GENERAL STANDARDS SIO4 LABVIEW DRIVER FOR WINDOWS NT AND WIN 2000

Revision B (October 2003)

Additional copies of this manual or other General Standards Co. literature may be obtained from:

General Standards Corp.
8302A Whitesburg Dr.
Huntsville, Alabama 35802
Telephone: (256) 880-8787
FAX: (256) 880-8788

The information in this document is subject to change without notice.

General Standards Corp. 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 Corp. 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 Corp. 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 Corp. assumes no responsibility resulting from omissions or errors in this manual, or from the use of information contained herein.

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

INTRODUCTION

The General Standards Corp. SIO4 is a four channel full duplex RS422/485 (RS232 optional) serial I/O card. Each channel can operate synchronously up to 10Mbits/s, or asynchronously up to 1Mbit/s. Independent transmit and receive FIFOs (up to 32Kbyte) on each channel provide for a smooth and efficient interface between the serial interfaces and the PCI host system. The board is based on the Zilog 16C30 high speed Universal Serial Controller (USC). This part supports Asynchronous, Isochronus, Monosync, Bisync, HDLC, SDLC, External Sync and Nine-Bit protocols. The USC chip provides full duplex operation with baud rate generators, digital phase locked loop for clock recovery and a full duplex DMA interface.

The purpose of this document is to describe the LabView application developer's interface to the SIO4. Throughout the rest of the manual, the various transmit, receive, file I/O, and device control functions will be defined.

SETTING UP THE SIO4 DRIVER

For LabView Users:

Copy the SIO4 folder (and all its contents) from the directory:

Program Files\General Standards Corporation\SIO4 LabView\SIO4LabiewDriver

To the <inst.lib> folder where Labview is installed.

When LabView is started, it will automatically add the SIO4 menu to the Functions palette, Instrument I/O sub-palette.

For LabWindows CVI Users:

The LabWindows CVI user has only to load the SIO4 function panel into the new project. Under the tool bar menu "Instrument" select the "Load" operation.

The SIO4.fp and SIO4.dll files will reside in the directory:
Program Files\General Standards Corporation\SIO4 LabView\SIO4LabiewDriver

Installing the hardware:

If you are using Windows NT, you can install the SIO4 cards before or after installing the LabView driver suite. If the card is not present, the system will indicate an error at the next restart such as "Driver or Service Failed to Start". This error will go away as soon as the card is installed.

If you are using Windows 2000 or Windows XP, it is recommended you installed the LabView driver suite first, and then install the hardware. This way, the Windows plug and play manager can easily find the required driver. If you happen to have installed the card first, simple go to the device manager (located in the Control Panel under the System applet under the Hardware tab) and delete the SIO4 card (will likely be listed as Unknown hardware or Other PCI Bridge Device) and restart the system. The next time the plug and play manager runs, it will find and install the SIO4 driver.

SIO4_OPEN

This is the driver entry point and must always be the first function called. The Open function allocated the system resources to allow communication with the SIO4 device.

input parameters:
- signed 32-bit integer — **ChannelNumber**
- error cluster — **error in**

output parameters:
- signed 32-bit integer — **Status**
- error cluster — **error out**

ChannelNumber defines the serial channel to open. The four channels on the first SIO4 board are accessed through Channel Numbers 1-4. If a second SIO4 board is present in the system, it is accessed by Channel Numbers 5-8. Additional boards are accessed by each subsequent set of four Channel Numbers.

After a successful call, the channel opened will always be accessed and referenced by the same **ChannelNumber**. All calls to different functions on the same channel would use the same ChannelNumber argument.

A returned **Status** of zero indicates no error – a non-zero value indicates an error condition.

SIO4_CLOSE

This routine is the driver exit point and should be the last function called to deallocate system resources.

input parameters:
- signed 32-bit integer — **ChannelNumber**

output parameters:
- none

ChannelNumber is the same channel number used to open the channel.

SIO4_READ_REGISTER

This routine reads the GSC Local Registers (see SIO4 User's manual) and the USC Serial Registers (see Zilog Z16C30 user's manual).

input parameters:

signed 32-bit integer	**ChannelNumber**
unsigned 32-bit integer	**Offset**
error cluster	**error in**

output parameters:

unsigned 32-bit integer	**Data**
error cluster	**error out**

The **Offset** specifies the local offset of the register to read, along with the type of register. The Offsets of the local registers are listed with the SIO4_Write_Register command. The read register data is returned in the **Data** field.

