDLC_TX_FRAME_SETUP`
`arg`	`s32*`

The table below lists the options used with this service. The flag is optional.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported or HDLC Tx frame processing is disabled.
`0`	This indicates that the data need not be sent as a single frame.	
`SIO4_HDLC_TX_FRAME_SETUP_SINGLE`	This directs the write service to attempt sending the requested data as a single frame. *	

* This may involve polling for completion of any prior frames. The interval between checks is one system timer tick.

4.10.5.3. SIO4_IOCTL_HDLC_TX_FRAME_STATUS

This service reports the current HDLC frame transmission status.

NOTE: This service was implemented specifically for support of the HDLC Protocol Library and may not be suitable for any other use.

NOTE: This service is serialized with write requests and the other `…_HDLC_TX_FRAME_…` services to prevent interference with HDLC frame transmission operations.

Usage

Argument	Description
`request`	`SIO4_IOCTL_HDLC_TX_FRAME_STATUS`
`arg`	`s32*`

The table below lists the options returned by this service. The flags may appear in any combination.

NOTE: The driver polls for the specified conditions and waits one system timer tick between checks.

Values	Returned by Driver
`-1`	The service is unsupported or HDLC Tx frame processing is disabled.
`SIO4_HDLC_TX_FRAME_STATUS_ABORT`	The Abort Sent status was asserted, meaning an Abort sequence was sent out the cable interface. The status is cleared upon detection.
`SIO4_HDLC_TX_FRAME_STATUS_CRC`	The CRC Sent status was asserted, meaning a frame CRC was sent out the cable interface. The status is cleared upon detection.
`SIO4_HDLC_TX_FRAME_STATUS_EOF`	The EOF Sent status was asserted, meaning a frame's closing Flag was sent out the cable interface. The status is cleared upon detection.
`SIO4_HDLC_TX_FRAME_STATUS_M_FIFO`	The SIO4's main Tx FIFO is empty.
`SIO4_HDLC_TX_FRAME_STATUS_M_OVER`	There was a main Tx FIFO overrun. The status is reported, but not cleared. This is an error condition and requires recovery.
`SIO4_HDLC_TX_FRAME_STATUS_PREAMB`	The Preamble Sent status was asserted, meaning a frame preamble was sent out the cable interface. The status is cleared upon detection.
`SIO4_HDLC_TX_FRAME_STATUS_SPLIT`	A write request was split into two or more frames. The status is cleared upon detection.
`SIO4_HDLC_TX_FRAME_STATUS_U_FIFO`	The USC's internal Tx FIFO is empty.

4.10.5.4. SIO4_IOCTL_HDLC_TX_FRAME_WAIT

This service has two functions, based on the presence or absence of the SETUP flag described below. If this flag is present, then the argument value is recorded for use by subsequent write requests as part of Tx HDLC frame processing. In this case, it specifies the conditions that the write service is to wait for before returning. The first of the specified conditions detected terminates the wait. If the write I/O timeout expires before any of the specified conditions appear, then the wait is silently terminated. If the SETUP flag is absent, then the service directs the driver to wait for the appearance of any of the specified status conditions. The driver will wait for up to the current write I/O timeout period. The argument value returned is the flag for the first event detected, or zero if no condition was specified or if the timeout lapsed.

> **NOTE**: This service was implemented specifically for support of the HDLC Protocol Library and may not be suitable for any other use.

> **NOTE**: The waiting functionality of this service is NOT serialized with write requests or the other …`_HDLC_TX_FRAME_`… services. The setting functionality of this service is serialized with write requests or the other …`_HDLC_TX_FRAME_`… services.

Usage

Argument	Description
`request`	`SIO4_IOCTL_HDLC_TX_FRAME_WAIT`
`arg`	`s32*`

The table below lists the options used with this service. The flags can be used in any combination, but typically only one should be used.

> **NOTE**: The driver polls for the specified conditions and waits one system timer tick between checks.

> **NOTE**: Refer to the …`HDLC_TX_FRAME_STATUS` service (section 4.10.5.3, page 74) for additional information on the flags.

Values	Passed to Driver	Returned by Driver
`-1`	Report current recorded setting.	Service is unsupported or HDLC Tx frame processing is disabled.
`-1 ^ SIO4_HDLC_TX_FRAME_STATUS_SETUP`	Report most recent write wait results.	Never returned.
`SIO4_HDLC_TX_FRAME_STATUS_ABORT`	Wait for the Abort Sent status to be asserted.	
`SIO4_HDLC_TX_FRAME_STATUS_CRC`	Wait for the CRC Sent status to be asserted.	
`SIO4_HDLC_TX_FRAME_STATUS_EOF`	Wait for the EOF Sent status to be asserted.	
`SIO4_HDLC_TX_FRAME_STATUS_M_FIFO`	Wait for the SIO4's main Tx FIFO to become empty.	
`SIO4_HDLC_TX_FRAME_STATUS_M_OVER`	Wait for a main Tx FIFO overrun.	
`SIO4_HDLC_TX_FRAME_STATUS_PREAMB`	Wait for the Preamble Sent status to be asserted.	
`SIO4_HDLC_TX_FRAME_STATUS_SETUP`	Control operation of the service as described above.	
`SIO4_HDLC_TX_FRAME_STATUS_SPLIT`	Wait for the USC transmitter to report that a frame has been subdivided.	
`SIO4_HDLC_TX_FRAME_STATUS_U_FIFO`	Wait for the USC's internal Tx FIFO to become empty.	

> **NOTE**: The …`SETUP` flag is not recorded and is excluded from the mask macro.

4.10.5.5. SIO4_IOCTL_USC_HDLC_RX_ADRS_CTRL

This service configures the receiver's Address and Control field detection method when using the HDLC protocol.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_HDLC_RX_ADRS_CTRL`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_HDLC_RX_ADRS_CTRL_A1_C1`	This specifies a one-byte Address field and a one-byte Control field.	
`SIO4_USC_HDLC_RX_ADRS_CTRL_A1_C2`	This specifies a one-byte Address field and a two-byte Control field.	
`SIO4_USC_HDLC_RX_ADRS_CTRL_A1_C3`	This specifies a one-byte Address field and a three-byte Control field.	
`SIO4_USC_HDLC_RX_ADRS_CTRL_AE_1_CE`	This specifies an Extended Address field and an Extended Control field, with a one-byte field between them.	
`SIO4_USC_HDLC_RX_ADRS_CTRL_AE_2_CE`	This specifies an Extended Address field and an Extended Control field, with a two-byte field between them.	
`SIO4_USC_HDLC_RX_ADRS_CTRL_AE_C2`	This specifies an Extended Address field and a two-byte Control field.	
`SIO4_USC_HDLC_RX_ADRS_CTRL_AE_C3`	This specifies an Extended Address field and a three-byte Control field.	
`SIO4_USC_HDLC_RX_ADRS_CTRL_OFF`	This disables Address and Control field detection.	

4.10.5.6. SIO4_IOCTL_USC_HDLC_TX_L_CHR_LEN

This service configures the transmitter's Last Character Length for a frame when using the HDLC protocol.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_HDLC_TX_L_CHR_LEN`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_HDLC_TX_L_CHR_LEN_1`	This sets the last character length to 1-bit.	
`SIO4_USC_HDLC_TX_L_CHR_LEN_2`	This sets the last character length to 2-bits.	
`SIO4_USC_HDLC_TX_L_CHR_LEN_3`	This sets the last character length to 3-bits.	
`SIO4_USC_HDLC_TX_L_CHR_LEN_4`	This sets the last character length to 4-bits.	
`SIO4_USC_HDLC_TX_L_CHR_LEN_5`	This sets the last character length to 5-bits.	
`SIO4_USC_HDLC_TX_L_CHR_LEN_6`	This sets the last character length to 6-bits.	

`SIO4_USC_HDLC_TX_L_CHR_LEN_7`	This sets the last character length to 7-bits.
`SIO4_USC_HDLC_TX_L_CHR_LEN_8`	This sets the last character length to 8-bits.

4.10.5.7. SIO4_IOCTL_USC_HDLC_TX_PREAMBLE

This service configures the transmitter to include, or not, a preamble as part of a frame when using the HDLC protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_HDLC_TX_PREAMBLE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_HDLC_TX_PREAMBLE_NO`	Do not include a preamble.	
`SIO4_USC_HDLC_TX_PREAMBLE_YES`	Include a preamble.	

4.10.5.8. SIO4_IOCTL_USC_HDLC_TX_SHARE_0

This service configures the transmitter to share the zero between consecutive flag sequences when using the HDLC protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_HDLC_TX_SHARE_0`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_HDLC_TX_SHARE_0_NO`	Do not share the zero. (Send two zeroes.)	
`SIO4_USC_HDLC_TX_SHARE_0_YES`	Share the zero. (Only send one zero.)	

4.10.5.9. SIO4_IOCTL_USC_HDLC_TX_UNDERRUN

This service configures the transmitter’s reaction to a Tx FIFO Underrun condition when using the HDLC protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_HDLC_TX_UNDERRUN`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_HDLC_TX_UNDERRUN_ABORT`	Send an Abort sequence.	
`SIO4_USC_HDLC_TX_UNDERRUN_CRC_F`	Send a CRC then a Flag sequence.	
`SIO4_USC_HDLC_TX_UNDERRUN_EXT_A`	Send an Extended Abort sequence.	
`SIO4_USC_HDLC_TX_UNDERRUN_FLAG`	Send a Flag sequence.	

4.10.6. USC HDLC Loop Protocol Specific IOCTL Services

These services are specific to the USC's HDLC Loop protocol support.

4.10.6.1. SIO4_IOCTL_USC_HDLCL_TX_ACTIVE

This service configures the transmitter to insert its own into the data stream when using the HDLC protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_HDLCL_TX_ACTIVE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_HDLCL_TX_ACTIVE_NONE`	Do not insert the transmitter's own data into the stream.	
`SIO4_USC_HDLCL_TX_ACTIVE_POLL`	Request that the transmitter insert its own data into the stream.	

4.10.6.2. SIO4_IOCTL_USC_HDLCL_TX_SHARE_0

This service configures the transmitter to share the zero between consecutive flag sequences when using the HDLC Loop protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_HDLCL_TX_SHARE_0`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_HDLCL_TX_SHARE_0_NO`	Do not share the zero. (Send two zeroes.)	
`SIO4_USC_HDLCL_TX_SHARE_0_YES`	Share the zero. (Only send one zero.)	

4.10.6.3. SIO4_IOCTL_USC_HDLCL_TX_UNDERRUN

This service configures the transmitter's reaction to a Tx FIFO Underrun condition when using the HDLC Loop protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_HDLCL_TX_UNDERRUN`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_HDLCL_TX_UNDERRUN_ABORT`	Send an Abort sequence.	
`SIO4_USC_HDLCL_TX_UNDERRUN_CRC_F`	Send a CRC then a Flag sequence.	
`SIO4_USC_HDLCL_TX_UNDERRUN_E_ABT`	Send an Extended Abort sequence.	
`SIO4_USC_HDLCL_TX_UNDERRUN_FLAG`	Send a Flag sequence.	

4.10.7. USC Isochronous Protocol Specific IOCTL Services

These services are specific to the USC’s Isochronous protocol support.

4.10.7.1. SIO4_IOCTL_USC_ISOC_TX_STOP_BIT

This service configures the transmitter’s number of Stop Bits when using the Isochronous protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_ISOC_TX_STOP_BIT`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_ISOC_TX_STOP_BIT_1`	Use one Stop Bit.	
`SIO4_USC_ISOC_TX_STOP_BIT_2`	Use two Stop Bits.	

4.10.8. USC Monosync Protocol Specific IOCTL Services

These services are specific to the USC’s Monosync protocol support.

4.10.8.1. SIO4_IOCTL_USC_MONO_RX_SHORT

This service configures the Short Sync Character selection for the receiver when using the Monosync protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_MONO_RX_SHORT`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_MONO_SHORT_NO`	Do not use a Short Sync Character. Use 8-bits.	
`SIO4_USC_MONO_SHORT_YES`	Use a Short Sync Character. Use less than 8-bits.	

4.10.8.2. SIO4_IOCTL_USC_MONO_RX_STRIP

This service configures the receiver to strip detected sync characters from the input stream when using the Monosync protocol.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_MONO_RX_STRIP`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_MONO_RX_STRIP_NO`	Do not Strip the Sync Character from the input stream.	
`SIO4_USC_MONO_RX_STRIP_YES`	Strip the Sync Character from the input stream.	

4.10.8.3. SIO4_IOCTL_USC_MONO_RX_SYNC

This service configures the receiver Sync character when using the Monosync protocol.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_MONO_RX_SYNC`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFF`	This is the valid range for this service.	

4.10.8.4. SIO4_IOCTL_USC_MONO_TX_CRC_UNDER

This service configures the transmitter’s reaction to an underrun when operating in Monosync mode.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_MONO_TX_CRC_UNDER`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_MONO_TX_CRC_UNDER_NO`	Do not output a CRC when an underrun occurs.	
`SIO4_USC_MONO_TX_CRC_UNDER_YES`	Output a CRC when an underrun occurs.	

4.10.8.5. SIO4_IOCTL_USC_MONO_TX_PREAMBLE

This service configures the transmitter's generation of a preamble sequence when operating in Monosync mode.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_MONO_TX_PREAMBLE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_MONO_TX_PREAMBLE_NO`	Do not produce a Preamble sequence.	
`SIO4_USC_MONO_TX_PREAMBLE_YES`	Produce a Preamble sequence.	

4.10.8.6. SIO4_IOCTL_USC_MONO_TX_SHORT

This service configures the Short Sync Character selection for the transmitter when using the Monosync protocol.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_MONO_TX_SHORT`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_MONO_SHORT_NO`	Do not use a Short Sync Character. Use 8-bits.	
`SIO4_USC_MONO_SHORT_YES`	Use a Short Sync Character. Use less than 8-bits.	

4.10.8.7. SIO4_IOCTL_USC_MONO_TX_SYNC

This service configures the transmitter Sync character when using the Monosync protocol.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_MONO_TX_SYNC`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFF`	This is the valid range for this service.	

4.10.9. USC Nine Bit Interprocessor Protocol (NBIP) Specific IOCTL Services

These services are specific to the USC's NBIP protocol support. NBIP stands for Nine-Bit Interprocessor Protocol.

4.10.9.1. SIO4_IOCTL_USC_NBIP_RX_CLK_RATE

This service configures the Clock Rate (the oversampling rate) for the receiver when using the Nin-Bit Interprocessor Protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_NBIP_RX_CLK_RATE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_NBIP_CLK_RATE_16X`	Divide the source clock by 16.	
`SIO4_USC_NBIP_CLK_RATE_32X`	Divide the source clock by 32.	
`SIO4_USC_NBIP_CLK_RATE_64X`	Divide the source clock by 64.	

4.10.9.2. SIO4_IOCTL_USC_NBIP_RX_PARITY

This service configures the receiver's use of parity when using the Nin-Bit Interprocessor Protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_NBIP_RX_PARITY`
`arg`	`s32*`

The table below lists the options used with this service.

Macros	Description
`-1`	Requests the current setting.
`SIO4_USC_NBIP_PARITY_NO`	Do not use Parity.
`SIO4_USC_NBIP_PARITY_YES`	Use Parity.

4.10.9.3. SIO4_IOCTL_USC_NBIP_TX_ADRS_BIT

This service configures the setting of the address bit for the transmitter when using the Nin-Bit Interprocessor Protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_NBIP_TX_ADRS_BIT`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_NBIP_TX_ADRS_BIT_NO`	Do not set the address bit.	
`SIO4_USC_NBIP_TX_ADRS_BIT_YES`	Set the address bit.	

