



Quantalink LLC

Tim Black
6754 Howe Rd
Middletown Oh 45052
T 513 268 0334
tblack@quantalink.com
www.quantalink.com

Intoxilyzer 5000EN Source Code Report

(c) 2010 Tim Black, written for Meaney & Patrin, P.A, All Rights Reserved.

BACKGROUND INFO & INITIAL REVIEW

Before creating this report, I travelled to Owensboro, Kentucky and personally reviewed the entire source code in its native electronic format as provided by CMI, Inc. I spent 10 hours over two days with the source code and spent another 20 hours reviewing my recorded and written notes taken during that review. I also found a detailed manual online for an Intoxilyzer 5000 machine that includes schematics and other useful details about the inner workings of the base model without the Minnesota modifications. I also reviewed the “Minnesota Source Code Modification” document, which logs the dates for problems with the source code from 3/20/1997 to 8/3/2007 and a brief description of repairs. My Curriculum Vitae is attached to this report, which details my extensive experience in the development, analysis, and reverse engineering of software, hardware, and systems.

The I5000 is what is known as an “embedded system,” which means the software and hardware work together for a specific purpose and to perform specific functions. The software is the “brains” of the device: it directs the hardware in the machine to perform a multitude of functions in a defined sequence depending on variable inputs, and it receives output from various sensors and other hardware in the system as it performs its physical tasks of internal & external diagnostics, sample collection, measurement, calibration, and purging. It is impossible to conduct a meaningful review of the software or its source code without also reviewing and analyzing the hardware of the instrument. My review of the software included a detailed review of the schematics for the hardware of the device to review the interaction between software and hardware.

The I5000 device uses a simple infrared (IR) spectrometer to detect the presence of organic vapors in a breath sample. The earliest version of the device used only a single IR

frequency line, and so it could not determine the difference between ethyl alcohol and many other organic chemicals. The hardware of the device has evolved, mostly by the upgrade from a single IR filter to a three-filter wheel, and then to a five-filter wheel. The filter wheel is mounted on a motor and the motor rotates the filters into the path of light between an IR source (a light bulb) and an IR sensor. The readings from each filter are recorded as it passes the detector, and the differences in the values are used to determine the ethyl alcohol level in the sample.

There are two microprocessors in the unit. The main processor is a Z80 and is used for running the display, keyboard, printer, and some of the hardware controls. The slave processor is an 8051 and is responsible for the actual measurements and calculation of results.

The I5000 design is over 20 years old and much of the microprocessor control hardware inside it has not been updated. The source code has been edited many times in attempts to correct errors in the operation of the system. Some parts of the code date back to its origin on a TRS80, an early computer offered by Radio Shack from 1977 to 1981.

The last 20 years of computer evolution would be 500 years of progress in any other profession. Making a working program has always been an act of creative writing. Making readable and maintainable programs has been developed over the last 20 years into a science of structure and discipline. The source code of the I5000 was written before the existence of modern computer science, so it is not surprising that this code is lacking both structure and internal documentation.

Modern software development processes use a “bug tracking” database, and a Revision Control System to manage the complex task of developing and maintaining programs. These tools are used to enforce a direct cause and effect link between problems and corrections. When reading this source code it is clear that no formal processes were used in the software

development. There is no evidence that any modern development tools have ever been adopted by the I5000 development team.

In addition, it seems that the revision process may have been deliberately obfuscated in order to conceal the number of problems the system has had over the years. *It is my opinion that the restrictions over source access that have been put upon this reviewer have been primarily designed to hide the poor quality of workmanship used in this Source Code, not to protect the code from being copied or otherwise compromised.*

There are software analytical tools used to create metrics, or measurements of areas in source code that might have errors. The source code for the I5000 is so old and unorganized that there would be a myriad of errors found in it by current software analytical tools. For example, the analytical tool “Lint” assumes that there is some structure, but the I5000 code will appear to it as random hacking. With code like this, Lint will create thousands of warnings about the structural quality of this code. Since those errors and warnings will not immediately tell us whether the machine actually performs up to its specifications, and since I had the understanding that other defense experts would be performing this type of analysis, I opted not to perform this type of analysis. Instead, I presented a number of readily evident problems with the code and the design of the machine, then followed instructions on which problems to tackle from that list.

The most obvious weakness in this device is its self-testing functions, which are performed on every start up and at the beginning of each test run. This is where the bulk of my analysis was focused. The other main area of weakness is the slope detection source code, so I also performed a detailed study of the machine to test the limits of its ability to detect and report contaminants in a breath sample.

SELF-TEST EVALUATION

OVERVIEW :

From the review of the Source Code it is obvious that the self-test function is very limited. It only checks the basic operation of the main and slave microprocessors, and a few basic functions. **The bulk of the device, the power supplies, the control solenoids, auxiliary heaters, even the air pump have no actual testing. The entire analog section (the core of the measurement process) is untested, as detailed later in MISSING ITEMS.**

When some of the untested items, like the air path control solenoids are disconnected (simulating failure) the self-test function still passes OK. However, when there is a fault in these components the reference solution cannot be measured, which means the fault is “detected” indirectly by the failure of the reference test. Faults in many of the untested parts would be “detected” by this indirect operation, as the reference solution reading would be found to be incorrect and the tester would be disabled. The design assumption seems to be that any faults in these untested areas will cause an obvious failure of operation that will be detected in the normal course of operation.

For our purposes the most interesting type of fault is one that is not covered by direct self-test, nor by indirect testing of the reference solution. The most interesting are the faults that would never be detected and would still affect the accuracy of the final results. Several examples of this kind of fault have come to light.

IR FILTER OPERATION:

There is a section of the source code in a file called “CHACTIBE.C” that can quietly downgrade the number of filters used from five to three, or even to just one. This seems to have

been done to allow the device to remain functional in a degraded condition, but again this is done with no change in the report data and it is not checked during the self-test function. Any given test run on this machine could be measuring data from one, three, or five filters without reporting this in any way or otherwise halting the test.

