**Freescale Semiconductor**
Application Note

Document Number: AN4161
Rev. 0,8/2010

# SCI Driver for the MC9S08GW64

**by: Tanya Malik**
**Reference Design and Applications Group**
**Noida**
**India**

## Contents

1 Introduction....1
1.1 Serial Communication Interface in the MC9S08GW64....1
1.2 Clock Gating In SCI....2
1.3 Baud Rate Generation....2
2 Software Driver Description....2
2.1 sci.h....3
2.2 sci_config.h....4
2.3 sci.c....5
2.3.1 SCI_Init....5
2.3.2 SCI_Set_BaudRate....6
2.3.3 SCI_PutChar....6
2.3.4 TERMIO_PutChar....6
2.3.5 SCI_SendArray....7
2.3.6 SCI_Get_Char....7
2.3.7 TERMIO_GetChar....8
2.3.8 SCI_GetArray....8
2.3.9 Interrupt Subroutines....8
3 Assumptions....9
4 Use Case....9
5 Conclusion....10

## 1 Introduction

This document describes a driver for the Serial Communication Interface (SCI), allowing users to customize all the possible configurations for this peripheral.

The software architecture is designed to provide seamless migration between devices that have the same peripheral module.

In this application note, the driver interfaces are explained. Various applications for the MC9S08GW64 can make use of this driver. The following sections describe the details and steps for creating an application using the SCI driver.

The SCI allows full duplex, asynchronous, NRZ serial communication among the Micro Controller Unit (MCU), and remote devices including other MCUs.

## 1.1 Serial Communication Interface in the MC9S08GW64

There are four SCIs in the MC9S08GW64—SCI0, SCI1, SCI2, and SCI3.

SCI0 is designed for the Automatic Meter Reading (AMR) operation by making the SCI0 (TXD0) transmitter open and drain.

Open drain circuits are used to interface different families of devices that have different operating logic voltage levels, or to control external circuitry that requires a higher voltage level.

Therefore, the SCI0 can be compatible with devices with higher voltage levels such as 5 V devices as shown in Figure 1.

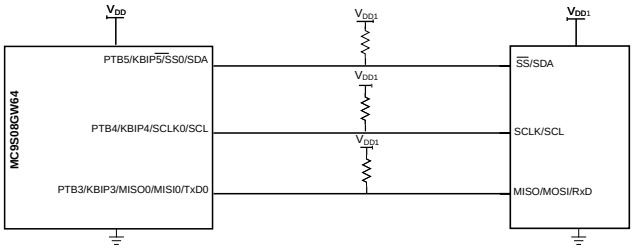


**Figure 1. Connection of SCI0 for AMR**

The SCI2 is designed with double strength to facilitate IR communication.

Both SCI1 and SCI2 can connect to the FTM channel, PCNT channel, MTIM output, and the PRACMP0 and PRACMP1 order to facilitate IR communication modulation and demodulation.

## 1.2 Clock Gating In SCI

The SCI module clock gating is controlled by SCGC1 (SCGC2_SCI0, SCGC2_SCI1, SCGC2_SCI2, SCGC1_SCI3). On **Reset** the clock is gated to all the SCI blocks.

## 1.3 Baud Rate Generation

SCI communications require the transmitter and receiver that derive baudrates from independent clock sources to use the same baudrate.

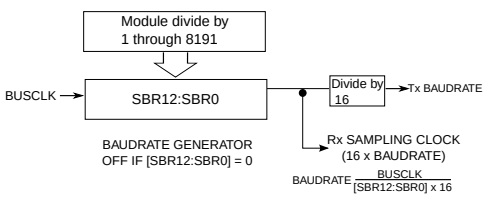


**Figure 2. Baudrate generation**

Example:

BaudRate expected is 38400 bps. The bus clock is 20 MHz. Thus, the values written in the baudrate register are calculated as:
BusClk/(BaudRate*16) = 32 SBR = 0x0020

# 2 Software Driver Description

The SCI driver is provided as C code files. You can add these files to your applications. With the integration of the SCI driver, you can now call SCI driver APIs to use the SCI functionality in your application.

There are three files associated with the SCI driver. This is a brief description:

**sci.h**—It contains all the high level APIs declarations and the various macros to be used in the functions. It defines the structure of the various SCI registers.

**sci_config.h**—This file contains the various defines to control the configuration of SCI. The user can make the changes in this file to get the required configuration.

