Freescale Semiconductor
Application Note

Document Number: AN4169
Rev. 2, 2010

ADC Driver for MC9S08GW64

by: Tanya Malik
Microcontroller Solutions Group
Noida
India

Contents

1 Introduction

ADC converts an input analog voltage (or current) to a digital number proportional to the magnitude of voltage (or current).

This document describes a driver for analog-to-digital converter (ADC), allowing users the customization of all the possible configurations for this peripheral.

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

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

1.1 ADC in MC9S08GW64

The MC9S08GW64 series includes two separately controllable 16-bit ADCs—ADC0 and ADC1. The 16-bit analog-to-digital converter is a successive approximation ADC designed for operation within the integrated microcontroller system-on-chip.

1.2 ADC clock gating

The bus clock to each ADC can be gated on or off using the SCGC1_ADC0 / SCGC1_ADC1 bits of “System clock gating control 1 register”. On reset the clock is gated to the ADC modules.

Table 1. SCGC1 Register Field Description

Field	Description
7 ADC1	**ADC1 Clock Gate Control**—This bit controls the clock gate to the ADC1 module. 0 Bus clock to the ADC1 module is disabled. 1 Bus clock to the ADC1 module is enabled.
6 ADC0	**ADC0 Clock Gate Control**—This bit controls the clock gate to the ADC0 module. 0 Bus clock to the ADC0 module is disabled. 1 Bus clock to the ADC0 module is enabled.
5 KBI	**KBI Clock Gate Control**—This bit controls the clock gate to the KBI module. 0 Bus clock to the KBI module is disabled. 1 Bus clock to the KBI module is enabled.
4 IIC	**IIC Clock Gate Control**—This bit controls the clock gate to the IIC module. 0 Bus clock to the IIC module is disabled. 1 Bus clock to the IIC module is enabled.
3 SCI3	**SCI3 Clock Gate Control**—This bit controls the clock gate to the SCI3 module. 0 Bus clock to the SCI3 module is disabled. 1 Bus clock to the SCI3 module is enabled.
2 SCI2	**SCI2 Clock Gate Control**—This bit controls the clock gate to the SCI2 module. 0 Bus clock to the SCI2 module is disabled. 1 Bus clock to the SCI2 module is enabled.
1 SCI1	**SCI1 Clock Gate Control**—This bit controls the clock gate to the SCI1 module. 0 Bus clock to the SCI1 module is disabled. 1 Bus clock to the SCI1 module is enabled.
0 SCI0	**SCI0 Clock Gate Control**—This bit controls the clock gate to the SCI0 module. 0 Bus clock to the SCI0 module is disabled. 1 Bus clock to the SCI0 module is enabled.

1.3 Signal description

Table 2. Signal Description

Name	Function
DADP0-DADP1	Differential analog channels input
DADM0-DADM1	Differential analog channels input
AD2-AD14	Single ended analog channel inputs
V_{REFH}	External voltage reference high
V_{REFL}	External voltage reference low
V_{DDA}	Analog power supply
V_{SSA}	Analog ground

1.3.1 Differential and single ended analog channels

The ADC module supports up to two pairs of differential inputs and 14 single-ended inputs. Each differential pair requires two inputs (DADP0 / DADM0) and (DADP1 / DADM1). The ADC also requires four supply / reference / ground connections.

1.3.2 Analog power (V_{DDA})

The ADC analog portion uses V_{DDA} as its power connection. In some packages, V_{DDA} is connected internally to V_{DD}. If externally available, connect the V_{DDA} pin to the same voltage potential as V_{DD}.

1.3.3 Analog ground (V_{SSA})

The ADC analog portion uses V_{SSA} as its ground connection. In some packages, V_{SSA} is connected internally to V_{SS}. If externally available, connect the V_{SSA} pin to the same voltage potential as V_{SS}.

1.3.4 Voltage reference

Voltage reference can have two values—voltage reference select high (V_{REFSH}) and voltage reference select low (V_{REFSL}).

