

Source Code Primer:
The Inside Scoop on ALL Breath Instrument Computers and Electronics¹
National College for DUI Defense, Inc and
Texas Criminal Defense Lawyers Association
14th Annual Mastering Scientific Evidence in DWI/DUI Cases
April 20th, 2007 – Dallas Texas
Presented by: Thomas E. Workman Jr.,²

Introduction:

The objective of this paper is to provide a primer on challenging the breath test software which controls every evidentiary breath test machine, and to provide to the DUI-DWI defense attorney a resource and plan as to what to do when the breath test employee appears at your hearing, or provides you discovery that contains the “source code” for their machine.

Every modern instrument designed to test for alcohol in the breath of a human subject is controlled by a computer, and every computer operates under the control of software. Software is represented by the source code, the set of instructions and procedures, designed to be interpreted by the computer inside the breath test machine, that implement the science as set forth in the specifications. Source code is translated into machine code using specialized computer programs called compilers, assemblers and loaders, to produce machine code that is typically delivered in a specialized memory device referred to as an EPROM.

All non-trivial software has defects. When the computer executes those portions of the software that are defective, the machine that is controlled by the software often malfunctions. Those malfunctions, or “faults”, sometimes produce failures. In breath testing machines, those failures can cause unexpected results: high BAC readings, unexplained readings, sample volume irregularities, and false reports that the defendant refused, because they failed to provide an adequate sample of their breath.

Unlike hardware components that ultimately wear out and fail, a software defect is always present, in every machine that contains the faulty software. Software failures are distinguished from hardware failures in that every machine that contains the faulty

¹ The author thanks attorney William Head for his inspiration to write this article, his guidance in the information that is presented, and his assistance in editing the drafts of the article. This article would not have been written but for his inspiration and guidance.

² Thomas E. Workman Jr. is an attorney with a criminal defense practice in Taunton, Massachusetts. He has a BS and an MS in electrical engineering from the University of Texas at Austin (1970 and 1974) and a JD from Suffolk Law School with a concentration in High Technology Law (1997). He is also admitted to practice before the US Patent Office and the First Circuit. He operates a computer forensic consulting business, with a website at: <http://www.computers-forensic-expert.com>. He can be reached in his office at 41 Harrison Street, Taunton, Mass 02780, by phone at (508) 822-7777, or email at Thomas.e.workman@verizon.net. He has testified in Federal and State Courts on computer forensic issues relating to criminal matters, including first degree murder trials, and has testified on source code issues relating to the Intoxilyzer 5000.

software is defective per se. Since a basic premise is that the machines are properly designed, there exists a presumption that if two machines produce the same results, the results must be correct, when in fact a software defect may produce identical, but wrong, results in every machine that is used.

An undetected software defect in a family of breath testing machines could send thousands of innocent citizens to prison for a crime that they did not commit. After all, the per se alcohol charge against a citizen requires no proof of alcohol impairment.

Primer for Attorneys

Attorneys need to have an understanding of fundamental concepts, so that they know when and how to challenge the results of the government's breath measurement machines. The vocabulary I will use includes:

Breath Testing Machine. An electro-mechanical device, usually with a model number and a serial number, to include all of the physical items required to operate the machine (e.g. disposable mouthpiece).

Breath Testing Environment. The environmental parameters relate to the location of and the place of administration of the Breath Test. These environmental parameters include the temperature, humidity, electronic interference, electrical power abnormalities, and atmospheric characteristics of the physical space where a Breath Test is administered.

Breath Testing Process. The process is defined by the training and regulations that define how the subject is instructed and monitored, how the Breath Testing Machine is operated and verified, and how the Breath Testing Environment is monitored and verified.

All Breath Testing Machines share significant common mechanisms. They are all:

1. *Electro-mechanical.* They operate with electricity, and they have moving parts to accommodate the detection of a breath sample delivered under pressure.
2. *Sensor dependent.* They depend either on an electrical detection of a chemical reaction between a human breath and chemicals in the machine, or they employ a semiconductor device that measures the intensity of light that has passed through a breath sample.
3. *Results only reporting.* They report a minimum amount of information so that the legal issue of intoxication may be inferred, but do not log all of the intermediate calculations that would be necessary to detect a malfunction with the device.
4. *Self calibrating.* They employ a limited calibration function that will detect some component failures, for a very limited stimulus that does not remotely resemble a human breath sample.
5. *Controlled by Software.* The measurement of alcohol levels is implemented through a computer program, as is the administration of testing steps, calibration, and the generation of results reports.

