SecretVoltmeter

Accessing the secret voltmeter on the Arduino 168 or 328

Featured

Updated Aug 2, 2009 by cathed...@gmail.com

Introduction

It turns out the Arduino 168 and 328 can measure their own voltage rail.

Code

Copy, paste into Arduino and see what it returns. This works on an Arduino 168 or 328.

```
long readVcc() {
  long result;
  // Read 1.1V reference against AVcc
  ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
  delay(2); // Wait for Vref to settle
  ADCSRA |= _BV(ADSC); // Convert
  while (bit_is_set(ADCSRA,ADSC));
  result = ADCL;
  result |= ADCH<<8;
  result = 1126400L / result; // Back-calculate AVcc in mV
  return result;
}

void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println( readVcc(), DEC );
  delay(1000);
}
```

The voltage is returned in millivolts. So 5000 is 5V, 3300 is 3.3V.

Note the following:

- This works on Arduinos with a **328 or 168 only**. It looks like the same trick might be possible on the Arduino Mega - experiments are ongoing, and will be reported here.

How it works

The Arduino 328 and 168 have a built in precision voltage reference of 1.1V. This is used sometimes for precision measurement, although for Arduino it usually makes more sense to measure against Vcc, the positive power rail.

The chip has an internal switch that selects which pin the analogue to digital converter reads. That switch has a few leftover connections, so the chip designers wired them up to useful signals. One of those signals is that 1.1V reference.

So if you measure how large the known 1.1V reference is in comparison to Vcc, you can back-calculate what Vcc is with a little algebra. That is how this works.

See also

The SecretThermometer in the Arduino 328

Comment by bla...@bravo5.org, Nov 4, 2009

Isn't this comparing the 1.1V analog reference voltage to what's connected to AVcc? So you're only measuring Vcc if it's tied to AVcc, correct?

Comment by project member cathed...@gmail.com, Nov 4, 2009

@blalor Yes. AVcc is connected to Vcc on Arduinos. The processor spec sheet says AVcc must be within 300mV of Vcc at all times, so I can't see any circuit scenario where it wouldn't be near identical to Vcc.

Comment by bla...@bravo5.org, Nov 5, 2009

Man, I'm really on a roll, eh? :-(I got ARef and AVcc confused. I don't understand why you'd need a separate voltage source for the ADC, but I do see that note and get the difference, now. Sorry for the noise…

Comment by project member cathed...@gmail.com, Nov 5, 2009

@blalor No problem. AVcc is separate from Vcc so you can add power supply filters to keep the digital noise away, or even use separate voltage regulators. But the interface between analog and digital sections breaks down unless you keep the voltages close.

Comment by dida...@gmail.com, Nov 22, 2009

Not having much progress with an Arduino Mega. It's spitting out ~1101 (1.101 volts), while displaying negative results every now and then.

Comment by project member cathed...@gmail.com, Nov 22, 2009

@didanix "This works on Arduinos with a 328 or 168 only."

Comment by davidk...@gmail.com, May 17, 2010

Thanks. Just found this through a link on the arduino forum: http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1263440087

Very useful. I can confirm that it works on a ATMega328P-PU (arduino duemilanove).

:)

Comment by bla...@bravo5.org, May 28, 2010

Looking through the datasheet, it should be possible to measure ARef by setting

```
ADMUX =  _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
```

(ie. leaving REFS1 and REFS0 at 0). Is that correct? Instead of selecting AVcc I'd be selecting ARef. Both against Vbg. Or is there a danger there with switching the voltage reference with something attached to ARef?

Comment by project member cathed...@gmail.com, May 28, 2010

@blalor Sounds fine to me. Aref could then potentially be used as an extra analogue input by that method. (It can also be used as a digital output - but not a very good one). Try it out!

Comment by nasuka...@gmail.com, Aug 13, 2010

Hmmm.... great work! Can't you use this to obviate the need for any external reference voltage whatsoever?

If you know the precise Vcc then you can just use it as the ARef, even if you don't have a regulated power supply (aka just running it off AAs).

This is great for my little thermometer project, thanks!

robokaren

Comment by nat...@chantrell.net, Feb 19, 2012

Can confirm that this also works on the ATtiny84 (using the arduino-tiny core) if you change the registers to:

ADMUX = *BV(MUX5)* | BV(MUX0);

Nathan.

