ContinuousADC

November 2009

Description

This program demonstrates continuous conversion of the 11 ADC channels available on the F28027 Piccolo controlSTICK.

Hardware Setup

🟦 **1** **ADC-A7**	🟦 **2** **ADC-A2** **COMP1 (+VE)**	🟦 **3** **ADC-A0** **Vref-HI**	**4** **3V3**
🟦 **5** **ADC-A4** **COMP2 (+VE)**	🟦 **6** **ADC-B1**	**7** **EPWM-4B** **GPIO-07**	**8** **TZ1** **GPIO-12**
9 **SCL** **GPIO-33**	🟦 **10** **ADC-B6**	**11** **EPWM-4A** **GPIO-06**	🟦 **12** **ADC-A1**
13 **SDA** **GPIO-32**	🟦 **14** **ADC-B7**	**15** **EPWM-3B** **GPIO-05**	**16** **5V0**
17 **EPWM-1A** **GPIO-00**	🟦 **18** **ADC-B4** **COMP2 (-VE)**	**19** **EPWM-3A** **GPIO-04**	**20** **SPISOMI** **GPIO-17**
21 **EPWM-1B** **GPIO-01**	🟦 **22** **ADC-B3**	**23** **EPWM-2B** **GPIO-03**	**24** **SPISIMO** **GPIO-16**
25 **SPISTE** **GPIO-19**	🟦 **26** **ADC-B2** **COMP1 (-VE)**	**27** **EPWM-2A** **GPIO-02**	**28** **GND**
29 **SPICLK** **GPIO-18**	**30** **GPIO-34** **(LED)**	**31** **PWM1A-DAC** **(Filtered)**	**32** **GND**

Table 1: J1 Connections

⬜	No connection
🟦	External DC supply (<3.3V)

Software Setup

Add the following variable to the watch window:

AdcResults - (format = hexadecimal) - This array stores the converted values from the device's ADC channels. These values range from 0x0000 to 0x0FFF, which linearly scale to the equivalent of 0 to 3 volts respectively.

Overview

This project is configured to continuously convert each of the 11 ADC channels available on the controlSTICK. This is achieved by configuring an ADC's end of conversion (EOC) event to create an ADC interrupt. This ADC interrupt is then configured to start the conversion of a set of ADC channels. Figure 1 illustrates the ADC conversion sequence used in this project.

