

Microcontroller-Based Nitrox Analyzer

Many scuba divers prefer to breathe Nitrox instead of regular air when they dive. To do so safely, they must use a special analyzer to measure the percentage of oxygen in the tank. David recently designed his own AT90S4433-based Nitrox analyzer.

Like many *Circuit Cellar* readers, I often find myself combining my enthusiasm for embedded systems with my other interests to come up with engaging design projects. The project I'll describe in this article is no exception. The Nitrox analyzer in Photo 1 is the result of my interest in a specialized form of diving in which you breathe a gas called Nitrox instead of regular air.

Typical recreational dives involve breathing air, which is made up of approximately 21% oxygen (O₂) and 79% nitrogen (N₂). When you breathe enriched air Nitrox, however, you're breathing a gas containing more oxygen and less nitrogen than regular air. It's important for reasons of safety that when breathing Nitrox you know exactly what percentage of oxygen you're breathing.

I originally considered purchasing an analyzer, but after some thought, I figured that designing one would be a far more rewarding experience. Although I'm always excited to develop and use my own embedded devices, I don't have a death wish. It's for this reason that I only use my analyzer as

a redundant check of a dive technician's analyzer. This is also the reason that I must recommend that you use my analyzer only as an example of an interesting embedded device.

For the non-divers out there, a little background information is in order. Scuba diving has several inherent risks that can lead to injury or death. Proper training is necessary to mitigate these risks and to allow you to execute safe dives. Although a comprehensive overview of diving procedures and physiology is beyond the scope of this article, there are a few general principles that are relevant to understanding the functions that my Nitrox analyzer performs.

DIVE SCIENCE

In general, the deeper you dive, the less time you can stay down. One tool you can use to increase your dive time is Nitrox, which decreases the amount of nitrogen in the gas you breathe. Because the percentage of nitrogen decreases, the percentage of oxygen increases to maintain the same overall volume and pressure. This can introduce complications. Oxygen can become toxic to the central nervous system (CNS) if it's breathed at too high a pressure and concentration. CNS oxygen toxicity can result in convulsions and loss of consciousness, which can lead to death when you're submerged under water. Because the pressure of the gas breathed increases with depth, the maximum dive depth must be limited to prevent the pressure of the oxygen from nearing toxic levels. The higher the percentage of oxygen in the Nitrox blend, the shallower the maximum allowable dive depth. For these

reasons, it's important for you to know the percentage of oxygen in the mixture that you'll be breathing so you can plan a safe dive.

When you purchase a tank of Nitrox, you must follow the specific procedures set by the major dive certification agencies to minimize the chances of accidents. Typically, when a technician creates a certain mixture of Nitrox and fills a tank with it, he checks the percentage of oxygen with a commercial analyzer. When you're ready to take possession of the tank, it's your responsibility to verify the percentage of oxygen in the Nitrox. Typically, you'll use the same analyzer that the technician used to make this measurement.

After you know the percentage of oxy-



Photo 1—I built the Nitrox analyzer because of my interest in both embedded systems and scuba diving. Here you see the analyzer along with the oxygen sensor in the PVC sensor mount and a tank of Nitrox.



Photo 2—Take a look at the Nitrox Analyzer's main electronic components, subsystems, and PCB. Note the homemade protoboard section in the upper right corner.

$$\text{Percent O}_2 = 100 \left[\frac{\text{maximum output}}{\text{amplitude gain}} \times \frac{\text{percent O}_2 \text{ air}}{\text{worst-case sensor output for air}} - \text{worst-case offset error} \right]$$

$$41.6\% = 100 \left[\left(\frac{4.8 \text{ V}}{184} \right) \times \left(\frac{21\%}{13 \text{ mV}} \right) - 0.5\% \right]$$

Figure 2—This equation illustrates the calculation of the worst-case maximum percentage O₂ that the analyzer can measure.

worst in-spec sensors (see Figure 2). In this case, full-scale is 4.8 V, which is used as the theoretical maximum because the op-amp's output is limited to 70 mV less than the positive supply rail of 5 V.

The user interface consists of one push button for input and a 2 × 8 LCD module for output. The push button turns the unit on and off and allows you to select modes and options. In addition, the unit saves battery power by automatically powering down after various specified timeouts have elapsed. Furthermore, a spare ADC channel measures battery voltage to provide a low-battery warning.

The Nitrox analyzer is housed in a PacTec HP-9VB project case that includes a built-in compartment for the unit's 9-V battery. I'd like to thank Gordon Fry and David Manley for their assistance in modifying the housing to accept a bezel for the LCD.

In order to measure the percentage of oxygen in the Nitrox in a scuba tank, I built a sensor mount from a small, T-shaped PVC tube. The sensor is in the top opening. A second opening is held against the tank valve. The third opening is covered with a cap with a small hole in it. The tank valve is opened slightly—just enough to allow the Nitrox to begin flowing through the tube. It's important to keep the pressure of the Nitrox flowing from the tank to a minimum, because pressure will affect the accuracy of the measurement.

