Freescale Semiconductor
Application Note

AN3048
Rev. 1.00, 11/2005

Using IIC to Read ADC Values on MC9S08QG8

by Donnie Garcia
Application Engineering
Microcontroller Division

1 Introduction

The MC9S08QG8 is a small pin-count, yet feature rich, microcontroller. The properties of this MCU make it ideal for an application such as a stand-alone inter-integrated circuit (IIC) analog-to-digital converter (ADC).

This document demonstrates how the MC9S08QG8 can be used as a stand-alone ADC accessed by the IIC. In this demonstration application, commands sent via IIC will select different ADC channels and set a threshold limit to be used for an alarm indicator.

Table of Contents

1.1 Requirements

To run this demo, you need:

- Two DEMO9S08QG8 demo boards
- DEMO9S08QG8 Quick Start Guide (comes with demo board)
- MC9S08QG8 data sheet from freescale.com.
- IIC driver code used in this document — available as AN3048SW.zip from freescale.com.
- True-Time Simulator CW3.1 with MC9S08QG8 patch

freescale semiconductor

1.2 IIC Features

The IIC communications protocol is a standard that can be used to interface an MCU with other devices such as temperature sensors, EEPROMS, LCDs, other MCUs, and A/D or D/A devices. This protocol makes it easy to add additional devices onto an existing IIC bus because each device contains a unique slave address. All devices on the IIC bus monitor activity, but slaves only respond to messages that are addressed to them. IIC communication is accomplished through a simple two-line interface. Data is relayed through the SDA pin, and the clock is supplied by the master on the SCL pin.

The MC9S08QG8 contains the standard IIC module that is in the HCS08 Family of microcontrollers. The features of this module include:

- Compatibility with the IIC bus standards
- Multi-master operation
- Flexible baud rate generator
- Interrupt-driven byte-by-byte transfer

The interrupt capabilities of this peripheral provide a very important feature that allows interrupts to occur at address match, arbitration loss, and data transfers. All of these features together allow IIC communication to be handled by an interrupt service routine (ISR) that can be used for both master and slave operation.

NOTE

For additional information on IIC module initialization, functionality, and registers and control and status bits, please see the IIC chapter of the MC9S08QG8 data sheet (available from freescale.com).

1.3 Implementing IIC Communication

The IIC chapter of the MC9S08QG8 data sheet provides an “IIC Module Quick Start” figure and “Typical IIC Interrupt Routine” flow chart. These detail the most straight-forward method to implement IIC communications. Because the application described in this document uses that method, please have the IIC chapter of the MC9S08QG8 data sheet near for reference.

The IIC master and slave code examples for this demonstration (AN3048SW.zip) use the documented method to create the interrupt service routine (ISR) for IIC. For example, the first decision in the “Typical IIC Interrupt Routine” flow chart is whether the device is set as a master or slave. This is accomplished in the example ISR with the code:

```
if (IICC_MST){ ... } else {...}
```

This checks the status of the MST bit in the IIC control register:

- If it is set (MST = 1), the code will proceed to execute what is defined inside the curly brackets, i.e., master mode operation will be handled.
- If MST is not set (MST = 0), code will execute the *else* statement, i.e., slave mode operation will be handled.

It is important to note that not all decisions that are defined in this flow chart can be made by the status and control bits that are provided in the IIC registers. For example, when in master mode, you must know when the second-to-last byte and the last byte to be read are received. This is accomplished in the example

ISR by a variable, *num_to_rec*, which defines the number of bytes to receive. This variable is used to compare against a count variable that keeps track of the amount of data that has passed from slave to master. So in the example code provided, you will see that variables are used along with status and control bits to achieve the routine depicted in the IIC chapter of the MC9S08QG8 data sheet.

1.3.1 Master and Slave Initialization

Both master and slave can use the same ISR routine as shown in the demo code, but there is a difference in the initialization and startup sequence between these two. See the “IIC Module Quick Start” figure in the IIC chapter of the MC9S08QG8 data sheet for complete initialization sequence.

