# 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

# INtime
# Driver Interface Library
# User Manual

**Manual Revision: May 18, 2014**
**Driver Release Version 2.7.44.3.0**

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

# Preface



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

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

**General Standards Corporation** makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Although extensive editing and reviews are performed before release to ECO control, **General Standards Corporation** assumes no responsibility for any errors that may exist in this document. No commitment is made to update or keep current the information contained in this document.

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

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

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



The Purchaser of this software may use or modify in source form the subject software, but not to re-market or distribute it to outside agencies or separate internal company divisions. The software, however, may be embedded in the Purchaser's distributed software. In the event the Purchaser's customers require the software source code, then they would have to purchase their own copy of the software.

**General Standards Corporation** makes no warranty of any kind with regard to this software, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose and makes this software available solely on an "as-is" basis. **General Standards Corporation** reserves the right to make changes in this software without reservation and without notification to its users.

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

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

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

Zilog and Z16C30 are trademarks of Zilog, Inc.

# Table of Contents

# Table of Figures

# 1. Introduction

This user manual applies to driver release version 2.7.44.3.0.

## 1.1. Purpose

The purpose of this document is to describe the interface to the SIO4 INtime Driver Interface Library, and to a lesser degree the Device Driver. This software provides the application level interface to SIO4 serial channels.

## 1.2. Acronyms

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

| Acronyms | Description |
|---|---|
| DMA | Direct Memory Access |
| DMDMA | Demand Mode DMA |
| GSC | General Standards Corporation |
| USC | Universal Serial Controller |

## 1.3. Definitions

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

| Term | Definition |
|---|---|
| Application | Application means the user mode process, which runs in the user space with user mode privileges. |
| Device Mailbox | These global mailboxes, "`sio4.x`", give access to individual SIO4 serial channels. |
| Driver | This is the application that communicates directly with the SIO4 hardware. |
| Driver Mailbox | This global mailbox, "`sio4`", accesses the driver for informational purposes only. |
| SIO4 | This is used as a general reference to any board supported by this driver. |

## 1.4. Software Overview

The SIO4 INtime driver was developed under Windows XP Professional, 32-bit. It is written in C using Microsoft Visual C++ 2008 with Visual Studio 2008, Professional Edition. Also, it is an INtime 4.2 application developed with the INtime Addin installed into Visual Studio. The interface to the driver is message based. The details of the messaging protocol are not documented here as the recommended interface is through the Driver Interface Library (section 5, page 17).

Upon start up, the driver creates a mailbox and a corresponding support thread for each SIO4 serial channel. The Device Mailboxes are named `sio4.x`, where the *x* is the zero based index of the serial channel. The Device Thread listens for activity on the Device Mailbox. Each message received is immediately handed off to a service thread taken from a service thread pool common to all of the serial channels. The service thread examines the message content and passes the content to a driver function specific to the message type. The message type corresponds to specific driver operations, such as open, close, read, write and IOCTL processing. When the specific operation is complete the service thread sends a response message back to the originating thread, and then returns to the service thread pool. For more information refer to section 4.6 on page 15.

Upon start up, the driver also creates a Driver Mailbox named `sio4`, which is serviced by the driver's main thread. The Driver Mailbox supports open, close and read operations. The purpose of this mailbox is to provide basic identification information about the driver and the installed SIO4 boards. The information returned by read requests includes the driver version number, the quantity of installed SIO4 boards, and the base model number and user jumper identifier value of each board. For more information refer to section 4.6.2 on page 15.

## 1.5. Hardware Overview

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

The SIO4 is a four channel high-speed serial interface I/O board. This board provides for bi-directional serial data transfers between two computers, or a computer and an external peripheral. Each SIO4 board includes two DMA controllers and comes with a maximum of 256K Bytes of FIFO storage, which is 32K per channel side (32K * 2 * 4). The FIFO configuration can vary greatly from one SIO4 version to another (i.e. 256K, 32K or 4K). The SIO4 comes with transceivers that are either fixed (RS232 or RS485/422) or are configurable by software. The SIO4 comes in two basic models, one based on a pair of Universal Serial Controllers (Zilog Z16C30 USC) and one based on firmware designed for SYNC style cable interface. The DMA controllers are capable of transferring data to and from host memory; whereas the FIFO memory provides a means for continuous transfer of data without interrupting the DMA transfers or requiring intervention from the host CPU. The board also provides for interrupt generation for various states of the board like Sync Character detection, FIFO empty, FIFO full and DMA complete.

## 1.6. Reference Material

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

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

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

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

* The Zilog material is available from:

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

# 2. Installation

## 2.1. File Extraction

The SIO4 INtime device driver and Driver Interface Library are distributed as a compressed archive. The default file name is `sio4.intime.tar.gz`, which is a gzip compressed tar file, though the distributed file may include the driver version number. The archive includes all source code and associated files for the device driver, the Driver Interface Library, the Protocol Libraries, the utility libraries, the sample applications and the documentation. The archive should be decompressed and the contents placed in a suitable location. A default location is as given below. This is the default that will be used throughout this manual.

```
c:\gsc\sio4.intime\
```

## 2.2. Documentation

The documentation is placed in the root destination directory. The following table briefly describes the documentation provided.

| File | Description |
|---|---|
| `sio4_intime_um.pdf` | This is a PDF version of this user manual. |
| `sio4_rm.pdf` | This is a PDF version of the driver Reference Manual. |
| `sio4_hdlc_rm.pdf` | This is a PDF version of the HDLC Protocol Library Reference Manual. |
| `sio4_sync_rm.pdf` | This is a PDF version of the SYNC Protocol Library Reference Manual. |

## 2.3. Directory Structure

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

| Directory | Content |
|---|---|
| `sio4.intime` | This is the driver's root directory. |
| `sio4.intime/docsrc` | This directory contains the C sources from this user manual. |
| `sio4.intime/driver` | This directory contains the driver executable and its sources. |
| `sio4.intime/exit` | This directory contains the `exit` sample application. |
| `sio4.intime/id` | This directory contains the `id` sample application. |
| `sio4.intime/irq` | This directory contains the `irq` sample application. |
| `sio4.intime/lib` | This directory contains SIO4 Driver Interface Library. |
| `sio4.intime/regs` | This directory contains the `regs` sample application. |
| `sio4.intime/services` | This directory contains the `services` sample application. |
| `sio4.intime/txrate` | This directory contains the `txrate` sample application. |
| `sio4.intime/utils` | This directory contains utility sources used by the sample applications. |
| `sio4.intime/hdlc/lib` | This directory contains the HDLC model SIO4 library sources. * |
| `sio4.intime/hdlc/utils` | This directory contains HDLC specific utility sources. * |
| `sio4.intime/sync/docsrc` | This directory contains the C sources from the SYNC Library user manual. |
| `sio4.intime/sync/lib` | This directory contains the SYNC model SIO4 library sources. |
| `sio4.intime/sync/syncc2c` | This directory contains the `syncc2c` sample application for the SYNC model of the SIO4. |
| `sio4.intime/sync/utils` | This directory contains SYNC specific utility sources. |

* This release includes a preliminary version of the HDLC Protocol Library.

## 2.4. Overall Build Batch File

The driver archive includes a batch file designed to produce fresh builds of all INtime content included in the archive. The batch file is designed to invoke the build tools included with Visual Studio 2008 as installed in their

default location. If your tool set is other than Visual Studio 2008, then the batch file may not work as intended. If using other than Visual Studio 2008, then the batch file may need to be edited for use with your build tools. Porting of the project files may also be required. Before attempting to invoke the batch file, please exit any IDE or editor accessing any of the included INtime project or source files. Follow the below instructions to perform an overall build.

1. In a Command Window, change to the SIO4 INtime root directory.

   ```
   cd c:\gsc\sio4.intime
   ```

2. Invoke the batch file.

   ```
   .\bmake.bat
   ```

3. The build process should take less than five minutes. At the conclusion of the build process a prompt is presented. Review the output produced by the batch file as any errors are reported to the screen. Press the “return” key when finished.