Comment by ddmi...@gmail.com, Apr 5, 2012

Is it possible to adapt the code for an ATmega32U4 based board? Did someone make it? What are the necessary changes? Thank you in advance for any advice on this isuue. Dimitar

---

Comment by ddmi...@gmail.com, Apr 8, 2012

Using trial and error approach, I found (I think so...) that it works on ATmega32U4 too and answered my own question. I used the following line:

ADMUX = *BV(REFS0)* | BV(MUX2);

And it seems that if you use:

ADMUX = *BV(REFS0)* | BV(MUX1);

you can measure the 1.1V line inside ATmega32U4. Is that correct? I experimented with my Olimexino-32U4, which is an Arduino Leonardo compatible board and posted the adapted code on my blog page: http://open-dc-ups-monitor.blogspot.com/ Best wishes, Dimitar

---

Comment by tmal...@gmail.com, Jun 10, 2012

This appears as though it would work on the ATmega1284P also. According to the ADC section in the datasheet, it actually has a selectable 1.1 or 2.56 reference voltage.

I haven't checked the datasheets for my other Atmel chips yet, but hopefully it will work for them as well.

Comment by strangeo...@googlemail.com, Aug 9, 2012

Change this: ADMUX = *BV(REFS0)* | BV(MUX3) | *BV(MUX2)* | BV(MUX1); To this: ADMUX = *BV(REFS0)* | BV(MUX4) | *BV(MUX3)* | BV(MUX2) | *BV(MUX1); ADCSRB &=* ~BV(MUX5)

And it should work with an Arduino Mega (either the 1280 or 2560, both use the same registers)

Comment by kllsa...@gmail.com, Oct 21, 2012

is there a working version for arduino MEGA already?

Comment by infinix...@gmail.com, Oct 24, 2012

```
result = 1126400L / result;
```

Can someone tell me where 1126400 come from?

TIA

Comment by Phatt...@gmail.com, Jan 7, 2013

I think it is the 1.1Vref value. I had to adjust it to "1100000L" in my code,to get an output reading of 5000mv. (My Vcc rail is regulated/adjusted to 5.00V exactly.)

Comment by tefer...@gmail.com, Jan 30, 2013

```
result = 1126400L / result; // Back-calculate AVcc in mV
```

was derived from the equation in section 24.7 of the ATmega data sheet. That says…

```
ADC = (Vin * 1024) / Vref
```

For the SecretVoltmeter, Vref is connected internally to Vcc and the input to the ADC comes from an internal reference source that supplies 1.1V. Thus, rearranging this equation and expressing voltage in millivolts gives…

```
Vcc = (1100 * 1024) / ADC

or

Vcc = 1126400L / ADC
```

Comment by the1cy...@gmail.com, Feb 20, 2013

So far this seems to be the code for the mega: long readVcc() {

long result; // Read 1.1V reference against AVcc ADMUX = *BV(REFS0)* | BV(MUX4) | *BV(MUX3)* | BV(MUX2) | *BV(MUX1);// ADCSRB &= ~BV(MUX5)*

delay(3); // Wait for Vref to settle ADCSRA |= *BV(ADSC); // Convert*

while (bit_is_set(ADCSRA,ADSC)); result = ADCL; result |= ADCH << 8; result = 1100000L / result; // Back-calculate AVcc in mV return result;

}

Is that correct?

Comment by xequ...@gmail.com, Mar 3, 2013

the posted code does not work: long readVcc() {

long result; // Read 1.1V reference against AVcc ADMUX = *BV(REFS0)* | BV(MUX4) | *BV(MUX3)* | BV(MUX2) | *BV(MUX1); ADCSRB =* BV(MUX5) ;

delay(3); // Wait for Vref to settle ADCSRA |= *BV(ADSC); // Convert*

while (bit_is_set(ADCSRA,ADSC)); result = ADCL; result |= ADCH << 8; result = 1126400L / result; // Back-calculate AVcc in mV return result;

}

this gives me 3,

Comment by sp33d...@gmail.com, Jul 29, 2013

Could this magical program work on a BlinkM MinM, I'd like to have the MinM powered by 4.2V and also read that 4.2V cell and fade from Green to Red based on the battery dying. Application is a micro helicopter running on a single 4.2volt cell.. Thanks for any input!! :)

Comment by depabloh...@gmail.com, Aug 20, 2013

This code works for me (copied from previous posts) for Arduino Mega2560 R2 board:

long readVcc() {

long result; // Read 1.1V reference against AVcc ADMUX = *BV(REFS0)* | BV(MUX4) | *BV(MUX3)* | BV(MUX2) | *BV(MUX1); ADCSRB &=* BV(MUX5); delay(3); // Wait for Vref to settle ADCSRA |= *BV(ADSC); // Convert while (bit_is_set(ADCSRA,ADSC)); result = ADCL; result |= ADCH << 8; result = 1100000L / result; // Back-calculate AVcc in mV return result;*

}

It returns a value of 4932... almost all the time.