V_{REFSH} is the high reference voltage for the converter. The ADC can be configured to accept one of three voltage reference pairs for V_{REFSH}. The three pairs are external (V_{REFH} and V_{REFL}), alternate (V_{ALTH} and V_{ALTL}) and the internal band gap (V_{BGH} and V_{BGL}). These voltage references are selected using the REFSEL bits in the ADCSC2 register.

V_{REFSL} is the low reference voltage for the converter. The ADC can be configured to accept one of three voltage reference pairs for V_{REFSL}. The three pairs are external (V_{REFH} and V_{REFL}), alternate (V_{ALTH} and V_{ALTL}) and the internal bandgap (V_{BGH} and V_{BGL}). These voltage references are selected using the REFSEL bits in the ADCSC2.

1.4 ADC channels

Single ended analog channels

There are 14 single ended analog channels. A single-ended input is selected for conversion through the ADCH channel select bits when the DIFF bit in the ADCSC1 register is low, that is, when differential mode is disabled.

Differential analog channels

There are two pairs of differential channels. Each differential analog input is a pair of external pins (DADP1/DADM1) and (DADP0/DADM0). A differential input is selected for conversion through the ADCH channel select bits when the DIFF bit in the ADCSC1 register bit is high, that is, when differential mode is enabled.

Table 3. ADC Channel Assignments

ADCH	Input function of ADC0	Input function of ADC1
0000	DADP0/DADM0	DADM1(As Single Ended)
0001	DADM0 (As Single Ended)	DADP1/DADM1
0010	AD2	NA[1]
0011	NA	AD3
00100	AD4[2]	Reserved
00101	AD5	NA
00110	VLL1	VREFH
00111	VCAP1	VREFL
01000	NA	AD6
01001	NA	AD7
01010	VLL2	NA
01011	VCAP2	Reserved
01100	AD8	Reserved
01101	AD9	Reserved
01110	Reserved	Reserved
01111	VREF OUT Internal	VREF OUT Internal
10000	NA	AD11
10001	PMC VREF 1.2V	AD10
10010	NA	NA
10011	NA	NA
10100	AD12	PRACMP0 DAC OUT
10101	AD13	PRACMP1 DAC OUT
10110	NA	AD15
10111	NA	AD14
11000	NA	PRACMP2 DAC OUT
11001	NA	NA
11010	Temperature Sensor	Temperature Sensor
11011	Bandgap	Bandgap
11100	NA	NA

ADCH	Input function of ADC0	Input function of ADC1
11101	NA	NA
11110	NA	NA
11111	NA	NA

[1] The unused channels are connected to V_{REFL}

[2] Users must not select AD4 – AD15 as a channel if $V_{LL3} > V_{DDA}$

2 Features

The following section describes the ADC features.

2.1 Hardware trigger

The ADC module has a selectable asynchronous hardware conversion trigger, ADHWT, that is enabled when the ADTRG bit is set and a hardware trigger select event (ADHWTSn) has occurred. When the ADHWT source is available and hardware trigger is enabled (ADTRG=1), a conversion is initiated on the rising edge of the ADHWT after a hardware trigger select event (ADHWTSn) has occurred.

The hardware triggers of ADC0 and ADC1 are provided from Programmable Delay Block (PDB) channel 1 and channel 2 respectively, when ADTRG is set in ADC0SC2 / ADC1SC2.

2.2 Hardware average function

Hardware averaging is available in both the ADCs. The number of samples to be taken for averaging can be 4, 8, 16, or 32 samples and is configurable.

2.3 Modes of operation

ADC can operate in both differential and single ended mode. It can operate in differential 16-bit, 13-bit, 11-bit, and 9-bit modes, or single-ended 16-bit, 12-bit, 10-bit, and 8-bit modes.

2.4 Self calibration

The ADC contains a self-calibration function that is required to achieve the specified accuracy. Calibration must be run before a conversion is initiated. The calibration function sets the offset calibration value and the plus-side and minus-side calibration values. The offset calibration value is automatically stored in the ADC Offset Correction Registers.