4. The build log file is then opened into the command line editor (by the “edit” command). Review the logs and exit when finished.

# 3. Basic Software Architecture

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

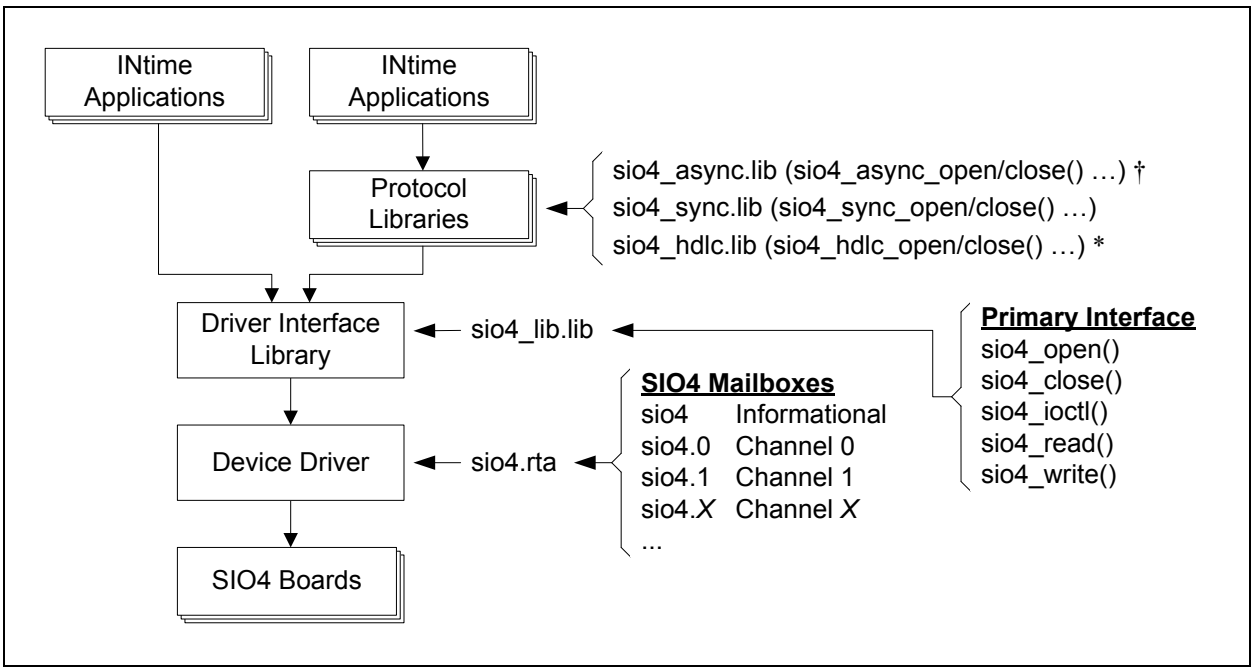


**Figure 1** The basic software architecture of INtime based SIO4 applications.

* This release includes a preliminary version of the HDLC Protocol Library.
† This release does not include an Asynchronous Protocol Library.

## 3.1. SIO4 Boards

At the bottom of Figure 1 are the SIO4 boards. Any number of boards may be utilized. While the end product of an effort may be a software application written for a specific model of SIO4, the software components provided by General Standards are generally written to function with all applicable SIO4 models.

## 3.2. Device Driver

The device driver is an INtime application, `sio4.rta`, written specifically to provide a software mechanism for interfacing with installed SIO4 hardware. The communications interface to the driver involves an intricate exchange of messages via a set of global mailboxes. These global mailboxes are created when the driver is started, and they are deleted when the driver exits. The Device Mailboxes are named "`sio4.x`", where the trailing "`x`" is the zero based index of the SIO4 serial channel to access. These are called *Device Mailboxes* as they provide access to individual SIO4 serial channels. The Driver Mailbox is named "`sio4`". This is called the *Driver Mailbox* as it provides information on the driver and the boards detected. Accessing this mailbox interacts with the driver only. The information includes the driver version and build date, the number of detected boards, their types, and their jumper id numbers. The figure below is an example of the text returned when reading from the Driver Mailbox. Refer to section 4 (page 13) for additional information.

## 3.3. Driver Interface Library

The Driver Interface Library is the sole means provided for communicating with the device driver and installed SIO4 hardware. This library implements the user side of the messaging protocol implemented by the driver for

interacting with an SIO4 serial channel. This is a statically linked library. The library's interface is defined in the header file `sio4_lib.h` and is limited to the services listed in Figure 1. The `sio4_ioctl()` service corresponds to the `DeviceIoControl()` call provided by Windows and the `ioctl()` call provided under many other operating systems, including Linux. The `sio4_ioctl()` service is the call used to apply changes to or read the setting for a serial channel's numerous parameters. The definitions for all parameters and their options are defined in the driver header files `sio4.h` and `sio4_usc.h`. Refer to the driver reference manual for a complete description of the definitions and their options. Refer to section 5 (page 17) for additional information.

## 3.4. Protocol Libraries

Protocol Libraries are libraries designed to facilitate the use of specific serial protocols. The interface provided by each library is tailored around the features and functionality specific to the respective protocol. Each library has its own interface, macros and services, which replace those of the Driver Interface Library. The protocol libraries sit on top of and replace the Driver Interface Library. Refer to section 6 (page 22) for additional information.

# 4. Device Driver

The paragraphs that follow give instructions on building and starting the device driver.

## 4.1. Build

Follow the below steps to build the driver.

1. Load the Visual Studio solutions file listed below.

   ```
   c:\gsc\sio4.intime\driver\sio4.sln
   ```

2. Select the Release configuration. Do so with the Configuration Manager, which is accessed with the Visual Studio menu selection "Build | Configuration Manager…" In the "Configuration Manager" window that appears, under the "Active solution configuration:" list, select the "Release" option. Then click the "Close" button.

3. Remove all existing build targets by selecting the Visual Studio menu option "Build | Clean Solution". The output in the status area should indicate one success, no failures and nothing skipped.

4. Rebuild all build targets by selecting the Visual Studio menu option "Build | Rebuild Solution". The output in the status area should indicate no errors, no warnings, one success, no failures and nothing skipped. The end product of build is the application listed below.

   ```
   c:\gsc\sio4.intime\driver\Release\sio4.rta
   ```

## 4.2. Host Preparation

Before the SIO4 INtime driver can be started there are several preparatory steps which must be performed first.

### 4.2.1. Windows' Driver Installation

The SIO4 Windows driver must be installed. This step is necessary so that appropriate system resources are assigned to each board. Each board's resources include memory regions, an I/O region and an interrupt. The assignment of the resources to each SIO4 is performed by the Windows driver. Installation of the Windows driver is described in documentation specific to that driver.

### 4.2.2. INtime Node Startup

Use of the SIO4 INtime driver on the local host requires that a local INtime Node be started. The INtime Node must be started each time the system is rebooted. Follow the below general procedures for manually starting a node.

1. Select the Windows main menu option "Start | Programs | INtime | INtime Configuration".

2. From the "INtime Configuration Panel" window that appears, double click on the "Node Management" icon.

3. From the "INtime Node Management" window that appears, select a local node in the upper left quadrant of the windows then click on the "Start Local" button. Wait for the node to start.

4. Dismiss the "INtime Node Management" window by clicking on the "Close" button.

5. Dismiss the "INtime Configuration Panel" window by clicking on the "Exit" button.

#### 4.2.3. SIO4 Interrupt Redirection

Use of the SIO4 INtime driver on the local host requires that each SIO4's interrupt be redirected from Window to INtime. The redirection must be done each time the system is rebooted. Follow the below general procedures to manually redirect the SIO4 interrupts.

1. Select the Windows main menu option "Start | Programs | INtime | INtime Configuration".

2. From the "INtime Configuration Panel" window that appears, double click on the "INtime Device Manager" icon.

3. From the "INtime Device Manager" window locate and right click on an SIO4 under the "Windows devices" list. The SIO4 boards may be listed under the "Other devices" device category.