SIO4_WRITE_REGISTER

This routine writes data to the GSC Local Registers (see SIO4 User's manual) and the USC Serial Registers (see Zilog Z16C30 user's manual).

input parameters:

signed 32-bit integer	**ChannelNumber**
unsigned 32-bit integer	**Offset**
unsigned 32-bit integer	**Data**

output parameters:

none

The Offset specifies the local offset of the register to write, along with the type of register, GSC Local (10xx) or USC (100xx) register. The Offsets for the local registers are listed in the following tables:

GSC Local Register	Offset (Decimal)	RW Access	Notes
Firmware Revision Register	1000	Read Only	
Board Control Register	1001	Read/Write	
Board Status Register	1002	Read Only	
Clock Control Register	1003	Read/Write	
Channel TX Almost Register	1004	Read/Write	
Channel RX Almost Register	1005	Read/Write	
Channel FIFO	1006	Read/Write	
Channel Control/Status Register	1007	Read/Write	
Channel Sync Register	1020	Read/Write	
Interrupt Control Register	1024	Read/Write	
Interrupt Status Register	1025	Read/Write	
Interrupt Edge/Level Register	1026	Read/Write	
Interrupt Hi/Low Register	1027	Read/Write	
Channel Pin Source Register	1032	Read/Write	SIO4A/SIO4B only
Channel Prog Clock Register	1040	Read/Write	SIO4A/SIO4B only
Channel FIFO Count Register	1052	Read/Write	SIO4A/SIO4B only
Channel FIFO Size Register	1056	Read/Write	SIO4A/SIO4B only
Features Register	1063	Read Only	SIO4A/SIO4B only

USC Register	Abbreviation	Offset (Decimal)
Bus Configuration Register	BCR	10000
Channel Command Address Register	CCAR	10000
Channel Mode Register	CMR	10002
Channel Command/Status Register	CCSR	10004
Channel Control Register	CCR	10006
Test Mode Data Register	TMDR	10008
Test Mode Control Register	TMCR	10014
Clock Mode Control Register	CMCR	10016
Hardware Configuration Register	HCR	10018
Interrupt Vector Register	IVR	10020
IO Control Register	IOCR	10022
Interrupt Control Register	ICR	10024
Daisy Chain Control Register	DCCR	10026
Misc Interrupt Status Register	MISR	10028
Status Interrupt Control Register	SICR	10030
Receive Data Register	RDR	10032
Receive Mode Register	RMR	10034
Receive Command/Status Register	RCSR	10036
Receive Interrupt Control Register	RICR	10038
Receive Sync Register	RSR	10040
Receive Count Limit Register	RCLR	10042
Receive Character Count Register	RCCR	10044
Time Constant 0 Register	TC0R	10046
Transmit Data Register	TDR	10048
Transmit Mode Register	TMR	10050
Transmit Command/Status Register	TCSR	10052
Transmit Interrupt Control Register	TICR	10054
Transmit Sync Register	TSR	10056
Transmit Count Limit Register	TCLR	10058
Transmit Character Count Register	TCCR	10060
Time Constant 1 Register	TC1R	10062

SIO4_RECEIVE_BYTE

Reads data from the SIO4 channel receive FIFO

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**BytesToRead**

output parameters:

unsigned 8-bit array	**RxBuffer**
unsigned 32-bit integer	**BytesRead**

Data is read to the channel receive FIFO into an unsigned 8-bit array, **RxBuffer**. **BytesToRead** defines the requested transfer size. **BytesRead** will return the actual number of bytes read. The actual number of bytes read may be less than the requested number of bytes to read if a read timeout occurs.

The transfer timeout will be set by the SIO4_SET_READ_TIMEOUT command. The transmit DMA mode is set by the SIO4_DMA_DEMAND_MODE and the SIO4_SET_INPUT_DMA commands.

SIO4_TRANSMIT_BYTE

Sends data to the SIO4 channel transmit FIFO.

input parameters:

signed 32-bit integer	**ChannelNumber**
unsigned 8-bit array	**TxBuffer**
unsigned 32-bit integer	**BytesToWrite**

output parameters:

unsigned 32-bit integer	**BytesWritten**

Data is sent to the transmit FIFO in an unsigned 8-bit array, **TxBuffer**. The **BytesToWrite** must be less than or equal to the **TxBuffer** size.

The returned **BytesWritten** value indicates the number of bytes actually transferred. This value may be less than the requested **BytesToWrite** if a write timeout occurs before all the requested data is transferred.

The transfer timeout will be set by the SIO4_SET_WRITE_TIMEOUT command. The transmit DMA mode is set by the SIO4_DMA_DEMAND_MODE and the SIO4_SET_OUTPUT_DMA commands.

SIO4_SET_READ_TIMEOUT

Sets the timeout period for a SIO4_RECEIVE_BYTE command to complete.