Prior to calibration, the user must configure the ADCs clock source and frequency, low power configuration, voltage reference selection, sample time and the high speed configuration according to the application’s clock source availability and needs.

3 Software driver description

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

There are two files associated with the ADC driver.

- **gw64_adc.h:** It contains all the high level API functions declarations and the various macros to be used in the functions. It also defines the structure of the various ADC registers.
- **gw64_adc.c:** It is the main file for the driver. It contains the various high level API definitions.

NOTE

The ADC driver code is available in a zipped file named **AN4169SW.zip**

3.1 gw64_adc.h

The macros provided are passed as arguments to the respective functions to get the required configuration. The next section in this document describes it in more detail.

Table 4. Macros used for ADC initialization

Macro	Description

`#define ADC_0`
`#define ADC_1`
	Used to select between the two ADCs. Both the ADCs can be used simultaneously for conversion.

`#define ADICLK_BUS`
`#define ADICLK_BUS_2`
`#define ADICLK_ALTCLK`
`#define ADICLK_ADACK`
	Used to select the clock used for both the ADC modules. The ADC clock can be selected between bus clock, bus clock/2, alternate clock, or asynchronous clock.

`#define ADIV_1`
`#define ADIV_2`
`#define ADIV_4`
`#define ADIV_8`
	Used to select the divide ratio of the ADC clock. The ADC clock can be divide by 1, 2, 4, or 8.

`#define ADC_AVG_SAMPLE_4`
`#define ADC_AVG_SAMPLE_8`
`#define ADC_AVG_SAMPLE_16`
`#define ADC_AVG_SAMPLE_32`
	Used to select the number of samples for hardware averaging.
	NOTE: These macros only make sense when hardware averaging is enabled.

`#define ADTRG_SW`
`#define ADTRG_HW`
	Used to select between software trigger or hardware trigger.

Table 5. Macros used to select the reference voltage for the ADC

Macro	Description
`#define ADC_REFSEL_EXT`	Default voltage reference pin pair (External pins V_{REFH} and V_{REFL}).
`#define ADC_REFSEL_ALT`	Alternate reference pair (V_{ALTH} and V_{ALTL})
`#define ADC_REFSEL_BG`	Internal bandgap reference and associated ground reference (V_{BGH} and V_{BGL}).
`#define ADC_REFSEL_RES`	Reserved - Selects default voltage reference (V_{REFH} and V_{REFL}) signals.

Table 6. Macros used in calibration function to check between the options

Macro	Description
`#define ADC_CAL_BEGIN`	Checks if the calibration is active.
`#define ADC_CAL_OFF`	Checks if the calibration is over.

Macro	Description
`#define ADC_CALF_NORMAL`	Checks if the calibration is completed normally.
`#define ADC_CALF_FAIL`	Checks if the calibration is failed.

Table 7. Macros used to select the various options for compare

Macro	Description
`#define ADC_COMPARE_LESS`	Compares if the converted value stored in data register is less than the specified value.
`#define ADC_COMPARE_GREATER`	Compares if the converted value stored in data register is greater than the specified value.
`#define ADC_COMPARE_RANGE_DISABLED`	Comparison within the specified range is disabled.
`#define ADC_COMPARE_RANGE_ENABLED`	Comparison within the specified range function is enabled.

Table 8. Macros to select between the various modes of conversion

Macro	Description
`#define ADC_MODE_8`	It selects 8-bit conversion mode when Single ended conversion occurs and 9-bit signed mode when differential mode is used.
`#define ADC_MODE_12`	It selects 12-bit conversion mode when Single ended conversion occurs and 13-bit signed mode when differential mode is used.
`#define ADC_MODE_10`	It selects 10-bit conversion mode when Single ended conversion occurs and 11-bit signed mode when differential mode is used.
`#define ADC_MODE_16`	It selects 16-bit conversion mode when Single ended conversion occurs and 16-bit signed mode when differential mode is used.
`#define ADCO_SINGLE` `#define ADCO_CONTINUOUS`	Used to select between continuous or single ADC conversion.
`#define ADC_CHANNEL_A` `#define ADC_CHANNEL_B`	Used to select between Channel A or Channel B for conversion.
`#define ADC_AIEN_OFF` `#define ADC_AIEN_ON`	Used for enabling or disabling the ADC conversion interrupt.
`#define ADC_SINGLE_MODE` `#define ADC_DIFF_MODE`	Used to select between differential mode or singled ended mode conversion.