4.10.9.4. SIO4_IOCTL_USC_NBIP_TX_CLK_RATE

This service configures the Clock Rate for the transmitter when using the Nin-Bit Interprocessor Protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_NBIP_TX_CLK_RATE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_NBIP_CLK_RATE_16X`	Divide the source clock by 16.	
`SIO4_USC_NBIP_CLK_RATE_32X`	Divide the source clock by 32.	
`SIO4_USC_NBIP_CLK_RATE_64X`	Divide the source clock by 64.	

4.10.9.5. SIO4_IOCTL_USC_NBIP_TX_PARITY

This service configures the transmitter’s use of parity when using the Nin-Bit Interprocessor Protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_NBIP_TX_PARITY`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_NBIP_PARITY_NO`	Do not use Parity.	
`SIO4_USC_NBIP_PARITY_YES`	Use Parity.	

4.10.10. USC Slaved Monosync Protocol Specific IOCTL Services

These services are specific to the USC’s Slaved Monosync protocol support.

4.10.10.1. SIO4_IOCTL_USC_SMONO_RX_SYNC

This service configures the receiver Sync character when using the Slaved Monosync protocol.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_SMONO_RX_SYNC`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFF`	This is the valid range for this service.	

4.10.10.2. SIO4_IOCTL_USC_SMONO_TX_ACTIVE

This service configures the USC transmitter activation when using the Slaved Monosync protocol.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_SMONO_TX_ACTIVE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_SMONO_TX_ACTIVE_ON_SYNC`	This refers to activation when the transmitter is synchronized.	
`SIO4_USC_SMONO_TX_ACTIVE_WAIT`	This refers to waiting for activation.	

4.10.10.3. SIO4_IOCTL_USC_SMONO_TX_SHORT

This service configures the Short Sync Character selection for the transmitter when using the Slaved Monosync protocol.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_SMONO_TX_SHORT`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_SMONO_TX_SHORT_NO`	This refers to the use of normal sync character lengths. Use 8-bits.	
`SIO4_USC_SMONO_TX_SHORT_YES`	This refers to the use of short sync character lengths. Use less than 8-bits.	

4.10.10.4. SIO4_IOCTL_USC_SMONO_TX_SYNC

This service configures the transmitter Sync character when using the Slaved Monosync protocol.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_SMONO_TX_SYNC`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFF`	This is the valid range for this service.	

4.10.10.5. SIO4_IOCTL_USC_SMONO_TX_UNDERRUN

This service configures the Underrun reaction of the transmitter when using the Slaved Monosync protocol.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_SMONO_TX_UNDERRUN`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_SMONO_TX_UNDERRUN_CRC`	This refers to the sending of a CRC sequence,	
`SIO4_USC_SMONO_TX_UNDERRUN_NONE`	This refers to no action taking place.	

4.10.11. USC Transparent BiSync (TBSC) Protocol Specific IOCTL Services

These services are specific to the USC’s Transparent BSC protocol support.

4.10.11.1. SIO4_IOCTL_USC_TBSC_RX_ENCODING

This service configures the receiver’s character encoding format when using the Transparent Bisynchronous Serial Communications protocol.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_TBSC_RX_ENCODING`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_TBSC_ENCODING_ASCII`	This refers to ASCII encoding.	
`SIO4_USC_TBSC_ENCODING_EBCDIC`	This refers to EBCDIC encoding.	

4.10.11.2. SIO4_IOCTL_USC_TBSC_TX_ENCODING

This service configures the transmitter's character encoding format when using the Transparent Bisynchronous Serial Communications protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TBSC_TX_ENCODING`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_TBSC_ENCODING_ASCII`	This refers to ASCII encoding.	
`SIO4_USC_TBSC_ENCODING_EBCDIC`	This refers to EBCDIC encoding.	

4.10.11.3. SIO4_IOCTL_USC_TBSC_TX_PREAMBLE

This service configures the Preamble output selection for the transmitter when using the Transparent Bisynchronous Serial Communications protocol (Transparent BiSync).

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TBSC_TX_PREAMBLE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_TBSC_TX_PREAMBLE_NO`	This refers to the Preamble being disabled.	
`SIO4_USC_TBSC_TX_PREAMBLE_YES`	This refers to the Preamble being enabled.	

4.10.11.4. SIO4_IOCTL_USC_TBSC_TX_UNDERRUN

This service configures the Underrun reaction selection for the transmitter when using the Transparent Bisynchronous Serial Communications protocol.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TBSC_TX_UNDERRUN`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_TBSC_TX_UNDERRUN_C_D_S`	This refers to the sequence CRC, DLE and SYN.	
`SIO4_USC_TBSC_TX_UNDERRUN_C_S`	This refers to the sequence CRC and SYN.	
`SIO4_USC_TBSC_TX_UNDERRUN_D_S`	This refers to the sequence DLE and SYN.	
`SIO4_USC_TBSC_TX_UNDERRUN_S`	This refers to the sequence SYN only.	

4.10.12. USC Bit Rate Configuration Specific IOCTL Services

These services are specific to the USC's bit rate configuration support.

4.10.12.1. SIO4_IOCTL_USC_BRG0_CLK_SRC

This service configures the Baud Rate Generator Zero Clock Source.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_BRG0_CLK_SRC`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_BRG_CLK_SRC_CTR0`	Drive BRG0 from the output of Counter Zero (CTR0).	
`SIO4_USC_BRG_CLK_SRC_CTR1`	Drive BRG0 from the output of Counter One (CTR1).	
`SIO4_USC_BRG_CLK_SRC_RXC_PIN`	Drive BRG0 from the signal on the USC RxC pin.	
`SIO4_USC_BRG_CLK_SRC_TXC_PIN`	Drive BRG0 from the signal on the USC TxC pin.	

4.10.12.2. SIO4_IOCTL_USC_BRG0_ENABLE

This service enables or disables Baud Rate Generator Zero.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_BRG0_ENABLE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_BRG_ENABLE_NO`	This disables BRG0.	
`SIO4_USC_BRG_ENABLE_YES`	This enables BRG0.	

4.10.12.3. SIO4_IOCTL_USC_BRG0_MODE

This service configures the Baud Rate Generator Zero operating mode.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_BRG0_MODE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_BRG_MODE_CONT`	Run BRG0 continuously.	
`SIO4_USC_BRG_MODE_SINGLE`	Run BRG0 once then stop when it rolls over to zero.	

4.10.12.4. SIO4_IOCTL_USC_BRG1_CLK_SRC

This service configures the Baud Rate Generator One Clock Source.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_BRG1_CLK_SRC`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_BRG_CLK_SRC_CTR0`	Drive BRG1 from the output of Counter Zero (CTR0).	
`SIO4_USC_BRG_CLK_SRC_CTR1`	Drive BRG1 from the output of Counter One (CTR1).	
`SIO4_USC_BRG_CLK_SRC_RXC_PIN`	Drive BRG1 from the signal on the USC RxC pin.	
`SIO4_USC_BRG_CLK_SRC_TXC_PIN`	Drive BRG1 from the signal on the USC TxC pin.	

4.10.12.5. SIO4_IOCTL_USC_BRG1_ENABLE

This service enables or disables Baud Rate Generator One.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_BRG1_ENABLE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_BRG_ENABLE_NO`	This disables BRG1.	
`SIO4_USC_BRG_ENABLE_YES`	This enables BRG1.	

4.10.12.6. SIO4_IOCTL_USC_BRG1_MODE

This service configures the Baud Rate Generator One operating mode.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_BRG1_MODE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_BRG_MODE_CONT`	Run BRG1 continuously.	
`SIO4_USC_BRG_MODE_SINGLE`	Run BRG1 once then stop when it rolls over to zero.	

4.10.12.7. SIO4_IOCTL_USC_CTR0_CLK_SRC

This service configures the Counter Zero Clock Source.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_CTR0_CLK_SRC`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_CTR_CLK_SRC_DISABLE`	Disable CTR0.	
`SIO4_USC_CTR_CLK_SRC_RXC_PIN`	Drive CTR0 from the signal on the USC RxC pin.	
`SIO4_USC_CTR_CLK_SRC_TXC_PIN`	Drive CTR0 from the signal on the USC TxC pin.	

4.10.12.8. SIO4_IOCTL_USC_CTR0_RATE

This service configures the Counter Zero dividing Rate.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_CTR0_RATE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_CTR0_RATE_4X`	Divide the source clock by 4.	
`SIO4_USC_CTR0_RATE_8X`	Divide the source clock by 8.	
`SIO4_USC_CTR0_RATE_16X`	Divide the source clock by 16.	
`SIO4_USC_CTR0_RATE_32X`	Divide the source clock by 32.	

4.10.12.9. SIO4_IOCTL_USC_CTR1_CLK_SRC

This service configures the Counter One Clock Source.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_CTR1_CLK_SRC`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_CTR_CLK_SRC_DISABLE`	Disable CTR1.	
`SIO4_USC_CTR_CLK_SRC_RXC_PIN`	Drive CTR1 from the signal on the USC RxC pin.	
`SIO4_USC_CTR_CLK_SRC_TXC_PIN`	Drive CTR1 from the signal on the USC TxC pin.	

4.10.12.10. SIO4_IOCTL_USC_CTR1_RATE

This service configures the Counter One dividing Rate.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_CTR1_RATE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_CTR1_RATE_CTR0`	Use the rate configured for CTR0.	
`SIO4_USC_CTR1_RATE_DPLL`	Use the rate configured for the DPLL.	

4.10.12.11. SIO4_IOCTL_USC_TC0

This service configures the Time Constant Zero values used by BRG0.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TC0`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFFFF`	This is the valid range for this service.	

4.10.12.12. SIO4_IOCTL_USC_TC1

This service configures the Time Constant One values used by BRG1.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TC1`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFFFF`	This is the valid range for this service.	

4.10.13. USC CRC Specific IOCTL Services

These services are specific to the USC's CRC support.

4.10.13.1. SIO4_IOCTL_USC_RX_CRC_ENABLE

This service configures the receiver's use of CRC.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_RX_CRC_ENABLE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_CRC_ENABLE_NO`	This refers to the receiver not using CRCs.	
`SIO4_USC_CRC_ENABLE_YES`	This refers to the receiver using CRCs.	

4.10.13.2. SIO4_IOCTL_USC_RX_CRC_PRESET

This service configures the receiver's CRC starting value when CRC use is enabled.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_RX_CRC_PRESET`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_CRC_PRESET_ALL_0`	This refers to a starting value in which each bit is zero.	
`SIO4_USC_CRC_PRESET_ALL_1`	This refers to a starting value in which each bit is one.	

4.10.13.3. SIO4_IOCTL_USC_RX_CRC_TYPE

This service configures the receiver's CRC type when CRC use is enabled.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_RX_CRC_TYPE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_CRC_TYPE_16`	This refers to a 16-bit CRC.	
`SIO4_USC_CRC_TYPE_32`	This refers to a 32-bit CRC.	
`SIO4_USC_CRC_TYPE_CCITT`	This refers to the CCITT style CRC.	

4.10.13.4. SIO4_IOCTL_USC_TX_CRC_ENABLE

This service configures the transmitter's use of CRC.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TX_CRC_ENABLE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_CRC_ENABLE_NO`	This refers to the transmitter not using CRCs.	
`SIO4_USC_CRC_ENABLE_YES`	This refers to the transmitter using CRCs.	

4.10.13.5. SIO4_IOCTL_USC_TX_CRC_ON_END

This service configures the transmitter's sending of a CRC at the end of a frame or message.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TX_CRC_ON_END`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_TX_CRC_ON_END_NO`	This refers to the transmitter not sending a CRC.	
`SIO4_USC_TX_CRC_ON_END_YES`	This refers to the transmitter sending a CRC.	

4.10.13.6. SIO4_IOCTL_USC_TX_CRC_PRESET

This service configures the transmitter's CRC starting value when CRC use is enabled.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TX_CRC_PRESET`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_CRC_PRESET_ALL_0`	This refers to a starting value in which each bit is zero.	
`SIO4_USC_CRC_PRESET_ALL_1`	This refers to a starting value in which each bit is one.	

4.10.13.7. SIO4_IOCTL_USC_TX_CRC_TYPE

This service configures the transmitter's CRC type when CRC use is enabled.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TX_CRC_TYPE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_CRC_TYPE_16`	This refers to a 16-bit CRC.	
`SIO4_USC_CRC_TYPE_32`	This refers to a 32-bit CRC.	
`SIO4_USC_CRC_TYPE_CCITT`	This refers to the CCITT style CRC.	

4.10.14. USC DPLL Configuration Specific IOCTL Services

These services are specific to the USC's DPLL configuration support.

4.10.14.1. SIO4_IOCTL_USC_DPLL_ADJ_SYNC

This service configures the DPLL's synchronization functionality.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_DPLL_ADJ_SYNC`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_DPLL_ADJ_SYNC_BOTH_EDGE`	Have the DPLL synchronize on both falling and rising edges.	
`SIO4_USC_DPLL_ADJ_SYNC_FALL_EDGE`	Have the DPLL synchronize on falling edges only.	
`SIO4_USC_DPLL_ADJ_SYNC_INHIBIT`	Inhibit the DPLL from synchronizing.	
`SIO4_USC_DPLL_ADJ_SYNC_RISE_EDGE`	Have the DPLL synchronize on rising edges only.	

4.10.14.2. SIO4_IOCTL_USC_DPLL_CLK_SRC

This service configures the DPLL’s clock source.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_DPLL_CLK_SRC`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_DPLL_CLK_SRC_BRG0`	Drive the DPLL from the output of BRG0.	
`SIO4_USC_DPLL_CLK_SRC_BRG1`	Drive the DPLL from the output of BRG1.	
`SIO4_USC_DPLL_CLK_SRC_RXC_PIN`	Drive the DPLL from the signal on the USC RxC pin.	
`SIO4_USC_DPLL_CLK_SRC_TXC_PIN`	Drive the DPLL from the signal on the USC TxC pin.	

4.10.14.3. SIO4_IOCTL_USC_DPLL_MISS_1

This service operates on the DPLL’s status reported when it misses one clock cycle.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_DPLL_MISS_1`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current status.	The service is unsupported.
`SIO4_USC_DPLL_MISS_1_CLEAR`	This clears the status.	N/A
`SIO4_USC_DPLL_MISS_1_NO`	N/A	The DPLL did not miss a clock cycle.
`SIO4_USC_DPLL_MISS_1_YES`	N/A	The DPLL missed a clock cycle.

4.10.14.4. SIO4_IOCTL_USC_DPLL_MISS_2

This service operates on the DPLL’s status reported when it misses two clock cycles.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_DPLL_MISS_2`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current status.	The service is unsupported.
`SIO4_USC_DPLL_MISS_2_CLEAR`	This clears the status.	N/A
`SIO4_USC_DPLL_MISS_2_NO`	N/A	The DPLL missed less than two clock cycles.
`SIO4_USC_DPLL_MISS_2_YES`	N/A	The DPLL missed two or more clock cycles.

4.10.14.5. SIO4_IOCTL_USC_DPLL_MODE

This service configures the DPLL's operating mode, which corresponds to the encoding of the source clock.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_DPLL_MODE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_DPLL_MODE_BIPH_LVL`	The source signal is encoded as Biphase Level.	
`SIO4_USC_DPLL_MODE_BIPH_MS`	The source signal is encoded as Biphase Mark or Space.	
`SIO4_USC_DPLL_MODE_DISABLE`	Disable the DPLL.	
`SIO4_USC_DPLL_MODE_NRZ_NRZI`	The source signal is encoded as NRZ or NRZ Inverted.	