TUBING HEATERS/RFI DETECTION:

The I5000 has internal and external heating elements lining the breath tubing. The function of these heaters is to stabilize the temperature of the breath sample, and to prevent condensation from forming inside the tubing. One information source pointed out that if the temperature of the breath varies by 1 degree C you get a 7% variation in the reading, so controlling the temperature of the sample is very important. Condensation inside the tubing is said to cause cross contamination between samples, which shows another important need for the tubing heaters. The heating elements also double as antenna for the detection of radio frequency interference (“RFI”) during testing. RFI has been shown to affect infrared spectrometry measurement of breath alcohol.

The external main breath tube and the reference sample tube both have heating elements embedded into them. Both of these heaters are connected to the I5000 by external connectors into the body of the device. The main breath tube is connected by a standard “RCA” type connector (as used on consumer audio gear). On the unit I had for testing, this connector was loose and was subject to stress depending on how the breath tube was handled. The reference tubing heater had a better quality connector, but this connector was on the back side of the unit and could not be seen if it was connected (or not) in normal operation. When either of these connectors is disconnected to simulate failure, the machine does not report any fault from the

self-test or during normal operation of a subject test. While disconnected, these tubes have no heat at all and they cannot detect RFI and report it back to the device.

Also there is a long internal tube that connects the external breath sample tube to the internal sample chamber. This tube has a heater along its entire length. This heater can be disconnected (simulating failure) with no self-test error and no errors in normal operation.

In fact, all three heater elements can be disconnected at the same time and the I5000 will not notice at all. Given the dual problems of cross contamination from condensation and the lack of temperature control of the breath sample, it would seem that the lack of any form of testing for these heaters is a serious oversight. The device does not have any internal diagnostic function that measures the heat of the tubing or whether the device is even attempting to detect RFI.

AIR PURGE:

A key part of the operation of the I5000 is the function of an “air purge.” This is the drawing of room air into the sample chamber to clear the chamber and simultaneously test the ambient room air for contaminants. The big problem here is that the system does not measure the volume of air drawn in during this purge. The breath detector measures pressure, but the air purge uses a vacuum to draw air in. The pressure sensor is set to detect only positive pressure and seems to not notice the negative pressure of the purge function. During a breath, the pressure is detected (older devices just have a pressure switch, newer one measures the amount of pressure), but during the air purge vacuum this detector is ignored. In fact, if you cap the breath sample tube, so no purge air can be drawn at all, the self-test will pass with no faults, and the system will seem to operate normally.

The same air pump is used for the air purge and the reference solution testing. The reference test requires very little air volume to generate a reference reading. The air pump could

be quite degraded, hardly working at all, and since the air purge does not measure any air volume, the purge function could effectively be failed, and no part of the I5000 would detect this. This means that the actual operation of the purge air pump is never tested. It is assumed that if the air pump is faulty the reference solution test will fail. But it takes very little volume of the reference to generate a test result, and anything in the breath air path (solenoids, tubing, pump degradation) could be restricting the air purge and the device would not know this.

OPTICAL PATH:

The operation of the system is based on an IR spectrum from a light bulb. This light bulb is exposed inside the I5000 and has room air forced around it by a cooling fan. The light bulb and the optical path into the measuring chamber is subject to dust buildup, as there is no protection around this part of the design. Depending on the nature of the dust in the local environment, the spectrum of light from the bulb could be affected by dust buildup without any part of the I5000 detecting this. Any shift in the spectrum of light from this bulb would affect the ability of the device to detect the differences between alcohol and other chemicals. This would degrade the operation of the I5000 with no indication from self-test or reference testing (since the reference is pure alcohol).

FALSE PRECISION:

I analyzed the temperature measurement system by disconnecting the heater from the main testing chamber and observing the values reported as the temperature in the chamber. Since there were no heaters running, and still air in the room, the temperature should have only changed by tiny amounts.

After the warm-up time, this message was shown in the display.



Standard Engineering practice contains the concept of implied precision. Implied precision is related to the number of digits on the right hand side of the decimal point. In the case of this display on the I5000, we find that temperature is reported with 4 digits of precision.

For any number written with a decimal value, the accuracy of the value should be reflected in the number of digits of precision to the right of the decimal point. In a typical electronic measurement system, accuracy would be implied to be +/-1 digit of the least significant decimal value plus some specified percentage of total accuracy. For instance, a number of 24.0000 would imply that the actual value is between 23.9999 and 24.0001. In any measurement system, there must be some additional accuracy in the underlying system to allow for rounding of the least significant digit to the more appropriate value. So for a legitimate reading of 4 decimal places, there would be an implied measurement precision of slightly more than 4, to allow for the digit transition to occur at 50% of the least significant digit value.

After some more time went by with the device running, the temperature reading drifted slowly upward due to the heat of the IR light source. If the measurement system was in fact capable of 4 digits of precision I would expect to see the least significant digit change in some sort of sequential manner. (1,2,3,4...) In fact, the values in the 4-digit decimal field always

changed by an increment of .0312 or .0313. (The actual incremental change was always .03125, and the print out math in the source code rounded this increment to either 0312 or 0313). This means that the implied precision shown in this display by reporting 4 decimal digits is completely misleading.

For the measurement system to conform to standard industry practices, this measurement should not be displayed with more than 1 digit on the right hand side of the decimal point. For every $1/10^{\text{th}}$ of a degree value, the measurement sub-system can only produce 3 incremental readings. These readings should be rounded to the nearest 10th of a degree and not expanded into 4 digits. The method chosen to display the internal value in this display has been deliberately inflated to create the appearance of a highly precise system. This is called False Precision. In fact, the actual quality of the measurement performed by the internal hardware is 1000 times worse than the implied precision in this display.

MISSING ITEMS:

In analyzing the self-test section of the I5000 source code, important questions remain unanswered. It's easy to see which components are actively tested, but it is also important to discover what components and functions have been ignored. Schematic drawings for the I5000 were obtained from an internet search. The section of interest, the analog board, was compared to the board inside the test I5000 and found to match the drawings. The schematic for this section is included in this document. Upon analyzing the schematic and comparing it to the source code for the self-test function, several glaring omissions were noted.