Table 9. Macros used to select the channel for conversion

Macro	Description
#define ADC_CHANNEL_AD0 #define ADC_CHANNEL_AD1 #define ADC_CHANNEL_AD2 #define ADC_CHANNEL_AD3 #define ADC_CHANNEL_AD4 #define ADC_CHANNEL_AD5 #define ADC_CHANNEL_AD6 #define ADC_CHANNEL_AD7 #define ADC_CHANNEL_AD8 #define ADC_CHANNEL_AD9 #define ADC_CHANNEL_AD10 #define ADC_CHANNEL_AD11 #define ADC_CHANNEL_AD12 #define ADC_CHANNEL_AD13 #define ADC_CHANNEL_AD14 #define ADC_CHANNEL_AD15	These channels are distributed between the two ADCs. They are explained in ADC channels.

Table 10. Macros to read the analog values of the signal read from the specific ADC

Macro	Description
#define Read_ADC0_A_AnalogValue ()	It reads the data from channel A of ADC0 and returns the analog value in float.
#define Read_ADC0_B_AnalogValue()	It reads the data from channel B of ADC0 and returns the analog value in float.
#define Read_ADC1_A_AnalogValue()	It reads the data from channel A of ADC1 and returns the analog value in float.
#define Read_ADC1_B_AnalogValue()	It reads the data from channel B of ADC1 and returns the analog value in float.

Table 11. Macro to set the values for V_{REFH} and V_{REFL}

Macro	Description
#define VREFH 3.3 #define VREFL 0	Used to set the values for V_{REFH} and V_{REFL} for conversion of the digital data read from ADC into analog value. The conversion includes the calculation Analog data = (Digital data / (2^n -1)) * (V_{REFH} – V_{REFL}).

3.2 gw64_adc.c

It contains the definition of functions to configure and use the various features of ADC.

3.2.1 ADC_Init ()

Description:

This function initializes the specific ADC interface by configuring the internal registers. It is also used to set the ADC clock, the ADC reference clock, and modes of operation.

Prototype:

```
void ADC_Init(unsigned char ADC_Index, unsigned char ADC_Clk_Select, unsigned char ADC_Clk_Div,
 unsigned char ADC_Mode, unsigned char Trigger_Select, unsigned char
Continuous_Conversion_Select, unsigned char Volt_Ref_Select)
```

Input Parameters:

- *ADC_Index*—Selects the ADC to be initiated using the macros ADC_0, ADC_1
- *ADC_Clk_Select*—Selects the clock for ADC using the macros ADICLK_BUS, ADICLK_BUS_2, ADICLK_ALTCLK, ADICLK_ADACK
- *ADC_Clk_Div*—Selects the divide factor for ADC clock using the macros ADIV_1, ADIV_2, ADIV_4, ADIV_8
- *ADC_Mode*—Selects the bit conversion mode of ADC using the macros ADC_MODE_8, ADC_MODE_12, ADC_MODE_10, ADC_MODE_16
- *Trigger_Select*—Selects between software / hardware trigger mode using the macros ADTRG_SW, ADTRG_HW
- *Continuous_Conversion_Select*—Selects between the single / continuous conversion mode using the macros ADCO_SINGLE, ADCO_CONTINUOUS
- *Volt_Ref_Select*—Selects the reference voltage for ADC using the macros ADC_REFSEL_EXT, ADC_REFSEL_ALT ADC_REFSEL_BG, ADC_REFSEL_RES

Output Parameters:

None

Example:

```
ADC_Init(ADC_0,ADICLK_BUS,ADIV_1,ADC_MODE_16,ADTRG_HW, ADCO_SINGLE,ADC_REFSEL_EXT)
```

Initializes ADC0 with bus clock as ADC clock, 16-bit mode, hardware trigger enabled, single conversion and external reference voltage.