4.10.14.6. SIO4_IOCTL_USC_DPLL_RATE

This service configures the DPLL's clock dividing rate.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_DPLL_RATE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_DPLL_RATE_CTR1_4X`	Divide the source by four. This is for CTR1 operation only. The DPLL should be disabled when this option is used.	
`SIO4_USC_DPLL_RATE_8X`	Divide the source by eight.	

`SIO4_USC_DPLL_RATE_16X`	Divide the source by 16.
`SIO4_USC_DPLL_RATE_32X`	Divide the source by 32.

4.10.14.7. SIO4_IOCTL_USC_DPLL_SYNC

This service operates on the DPLL's synchronization status.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_DPLL_SYNC`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_DPLL_SYNC_START`	This requests that the DPLL synchronize on its source signal.	N/A
`SIO4_USC_DPLL_SYNC_NO`	The DPLL is not synchronized.	
`SIO4_USC_DPLL_SYNC_YES`	The DPLL is synchronized.	

4.10.15. USC Miscellaneous IOCTL Services

4.10.15.1. SIO4_IOCTL_USC_ACCEPT_CV

This service configures the Code Violations. This is not applicable to all protocols.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_ACCEPT_CV`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_ACCEPT_CV_NO`	Do not accept Code Violations.	
`SIO4_USC_ACCEPT_CV_YES`	Accept Code Violations.	

4.10.15.2. SIO4_IOCTL_USC_LOOP_SENDING

This service reports the channel's Loop Sending status.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_LOOP_SENDING`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_LOOP_SENDING_NO`	N/A	The Loop Sending status is negated.
`SIO4_USC_LOOP_SENDING_YES`	N/A	The Loop Sending status is asserted.

4.10.15.3. SIO4_IOCTL_USC_ON_LOOP

This service reports the On Loop status for the USC.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_ON_LOOP`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current status.	The service is unsupported.
`SIO4_USC_ON_LOOP_NO`	N/A	The On Loop status is negated.
`SIO4_USC_ON_LOOP_YES`	N/A	The On Loop status is asserted.

4.10.15.4. SIO4_IOCTL_USC_OPER_MODE

This service configures the basic operating mode of the USC.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_OPER_MODE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_OPER_MODE_AUTO_ECHO`	The activity at the receiver is duplicated at the transmitter.	
`SIO4_USC_OPER_MODE_EXT_LOOPBACK`	The USC operates in External Loopback mode for testing.	
`SIO4_USC_OPER_MODE_INT_LOOPBACK`	The USC operates in Internal Loopback mode for testing.	
`SIO4_USC_OPER_MODE_NORMAL`	The USC operates in its normal mode where data is received and transmitted per the protocols selected.	

4.10.15.5. SIO4_IOCTL_USC_RESET

This service resets the USC. The USC is also reset when the channel is initialized (see `SIO4_IOCTL_INITIALIZE`, section 4.8.5.1, page 44).

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_RESET`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0`	Reset the USC.	The service is supported or the USC was reset.

4.10.15.6. SIO4_IOCTL_USC_SEND_COMMAND

This service sends a command to the USC.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_SEND_COMMAND`
`arg`	`s32*`

The table below lists the options used with this service. Refer to the USC hardware manual for additional information on sending commands to the USC.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_SEND_CMD_FIFOS_PURGE`	Purges the USC FIFOs.	
`SIO4_USC_SEND_CMD_LD_CHAR_CNTS`	Load the Receive and Transmit Character Counts.	
`SIO4_USC_SEND_CMD_LD_RX_CHAR_CNT`	Load the Receive Character Count.	
`SIO4_USC_SEND_CMD_LD_TC0`	Load the TC0 count.	
`SIO4_USC_SEND_CMD_LD_TC0_TC1`	Load the TC0 and TC1 counts.	
`SIO4_USC_SEND_CMD_LD_TC1`	Load the TC1 count.	
`SIO4_USC_SEND_CMD_LD_TX_CHAR_CNT`	Load the Transmit Character Count.	
`SIO4_USC_SEND_CMD_NONE`	This is ignored.	
`SIO4_USC_SEND_CMD_RESET_H_IUS`	Reset the Highest Interrupt Under Service.	
`SIO4_USC_SEND_CMD_RX_FIFO_PURGE`	Purge the USC receive FIFO.	
`SIO4_USC_SEND_CMD_RX_PURGE`	Purge the receiver.	
`SIO4_USC_SEND_CMD_SEL_LSB_FIRST`	Select Least Significant Byte First processing.	
`SIO4_USC_SEND_CMD_SEL_MSB_FIRST`	Select Most Significant Byte First processing.	
`SIO4_USC_SEND_CMD_SEL_STRAIT_MEM`	Select Strain memory access.	
`SIO4_USC_SEND_CMD_SEL_SWAP_MEM`	Select Swapped memory access.	
`SIO4_USC_SEND_CMD_TRIG_LD_DMA`	Trigger channel loading via DMA.	
`SIO4_USC_SEND_CMD_TRIG_RX_DMA`	Trigger the receiver to resume data transfer.	
`SIO4_USC_SEND_CMD_TRIG_TX_DMA`	Trigger the transmitter to resume data transfer.	
`SIO4_USC_SEND_CMD_TRIG_TX_RX_DMA`	Trigger the receiver and transmitter to resume data transfer.	
`SIO4_USC_SEND_CMD_TX_FIFO_PURGE`	Purge the USC transmit FIFO.	

4.10.16. USC Parity Specific IOCTL Services

These services are specific to the USC’s parity support.

4.10.16.1. SIO4_IOCTL_USC_RX_PAR_ENABLE

This service configures the USC receiver's use of Parity.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_RX_PAR_ENABLE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_PAR_ENABLE_NO`	This refers to Parity use being disabled.	
`SIO4_USC_PAR_ENABLE_YES`	This refers to Parity use being enabled.	

4.10.16.2. SIO4_IOCTL_USC_RX_PAR_TYPE

This service configures the type of parity used by the USC receiver when Parity use is enabled.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_RX_PAR_TYPE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_PAR_TYPE_EVEN`	This refers to Even Parity.	
`SIO4_USC_PAR_TYPE_ODD`	This refers to Odd Parity.	
`SIO4_USC_PAR_TYPE_ONE`	This refers to One Parity (parity bit is always one).	
`SIO4_USC_PAR_TYPE_ZERO`	This refers to Zero Parity (parity bit is always zero).	

4.10.16.3. SIO4_IOCTL_USC_TX_PAR_ENABLE

This service configures the USC transmitter's use of Parity.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TX_PAR_ENABLE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_PAR_ENABLE_NO`	This refers to Parity use being disabled.	
`SIO4_USC_PAR_ENABLE_YES`	This refers to Parity use being enabled.	

4.10.16.4. SIO4_IOCTL_USC_TX_PAR_TYPE

This service configures the type of parity used by the USC transmitter when Parity use is enabled.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_TX_PAR_TYPE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_PAR_TYPE_EVEN`	This refers to Even Parity.	
`SIO4_USC_PAR_TYPE_ODD`	This refers to Odd Parity.	
`SIO4_USC_PAR_TYPE_ONE`	This refers to One Parity (parity bit is always one).	
`SIO4_USC_PAR_TYPE_ZERO`	This refers to Zero Parity (parity bit is always zero).	

4.10.17. USC Receive Character Count FIFO Specific IOCTL Services

The USC maintains a small Receive Character Count (RCC) FIFO that records frame size information at the end of each received frame. This information is used to determine the size of the received frame, but the FIFO is only four elements deep. The driver implements a replacement software FIFO that is 256 elements deep. The below services are used to interact with both RCC FIFOs.

> **NOTE**: Other related IOCTL services are defined, but are not intended for application use.

4.10.17.1. SIO4_IOCTL_USC_RCC_FIFO_CLEAR

This service operates on the Receive Character Counter FIFO.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_RCC_FIFO_CLEAR`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`0`	The service is supported.	N/A
`SIO4_USC_RCC_FIFO_CLEAR_NO`	This refers to the FIFO not being cleared.	
`SIO4_USC_RCC_FIFO_CLEAR_YES`	This refers to the FIFO being cleared.	

4.10.17.2. SIO4_IOCTL_USC_RCC_FIFO_OVERRUN

This service reports on the Overrun status of the Receive Character Counter FIFO.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_RCC_FIFO_OVERRUN`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current status.	The service is unsupported.
`SIO4_USC_RCC_FIFO_OVERRUN_NO`	N/A	An overrun has not occurred.
`SIO4_USC_RCC_FIFO_OVERRUN_YES`	N/A	An overrun has occurred.

4.10.17.3. SIO4_IOCTL_USC_RCC_FIFO_VALID

This service reports on the validity of the Receive Character Counter FIFO content.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_RCC_FIFO_VALID`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_RCC_FIFO_VALID_NO`	N/A	The RCC FIFO does not contain valid data.
`SIO4_USC_RCC_FIFO_VALID_YES`	N/A	The RCC FIFO does contain valid data.

4.10.18. USC Receiver Specific IOCTL Services

These services are specific to the USC’s receiver.

4.10.18.1. SIO4_IOCTL_USC_RX_CHAR_CNT

This service reports the receive Character Count, which is the size of the most recent frame or message.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_RX_CHAR_CNT`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFFFF`	N/A	This is the valid range for this service.

4.10.18.2. SIO4_IOCTL_USC_RX_CHAR_CNT_LIM

This service configures the upper size limit for received frames or messages.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_RX_CHAR_CNT_LIM`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFFFF`	This is the valid range for this service.	

4.10.18.3. SIO4_IOCTL_USC_RX_CHAR_LEN

This service configures the bit length of received characters.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_RX_CHAR_LEN`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_CHAR_LEN_1`	This refers to a character length of 1-bit.	
`SIO4_USC_CHAR_LEN_2`	This refers to a character length of 2-bits.	
`SIO4_USC_CHAR_LEN_3`	This refers to a character length of 3-bits.	
`SIO4_USC_CHAR_LEN_4`	This refers to a character length of 4-bits.	
`SIO4_USC_CHAR_LEN_5`	This refers to a character length of 5-bits.	
`SIO4_USC_CHAR_LEN_6`	This refers to a character length of 6-bits.	
`SIO4_USC_CHAR_LEN_7`	This refers to a character length of 7-bits.	
`SIO4_USC_CHAR_LEN_8`	This refers to a character length of 8-bits.	

4.10.18.4. SIO4_IOCTL_USC_RX_CLK_SRC

This service configures receiver clock source selection.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_RX_CLK_SRC`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_CLK_SRC_BRG0`	This refers to the output from BRG0.	
`SIO4_USC_CLK_SRC_BRG1`	This refers to the output from BRG1.	
`SIO4_USC_CLK_SRC_CTR0`	This refers to the output from CTR0.	
`SIO4_USC_CLK_SRC_CTR1`	This refers to the output from CTR1.	
`SIO4_USC_CLK_SRC_DPLL`	This refers to the output from DPLL.	
`SIO4_USC_CLK_SRC_DISABLE`	This disables the receiver's operation.	
`SIO4_USC_CLK_SRC_RXC_PIN`	This refers to the USC's RxC pin.	
`SIO4_USC_CLK_SRC_TXC_PIN`	This refers to the USC's TxC pin.	

4.10.18.5. SIO4_IOCTL_USC_RX_CMD

This service sends a command to the USC receiver.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_RX_CMD`
`arg`	`s32*`

The table below lists the options used with this service. Refer to the USC hardware manual for additional information on sending commands to the receiver.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_RX_CMD_ENTER_HUNT_MODE`	This refers to the USC Enter Hunt Mode operation.	
`SIO4_USC_RX_CMD_NULL`	This refers to no command at all.	
`SIO4_USC_RX_CMD_PRESET_CRC`	This refers to presetting the CRC.	
`SIO4_USC_RX_CMD_SEL_FIFO_STATUS`	This refers to selection of the USC's Rx FIFO status.	
`SIO4_USC_RX_CMD_SEL_FIFO_INT_LVL`	This refers to USC Rx FIFO fill threshold level for interrupt generation.	
`SIO4_USC_RX_CMD_SEL_FIFO_STS_LVL`	This refers to USC Rx FIFO fill level.	

4.10.18.6. SIO4_IOCTL_USC_RX_DATA_ENCODE

This service configures the data encoding format used by the USC receiver.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_RX_DATA_ENCODE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_DATA_ENCODE_BI_LEVEL`	This refers to Biphase-Level encoding (Manchester).	
`SIO4_USC_DATA_ENCODE_BI_MARK`	This refers to Biphase-Mark encoding (FM1).	
`SIO4_USC_DATA_ENCODE_BI_SPACE`	This refers to Biphase-Space encoding (FM0).	

`SIO4_USC_DATA_ENCODE_D_BI_LEVEL`	This refers to Differential Biphase Level encoding (Differential Manchester).
`SIO4_USC_DATA_ENCODE_NRZ`	This refers to Non-Return to Zero encoding.
`SIO4_USC_DATA_ENCODE_NRZB`	This refers to Inverted NRZ encoding.
`SIO4_USC_DATA_ENCODE_NRZI_MARK`	This refers to NRZI-Mark encoding.
`SIO4_USC_DATA_ENCODE_NRZI_SPACE`	This refers to NRZI-Space encoding.

4.10.18.7. SIO4_IOCTL_USC_RX_ENABLE

This service configures the USC receiver enabled state.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_RX_ENABLE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_ENABLE_NO_AFTER`	This refers to the receiver being disabled after receipt of the current character, if one is being received.	
`SIO4_USC_ENABLE_NO_NOW`	This refers to the receiver being disabled immediately.	
`SIO4_USC_ENABLE_YES_NOW`	This refers to the receiver being enabled immediately.	
`SIO4_USC_ENABLE_YES_W_AE`	This refers to the receiver being enabled immediately along with the hardware flow control signals.	

4.10.18.8. SIO4_IOCTL_USC_RX_MODE

This service configures the protocol used by the USC receiver for data reception.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_RX_MODE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_MODE_8023`	This refers to the 802.3 protocol.	
`SIO4_USC_MODE_ASY_CV`	This refers to the Asynchronous with Code Violations protocol.	
`SIO4_USC_MODE_ASYNC`	This refers to the Asynchronous protocol.	
`SIO4_USC_MODE_BSC`	This refers to the Bisynchronous Serial Communications protocol.	
`SIO4_USC_MODE_E_SYNC`	This refers to the External Sync protocol.	
`SIO4_USC_MODE_HDLC`	This refers to the HDLC protocol.	
`SIO4_USC_MODE_ISOC`	This refers to the Isochronous protocol.	
`SIO4_USC_MODE_MONO`	This refers to the Monosync protocol.	
`SIO4_USC_MODE_NBIP`	This refers to the Nine-Bit Interprocessor Protocol.	
`SIO4_USC_MODE_TBSC`	This refers to the Transparent Bisynchronous Serial Communications protocol.	

4.10.18.9. SIO4_IOCTL_USC_RX_QUEUE_ABORT

This service operates on the USC receiver to queue Abort detection status through the Rx FIFO.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_RX_QUEUE_ABORT`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_RX_QUEUE_ABORT_NO`	This specifies to not queue the status through the Rx FIFO.	
`SIO4_USC_RX_QUEUE_ABORT_YES`	This specifies to queue the status through the Rx FIFO.	

4.10.18.10. SIO4_IOCTL_USC_RX_STATUS

This service retrieves the USC receiver status, which is essentially the value read from the Receive Command/Status Register.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_RX_STATUS`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFFFF`	N/A	This is the valid range for this service.