As soon as the slave device is initialized, it is ready to receive IIC data. A master device must initiate communication by setting up a master transmit or a master receive. This is done in the addressing sequence because the LSB of the address is the R/W bit. So, to initiate IIC communication, the address along with the R/W bit must be written to the IIC data register by the master device.

1.4 Defining the Messaging Strategy

The goal of this application is to get an analog reading from a MC9S08QG8 MCU via IIC communication. The master MCU will be allowed to select the ADC channel and set a threshold value that can be used to compare against. A very important step in accomplishing this is defining the messages that will be sent and received. For this application, the following messaging strategy was used.

1.4.1 Master-to-Slave Communication

Table 1 shows the messages that will pass from master to slave. All possible values are shown in the data range.

Table 1. Master-to-Slave Communication

Byte	Address (upper 7bits) R/W(1bit)	Threshold	Threshold	Channel Select
	1	2	3	4
Data Range	Address (0xFE–0x00), R/W (0x1–0x0)	0x03–0x00	0xFF–0x00	0x01–0x00
Description	Concatenation of address and read/write	Upper byte of a 10-bit threshold	Lower byte of a 10-bit threshold	A/D channel to be read

1.4.2 Slave-to-Master Communication

Table 2 shows all data that will pass from slave to master. Not shown here is the address that must first pass from master to slave; all IIC communication is initiated by the master.

Table 2. Slave-to-Master Communication

Byte	Result	Result	Alarm
	1	2	3
Data Range	0x03-0x00	0xFF-0x00	0x01-0x00
Description	Upper byte of a 10-bit result	Lower byte of a 10-bit result	Status of the compare to the threshold

With the messaging strategy defined, software development is as simple as parsing the data that is received by the master or slave. The ISR that has been created is pre-configured to create receive arrays with the data. Both slave and master contain a receive data array that can be examined while the IIC bus is inactive. It is important to wait for the IIC bus to be inactive to ensure a valid data reading. The arrays allow easy access to the messages that have been received, and C code is used to look at individual bytes in the receive data array. For example, the Alarm byte is examined with the following code:

```
if (IIC_Rec_Data[2] == 1) {

//Turn on Alarm Indicator
Alarm = 1;
} else {

Alarm =0;
}
```

This code accesses the Alarm byte in the receive data array using an “if” statement. The same method can be used on the other bytes in the receive array so that the application code can complete the necessary tasks.

2 Demonstrating the Stand-Alone ADC Application

To demonstrate this application, you will need to connect and configure the two MC9S08QG8 demo boards. Designate one of the boards to be the master board, and the other as the slave board.

2.1 Slave Board

Program the slave code (QG_IIC_Slave.c) into the slave board. Follow the directions for programming the demo board provided by the quick start guide included with the demo board. This should guide you step-by-step through opening a project, connecting through USB, and programming a demo board. This board must be powered externally, so after you have programmed it, connect an external power supply (9 V, nominally) to the power connector.

2.2 Master Board

The master code should be programmed into the master board. To visualize our demonstration, we will use the True-Time Simulator interface and a visualization tool. So, after the board is programmed, we will be using the debug environment.

The master board must be modified to contain pullup resistors on the SDA and SCL pins. The IIC driver code in AN3048SW.zip uses PTB6 and PTB7 as the IIC pins. This is configured in software by setting the IICPS bit in SOPT2. This is a useful feature of the MC9S08QG8 that allows the IIC pins to be switched from PTA2/PTA3 to PTB6/PTB7. For IIC communication to operate, you must connect pullups on the PTB6 and PTB7 pins.

2.3 Connecting the Demo Boards

After the pullups have been added to the PTB6 and PTB7 pins on the master board, three connections must be made, as shown in Figure 1:

- PTB6, PTB7, and GND must be connected between the two boards.
- GND must be connected; connect GND from J1 pin 3 to J1 pin 3 of the slave board.
- PTB7 and PTB6 are J1 pins 8 and 10, respectively; connect them as shown in Figure 1.

