SparkFun Blocks for Intel® Edison - ADC V20

Introduction

SparkFun's ADC Block for the Intel Edison allows you to add four channels of I2C controlled ADC input to your Edison stack. These four channels can be used as single-ended inputs, or in pairs as differential inputs. A ground reference is provided for each channel.

The maximum resolution of the converters is 12 bits, or 11 bits bipolar in differential mode. Step sizes range from 125uV per count to 3mV per count.

ADC Block

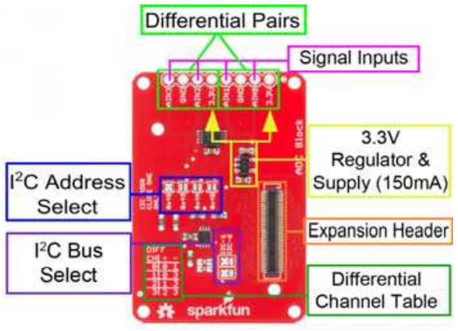
Suggested Reading

If you are unfamiliar with Blocks, take a look at the General Guide to Sparkfun Blocks for Intel Edison.

Other tutorials that may help you on your Edison adventure include:

- Programming the Edison - This tutorial assumes you are **not** using the Arduino IDE, so you'll want to familiarize yourself with C++ development on the Edison.
- Powering Your Project
- Connector Basics
- Analog-to-digital conversion

Board Overview



ADC Block Functional Diagram

- Signal Inputs - Four single inputs are available. The reference voltage for each is produced internal to the ADC under software control; do not exceed 3.3V input to these pins!
- Differential Channel Setting Table - Use two inputs to create a differential pair. Useful for eliminating noise in some sensors or measuring very small signals. This table shows the channel options for the “getDiffResult(channel)” function.
- I2C Address Select - Apply solder to **only one** of the four jumpers to select the address. **Do not short two of these at once.** Bad stuff will happen.
- I2C Bus Select - Change **both** of these jumpers to select between routing the I^{2}C signals to bus 6 or bus 1. Bus 1 is the default (and preferred) channel, as it has no other system devices on it. Bus 6 is shared with some internal devices, but if you wish to use this block with the Arduino IDE, you’ll want to change these jumpers so the solder blobs connect the bottom pad with the center pad.
- 3.3V 150mA Supply - This supply provides an on-board reference for the ADC, and can power small sensors (for example, potentiometers or temperature sensors).

Using the ADC Block

To use the ADC Block simply attach an Intel Edison to the back of the board or add it to your current stack. Blocks can be stacked without hardware but it leaves the expansion connectors unprotected from mechanical stress.

ADC Block Installed

We have a nice Hardware Pack available that gives enough hardware to secure three blocks and an Edison.

Intel Edison Hardware Pack

NOTE: The ADC Block does not have console access or a voltage regulator. It is recommended to use a console communication block in conjunction with this block like ones found in the General Guide to Sparkfun Blocks for Intel Edison.

C++ Code Examples

We're assuming that you're using the Eclipse IDE as detailed in our Beyond Arduino tutorial. If you aren't, you'll need to go to that tutorial to get up to speed.

Getting Started

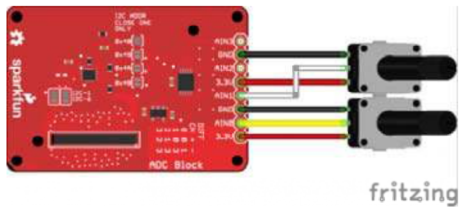
Follow the instructions in the programming tutorial to create a new project named "SparkFun_ADC_Edison_Block_Example". Once you've created the project, open the project files on disk (hint: you can find the path to the project by choosing "Properites" from the project menu) and copy the three source files found in the Edison ADC Block CPP library GitHub repository into the "src" directory.

DOWNLOAD A ZIP FILE OF THE REPOSITORY

Hardware Connection

For this example, we've just got two 5k potentiometers connected between 3.3V and GND, with the wipers connected to channels 0 and 1.

V20 of the board adds a 3.3V reference supply (capable of sourcing up to 150mA, so it can power small sensors directly!).



Of course, you can connect any other analog voltage signal in place of the potentiometers; we're using them because they're convenient to demonstrate the concepts.