3.2.2 ADC_Channel_Config ()

Description:

This function is used to select between the Channel A or Channel B of the selected ADC. It also configures the channel properties such as differential mode / single ended mode and enabling or disabling the interrupt.

Prototype:

```
void ADC_Channel_Config(unsigned char ADC_Index, unsigned char Channel_Index, unsigned char
Interrupt_Enable, unsigned char Diff_Mode_Sel)
```

Input Parameters:

- *ADC_Index*—Selects the ADC to be initiated using the macros ADC_0, ADC_1
- *Channel_Index*—Selects between the Channel A / Channel B of the selected ADC using the macros ADC_CHANNEL_A, ADC_CHANNEL_B
- *Interrupt_Enable*—Enables or disables the conversion complete interrupt by using the macros ADC_AIEN_OFF, ADC_AIEN_ON
- *Diff_Mode_Sel*—Selects between differential / single ended mode using the macros ADC_SINGLE_MODE/ ADC_DIFF_MODE

Output Parameters:

None

Example:

```
ADC_Channel_Config(ADC_0,ADC_CHANNEL_A,ADC_AIEN_OFF, ADC_SINGLE_MODE)
```

Configures Channel A for ADC0 with conversion complete interrupt disabled and single ended conversion mode enabled.

3.2.3 ADC_Compare_Enable ()

Description:

This function is used to enable the compare function of the selected ADC and configures the various compare properties.

Prototype:

```
void ADC_Compare_Enable(unsigned char ADC_Index, unsigned char
Compare_GreaterThan_Enable,unsigned char Compare_Range_Enable, unsigned int
Compare_Value1,unsigned int Compare_Value2)
```

Input Parameters:
- *ADC_Index*—Selects the ADC to be initiated using the macros ADC_0, ADC_1
- *Compare_GreaterThan_Enable*—Compares the result is Greater than or less than the specific value by using the macros ADC_COMPARE_LESS, ADC_COMPARE_GREATER
- *Compare_Range_Enable*—Enables or disables the comparison of the result within the range specified using the macros ADC_COMPARE_RANGE_ENABLED, ADC_COMPARE_RANGE_DISABLED
- *Compare_Value1*—Sets the lower compare value of the range
- *Compare_Value2*—Sets the higher compare value of the range

Output Parameters:

None

Example:

```
ADC_Compare_Enable (ADC_0, ADC_COMPARE_LESS, ADC_COMPARE_RANGE_ENABLED,0x22, 0xFF)
```

Enables the Compare function for ADC0 within the range 0x22 and 0xFF. The conversion complete flag is set only when the converted value is within the specified range.

3.2.4 ADC_Compare_Disable ()

Description:

This function is used to disable the compare function for the selected ADC.

Prototype:

```
void ADC_Compare_Disable(unsigned char ADC_Index)
```

Input Parameters:
- *ADC_Index*—Selects the ADC to be initiated using the macros ADC_0, ADC_1

Output Parameters:

None

Example:

```
ADC_Compare_Disable(ADC_0)
```

Disables the compare function for the conversion taking place at ADC0.

3.2.5 ADC_Avg_Enable ()

Description:

This function is used to enable the hardware averaging for the selected ADC and select the number of samples for the hardware averaging.