4.10.18.11. SIO4_IOCTL_USC_RX_STATUS_BLOCK

This service configures the USC receiver’s use of Receive Status Blocks.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_RX_STATUS_BLOCK`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_STATUS_BLOCK_1_WORD`	This refers to Receive Status Blocks consisting of one word (two bytes).	
`SIO4_USC_STATUS_BLOCK_2_WORD`	This refers to Receive Status Blocks consisting of two words (four bytes).	
`SIO4_USC_STATUS_BLOCK_NO`	This refers to not using Receive Status Blocks.	

4.10.18.12. SIO4_IOCTL_USC_RX_WAIT_DMA_TRIG

This service configures the USC receiver’s pausing of data transfer to the channel’s main Rx FIFO.

> **NOTE**: By default, each channel is configured to automatically transfer data to the channel’s main Rx FIFO as data is received by the USC. This service deals with the pausing of such data transfer until commanded to resume the transfer.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_RX_WAIT_DMA_TRIG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_WAIT_DMA_TRIG_NO`	This refers to data transfer being unimpeded.	
`SIO4_USC_WAIT_DMA_TRIG_YES`	This refers to data transfer being paused.	

4.10.19. USC Signal Routing Specific IOCTL Services

These services are specific to the USC’s signal routing support.

4.10.19.1. SIO4_IOCTL_USC_CTS_CFG

This service configures the USC’s CTS (Clear to Send) signal functionality.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_CTS_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_CTS_CFG_IN_CBL_CTS`	Configure the signal as an input for CTS operation.	
`SIO4_USC_CTS_CFG_OUT_0`	Configure the signal as an output driven low.	
`SIO4_USC_CTS_CFG_OUT_1`	Configure the signal as an output driven high.	
`SIO4_USC_CTS_CFG_TRI`	Tri-state the signal.	

4.10.19.2. SIO4_IOCTL_USC_CTS_LEG

This service configures the USC's CTS (Clear to Send) pin for legacy cable interface configurations. Please also read Cable Configuration Modes (section 8.6, page 137).

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_CTS_LEG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The feature is unsupported or inaccessible.
`SIO4_USC_CTS_LEG_IN`	Configure the signal as an input.	
`SIO4_USC_CTS_LEG_OUT_0`	Configure the signal as an output driven low.	
`SIO4_USC_CTS_LEG_OUT_1`	Configure the signal as an output driven high.	

4.10.19.3. SIO4_IOCTL_USC_DCD_CFG

This service configures the USC's DCD (Data Carrier Detect) signal functionality.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_DCD_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_DCD_CFG_DISABLE`	Disable the signal.	
`SIO4_USC_DCD_CFG_IN_DCD_CBL_DCD`	Use the cable DCD signal for DCD operation.	
`SIO4_USC_DCD_CFG_IN_SYNC_CBL_DCD`	Use the cable DCD signal for SYNC operation.	
`SIO4_USC_DCD_CFG_OUT_0`	Configure the signal as an output driven low. *	
`SIO4_USC_DCD_CFG_OUT_1`	Configure the signal as an output driven high. *	

* This option enables the cable DCD signal to be driven, though the `SIO4_IOCTL_Z16_CBL_DCD_CFG` IOCTL service (section 4.9.1.1, page 61) may configure the cable to output an alternate selection.

4.10.19.4. SIO4_IOCTL_USC_DCD_LEG

This service configures the USC's DCD (Data Carrier Detect) pin for legacy cable interface configurations. Please also read Cable Configuration Modes (section 8.6, page 137).

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_DCD_LEG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The feature is unsupported or inaccessible.
`SIO4_USC_DCD_LEG_IN_DCD`	Use the pin as the DCD input.	
`SIO4_USC_DCD_LEG_IN_SYNC`	Use the pin as the SYNC input.	
`SIO4_USC_DCD_LEG_OUT_0`	Configure the signal as an output driven low.	
`SIO4_USC_DCD_LEG_OUT_1`	Configure the signal as an output driven high.	

4.10.19.5. SIO4_IOCTL_USC_RXC_CFG

This service configures the clock source and signal routing of the USC’s RxC pin.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_RXC_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_RXC_CFG_IN_0`	This refers to RxC being an input which is driven low.	
`SIO4_USC_RXC_CFG_IN_1`	This refers to RxC being an input which is driven high.	
`SIO4_USC_RXC_CFG_IN_CBL_RXAUXC`	This refers to RxC being an input which is driven from the Rx Auxiliary Clock signal at the cable interface.	
`SIO4_USC_RXC_CFG_IN_CBL_RXC`	This refers to RxC being an input which is driven from the Rx Clock signal at the cable interface.	
`SIO4_USC_RXC_CFG_IN_OSC`	This refers to RxC being an input which is driven from the on-board oscillator.	
`SIO4_USC_RXC_CFG_IN_OSC_INV`	This refers to RxC being an input which is driven from the inverted form of the on-board oscillator.	
`SIO4_USC_RXC_CFG_OUT_BRG0`	This refers to RxC being an output which is driven from BRG0.	
`SIO4_USC_RXC_CFG_OUT_BRG1`	This refers to RxC being an output which is driven from BRG1.	
`SIO4_USC_RXC_CFG_OUT_CTR0`	This refers to RxC being an output which is driven from CTR0.	
`SIO4_USC_RXC_CFG_OUT_DPLL_RX`	This refers to RxC being an output which is driven from the DPLL’s receive output clock.	
`SIO4_USC_RXC_CFG_OUT_RX_BYTE_CLK`	This refers to RxC being an output which is driven from the receiver’s Byte Clock.	
`SIO4_USC_RXC_CFG_OUT_RX_CLK`	This refers to RxC being an output which is driven from the Receive Clock.	
`SIO4_USC_RXC_CFG_OUT_SYNC`	This refers to RxC being an output which is driven from the Sync signal.	

4.10.19.6. SIO4_IOCTL_USC_RXC_LEG

This service configures the signal routing of the USC’s RxC pin for legacy cable interface configurations. Please also read Cable Configuration Modes (section 8.6, page 137).

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_RXC_LEG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The feature is unsupported or inaccessible.
`SIO4_USC_RXC_LEG_IN`	This refers to RxC being an input.	
`SIO4_USC_RXC_LEG_OUT_BRG0`	This refers to RxC being an output which is driven from BRG0.	
`SIO4_USC_RXC_LEG_OUT_BRG1`	This refers to RxC being an output which is driven from BRG1.	
`SIO4_USC_RXC_LEG_OUT_CTR0`	This refers to RxC being an output which is driven from CTR0.	
`SIO4_USC_RXC_LEG_OUT_DPLL_RX`	This refers to RxC being an output which is driven from the DPLL's receive output clock.	
`SIO4_USC_RXC_LEG_OUT_RX_BYTE_CLK`	This refers to RxC being an output which is driven from the receiver's Byte Clock.	
`SIO4_USC_RXC_LEG_OUT_RX_CLK`	This refers to RxC being an output which is driven from the Receive Clock.	
`SIO4_USC_RXC_LEG_OUT_SYNC`	This refers to RxC being an output which is driven from the Sync signal.	

4.10.19.7. SIO4_IOCTL_USC_TXC_CFG

This service configures the clock source and signal routing of the USC's TxC pin.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TXC_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_TXC_CFG_IN_0`	This refers to TxC being an input which is driven low.	
`SIO4_USC_TXC_CFG_IN_1`	This refers to TxC being an input which is driven high.	
`SIO4_USC_TXC_CFG_IN_CBL_RXAUXC`	This refers to TxC being an input which is driven from the Rx Auxiliary Clock signal at the cable interface.	
`SIO4_USC_TXC_CFG_IN_CBL_RXC`	This refers to TxC being an input which is driven from the Rx Clock signal at the cable interface.	
`SIO4_USC_TXC_CFG_IN_OSC`	This refers to TxC being an input which is driven from the on-board oscillator.	
`SIO4_USC_TXC_CFG_IN_OSC_INV`	This refers to TxC being an input which is driven from the inverted form of the on-board oscillator.	
`SIO4_USC_TXC_CFG_OUT_BRG0`	This refers to TxC being an output which is driven from BRG0.	

`SIO4_USC_TXC_CFG_OUT_BRG1`	This refers to TxC being an output which is driven from BRG1.
`SIO4_USC_TXC_CFG_OUT_CTR1`	This refers to TxC being an output which is driven from CTR1.
`SIO4_USC_TXC_CFG_OUT_DPLL_TX`	This refers to TxC being an output which is driven from the DPLL's transmitter output clock.
`SIO4_USC_TXC_CFG_OUT_TX_BYTE_CLK`	This refers to TxC being an output which is driven from the transmitter's Byte Clock.
`SIO4_USC_TXC_CFG_OUT_TX_CLK`	This refers to TxC being an output which is driven from the Transmit Clock.
`SIO4_USC_TXC_CFG_OUT_TX_COMP`	This refers to TxC being an output which is driven from the Complete signal.

4.10.19.8. SIO4_IOCTL_USC_TXC_LEG

This service configures the signal routing of the USC's TxC pin for legacy cable interface configurations. Please also read Cable Configuration Modes (section 8.6, page 137).

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TXC_LEG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The feature is unsupported or inaccessible.
`SIO4_USC_TXC_LEG_IN`	This refers to TxC being an input.	
`SIO4_USC_TXC_LEG_OUT_BRG0`	This refers to TxC being an output which is driven from BRG0.	
`SIO4_USC_TXC_LEG_OUT_BRG1`	This refers to TxC being an output which is driven from BRG1.	
`SIO4_USC_TXC_LEG_OUT_CTR1`	This refers to TxC being an output which is driven from CTR1.	
`SIO4_USC_TXC_LEG_OUT_DPLL_TX`	This refers to TxC being an output which is driven from the DPLL's transmitter output clock.	
`SIO4_USC_TXC_LEG_OUT_TX_BYTE_CLK`	This refers to TxC being an output which is driven from the transmitter's Byte Clock.	
`SIO4_USC_TXC_LEG_OUT_TX_CLK`	This refers to TxC being an output which is driven from the Transmit Clock.	
`SIO4_USC_TXC_LEG_OUT_TX_COMP`	This refers to TxC being an output which is driven from the Complete signal.	

4.10.19.9. SIO4_IOCTL_USC_TXD_CFG

This service configures the clock source and signal routing of the USC's TxD pin.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TXD_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_TXD_CFG_OUT_0`	This refers to TxD being an output which is driven low.	
`SIO4_USC_TXD_CFG_OUT_1`	This refers to TxD being an output which is driven high.	
`SIO4_USC_TXD_CFG_OUT_TXD`	This refers to TxD being an output which is driven to the Tx Data signal at the cable interface.	
`SIO4_USC_TXD_CFG_TRI`	This refers to TxD being tri-stated.	

4.10.20. USC Transmitter Specific IOCTL Services

These services are specific to the USC's transmitter.

4.10.20.1. SIO4_IOCTL_USC_TX_CHAR_CNT

This service reports the size to be used for the next transmission frame or message.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TX_CHAR_CNT`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFFFF`		This is the valid range for this service.

4.10.20.2. SIO4_IOCTL_USC_TX_CHAR_CNT_LIM

This service configures the upper size limit for transmission frames or messages.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TX_CHAR_CNT_LIM`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFFFF`	This is the valid range for this service.	

4.10.20.3. SIO4_IOCTL_USC_TX_CHAR_LEN

This service configures the bit length of transmitted characters.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TX_CHAR_LEN`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_CHAR_LEN_1`	This refers to a character length of 1-bit.	
`SIO4_USC_CHAR_LEN_2`	This refers to a character length of 2-bits.	
`SIO4_USC_CHAR_LEN_3`	This refers to a character length of 3-bits.	
`SIO4_USC_CHAR_LEN_4`	This refers to a character length of 4-bits.	
`SIO4_USC_CHAR_LEN_5`	This refers to a character length of 5-bits.	
`SIO4_USC_CHAR_LEN_6`	This refers to a character length of 6-bits.	
`SIO4_USC_CHAR_LEN_7`	This refers to a character length of 7-bits.	
`SIO4_USC_CHAR_LEN_8`	This refers to a character length of 8-bits.	

4.10.20.4. SIO4_IOCTL_USC_TX_CLK_SRC

This service configures transmitter clock source selection.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TX_CLK_SRC`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_CLK_SRC_BRG0`	This refers to the output from BRG0.	
`SIO4_USC_CLK_SRC_BRG1`	This refers to the output from BRG1.	
`SIO4_USC_CLK_SRC_CTR0`	This refers to the output from CTR0.	
`SIO4_USC_CLK_SRC_CTR1`	This refers to the output from CTR1.	
`SIO4_USC_CLK_SRC_DPLL`	This refers to the output from DPLL.	
`SIO4_USC_CLK_SRC_DISABLE`	This disables the receiver's operation.	
`SIO4_USC_CLK_SRC_RXC_PIN`	This refers to the USC's RxC pin.	
`SIO4_USC_CLK_SRC_TXC_PIN`	This refers to the USC's TxC pin.	

4.10.20.5. SIO4_IOCTL_USC_TX_CMD

This service sends a command to the USC transmitter.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TX_CMD`
`arg`	`s32*`

The table below lists the options used with this service. Refer to the USC hardware manual for additional information on sending commands to the transmitter.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_TX_CMD_NULL`	This refers to no command at all.	
`SIO4_USC_TX_CMD_PRESET_CRC`	This refers to presetting the CRC.	
`SIO4_USC_TX_CMD_ENTER_HUNT_MODE`	This refers to the USC Enter Hunt Mode operation.	
`SIO4_USC_TX_CMD_RST_DLE_INHIBIT`	This refers to resetting the DLE inhibit status.	
`SIO4_USC_TX_CMD_RST_EOF_EOM`	This refers to resetting the End of Frame or End of Message status.	
`SIO4_USC_TX_CMD_SEL_FIFO_INT_LVL`	This refers to USC Tx FIFO fill threshold level for interrupt generation.	
`SIO4_USC_TX_CMD_SEL_FIFO_STATUS`	This refers to selection of the USC's Tx FIFO status.	
`SIO4_USC_TX_CMD_SEL_FIFO_STS_LVL`	This refers to USC Tx FIFO fill level.	
`SIO4_USC_TX_CMD_SEND_ABORT`	This refers to sending an abort sequence.	
`SIO4_USC_TX_CMD_SEND_FRM_MSG`	This refers to sending a frame or message.	
`SIO4_USC_TX_CMD_SET_DLE_INHIBIT`	This refers to setting the DLE inhibit status.	
`SIO4_USC_TX_CMD_SET_EOF_EOM`	This refers to setting the End of Frame or End of Message status.	

4.10.20.6. SIO4_IOCTL_USC_TX_CTRL_BLOCK

This service configures the USC transmitter's use of Transmit Control Blocks.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_TX_CTRL_BLOCK`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_CTRL_BLOCK_1_WORD`	This refers to Transmit Status Blocks consisting of one word (two bytes).	
`SIO4_USC_CTRL_BLOCK_2_WORD`	This refers to Transmit Status Blocks consisting of two words (four bytes).	
`SIO4_USC_CTRL_BLOCK_NO`	This refers to not using Transmit Status Blocks.	

4.10.20.7. SIO4_IOCTL_USC_TX_DATA_ENCODE

This service configures the data encoding format used by the USC transmitter.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_TX_DATA_ENCODE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_DATA_ENCODE_BI_LEVEL`	This refers to Biphase-Level encoding (Manchester).	
`SIO4_USC_DATA_ENCODE_BI_MARK`	This refers to Biphase-Mark encoding (FM1).	
`SIO4_USC_DATA_ENCODE_BI_SPACE`	This refers to Biphase-Space encoding (FM0).	
`SIO4_USC_DATA_ENCODE_D_BI_LEVEL`	This refers to Differential Biphase Level encoding (Differential Manchester).	
`SIO4_USC_DATA_ENCODE_NRZ`	This refers to Non-Return to Zero encoding.	
`SIO4_USC_DATA_ENCODE_NRZB`	This refers to Inverted NRZ encoding.	
`SIO4_USC_DATA_ENCODE_NRZI_MARK`	This refers to NRZI-Mark encoding.	
`SIO4_USC_DATA_ENCODE_NRZI_SPACE`	This refers to NRZI-Space encoding.	