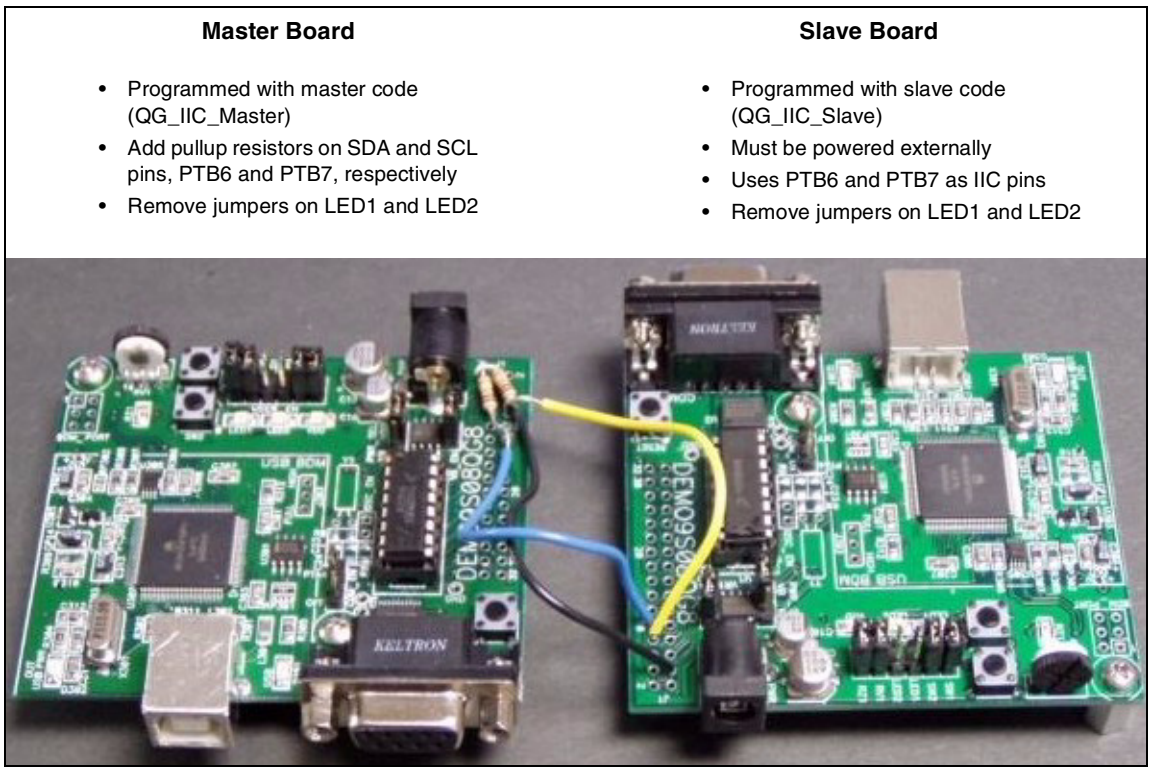

Figure 1. Connecting the Two MC9S08QG8 Demo Boards

2.4 Starting the Demonstration

1. Turn on the slave board by moving the PWR_SEL jumpers to the VDD side.
2. Press the green run arrow in the True-Time Simulator to begin the master code (see Figure 2).

   In the visualization tool, the LED shows the status of the ALARM indicator; the graph is a visual representation of the ADC data. You can switch between channel 1 and channel 0 by pressing the SW1 (channel 0) or SW2 (channel 1) on the master demo board:

   — When channel 0 is selected, the potentiometer of the slave board is being read.

   Vary the potentiometer to see the result on the graphical representation (Figure 2) change.

   — When channel 1 is selected, the photo sensor is being read.

   Cover the photo sensor to see the graphical representation (Figure 2) change.