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**Seconds**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

Seconds sets the wait period for a receive byte command to complete. The receive byte command completes normally within the set timeout (number of bytes read equal to requested number of bytes to read). If the timeout occurs before the requested number bytes are read, the receive byte routine will return normally with the number of bytes actually read indicated. A value of zero seconds will set an indefinite wait period.

SIO4_SET WRITE_TIMEOUT

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**Seconds**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

Seconds sets the wait period for a transmit byte command to complete. The transmit byte command completes normally within the set timeout (number of bytes written equal to requested number of bytes to write). If the timeout occurs before the requested number bytes are written, the receive byte routine will return normally with the number of bytes actually written indicated. A value of zero seconds will set an indefinite wait period.

The user has a choice to perform programmed I/O read/write operations or DMA transfers. If only a small block of data is to me transmitted or received then programmed I/O is the best solution in which case the user would set the appropriate BOOLEAN to "Off".
Turning either BOOLEAN on will cause DMA transfers to take place when TRANSMIT or RECEIVE are called.

SIO4_SET_DEMAND_MODE_DMA

Sets DMA Mode to Demand Mode or Non_Demand Mode for data transfers to/from transmit and receive FIFOs.

input parameters:

signed 32-bit integer	**ChannelNumber**
boolean	**DemandMode**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

The SIO4_SET_DEMAND_MODE_DMA command works with the SIO4_SET_DMA_INPUT_ENABLE and SIO4_SET_DMA_OUTPUT_ENABLE to define the Rx and Tx DMA modes. If DMA is enabled by either or both of the DMA Input/Ouput Enable commands, this routine will define the DMA as either Demand Mode or Non-Demand Mode DMA.

There are three different transfer modes for moving data to/from the channel FIFOs: PIO Mode, Non-Demand DMA, and Demand Mode DMA. Each mode has advantages for different data transfer conditions

PIO Mode (DMA disabled) is the slowest, but safest mode. In PIO mode, the FIFO flags are checked to determine if a data byte can be transferred (FIFO not full for Tx, FIFO not empty for Rx). If the FIFO is full/empty, the FIFO status will continue to be polled until a data byte is ready to transfer. When available, a single byte will be transferred to/from the FIFO, and the FIFO status will be checked again. This continues until all data is transferred or a timeout occurs. Unless the speed of DMA Modes is required, PIO mode is the best choice.

In the DMA modes, data bursting is enabled – several bytes are transferred at a time achieving a much faster throughput than PIO mode. For Non-Demand Mode (DMA enabled, Demand Mode DMA disabled), the FIFO flags are not checked during the transfer – the requested data size is transferred regardless of whether the FIFO becomes full or empty. For Rx, if the requested transfer size is greater than the actual data present in the Rx FIFO, the FIFO will be read until empty and then erroneous data will be returned for the balance of the transfer. No error will be indicated and no timeout will occur. For Tx, if the requested transfer size is greater than the available space in the Tx FIFO, data will be written until the FIFO becomes full and then the remain data will be lost. Again, no error will be indicated and no timeout will occur. Therefore, the user must ensure the requested transfer size is valid in Non-Demand mode. The FIFO Count Registers on the SIO4A/B may be used to determine the available transfer size.

For Demand Mode DMA, data will be transferred only if data is present in the Rx FIFO or space is available in the Tx FIFO. The potential problem for Demand Mode DMA occurs if a transfer timeout occurs. Due to chipset limitations, if a timeout occurs, there is no way to determine how much data was transferred before the timeout (BytesRead and Bytes Written variables will return zero), so all transferred data is potentially lost (The transferred data will be returned in the buffer, but there is no way to determine where the timeout). Therefore, the user must ensure the requested transfer size will not cause a transfer timeout in Demand mode.

SIO4_SET_DMA_INPUT_ENABLE

Sets the DMA mode for the SIO4_RECEIVE_BYTE command.

input parameters:

signed 32-bit integer	**ChannelNumber**
boolean	**EnableInputDMA**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

Data may be written from the transmit FIFOs using DMA Modes or PIO mode. If **EnableInputDMA** is FALSE, PIO Mode is selected. If **EnableInputDMA** is TRUE, a DMA mode is enabled. The type of DMA (Non-Demand Mode or Demand Mode) is selected by the SIO4_SET_DEMAND_MODE_DMA command. For a description of the DMA modes, see SIO4_SET_DEMAND_MODE_DMA.