Prototype:

```
void ADC_Avg_Enable(unsigned char ADC_Index,unsigned char Avg_Sample_Select)
```

Input Parameters:

- *ADC_Index*—Selects the ADC to be initiated using the macros ADC_0, ADC_1
- *Avg_Sample_Select*—Selects the number of samples for the hardware averaging using the macros ADC_AVG_SAMPLE_4, ADC_AVG_SAMPLE_8, ADC_AVG_SAMPLE_16, ADC_AVG_SAMPLE_32

Output Parameters:

None

Example:

```
ADC_Avg_Enable(ADC_0, ADC_AVG_SAMPLE_32)
```

Enables the hardware averaging for ADC0 with 32 samples taken for averaging.

3.2.6 ADC_Avg_Disable ()

Description:

This function is used to disable the hardware averaging function for the selected ADC.

Prototype:

```
void ADC_Avg_Disable(unsigned char ADC_Index)
```

Input Parameters:

- *ADC_Index*—Selects the ADC to be initiated using the macros ADC_0, ADC_1

Output Parameters:

None

Example:

```
ADC_Avg_Disable(ADC_0)
```

Disables the hardware averaging for the conversion taking place at ADC0.

3.2.7 ADC_Diff_Channel_Select ()

Description:

This function is used to select the differential channel for conversion. DADP0 / DADM0 is selected by selecting ADC0 in the function and DADP1 / DADM1 is selected by selecting ADC1 in the function.

Prototype:

```
unsigned char ADC_Diff_Channel_Select(unsigned char ADC_Index, unsigned char Channel_Index)
```

Input Parameters:

- *ADC_Index*—Selects the ADC to be initiated using the macros ADC_0, ADC_1
- *Channel_Index*—Selects between the Channel A / Channel B of the selected ADC using the macros ADC_CHANNEL_A, ADC_CHANNEL_B

Output Parameters:

None

Example:

```
ADC_Diff_Channel_Select(ADC0, ADC_CHANNEL_A)
```

Selects the differential channel available on ADC0, that is, DADP0 / DADM0 and configured channel A for conversion.

3.2.8 ADC_Single_Channel_Select ()

Description:

This function is used to select the single ended channel among the 16 channels available (including the differential channels which can be used as single ended as well).

Prototype:

```
unsigned char ADC_Single_Channel_Select(unsigned char Channel_Index,unsigned char Channel)
```

Input Parameters:

- *Channel_Index*—Selects between the Channel A / Channel B of the selected ADC using the macros ADC_CHANNEL_A, ADC_CHANNEL_B
- *Channel*—Selects the channel for ADC conversion using the macros ADC_CHANNEL_AD0, ADC_CHANNEL_AD1 ...... ADC_CHANNEL_AD15

Output Parameters:

None

Example:

```
ADC_Single_Channel_Select(ADC_CHANNEL_A, ADC_CHANNEL_AD2)
```

Selects the single ended channel AD2 available in ADC0 for conversion.

3.2.9 ADC_Cal ()

Description:

This function is used to calibrate the selected ADC and store the calibrated values in respective gain and offset registers.

NOTE

This function should be always called in the beginning after initializing the ADC. All the conversions should be done after the calibration.

Prototype:

```
unsigned char ADC_Cal(unsigned char ADC_Index)
```

Input Parameters:

ADC_Index —Selects the ADC to be initiated using the macros ADC_0, ADC_1

Output Parameters:

None

Example:

```
ADC_Cal(ADC_0)
```

It calculates the ADC gain and offset and stores it in respective ADC registers.

3.2.10 Read_ADC0_A ()

Description:

This function reads and stores the digital value of data from channel A of ADC_0.

Prototype:

```
void Read_ADC0_A(unsigned int* Data)
```

Input Parameters:

- *Data*—Data read is stored in the address pointed by this pointer passed

Output Parameters:

None

Example:

```
unsigned int value
Read_ADC0_A(&value)
```

Reads the data from channel A of ADC_0 and stores it in the variable ‘value’.

3.2.11 Read_ADC0_B ()

Description:

This function reads and stores the digital value of data from channel B of ADC_0.