4. From the popup menu that appears, select the option titled "Pass to INtime (with legacy IRQ)".

   **NOTE**: If INtime reports a conflict with an SIO4 in the right pane, then that conflict must be resolved before proceeding with driver startup.

5. Save the configuration by selecting the menu option "File | Save the configuration".

   **NOTE**: You may be presented with a "System Settings Change" window requesting a reboot. If so, perform the reboot after completing the interrupt redirection process.

6. Dismiss the "INtime Device Manager" window by clicking on the close button in the upper right corner of the window.

7. Dismiss the "INtime Configuration Panel" window by clicking on the "Exit" button.

### 4.3. Startup

Follow the below steps to start driver execution.

1. Select the Windows main menu option "Start | Programs | INtime | RT Application Loader". From the "RT Application Loader" window navigate to the driver, enter no command line arguments, and then click the "Open" button. The location is `c:\gsc\sio4.intime\driver\Release\sio4.rta`. (The driver has a `-u` command line argument, but that is used to initiate driver unloading.)

2. The driver should start and generate a small amount of display output. The driver will continue to execute until told to exit.

### 4.4. Shutdown

Shutdown the driver using the below options.

#### 4.4.1. Shutdown using driver command line arguments.

Follow the below steps to tell the driver to exit.

1. Select the Windows main menu option "Start | INtime | RT Application Loader". From the "RT Application Loader" window navigate to the executable listed below. Then enter the command line argument from the below table, then click the "Open" button.

   ```
   c:\gsc\sio4.intime\driver\Release\sio4.rta <-h> <-x>
   ```

| Argument | Description |
|---|---|
| `-h` | Display usage/help information. |
| `-x` | This initiates driver exit. |

2. The driver instance being executed should take only a second or so to complete its task. The driver already running should exit shortly thereafter.

#### 4.4.2. Shutdown using the *exit* sample application.

Follow the below steps to tell the driver to exit.

1. Build the *exit* sample application as described in section 8.1 on page 28.

2. Select the Windows main menu option "Start | INtime | RT Application Loader". From the "RT Application Loader" window navigate to the executable listed below. Then click the "Open" button.

   `c:\gsc\sio4.intime\exit\Debug\exit.rta`

3. The application should take one a second or so to complete its task. The driver should exit shortly thereafter.

## 4.5. Version

The driver version number can be obtained in two ways. First, it is reported by the driver both when the driver is loaded and when it is unloaded. This is part of the display output generated by the driver. Second, it is reported in the text obtained by reading from the mailbox named "sio4" cataloged under the root process. This mailbox is accessed using the Driver Interface Library call `sio4_open(-1)`. The driver must be running for this mailbox to be available.

## 4.6. Access

Access to the SIO4 driver and individual SIO4 serial channels is via mailboxes cataloged into the root process.

> **NOTE:** While direct application based messaging with the device driver is possible, it is recommended that all applications use the interface provided by the Driver Interface Library (section 5, page 17).

> **NOTE:** If a serial channel is in an open state when an application exits, the serial channel will remain in the opened state. If this occurs the driver may have to be reloaded to gain subsequent access to the same channel.

#### 4.6.1. Device Mailboxes: sio4.*x*

The SIO4 driver provides individual mailboxes for access to each SIO4 serial channel. These Device Mailboxes are cataloged into the root process and are named "sio4.*x*". The suffix "*x*" is the zero based index of the serial channel to access. (For example, the serial channels for the first board are accessed via Device Mailboxes "sio4.0", "sio4.1", "sio4.2" and "sio4.3".) Using the Driver Interface Library (section 5, page 17) an application calls `sio4_open(x)` to access a Device Mailbox, where the "x" in the call corresponds directly to the mailbox suffix.

#### 4.6.2. Driver Mailbox: sio4

The SIO4 driver provides a mailbox named "sio4". This mailbox is maintained so that applications can gain information about the driver and installed devices without having to query individual devices. (Such queries may not be possible as the driver restricts device access to only one application at a time.) This Driver Mailbox is accessed

using the `sio4_open(-1)` call from the Driver Interface Library (section 5, page 17). Once opened, a read returns the below information.

```
version: 2.7.44.3
built: May 18 2014, 14:49:03
boards: 1
models: SIO4BX
ids: 0x3
```

| Entry | Description |
|---|---|
| `version` | This gives the driver version number in the form `x.x.x.x`. |
| `built` | This gives the driver build date and time as a string. It is given in the C form of `printf("%s, %s", __DATE__, __TIME__)`. |
| `boards` | This identifies the total number of SIO4 boards the driver detected. |
| `models` | This is a list that identifies the base model numbers of the boards detected by the driver. The order in the list follows the Device Mailbox suffix sequence (i.e. the first board in the list corresponds to suffixes 0, 1, 2 and 3). If the driver cannot specifically identify a board's type it will be listed only as "SIO4". |
| `ids` | This is a list identifying the values read from the boards' user jumpers. |

## 4.7. Driver Interface

At the lowest level, direct access to the device driver is through the Device Mailboxes, which involves message based transactions. It is through these mailboxes that communication is established with a serial channel, through which the channel is configured and through which data is written to and read from the serial channels. The messaging interface is defined within the driver sources, but it is not described in any documentation. Instead, the interface that is documented, and which is recommended, is that presented by the Driver Interface Library (section 5, page 17).

> **NOTE:** While direct application based messaging with the device driver is possible, it is recommended that all applications use the interface provided by the Driver Interface Library (section 5, page 17).

The functional interface to the driver is IOCTL based. The IOCTL services are utilized in combination to achieve a higher level of functionality. The IOCTL command codes are defined in the header files `sio4.h` and `sio4_usc.h`. Their default location is `c:\gsc\sio4.intime\driver`. The IOCTL command codes are documented in the SIO4 Driver Reference Manual (`sio4_rm.pdf`).

An alternative to relying strictly on the IOCTL services is to use one of the libraries designed to support a specific serial protocol. These Protocol Libraries provide an interface tailored for use of the corresponding protocol. The libraries provide a higher level interface that encapsulates all the IOCTL service calls necessary for proper configuration and use of that protocol with the SIO4. The Protocol Libraries don't prevent the use of the IOCTL services, nor do they preclude their use. What the libraries do is significantly reduce the need for their use as well as significantly reduce the learning curve when using an SIO4. For additional information refer to the specific Protocol Library documentation.

# 5. Driver Interface Library

The Driver Interface Library is the sole means provided for applications to communicate with an SIO4 serial channel. The library sits between the application and the device driver (see Figure 1, page 11). The library handles all of the messaging necessary for proper communication with the driver.

## 5.1. Build

Follow the below steps to build the Driver Interface Library.

1. Load the Visual Studio solutions file listed below.

   ```
   c:\gsc\sio4.intime\lib\sio4_lib.sln
   ```

2. Select the Release configuration. Do so with the Configuration Manager, which is accessed with the Visual Studio menu selection "Build | Configuration Manager…" In the "Configuration Manager" window that appears, under the "Active solution configuration:" list, select the "Release" option. Then click the "Close" button.

3. Remove all existing build targets by selecting the Visual Studio menu option "Build | Clean Solution". The output in the status area should indicate one success, no failures and nothing skipped.

4. Rebuild all build targets by selecting the Visual Studio menu option "Build | Rebuild Solution". The output in the status area should indicate no errors, no warnings, one success, no failures and nothing skipped.

5. The product of the build is the statically linkable Driver Interface Library named below.

   ```
   c:\gsc\sio4.intime\lib\Release\sio4_lib.lib
   ```

## 5.2. Library Interface

The following describes the software interface defined for the Driver Interface Library. The library's interface is defined in the header file listed below.

```
c:\gsc\sio4.intime\lib\sio4_lib.h
```

### 5.2.1. sio4_open()

This function opens communication with the driver to a specified serial channel. Once opened the channel must be closed before it can be accessed by another application. The driver does not permit simultaneous access to a single channel by multiple applications.