SIO4_SET_DMA_OUTPUT_ENABLE

Enables DMA (Demand Mode or Non-Demand Mode) for transmitting data to FIFOs.

input parameters:

signed 32-bit integer	**ChannelNumber**
boolean	**EnableOutputDMA**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

Data may be written from the transmit FIFOs using DMA Modes or PIO mode. If **EnableOutputDMA** is FALSE, PIO Mode is selected. If **EnableOutputDMA** is TRUE, a DMA mode is enabled. The type of DMA (Non-Demand Mode or Demand Mode) is selected by the SIO4_SET_DEMAND_MODE_DMA command. For a description of the DMA modes, see SIO4_SET_DEMAND_MODE_DMA.

SIO4_RESET_UART

Resets the Z16C30 USC.

input parameters:
signed 32-bit integer **ChannelNumber**
output parameters:
none

Performs a hardware reset on the Z16C30 USC chip. The Z16C30 should be reset only once as the SIO4 board is initialized. After the USC has been reset, it is necessary to initialize the USC by writing "0000" to the USC Bus Configuration register. The USC registers should then be accessible.

It is important to realize that since the Z16C30 USC contains two channels, both channels will be reset when the SIO4_RESET_UART command is executed. For example, if this routine is executed for channel 2, channel 1 will also be reset.

SIO4_RESET_FIFO

Clears the selected FIFO - the Transmit FIFO, the Receive FIFO or both.

input parameters:
signed 32-bit integer **ChannelNumber**
signed 32-bit integer **ResetFIFO**
error cluster **error in**
output parameters:
signed 32-bit integer **Status**
error cluster **error out**

Resets the selected FIFO(s) and re-initialized the Almost Flag values appropriately.

ResetFIFO:
1 = TRANSMIT_FIFO
2 = RECEIVE_FIFO
3 = TX_AND_RX

SIO4_READ_FIFO_STATUS

Returns the FIFO Empty/Full status for both the transmit and receive FIFOs.

input parameters:

signed 32-bit integer	**ChannelNumber**
error cluster	**error in**

output parameters:

signed 32-bit integer	**FIFOStatus**
error cluster	**error out**

FIFOStatus returns 8 bits representing g the RX and TX FIFO Status Flags. These flags are all active low ('0' = condition true). The lower nybble (bits3-0) represent the Rx FIFO, and the upper nybble (bits 7-4) gives Tx FIFO Status. The 8 bits are as follows:

D0 = Rx Empty Flag (0 = Empty)
D1 = Rx Almost Empty Flag (0 = Almost Empty)
D2 = Rx Almost Full Flag (0 = Almost Full)
D3 = Rx Full Flag (0 = Full)
D4 = Tx Empty Flag (0 = Empty)
D5 = Tx Almost Empty Flag (0 = Almost Empty)
D6 = Tx Almost Full Flag (0 = Almost Full)
D7 = Tx Full Flag (0 = Full)

The following values represent the various valid stated for each nibble:

0xC = FIFO Empty
0xD = FIFO Almost Empty
0xF = FIFO between Almost Empty and Almost Full
0xB = FIFO Almost Full
0xC = FIFO Full
0x0 = Invalid

For example, a **FIFOStatus** of 0xCF means Rx FIFO is Empty / Tx FIFO is Full
.

SIO4_SET_SYNC_BYTE

Sets the Sync Byte in the Local GSC Sync Register.

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**SyncByte**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

Allows the user to load an integer value into the Local GSC Sync Register. The Receive line is monitored and an interrupt may be generated if a match occurs. Only the lower 8-bits of **SyncByte** are used.

SIO4_INIT_CHANNEL

Performs miscellaneous channel initialization functions.

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**Mode**
unsigned 32-bit integer	**BaudRate**
signed 32-bit integer	**RxEnable**
signed 32-bit integer	**RxDataFormat**
signed 32-bit integer	**RxDataLength**
signed 32-bit integer	**RxParity**
signed 32-bit integer	**TxEnable**
signed 32-bit integer	**TxDataFormat**
signed 32-bit integer	**TxDataLength**
signed 32-bit integer	**TxParity**
signed 32-bit integer	**TxIdleLineCondition**
boolean	**TxWaitOnUnderrun**
unsigned 16-bit integer	**RxAlmostEmpty**
unsigned 16-bit integer	**RxAlmostFull**
unsigned 16-bit integer	**TxAlmostEmpty**
unsigned 16-bit integer	**TxAlmostFull**
boolean	**RxUpperClkEnable**
boolean	**RxLowerClkEnable**
boolean	**TxUpperClkEnable**
boolean	**TxLowerClkEnable**
boolean	**RxUpperDatEnable**
boolean	**RxLowerDatEnable**
boolean	**TxUpperDatEnable**
boolean	**TxLowerDatEnable**

output parameters:

none

Mode - sets the Mode Control field of the Channel Command/Address Register (CCAR)

0 = NORMAL
1 = ECHO
2 = EXTERNAL_LOCAL_LOOPBACK
3 = INT_LOCAL_LOOPBACK

BaudRate – Attempts to set the TCR0 register to achieve the desired baud rate. The TCR0 register will be set according to the following equation:

TCR0 Value = (5000000 / **BaudRate**) -1

Since the BaudRate setup in the Z16C30 is so flexible, several assumptions were made to arrive at this BaudRate equation. The user may need to adjust the BaudRate to account for different setup.