4.10.20.8. SIO4_IOCTL_USC_TX_ENABLE

This service configures the USC transmitter enabled state.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TX_ENABLE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_ENABLE_NO_AFTER`	This refers to the transmitter being disabled after sending of the current character, if one is being sent.	
`SIO4_USC_ENABLE_NO_NOW`	This refers to the transmitter being disabled immediately.	
`SIO4_USC_ENABLE_YES_NOW`	This refers to the transmitter being enabled immediately.	
`SIO4_USC_ENABLE_YES_W_AE`	This refers to the transmitter being enabled immediately along with the hardware flow control signals.	

4.10.20.9. SIO4_IOCTL_USC_TX_IDLE_COND

This service configures the USC transmitter output on the Tx Data signal when no data is available to send.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TX_IDLE_COND`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_TX_IDLE_COND_0`	The Tx Data signal outputs a continuous data “0” value.	

`SIO4_USC_TX_IDLE_COND_0_1`	The Tx Data signal is alternately data “0” and data “1” values.
`SIO4_USC_TX_IDLE_COND_1`	The Tx Data signal outputs a continuous data “1” value.
`SIO4_USC_TX_IDLE_COND_DEFAULT`	This refers to the default, which varies with each protocol.
`SIO4_USC_TX_IDLE_COND_MARK`	The Tx Data signal is held high.
`SIO4_USC_TX_IDLE_COND_MARK_SPACE`	The Tx Data signal is alternately driven with the high then low.
`SIO4_USC_TX_IDLE_COND_SPACE`	The Tx Data signal is held low.

4.10.20.10. SIO4_IOCTL_USC_TX_MODE

This service configures the protocol used by the USC transmitter for data transmission.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TX_MODE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_MODE_8023`	This refers to the 802.3 protocol.	
`SIO4_USC_MODE_ASY_CV`	This refers to the Asynchronous with Code Violations protocol.	
`SIO4_USC_MODE_ASYNC`	This refers to the Asynchronous protocol.	
`SIO4_USC_MODE_BSC`	This refers to the Bisynchronous Serial Communications protocol.	
`SIO4_USC_MODE_HDLC`	This refers to the HDLC protocol.	
`SIO4_USC_MODE_HDLC_L`	This refers to the HDLC Loop protocol.	
`SIO4_USC_MODE_ISOC`	This refers to the Isochronous protocol.	
`SIO4_USC_MODE_MONO`	This refers to the Monosync protocol.	
`SIO4_USC_MODE_NBIP`	This refers to the Nine-Bit Interprocessor Protocol.	
`SIO4_USC_MODE_S_MONO`	This refers to the Slaved Monosync protocol.	
`SIO4_USC_MODE_TBSC`	This refers to the Transparent Bisynchronous Serial Communications protocol.	

4.10.20.11. SIO4_IOCTL_USC_TX_PREAMBLE_FLAG

This service configures the Preamble Flag output selection for the transmitter.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TX_PREAMBLE_FLAG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_TX_PREAMBLE_FLAG_NO`	This refers to sending no Preamble Flag.	
`SIO4_USC_TX_PREAMBLE_FLAG_YES`	This refers to sending a Preamble Flag.	

4.10.20.12. SIO4_IOCTL_USC_TX_PREAMBLE_LEN

This service configures the Preamble length selection for the transmitter.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_TX_PREAMBLE_LEN`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_TX_PREAMBLE_LEN_8_BITS`	This refers to sending a Preamble length of 8-bits.	
`SIO4_USC_TX_PREAMBLE_LEN_16_BITS`	This refers to sending a Preamble length of 16-bits.	
`SIO4_USC_TX_PREAMBLE_LEN_32_BITS`	This refers to sending a Preamble length of 32-bits.	
`SIO4_USC_TX_PREAMBLE_LEN_64_BITS`	This refers to sending a Preamble length of 64-bits.	

4.10.20.13. SIO4_IOCTL_USC_TX_PREAMBLE_PAT

This service configures the Preamble pattern selection for the transmitter.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_TX_PREAMBLE_PAT`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_TX_PREAMBLE_PAT_0`	This refers to continuous low output.	
`SIO4_USC_TX_PREAMBLE_PAT_01`	This refers to continuous low then high pattern.	
`SIO4_USC_TX_PREAMBLE_PAT_1`	This refers to continuous low high.	
`SIO4_USC_TX_PREAMBLE_PAT_10`	This refers to continuous high then low pattern.	

4.10.20.14. SIO4_IOCTL_USC_TX_STATUS

This service retrieves the USC transmitter status, which is essentially the value read from the lower byte of the Transmit Command/Status Register.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_USC_TX_STATUS`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFF`	This is the valid range for this service.	

4.10.20.15. SIO4_IOCTL_USC_TX_WAIT_DMA_TRIG

This service configures the USC transmitter's pausing of data transfer from the channel's main Tx FIFO.

> **NOTE**: By default, each channel is configured to automatically transfer data from the channel's main Tx FIFO as data is received over the PCI bus. This service deals with the pausing of such data transfer until commanded to resume the transfer.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TX_WAIT_DMA_TRIG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_WAIT_DMA_TRIG_NO`	This refers to data transfer being unimpeded.	
`SIO4_USC_WAIT_DMA_TRIG_YES`	This refers to data transfer being paused.	

4.10.20.16. SIO4_IOCTL_USC_TX_WAIT_UNDERRUN

This service configures the USC transmitter's Wait On Underrun feature.

Usage

Argument	Description
`request`	`SIO4_IOCTL_USC_TX_WAIT_UNDERRUN`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_USC_TX_WAIT_UNDERRUN_NO`	This refers to not waiting.	
`SIO4_USC_TX_WAIT_UNDERRUN_YES`	This refers to waiting.	

4.11. SYNC Model Specific IOCTL Services

These IOCTL services relate to SIO4-SYNC model boards only.

> **NOTE**: All services whose argument data type is `s32*` accept the value of `-1` being passed to the driver. If there is a setting associated with the service, then passing in the value `-1` is a request for the current setting. If the service is supported, then the current setting is returned. If the service is not supported, then the value returned by the driver will be `-1`. If there isn't a setting associated with the service, then passing in the value `-1` is a request to determine if the service is supported.

If the service is supported, then the value zero is returned. If the service is not supported, then the value `-1` is returned.

4.11.1. SYNC Model Legacy Cable Interface Specific IOCTL Services

These IOCTL services relate to legacy cable operating services for SIO4-SYNC model boards only.

NOTE: These services are available only when the legacy cable configuration is supported by firmware. Please also read Cable Configuration Modes (section 8.6, page 137).

4.11.1.1. SIO4_IOCTL_SYNC_LEG_RXD_CFG

This service configures the routing of the cable Rx Data signal when using legacy cable configuration mode. Please also read Cable Configuration Modes (section 8.6, page 137).

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_LEG_RXD_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The feature is unsupported or inaccessible.
`SIO4_SYNC_LEG_RXD_CFG_LOW`	Connect to the signal at the lower cable portion.	
`SIO4_SYNC_LEG_RXD_CFG_TRI`	Tri-state the cable signal.	
`SIO4_SYNC_LEG_RXD_CFG_UP`	Connect to the signal at the upper cable portion.	

4.11.1.2. SIO4_IOCTL_SYNC_LEG_TXD_CFG

This service configures the routing of the cable Tx Data signal when using legacy cable configuration mode. Please also read Cable Configuration Modes (section 8.6, page 137).

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_LEG_TXD_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The feature is unsupported or inaccessible.
`SIO4_SYNC_LEG_TXD_CFG_LOW`	Connect to the signal at the lower cable portion.	
`SIO4_SYNC_LEG_TXD_CFG_TRI`	Tri-state the cable signal.	
`SIO4_SYNC_LEG_TXD_CFG_UP`	Connect to the signal at the upper cable portion.	
`SIO4_SYNC_LEG_TXD_CFG_UP_LOW`	Connect to the signal at the upper and lower cable portions.	

4.11.2. SYNC Model Miscellaneous IOCTL Services

4.11.2.1. SIO4_IOCTL_SYNC_MODE

This service configures the SYNC Mode's operating mode.

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_MODE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_MODE_DUAL`	This refers to the dual mode of operation.	
`SIO4_SYNC_MODE_NORMAL`	This refers to the normal mode of operation.	

4.11.3. SYNC Model Receiver Specific IOCTL Services

These IOCTL services configure the operation of the receiver.

4.11.3.1. SIO4_IOCTL_SYNC_RX_BIT_COUNT

This service retrieves the current received Bit Count.

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_RX_BIT_COUNT`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFFFF`	NA	This is the valid range for this service.

4.11.3.2. SIO4_IOCTL_SYNC_RX_BIT_ORDER

This service configures the order in which data bits are received over the cable interface.

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_RX_BIT_ORDER`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_BIT_ORDER_LSB`	This refers to the Least Significant Bit arriving first.	
`SIO4_SYNC_BIT_ORDER_MSB`	This refers to the Most Significant Bit arriving first.	

4.11.3.3. SIO4_IOCTL_SYNC_RX_COUNT_ERROR

This service reports the occurrence of an Rx Bit Count error.

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_RX_COUNT_ERROR`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_RX_COUNT_ERROR_NO`	N/A	There was not an error.
`SIO4_SYNC_RX_COUNT_ERROR_YES`	N/A	There was an error.

4.11.3.4. SIO4_IOCTL_SYNC_RX_COUNT_RESET

This service clears the Rx Bit Count to zero.

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_RX_COUNT_RESET`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests support information.	The service is unsupported.
`0`	Reset the count.	The service is supported or the count was reset.

4.11.3.5. SIO4_IOCTL_SYNC_RX_ENABLE

This service enables and disables the receiver.

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_RX_ENABLE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_ENABLE_NO`	Disable the receiver.	
`SIO4_SYNC_ENABLE_YES`	Enable the receiver.	

4.11.3.6. SIO4_IOCTL_SYNC_RX_GAP_ENABLE

This service enables and disables the Rx Gap feature.

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_RX_GAP_ENABLE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_RX_GAP_ENABLE_NO`	Disable the feature.	
`SIO4_SYNC_RX_GAP_ENABLE_YES`	Enable the feature.	

4.11.4. SYNC Model Rx Clock Specific IOCTL Services

These IOCTL services configure the operation of the Rx Clock signal.

4.11.4.1. SIO4_IOCTL_SYNC_RXC_CFG

This service configures the polarity of the Rx Clock cable signal.

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_RXC_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_RXC_CFG_FALL_EDGE`	Clock in data on the falling edge.	
`SIO4_SYNC_RXC_CFG_RISE_EDGE`	Clock in data on the rising edge.	

4.11.4.2. SIO4_IOCTL_SYNC_RXC_POL

This service configures the polarity of the Rx Clock signal on which data bits are clocked into the receiver.

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_RXC_POL`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_CLOCK_POL_FALL`	Data is clocked in on the clock's falling edge.	
`SIO4_SYNC_CLOCK_POL_RISE`	Data is clocked in on the clock's rising edge.	

4.11.5. SYNC Model Rx Data Specific IOCTL Services

These IOCTL services configure the operation of the Rx Data signal.

4.11.5.1. SIO4_IOCTL_SYNC_RXD_CFG

This service configures the polarity of the Rx Data cable signal.

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_RXD_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_RXD_CFG_ACTIVE_HI`	This refers to the Active High configuration.	
`SIO4_SYNC_RXD_CFG_ACTIVE_LO`	This refers to the Active Low configuration.	

4.11.6. SYNC Model Rx Envelope Specific IOCTL Services

These IOCTL services configure the operation of the Rx Envelope signal.

4.11.6.1. SIO4_IOCTL_SYNC_RXE_CFG

This service configures the Rx Envelope cable signal.

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_RXE_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_RXE_CFG_ACTIVE_HI`	This refers to the Active High configuration.	
`SIO4_SYNC_RXE_CFG_ACTIVE_LO`	This refers to the Active Low configuration.	
`SIO4_SYNC_RXE_CFG_DISABLE`	This disables the signal.	

4.11.6.2. SIO4_IOCTL_SYNC_RXE_POL

This service configures the polarity of the Rx Envelope cable signal.

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_RXE_POL`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_ENV_POL_ACTIVE_HI`	This refers to the Active High configuration.	
`SIO4_SYNC_ENV_POL_ACTIVE_LO`	This refers to the Active Low configuration.	

4.11.7. SYNC Model Transmitter Specific IOCTL Services

These IOCTL services configure the operation of the transmitter.

4.11.7.1. SIO4_IOCTL_SYNC_TX_BIT_ORDER

This service configures the order in which data bits are transmitted over the cable interface.

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_TX_BIT_ORDER`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_BIT_ORDER_LSB`	This refers to the Least Significant Bit being transmitted first.	
`SIO4_SYNC_BIT_ORDER_MSB`	This refers to the Most Significant Bit being transmitted first.	

4.11.7.2. SIO4_IOCTL_SYNC_TX_ENABLE

This service enables and disables the transmitter.

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_TX_ENABLE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_ENABLE_NO`	Disable the transmitter.	
`SIO4_SYNC_ENABLE_YES`	Enable the transmitter.	

4.11.7.3. SIO4_IOCTL_SYNC_TX_GAP_SIZE

This service configures the length of the gap between successive Envelope active periods.

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_TX_GAP_SIZE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFFFF`	This is the valid range for this service.	

4.11.7.4. SIO4_IOCTL_SYNC_TX_WORD_SIZE

This service configures the Word Size for the data to be transmitted.

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_TX_WORD_SIZE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0xFFFF`	This is the valid range for this service.	

4.11.8. SYNC Model Tx Auxiliary Clock Specific IOCTL Services

These IOCTL services configure the operation of the Tx Auxiliary Clock signal.

4.11.8.1. SIO4_IOCTL_SYNC_TXAUXC_CFG

This service configures the Tx Auxiliary Clock signal at the cable interface.

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_TXAUXC_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_TXAUXC_CFG_OSC_HALF`	This refers to the on-board oscillator divided by two.	
`SIO4_SYNC_TXAUXC_CFG_OUT_0`	This refers to the signal being driven low.	
`SIO4_SYNC_TXAUXC_CFG_OUT_1`	This refers to the signal being driven high.	
`SIO4_SYNC_TXAUXC_CFG_TRI`	This refers to the signal being tri-stated.	

4.11.9. SYNC Model Tx Clock Specific IOCTL Services

These IOCTL services configure the operation of the Tx Clock signal.

4.11.9.1. SIO4_IOCTL_SYNC_TXC_CFG

This service configures the source selection for the Tx Clock signal.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_SYNC_TXC_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_TXC_CFG_EXT`	This refers to an external clock source.	
`SIO4_SYNC_TXC_CFG_INT`	This refers to the on-board clock source.	

4.11.9.2. SIO4_IOCTL_SYNC_TXC_IDLE

This service configures the operation of the Tx Clock signal when no data is being transmitted. (This service is unavailable when the `SIO4_IOCTL_SYNC_TXC_IDLE_CFG` service is available.)

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_SYNC_TXC_IDLE`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_TXC_IDLE_NO`	No, the clock is not to be idle. The clock will be active.	
`SIO4_SYNC_TXC_IDLE_YES`	Yes, the clock is to be idle. The clock will be idle.	