Power Supplies:

Throughout the system, there are multiple power supplies, (plus and minus 15 volts, plus and minus 12 volts, plus 5 volts and plus 24 volts). None of these internal power supplies have

any means of self-test. Also important is to note that the plus and minus 12 volts (VR1 and VR2) are not tracking power supplies, so any drift or error on one of these two critical analog power sources could contribute to numerous errors without being detected. The most common failure of the kind of power supply used in this device is an internal hum, or 60hz line frequency getting into the internal power lines. This would create unstable readings. Since there are so many auto-calibration functions going on, an unstable internal supply would be corrected during the purge or reference test, but could cause the actual test to vary.

Internal Reference Voltage:

During self-test, there is an internal check that measures some value at .100, .200 and at .300. In examining the schematic for this section of the device, there is a feedback path created between a digital to analog converter “DAC” (U10) and the analog to digital converter “ADC” (U12). It appears that this feedback path is used in the internal check section of the self test. The flaw here is that both sections are fed by the same voltage reference internal to U12. The reference voltage from U12 is buffered by U2/B and fed into U10 as its reference. This is an invalid test, since both the DAC and the ADC are only displaying ratios of a single reference. Since both sections are referenced to the same internal source, that source could be in error without this feedback loop being able to detect it. The only way that this type of feedback loop could provide a valid self test would be if there were two different references that could be compared to each other.

The test is using two converters to measure each other, but both converters are ratio-based. They only report the ratio of the sample to the reference. During actual testing the sample is compared to the reference, but during internal testing, the reference is compared to itself. No matter what the reference really is, it will always pass this test. For a self-test to be valid there

needs to be a second reference. If two references do not match, then one of them must be wrong, but a single reference will always pass when compared to itself.

IR Detector Cooling:

A critical part of the optical detector is a thermoelectric cooling system. This cooling system has a temperature regulator but **nothing** about the regulator U2/A or the temperature of the detector is monitored or checked by the self-test routine. Due to the number of internal compensations, this thermal regulator could be unstable, or substantially malfunctioning without the system being able to detect this fault. Faults in the thermal cooler could result in non-linear behavior or drifting values. Since zero and max values are forced to be correct by the source code, nothing would detect errors in the linearity of the detector.

AGC:

An interesting feature of the analog section is an automatic gain control “AGC”. This is an analog feedback path which adjusts the gain of the IR detector in order to keep the signal within the linear range of the analog components. This entire sub-section has no means of self test, nor any means of monitoring its operation. Instabilities in this section would not result in direct errors, but instead would reduce the sensitivity to contaminants by varying the gain between the individual IR channels. The untested area consists of 7 op-amp (U1, U6/B, U3/A, U6/A, U3/B, U2/D, and U2/C), 3 transistors (Q1, Q2, Q3), and numerous resistors and capacitors. An interesting component is labeled CR1. This is an optically coupled resistor used to control the gain of the input stage. This type of component is often subject to non-linear behavior with aging and there is nothing in the system that would detect this.

In fact, there is nothing in the system that could detect if this large sub-system was degraded or operating normally. This unmonitored AGC could mask faults in the IR detector

cooling, light source, filters and power supplies. Faults here could completely compromise the ability of the system to detect contaminants without affecting the reference solution check. The zero value is forced during air purge. There is a section of code that "adjusts" the calibration values if they were too far out of normal (without reporting this). This kind of auto-resetting gets very problematic if the underlying hardware gets unstable. Power supply hum, Auto-gain instability, Sample and hold errors, purge faults, drift in the cooling system or heating system, even dirt on the optics, these would all get hidden from the readings by the internal value resets, and all could affect the actual results if any instability was present.

Sample and Hold:

There is a peak-detecting sample and hold circuit. (U3/A, U7/D, CR6, C31). This is critical to accurate readings but contains no provisions for self-test. In fact, the circuit could be subject to drift or other error accumulation and be undetected in normal operation. Errors in this part of the circuit would be manifest as cross talk between the IR channels, and the system has no means of detecting that. This would also degrade the ability of the system to detect contaminants without affecting the reference solution check.

SELF-TEST CONCLUSION:

Even after conducting the limited review I did with the source code and the schematics, numerous serious self-test errors have been identified. With further analysis, it is highly likely that even more self-test problems and omissions could be discovered. The scope of my analysis has already revealed troubling problems with the integrity of this system.

SLOPE DETECTION & INTERFERENT HANDLING

The slope detection section of the source code is particularly concerning to me as a software analyst. This section is very unorganized and contains wholesale “hacked-in” changes that appear largely untested. I suspect that this section of code will have numerous unexpected faults and errors, which is detrimental to the integrity of the machine since the slope detection is responsible for detecting the suitability of the breath sample’s volume and pressure, plus it helps with detection of interferents and contaminants in the breath sample.

THE TESTING PROBLEM:

Since I had no expectation of getting to copy or use the actual source outside of CMI’s headquarters in Owensboro, Kentucky, I abandoned any plans to do a “test bench” or static code analysis. Without direct code access, I had to develop an indirect approach to demonstrating the weaknesses in the device with the slope detection and interferent detection.

Any embedded system will reveal its weaknesses in the “corner cases”. These are the areas of operation that are near the limits of normal operation. If you were testing a car you would not drive it in a straight line for 100 feet and then declare it fit for the road. A real test of a car would include hard corners, quick stops, floorboard acceleration, and behavior on poor roads. Like a car, these I5000 devices are used out in the field, not in a controlled laboratory setting.

A real test for any embedded system MUST include variations to the point of failure. What is the min/max breath volume and what effect does this have on results? What is the effect of various levels of contaminants and how are results effected by non-error causing amounts? What is the result of changes in pressure during a sample? Does the mouth alcohol lock-out function work and at what threshold will it still produce a reading?

The hardware and the firmware work as a unit, you cannot separate problems into one or the other category without well-controlled testing and careful diagnosis of the root cause. Some changes in the source code are to compensate for deficiencies in the hardware. Since the hardware requires replacement parts to correct problems, most hardware problems have workarounds created in software. This is a common practice since firmware is much easier and cheaper to upgrade than changing the hardware itself, but this is not the best way to ensure integrity of the operation.