OPERATION

The first time the Nitrox analyzer powers up, it enters Sensor Calibration mode, which you can also achieve by holding down the button for several seconds at power-up. Calibration mode allows the AT90S4433 to compensate for deviations from the ideal of the oxygen sensor's output characteristic.

To perform the calibration, select your approximate altitude above sea level and

then expose the sensor to the air. The unit will take an initial reading, and then prompt you to expose the sensor to a known sample of Nitrox. After the measurement, you can specify the actual percentage of oxygen in the reference sample. The unit then computes a slope correction calibration factor to be applied to all future measurements and then stores it in internal EEPROM. The correction factor essentially converts the nonideal sensor characteristic to a reference ideal characteristic on which the AT90S4433 bases its measurements. The sensor slope calibration should be performed any time a different R-17D O₂ sensor is used with the unit.

When the analyzer powers up normally, it first prompts you to select the approximate altitude in feet above sea level. This allows it to approximate the ambient atmospheric pressure, which is required to convert the oxygen sensor's partial pressure measurement to the percentage of oxygen:

$$\text{percent O}_2 = \frac{\text{partial pressure O}_2}{\text{ambient pressure}}$$

Next, you're prompted to expose the sensor to air, and then the unit takes a measurement. Using this information, an additive offset compensation factor is computed to eliminate sensor and amplifier offset error. At this point, the device displays the percentage oxygen measured by the sensor and updates it every second.

To measure the percentage of oxygen from a Nitrox tank, place the oxygen sensor in the sensor mount, hold the mount to the tank valve, and open the valve just enough to allow a bit of Nitrox to flow through the assembly. The LCD should begin to indicate an increase in the percentage of oxygen.

After the reading stabilizes on a value for about 10 s, the unit displays the calculated percentage of oxygen in the tank. At this point, if you press and release the

push button, the analyzer will calculate and display the maximum dive depth at which the partial pressure of oxygen being breathed reaches 1.4 atmospheres, which is otherwise known as the maximum operating depth (MOD). After a few seconds, the LCD will begin displaying the percentage of oxygen currently detected by the sensor. If you don't press a button for 5 min., the unit automatically shuts down to conserve battery power. To shut down manually, press and hold the button for several seconds.

FIRMWARE

I wrote the AT90S4433's firmware in C and compiled it with an ImageCraft C compiler. The structure is the typical foreground-background paradigm. The majority of the code is executed in the foreground, with interrupts performing specific functions in the background, such as timeout management. Because power consumption is an issue with battery-powered devices, the firmware places the microcontroller in a low-power, idle Sleep mode during delays.

The calculations are performed with fixed-point arithmetic, which is a method of representing numbers with a fractional component as integers (using programmer-specified precision). This method is beneficial because it's treated as integer math by the compiler and the microcontroller, which means floating-point libraries don't need to be linked in. This saves a substantial amount of valuable code space.

Fixed-point math has two main shortcomings. First, it creates the burden of manually keeping track of the decimal location during calculations. Second, it requires you to know the exact range of the results of all the calculations in order to maximize the precision and to prevent numerical overflow. Early in the project, I experimented with floating-point math and quickly realized that there wouldn't be enough code space to accomplish what I wanted without switching to fixed-point math. That's when the project became a personal challenge to wring out as much functionality as possible from the AT90S4433's 4 KB of code space.

To understand what's involved in fixed-point computations, compare

Listing 1—This code snippet performs the same calculation in fixed-point and floating-point arithmetic. It partially illustrates the additional complexity involved in using fixed-point math.

```
// Fixed point
unsigned short usMaxDepth, usPercentO2x10000;
usMaxDepth = ((unsigned short)(((unsigned long)(46.2*10000)<<12)
/ (unsigned long)usPercentO2x10000)>>12) - 33);

// Floating point
float fPercentO2;
usMaxDepth = (unsigned short)(46.2 / fPercentO2 - 33.0);
```

Listing 1, which is a fixed-point version of the statement that determines MOD, to its floating-point equivalent. Obviously, the floating-point expression is easy to read. The MOD is 46.2 divided by the percentage of oxygen minus 33. The result is then typecast to an unsigned short, which results in the depth being truncated to an integral number of feet.

The fixed-point version is more difficult to understand. During the compiling process, the C preprocessor converts $((\text{ulong})46.2 * 10000) \ll 12$ to a 4-byte constant value of 1.892352×10^9 . This represents 462,000 with an implied decimal point between bit 12 and bit 11. The usPercentO2x10000 variable is the percentage of oxygen scaled by 10,000. Because fixed-point calculations are integer-based, fractional values must be manipulated so they can be represented as integers with enough bits to ensure that calculations are performed with precision. For example, 32.6% oxygen scaled by 10,000 results in 3,260, which can be stored in usPercentO2x10000. Next, usPercentO2x10000 is typecast to be 4 bytes long and divided into 1.892352×10^9 . The result is shifted 12 bits to the right to move the implied decimal point to just before bit 0 and then typecast to an unsigned short. Finally, 33 is subtracted from the value to obtain the depth truncated to an integral number of feet.