4.11.9.3. SIO4_IOCTL_SYNC_TXC_IDLE_CFG

This service configures the operation of the Tx Clock signal when no data is being transmitted. (This service is unavailable when the `SIO4_IOCTL_SYNC_TXC_IDLE` service is available.)

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_SYNC_TXC_IDLE_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_TXC_IDLE_CFG_ACTIVE`	This refers to the clock being fully active.	

`SIO4_SYNC_TXC_IDLE_CFG_IDLE_0`	This refers to the signal being idle, or when supported, driven low.
`SIO4_SYNC_TXC_IDLE_CFG_IDLE_1`	This refers to the signal being driven high. If the board does not support this option, then the *`IDLE_0`* option is used.

4.11.9.4. SIO4_IOCTL_SYNC_TXC_POL

This service configures the polarity of the Tx Clock signal on which data bits are clocked out of the transmitter.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_SYNC_TXC_POL`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_CLOCK_POL_FALL`	Data is clocked out on the clock's falling edge.	
`SIO4_SYNC_CLOCK_POL_RISE`	Data is clocked out on the clock's rising edge.	

4.11.9.5. SIO4_IOCTL_SYNC_TXC_SRC

This service configures the source selection for the Tx Clock signal.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_SYNC_TXC_SRC`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_TXC_SRC_0`	This refers to the signal being driven low.	
`SIO4_SYNC_TXC_SRC_1`	This refers to the signal being driven high.	
`SIO4_SYNC_TXC_SRC_EXT_FALL`	This refers to the falling edge of the external Tx Clock signal.	
`SIO4_SYNC_TXC_SRC_EXT_RISE`	This refers to the rising edge of the external Tx Clock signal.	
`SIO4_SYNC_TXC_SRC_OSC_HALF_FALL`	This refers to the falling edge of the on-board clock (divided by two).	
`SIO4_SYNC_TXC_SRC_OSC_HALF_RISE`	This refers to the rising edge of the on-board clock (divided by two).	

4.11.10. SYNC Model Tx Data Specific IOCTL Services

These IOCTL services configure the operation of the Tx Data signal.

4.11.10.1. SIO4_IOCTL_SYNC_TXD_CFG

This service configures the operation of the Tx Data cable signal.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_SYNC_TXD_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_TXD_CFG_OUT_0`	This refers to the signal being driven low.	
`SIO4_SYNC_TXD_CFG_OUT_1`	This refers to the signal being driven high.	
`SIO4_SYNC_TXD_CFG_ACTIVE_HI`	This refers to the Active High configuration.	
`SIO4_SYNC_TXD_CFG_ACTIVE_LO`	This refers to the Active Low configuration.	

4.11.10.2. SIO4_IOCTL_SYNC_TXD_IDLE_CFG

This service configures the operation of the Tx Data cable signal while no data is being transmitted.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_SYNC_TXD_IDLE_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_TXD_IDLE_CFG_OUT_0`	This refers to the signal being driven low.	
`SIO4_SYNC_TXD_IDLE_CFG_OUT_1`	This refers to the signal being driven high.	

4.11.11. SYNC Model Tx Envelope Specific IOCTL Services

These IOCTL services configure the operation of the Tx Envelope signal.

4.11.11.1. SIO4_IOCTL_SYNC_TXE_CFG

This service configures the operation of the Tx Envelope cable signal.

Usage

Argument	**Description**
`request`	`SIO4_IOCTL_SYNC_TXE_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_TXE_CFG_ACTIVE_HI`	This refers to the Active High configuration.	
`SIO4_SYNC_TXE_CFG_ACTIVE_LO`	This refers to the Active Low configuration.	
`SIO4_SYNC_TXE_CFG_OUT_0`	This refers to the signal being driven low.	
`SIO4_SYNC_TXE_CFG_OUT_1`	This refers to the signal being driven high.	

4.11.11.2. SIO4_IOCTL_SYNC_TXE_POL

This service configures the polarity of the Tx Envelope cable signal.

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_TXE_POL`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_ENV_POL_ACTIVE_HI`	Data is transmitted while the envelope signal is high.	
`SIO4_SYNC_ENV_POL_ACTIVE_LO`	Data is transmitted while the envelope signal is low.	

4.11.12. SYNC Model Tx Spare Specific IOCTL Services

These IOCTL services configure the operation of the Tx Spare signal.

4.11.12.1. SIO4_IOCTL_SYNC_TXSP_CFG

This service configures the operation of the Tx Spare cable signal.

Usage

Argument	Description
`request`	`SIO4_IOCTL_SYNC_TXSP_CFG`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_SYNC_TXSP_CFG_DISABLE`	This refers to the signal being disabled.	
`SIO4_SYNC_TXSP_CFG_INPUT`	This refers to the signal operating as an input.	
`SIO4_SYNC_TXSP_CFG_OUT_0`	This refers to the signal being driven low.	
`SIO4_SYNC_TXSP_CFG_OUT_1`	This refers to the signal being driven high.	

4.12. SIO4A Model Specific IOCTL Services

These IOCTL services are specific to the SIO4A versions of the board.

> **NOTE**: All services whose argument data type is `s32*` accept the value of `-1` being passed to the driver. If there is a setting associated with the service, then passing in the value `-1` is a request for the current setting. If the service is supported, then the current setting is returned. If the service is not supported, then the value returned by the driver will be `-1`. If there isn't a setting associated with the service, then passing in the value `-1` is a request to determine if the service is supported. If the service is supported, then the value zero is returned. If the service is not supported, then the value `-1` is returned.

4.12.1. SIO4_IOCTL_GPIO_DIRECTION_OUT

This service configures the direction of the General Purpose I/O pins.

Usage

Argument	Description
`request`	`SIO4_IOCTL_GPIO_DIRECTION_OUT`
`arg`	`s32*`

The table below lists the options used with this service. If a bit is set, then the pin is an output. If a bit is clear, then the pin is an input.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0x3F`	This is the valid range for this service.	

4.12.2. SIO4_IOCTL_GPIO_INPUT_LATCHING

This service configures the General Purpose I/O input pins as latching or non-latching.

Usage

Argument	Description
`request`	`SIO4_IOCTL_GPIO_INPUT_LATCHING`
`arg`	`s32*`

The table below lists the options used with this service. If a bit is set, then the pin is a latching when it is an input. If a bit is clear, then the pin is non-latching when it is an input.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0x3F`	This is the valid range for this service.	

4.12.3. SIO4_IOCTL_GPIO_INPUT_READ

This service returns the state of the General Purpose I/O pins configured as inputs.

Usage

Argument	Description
`request`	`SIO4_IOCTL_GPIO_INPUT_READ`
`arg`	`s32*`

The table below lists the options used with this service. Bits are set if the corresponding pins are configured as inputs and the input pins are at a high state. Bits are clear if the corresponding pins are configured as inputs and the input pins are at a low state. Bits are also clear if the corresponding pins are configured as outputs.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0x3F`	N/A	This is the valid range for this service.

4.12.4. SIO4_IOCTL_GPIO_OUTPUT_WRITE

This service sets the state of the General Purpose I/O pins configured as outputs.

Usage

Argument	Description
`request`	`SIO4_IOCTL_GPIO_OUTPUT_WRITE`
`arg`	`s32*`

The table below lists the options used with this service. If a bit is set and configured as an output, then the cable pin is driven high. If a bit is clear and configured as an output, then the cable pin is driven low. Cable pins configured as inputs are unaffected.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`0` to `0x3F`	This is the valid range for this service.	

4.12.5. SIO4_IOCTL_GPIO_POLARITY

This service configures the latching polarity of the General Purpose I/O pins configured as inputs.

Usage

Argument	Description
`request`	`SIO4_IOCTL_GPIO_POLARITY`
`arg`	`s32*`

The table below lists the options used with this service.

Values	Passed to Driver	Returned by Driver
`-1`	Requests current setting.	The service is unsupported.
`SIO4_GPIO_POLARITY_HIGH`	Latch on a level-high or a high going edge.	
`SIO4_GPIO_POLARITY_LOW`	Latch on a level-low or a low going edge.	

4.12.6. SIO4_IOCTL_GPIO_SENSE_EDGE

This service sets the latching sense of the General Purpose I/O pins configured as latching inputs.

Usage

Argument	Description
`request`	`SIO4_IOCTL_GPIO_SENSE_EDGE`
`arg`	`s32*`

The table below lists the options used with this service. If a bit is set and the corresponding pin is configured as a latching input, then the latch responds to changes in cable signal state. If a bit is clear and the corresponding pin is configured as a latching input, then the latch responds to the level of the cable signal state. Cable pins configured as non-latching inputs or as outputs are unaffected.

Values	**Passed to Driver**	**Returned by Driver**
`-1`	Requests current setting.	The service is unsupported.
`0` to `0x3F`	This is the valid range for this service.	

5. The Driver

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

5.1. Files

The driver is built into an OS specific executable. The device driver files are summarized in the following table. Some source files are specific to the SIO4, some are specific only to the OS and some are SIO4 and OS independent.

Description	**Files**	**Location**	**OS**
Source Files	`*.c`, `*.h`	…`/driver/`	All
Header File	`sio4.h`	…`/driver/`	All
Driver File	`sio4.ko` †	…`/driver/`	Linux (kernels version 2.6 and later)
	`sio4.o` †	…`/driver/`	Linux (kernels version 2.4 and earlier)
	`sio4.rta` ‡	…`\driver\`	INtime
	`sio4.rtss`	…`\driver\`	RTX *

† The Linux run time executable is implemented as a loadable kernel module.
‡ The INtime run time executable is implemented as an INtime executable.
* The RTX run time executable is implemented as an RTX executable.

5.2. Build

For instructions on building the driver refer to this same section number in the OS specific SIO4 driver user manual.

5.3. Startup

For instructions on starting the driver executable refer to this same section number in the OS specific SIO4 driver user manual.

5.4. Verification

For instructions on verifying that the driver has been loaded and is running refer to this same section number in the OS specific SIO4 driver user manual.

5.5. Version

For instructions on obtaining the driver version number refer to this same section number in the OS specific SIO4 driver user manual.

5.6. Shutdown

For instructions on terminating the driver executable refer to this same section number in the OS specific SIO4 driver user manual.

6. Document Source Code Examples

The source code examples included in this document are built into a static library usable with SIO4 applications. The purpose of these files is to verify that the documentation samples compile and to provide a library of working sample code to assist in a user's learning curve and application development effort.

6.1. Files

The library files are summarized in the table below.

Description	**Files**	**Location**	**OS**
Source Files	`*.c`, `*.h` …	…`/docsrc/`	All
Header File	`sio4_dsl.h`	…`/include/`	All
Library File	`sio4_dsl.a`	…`/lib/`	Linux
	`sio4_dsl.lib`	…`\lib\`	INtime
	`sio4_dsl.lib`	…`\lib\`	RTX

6.2. Build

For library build instructions refer to this same section number in the OS specific SIO4 driver user manual.

6.3. Library Use

For library usage information refer to this same section number in the OS specific SIO4 driver user manual.

7. Utilities Source Code

The API Library installation includes a body of utility source code designed to aid in the understanding and use of the interface calls and IOCTL services. Utility sources are also included for device independent and common, general-purpose services. Most of the utilities are implemented as visual wrappers around the corresponding services to facilitate structured console output for the sample applications. The utility sources are compiled and linked into static libraries to simplify their use. An additional purpose of these utility services is to provide a library of working sample code to assist in a user's learning curve and application development effort.

For each API function there is a corresponding utility source file with a corresponding utility service. As an example, for the API function `sio4_open()` there is the utility file `open.c` containing the utility function `sio4_open_util()`. The naming pattern is as follows: API function `sio4_xxxx()`, utility file name `xxxx.c`, utility function `sio4_xxxx_util()`. Additionally, for each IOCTL code there is a corresponding utility source file with a corresponding utility service. As an example, for IOCTL code `SIO4_IOCTL_QUERY` there is the utility file `util_query.c` containing the utility function `sio4_query()`. The naming pattern is as follows: IOCTL code `SIO4_IOCTL_XXXX`, utility file name `util_xxxx.c`, utility function `sio4_xxxx()`.

7.1. Files

The utility files are summarized in the table below.

Description	**Files**	**Location**	**OS**
Source Files	`*.c`, `*,h`, `makefile` …	…`/utils/`	All
Header File	`sio4_utils.h`	…`/include/`	All

Library Files	`sio4_utils.a`
	`gsc_utils.a`
	`os_utils.a`
	`plx_utils.a`
		…`/lib/`	Linux

	`sio4_utils.lib`
	`gsc_utils.lib`
	`os_utils.lib`
	`plx_utils.lib`
		…`\lib\`	INtime

	`sio4_utils.lib`
	`gsc_utils.lib`
	`os_utils.lib`
	`plx_utils.lib`
		…`\lib\`	RTX

7.2. Build

For library build instruction refer to this same section number in the OS specific SIO4 driver user manual.

7.3. Library Use

For library usage information refer to this same section number in the OS specific SIO4 driver user manual.

8. Operating Information

This section explains some basic operational procedures for using the SIO4. This is in no way intended to be a comprehensive guide. This is simply to address a very few issues relating to their use. For additional operating information refer to this same section number in the OS specific *SIO4 Driver User Manual*.

8.1. Debugging Aids

The driver package includes the following items useful for development and/or debugging aids.

8.1.1. Device Identification

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

Description	File	Location	OS
Application	`id`	…`/id/`	Linux
	`id.rta`	…`\id\`	INtime
	`id.exe`	…`\id\`	RTX

8.1.2. Detailed Register Dump

Among the utility services provided is a function to generate a detailed listing of device registers to the console. When used, the function is typically used to verify device configuration. In these cases, the function should be called after complete device configuration and before the first I/O call. When intended for sending to GSC tech support, please set the *detail* arguments to `1`. The function arguments are as follows. The utility location is given in the subsequent table.

Argument	Description
`fd`	This is the file descriptor used to access the device.
`gsc`	If non-zero the firmware registers will be displayed.
`gsc_detail`	If non-zero the register dump will include details of each register field.
`usc`	If non-zero the USC registers will be displayed.
`usc_detail`	If non-zero the USC register dump will include details of each register field.

Description	File/Name	Location	OS
Function	`sio4_reg_list()`	Source File	ALL
Source File	`util_reg.c`	…`/utils/`	ALL
Header File	`sio4_utils.h`	…`/include/`	ALL
Library File	`sio4_utils.a`	…`/lib/`	Linux
	`sio4_utils.lib`	…`\lib\`	INtime
	`sio4_utils.lib`	…`\lib\`	RTX

If needed a similar utility service is available for printing the programmable oscillator registers. The function name is `sio4_reg_list_cy22393()`, which is located per the above table.

8.2. Data Transfer Modes

The following describes the three supported I/O modes used for data transfer between the host and the SIO4. All three modes are available using the routines `sio4_read()` and `sio4_write()`. Applications select the desired mode using IOCTL services. Use the `SIO4_IOCTL_TX_IO_MODE` IOCTL service (section 4.8.3.9, page 39) to configure the `sio4_write()` data transfer mode and `SIO4_IOCTL_RX_IO_MODE` (section 4.8.3.2, page 37) to configure the `sio4_read()` data transfer mode.

The driver's effort to fulfill I/O requests involves a repetitive, two-step process. The first step is to determine how much data can be moved at that specific instant. The second step is to perform the actual transfer. The method used in step one depends on the I/O mode (PIO, BMDMA, DMDMA) and the data direction (read or write). Step two depends on the I/O mode selection and on the results of the first step. The effect of this repetitive process is that application I/O requests may be subdivided into multiple smaller transfers.