> **NOTE:** If a serial channel is in an open state when an application exits, the serial channel will remain in the opened state. If this occurs the driver may have to be reloaded to gain subsequent access to the same channel.

Prototype

```
int sio4_open(int index);
```

| Argument | Description |
|---|---|
| `index` | This is the zero based index of the board to access. Values zero and above will access serial channels. The value `-1` will access the Driver Mailbox (see section 4.6.2, page 15). |

| Return Value | Description |
|---|---|
| -1 | An error occurred. Consult errno. |
| >= 0 | The operation succeeded. Use this value for all future accesses until the device is closed. |

Example

```
#include <stdio.h>

#include "sio4_dsl.h"

int sio4_dsl_open(int index)
{
    int fd;

    fd  = sio4_open(index);

    if (fd == -1)
    {
        printf( "ERROR: sio4_open()"
                " failure on SIO4 #%d, errno = %d\n",
                index,
                errno);
    }

    return(fd);
}
```

### 5.2.2. sio4_close()

This function ends previously established communication with the driver to a serial channel. Once closed the serial channel becomes available for use by other applications.

Prototype

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

| Argument | Description |
|---|---|
| fd | This is the file descriptor used to access the device. This is the value previously returned by sio4_open(). The file descriptor is not valid after the close operation. |

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

Example

```
#include <stdio.h>

#include "sio4_dsl.h"

int sio4_dsl_close(int fd)
{
```

```
    int errs;
    int status;

    status  = sio4_close(fd);

    if (status == -1)
        printf("ERROR: sio4_close() failure, errno = %d\n", errno);

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

### 5.2.3. sio4_ioctl()

This function is the entry point to performing I/O control operations (IOCTL operations). The operations recognized by the driver are defined in header files `sio4.h` and `sio4_usc.h` (both header files can be found in the directory `c:\gsc\sio4.intime\driver`). The operations supported by any given SIO4 board can be determined by using the `services` sample application (see section 8.7, page 34). All IOCTL requests on a given serial channel are serialized and will be processed in the order received by the driver. This is a blocking call, though most requests will return almost immediately. The only exception is the Wait Event service (`SIO4_IOCTL_WAIT_EVENT`). IOCTL service requests are not blocked while threads sleep as part of the Wait Event service.

#### Prototype

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

| Argument | Description |
|---|---|
| `fd` | This is the file descriptor used to access the device. This is the value previously returned by `sio4_open()`. |
| `cmd` | This is one of the IOCTL command codes defined in the header files `sio4.h` and `sio4_usc.h`. |
| `arg` | This is a pointer to the variable or structure being passed to the driver. The data type is specific to the IOCTL command code being used. |

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

#### Example

```
#include <stdio.h>

#include "sio4_dsl.h"

int sio4_dsl_ioctl(int fd, int request, void* arg)
{
    int errs;
    int status;

    status  = sio4_ioctl(fd, request, arg);

    if (status == -1)
        printf("ERROR: sio4_ioctl() failure, errno = %d\n", errno);
```

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

### 5.2.4. sio4_read()

This function is the entry point to retrieving data received by an SIO4 serial channel. The call will return either when the data request has been satisfied or when the read I/O timeout expires. Read requests are blocking calls, except when a PIO mode request is made with the read I/O timeout set at zero. See the driver reference manual for additional information. Read requests to a given serial channel are serialized as only one read can be active at a time.

Prototype

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

| Argument | Description |
|---|---|
| fd | This is the file descriptor used to access the device. This is the value previously returned by sio4_open(). |
| dst | This is the destination buffer for the data retrieved from the board. |
| bytes | This is the maximum number of bytes being requested, and must not exceed the size of the specified destination buffer. |

| Return Value | Description |
|---|---|
| -1 | An error occurred. Consult errno. |
| >= 0 | The operation succeeded. The value indicates the number of bytes placed in the destination buffer. A value less than the bytes argument typically means the read I/O timeout expired. |

Example

```
#include <stdio.h>

#include "sio4_dsl.h"

int sio4_dsl_read(int fd, void* src, int bytes)
{
    int got;

    got = sio4_read(fd, src, bytes);

    if (got == -1)
        printf("ERROR: sio4_read() failure, errno = %d\n", errno);

    return(got);
}
```

### 5.2.5. sio4_write()

This function is the entry point to providing data to be transmitted by an SIO4 serial channel. The call will return either when the data transfer has been completed or when the write I/O timeout expires. Write requests are blocking calls, except when a PIO mode request is made with the write I/O timeout set at zero. See the driver reference

manual for additional information. Write requests to a given serial channel are serialized as only one write can be active at a time.

### Prototype

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

| Argument | Description |
|---|---|
| fd | This is the file descriptor used to access the device. This is the value previously returned by sio4_open(). |
| src | This is the source buffer for the data being provided to the board. |
| bytes | This is the maximum number of bytes to send, and must not exceed the size of the specified source buffer. |

| Return Value | Description |
|---|---|
| -1 | An error occurred. Consult errno. |
| >= 0 | The operation succeeded. The value indicates the number of bytes taken from the source buffer. A value less than the bytes argument typically means the write I/O timeout expired. |

### Example

```
#include <stdio.h>

#include "sio4_dsl.h"

int sio4_dsl_write(int fd, const void* src, int bytes)
{
    int sent;

    sent    = sio4_write(fd, src, bytes);

    if (sent == -1)
        printf("ERROR: sio4_write() failure, errno = %d\n", errno);

    return(sent);
}
```

# 6. Protocol Libraries

Protocol Libraries are statically linked utility libraries which provide serial protocol specific interfaces to the SIO4. The set of libraries will vary over time, but each is built in a similar manner. For additional information on these libraries please refer to the user manual for the serial protocol of interest.

## 6.1. The SYNC Protocol Library

The SYNC Protocol Library is designed to facilitate use of the SYNC versions of the SIO4. The SYNC protocol is built into the SIO4 firmware and does not use the Zilog USC. The SYNC library and its support code is located under the directory `c:\gsc\sio4.intime\sync`. The utility libraries included with the SYNC Protocol are described under the Utility Libraries section (see section 6.2, page 22). The sample applications included with the SYNC Library are described under the Sample Applications section (see section 7.1, page 24). The library's interface is defined in the header file listed below.

```
c:\gsc\sio4.intime\sync\lib\sio4_sync.h
```

### 6.1.1. Build

Follow the below steps to build the library.

1. Load the Visual Studio solutions file listed below.

   ```
   c:\gsc\sio4.intime\sync\lib\lib.sln
   ```

2. Select the Release application configuration. Do so with the Configuration Manager, which is accessed with the Visual Studio menu selection "Build | Configuration Manager…" In the "Configuration Manager" window that appears, under the "Active solution configuration:" list, select the "Release" option. Then click the "Close" button.

3. Remove all existing build targets by selecting the Visual Studio menu option "Build | Clean Solution". The output in the status area should indicate one success, no failures and nothing skipped.

4. Rebuild all build targets by selecting the Visual Studio menu option "Build | Rebuild Solution". The output in the status area should indicate no errors, no warnings, one success, no failures and nothing skipped. The final built target is the file below.

   ```
   c:\gsc\sio4.intime\sync\lib\Release\sio4_sync.lib
   ```

## 6.2. The HDLC Protocol Library

The HDLC Protocol Library is provided as part of this release, but the implementation is preliminary. Only limited portions of the library have been tested. Future releases will expand the HDLC support.

**NOTE:** The HDLC Protocol Library is preliminary and untested.

The HDLC Protocol Library is designed to facilitate use of the HDLC serial protocol with Z16C30 based SIO4 boards. The HDLC protocol is a part of the Z16C30 DUART included on the SIO4. The HDLC library and its support code are located under the directory `c:\gsc\sio4.intime\hdlc`. The library's interface is defined in the header file listed below.