Assumptions:

1) CTR0 input clock frequency = 20MHz.
2) CTR0 is set to divide by 4 in HCR (CTR0 output/BRG0 input is 5MHz).
3) BRG0 input is CTR0 output.
4) For Async Modes, actual baudrate will be further affected by the Data Rate settings in the CMR (16x, 32x, or 64x).

If input clock is other than 20MHz:

BaudRate = (20MHz / Input Clock freq) * Desired_BaudRate.

If CTR0 is not used (BRG input = TxC pin or RxC pin),

BaudRate = Desired_BaudRate / 4.

If using Async Mode, adjust BaudRate by Async Data Rate (from CMR),

BaudRate = (Async_Data_Rate) * Desired_BaudRate.

The BaudRate may be setup separately if more advanced setup is required. In this case, set BaudRate to 10000000 as a default value which can be reset later.

IMPORTANT! DO NOT SET THE **BAUDRATE** TO ZERO, as this may cause a driver divide-by-zero error.

RxEnable – sets the Rx Enable field of the Receive Mode Register (RMR).

0 = DISABLE_IMMED
1 = DISABLE_AFTER_TX_RX
2 = ENABLE_WO_AUTO
3 = ENABLE_WITH_AUTO

RxDataFormat – sets the Rx Data Decoding field of the RMR.

0 = NRZ
1 = NRZB
2 = NRZI_MARK
3 = NRZI_SPACE
4 = BIPHASE_MARK
5 = BIPHASE_SPACE
6 = BIPHASE_LEVEL
7 = DIFF_BIPHASE_LEVEL

RxDataLength – sets the Rx Character Length field of the RMR.

0 = 8BIT
1 = 1BIT
2 = 2BIT
3 = 3BIT
4 = 4BIT
5 = 5BIT
6 = 6BIT
7 = 7BIT

RxParity – sets the Rx Parity Enable and Rx Parity Sense field of the RMR.

0x0 = EVEN
0x1 = ODD
0x2 = SPACE
0x3 = MARK
0xF = NO_PARITY

TxEnable – sets the Tx Enable field of the Transmit Mode Register (TMR).

0 = DISABLE_IMMED
1 = DISABLE_AFTER_TX_RX
2 = ENABLE_WO_AUTO
3 = ENABLE_WITH_AUTO

TxDataFormat – sets the Tx Data Encoding field of the TMR.

0 = NRZ
1 = NRZB
2 = NRZI_MARK
3 = NRZI_SPACE
4 = BIPHASE_MARK
5 = BIPHASE_SPACE
6 = BIPHASE_LEVEL
7 = DIFF_BIPHASE_LEVEL

TxDataLength – sets the Tx Character Length field of the TMR.

0 = 8BIT
1 = 1BIT
2 = 2BIT
3 = 3BIT
4 = 4BIT
5 = 5BIT
6 = 6BIT
7 = 7BIT

TxParity – sets the Tx Parity Enable and Tx Parity Sense fields of the TMR

0x0 = EVEN
0x1 = ODD
0x2 = SPACE
0x3 = MARK
0xF = NO_PARITY

TxIdleLineCondition - sets the Tx Idle Line Condition field of the TCSR.

0 = SYNC_FLAG_NORMAL_IDLE
1 = ALTERNATE_1_AND_0_IDLE
2 = ALL_ZEROS_IDLE
3 = ALL_ONES_IDLE
4 = RESERVED_IDLE
5 = RESERVED_IDLE
6 = ALTERNATE_MARK_AND_SPACE_IDLE
7 = SPACE_IDLE
8 = MARK_IDLE

TxWaitOnUnderrun- sets the Tx Idle Line Condition field of the TCSR.

RxAlmostEmpty - sets the RxAlmostEmpty field in the GSC Rx Almost Register

RxAlmostFull - sets the RxAlmostFull field in the GSC Rx Almost Register

TxAlmostEmpty – sets the TxAlmostEmpty field in the GSC Tx Almost Register

TxAlmostFull – sets the TxAlmostFull field in the GSC Tx Almost Register

RxUpperClkEnable – sets the RxUpperClkEnable bit in the GSC Channel Control Register (Enables RxClk to be received from Upper Channel from user cable).