> **NOTE**: When using either DMA mode, the driver automatically reverts to PIO for transfer requests that are less than 8 bytes, regardless of the corresponding PIO Threshold setting (Rx: section 4.8.3.5, page 38, Tx: section 4.8.3.12, page 40). However, this check is disabled if the corresponding PIO Threshold setting or DMA Threshold settings (Rx: section 4.8.3.2, page 37, Tx: section 4.8.3.9, page 39) is zero.

8.2.1. PIO - Programmed I/O

In this mode data is transferred using repetitive register accesses. This is most applicable for low throughput requirements or for small transfer requests. The driver continues the operation until either the I/O request is fulfilled or the I/O timeout expires, whichever occurs first. This is generally the least efficient mode, but for very small transfers it is more efficient than DMA.

8.2.2. BMDMA - Block Mode DMA

This mode transfers data with little CPU overhead, but limits transfers to at most the size of the installed FIFOs. An I/O timeout during a BMDMA transfer rarely results in data loss. Since an SIO4 can have up to eight data streams (four Tx and four Rx) but has only two DMA engines, the driver assigns a DMA engine to a stream only while it is needed. After a DMA transfer has completed, the driver releases the DMA engine for use by the other streams.

> **NOTE:** When three or more streams require use of the DMA engines at the same instant, the third and subsequent streams must poll for access. This is done by repeated attempts of acquiring access, which is controlled, checking for availability, then releasing access. Once a DMA engine is released for reuse, it is granted to the very next thread of execution that just happens to check for availability.

8.2.3. DMDMA - Demand Mode DMA

This is the most efficient option. It accommodates transfers that exceed the size of the installed FIFOs and uses the FIFO fill level to throttle data movement over the PCI bus. This permits efficient data movement over the PCI bus and also permits the transfer to remain active while data is being transferred over the cable interface. When using this mode, the driver typically performs a single DMDMA for each subdivided transfer. An I/O timeout during a DMDMA transfer normally results in data loss. Since an SIO4 can have up to eight data streams (four Tx and four Rx) but has only two DMA engines, applications can have only two DMDMA I/O requests active at the same time.

> **NOTE**: Applications should not initiate three or more simultaneous Demand Mode DMA based I/O transfers. The third and subsequent transfers will most certainly fail because they won't be able to gain access to a DMA engine until one of the first two transfers goes to completion. The typical result is data loss on the third and subsequent data streams.
>
> **NOTE**: Applications should avoid initiating Block Mode DMA transfers while two simultaneous Demand Mode DMA transfer requests are active. The BMDMA transfers will most certainly fail because they won't be able to gain access to a DMA engine until one of the DMDMA transfers goes to completion. The typical result is data loss on the BMDMA data streams.

8.3. PIO Threshold

The PIO Threshold mechanism was implemented to improve overall performance because PIO can achieve higher throughput than DMA when the data transfers are rather small. The configured threshold is evaluated both for the overall transfer and for each subdivided transfer. If the size of any such transfer is equal to or less than the configured threshold, then the driver will override the configured I/O Mode selection and use PIO instead.

> **NOTE**: While the PIO Threshold setting can be set as low as zero, the driver will unconditionally use PIO for any request under eight bytes.

8.4. DMA Threshold

At intermediate baud rates and below, the subdivided transfer sizes (see above) can often result in PIO being used for entire I/O requests in spite of DMA being the selected as the configured I/O Mode. The result can be a significant reduction in I/O efficiency. The DMA Threshold mechanism was implemented to address the drop in efficiency observed at lower bit rates. The mechanism works by reporting the calculated availability count as zero if 1) the actual availability count doesn’t fulfill the I/O request, 2) the actual availability count is less than the configured DMA Threshold and 3) the I/O stream doesn’t appear to be idle at the cable interface. Otherwise, the calculated availability count is reported as is.

> **NOTE**: The DMA Threshold settings apply to all Block Mode DMA transfers. The settings apply to Demand Mode DMA transfers only if the corresponding I/O Timeout setting is zero.

> **NOTE**: No matter how many streams use DMA, even just one, efficiency may be improved by careful adjustment of the DMA Threshold settings (section 4.8.3.2, page 37, and section 4.8.3.9, page 39). In general, the higher the setting the better, but it should not exceed the level corresponding to the FIFO size minus two or more system timer tick’s worth of data. If this level is exceeded for read operations, then there is a risk of data loss due to an Rx FIFO overrun. If the level is exceeded for write operations, then there is an increased chance of the transmitter becoming idle. The best DMA Threshold setting can depend on the serial protocol in use. Refer to the appropriate protocol API Library for additional information. Careful testing may be required to determine the best setting.

8.5. Onboard DMA with the USC

For Zilog based SIO4 boards, the SIO4 is designed to automatically transfer data between the USC and the channel FIFOs. This is done using built-in USC hardware. This feature is configured automatically when using the Initialize IOCTL service (`SIO4_IOCTL_INITIALIZE`, section 4.8.5.1, page 44) and the USC Reset service (`SIO4_IOCTL_USC_RESET`, section 4.10.15.5, page 97). Doing this manually requires that register fields be set as follows.

Register	**Setting**
`USC.IOCR.TxRMode`	1
`USC.IOCR.RxRMode`	1
`USC.HCR.TxAMode`	1
`USC.HCR.RxAMode`	1

8.6. Cable Configuration Modes

The SIO4 supports two cable interfacing modes; DCE/DTE mode and Legacy mode. Older boards support only Legacy mode. More recent boards support only DCE/DTE mode. Intermediate boards support both modes. When both are supported selection of the active mode is governed by enabling or disabling the transceivers. This is done through the `SIO4_IOCTL_XCVR_ENABLE` service (section 4.8.1.4, page 28). If the transceiver enable option is selected then the DCE/DTE mode is active. If the transceiver disable option is selected then the Legacy mode is active.

8.6.1. DCE/DTE Mode

The DCE/DTE Mode controls cable signal routing according to the `SIO4_IOCTL_CBL_MODE` service (section 4.8.1.1, page 27). When available in firmware this feature is activated by enabling the cable transceivers (see paragraph above). DCE/DTE mode settings received by the driver are applied only if this mode is supported by the board. The driver otherwise ignores these settings.

8.6.2. Legacy Mode

The Legacy Mode of operation controls signal routing for the cable interface according to the Upper and Lower settings and a few USC settings, as described in the board user manual. When available in firmware this is the default mode of operation. When selectable this mode is activated by disabling the cable transceivers (see paragraph above). Legacy mode settings received by the driver are applied only if this mode is supported by the board. The driver otherwise ignores these settings.

8.7. Configuration Aid For The SIO4B And Later Boards

The figure below is intended as an aid to those trying to configure an SIO4B (or later) with a USC.

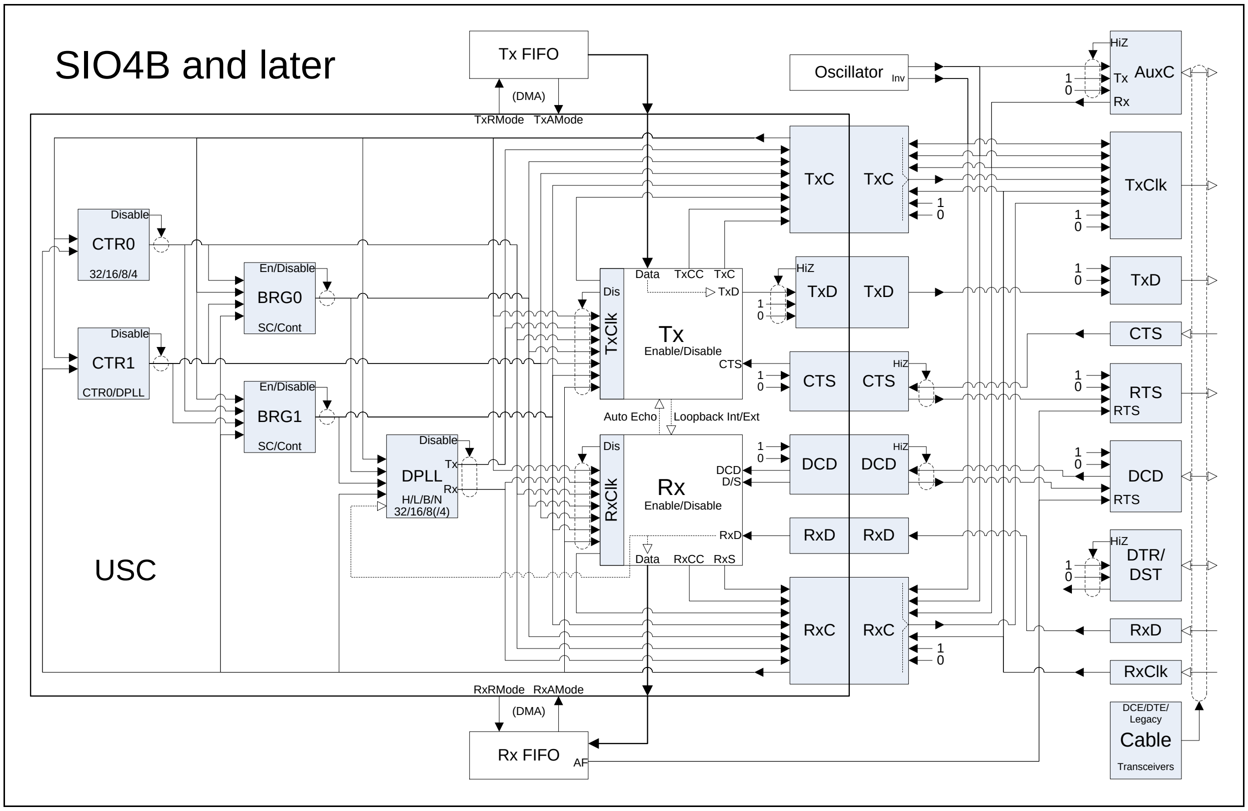

Figure 2 A configuration aid for Zilog based SIO4B and later boards.

8.8. Configuration Aid For The SIO4B-SYNC And Later Boards

The figure below is intended as an aid to those trying to configure an SIO4B-SYNC (or later).

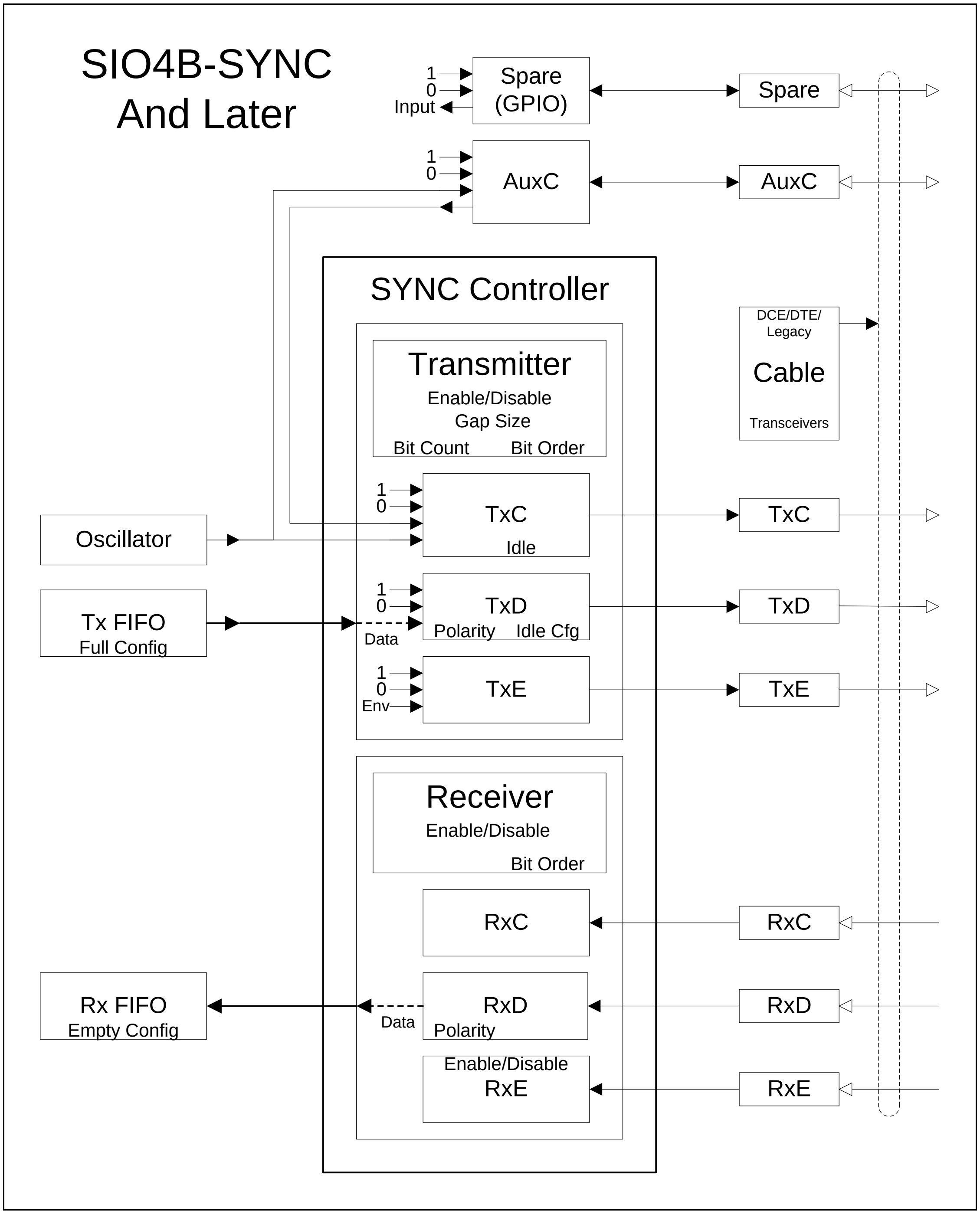

Figure 3 A configuration aid for SIO4B-SYNC and later boards.

8.9. Configuration Aid For The SIO4A Board

The figure below is intended as an aid to those trying to configure an SIO4A with a USC.