For example, there is a section of the source code that averages the readings of the IR filters. According to the comments in the code, this is partly to correct a problem with the accuracy of the filter wheel trigger. The “correctness” of this kind of code is very hard to verify. It is already compensating for one problem, so it can be very hard to see if this has created a new problem. The averaging will change the overall response time of the data channel and may affect the slope detection. Interactions like this can create unpredictable results. The only way to test these issues is to generate repeatable inputs to the source code and test how it responds to these “corner cases”. This problem (how to generate repeatable input profiles) is the motivation for creating a real-time breath simulator.

THE TESTING RIG:

Because of the restrictions on the source code access and the limits on the scope of my analysis, I can only test the function of the source code using the entire I5000 as a closed box. In order to do this, I have built what amounts to a "real life" breath simulator.

The factory simulator solutions provide constant pressure and well-controlled samples. My breath simulator can create dynamic breath profiles, creating variable mixtures of clean air, alcohol, methyl-ethyl-ketone (MEK), and vinegar. Other combinations can be created in sets of

four. It can do this with reproducible pressure profiles. This can directly demonstrate the weaker parts of the source code, showing the dependance of the I5000 on controlled breath pressure, and its poor resolution in separating primary ethyl alcohol from common contaminants.

The tester is based on a precision 4-channel air valve body. It has a repeatability of $\pm 0.02\%$ so it serves as a good reference source for controlled air pressure. I'm using standard interface parts and existing programs for the profile creation and playback, so nothing new has to be written for this. One air channel will go through distilled water to represent clean breath, one will use the existing simulator solution, and the other two will go through the dilute contaminant solutions.

The breath simulator is able to create profiles that show weakness based on my study of the source code. It is able to reproduce the factory test sequences with profiles that match the simulator solution tests, and then show that "real life" profiles generate different results. It will work on any I5000, or any other breath analyzer, in real time, and does not require opening the I5000 to show the behavior of the device.

This Testing Rig allows controlled input to the hardware, which in turn provides controlled inputs to the source code. This lets me use the I5000 hardware as a simulator for the code and show errors in the overall function. Varying the ratio of clean air to reference simulator solution can simulate mouth alcohol (an early AL peak) or the effect of a clean air breath at the start of a test (a late AL peak). Contaminants can be introduced early or late in the breath cycle, (simulating a mouth or deep lung source) and this is another thing that I think the code cannot handle very well.

Poorly written source code always produces erratic output with unexpected inputs. The breath profile is the only real "input" to the entire device, so it makes sense to create a way to

generate controllable inputs. In the case of the I5000 there is lots of room for errors in the variations of breath pressure / time and small contaminants causing larger than expected effects.

There is even a possible interaction between the analog AGC (hardware that processes the data from the optical section) and the breath profile. This involves the physics of the optical section, the time constants of the analog amplifiers, and the source code. It cannot be explored without a breath profile simulator.

This is an unconventional approach to a source code review, but it gets around all of the obstructions placed on the Source Code access, and creates obvious results that should be easy to understand.

TESTING RESULTS:

Acetone, MEK, and vinegar could occur in breath. There is a test in the software for vinegar, but it is not used in any calibration testing, so the code is very weak here.

A wide variety of Volatile Organic Compounds (VOCs) can occur in the environment from industrial processes, paint, glue and common building materials. This Source Code was written well before these kinds of contaminant were a common concern and it contains no methods for dealing with them.

It has been known for years that a person can give two different breath samples and vary the amount of pressure (or belch first) and get very different results on the Intoxilyzer. This is because of defects in the slope detection section of the source code.

The reading of a breath alcohol value is never a constant. The test chamber is filled with different air before and after a test, so the concentration of vapors always varies as a test is run. The function of the slope detector is to find the best, most consistent reading in the “slope” of values that come from the optical test section.

There are sections of the source code that contain many vague and poorly documented changes. These changes relate to the slope detector, the isolation of contaminants, self-adjustment of internal values, compensation for unstable values from the optical section, and changes in the way errors were recorded and reported. All of these items are crucial to the proper operation of the device.

I see much weakness in the source code of the slope detector. It has no method of dealing with a slope in the contaminants. My testing has shown that sub-critical levels of contaminants will affect the results without setting any warning or error message. There are many normal field variations where the device can be very inaccurate. Variations such as the presence of sub-critical contaminants can cause incorrect readings and different pressure profiles can create large variations in the reported results.

Inconsistent results are the very definition of poor quality control. One side effect of this is that even in testing it can be hard to get reproducible results. I can cause the device to do wrong things, but it does different wrong things every time. Using low pressure breath really seems to confuse it. There are no error reports, everything on the printout looks normal, but several wrong things can happen. A set of these problems can be traced to faulty operations in the Slope Detector source code.

OBSERVED ERRORS IN THE SLOPE DETECTOR CODE:

FAILURE TO CORRECTLY END THE SAMPLE:

The air volume is not directly measured by the hardware. The hardware measures pressure, and the source code translates pressure over time into volume. The algorithms for this translation do not work correctly, or the pressure transducer is non-linear and they have not correctly compensated for the sensor transfer curve.

This error shows up most clearly in the way that the sample beep function can fail to end the test. With low pressures the beep may never stop, even after 9.9 liters have been blown. At low pressures the beep may restart after it has stopped, and the results will be different if the breath pressure continues after the beep has restarted. There is a pressure input that will cause the device to just hang up at the sampling stage and never time out. This kind of problem should never occur if the slope detector code was working correctly.

FAILURE TO DETECT MOUTH ALCOHOL:

When a breath sample shows an early high peak, the most likely cause of this is alcohol in the mouth instead of in the deep lung air. This should be detected and reported as an invalid sample, and NOT reported as a valid blood alcohol value.

The mouth alcohol detection does not work reliably, causing different pressure profiles to result in widely different readings. Replaying the exact same “early peak” breath profile produces different results: about 50% of the time it will report an invalid sample, and the other 50% it will report the early peak as the actual result.

FALSE RESULTS:

*Using just air through distilled water, I found a pressure profile that causes a false positive result. I have been able to get a 0.019 reading from room air using a very low pressure that ramps up to a normal pressure.

*With small amounts of acetic acid (vinegar) and low breath pressures, it is possible to get an alcohol result *without there being any alcohol present*.