```
c:\gsc\sio4.intime\hdlc\lib\sio4_hdlc.h
```

### 6.2.1. Build

Follow the below steps to build the library.

1. Load the Visual Studio solutions file listed below.

   ```
   c:\gsc\sio4.intime\hdlc\lib\lib.sln
   ```

2. Select the Release application configuration. Do so with the Configuration Manager, which is accessed with the Visual Studio menu selection "Build | Configuration Manager…" In the "Configuration Manager" window that appears, under the "Active solution configuration:" list, select the "Release" option. Then click the "Close" button.

3. Remove all existing build targets by selecting the Visual Studio menu option "Build | Clean Solution". The output in the status area should indicate one success, no failures and nothing skipped.

4. Rebuild all build targets by selecting the Visual Studio menu option "Build | Rebuild Solution". The output in the status area should indicate no errors, no warnings, one success, no failures and nothing skipped. The final built target is the file below.

   ```
   c:\gsc\sio4.intime\hdlc\lib\Release\sio4_hdlc.lib
   ```

## 6.3. The Asynchronous Protocol Library

The Asynchronous Protocol Library is not available at this time.

# 7. Utility Libraries

The driver is accompanied by various utility libraries whose purpose is to aid in reducing the learning curve when using the SIO4.

## 7.1. The Document Source Code Library

This library contains the source code examples included in this user manual. The purpose is to insure that the examples compile and build. The library's interface is defined in the header file listed below.

```
c:\gsc\sio4.intime\docsrc\sio4_dsl.h
```

### 7.1.1. Build

Follow the below steps to build the library.

1. Load the Visual Studio solutions file listed below.

   ```
   c:\gsc\sio4.intime\docsrc\sio4_dsl.sln
   ```

2. Select the Release application configuration. Do so with the Configuration Manager, which is accessed with the Visual Studio menu selection "Build | Configuration Manager…" In the "Configuration Manager" window that appears, under the "Active solution configuration:" list, select the "Release" option. Then click the "Close" button.

3. Remove all existing build targets by selecting the Visual Studio menu option "Build | Clean Solution". The output in the status area should indicate one success, no failures and nothing skipped.

4. Rebuild all build targets by selecting the Visual Studio menu option "Build | Rebuild Solution". The output in the status area should indicate no errors, no warnings, one success, no failures and nothing skipped. The final build target is given below.

   ```
   c:\gsc\sio4.intime\docsrc\Release\sio4_dsl.lib
   ```

## 7.2. Utility Source Code

The driver archive includes utility source code suitable mostly for command line applications. A utility source file is included for every driver IOCTL service. Their purpose is to permit simple use of each service with suitable display output, if desired, where applicable, along with status. When there is an error, an error message is reported to the screen. Each service may be used to apply a setting, request the current setting, or to request support status. Other utility services are included as well. Of specific interest is `sio4_reg_list()`, which provides a detailed register dump of a channel's registers. This is especially useful for customer support when generated at a point in your code just prior to the appearance of a problem. The library's interface is defined in the header files listed below.

```
c:\gsc\sio4.intime\utils\sio4_utils.h
c:\gsc\sio4.intime\utils\gsc_utils.h
```

### 7.2.1. Build

Follow the below steps to build the library.

1. Load the Visual Studio solutions file listed below.

   ```
   c:\gsc\sio4.intime\utils\sio4_utils.sln
   ```

2. Select the Release application configuration. Do so with the Configuration Manager, which is accessed with the Visual Studio menu selection "Build | Configuration Manager…" In the "Configuration Manager" window that appears, under the "Active solution configuration:" list, select the "Release" option. Then click the "Close" button.

3. Remove all existing build targets by selecting the Visual Studio menu option "Build | Clean Solution". The output in the status area should indicate one success, no failures and nothing skipped.

4. Rebuild all build targets by selecting the Visual Studio menu option "Build | Rebuild Solution". The output in the status area should indicate no errors, no warnings, two successes, no failures and nothing skipped. The final build targets are given below.

   ```
   c:\gsc\sio4.intime\utils\Release\sio4_utils.lib
   c:\gsc\sio4.intime\utils\Release\gsc_utils.lib
   ```

## 7.3. The SYNC Document Source Code Library

This library contains the source code examples included in the SYNC Protocol Library user manual. The purpose is to insure that the example compile and build. The library's interface is defined in the header file listed below.

```
c:\gsc\sio4.intime\sync\docsrc\sio4_sync_dsl.h
```

### 7.3.1. Build

Follow the below steps to build the library.

1. Load the Visual Studio solutions file listed below.

   ```
   c:\gsc\sio4.intime\sync\docsrc\sio4_sync_dsl.sln
   ```

2. Select the Release application configuration. Do so with the Configuration Manager, which is accessed with the Visual Studio menu selection "Build | Configuration Manager…" In the "Configuration Manager" window that appears, under the "Active solution configuration:" list, select the "Release" option. Then click the "Close" button.

3. Remove all existing build targets by selecting the Visual Studio menu option "Build | Clean Solution". The output in the status area should indicate one success, no failures and nothing skipped.

4. Rebuild all build targets by selecting the Visual Studio menu option "Build | Rebuild Solution". The output in the status area should indicate no errors, no warnings, one success, no failures and nothing skipped. The final build target is given below.

   ```
   c:\gsc\sio4.intime\sync\docsrc\Release\sio4_sync_dsl.lib
   ```

## 7.4. SYNC Utility Source Code

The SYNC utility sources include a small set of files. The library's interface is defined in the header file listed below.