RxLowerClkEnable – sets the RxLowerClkEnable bit in the GSC Channel Control Register (Enables RxClk to be received from Lower Channel from user cable).

TxUpperClkEnable – sets the TxUpperClkEnable bit in the GSC Channel Control Register (Enables TxClk to be transmitted to Upper Channel of user cable).

TxLowerClkEnable – sets the TxLowerClkEnable bit in the GSC Channel Control Register (Enables TxClk to be transmitted to Lower Channel of user cable).

RxUpperDatEnable - sets the RxUpperDatEnable bit in the GSC Channel Control Register (Enables RxD to be received from Upper Channel from user cable).

RxLowerDatEnable- sets the RxLowerDatEnable bit in the GSC Channel Control Register (Enables RxD to be received from Lower Channel from user cable).

TxUpperDatEnable - sets the TxUpperDatEnable bit in the GSC Channel Control Register (Enables TxD to be transmitted to Upper Channel of user cable).

TxLowerDatEnable- sets the TxLowerDatEnable bit in the GSC Channel Control Register (Enables TxD to be transmitted to Lower Channel of user cable).

SIO4_SEND CHANNEL COMMAND

Sets the RTCmd field of the Channel Command Address Register (CCAR).

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**ChannelCommand**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

ChannelCommand:

0 = NULL_COMMAND
2 = RESET_HIGHEST_IUS
4 = TRIG_CHANNEL_LOAD_DMA
5 = TRIG_RX_DMA
6 = TRIG_TX_DMA
7 = TRIG_RX_TX_DMA
9 = RX_FIFO_PURGE.
10 = TX_FIFO_PURGE.
11 = RX_TX_FIFO_PURGE
13 = LOAD_RX_CHAR_CNT
14 = LOAD_TX_CHAR_CNT
15 = LOAD_RX_TX_CHAR_CNT
17 = LOAD_TCO0
18 = LOAD_TCO1
19 = LOAD_TC0_TC1
20 = SEL_LSB_FIRST
21 = SEL_MSB_FIRST
22 = SEL_STRAIGHT
23 = SEL_SWAPPED
25 = RX_PURGE

The Channel Command sets the RTCmd field of the CCAR – CCAR15:CCAR11. See Zilog 16C30 User's manual for further details.

SIO4_SET_TRANSMIT_CLOCK_SOURCE

Sets the Transmit Clock Source field in the Clock Mode Control Register (CMCR).

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**TxClockSource**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

TxClockSource:

0 = CLOCK_DISABLED (transmitter disabled)
1 = RXC_PIN_CLOCK
2 = TXC_PIN_CLOCK
3 = DPLL_CLOCK (Tx Output of DPLL)
4 = BRG0_CLOCK
5 = BRG1_CLOCK
6 = CTR0_CLOCK
7 = CTR1_CLOCK

The Tx Clock Source sets the input clock to the Transmitter. This routine sets bits CMCR5, CMCR4, and CMCR3 – the TxCLKSrc field of the CMCR.

SIO4_SET_RECEIVE_CLOCK_SOURCE

Sets the Receiver Clock Source field in the Clock Mode Control Register (CMCR).

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**RxClockSource**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

RxClockSource:

0 = CLOCK_DISABLED (Receiver disabled)
1 = RXC_PIN_CLOCK
2 = TXC_PIN_CLOCK
3 = DPLL_CLOCK (Rx output of DPLL)
4 = BRG0_CLOCK
5 = BRG1_CLOCK
6 = CTR0_CLOCK
7 = CTR1_CLOCK

The Rx Clock Source sets the input clock to the Receiver. This routine sets bits CMCR2, CMCR1, and CMCR0 – the RxCLKSrc field of the CMCR.

SIO4_SET_COUNTER_0_CLOCK_SOURCE

Sets the CTR0 Source field in the Clock Mode Control Register (CMCR).

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**CTR0_ClockSource**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

CTR0_ClockSource:

0 = CLOCK_DISABLED (CTR0 disabled)
1 = RXC_PIN_CLOCK
2 = TXC_PIN_CLOCK

The CTR0 Clock Source sets the input clock to Counter/Timer 0. This routine sets bits CMCR13 and CMCR12 – the CTR0Src field of the CMCR.

SIO4_SET_COUNTER_1_CLOCK_SOURCE

Sets the CTR1 Source field in the Clock Mode Control Register (CMCR).

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**CTR1_ClockSource**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

CTR1_ClockSource:

0 = CLOCK_DISABLED (CTR1 disabled)
1 = RXC_PIN_CLOCK
2 = TXC_PIN_CLOCK

The CTR1 Clock Source sets the input clock to Counter/Timer 1. This routine sets bits CMCR15 and CMCR14 – the CTR1Src field of the CMCR.