**sci.c**—It is the main file for the driver. It contains the various high level API definitions.

## 2.1 sci.h

**NOTE**

The macros provided are passed as arguments to the respective functions to get the required configuration. It has been explained in details in Section 2.3 sci.c.

**Table 1. Defined macros**

| Marcos | Descriptions |
|---|---|
| # define SCI_Index0 | There are macros to choose between the four SCIs. The four SCIs can also be chosen simultaneously. Choose the following three macros, between the four SCIs. |
| # define SCI_Index1 | |
| # define SCI_Index2 | |
| # define SCI_Index3 | |
| | |
| # define SCI_Interrupt_Enable | Macros for enabling and disabling the interrupt for a specific SC. |
| # define SCI_Interrupt_Disable | |
| | |
| # define SCI0_RX | To indicate that the data has been received on the specific SCI port, three macros are used for the four SCIs for enabling. |
| # define SCI1_RX | |
| # define SCI2_RX | |
| # define SCI3_RX | |
| | |
| # define SCI0_TX | To indicate that the data has been transmitted from the specific SCI port. Three macros are used for the four SCIs. |
| # define SCI1_TX | |
| # define SCI2_TX | |
| # define SCI3_TX | |
| | |
| # define SCI_Set_BaudRate_9600(x) | There are macros to set the baudrate of the SCI (9600 / 19200 / 38400 / 115200 bps). x is the SCI where the baudrate is to be set. |
| # define SCI_Set_BaudRate_19200(x) | |
| # define SCI_Set_BaudRate_38400(x) | |
| # define SCI_Set_BaudRate_115200(x) | |
| | |
| unsigned char* SCI0_Data | Global variables are also defined, used globally, and can be accessed by any function. There are four global char pointers used to store the data for SCI0 Tx and Rx in case of an interrupt. |
| unsigned char* SCI1_Data | |
| unsigned char* SCI2_Data | |
| unsigned char* SCI3_Data | |
| | |
| unsigned char SCI0_DataLen | There are four char variables to store the length of the data to be transmitted or read from the SCI. |
| unsigned char SCI1_DataLen | |
| unsigned char SCI2_DataLen | |
| unsigned char SCI3_DataLen | |

## 2.2 sci_config.h

This file contains the various defines to control the configuration of the SCI. The user can make the changes in this file to get the required configuration.

Macro used to define the value of the bus clock.

- **# define BUS_CLK**

**NOTE**

It does not change the bus clock of the device. It only sets the value of the macro which is used in the calculation of baudrate.

Example:
# define BUS_CLK 20000000 defines the bus clock to be 20 MHz, therefore the baudrate is calculated assuming the bus clock to be 20 MHz

- **# define UART_SEL**

  It is a macro to define the SCI port used for printf function. The printf function sends the string to the SCI defined by UART_SEL.
  Example:
  # define UART_SEL SCI_Index3 selects SCI3 for printf function

**NOTE**

No macro is provided for SCI0 pin muxing because there is no pin muxing in SCI0. Only port B is used with Pin B3 for TXD0, pin B2 for RXD0.

The following macros are used for pin muxing SCI1:

- **# SCI1_TxRx_PortC**

  It is used to configure pin C5 for TXD1 and C4 for RXD1

  Example:
  # define SCI1_TxRx_PortC 1

- **# SCI1_TxRx_PortB**

  It is used to configure pin B1 for TXD1 and B0 for RXD1
  Example:
  # define SCI1_TxRx_PortB 1

**NOTE**

Both SCI1_TxRx_PortC and SCI1_TxRx_PortB cannot be 1 at the same time. It will give an error.

The following macros are used for pin muxing SCI2

- **# define SCI2_TxRx_PortA**

  It is used to configure pin A5 for TXD2 and A4 for RXD2
  Example:
  # define SCI2_TxRx_PortA 1

- **# define SCI2_TxRx_PortB**

  It is used to configure pin B7 for TXD2 and B6 for RXD2
  Example:

# define SCI2_TxRx_PortB 1

**NOTE**

Both SCI2_TxRx_PortA and SCI2_TxRx_PortB cannot be 1 at the same time. It gives an error.