*I have seen a pressure profile with very small amounts of mixed contaminants create false readings as high as .400 Most of the time this was reported as an invalid sample, but it is very interesting that non-alcohol samples can create false results.

With variable pressure, the I5000 is very erratic. The sloppiness I saw in the slope detector source code produces wildly inconsistent results.

EXTREME SENSITIVITY TO GLUE VAPORS:

In between testing of this device, I used a small amount of industrial glue (E6000) on a unrelated project. This glue uses tetrachloroethylene as a solvent. The I5000 failed to pass the air blank test while any of this solvent was in the air. Further testing showed that an extremely small amount of this solvent in a breath sample will be reported as alcohol. Glue vapors contain tetrachloroethylene, formaldehyde, ether, methyl alcohols, unpolymerized monomers, and acetone. Only acetone is directly addressed in the I5000 source code. Analyzing the failure of the I5000 to discriminate these various chemicals is outside the scope of my work. However the Source Code of the I5000 shows a lack of ability in sorting most of these contaminants from the breath sample.

Since glue vapors are common in many construction materials, the failure of the I5000 to discriminate between these solvents and alcohol could cause false readings from people exposed to manufactured housing or other sources of glue vapors.

SUMMARY

My review of the Intoxilyzer 5000EN Source Code has shown that the authors of the code had trouble getting several parts of the system to function correctly. These sections have multiple edits, partial notes, and disabled sections of old code. **These are all indications that the programmers were “hacking” on these sections, trying to get them to work correctly by a process of trial and error.**

A review of the Revision Document: “Minnesota Source Code Modification” reinforces the impression of development by trial and error. **Over the history of the I5000 field operation, many errors have been corrected, and new errors introduced by the repeated attempts to address deficits in the design of the device.**

As a result of this trial and error method of developing the source code, the I5000 is a “brittle” device. **In real-life conditions of variable breath flows and minor contaminants, the poor design quality of the I5000 hardware plus the errors and omissions in the source code produces unpredictable results.**

The I5000 uses only five wide band IR filters, and does not calculate slope values for the contaminant readings during a breath sample. **This allows many industrial chemicals and even food odors to be incorrectly reported as alcohol content.**

CURRICULUM VITAE

Timothy E. Black

6754 Howe Rd Middletown Oh 45042
Web Site: <http://www.quantalink.com>
email: tblack@quantalink.com
home 513 217 9679
cell 513 267 9653

26000 Aldercroft Heights Rd
Los Gatos CA 95031

Overview

- Has had complete system design responsibility for many projects.
- Expert in Hardware/Software system design, integration, and testing
- Has brought up new boards, written device drivers and isolated difficult errors.
- Has extensive experience with Power control, USB, Audio, Video, and Servo hardware.
- Skilled with Oscilloscopes, logic analyzers, and other lab equipment.
- Skilled in reverse engineering of devices and protocols.
- Skilled in rapid prototyping, system debug and problem analysis.
- Has worked with end users to create systems based on needs.

Key Skills

- **Senior Expert** coding in "C" for real-time embedded systems.
- **Senior Expert** in embedded system design and system level troubleshooting.
- **Senior Expert** in hardware/software interaction and debugging.
- **Expert Analog design skills**; sensors, servos, audio and video.
- **Expert fabricator with complete off-site development lab facilities.**

Interface Skills

- Microcontrollers**; Hardware and firmware for AVR, PIC, Z8, Z80, Z180, Rabbit 2000
- **Analog design**; op-amps, a/d, d/a, power supplies, sensor conditioning, and accelerometers.
- **Power Management**; Smart batteries, chargers, solar chargers, SOC metering.
- USB**; kernel drivers, diags, protocol analysis
- Ethernet**; drivers, diags, line protocol
- Video**; drivers, stereo vision object location
- RS232, 422**; drivers, diags, protocol
- Smbus**, Smart battery interfacing, power management systems
- IIC**; drivers, diags, protocol

Work History:

Aug 1999 – current: Consultant, owner of Quantalink LLC

Many of these projects are still covered by NDA agreements, so details may be limited.

2010: Product investigation for ongoing litigation,

This project involves a very restrictive NDA and involves reverse engineering of a analytical device to find errors in the design.

2009-2010: Design of an audio sensing alarm for large building intercom systems.

This project has resulted in a new product for the client, with a 90% reduction of BOM costs from the previous solution.

2009: Research into low power, network controlled lighting for retail stores.

Provided power analysis and development of new lighting control methods.

2009: Development of a new form of solar panel for a client in Canada.

Hardware and firmware for on-panel high voltage power conversion for improved solar panel interconnection.

2008: Design and development of a toy robot.

Mechanical, electronic, and firmware development for a robot arm that can fold into itself and move board game pieces (for remote playing of GO).

2008: Design and development of a automatic camera.

hardware and software design, with board layout for a very small smart camera.

2008: reverse engineering for competitive analysis.

Covered by NDA.

2007: Hardware and firmware lead designer for created the LENA recording device for Infoture at Pemstar. This is an ultra-low power audio device using the BelaSigna 200 and SD flash memory.

As of 2010 this device is still on the market.

2006: worked on control firmware for a Xray system at VARIAN

Contracted with Conexant

- 2005: Debugged DDR ram layout problems
- 2005: Did firmware bring up testing on a new SOIC.
- 2005: Worked with ASIC engineers to isolate errors in RTL.

2004: Patent research for Silicon Valley Expert Witness Group

Contracted with Pemstar on the Land Warrior System.

- 2004: Lead for the Power Interface Controller.
- 2004: Debugged SMBus battery problems
- 2004: Debugged power controller hardware, made design improvements
- 2003: Hardware engineering and debug of main computer IO board
- 2003: Wrote firmware for Atmel AVR Mega16

2003: Co-inventor for Patent # 7154526 filed by Fuji Xerox Palo Alto Laboratory
“TELEPRESENCE SYSTEM AND METHOD FOR VIDEO TELECONFERENCING:

2002: Applied for Patent 60/446,586 filed by J. E. Taggart
“ADDRESSABLE MULTI_COLOR LIGHTING DEVICE”

2001: Developed an object inspection computer prototype for the NASA shuttle processing facility.

