Freescale Semiconductor
Application Note

Document Number: AN4158
Rev. 0, 8/2010

IIC Driver for the MC9S08GW64

by: Tanya Malik
Reference Design and Applications Group
Noida
India

Contents

1 Introduction

This document describes a driver for the Inter Integrated Circuit (IIC). This allows users the customization of all the possible configurations for this peripheral.

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

In this application note, the driver interfaces are explained. Various applications for the MCS08GW64 can make use of this driver. The following sections describe the details and steps for creating an application using the IIC driver.

The inter-integrated circuit (IIC) provides a method of communication between a number of devices. The interface can operate up to 100 kbps with maximum bus loading and timing.

1.1 IIC in the MC9S08GW64

There is one IIC in the MC9S08GW64. It is two-wire communication consisting of:

- SCL—Serial Clock Line

The bidirectional SCL is the serial clock line of the IIC system.

- SDA—Serial Data Line

The bidirectional SDA is the serial data line of the IIC system.

© 2010 Freescale Semiconductor, Inc.

1.2 Clock Gating In IIC

The bus clock to the IIC can be gated on and off using the IIC bit in SCGC1. This bit is cleared after any reset, which disables the bus clock to this module.

1.3 Baudrate Calculation

Baudrate is calculated by setting the following bits of the IICF register, IIC Multiplier Factor (IICF_MULT), and the IIC Clock Rate (IICF_ICR).

Table 1. IIC Multiplier Factor

IICF_MULT	MUL
00	1
01	2
10	4
11	Reserved

Table 2. IIC Clock Rate

ICR (hex)	SCL Divider	ICR (hex)	SCL Divider
00	20	20	160
01	22	21	192
02	24	22	224
03	26	23	256
04	28	24	288
05	30	25	320
06	34	26	384
07	40	27	480
08	28	28	320
09	32	29	384
0A	36	2A	448
0B	40	2B	512
0C	44	2C	576
0D	48	2D	640
0E	56	2E	768
0F	68	2F	960
10	48	30	640
11	56	31	768
12	64	32	896
13	72	33	1024
14	80	34	1152

ICR (hex)	SCL Divider	ICR (hex)	SCL Divider
15	88	35	1280
16	104	36	1536
17	128	37	1920
18	80	38	1280
19	96	39	1536
1A	112	3A	1792
1B	128	3B	2048
1C	144	3C	2304
1D	160	3D	2560
1E	192	3E	3072
1F	240	3F	3840

The calculation is done as follows:

IIC baud rate = bus speed (Hz)/(mul × SCL divider)

2 Software Driver Description

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

There are three files associated with the IIC driver. The following is a brief description of them:

- **iic.h**— It contains all the high level API declarations and the various macros to be used in the functions. It defines the structure of the various IIC registers.
- **iic_config.h**—This file contains the various defines to control the configuration of IIC. The user can make the changes in this file to get the required configuration.
- **iic.c**—It is the main file for the driver. It contains the various high level API definitions.

2.1 iic.h

NOTE

The macros provided are passed as arguments to the respective functions to get the required configuration. It is explained in detail in this section.

Table 3. Defined macros

Macros	Definitions
#define IIC_INTERRUPT_ENABLE	Enables or disables the IIC interrupt
#define IIC_INTERRUPT_DISABLE	
#define IIC_10_BIT_SLAVE_ADDR	Selects between the 7-bit slave address and a 10-bit slave address
#define IIC_7_BIT_SLAVE_ADDR	

Macros	Definitions
#define IIC_MULT_1	Selects the multiplier used to calculate the IIC baud rate
#define IIC_MULT_2	
#define IIC_MULT_4	
#define IIC_MWSR // (Master write slave read)	Defines if the command is master write or master read
#define IIC_MRSW // (Master read slave write)	
#define IIC_Enter_TxMode()	Defined to configure the master in transmit mode
#define IIC_Enter_RxMode()	Defined to configure the master in receive mode
#define IIC_Send_Start()	Defined to send the start symbol for starting the Tx/Rx transaction
#define IIC_Repeated_Start()	Defined to send a repeated start symbol on SDA for continuing data transfer without bus retention loss
#define IIC_Send_Stop()	Defined to send the STOP signal through the SDA to stop the Tx/Rx transaction and release the IIC bus
#define IIC_Enable_Ack()	Defined to enable the send acknowledge procedure by configuring the internal registers
#define IIC_Disable_Ack()	Defined to disable the send acknowledge procedure by configuring the internal registers
#define IIC_Write_Byte(data)	Defined for the master write transaction. Data is the data byte to be sent from master to slave

2.2 iic_config.h

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

The following macros are used for pin muxing in IIC:

#define IIC_SDA_SCA_PortA 1

It is used to configure pin A0 as SCA and pin A1 as SDA

#define IIC_SDA_SCA_PortB 1

It is used to configure pin B4 as SCA and pin B5 as SDA

NOTE

Both IIC_SDA_SCA_PortA and IIC_SDA_SCA_PortB cannot be 1 at the same time. It gives an error.