```
c:\gsc\sio4.intime\sync\utils\sio4_sync_utils.h
```

### 7.4.1. Build

Follow the below steps to build the library.

1. Load the Visual Studio solutions file listed below.

   ```
   c:\gsc\sio4.intime\sync\utils\sio4_sync_utils.sln
   ```

2. Select the Release application configuration. Do so with the Configuration Manager, which is accessed with the Visual Studio menu selection "Build | Configuration Manager…" In the "Configuration Manager" window that appears, under the "Active solution configuration:" list, select the "Release" option. Then click the "Close" button.

3. Remove all existing build targets by selecting the Visual Studio menu option "Build | Clean Solution". The output in the status area should indicate one success, no failures and nothing skipped.

4. Rebuild all build targets by selecting the Visual Studio menu option "Build | Rebuild Solution". The output in the status area should indicate no errors, no warnings, one success, no failures and nothing skipped. The final build target is given below.

   ```
   c:\gsc\sio4.intime\sync\utils\Release\sio4_sync_utils.lib
   ```

# 8. Sample Applications

The sample applications were developed under Windows 7 Professional, 32-bit. All of the applications are written in C using Microsoft Visual C++ 2008 with Visual Studio 2008, Professional Edition. Also, each is an INtime 4.2 application developed with the INtime Addin installed into Visual Studio. All project files include Debug and Release configurations for 32-bit versions of Windows. Building the applications first requires that the libraries from the previous section be built. Using the applications requires that the driver be built and that it is running.

## 8.1. drvr_mbox – Driver Mailbox

This sample application opens the Driver Mailbox and dumps its contents the console. The Driver Mailbox includes various pieces of information about the driver and the SIO4 boards it has detected. For additional information refer to section 4.6.2, page 15.

### 8.1.1. Build

Follow the below steps to build the sample application.

1. Load the Visual Studio solutions file listed below.

   `c:\gsc\sio4.intime\drvr_mbox\drvr_mbox.sln`

2. Select the desired application configuration to be built. Do so with the Configuration Manager, which is accessed with the Visual Studio menu selection "Build | Configuration Manager…" In the "Configuration Manager" window that appears, under the "Active solution configuration:" list, select either the "Debug" or the "Release" option. Then click the "Close" button.

3. Remove all existing build targets by selecting the Visual Studio menu option "Build | Clean Solution". The output in the status area should indicate one success, no failures and nothing skipped.

4. Rebuild all build targets by selecting the Visual Studio menu option "Build | Rebuild Solution". The output in the status area should indicate no errors, no warnings, one success, no failures and nothing skipped.

### 8.1.2. Execute

Follow the below steps to execute the sample application, which should have been built using the above procedure.

1. The easiest way to run the program is to select the Visual Studio menu option "Debug | Start Debugging".

2. The preferred means of executing the application is via the INtime tools using the menu option "Start | INtime | RT Application Loader". From the RT Application Loader window navigate to the application then click the "Open" button. The default application location is `c:\gsc\sio4.intime\drvr_mbox\Debug\drvr_mbox.rta`.

| Argument | Description |
|---|---|
| | This application has no command line arguments. |

3. The application should take only a second or so to complete its task.

## 8.2. exit – Driver Exit

This sample application communicates directly with the driver to initiate driver unloading. When the driver is executed it is usually started so that it establishes mailboxes for access to the driver and to each SIO4 device channel. The driver then waits for mailbox traffic for corresponding driver or device communication. This application provides a quick and easy means of telling the driver to close all channels, if any are open, remove all SIO4 mailboxes and then exit.

### 8.2.1. Build

Follow the below steps to build the sample application.

1. Load the Visual Studio solutions file listed below.

   `c:\gsc\sio4.intime\exit\exit.sln`

2. Select the desired application configuration to be built. Do so with the Configuration Manager, which is accessed with the Visual Studio menu selection "Build | Configuration Manager…" In the "Configuration Manager" window that appears, under the "Active solution configuration:" list, select either the "Debug" or the "Release" option. Then click the "Close" button.

3. Remove all existing build targets by selecting the Visual Studio menu option "Build | Clean Solution". The output in the status area should indicate one success, no failures and nothing skipped.

4. Rebuild all build targets by selecting the Visual Studio menu option "Build | Rebuild Solution". The output in the status area should indicate no errors, no warnings, one success, no failures and nothing skipped.

### 8.2.2. Execute

Follow the below steps to execute the sample application, which should have been built using the above procedure.

1. The easiest way to run the program is to select the Visual Studio menu option "Debug | Start Debugging".

2. The preferred means of executing the application is via the INtime tools using the menu option "Start | INtime | RT Application Loader". From the RT Application Loader window navigate to the application, enter the command line arguments from the below table, then click the "Open" button. The default application location is `c:\gsc\sio4.intime\exit\Debug\exit.rta`.

| Argument | Description |
|---|---|
| | This application has no command line arguments. |

3. The application should take only a second or so to complete its task. The driver should exit shortly thereafter.

## 8.3. id - Identify Board

This sample console application is a command line driven INtime application that reports detailed board identification information. This can be used with tech support to help identify as much technical information about the board as possible from software.

### 8.3.1. Build

Follow the below steps to build the sample application.

1. Load the Visual Studio solutions file listed below.

   `c:\gsc\sio4.intime\id\id.sln`

2. Select the desired application configuration to be built. Do so with the Configuration Manager, which is accessed with the Visual Studio menu selection "Build | Configuration Manager…" In the "Configuration Manager" window that appears, under the "Active solution configuration:" list, select either the "Debug" or the "Release" option. Then click the "Close" button.

3. Remove all existing build targets by selecting the Visual Studio menu option "Build | Clean Solution". The output in the status area should indicate one success, no failures and nothing skipped.

4. Rebuild all build targets by selecting the Visual Studio menu option "Build | Rebuild Solution". The output in the status area should indicate no errors, no warnings, one success, no failures and nothing skipped.

### 8.3.2. Execute

Follow the below steps to execute the sample application, which should have been built using the above procedure.

1. The easiest way to run the program is to select the Visual Studio menu option "Debug | Start Debugging". The command line options can be set in the project's Property Pages.

2. The preferred means of executing the application is via the INtime tools using the menu option "Start | INtime | RT Application Loader". From the RT Application Loader window navigate to the application, enter the command line arguments from the below table, then click the "Open" button. The default application location is `c:\gsc\sio4.intime\id\Debug\id.rta`.

| Argument | Description |
|---|---|
| `-fw` | This causes the output to include a detailed dump of the channel's firmware register. |
| `-usc` | This causes the output to include a detailed dump of the channel's USC register. |