Worked for Sun Microsystems on the firmware and diags for the SunRay product line:

- 2001: Helped troubleshoot stubborn manufacturing problems.
- 2000: Developed Ethernet and USB diagnostic test programs.
- 2000: Created test fixture for extended burn-in audio testing.
- 2000: Did extensive testing of USB device interaction and fault analyses
- 2000: Created a "patch gadget:" to enable testing of a new ASIC (saved \$500k)

Other Projects covered by NDA agreements, have included proprietary prototype development for several startups, a myoelectric pre-amp, and custom control systems for a private “theme park”.

**Feb 1999 – Aug 1999: Senior Product Engineer, Avio Digital Inc.
(www.aviodigital.com)**

Avio tried (and failed) to design a new type of home networking. My tasks included working on reference designs, doing FCC type 15 measurements, working with EMI testing labs, and reverse engineered competitive products.

Feb 1998 – Feb 1999: Interface Wizard, Kodak Imagination Works (KIW)

KIW was a short-lived advanced research group for Eastman Kodak Entertainment Imaging. Prototypes were created to test new forms of user interfaces. A patent was filed for an invention I did using dual color TV cameras to track objects for a new kind of arcade game platform. Custom Video multiplexes were built and ‘C’ code written to do object tracking at full video frame rates. Interface devices were created to control lights and motors using PIC micros.

July 1997 – Jan 1998: Senior Software Engineer, Be Inc.

Be Inc. made BeOS, a multi-processor, multi-threaded, multi-media operating system. Tasks include writing device drivers (CAM-SIM SCSI format), testing and debugging kernel functions, and creating debugging tools. (Be Inc. is gone...)

Jan 1997 – July 1997: Senior Tech Support, Microtec VRTX RTOS

Responsible for interfacing with customers and recreating problems with the VRTX Real Time Operating System. Used HP 16500 logic analyzer to verify behavior of program

test cases. Did testing and verification of new releases of compilers, debugging tools, and RTOS packages. (Left when Microtec was acquired by Mentor Graphics.)

Nov 1994 - Dec 1996: Independent Consultant

A computer consulting office called Tech Knowledge is operated in Sedona. This office provided custom programming and system design for a wide variety of clients. Tasks included designing and testing a controller for a RAID based MPEG 2 Video server.

Aug 1993 - Oct 1994: Consultant , Spencer & Spencer Systems

Provided technical leadership for a team converting 15 year old mainframe PL/1 to GNU 'C'.

Wrote a drawing generator in Gnu 'C' on a DG/UX 88k system with ESQL access to an Ingres database on a HP - Apollo X station. Wrote a 'C' program to interface a production testing device to a PC. The program provided Statistical Process Control charting from test runs. Rebuilt a 20 year old mainframe FORTRAN IV program to run on a Data General AViiON 88000 system and added a DXF file feed to Autocad Ver 11.

June 1992- July 1993 Senior Programmer, Print Research Tech.

Programming the main software of a press control in Metaware HighC 29K and 29K Asmb. Wrote the low level drivers for the AMD Mace Ethernet controller and ported the PhoenixPage postscript source to the PRT multiple- processor 29000 RISC system. Used Nucleus RTOS. Worked on the real time code for bringing up the new multi-processor system board. Did some system work for the in-house network (10baseT Ethernet, NFS) on Sun workstations

Oct. 1990 - July 1992 T. E. B. Engineering.

Owner of T. E. B. Engineering, Providing custom software to the cable and broadcast industries. Installed automation systems at many sites across the Midwest. These systems include an election tracking database package at WMC Memphis, a controller for movie playback with 50 VCRs in Warner Cable Houston, a system controller for WCSC, Charleston SC. and a retro fit system for Spectradyne, San Francisco. Tasks included on site troubleshooting and debug of new installations.

Mar 1984 - Oct 1990Asst., VP Engineering for Scripps Howard Broadcasting

Designed, wrote, and installed four complete newsroom computer systems, two commercial spot playback systems, two net delay systems, and several other systems. These systems included on-site installation, troubleshooting and hardware/software debugging.

May 1980 - Mar. 1984 R&D manager for KAVCO, Dayton Oh.

Responsible for the creation and production of a complete family of automation products. Managed the R&D office with six employees in Terre Haute. Wrote a number of embedded Z80 programs. Wrote a complete database system for automation support. Designed the hardware for interfacing to tape decks and switchers. Provided on-site support at trade shows and new system installations.

July 1978 - May 1980 Chief Eng. of E.S.I. Terre Haute IN.

Wrote an embedded 8080 program (4k in 1702 EPROM's) for a security system. Was responsible for repair of many different types of electronic systems, including NC punches, Sheeter mills, 50 HP DC motor servos, Video production and security systems.