2.3 iic.c

This file contains the definition of various functions.

2.3.1 IIC_Init

Description:

This function is used to initialize IIC by configuring the internal registers. The user can enable or disable the interrupt and select the number (no) of bits for the slave address.

Prototype:

void IIC_Init(unsigned char Interrupt_Enable,unsigned char Addr_No_Of_Bits,void (*p)(void))

Input parameters:

1. Interrupt_Enable—To enable or disable the interrupt using the following macros:
   IIC_INTERRUPT_ENABLE
   IIC_INTERRUPT_DISABLE

2. Addr_No_Of_Bits—The user can select between a 7-bit slave address or 10-bit slave address using the macros:
   IIC_10_BIT_SLAVE_ADDR
   IIC_7_BIT_SLAVE_ADDR

3. p—The user can pass the address of the callback function if you want to execute the function in case of interrupts

Output parameters:

None

Example:

```
void iic_callback() {}

IIC_Init(IIC_INTERRUPT_DIABLE, IIC_7_BIT_SLAVE_ADDR, & iic_callback)
```

2.3.2 IIC_Set_BaudRate

Description:

This function is used to set the baudrate for the IIC by setting the multiplier and SCL divider.

Prototype:

void IIC_Set_Baud_Rate(unsigned char Multiplier, unsigned char SCL_Divider)

Input parameters:

1. Multiplier—The user can select the multiplier from the following macros:
   III_MULT_1
   III_MULT_2
   III_MULT_3
   III_MULT_4

2. SCL_Divider—The user can pass the required value of the SCL divider. Explained in Section 1.3 Baudrate Calculation

Output parameters:

None

Example:

```
IIC_Set_Baud_Rate(IIC_MULT_1, 14);
```

Baudrate is set to the Bus Clock/(IIC_MULT_1* 14)

2.3.3 IIC_Wait_Ack

Description:
This function is called to wait for the acknowledgement. It waits on the IIC flag and clears it afterwards.

Prototype:
void IIC_Wait_Ack(void)

Input parameters:
void

Output parameters:
None

Example:
IIC_Wait_Ack();

2.3.4 IIC_Start_Transmission

Description:
This function starts the transaction of read and write with a slave.

Prototype:
void IIC_Start_Transmission(unsigned int SlaveAddress, unsigned char TxRxMode)

Input parameters:
1. SlaveAddress—Enter the address of the slave.
2. TxRxMode—To select if it is master, write or read using the following macros:
   IIC_MWSR
   IIC_MWSW

Output parameters:
None

Example—Let slave address be 0x50
IIC_Start_Transmission(0x50,MWSR)

2.3.5 IIC_ISR

Description:
It is the interrupt subroutine executed when IIC interrupt occurs. If the user has passed the address of the callback function in IIC_Init then the callback function is executed after execution of the ISR.

NOTE

This interrupt subroutine is executed only if the IIC interrupt is enabled.

Prototype:
void interrupt VectorNumber_Viic IIC_ISR()

Input parameters:
void

Output parameters:
None

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 IIC 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 iic.h in the main file.

Step 1—Make the required changes in iic_config.h to get the required configuration. To configure pin B4 as SCA and pin B5 as SDA make the following setting in iic_config.h:
#define IIC_SDA_SCA_PortB 1

Step 2—Define the callback function required in case of an interrupt:

```
void iic_callback(unsigned char a , unsigned char b)
{
   //write the action to be taken in case of interrupt
   return;
}
```

Step 3—Initialize the IIC module with the required configuration:
IIC_Init(IIC_INTERRUPT_DISABLE,IIC_7_BIT_SLAVE_ADDR,&iic_callback);

The interrupt is then disabled, the slave address is set to 7-bit, and the address of the callback function is passed.

Step 4—IIC is ready to be used. To start the communication with the salve (for example: EEPROM) Call the function as shown:
IIC_Start_Transmission(Slave address,write_bit);

Step 5—Wait for the transfer complete flag to be set. When the flag is set, the communication is established.
IIC_Wait_Ack();

5 Conclusion

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

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

Information in this document is provided solely to enable system and sofware implementers to use Freescale Semiconductors products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor prodcuts are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claims alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-complaint and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to http://www.freescale.com/epp.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2010 Freescale Semiconductor, Inc.

Document Number: AN4158
Rev. 0, 8/2010