| `index` | This is the zero based index of the channel to access. |

3. The application should take only a second or so display all output then exit.

## 8.4. irq – Interrupt Test

*THIS APPLICATION AND THE DRIVER INTERRUPTS ARE NOT FUNCTIONAL IN THIS RELEASE.*

This sample application provides a command line driven application intended to verify the operation of some of the board's interrupts.

### 8.4.1. Build

Follow the below steps to build the sample application.

1. Load the Visual Studio solutions file listed below.

   `c:\gsc\sio4.intime\irq\irq.sln`

2. Select the desired application configuration to be built. Do so with the Configuration Manager, which is accessed with the Visual Studio menu selection "Build | Configuration Manager…" In the "Configuration Manager" window that appears, under the "Active solution configuration:" list, select either the "Debug" or the "Release" option. Then click the "Close" button.

3. Remove all existing build targets by selecting the Visual Studio menu option "Build | Clean Solution". The output in the status area should indicate one success, no failures and nothing skipped.

4. Rebuild all build targets by selecting the Visual Studio menu option "Build | Rebuild Solution". The output in the status area should indicate no errors, no warnings, one success, no failures and nothing skipped.

### 8.4.2. Execute

Follow the below steps to execute the sample application, which should have been built using the above procedure.

1. The easiest way to run the program is to select the Visual Studio menu option "Debug | Start Debugging". The command line options can be set in the project's Property Pages.

2. The preferred means of executing the application is via the INtime tools using the menu option "Start | INtime | RT Application Loader". From the RT Application Loader window navigate to the application, enter the command line arguments from the below table, then click the "Open" button. The default application location is `c:\gsc\sio4.intime\irq\Debug\irq.rta`.

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

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

## 8.5. led – LED Test

This sample application provides a command line driven application that exercises the software accessible LEDs on the SIO4. The purpose of this application is to provide a working example of how to control the LEDs.

### 8.5.1. Build

Follow the below steps to build the sample application.

1. Load the Visual Studio solutions file listed below.

   `c:\gsc\sio4.intime\led\led.sln`

2. Select the desired application configuration to be built. Do so with the Configuration Manager, which is accessed with the Visual Studio menu selection "Build | Configuration Manager…" In the "Configuration Manager" window that appears, under the "Active solution configuration:" list, select either the "Debug" or the "Release" option. Then click the "Close" button.

3. Remove all existing build targets by selecting the Visual Studio menu option "Build | Clean Solution". The output in the status area should indicate one success, no failures and nothing skipped.

4. Rebuild all build targets by selecting the Visual Studio menu option "Build | Rebuild Solution". The output in the status area should indicate no errors, no warnings, one success, no failures and nothing skipped.

### 8.5.2. Execute

Follow the below steps to execute the sample application, which should have been built using the above procedure.

1. The easiest way to run the program is to select the Visual Studio menu option "Debug | Start Debugging". The command line options can be set in the project's Property Pages.

2. The preferred means of executing the application is via the INtime tools using the menu option "Start | INtime | RT Application Loader". From the RT Application Loader window navigate to the application, enter the command line arguments from the below table, then click the "Open" button. The default application location is `c:\gsc\sio4.intime\led\Debug\led.rta`.

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

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

## 8.6. regs - Register Access

This sample application is a menu based command line INtime application that permits interactive access to the board's registers, including write access to the GSC and USC specific registers.

### 8.6.1. Build

Follow the below steps to build the sample application.

1. Load the Visual Studio solutions file listed below.

   `c:\gsc\sio4.intime\regs\regs.sln`

2. Select the desired application configuration to be built. Do so with the Configuration Manager, which is accessed with the Visual Studio menu selection "Build | Configuration Manager…" In the "Configuration Manager" window that appears, under the "Active solution configuration:" list, select either the "Debug" or the "Release" option. Then click the "Close" button.

3. Remove all existing build targets by selecting the Visual Studio menu option "Build | Clean Solution". The output in the status area should indicate one success, no failures and nothing skipped.

4. Rebuild all build targets by selecting the Visual Studio menu option "Build | Rebuild Solution". The output in the status area should indicate no errors, no warnings, one success, no failures and nothing skipped.

### 8.6.2. Execute

Follow the below steps to execute the sample application, which should have been built using the above procedure.

1. The easiest way to run the program is to select the Visual Studio menu option "Debug | Start Debugging". The command line options can be set in the project's Property Pages.

2. The preferred means of executing the application is via the INtime tools using the menu option "Start | INtime | RT Application Loader". From the RT Application Loader window navigate to the application, enter the command line arguments from the below table, then click the "Open" button. The default application location is `c:\gsc\sio4.intime\regs\Debug\regs.rta`.

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

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

## 8.7. services – Supported IOCTL Services

This sample application is a command line INtime application that reports the set of driver IOCTL services which are supported for the board being accessed. The SIO4 driver includes IOCTL services for virtually all features of every standard model SIO4 produced. This application will identify which of the many services are applicable to the board being accessed.

### 8.7.1. Build

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

1. Load the Visual Studio solutions file listed below.

   `c:\gsc\sio4.intime\services\services.sln`

2. Select the desired application configuration to be built. Do so with the Configuration Manager, which is accessed with the Visual Studio menu selection "Build | Configuration Manager…" In the "Configuration Manager" window that appears, under the "Active solution configuration:" list, select either the "Debug" or the "Release" option. Then click the "Close" button.

3. Remove all existing build targets by selecting the Visual Studio menu option "Build | Clean Solution". The output in the status area should indicate one success, no failures and nothing skipped.

4. Rebuild all build targets by selecting the Visual Studio menu option "Build | Rebuild Solution". The output in the status area should indicate no errors, no warnings, one success, no failures and nothing skipped.

### 8.7.2. Execute

Follow the below steps to execute the sample application, which should have been built using the above procedure.

1. The easiest way to run the program is to select the Visual Studio menu option "Debug | Start Debugging". The command line options can be set in the project's Property Pages.

2. The preferred means of executing the application is via the INtime tools using the menu option "Start | INtime | RT Application Loader". From the RT Application Loader window navigate to the application, enter the command line arguments from the below table, then click the "Open" button. The default application location is `c:\gsc\sio4.intime\services\Debug\services.rta`.