The following macros are used for pin muxing SCI3

- **# define SCI3_TxRx_PortC**

  It is used to configure pin C7 for TXD3 and C6 for RXD3
  Example:
  # define SCI3_TxRx_PortC 1

- **# define SCI3_TxRx_PortG**

  It is used to configure pin G5 for TXD3 and G4 for RXD3k
  Example:
  # define SCI3_TxRx_PortG 1

**NOTE**

Both SCI3_TxRx_PortC and SCI3_TxRx_PortG cannot be one at the same time. It gives an error.

## 2.3 sci.c

This file contains the definition of various functions.

### 2.3.1 SCI_Init

Description:

This function is used to initialize the specific SCI by configuring the internal registers. It enables the transmitter and receiver of the specific SCI and sets the baudrate to a value of 38400 bps when the bus clock is assumed to be 20 MHz.

Prototype:

void SCI_Init ((unsigned char SCI_Index,void (*p)(unsigned char1, unsigned char2);

Input parameters:

- SCI_Index—To select the SCI to be initiated
  SCI_Index0
  SCI_Index1
  SCI_Index2
  SCI_Index3
- p—Function is used only in case of interrupts. The user can pass the address of the callback function or pass zero.
- char1—Is passed as an argument to the callback function and specifies whether the interrupt is due for transmission or reception.
- char2—Passes the data received (in case of reception) or pass 0 (in case of transmission) to the callback function

Output parameters:

None

Example:

```
void func (unsigned char, unsigned char)
{}

SCI_Init (SCI_Index1, &func);
```

Initializes SCI1 and passes the address of a function for the callback

## 2.3.2 SCI_Set_BaudRate

Description:
This function is used internally by calling it in other function. This function is used to set the required baudrate of a specific SCI.

Prototype:
void SCI_Set_BaudRate (unsigned char SCI_Index, unsigned int Baud_Rate)

Input parameters:
1. SCI_Index—Selects the SCI to be initiated
   SCI_Index0, SCI_Index1, SCI_Index2, SCI_Index3

2. Baud_Rate—The baudrate to be set.

Output parameters:
None

## 2.3.3 SCI_PutChar

Description:
This function sends one byte of data to the specific SCI port.

Prototype:
void SCI_PutChar (unsigned char SCI_Index, unsigned char Send_Data)

Input parameters:
1. SCI_Index—Selects the SCI to be initiated
   SCI_Index0, SCI_Index1, SCI_Index2, SCI_Index3

2. Send_Data—The one byte data to be sent

Output parameters:
None

Example:
SCI_PutChar(SCI_Index1, 0xAA);
Sends a char 0xAA to SCI1

**NOTE**

Enabling the interrupt is not available while sending a char. The interrupt can only be enabled while sending an array.

## 2.3.4 TERMIO_PutChar

Description:
This function sends one byte of data to the specific SCI port selected by UART_SEL. It is used in printf function.

Prototype:
void TERMIO_PutChar (unsigned char Character)

Input parameters:
1. Character—Sends one byte of data to the SCI selected by the macro UART_SEL

Output parameters:
None

Example:
# define UART_SEL SCI_Index0
TERMIO_PutChar(0xAA);
It sends the character 0xAA to SCI0.

## 2.3.5 SCI_SendArray

Description:
This function sends the data array to the specific SCI port.

Prototype:
void SCI_SendArray(unsigned char SCI_Index, unsigned char interrupt_enable, unsigned char* array, unsigned char length)

Input parameters:
1. SCI_Index—To select the SCI to be initiated
   SCI_Index0, SCI_Index1, SCI_Index2, SCI_Index3
2. interrupt_enable—To enable or disable the interrupt for Tx and Rx
3. array—Data array to be transmitted to the selected SCI port
4. length—Length of the array to be sent

Output parameters:
None

Example:
SCI_SendArray (SCI_Index1, SCI_Interrupt_Disable, Data_Array, 10);
Sends an array of length 10 to SCI1 with interrupt disabled.

## 2.3.6 SCI_Get_Char

Description:
This function reads and returns the data (1 byte) read from the selected SCI port.

Prototype:
unsigned char SCI_GetChar(unsigned char SCI_Index)

Input parameters:
1. SCI_Index—Selects the SCI to be initiated
   SCI_Index0
   SCI_Index1
   SCI_Index2
   SCI_Index3

Output parameters:
Returns the data byte read from the selected SCI port.