The electronic control portion of a Breath Testing Machine consists of several categories:

1. **Hardware.** The power supply, display and printer circuitry, memory, electrical circuits to include the computer chip.
2. **Firmware.** Software routines stored in read-only memory (such as a **ROM**³, an **EPROM**⁴ or an **EEPROM**⁵).⁶
3. **Software.** Computer programs; instructions that make hardware work. Two main types of software are **systems software** (operating systems), which controls the workings of the computer, and **applications**. Two additional categories are **network software**, which enables groups of computers to communicate, and **language software**, which provides programmers with the tools they need to write programs.⁷

A closer examination of the different categories of software is necessary to focus our attention on the software that is most relevant to the task of determining whether the Breath Test Machine employed for your client produced accurate and correct results.

These categories are:

1. **Systems Software.** Every computer has some software that manages how the computer chip is operated, how peripheral devices like printers are operated, and which provide common and basic functionality so that applications do not have to “reinvent the wheel”. Systems software is usually purchased for a given processor design, and is usually more robust in terms of reliability as contrasted to application software.
2. **Language Software.** Programmers depend upon application programs to edit their software, to convert it from a human-readable language to machine language, to bind together all of the individual pieces of software into one block of code that makes up all of the software that controls the on-board computer, to manage the known problems that exist in each version, and to test the various parts of the software. These support applications tend to be reliable and well tested, but do contain defects. Knowing what particular compilers, and the versions of the compilers, were used to translate the program source into machine language is important to obtain if the breath test manufacturer is less than forthcoming (so that your expert can insure that he has all of the source code, by recreating a “build” of the machine code from all of the software source disclosed).

³ A ROM is a Read Only Memory, a circuit that is custom designed and cannot ever be changed without a re-design of the semiconductor device. These are the least expensive to manufacture, but have a high initial setup design cost.

⁴ An EPROM is an “Electrically Programmable Read Only Memory”, and the contents can be erased by exposing the device to an ultraviolet light for a short duration of several minutes, and then the blank device can be loaded with any data using a programming device, a piece of equipment that is used to load machine language instructions onto the EPROM.

⁵ An EEPROM is an “Electrically Erasable Programmable Read Only Memory”, and is distinguished from an EPROM in that the circuit can be erased with an electronic signal. A machine with an EEPROM in it could be changed by the manufacturer, if they had remote access to the machine over a network, and if the equipment was designed to permit a remote user to change the device. Traditional computer users perform this step when they “flash the BIOS” of their computer or of a peripheral device.

⁶ Microsoft Press Computer Dictionary, Third Edition, page 197.

⁷ Microsoft Press Computer Dictionary, Third Edition, page 441.

3. **Network Software.** If the model is equipped with communication hardware, such as a modem or network interface card, then the machine has the ability to interface with a computer that is operated by either your state's laboratory, or with the manufacturer's computers (for the purpose of updating, or possibly disabling, your application software).
4. **Application Software.** Customized programs provide the personality for the specific machine, and usually are written so that they only operate on a particular manufacturer's machine, often only on a particular model of a machine. Because most states have statutes or regulations that are different, the software in a Texas machine is not identical to the software in a Florida machine, even if the brand and model numbers are the same.

The Application Software is what is most often requested by Defense Counsel (though it may not be set out with specificity). This Application Software is most likely to contain the significant defects that would cause failures of the Breath Testing Device. Not coincidentally, this software is the type most vigorously defended by the Government and the manufacturer, who truly fear that a careful examination will reveal major defects. Such a disclosure could invalidate the design of the machine, and call into question the results of thousands of decided cases.

When requesting the source code, one must request the source code for a specific model of the manufacturer's machine, as it is deployed in a specific state, at a specific date in time. The source code is different depending on the serial number, and certainly it will vary as a function of time (the software is changed from time to time, usually during a maintenance cycle).

What Every DUI Attorney Needs to Know About Application Software

Attorneys need to understand enough about how software operates and is constructed, and in particular the application software for their jurisdiction's breath test machine, so that they can appreciate and communicate how a given machine may produce unreliable results.