Comment by Liquib...@gmail.com, Oct 15, 2013

The number 1126400 is really arbitrary and will only work ideally in theory. Measure the voltage at the point of entry and calculate backwards and you'll get a much better result displayed. For instance, I measure 4.85V from my USB consistently so the calculated value of Vref is actually 1156.9 or actually 1156.88 rounded up. So (1156.9 x 1024) = 1184665.6 or 1184665L rounded down. I'm getting better readings against my meter now.

And then I found this at http://www.rcgroups.com/forums/showthread.php?t=1874973

```
//const long scaleConst = 1071.4 * 1000 ; // internalRef * 1023 * 1000;
//const long scaleConst = 1125.300 * 1000 ; // internalRef * 1023 * 1000;
const long scaleConst = 1156.300 * 1000 ; // internalRef * 1023 * 1000;
int readVccMv() {
  // Read 1.1V reference against AVcc
  // set the reference to Vcc and the measurement to the internal 1.1V reference
#if defined(__AVR_ATmega32U4__) || defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
  ADMUX = _BV(REFS0) | _BV(MUX4) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
#elif defined (__AVR_ATtiny24__) || defined(__AVR_ATtiny44__) || defined(__AVR_ATtiny84__)
  ADMUX = _BV(MUX5) | _BV(MUX0);
#elif defined (__AVR_ATtiny25__) || defined(__AVR_ATtiny45__) || defined(__AVR_ATtiny85__)
  ADMUX = _BV(MUX3) | _BV(MUX2);
#else
  ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
#endif

  delay(2); // Wait for Vref to settle
  ADCSRA |= _BV(ADSC); // Start conversion
  while (bit_is_set(ADCSRA,ADSC)); // measuring

  uint8_t low  = ADCL; // must read ADCL first - it then locks ADCH
  uint8_t high = ADCH; // unlocks both

  long result = (high<<8) | low;

  //result = 1125300L / result; // Calculate Vcc (in mV); 1125300 = 1.1*1023*1000
  result = scaleConst / result; // Calculate Vcc (in mV); 1125300 = 1.1*1023*1000
  return (int)result; // Vcc in millivolts
}
```

Which I then turned into this for my purposes which reiterates my original post that the math wasn't quite working out right on my end as evidenced by the scaleConst equation below. I guess I got close enough and what I'm reading now is spot on with my meter. Based on my readings I had to use a slightly modified equation of 1184.665 1000 to get the readings right.

```
#include <GU7000.h>
#include <GU7000_Interface.h>
#include <GU7000_Parallel.h>
#include <GU7000_Serial_Async.h>
#include <GU7000_Serial_SPI.h>
#include <GU7000_Serial_Sync.h>
#include <Noritake_VFD_GU7000.h>

GU7000_Serial_Async interface(115200, 2, 3, 4); // BAUD,SIN,BUSY,RESET
const long scaleConst = 1184.665 * 1000;  // internalRef * 1023 * 1000;
long readVcc() {
  // Read 1.1V reference against AVcc
  // set the reference to Vcc and the measurement to the internal 1.1V reference
  ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
    delay(2);           // Wait for Vref to settle
   ADCSRA |= _BV(ADSC);     // Start conversion
  while (bit_is_set(ADCSRA,ADSC)); // measuring
   uint8_t low  = ADCL;    // must read ADCL first - it then locks ADCH
   uint8_t high = ADCH;     // unlocks both
  long result = (high<<8) | low;
  result = scaleConst / result;  // Calculate Vcc (in mV); 1125300 = 1.1*1023*1000
   return (int)result;     // Vcc in millivolts
}

Noritake_VFD_GU7000 vfd;
void setup() {
   vfd.begin(128, 32);    // 128x32 module
   vfd.interface(interface);  // select which interface to use
   vfd.isModelClass(7003);  // select display model
   vfd.GU7000_reset();    // reset module
   vfd.GU7000_init();    // initialize module
   _delay_ms(300);     // wait for device to power up

}

void printVoltage(){
  int mv = readVcc();
  vfd.GU7000_defineWindow(1, 0, 0, 256, 32);
  vfd.GU7000_selectWindow(10);
  vfd.GU7000_setFontSize(3, 2, true); // wait for device to power up
  vfd.GU7000_setFontStyle(false, true);
   vfd.GU7000_clearScreen();     // little bit of flickering going on
   vfd.print( mv / 1000, DEC );   // print the integer value of the voltage
    vfd.print('.');          // print the decimal place
   vfd.print(mv % 1000, DEC);   // print the rest of the voltage in millivolts
    vfd.GU7000_lineFeed();      // start new line
   vfd.GU7000_carriageReturn();   // return to the left
  vfd.print("Volts");
}

void loop() {
  printVoltage();
  _delay_ms(300);
}
```