Another technique I employed to reduce the size of the compiled executable was based on analysis of the assembly output produced by the C compiler. I found that combining multiple statements of a complex calculation into a single C statement significantly reduced the amount of code generated by the compiler. This is

largely due to the elimination of many unnecessary register loads and stores that were occurring as values were swapped in and out of SRAM. Specifically, this technique resulted in an 18% reduction in the code produced for one particularly large calculation.

Another method I used to conserve code flash memory was to relocate the LCD text strings to EEPROM. This freed up an additional 244 bytes for the executable. Although this may not sound like much, it represents around 6% of the microcontroller's entire code flash memory space.

TEST RESULTS

Atmel's AVR Studio simulator is a valuable aid for testing firmware. I used it for the verification of the fixed-point code to step through the assembly. I also used the AVR Studio to verify the calculations for a wide range of input values. Simulation can take you only so far though. To truly test the device, I needed access to various samples of Nitrox. Thanks to Dive Connections in Charlottesville, Virginia and the Olympus Dive Center and Discovery Diving Company in Morehead, North Carolina, I was able to test my device's results against those produced by the shops' commercial analyzers.

I tested 17 different tanks of Nitrox. It's evident that the worst-case deviation between the Nitrox analyzer and the commercial analyzers is 0.4%, with the average deviation being on the order of 0.1% (see Figure 3 on page 26). The Professional Association of Dive Instructors (PADI) recommends that a Nitrox dive be planned using a percentage of oxygen within 1% of the

actual value in the tank.^[1] So, the accuracy is acceptable.

Although the results look encouraging, it's important to consider the following factors. Commercial analyzers are also susceptible to errors in measurement and calibration, which means that the test results illustrated in Figure 3 only provide an indication of accuracy relative to the commercial device being used as the reference. In other words, it isn't a good indicator of absolute accuracy. To measure absolute accuracy, precision Nitrox reference samples are required. Another factor to consider is that the available samples were all between 29% and 32% oxygen. More exhaustive testing would require samples across the 21% to 40% operable range of the analyzer.

FINAL THOUGHTS

I started working on this design approximately two years ago. It was one of those on again, off again projects marked by periods of fevered activity followed by months of shelf time. I hate leaving things unfinished, so I always found myself returning to it. I'm pleased with the results now that the Nitrox analyzer is complete.

The challenge of packing all of the functionality into the 4-KB code space was definitely instructive. The Nitrox analyzer project serves as yet another example of the versatility of 8-bit microcontrollers and the usefulness of the C language for programming them. ☒

David Smith is a senior electrical design engineer with National Optronics in Charlottesville, VA. He has been designing and programming embedded systems for the past eight years. David is a licensed professional engineer and holds B.S. and M.S. degrees in electrical engineering from Virginia Tech. You may contact him at bitwiz03@yahoo.com.

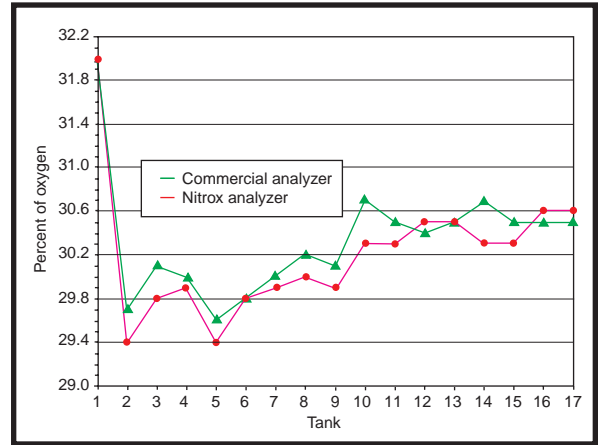


Figure 3—I put 17 different Nitrox tanks to the test. Here you see the percentages of oxygen in each tank tested with the analyzer versus the percentages found with a commercial analyzer.

PROJECT FILES

To download the code, go to [ftp.circuitcellar.com/pub/Circuit_Cellar/2005/174](ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2005/174).

REFERENCE

[1] PADI, *Enriched Air Diver Manual*, Rancho Santa Margarita, CA, www.padi.com.

RESOURCES

Atmel Corp., "8-Bit AVR Microcontroller with 4K Bytes of In-System Programmable Flash: AT90S4433," rev. 1042H-AVR, 2003.

National Association of Underwater Instructors (NAUI), www.nauai.com.

Professional Association of Diving Instructors, www.padi.com.

Teledyne, "Oxygen Sensors," 2001, www.teledyne-ai.com.

SOURCES

AT90S4433 and AVR Studio

Atmel
www.atmel.com

ICCAVR C Compiler

ImageCraft
www.imagecraft.com

HP-9VB Handheld enclosure

PacTec Enclosures
www.pactecenclosures.com

R17-D Oxygen sensor

Teledyne Analytical Instruments
www.teledyne-ai.com