1. **Source Code Ambiguities.** When Source Code is converted to Machine Code, any ambiguities are either resolved (sometimes with "Warnings") or the Compiler that performs this translation documents "Fatal Errors" and refuses to produce the Machine Code. Unlike natural languages, like English, computer languages operate on rigid syntax rules, and unlike spoken languages, they do not permit an interpretation based on the context of a communication. Unlike the law, where ambiguities ALWAYS exist which are resolved at a trial or an appeal, when software ambiguities exist, the machine often malfunctions! With software, each program step is rigidly interpreted, according to a strict set of syntactical rules, which sometimes create unintended results.
2. **Specialization and multiple handoffs.** The programmer who writes the source code is usually not the scientist upon whose science the machine is based, and is rarely the person who designed the hardware. Rarely does one person have all of the skillsets required to design, build, and program a machine. The

programmer works from a written specification, and often a “Systems Analyst” is employed to work with the author of the specification (the scientists and hardware designers) to write a detailed specification that is similar in concept to a blueprint for a building. With each handoff, opportunity for misunderstanding and mistakes in the final product, the source code, increases thereby degrading the quality of the product.

3. **Size of the Team.** As programs become larger, they tend to be managed by multiple programmers, and just as a legal pleading that is constructed by multiple attorneys has a different degree of difficulty to manage, so to does larger software have its own peculiar problems. The more people involved, the greater the opportunity for miscommunication, increased complexity, and mistakes.
4. **Revision Management.** Each unique combination of software that is released to a customer in a manufactured or upgraded machine is a “software release”. There exist accepted procedures to define what is incorporated in a release, how it is tested, and how it is managed with respect to installation in the field. Some companies, like CMI, have a history of sloppy management of releases.
5. **Common Errors and Code Reviews.** Just as proofreaders have developed lists of common errors in written works, so too have software quality engineers compiled lists of common errors that programmers often make while writing source code. A computer scientist, preferably with a background in verifying software quality, can frequently find defects in software by “reading” the source code. In addition, source code may be reviewed automatically⁸ through automated review of the source code. Whether automated or performed by a team of people who read the source code listings, this process is referred to as a “code review. When a DUI-DWI defense attorney requests the source code for a Breath Testing Machine, it is with an aim to conduct a “code review” to look for defects.
6. **Good Programmers** always leave hints, or comments, in their source code. Programmers have to come back to their work months or years later, and if they are experienced, they add documentation in their source code, referred to as “comments”. These comments are ignored by the compilers that convert the source code to machine code. They provide context and logic as to why and how the source code is written. Comments often pose suspicions about problems that have been elusive and have not been specified and may not have been corrected. They may also contradict the instructions to the computer, as represented in the source code, meaning that either the comments are wrong, or the software is not correctly written.
7. **Programmers** often record information that will be helpful in resolving ongoing errors and defects. An examination of the source code will usually reveal extra steps that are not necessary to the computation of the results, but which will record information that relates to actual or potential error

⁸ One listing of such resources can be found at: <http://www.programurl.com/software/automated-code-reviews.htm>

conditions. The mere existence of this kind of activity suggests that the programmers are trying to collect additional information in order to resolve problems they have seen, but have been unable to isolate and fix. Another alternative is that the programmers have observed suspicious performance of the equipment during quality testing.

8. A Defective Breath-Testing Environment means all bets are off. The computer processor in the Breath Testing Machine will not operate correctly, and the software will produce unexpected results, if the environment is not controlled within the specifications. The “environment” includes temperature, humidity, contaminants in the air, radio frequency interference, electro magnetic interference, and “dirty” electrical power supply sources. Some deviations in the environmental specifications may damage the hardware so that it can no longer properly execute the instructions set forth in the source code, thereby creating permanent malfunctions which are not corrected when the environment is restored. Lightning strikes are one example of a phenomenon that delivers an unexpected electrical power surge that damages the internal electrical circuits, and probably causes a malfunction of the software that is operating at the time of the power surge caused by the lightning strike.

How Breath-Test Software Fails

Software is written by humans, and humans are not perfect. All Software, of any significance, has some density of “defects”. A defect is a condition that will result in a fault, if the state of the machine during a breath test presents conditions that cause the defect to cause an incorrect result.