June 1974 - July 1978 Electronic Tech for B&A Electronics Terre Haute IN

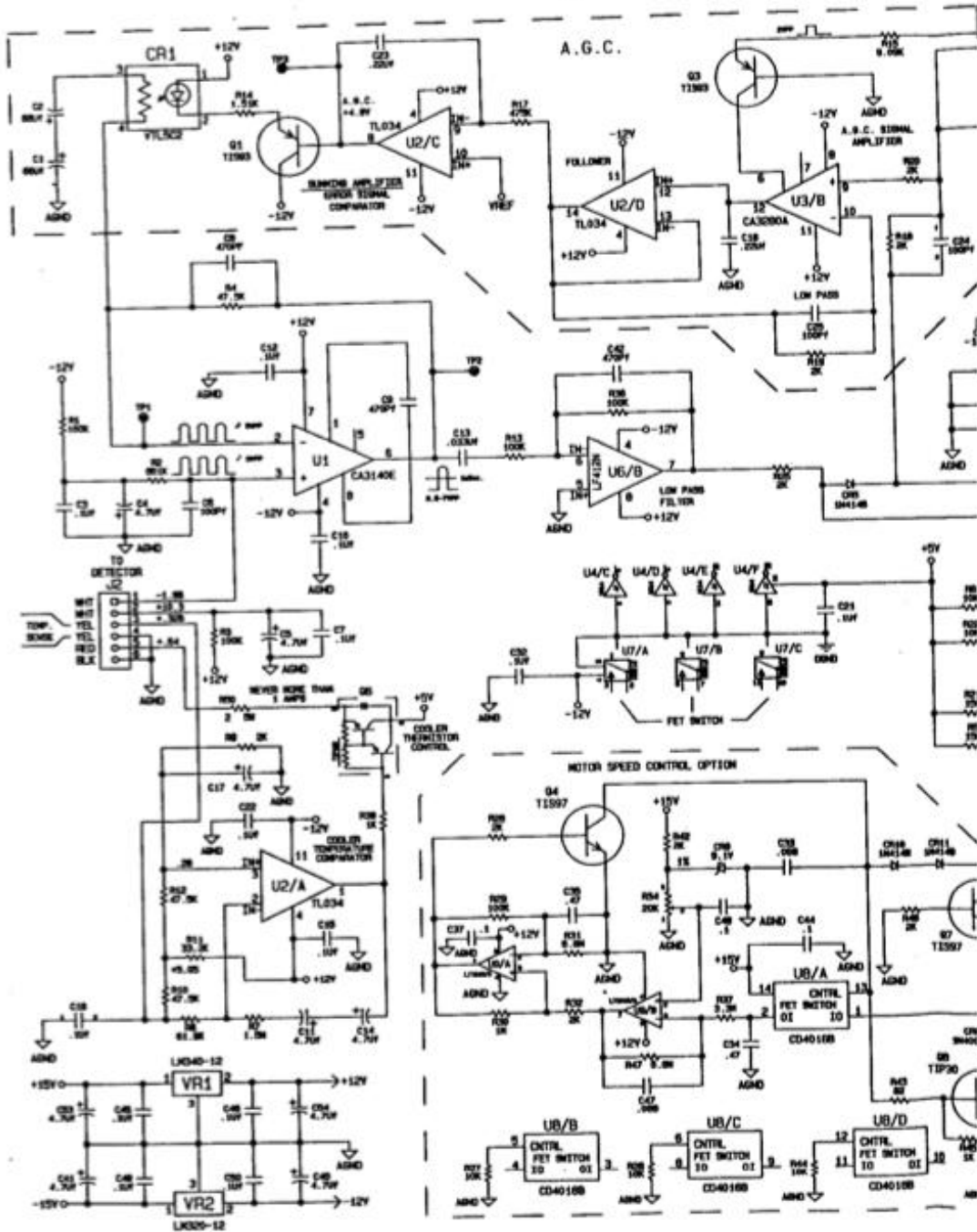
Designed and built audio/video editing systems and repaired aircraft avionics.

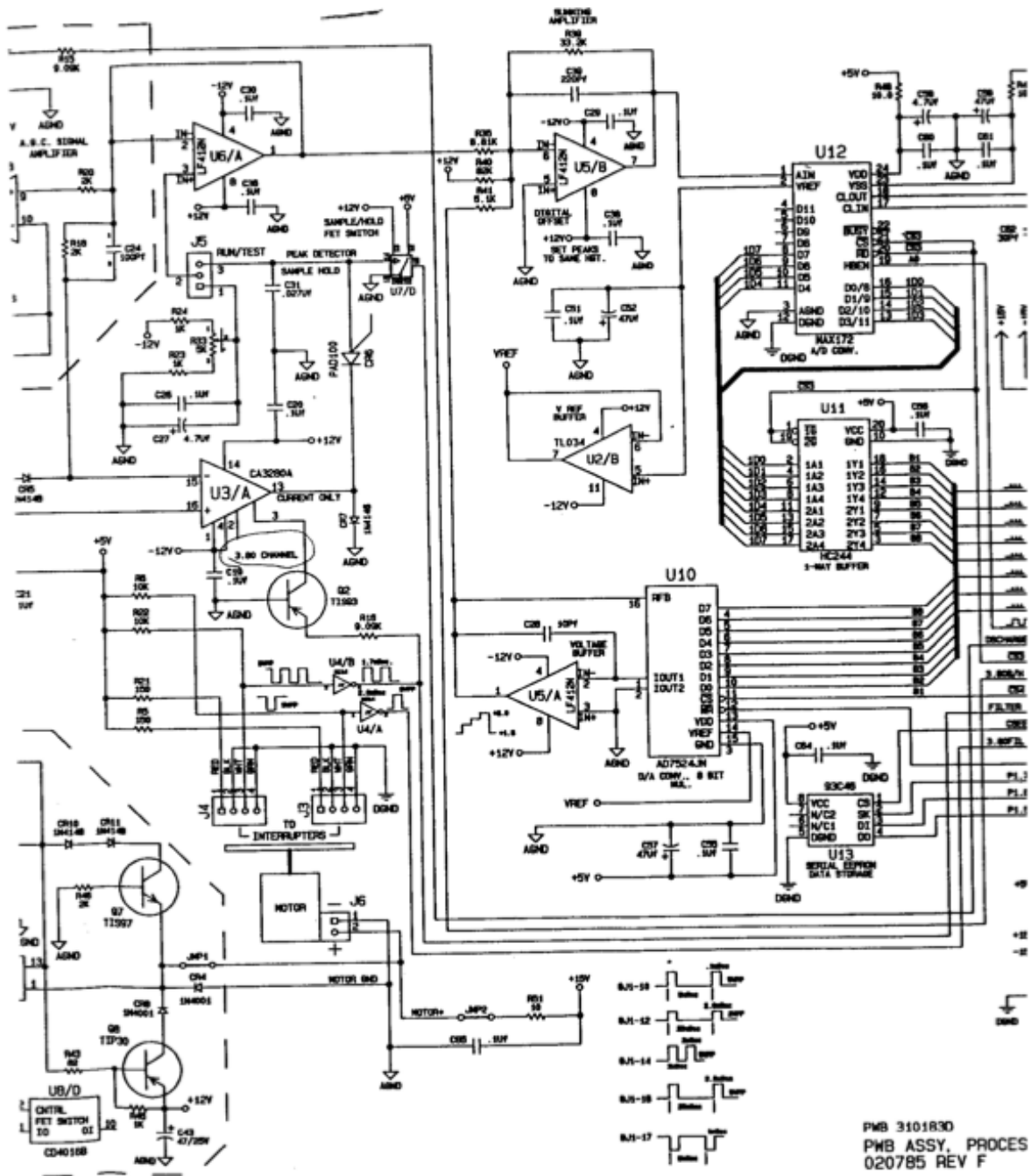
1972 - 1975 Indiana State University EE and CS majors

Public Art Installations

- **L2K:** A 2000 foot long wiring harness with 200 controllers and 2000 light pods
- **The Hypknowtrons:** Spinning blades covered with hundreds of full color LEDs
- **STS:** A xenon spotlight converted into a audio transmitter with a 2800 watt class A amplifier
- **Voice of Fire:** 13 Flame speakers on a 9 foot wide steel jewel box

Analog section drawings.





