TI TechNotes

ADC Wake and Transmit on Threshold Using MSP430™ MCUs

TEXAS INSTRUMENTS

Introduction

Many applications, such as battery monitors and thermostats, require sampling analog signals periodically so action can be taken based on the behavior of those signals. Analog-to-digital convertors (ADCs) can be triggered with precise timers to provide a solution to this requirement. This consumes a lot of microcontroller (MCU) resources and also causes higher power consumption as the timer needs to be active along with the ADC. Alternatively, many ADCs allow signals to be monitored continuously, but doing so can also require high power consumption. Operating the ADC integrated in the MCU independently of the CPU allows the CPU and all clocks other than the one being used by the ADC to be disabled to save power. The ADC also includes logic to determine conditions that require the CPU and the rest of the MCU to wake up. This is achieved by the ADC generating an interrupt to wake up the CPU when a configurable threshold is crossed. In addition, interrupts can be used to also send out ADC conversion data serially without having to power on the CPU.

This implementation uses UART to set the ADC threshold value, which when crossed will result in waking up the CPU, setting a GPIO and transmitting ADC data over the UART. It has been optimized for lowest code size, fitting in a low-cost 1KB MSP430FR2100 MCU, and for lowest power while still able to continuously monitor analog signals using the internal ADC. To get started, download project files and a code example demonstrating this functionality.

Implementation

The solution uses the default auxiliary clock (ACLK) supply from the internal trimmed low-frequency oscillator (REFO), which operates at approximately 32 kHz, to operate the internal 10-bit SAR ADC, as well as the universal serial communication interface (eUSCI) UART peripheral and Timer_B. Samples are taken continuously from P1.3 (A3), with each conversion taking place immediately after the preceding one has finished, with a periodicity of 2.7307 kHz. After configuring all necessary peripherals, the device goes into low-power mode 3 (LPM3). The ADC continually monitors the analog input to the MCU in this mode, while the CPU and all clocks other than ACLK are tuned off to reduce power consumption. When the set voltage threshold is reached by the analog input to the ADC, P1.0 is set to indicate that the threshold has been reached, and a timer is started to send out the ADC conversion results over UART at a defined interval.

Figure 1 shows the block diagram for this implementation. The MSP-TS430PW20 target development board was used for connecting the peripherals to the MSP430FR2100 MCU. To set up the target board to run the demo, first ensure that jumpers JP14 and JP15 are populated (leave JP13 unpopulated), that jumper J16 is set to UART, and that jumpers JP11, JP17, and JP18 are all removed. These jumper settings will enable the back-channel UART interface on the MSP-FET programmer and debugger. For the MSPFR2100 MCU, make sure that the jumpers on JP11 are connected on 1-3 and 2-4.

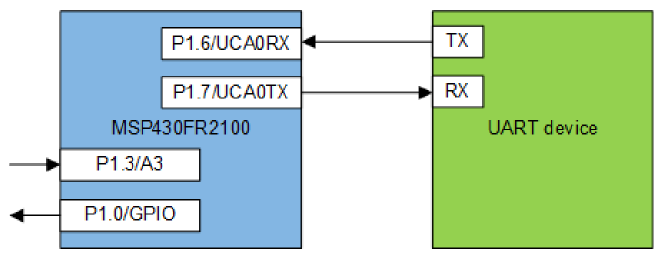

Figure 1. ADC Wake and Transmit on Threshold Block Diagram

A terminal program was used to provide a simple method for controlling the ADC threshold value and receiving the ADC conversion data. The eUSCI_A0 peripheral was used in UART mode to enable commands to be received on P1.6/UCA0RXD and transmitted on P1.7/UCA0TXD. The MSP-FET was used for evaluation. A baud rate of 4800 must be selected with one stop bit and no parity. Two hexadecimal inputs must be sent to the MCU individually. The first specifies the high byte of the ADC threshold value, and the second specifies the low byte. The maximum value that can be represented by the ADC is 0x03FF, and the minimum is 0x0000. All ADC values are represented by unsigned 16-bit integers. The high threshold value is stored in FRAM, so it is retained on reset of the device. When the ADC threshold value is reached, ADC conversion values are sent over UART with a period that is set by default to 1 second. To change this, modify the value of the TB0CCR0 register, keeping in mind that the timer is using a clock of 32768 Hz. So, for example, a

TB0CCR0 value of 32768 results in a timer period of 1 second. Because the ADC produces a 10-bit unsigned integer, the data is sent over UART in two bytes, with the high byte first. This is illustrated in Figure 2. Two UART transmissions are shown. The red boxes show the high bytes, and the green boxes show the low bytes. These UART transmissions occurred after a threshold of 0x01FF was reached.