If you happened to take advantage of the Noritake VFD givaways lately and want to see this in action on a VFD, the above code should help out. I Hope this helps a few people in their searches for answers like I was when I first found the secret voltmeter.

Comment by Liquib...@gmail.com, Oct 15, 2013

I hate wiki syntax with a passion!

Comment by hthr...@gmail.com, Jan 10, 2014

this is so cool !!! thanks so much it worked first go and great on my 328

Comment by mikecahi...@gmail.com, Feb 12, 2014

OK, I have tried USB power and 9V power the numbers are always the same for voltage 3.4, and I missing something?

Comment by andrew.j...@gmail.com, May 10, 2014

Works well on a Nano and Pro Mini (Atmel 328) - however trying to measure the internal temperature sensor, and the supply voltage in the same loop breaks something.. If I figure out why, I'll let you know. Thanks for this, it is perfect for a battery powered project I am working on.

Comment by Liquib...@gmail.com, Jun 2, 2014

The Noritake displays, as far as I'm aware, require digital pins. These won't work with the analog pins A0-A5.

I also ran into an issue lately after my OS hard drive failed and had to be replaced. The default avr-gcc that ships with Arduino won't compile this code correctly and you'll notice that this code, or any of the demo code that you get with the Noritake libraries, will output garbage on the screen instead of what you're expecting. Call me dense, but it took me two days to work out why. I run Archlinux so I can't speak for windows users or other Linux OS's but this is the fix for us Arch zealots.

Install avr-gcc with pacman (not an AUR version) and do the following.

cd /usr/share/arduino/hardware/tools/avr/bin # mv ./avr-gcc ./avr-gcc-backup # ln -s /usr/bin/avr-gcc ./

I found this fix over at the Archlinux forums so I can't take the credit for the fix. I have no idea why the avr-gcc in Arduino is broken this way. If you experience this issue, consider a bug report upstream and maybe they'll eventually fix this.

As for the USB/9V power issue showing 3.4V from Mike. We'll need to see your code and a pic of your wiring might not be a bad idea. I'll certainly do my best to help anyone get this code running because it was fun to figure out the first time and it's sort of been fun figuring it out the second.

As far as getting two readings going from the temp sensor and the supply voltage at the same time, I'll suggest the same thing as above. We'll need to see the code and possibly pics of the wiring to see if we can replicate it. I for one would like to implement multiple readings from multiple sources and would like to see any code along those lines.

I plan on doing a fan speed sensor readout soon with one of these and I wouldn't mind ideas and the more code we get here for this, the more could probably be worked out for ourselves and others.

One thing I'd like to eliminate is the flicker I get from the clear screen line but I have no way to refresh these other than that command. I'll work on it and post what I find out later.

Comment by Liquib...@gmail.com, Jun 2, 2014

If you use this code in place of the above, you can eliminate the flicker. Basically, you just have to add another line feed and carriage return after the vfd.print("Volts") and then delete the clear screen line.

```
void printVoltage(){

    int mv = readVcc();
    vfd.GU7000_setFontSize(3, 2, false);
    vfd.GU7000_setFontStyle(false, false);
    vfd.print(mv / 1000, DEC); // print the integer value of the voltage
    vfd.print("."); // print the decimal place
    vfd.print(mv % 1000, DEC); // print the rest of the voltage in millivolts
    vfd.GU7000_lineFeed(); // start new line
    vfd.GU7000_carriageReturn(); // return to the left
    vfd.print("Volts");
    vfd.GU7000_lineFeed(); // start new line
    vfd.GU7000_carriageReturn(); // return to the left
}
```