SIO4_SET_ BAUD_RATE_0_CLOCK_SOURCE

Sets the BRG0 Source field in the Clock Mode Control Register (CMCR).

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**BRG0_Source**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

BRG0_Source:

1 = RXC_PIN_CLOCK
2 = TXC_PIN_CLOCK
6 = CTR0_CLOCK
7 = CTR1_CLOCK

The BRG0 Source sets the input clock to Baudrate Generator 0. This routine sets bits CMCR9 and CMCR8 – the BRG0Src field of the CMCR.

SIO4_ SET_ BAUD_RATE_1_CLOCK_SOURCE

Sets the BRG1 Source field in the Clock Mode Control Register (CMCR).

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**BRG1_Source**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

BRG1_Source:

1 = RXC_PIN_CLOCK
2 = TXC_PIN_CLOCK
6 = CTR0_CLOCK
7 = CTR1_CLOCK

The BRG1 Source sets the input clock to Baudrate Generator 1. This routine sets bits CMCR11 and CMCR10 – the BRG1Src field of the CMCR.

SIO4_SET_ BAUD_RATE_0_MODE

Sets the BRG0 Single Cycle Mode in the Hardware Configuration Register (HCR).

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**BRG0_Mode**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

BRG0_Mode:

0 = BRG_CONTINUOUS
1 = BRG_SINGLE_CYCLE

BaudRate Generator 0 (BRG0) may or may not continue to operate after counting down to zero, depending on the BRG0S bit (HCR1). In Continuous Mode (**BRG0_Mode**=0), BRG0 will continuously reload the TC0 value automatically after counting down to zero. In Single Cycle Mode (**BRG0_Mode**=1), BRG0 will stop when it reaches zero.

SIO4_SET_ BAUD_RATE_1_MODE

Sets the BRG 1 Single Cycle Mode in the Hardware Configuration Register (HCR).

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**BRG1_Mode**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

BRG1_Mode:

0 = BRG_CONTINUOUS
1 = BRG_SINGLE_CYCLE

BaudRate Generator 1 (BRG1) may or may not continue to operate after counting down to zero, depending on the BRG1S bit (HCR5). In Continuous Mode (**BRG1_Mode**=0), BRG1 will continuously reload the TC1 value automatically after counting down to zero. In Single Cycle Mode (**BRG1_Mode**=1), BRG1 will stop when it reaches zero.

SIO4_SET_BAUD_RATE_0_ENABLE

Enables BRG0 in the Hardware Configuration Register (HCR).

input parameters:

signed 32-bit integer	**ChannelNumber**
boolean	**BaudRate0Enable**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

BaudRate Generator 0 (BRG0) is enabled by setting the BRG0E bit in the Hardware Configuration Register (HCR0). This routine sets (enables) or clears (disables) this bit.

SIO4_SET_ BAUD_RATE_1_ENABLE

Enables BRG1 in the Hardware Configuration Register (HCR).

input parameters:

signed 32-bit integer	**ChannelNumber**
boolean	**BaudRate1Enable**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

BaudRate Generator 1 (BRG1) is enabled by setting the BRG1E bit in the Hardware Configuration Register (HCR4). This routine sets (enables) or clears (disables) this bit.

SIO4_SET_PHASE_LOCKED_LOOP_SOURCE

Sets the DPLL Source field in the Clock Mode Control Register (CMCR).

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**ClockSource**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

Clock Source:

1 = RXC_PIN_CLOCK
2 = TXC_PIN_CLOCK
4 = BRG0_CLOCK
5 = BRG1_CLOCK

The Clock Source sets the input clock to the Digital Phase Locked Loop (DPLL). This routine sets bits CMCR7 and CMCR6 – the DPLLSrc field of the CMCR.

SIO4_CLEAR_DPLL_STATUS

Clear various DPLL status bits in the Channel Command/Status Register (CCSR).

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**ClearDPLLValue**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

ClearDPLLValue:

0 = CLEAR_DPLL_IN_SYNC
1 = CLEAR_DPLL_MISSING_2_CLOCKS
2 = CLEAR_DPLL_MISSING_1_CLOCK
3 = CLEAR_DPLL_ALL_STATUS

SIO4_SELECT_DPLL_RESYNC

Sets the DPLL Adjust/Sync Edge field in the Channel Command/Status Register (CCSR).

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**DPLL_Resync**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

DPLL_Resync:

0 = BOTH_EDGES
1 = RISING_EDGE
2 = FALLING_EDGE
3 = SYNC_INHIBIT

SIO4_SELECT_DPLL_DIVISOR

Sets the DPLL Clock Rate field in the Hardware Configuration Register (HCR).