Figure 2. Two Sample UART Transmissions

Performance

The operation of the demo can be run as described in the implementation section regarding the use of UART to control the threshold value and receive ADC conversion values. P1.3 (A3) is used for the analog input to the ADC, and a voltage can be applied to this pin to trigger the high threshold value of the ADC.

Figure 3 shows an oscilloscope shot that illustrates this. Channel 1 shows a triangular wave input to A3 on the MSP430FR2100 MCU, and channel 3 shows the output of P1.0 (GPIO). That GPIO gets set high when the triangular wave input hits a voltage equivalent to 0x01FF (the set threshold) for the ADC conversion value. This point is represented by the dashed red line.

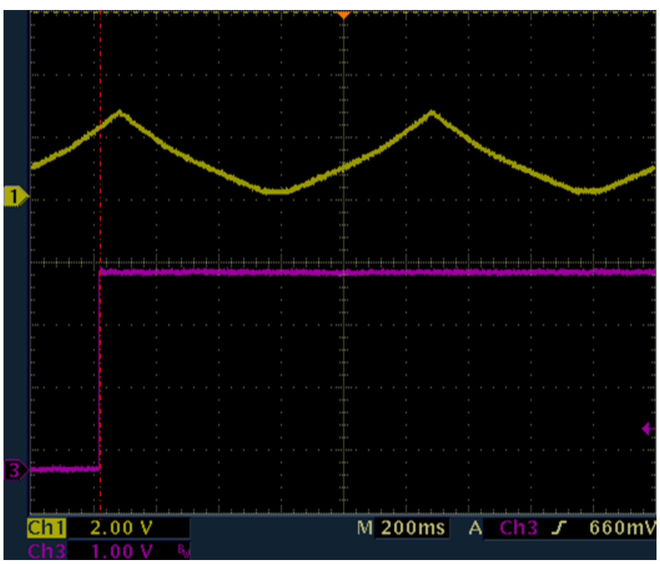

Figure 3. Oscilloscope View

An important metric in measuring the performance of this implementation is power consumption. Power numbers were taken using the Keysight Technologies N6705B DC Power Analyzer. A 3 V supply was used, and the average current consumption of operating the ADC continuously in LPM3 was compared to that of operating in active mode (with the CPU and all clocks powered on). Table 1 lists the results.

Table 1. Current Consumption in Active Mode and LPM3

Power Mode	Average Current Consumption at 3 V (µA)
Active	97
LPM3	25

It can be seen that operating the ADC while in LPM3 and waiting for an interrupt from the ADC reduces the power consumption significantly. Another low power alternative would be to put the device into low-power mode 3.5 (LPM3.5), which also shuts down ACLK, and use the real time clock (RTC) to wake up the CPU and ADC at specified intervals to take a sample, and then go back to LPM3.5. This would require relatively large wakeup intervals in order to get the power as low as it is in the LPM3 implementation, which would not be ideal for applications that need high sampling rates.

Device Recommendations

The device used in this example is part of the MSP430 Value Line Sensing portfolio of low-cost MCUs, designed for sensing and measurement applications. This example can be used with the devices shown in Table 2 with minimal code changes. For more information on the entire Value Line Sensing MCU portfolio, visit www.ti.com/MSP430ValueLine.

Table 2. Device Recommendations

Part Number	Key Features
MSP430FR2000	0.5KB FRAM, 0.5KB RAM, eComp
MSP430FR2100	1KB FRAM, 0.5KB RAM, 10-bit ADC, eComp
MSP430FR2110	2KB FRAM, 1KB RAM, 10-bit ADC, eComp
MSP430FR2111	3.75KB FRAM, 1KB RAM, 10-bit ADC, eComp