Comment by eckel.tim, Jun 11, 2014

Because I typically use (4) AA batteries with an inline diode to reduce the voltage a little and protect against reversed voltage, the function I use is a bit different. Also, many of the other calculations above are based on theory, not practice. I find the following returns more accurate real-world results:

```
void setup() {
  #if defined(__AVR_ATmega32U4__) || defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
    ADMUX = _BV(REFS0) | _BV(MUX4) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
  #elif defined (__AVR_ATtiny24__) || defined(__AVR_ATtiny44__) || defined(__AVR_ATtiny84__)
    ADMUX = _BV(MUX5) | _BV(MUX0);
  #elif defined (__AVR_ATtiny25__) || defined(__AVR_ATtiny45__) || defined(__AVR_ATtiny85__)
    ADMUX = _BV(MUX3) | _BV(MUX2);
  #else
    ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
  #endif
}

void loop() {}

int readVcc() {
  ADCSRA |= _BV(ADSC);
  while (ADCSRA & _BV(ADSC));
  int volts = (1420288L / ADC) + 155;
  return volts;
}
```

I use INT instead of LONG because there's no way its ever going to return more than 32 volts. Also, note that this will read the battery voltage down to about 1.54 volts.

Comment by eckel.tim, Jun 11, 2014

To get the internal voltage for a particular ATmega chip I use the following code:

```
#include <avr/sleep.h>

void setup() {
  analogReference (INTERNAL); // Sets the internal 1.1 volt reference.
  analogRead (A0);            // Forces voltage reference to be turned on.

  // Put the ATmega in a low-power deep sleep.
  ADCSRA &= ~(1 << ADEN);
  set_sleep_mode(SLEEP_MODE_PWR_DOWN);
  noInterrupts();
  sleep_enable();
  sleep_cpu();
}

void loop() {}
```

I run this sketch and let the voltage stabilize for a few seconds then read the voltage on GND and AREF. My 1420288L number is based on a 1.065v reading across GND and AREF (which is the value I get on most chips). If the voltage reading is different, adjust accordingly.

An example project that uses this can be found here: http://www.dogblocker.com/

Comment by futzw...@gmail.com, Sep 27, 2014

Giday EH!

I think I am Missing Something, I used the code from the very top of the page. I get a voltage of 3.352 to 3.363 or 3.362 if I am using a battery. I am using a "Redesigned Pro Mini AtMega328? 3.3v" (which is a clone from eBay seller alice1101983) the USB to Serial board supplies 3.316v, I also use a LiPo? bag cell 3.7V but actually 4.1 full charge to 2.4 when the protection cuts it off.

Unless I misunderstand, the code at the top should show the voltage of the (battery(4.09 at the moment)) or the (usb to serial(3.316)), but not 3.352 to 3.363 for USB to Serial or 3.362 for battery. Or does it show the voltage the chip runs at only, which is 3.3V unless it drops below 3.3V, I guess I don't understand the code.

I had planed to use a MAX1701 fuel gauge module to get the voltage, state of charge and voltage alerts but I thought it would be nice to just use some code instead. I did try some of the other code posted as well and received different voltages but with the same type of problem.

Sorry if the question is a dumb one, I am starting to learn how the Arduino works.

Thanks! Later, FW.

Comment by gajananv...@gmail.com, Oct 19, 2014

Can you post a fritzing of the setup used. I suppose we at least need a voltage divider here.

Comment by off...@mandelbrot-solutions.at, Dec 22, 2014

How about to measure against raw input ?

Comment by markasw...@gmail.com, Jan 28, 2015

Hi Guys,

I'm also getting 3.5 Volts (ish) returned. I'm using a Arduino Nano V3: http://goo.gl/U6FgFD

My code: http://pastebin.com/n7X3uXY2

Comment by markasw...@gmail.com, Jan 28, 2015

You can see the voltage graph here also: https://thingspeak.com/channels/22352

Comment by iceaddf...@gmail.com, Jan 30, 2015

Hey. Thanks for the interesting and useful idea. I'm completely noob in this. Just wondering what library/header file does one need to include to use ADMUX, *BV and those functions. Is it avr/io.h? How can I include avr/io.h because when I tried using Arduino, it didn't let me.*

► Sign in to add a comment