Document Number: AN3405
Rev. 0, 02/2007

Hardware-Triggered ADC

Using Internal Hardware to Start an Analog Conversion v1.0

by: Gabriel Sanchez
RTAC Americas

Contents

1 Introduction 1
2 Abstract 1
3 Objective 2
3.1 Hardware Description 2
3.2 Example Application Software Overview. 3
3.3 ADC Interrupt. 5
4 Conclusion. 6
5 References. 6

1 Introduction

Most applications today need checked for an analog signal being measured. In some applications this is done by having the analog-to-digital converter (ADC) constantly sampling a signal, but in many battery-powered applications where low-power use is critical, it is best to have analog-to-digital conversions only when necessary. This application note describes how to set up the ADC module so that it is triggered by a hardware source.

2 Abstract

The MC9S08LC60 contains a 12-bit ADC based on successive approximation. This ADC module can start a conversion from a software or hardware trigger. This hardware trigger is the output from the real-time interrupt (RTI) counter. The clock for the RTI module can be the ICGERCLK or a nominal 1 kHz clock source within the RTI.

3 Objective

In this application note, the MC9S08LC60 is used to take analog measurements every second off a potentiometer using the RTI as the hardware trigger to start the conversion. After the conversion is finished, the MCU displays the converted value on an LCD display.

3.1 Hardware Description

This application note uses the DEMOLC60 EVB hardware to demonstrate the ADC module's abilities. Figure 1 shows a photo of DEMOLC60 with the LCD screen and potentiometer used in the example application.

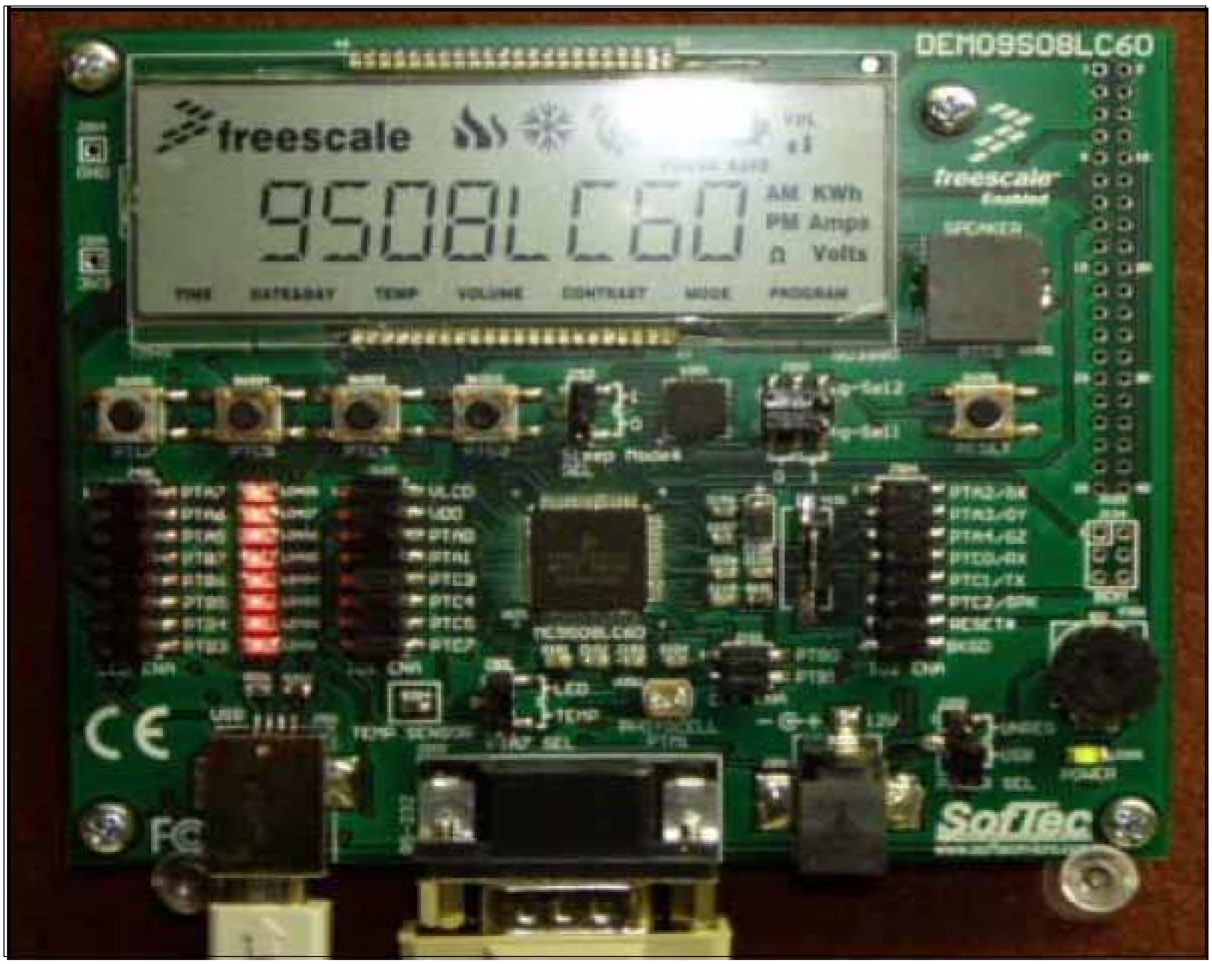

Figure 1. DEMOLC60 EVB

With this hardware, the example application uses the potentiometer set to port A bit 0.