Prototype:

```
void Read_ADC0_B(unsigned int* Data)
```

Input Parameters:

- *Data*—Data read is stored in the address pointed by this pointer passed

Output Parameters:

None

Example:

```
unsigned int value
Read_ADC0_B(&value)
```

Reads the data from channel B of ADC_0 and stores it in the variable ‘value’.

3.2.12 Read_ADC1_A ()

Description:

This function reads and stores the digital value of data from channel A of ADC_1.

Prototype:

```
void Read_ADC1_A(unsigned int* Data)
```

Input Parameters:

- *Data*—Data read is stored in the address pointed by this pointer passed

Output Parameters:

None

Example:

```
unsigned int value
Read_ADC1_A( &value)
```

Reads the data from channel A of ADC_1 and stores it in the variable ‘value’.

3.2.13 Read_ADC1_B ()

Description:

This function reads and stores the digital value of data from channel B of ADC_1.

Prototype:

```
void Read_ADC1_B(unsigned int* Data)
```

Input Parameters:

- *Data*—Data read is stored in the address pointed by this pointer passed

Output Parameters:

None

Example:

```
unsigned int value
Read_ADC1_B( &value)
```

Reads the data from channel B of ADC_1 and stores it in the variable ‘value’.

3.2.14 Read_ADC_AnalogValue ()

Description:

This function reads the digital value from the specified ADC channel. It converts the digital value into analog value depending upon the ADC reference voltage.

Prototype:

- *ADC_Index*—Selects the ADC to be initiated using the macros ADC_0, ADC_1
- *Channel_Index*—Selects between the Channel A / Channel B of the selected ADC using the macros ADC_CHANNEL_A, ADC_CHANNEL_B

Output Parameters:

Analog value of the data read (in float)

Example:

```
float value
value = Read_ADC_AnalogValue(ADC_0, ADC_CHANNEL_A)
```

Stores the analog value of the data at channel A of ADC_0.

3.3 Interrupt Subroutines

If the conversion complete interrupt is enabled then the interrupt subroutine for the respective ADC is executed. There are two different interrupts for the two ADCs. Thus there are two interrupt subroutines as follows:

3.3.1 ADC0_ServiceInterrupt ()

Description:

This subroutine is called when the conversion complete interrupt occurs for ADC_0. Status bits are checked to find on which channel conversion has taken place and it stores the data in the global variable ADC_Data_A / ADC_Data_B.

Prototype:

```
void interrupt VectorNumber_Vadc0 ADC0_ServiceInterrupt(void)
```

Input Parameters:

None

Output Parameters:

None

3.3.2 ADC1_ServiceInterrupt()

Description:

This subroutine is called when the conversion complete interrupt occurs for ADC_1. Status bits are checked to find on which channel conversion has taken place and it stores the data in the global variable ADC_Data_A / ADC_Data_B.

Prototype:

```
void interrupt VectorNumber_Vadc1 ADC1_ServiceInterrupt(void)
```

Input Parameters:

None

Output Parameters:

None

4 Assumptions

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

5 Use Case

Assuming that the clock settings are done and the bus clock is running on 20MHz. Include the file adc_flowtron.h in the main file and perform the following steps.

1. Declare an unsigned int variable data and initialize the respective ADC with the required configuration

   ```
   unsigned int data;
   ADC_Init(ADC_0,ADICLK_BUS, ADIV_1, ADC_MODE_12, ADTRG_SW, ADCO_SINGLE,
      ADC_REFSEL_EXT);
   ```

2. Calibrate the ADC_0

   ```
   ADC_Cal(ADC_0);
   ```

3. Select single channel AD2 and select the channel for conversion

   ```
   ADC_Single_Channel_Select(ADC_CHANNEL_A, ADC_CHANNEL_AD2);
   ```

4. Read the data from AD2 and store it in a variable

   ```
   Read_ADC0_A(&data);
   ```

6 Conclusion

This driver provides a software base for applications that needs the implementation of ADC.

7 References

MC9S08GW64/MC9S08GW32 Reference Manual(document: MC9S08GW64RM)

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: AN4169
Rev. 2, 2010