| Argument | Description |
|---|---|
| `-u` | List only unsupported services. |
| `index` | This is the zero based index of the channel to access. |

3. The application will report information based on queries of the driver. A single iteration should take just a few seconds.

## 8.8. syncc2c – SYNC Channel-to-Channel Data Transfer

This sample application performs a data transfer test between a designated transmit channel and a corresponding receive channel. The channels designated may be the same exact channel, two different channels on the same board, or two different channels on two different boards.

### 8.8.1. Build

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

1. Load the Visual Studio solutions file listed below.

   `c:\gsc\sio4.intime\sync\syncc2c\syncc2c.sln`

2. Select the desired application configuration to be built. Do so with the Configuration Manager, which is accessed with the Visual Studio menu selection "Build | Configuration Manager…" In the "Configuration Manager" window that appears, under the "Active solution configuration:" list, select either the "Debug" or the "Release" option. Then click the "Close" button.

3. Remove all existing build targets by selecting the Visual Studio menu option "Build | Clean Solution". The output in the status area should indicate one success, no failures and nothing skipped.

4. Rebuild all build targets by selecting the Visual Studio menu option "Build | Rebuild Solution". The output in the status area should indicate no errors, no warnings, one success, no failures and nothing skipped.

### 8.8.2. Execute

Follow the below steps to execute the sample application, which should have been built using the above procedure.

1. The easiest way to run the program is to select the Visual Studio menu option "Debug | Start Debugging". The command line options can be set in the project's Property Pages.

2. The preferred means of executing the application is via the INtime tools using the menu option "Start | INtime | RT Application Loader". From the RT Application Loader window navigate to the application, enter the command line arguments from the below table, then click the "Open" button. The default location is `c:\gsc\sio4.intime\sync\syncc2c\Debug\syncc2c.rta`.

| Argument | Description |
|---|---|
| `-c` | Perform continuous testing until an error occurs. |
| `-C` | Perform continuous testing. |
| `-m#` | Perform continuous testing for at most this many minutes. |
| `-n#` | Perform continuous testing for at most this many iterations. |
| `tx` | This is the zero based index of the transmit channel. |
| `rx` | This is the zero based index of the receive channel. |

3. The application will perform a series of tests followed by a period of data transfer. A single iteration should take about 30 seconds.

## 8.9. txrate – Transmit Bit Rate Calculation

This sample application will compute the board's necessary programmable oscillator settings for user specified bit rates. The calculations are first performed using the driver sources and then they are performed by the driver itself. The results of both are reported for each channel.

### 8.9.1. Build

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

1. Load the Visual Studio solutions file listed below.

   `c:\gsc\sio4.intime\sync\txrate\txrate.sln`

2. Select the desired application configuration to be built. Do so with the Configuration Manager, which is accessed with the Visual Studio menu selection "Build | Configuration Manager…" In the "Configuration Manager" window that appears, under the "Active solution configuration:" list, select either the "Debug" or the "Release" option. Then click the "Close" button.

3. Remove all existing build targets by selecting the Visual Studio menu option "Build | Clean Solution". The output in the status area should indicate one success, no failures and nothing skipped.

4. Rebuild all build targets by selecting the Visual Studio menu option "Build | Rebuild Solution". The output in the status area should indicate no errors, no warnings, one success, no failures and nothing skipped.

### 8.9.2. Execute

Follow the below steps to execute the sample application, which should have been built using the above procedure.

1. The easiest way to run the program is to select the Visual Studio menu option "Debug | Start Debugging". The command line options can be set in the project's Property Pages.

2. The preferred means of executing the application is via the INtime tools using the menu option "Start | INtime | RT Application Loader". From the RT Application Loader window navigate to the application, enter the command line arguments from the below table, then click the "Open" button. The default application location is `c:\gsc\sio4.intime\sync\txrate\Debug\txrate.rta`.

| Argument | Description |
|---|---|
| `-async` | Perform the calculation for use with the Asynchronous serial protocol. |
| `-b#` | Begin a sequential rate calculation at the specified rate. |
| `-best` | Take additional time to produce the best possible results. This is specific to the `-sync` option on Z16C30 based USC boards. |
| `-e#` | End a sequential rate calculation at the specified rate. |
| `-h#` | Hold the –b output signals active at the cable interface for this number of seconds. |
| `-m` | Measure the oscillator output for each computed scan results. This is applicable only when the `-osc` and `-p` options are also specified. |
| `-osc` | Report the oscillator configuration information. |
| `-p` | Program the oscillator for each scanned rate. This is applicable only when the `-osc` option is also specified. |
| `-s` | Save the scan results to the text file `txrate.txt`. |
| `-sync` | Perform the calculation for use synchronous serial operation. For Z16C30 based boards this assumes the HDLC serial protocol. For `-SYNC` model boards this is the default. |
| `index` | This is the zero based index of the serial channel to access. |

**NOTE**: If the `-b` and the `-e` options are both given, then the operation is carried out for each bit rate in the specified inclusive range.

**NOTE**: If the `-b` option is given and the `-e` option is omitted, then the results computed for the specified `-b` bit rate will appear at the cable interface for the `-h` specified period.

**NOTE**: If the `-osc` option is given and the `-b` and `-e` options are omitted, then the oscillator output will appear at the cable interface.

3. The application will make the calculations and complete very quickly. Computations for a range may take significantly longer.

# 9. Operation

This section explains some operational procedures on using the driver. This is in no way intended to be a comprehensive guide on using the SIO4 and makes no attempt at explaining configuration of the Zilog Z16C30. This is simply to address a very few issues relating to GSC specific features of the SIO4.

## 9.1. SIO4 Zilog Configuration Aid

The figure below is intended as an aid to those trying to configure an SIO4 with a USC.

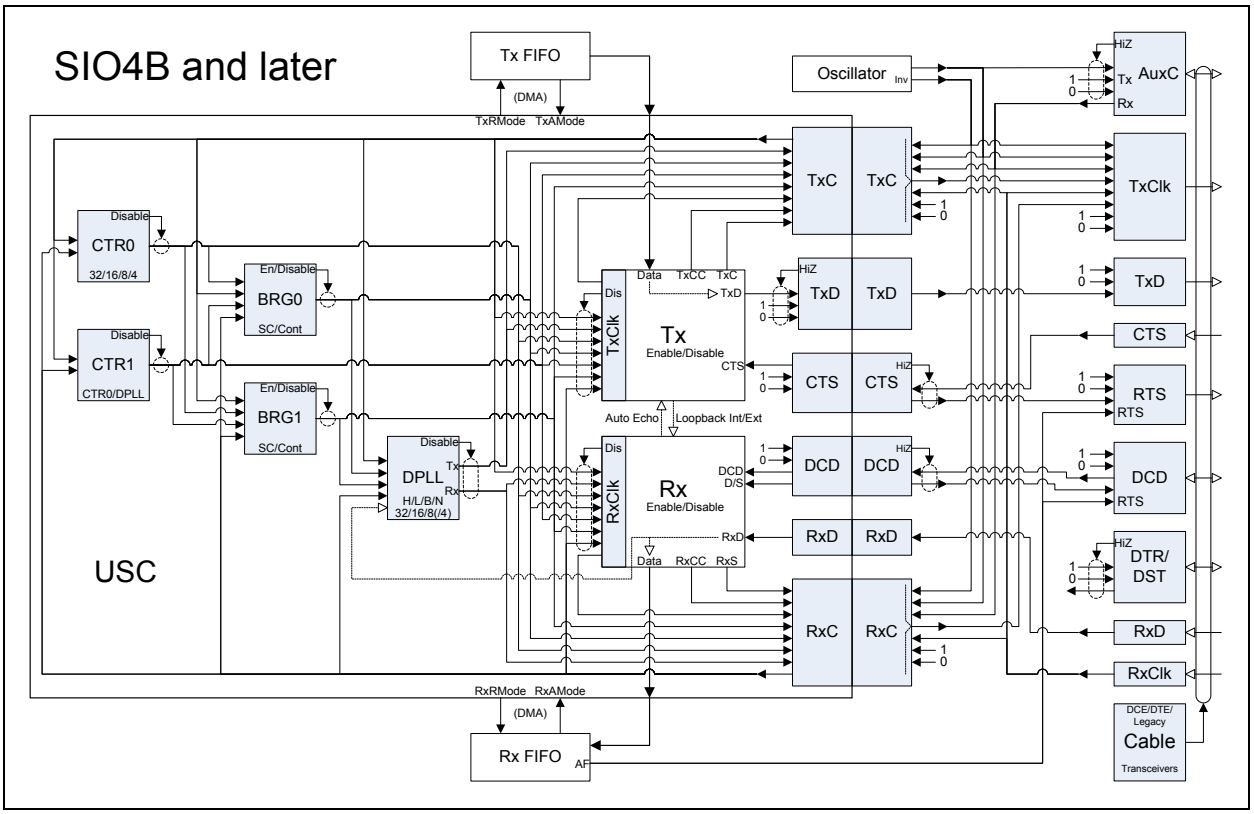


**Figure 2** A configuration aid for Zilog based SIO4B and later boards.

## 9.2. SIO4-SYNC Configuration Aid

The figure below is intended as an aid to those trying to configure SIO4-SYNC model boards.

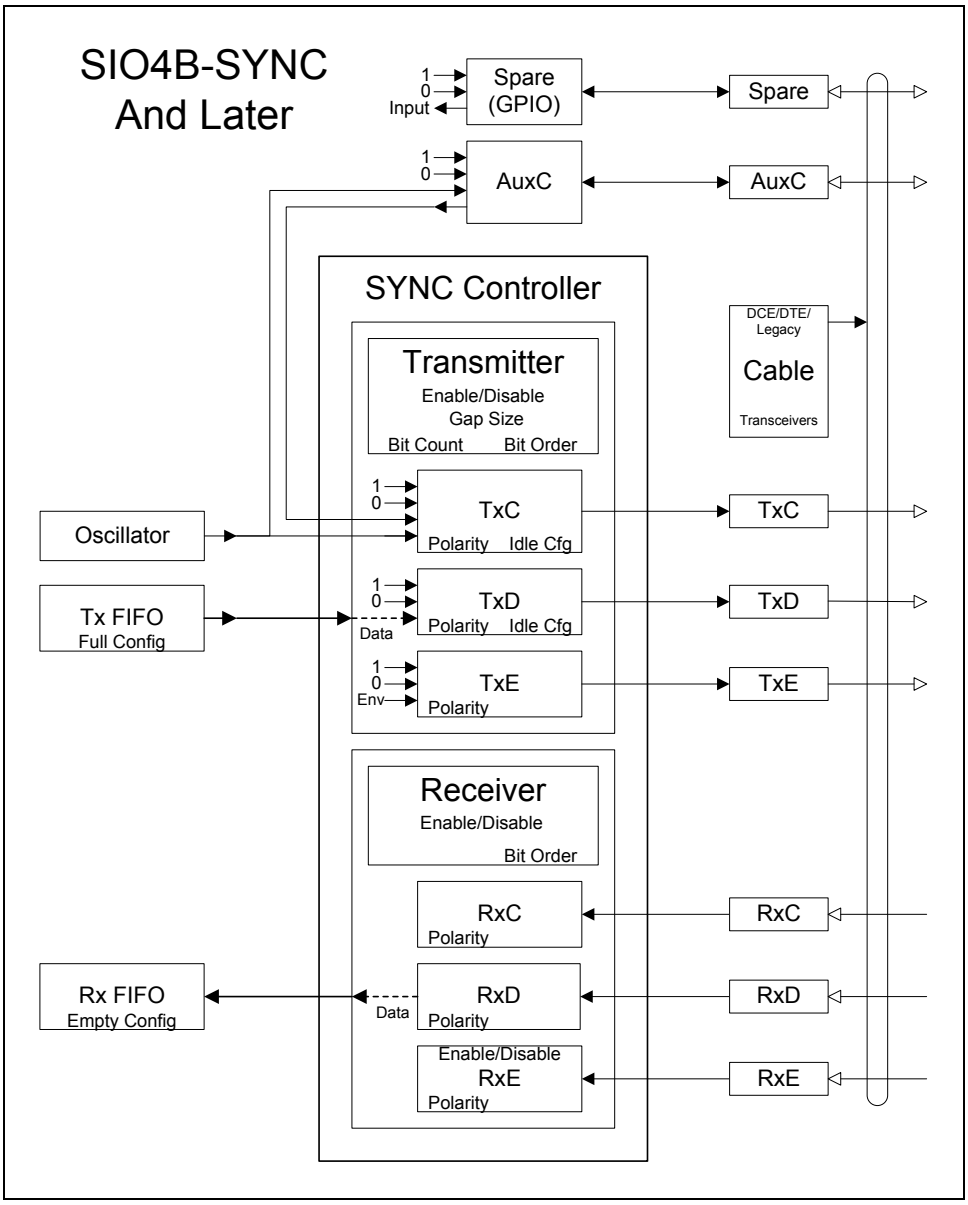


**Figure 3** A configuration aid for SIO4B and later –SYNC boards.

# Document History

| Revision | Description |
|---|---|
| May 18, 2014 | Updated to driver release version 2.7.44.3.0. Added the Driver Mailbox application. Added information on setting command line options while running debugging application builds. |
| October 22, 2013 | Updated to driver release version 2.4.42.2.0. |
| August 29, 2013 | Updated to driver release version 2.2.42.1.0. Numerous updates. Removed the `sync/libtest` application. |
| September 16, 2012 | Initial release of the INtime Device Driver and Driver Interface Library. Driver release version 2.0.40.0.0. |