3.2 Example Application Software Overview

This section provides an overview of the example application demo's configuration. The overview reviews settings for:

- ICG — Internal clock generator
- LCD — Liquid crystal display
- RTI — Real-time interrupt
- ADC — Analog-to-digital converter

The LCD drivers are a subset of the application. The goal of this section is not to describe every function in detail but only to describe the initialization of the modules used.

3.2.1 ICG Configuration

The ICG has four possible configurations, including the following three modes:

- FLL bypassed, external clock (FBE) mode
- FLL engaged, internal clock (FEI) mode
- FLL engaged, external clock (FEE) mode

FBE and FEE use the 32.768 kHz crystal on the DEMO9SLC60. For these modes, you must install the DEMO9S08LC60 CLK ENA jumper. In the example application, the MCU is set up in FEI mode.

```
#if !CLOCK
  // Selects FEE mode for fBUS = 8 MHz
  // Using crystal low-range oscillator. FLL prescaler factor (P) = 64
  ICGC1 = 0x38;
  // Sets MFD multiplication factor (N) to 4 and RFD divisor (R) to 1
  ICGC2 = 0x20;
  // Waita until FLL frequency is locked
  while (!(ICGS1 & 0x08))
    ;
#else
  // Selecta FEI mode
  // Sets trimming for fBUS = 8 MHz
  ICGTRM = NVICGTRM;
  // Using internal reference clock. FLL prescaler factor (P) = 64
  ICGC1 = 0x28;
  // Sets MFD multiplication factor (N) to 14 and RFD divisor (R) to 2
  ICGC2 = 0x51;
  // Waits until FLL frequency is locked
  while (!(ICGS1 & 0x08))
    ;
#endif
```

To select FEE mode, undefine CLOCK, which is found in the lcd.h file.

3.2.2 LCD Configuration

The LCD drivers included in the example applications allow you to configure the LCD, write to the display, enable and disable the LCD interrupts, as well as the interrupt handler. The LCD initialization routine configures the LCD clock and the other parameters of the LCD module. Because the LCD module clock source may be an external clock or the internal clock source, you may undefine CLOCK, which is defined in the lcd.h file.

```
void LCDInit(void)
{
   #if !CLOCK
  // Select External Crystal clock (32.768 kHz) as LCD Clock Source.
  LCDCLK=32.768 kHz
  LCDCLKS = 0x00;

  #else
  // Select fBUS (8MHz) as LCD Clock Source. Enable prescaler by 16 and CLKADJ=14.
  // LCDCLK about 32.768 kHz
  LCDCLKS = 0xCE;
  #endif

   // Disable LCD frame frequency interrupt and configure LCD Module to continue
running
  // during wait and stop3 mode
  LCDCR1 = 0x00;

  // Configures 1/4 duty cycle and 128 as LCD clock input divider (LCLK=3)
  // LCD Waveform Base Clock = 256 Hz
  LCDCR0 = 0x1B;

  //Enable charge pump (The internal 1/3-bias is forced)
  LCDSUPPLY |= 0x80;

  //Configures 2 Hz as LCD blink frequency (blink only individual LCD segments)
  LCDBCTL = 0x02;

  //Enable the LCD module frontplane waveform output (FP[39:0])
  FPENR0 = 0xFF;
```

```
  FPENR1 = 0xFF;
  FPENR2 = 0xFF;
  FPENR3 = 0xFF;
  FPENR4 = 0xFF;

  //Enables LCD driver module
  LCDCR0  |= 0x80;
}
```

3.2.3 RTI Configuration

The real-time interrupt is configured in the system real-time interrupt status and control register (SRTISC). The register is set up to enable the RTI using the internal 1 kHz source and to set the period select to 1.024 s.

```
  //Setup RTI
  SRTISC = 0x47;
```

3.2.4 ADC Configuration

The ADC module is set up to enable a 12-bit conversion to the AD0 input and enable hardware-triggered initialization, with the input clock being run at the bus clock. After the ADC is set up, the ADC interrupt is enabled.

```
//Setup ADC
  ADCCFG = 0x04;
  ADCSC2 = 0x40;
  ADCSC1 = 0x00;
  APCTL1 = 0x01;
  //Enable ADC Interrupts
  ADCSC1_AIEN = 1;
```

3.3 ADC Interrupt

The ADC module starts a conversion every time the RTI module generates an interrupt. After the conversion, the ADC module creates an interrupt used to update the two right-most characters in the LCD module with the obtained value from the ADC module.

```
void interrupt 18 ADCInterrrupt(void){

  DispHexVal(ADCRH,5);
  DispHexVal(ADCRL,7);
}
```

The `DispHexVal` function is used convert the first parameter passed to two characters in ASCII. These values are then placed onto the LCD in the two characters, starting at the value given by the second parameter.

4 Conclusion

Using the RTI module to trigger the ADC can be a very useful means of generating a timebase for which the MCU can check an analog channel. This can help you create applications that use less power and fewer CPU cycles.

5 References

AN3280, Interfacing an LCD to the MC9S08LC60 —
http://www.freescale.com/files/microcontrollers/doc/app_note/AN3280.pdf?fsrch=1

AN3404, How to do EEPROM Emulation Using Double Flash Array on MC9S08LC60 — To be released to www.freescale.com in Spring 2007.

THIS PAGE IS INTENTIONALLY BLANK

How to Reach Us:

Home Page:
www.freescale.com

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

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
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 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

Document Number: AN3405
Rev. 0
02/2007

freescale™
semiconductor