Sometimes a defect will lie silent for years, waiting for some event to occur that causes the defect to cause faults. The Arizona Intoxilyzer failed on January 1, 2006 because of a defect in this category.⁹ Many machines failed to change their time by an hour when daylight savings time arrived in March of 2007, resulting in printed evidence reports that were “off” by an hour.¹⁰

Other defects may present themselves when an interferent is present, when the breath test machine’s pressure sensor detects parameters that are out of limits, or when some calculated result is deemed to be outside of accepted limits. When the state of the machine is such that a mistake in the source code is encountered, and acted on, it is referred to as a “bug” in the software, and an unexpected result, or a “fault” occurs.

⁹ CMI made changes to the Arizona machine so that an age limitation of over 100 years was not allowed. The accomplished this by processing the two digit year, and on January 1, 2006, the machine would not process any samples, because every sample appeared to be of a person who was older than 100 years old. The source code was defective in its implementation, and the defect never presented itself until January 1, 2006, at which time every Arizona machine malfunctioned for every sample.

¹⁰ In New Jersey, police officers were instructed to wait an additional hour before making their 20 minute observation, so that the time printed would always appear to be after the time of arrest. Some attorneys mused that on a leap year, defendants might have to wait a whole extra day.

Faults may create results that are incorrect, perhaps logged or communicated to the operator, or they may be “handled” by exception routines in the software that are designed to “deal with” the faults, by taking some predetermined action to block the fault¹¹. Fault-tolerant equipment is often designed to produce correct results in spite of faults, by making alternative calculations or by ignoring unreasonable data. If the computer is aware that a fault has occurred, it is common to log that fault so that a technician can later diagnose and fix the problem.¹²

While some faults may create insignificant mistakes in the results that are reported, when the results are deemed to be “incorrect”, a failure has occurred. For the breath test machine, **notable categories of errors** are:

1. ***Whether a test was attempted.*** The proper design of the source code should make it impossible to “cancel” a test once it is initiated, and like citation booklets, each test should produce, record, and log a unique serial number that would facilitate the detection of tampering by those who are responsible for maintaining the integrity of the breath test process. Law enforcement should not be able to claim that no test was performed, just because they do not like the answer. In many jurisdictions, the claim by law enforcement that the subject refused to provide a sample carries the significant penalty of loss of an operator’s license, and in some jurisdictions a refusal is itself a crime that is punishable by a full year in prison¹³. The software should ensure that every test that is initiated is recorded and reported, the failure to do this correctly is a failure of the machine. An examination of the source code may be the only way to determine under what conditions a test can be aborted and not reported.
2. ***Whether an adequate breath sample was provided.*** A cooperating subject may provide a full breath sample, and the machine may improperly label it an “insufficient sample”, or the machine may provide a reading when no sample or an insufficient sample is provided.
3. ***Whether the correct portion of a breath was analyzed.*** The source code reveals exactly how the machine attempts to exclude portions of the breath sample which do not represent alveolar air. Failures to correctly determine which part of the breath sample should be tested, and which part should be excluded, go directly to the integrity of the measurement.

¹¹ Most computer systems will interrupt the flow of the software when known error conditions are detected, e.g. dividing by zero or attempting to access a memory location that does not exist. These error situations can be dealt with by ignoring the error, or by taking some predetermined action. By definition, there is no way to present a “correct” result. These errors indicate that the results from this execution of the software are invalid, yet many systems will deliver some answer “in spite of” knowing that the calculations are defective per se.

¹² Automobiles and copy machines routinely log error codes, those cryptic letters and numbers that direct the repair person to the correct area of the machine for repairs. Some intelligent copy machines will even dial the repair technician and communicate the logged error codes, so that the technician can show up with the correct parts and tools to effect repairs.

¹³ For a second refusal, the state of Florida can incarcerate a citizen for up to a year, a draconian penalty if the citizen actually did provide an adequate sample, and the machine incorrectly reported that the citizen had not. FSA sec 316.1939.