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**DPLL_Divisor**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

DPLL_Divisor:

0 = DIVIDE_BY_32
1 = DIVIDE_BY_16
2 = DIVIDE_BY_8

SIO4_SET_DPLL_MODE

Sets the DPLL Mode field in the Hardware Configuration Register (HCR).

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**DPLL_Mode**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

DPLL_Mode:

0 = DPLL_DISABLED
1 = DPLL_NRZ_NRZI
2 = DPLL_DIPHASE_MARK_SPACE
3 = DPLL_BIPHASE_LEVEL

SIO4_SET_TRANSMIT_ASYNC_PROTOCOL

Sets the Transmit Protocol to Asynchronous in the Channel Mode Register (CMR).

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**ClockRate**
signed 32-bit integer	**StopBits**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

ClockRate – sets the Tx Clock Rate field in the CMR (Async Mode).

0 = DIVIDE_BY_16
1 = DIVIDE_BY_32
2 = DIVIDE_BY_64

StopBits – sets the Tx Stop Bits field in the CMR (Async Mode).

0 = ONE_STOP_BIT
2 = TWO_STOP_BITS
3 = ONE_STOP_BIT_SHAVED
4 = TWO_STOP_BITS_SHAVED

SIO4_SET_RECEIVE_ASYNC_PROTOCOL

Sets the Receive Protocol to Asynchronous in the Channel Mode Register (CMR).

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**ClockRate**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

ClockRate– sets the Rx Clock Rate field in the CMR (Async Mode).

0 = DIVIDE_BY_16
1 = DIVIDE_BY_32
2 = DIVIDE_BY_64

SIO4_SET_TRANSMIT_ISOCHR_PROTOCOL

Sets the Transmit Protocol to Isochronous in the Channel Mode Register (CMR).

input parameters:

signed 32-bit integer	**ChannelNumber**
boolean	**TwoStopBits**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

TwoStopBits – sets the Tx Two Stop Bits field in the CMR (Isochronous Mode).

SIO4_SET_RECEIVE_ISOCHR_PROTOCOL

Sets the Receive Protocol to Isochronous in the Channel Mode Register (CMR).

input parameters:

signed 32-bit integer	**ChannelNumber**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

SIO4_SET_TRANSMIT_HDLC_PROTOCOL

Sets the Transmit Protocol to HDLC in the Channel Mode Register (CMR).

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**TxUnderrun**
boolean	**TransmitPreambleEnable**
boolean	**SharedZeroFlags**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

TxUnderrun – sets the Tx Underrun Condition field in the CMR (HDLC Mode).
0 = ABORT_COND
1 = EXT_ABORT_COND
2 = FLAG_COND
3 = CRC_FLAG_COND

TransmitPreambleEnable - sets the Tx Preamble Enable field in the CMR (HDLC Mode).

SharedZeroFlags - sets the Shared Zero Flags field in the CMR (HDLC Mode).

SIO4_SET_RECEIVE_HDLC_PROTOCOL

Sets the Receive Protocol to HDLC in the Channel Mode Register (CMR).

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**AddressSearchMode**
boolean	**16BitControlEnable**
boolean	**LogicalControlEnable**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

AddressSearchMode – sets the Rx Address Search field in the CMR (HDLC Mode).

0 = DISABLED
1 = ONE_BYTE_NO_CONTROL
2 = ONE_BYTE_PLUS_CONTROL
3 = EXT_PLUS_CONTROL

16BitControlEnable - sets the Rx 16-Bit Control field in the CMR (HDLC Mode).

LogicalControlEnable - sets the Rx Logical Control Enable field in the CMR (HDLC Mode).

SIO4_SET_TRANSMIT_HDLC_SDLC_LOOP_PROTOCOL

Sets the Transmit Protocol to HDLC Loop in the Channel Mode Register (CMR).

input parameters:

signed 32-bit integer	**ChannelNumber**
signed 32-bit integer	**TxUnderrun**
boolean	**TxActiveOnPoll**
boolean	**SharedZeroFlags**
error cluster	**error in**

output parameters:

signed 32-bit integer	**Status**
error cluster	**error out**

TxUnderrun – sets the Tx Underrun Condition field in the CMR (HDLC Loop Mode).

0 = ABORT_COND
1 = EXT_ABORT_COND
2 = FLAG_COND
3 = CRC_FLAG_COND

TxActiveOnPoll - sets the Tx Active On Poll field in the CMR (HDLC Loop Mode).

SharedZeroFlags - sets the Shared Zero Flags field in the CMR (HDLC Loop Mode).