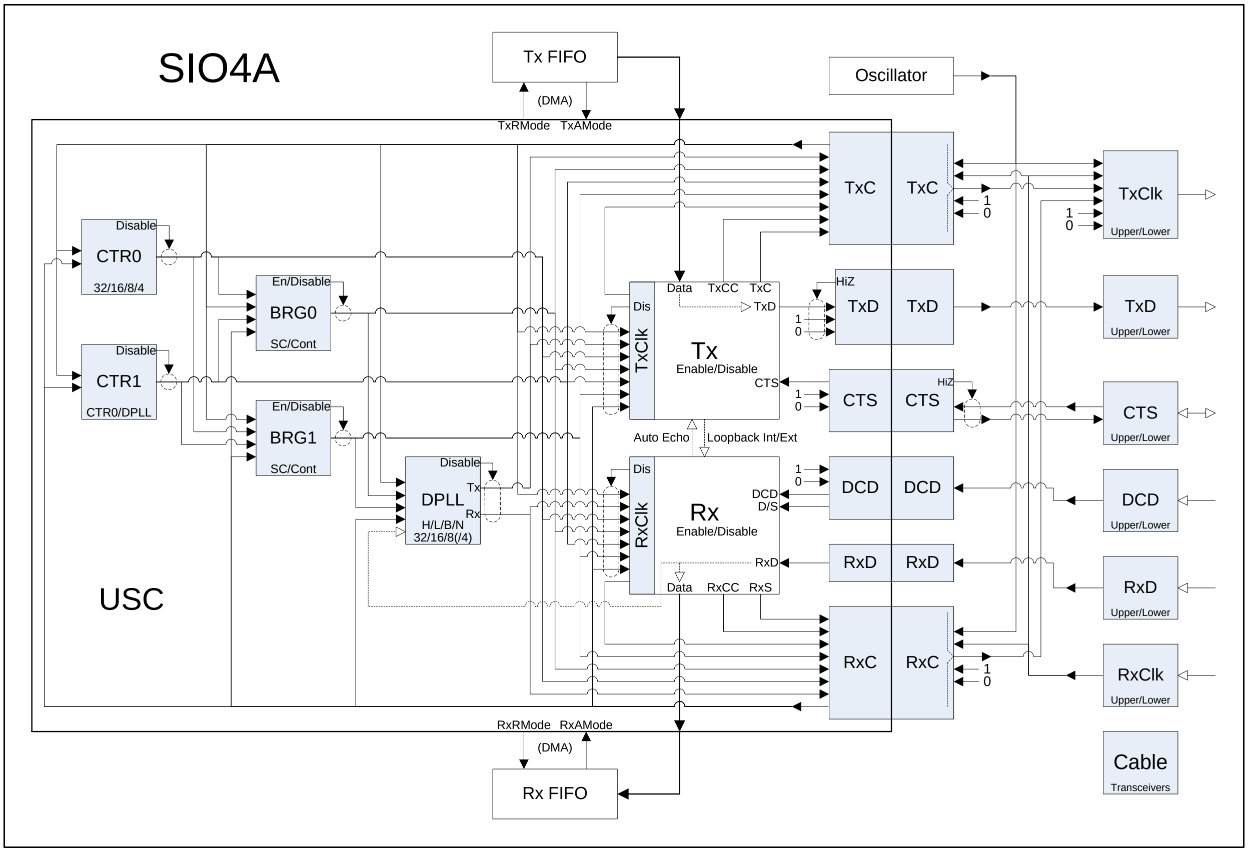

Figure 4 A configuration aid for Zilog based SIO4A boards.

8.10. Configuration Aid For The Original SIO4 Board

The figure below is intended as an aid to those trying to configure an original SIO4 with a USC.

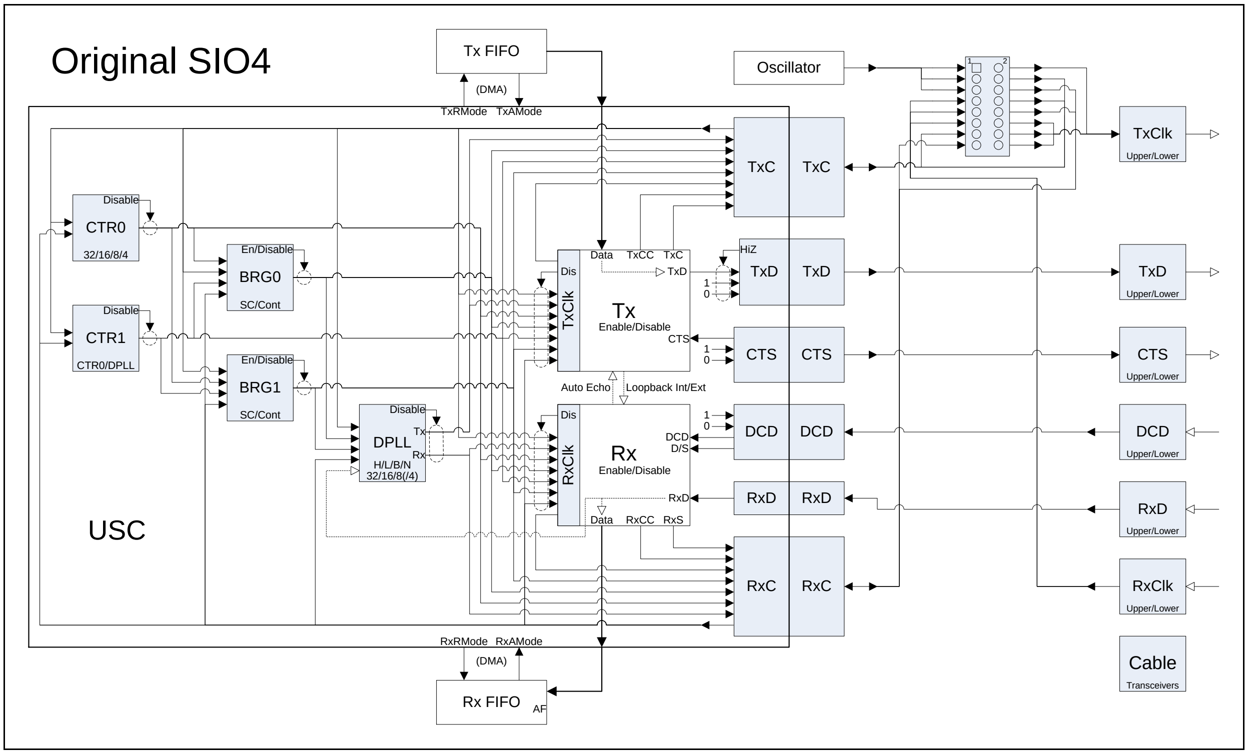

Figure 5 A configuration aid for original Zilog based SIO4 boards.

9. Sample Applications

For information on the sample applications refer to this same section number in the OS specific SIO4 driver user manual.

10. Protocol Libraries

The Protocol Libraries provide application interfaces that are tailored to the chosen serial communications protocol. This allows one to focus on use of the protocol rather than the extensive features of the SIO4. Each Protocol Library implements a set of function calls that mostly mirror those of the SIO4 API Library. In addition, each Protocol Library implements data structures designed around the specific protocol and the underlying SIO4 hardware. This allows a user to focus on the use of the protocol rather than on configuring numerous IOCTL services individually, especially when their use may be order dependent or not applicable. The Protocol Libraries are bundled in their entirety with the driver package. This includes source code and affiliated files for the statically linked Protocol Libraries, utility code, samples and documentation. The table below summarizes the Protocol Libraries. For additional information refer to the identified reference manual.

Protocol	Reference Manual File	Source Code Directory
Asynchronous	`sio4_async_rm.pdf`	…`/sio4/async/`
HDLC	`sio4_hdlc_rm.pdf`	…`/sio4/hdlc/`
Isochronous	`sio4_isoc_rm.pdf`	…`/sio4/isoc/`
SYNC	`sio4_sync_rm.pdf`	…`/sio4/sync/`

10.1. Files

The Protocol Library source files are summarized in the tables below. This first table gives the location of the files pertaining to each protocol.

Protocol	Source Location
Asynchronous	…`/async/`
HDLC	…`/hdlc/`
Isochronous	…`/isoc/`
SYNC	…`/sync/`

This table summarizes the Protocol Library header files.

Protocol	Header File	Location
Asynchronous	`sio4_async.h`	…`/include/`
HDLC	`sio4_hdlc.h`	
Isochronous	`sio4_isoc.h`	
SYNC	`sio4_sync.h`	

This table summarizes the Protocol Library static library files.

Protocol	Library File	Location	OS
Asynchronous	`sio4_async.a`	…`/lib/`	Linux
	`sio4_async.lib`	…`\lib\`	INtime
	`sio4_async.lib`	…`\lib\`	RTX
HDLC	`sio4_hdlc.a`	…`/lib/`	Linux
	`sio4_hdlc.lib`	…`\lib\`	INtime
	`sio4_hdlc.lib`	…`\lib\`	RTX
Isochronous	`sio4_isoc.a`	…`/lib/`	Linux
	`sio4_isoc.lib`	…`\lib\`	INtime
	`sio4_isoc.lib`	…`\lib\`	RTX
SYNC	`sio4_sync.a`	…`/lib/`	Linux
	`sio4_sync.lib`	…`\lib\`	INtime
	`sio4_sync.lib`	…`\lib\`	RTX

10.2. Build

For library build instruction refer to this same section number in the OS specific SIO4 driver user manual.

10.3. Library Use

For library usage information refer to this same section number in the OS specific SIO4 driver user manual.

11. Protocol Library Utilities

Each of the Protocol Libraries includes a body of utility source code designed to aid in the understanding and use of their respective APIs. The utility services provide wrappers, mostly visual, around the respective services. The aim of the visual wrappers is to facilitate structured console output for the sample applications. The utility services are used extensively by the sample applications. An additional purpose of these utility services is to provide a library of working sample code to assist in a user's learning curve and application development effort. For additional information refer to the Protocol Library reference manual as given in the previous section.

11.1. Files

The utility source files are summarized in the tables below. This first table gives the location of the utility sources.

Protocol	**Source Location**
Asynchronous	`…/async/utils/`
HDLC	`…/hdlc/utils/`
Isochronous	`…/isoc/utils/`
SYNC	`…/sync/utils/`

This table summarizes the Protocol Library utility header files.

Protocol	**Header File**	**Location**
Asynchronous	`sio4_async_util.h`	`…/include/`
HDLC	`sio4_hdlc_util.h`	
Isochronous	`sio4_isoc_util.h`	
SYNC	`sio4_sync_util.h`	

This table summarizes the Protocol Library utility static library files.

Protocol	**Library File**	**Location**	**OS**
Asynchronous	`sio4_async_util.a`	`…/lib/`	Linux
	`sio4_async_util.lib`	`…\lib\`	INtime
	`sio4_async_util.lib`	`…\lib\`	RTX
HDLC	`sio4_hdlc_util.a`	`…/lib/`	Linux
	`sio4_hdlc_util.lib`	`…\lib\`	INtime
	`sio4_hdlc_util.lib`	`…\lib\`	RTX
Isochronous	`sio4_isoc_util.a`	`…/lib/`	Linux
	`sio4_isoc_util.lib`	`…\lib\`	INtime
	`sio4_isoc_util.lib`	`…\lib\`	RTX
SYNC	`sio4_sync_util.a`	`…/lib/`	Linux
	`sio4_sync_util.lib`	`…\lib\`	INtime
	`sio4_sync_util.lib`	`…\lib\`	RTX

11.2. Build

For library build instruction refer to this same section number in the OS specific SIO4 driver user manual.

11.3. Library Use

For library usage information refer to this same section number in the OS specific SIO4 driver user manual.

12. Protocol Library Sample Applications

The Protocol Libraries include protocol specific sample applications. While they are provided without support and without any external documentation, any problems reported will be addressed as time permits. The applications are command line based and produce text output for display on a console.

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

The sample applications are located per the below table.

Protocol	**Location**
Asynchronous	…`/async/samples/`
HDLC	…`/hdlc/samples/`
Isochronous	…`/isoc/samples/`
SYNC	…`/sync/samples/`

12.1. Asynchronous

For information on the Asynchronous protocol sample applications refer to the same section number in the OS specific driver user manual.

12.2. HDLC

For information on the HDLC protocol sample applications refer to the same section number in the OS specific driver user manual.

12.3. Isochronous

For information on the Isochronous protocol sample applications refer to the same section number in the OS specific driver user manual.

12.4. SYNC

For information on the SYNC protocol sample applications refer to the same section number in the OS specific driver user manual.

Document History

Revision	**Description**
March 19, 2024	Updated to version 3.20.109.x.x. Added FASYNC (Fast Async) support. Deleted the Multi-Protocol Disable option as it is not supported by hardware.
November 16, 2023	Updated to version 3.19.106.x.x. Minor editorial updates.
June 15, 2023	Updated to version 3.18.104.x.x. Numerous, minor editorial changes. Updated the Rx and Tx FIFO Reset services.
December 13, 2022	Updated to version 3.17.101.x.x. Added RTX information. Updated the information for the open and close calls. Minor editorial updates. Added missing references for RTX specific files. Updated the programmable oscillator IOCTL service information.
June 2, 2022	Updated to version 3.16.96.x.x. Updated the description of the Tx Idle Line conditions.
February 8, 2022	Updated to version 3.15.96.x.x. Updated the DMA Threshold default information. Corrected the Tx and Rx FIFO Reset value options. Removed the RCC Software FIFO services, which are not intended for application use. Updated the content for the LED services as the corresponding options have changed.
August 9, 2021	Updated to version 3.14.94.x.x. Updated the I/O timeout field descriptions. Updated the descriptions of the I/O return values. Changed some argument and field names for consistency and clarity. Updated PIO Threshold and DMA Threshold descriptions. Added statements about the read and write services being blocking calls.
July 8, 2021	Updated to version 3.13.93.x.1.
May 5, 2021	Updated to version 3.13.93.x.0. Added notes about DMA transfers conditionally reverting to PIO.
March 25, 2021	Updated to version 3.12.93.x.1. Numerous minor editorial changes.
February 26, 2021	Updated to version 3.12.93.x.0.
February 18, 2021	Updated to version 3.11.93.x.0. Updated the upper index limit of the raw register read service. Added the raw register write service. Updated information on the Tx and Rx I/O DMA Threshold settings.
October 12, 2020	Updated to version 3.10.91.x.0. Added WAIT_EVENT note. Minor editorial changes.
July 30, 2019	Updated to version 3.9.87.x.0. Minor editorial changes. Added a licensing subsection.
March 24, 2019	Updated to version 3.8.83.x.0.
March 15, 2019	Updated to version 3.7.82.x.0. Added more OS specific documentation. Minor editorial changes. Some document reorganization. Renamed `GSC_WAIT_IO_xxx` macros to `SIO4_WAIT_IO_xxx`.
October 18, 2018	Updated to version 3.6.81.x.0. Added the default PIO threshold values, which are 44 bytes. Updated the inside cover page. Updated the Tx Char Count service description. Updated Block Mode DMA macro and associated information. Added Tx and Rx I/O DMA Threshold fields.
October 31, 2017	Updated to version 3.5.73.x.0. Added information on the Cable Configuration Modes. Updated some legacy setting information. Major document reorganization.
December 7, 2016	Updated to version 3.4.69.x.0. Updated the operating information section. Made various miscellaneous updates.
October 6, 2016	Updated to version 3.3.67.x.0. Added support for infinite I/O timeouts.
September 16, 2016	Updated to version 3.2.67.x.0. Updated the text of the Rx Address Control macros. Updated the usage of the Wait Event `timeout_ms` field. Updated material on the open call. Added open access mode descriptions.
April 13, 2016	Updated to version 3.1.65.x.0.
September 7, 2015	Updated to version 3.0.59.x.0. Added information about the FIFO Almost Empty and Almost Full states.
December 9, 2014	Updated the release date.
December 4, 2014	Updated to version 2.8.47.x.0. All occurrences of “isoch” have been changed to “isoc”, both lower and upper case, including file names. Updated the configuration aids and added figures for the SIO4A and the original SIO4.
July 31, 2014	Updated to version 2.7.44.5.0. Updated the SIO4-SYNC configuration aid figure.

May 17, 2014	Updated to version 2.7.44.x.0.
May 17, 2014	Updated to version 2.6.44.5.0.
April 16, 2014	Updated to version 2.5.43.4.0.
October 22, 2013	Updated to version 2.4.42.3.1.
October 22, 2013	Updated to version 2.4.42.3.0.
October 15, 2013	Updated to version 2.3.42.3.0.
August 27, 2013	Updated to version 2.2.42.x.0. Updated the `SIO4_IOCTL_OSC_REFERENCE` service for the CY22393 programmable oscillator. Updated the oscillator measurement service description.
October 11, 2012	Updated to version 2.1.41.1.0. Renamed `SIO4_SYNC_TX_BIT_COUNT` to `SIO4_SYNC_TX_WORD_SIZE`. Renamed `SIO4_USC_TX_STATUS_BLOCK` to `SIO4_USC_TX_CTRL_BLOCK`. Added the Rx and Tx I/O Abort services.
September 9, 2012	Initial release of the 2.x series driver.