Glossary

ADC:

An **analog-to-digital converter** (abbreviated **ADC**, **A/D** or **A to D**) is a device which converts continuous signals to discrete digital numbers. The reverse operation is performed by a digital-to-analog converter (**DAC**). Typically, an ADC is an electronic device that converts an input analog voltage (or current) to a digital number proportional to the magnitude of the voltage or current. However, some non-electronic or only partially electronic devices, such as rotary encoders, can also be considered ADCs. The digital output may use different coding schemes, such as binary, Gray code or two's complement binary.

AGC:

Automatic gain control (AGC) is an adaptive system found in many electronic devices. The average output signal level is fed back to adjust the gain to an appropriate level for a range of input signal levels. For example, without AGC the sound emitted from an AM radio receiver would vary to an extreme extent from a weak to a strong signal; the AGC effectively reduces the volume if the signal is strong and raises it when it is weaker.

DAC:

In electronics, a **digital-to-analog converter** (**DAC** or **D-to-A**) is a device for converting a digital (usually binary) code to an analog signal (current, voltage or electric charge). An analog-to-digital converter (ADC) performs the reverse operation.

False Precision:

False precision occurs when numerical data are presented in a manner that implies better precision than is actually the case; since precision is a limit to accuracy, this often leads to overconfidence in the accuracy as well.

In science and engineering, convention dictates that unless a margin of error is explicitly stated, the number of significant figures used in the presentation of data should be limited by the precision of those data. For example, if one instrument can read to tenths of a unit of measurement, calculations related to data obtained from that instrument can only be confidently stated to the tenths place, regardless of what the raw calculation returns or even if other data used in the calculation can be obtained more precisely. Even outside these disciplines, there is a tendency to assume that all the non-zero digits of a number are meaningful; thus, providing excessive figures may lead the viewer to expect better precision than actually exists.

Firmware:

Software is written as Source Code (readable by people). This is compiled into Object code (readable by computers). When the Object code is stored in fixed memory (PROMS, or ROMS) the finished Object code is called Firmware.

IR:

Infrared (IR) radiation is electromagnetic radiation with a wavelength between 0.7 and 300 micrometres, which equates to a frequency range between approximately 1 and 430 THz.^[1]

Its wavelength is longer (and the frequency lower) than that of visible light, but the wavelength is shorter (and the frequency higher) than that of terahertz radiation microwaves.

OPAMP:

An **operational amplifier**, which is often called an **op-amp**, is a DC-coupled high-gain electronic voltage amplifier with differential inputs and, usually, a single output.[1]. Typically the op-amp's very large gain is controlled by negative feedback, which largely determines the magnitude of its output ("closed-loop") voltage gain in amplifier applications, or the transfer function required (in analog computers). Without negative feedback, and perhaps with positive feedback for regeneration, an op-amp essentially acts as a comparator. High input impedance at the input terminals (ideally infinite) and low output impedance at the output terminal(s) (ideally zero) are important typical characteristics.

PROM:

Programmable Read Only Memory. A hardware device (a memory chip) that can store Object code for use in an embedded computer. PROM chips are often have to be removed from the device and reprogrammed in a external programming socket. Older versions are erased by exposing a window on the chip to ultraviolet light. Newest devices can be reprogrammed in place by a remote computer.

RCA connector:

An **RCA connector**, sometimes called a **phono connector** or **cinch connector**, is a type of electrical connector commonly used to carry audio and video signals. The name "RCA" derives from the Radio Corporation of America, which introduced the design by the early 1940s to allow mono phonograph players to be connected to amplifiers. They began to replace the older TRS connectors (also called jack plugs) for many other applications in the audio world when component high fidelity systems started becoming popular in the 1950s. The connection's plug is called an **RCA plug** or **phono plug**, for "phonograph". The name "phono plug" is sometimes confused with a "phone plug" which refers to a TRS connector plug.

Sample and Hold:

In electronics, a **sample and hold** circuit is used to interface real-world signals, by changing analog signals to a subsequent system such as an analog-to-digital converter. The purpose of this circuit is to hold the analog value steady for a short time while the converter or other following system performs some operation that takes a little time. In most circuits, a capacitor is used to store the analog voltage, and an electronic switch or gate is used to alternately connect and disconnect the capacitor from the analog input. The rate at which this switch is operated is the sampling rate of the system. In a sample and hold circuit the switch opens for a very short duration, while in a track and hold circuit the switch can be opened continuously.

Slope Detector:

The section of the source code that is used to determine the valid values from the range of values recorded during the breath testing. A valid sample should rise to a stable reading soon after the breath starts and hold this value over some time. The Slope Detector is supposed to

reject tests that show too much variation over time, or sample sets that peak early and then decline.

Thermoelectric Cooling:

Thermoelectric cooling uses the Peltier effect to create a heat flux between the junction of two different types of materials. A Peltier cooler, heater, or thermoelectric heat pump is a solid-state active heat pump which transfers heat from one side of the device to the other side against the temperature gradient (from cold to hot), with consumption of electrical energy. Such an instrument is also called a Peltier device, Peltier diode, cooling diode, Peltier heat pump, solid state refrigerator, or thermoelectric cooler (TEC).^[1] Because heating can be achieved more easily and economically by many other methods, Peltier devices are mostly used for cooling. However, when a single device is to be used for both heating and cooling, a Peltier device may be desirable. Simply connecting it to a DC voltage will cause one side to cool, while the other side warms. The effectiveness of the pump at moving the heat away from the cold side is dependent upon the amount of current provided and how well the heat can be removed from the hot side.