Example:
unsigned char data;
data = SCI_GetChar(SCI_Index1);
Returns the char read from SCI1

**NOTE**

Enabling the interrupt is not available in this software driver while receiving a char. The interrupt can only be enabled for receiving an array.

## 2.3.7 TERMIO_GetChar

Description:
This function reads one byte of data from the specific SCI port selected by UART_SEL. It is used in the printf function.

Prototype:
unsigned char TERMIO_GetChar ( )

Input parameters:
None

Output parameters:
Returns the data byte read from the specific SCI selected by UART_SEL

Example:
#define UART_SEL SCI_Index0
unsigned char data;
data = TERMIO_GetChar();
Returns the data read from SCI0.

## 2.3.8 SCI_GetArray

Description:
This function sends the data array to the specific SCI port.

Prototype:
void SCI_GetArray(unsigned char SCI_Index, unsigned char interrupt_enable, unsigned char* array, unsigned char length)

Input parameters:
1. SCI_Index—To select the SCI to be initiated
   SCI_Index0, SCI_Index1, SCI_Index2, SCI_Index3
2. interrupt_enable—To enable or disable the interrupt for Tx and Rx
3. array—Stores the data array received
4. length—Length of the array to be sent

Output parameters:
None

Example:
SCI_GetArray (SCI_Index1, SCI_Interrupt_Disable, Data_Array, 10);
It gets an array of length 10 from the SCI with the interrupt disabled.

## 2.3.9 Interrupt Subroutines

There are three types of interrupts:
- Transmit Data Register Empty
- Transmission Complete
- Receive Data Register Full

**SCITX0_ISR**

Description:
It is an interrupt subroutine for SCI0. The function is executed when either of the two interrupt occurs; Transmit Data Register Empty or Transmission Complete Provided. The respective interrupts are then enabled. The routine clears the interrupt and sends the data to the SCI0 port.

Prototype:
void interrupt VectorNumber_Vsci0tx SCITX0_ISR()

Input parameters:
None

Output parameters:
None

There are similar interrupt subroutines for SCI1, SCI2, SCI3:
SCITX1_ISR for SCI1
SCITX2_ISR for SCI2
SCITX3_ISR for SCI3

**SCIRX0_ISR**

Description:
It is the interrupt subroutine for SCI0. The function is executed when the following interrupt occurs Receive Data Register Full, provided that the respective interrupts are enabled. The routine clears the interrupt and takes the required action by receiving the data from SCI0.

Prototype:
void interrupt VectorNumber_Vsci0rx SCIRX0_ISR()

Input parameters:
None

Output parameters:
None

There are similar interrupt subroutines for SCI1, SCI2, SCI3:
SCIRX1_ISR for SCI1
SCIRX2_ISR for SCI2
SCIRX3_ISR for SCI3

# 3 Assumptions

The descriptions in this document assumes the person reading it has full knowledge of all the configuration registers of all the blocks in the MC9S08GW64, especially the SCI and Internal Clock Source (ICS) blocks.

# 4 Use Case

Assuming that the clock settings are done and the bus clock is running on 20 Mhz. Include the sci.c in the main file.

To initialize the respective SCI1 with the desired configuration and establish communication:

Step 1—If you want to choose port C for SCI, make this setting in sci_config.h:
**# SCI1_TxRx_PortC 1**

Step 2—Suppose there is a callback function defined as:

```
void callback(unsigned char a, unsigned char b)
{
  return;
}
```

**SCI_Init (SCI_Index1, & callback);**
This initialized the SCI1 by enabling the transmitter and receiver. The address of callback function is passed which is used in case of interrupts.The default baudrate is set as 38400 bps.

Step 3—The baudrate can be changed by calling:
**SCI_Set_BaudRate_9600(SCI_Index1);**
It sets the baudrate of SCI1 to 9600bps

Step 4—To send data, the following function is called:
**SCI_PutChar(SCI_Index1, 0xAA);**
It sends a char 0xAA to SCI1 with the interrupt disabled by default.

# 5 Conclusion

This driver provides a software base for applications that need the implementation of SCI.

## *How to Reach Us:*

**Home Page:**
www.freescale.com

**Web Support:**
http://www.freescale.com/support

**USA/Europe or Locations Not Listed:**
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

**Europe, Middle East, and Africa:**
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

**Japan:**
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

**Asia/Pacific:**
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

***For Literature Requests Only:***
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com



Document Number: AN4161
Rev. 0,8/2010