Code

Everything you need to know is in the comments.

```
/**************************************************************
***
Example file for SparkFun ADC Edison Block Support

1 Jun 2015- Mike Hord, SparkFun Electronics
Code developed in Intel's Eclipse IOT-DK

This code requires the Intel mraa library to function; for mor
e
information see https://github.com/intel-iot-devkit/mraa

This code is beerware; if you use it, please buy me (or any ot
her
SparkFun employee) a cold beverage next time you run into one
of
us at the local.
**************************************************************
**/

#include "mraa.hpp"

#include <iostream>
#include <unistd.h>
#include "SparkFunADS1015.h"

using namespace std;

// Declare a variable for our i2c object. You can create an
//  arbitrary number of these, and pass them to however many
//  slave devices you wish.
mraa::I2c* adc_i2c;

int main()
{
    // The ADC is (by default) connected to I2C channel 1. Her
e, we create
    //  a device to pass to the ads1015 object constructor.
    adc_i2c = new mraa::I2c(1);

    // Now, pass that I2C object and the address of the ADC bl
ock in your
    //  system to the ads1015 object constructor. Note that th
ere are up to
    //  four different addresses available here, settable by j
umper on the
    //  board. You'll need to create an ads1015 object for eac
h one.
    ads1015 adc(adc_i2c, 0x48);

    // There are 6 settable ranges:
    //  _0_256V - Range is -0.256V to 0.255875V, and step siz
e is 125uV.
    //  _0_512V - Range is -0.512V to 0.51175V, and step size
is 250uV.
    //  _1_024V - Range is -1.024V to 1.0235V, and step size i
s 500uV.
    //  _2_048V - Range is -2.048V to 2.047V, and step size i
s 1mV.
    //  _4_096V - Range is -4.096V to 4.094V, and step size i
s 2mV.
    //  _6_144V - Range is -6.144V to 6.141V, and step size i
s 3mV.
    // The default setting is _2_048V.
    // NB!!! Just because FS reading is > 3.3V doesn't mean yo
u can take an
    //  input above 3.3V! Keep your input voltages below 3.3V
to avoid damage!
    adc.setRange(_0_512V);
    // getResult() returns a normalized floating point value r
epresenting the
    //  current voltage of the passed channel. User is respons
ible for
    //  logic to determine whether the value is at min or max.
    cout<<"Ch 0: "<<adc.getResult(0)<<endl;
    cout<<"Ch 1: "<<adc.getResult(1)<<endl;
    // getDiffResult() returns a normalized fp value represent
ing the
    //  difference between two channels. Options are
    //  0 - Ch0 - Ch1
    //  1 - Ch0 - Ch3
    //  2 - Ch1 - Ch3
    //  3 - Ch2 - Ch3
    cout<<"Ch 0 - ch 1: "<<adc.getDiffResult(0)<<endl;

    // If you want to do the math yourself, you can determine
the current gain
    //  setting by using the getScaler() command.
    cout<<"Current scaler: "<<adc.getScaler()<<"V per bit"<<en
dl;
    // The current voltage is the scaler/1000 multiplied by th
e raw value. You
    //  can get the raw ADC readings using the getRawResult()
and
    //  getRawDiffResult() functions.
    cout<<"Ch 0 raw: "<<adc.getRawResult(0)<<endl;
    cout<<"Ch 1 raw: "<<adc.getRawResult(1)<<endl;
    cout<<"Ch 0 - ch 1 raw: "<<adc.getRawDiffResult(0)<<endl;

    // If you want to get *really* crazy, you can always go lo
ok up the
    //  datasheet and read and write the configuration registe
r directly.
    cout<<"Config register: "<<hex<<adc.getConfigRegister()<<e
ndl;
    // There's a "setConfigRegister()" function, too, which ex
pects an
    //  unsigned 16-bit integer. Just FYI.

    return MRAA_SUCCESS;
}
```

Resources and Going Further

Now that you have had a brief overview of the ADC Block, take a look at some of these other tutorials. These tutorials cover programming, Block stacking, and interfacing with the Intel Edison ecosystems.

Edison General Topics:

- General Guide to Sparkfun Blocks for Intel Edison
- Edison Getting Started Guide
- Loading Debian (Ubilinix) on the Edison

Block Specific Topics:

- ADC Block Git Repo
- ADC Block CPP Library Git Repo

Check out these other Edison related tutorials from SparkFun:

SparkFun Blocks for Intel® Edison - OLED Block
A quick overview of the features of the OLED Block for the Edison.

SparkFun Blocks for Intel® Edison - microSD Block
A quick overview of the features of the microSD Block.

SparkFun Blocks for Intel® Edison - Battery Block
A quick overview of the features of the Battery Block.

Single Board Computer Benchmarks
How to set up different benchmarking programs on single board computers or computing modules and run them. The results for various generations are shown on the subsequent pages.