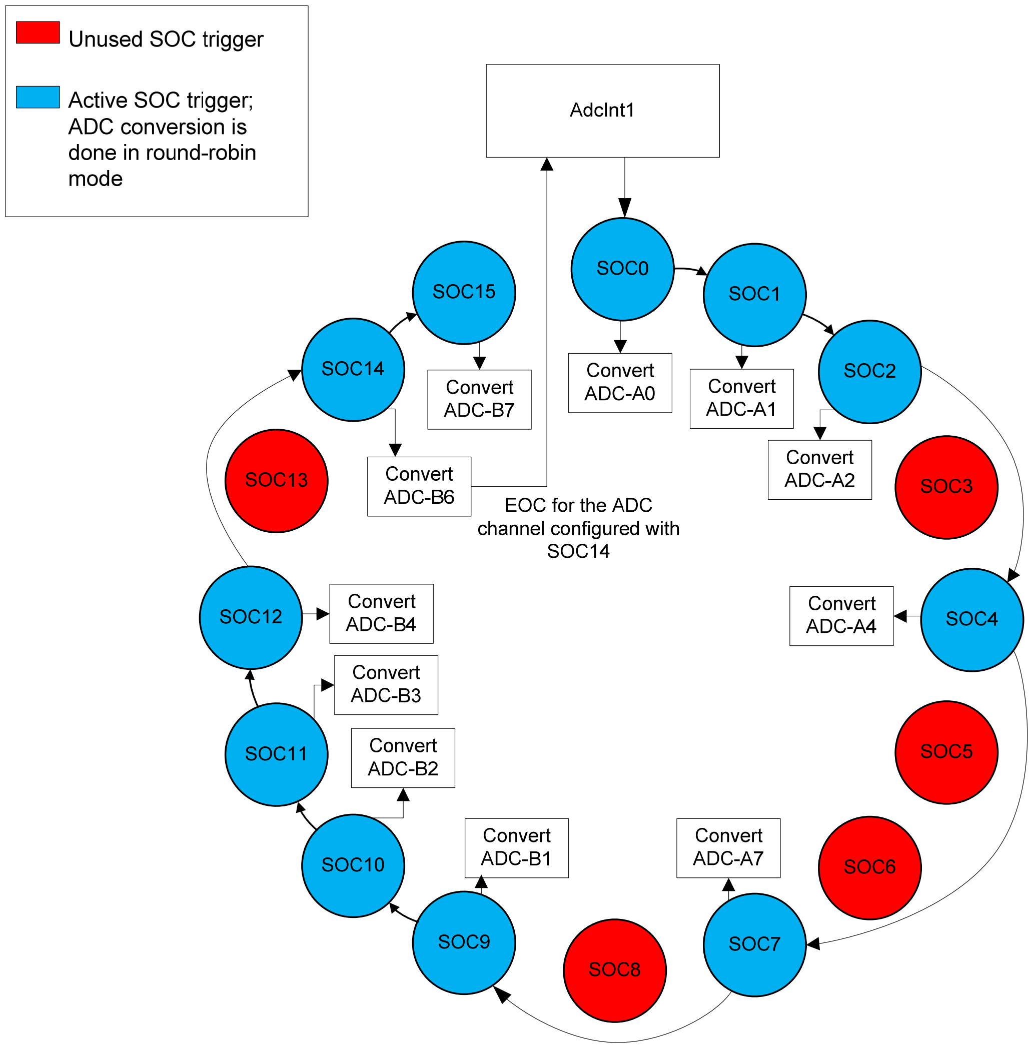

Figure 1: Continuous ADC conversion sequence

Note that ADCInterrupt1 chooses which SOC triggers to begin, and in this project all SOC triggers are chosen with ADC Interrupt 1 except SOC3, SOC5, SOC6, SOC8 and SOC13. The arbitration logic will choose to convert the channels in the sequence as shown in Figure 1.

Also note that each SOC trigger can be configured to convert any ADC channel. For example, SOC11 is used to convert ADC-B3 in this example, but it could have been used to convert ADC-A1 a second time to verify that the voltage converted the first time by SOC1 is correct. Since ADCs A3, A5, A6, B0, and B7 are not available on the Piccolo controlSTICK, the SOC triggers SOC3, SOC5, SOC6, SOC8, and SOC13 are unused in this project. This is done to make understanding of the ADC easier, but these SOC triggers are valid and can be configured as desired.

In this project, ADCInterrupt1 is used exclusively. For greater configurability, ADCInterrupt2 could be used to begin some subset of ADC SOC triggers. This would effectively allow there to be two separate ADC conversion sequences with a configurable set of ADC channel conversions.

In this project, an ADC channel 14 (ADC-B6) end of conversion event (EOC) causes the ADCInterrupt1 interrupt to fire. The interrupt pulse can be created before ADC-B7 is converted because of the dual sample and hold feature of the ADC. A write to the INTPULSEPOS field of the ADCCTL1 register allows the ADCInterrupt1 interrupt to be generated one cycle prior to output latch ADCInterrupt1. See the code snippet below:

```
// create int pulse 1 cycle prior to output latch
AdcRegs.ADCCTL1.bit.INTPULSEPOS = 1;

AdcRegs.INTSEL1N2.bit.INT1SEL = 14;       // ADCCH14 EOC causes ADCInterrupt 1
AdcRegs.INTSEL1N2.bit.INT1CONT = 1;       // set ADCInterrupt 1 to auto clr
AdcRegs.INTSEL1N2.bit.INT1E = 1;          // enable ADC interrupt 1
```

Next, ADCInterrupt1 is configured to generate a series of start of conversion (SOC) events using the ADCINTSOCSELx register. Each of these ADC SOC event is then configured to begin the conversion of a specific ADC channel via the CHSEL field of the ADCSOCxCTL register. The CHSEL field expects a value 0-15 where ADC channels ADCA0-ADCA7 correspond to 0-7 and ADCB1-ADCB7 correspond to 9-15.

If the SOCy field of the ADCINTSOCSELx is set to 0, no interrupt will cause an ADC start of conversion (SOC) interrupt. Setting this field to 1 or 2 allows the ADCInterrupt1 or ADCInterrupt2 respectively to cause an ADC SOCx trigger.

```
AdcRegs.ADCINTSOCSEL1.bit.SOC0 = 1;       // ADCInterrupt 1 causes SOC0
...
...
AdcRegs.ADCINTSOCSEL2.bit.SOC15 = 1;


AdcRegs.ADCSOC0CTL.bit.CHSEL = 0;   //convert ADCA0(CH0) when SOC0 is received
AdcRegs.ADCSOC1CTL.bit.CHSEL = 1;   //convert ADCA1(CH1) when SOC1 is received
AdcRegs.ADCSOC2CTL.bit.CHSEL = 2;
AdcRegs.ADCSOC4CTL.bit.CHSEL = 4;
AdcRegs.ADCSOC7CTL.bit.CHSEL = 7;
AdcRegs.ADCSOC9CTL.bit.CHSEL = 9;   //convert ADCB1(CH9) when SOC9 is received
AdcRegs.ADCSOC10CTL.bit.CHSEL = 10;
AdcRegs.ADCSOC11CTL.bit.CHSEL = 11;
AdcRegs.ADCSOC12CTL.bit.CHSEL = 12;
AdcRegs.ADCSOC14CTL.bit.CHSEL = 14;
AdcRegs.ADCSOC15CTL.bit.CHSEL = 15;
```