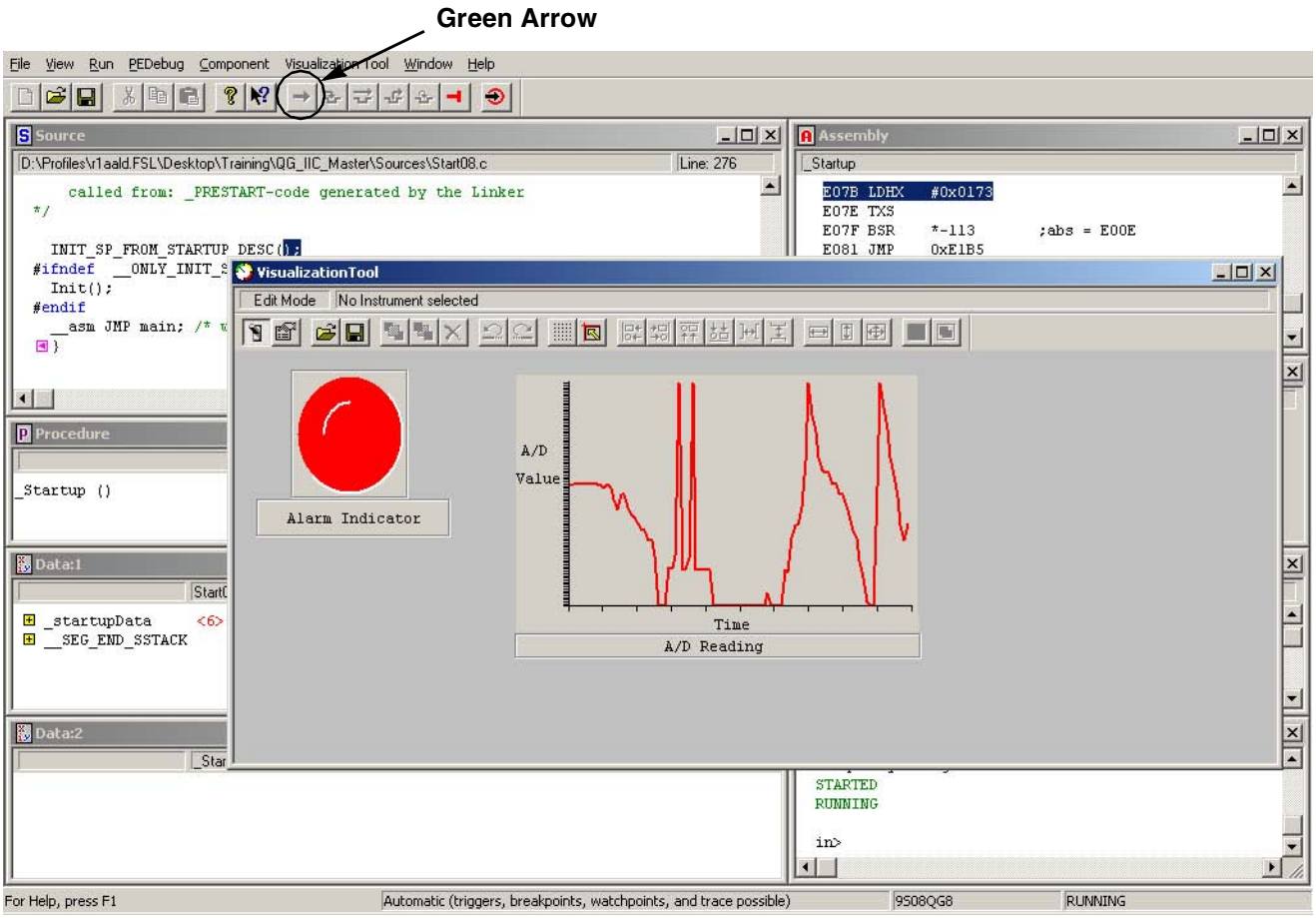

Figure 2. Visualization Tool in True-Time Simulator & Real-Time Debugger

3 Conclusion

This document and provided application code (AN3048SW.zip) demonstrate how the MC9S08QG8 is well suited for an IIC application. The provided IIC driver code and application example can be used to accelerate understanding of IIC and development time.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

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)
support@freescale.com

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 Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

AN3048
Rev. 1.00, 11/2005

freescale™ semiconductor