4. ***Whether interferences are correctly excluded.*** A human breath may contain complex hydrocarbons other than alcohol, and the manner in which the machine, and the software that may perform the calculations, excludes these compounds from the reported results is crucial. The science of accounting for many interferences requires that logical readings be subtracted in order to eliminate over-reporting. In the event that these functions are implemented in source code, and are not being properly implemented, the incorrect calculations may inadvertently add the presence of these chemicals to the calculated numerical results, producing incorrect and unintended results.
5. ***Whether results are under-reported.*** The software may encounter errors which cause the machine to under-report results. This would cause the breath test machine to report results that are less than the correct results for a given breath sample. These results are sometimes not disclosed to the DUI-DWI defense attorney, or when they are disclosed, traditional impaired driving charges are often brought based upon the observations of the officer and based on field sobriety tests.
6. ***Whether results are over-reported.*** The software may encounter situations that cause the machine to report results that inflate the results. When the inflated results exceed legal *per se* thresholds, the error may result in a criminal prosecution where none should be brought.
7. ***Whether the reported results are accurate enough to be legally significant.*** Most manufacturers report that their machines have an accuracy rating of .002 or smaller during non-human simulator testing. Yet it is common for state regulations to routinely accept two results within .02 of each other (+/-) as being acceptable to indicate an “accurate” result. The manner of calculations, how the variables that make up the calculations are captured and stored, and the techniques employed by the programmers in writing the source code may explain the inaccuracies. A machine capable of measuring a sample within .002 should never return two calculations that are .02 apart, unless the human sample varies by at least .016¹⁴, or unless there exist mistakes in the source code which permit the results to be considered accurate when in fact they are not.
8. ***Whether the wet bath or dry gas simulation cycle fairly exercises the machine so that it presents a meaningful result.*** Many of the software features and the hardware sensors deal with the intricacies of processing a human breath, and are not exercised by a sample of air (not carbon dioxide) that contains a uniform portion of alcohol (or a gas from a cylinder) at a known concentration. An examination of the source code would reveal what portion of the software is never validated by the current simulation scheme for calibration and diagnostics.

¹⁴ If the higher reading sample resulted in a calculation that was .002 too low, and the lower sample resulted in an error that was .002 too high, then a .02 variation could be the result of a .016 difference that was understated by the sum of the two error amounts, or .004 – a .02 difference could also be stated on a sample that is in fact .024 different, if the .002 accuracy errors were both in the “unfavorable” direction.

Why Don't We Test for All Failures, and Fix Everything

The short answer is that this is not possible.

As to testing everything, to accomplish this, one must exercise every path in the software, where a path is a sequence of instructions that are used to instruct the processor.

By way of a simple example, suppose that a 1,000 line program contains an instruction that can alter the flow of what is next (a control statement, in the computer science language) as every 20th instruction. This would result in 50 control instructions¹⁵, which in their simplest form would permit two options for each control statement. For every control statement added, we double the number of paths possible. We express the number of paths as two raised to the nth power. For a program with 50 control points, the number of combinations is 2⁵⁰, which is 1,125,899,906,842,620 unique paths. If we assume that we require one minute to initiate each unique combination and to check the results of each unique path through the software, then we need over 2 billion years¹⁶ for one person to complete the testing, or if we could enlist every person in the United States to work on this task, we could complete it in a little over seven years of continuous work, 24 hours a day, seven days a week.

In the software industry, it is not possible for a programmer to perform perfectly when correcting a defect. The rate at which new problems are introduced is referred to as the **error re-insertion ratio or rate**, and is typically between 15% and 50%. That is, if you correct a problem, the probability of introducing at least one new problem is between 15 and 50 percent¹⁷. Adding changes associated with changes in scope, or “enhancements”, often insert defects at a higher rate.¹⁸ If the source code is poorly written, or does not contain adequate comments so that a person modifying the source code has good documentation of how the program functions, or good written specifications do not exist, in sufficient detail to communicate the details of what the software must do, then the insertion rate for new problems is higher. A finite risk of making things worse exists every time the software engineers correct the software or make changes, and for any significant program, there comes a time when the program cannot be improved with maintenance, because new problems are introduced faster than they can be corrected.

¹⁵ If every 20th instruction is a control statement, and there are 1,000 statements total, then there are 1,000 divided by 20 control statements, or 50 control statements.

¹⁶ The number is computed by dividing 2⁵⁰ by 60 to compute the number of hours; divide that by 24 to get days; divide that by 365 to get years – the result is 2,142,123,110 years.

¹⁷ Adams, Edward N., “Optimizing Preventing Service of Software Products”, IBM Journal of Research and Development, Vol 28, No 1, pp 2-14, January 1984

¹⁸ One study reported that 0.628 defects were introduced for each change and enhancement made. Kan, Stephen H, *Metrics and Models in Software Quality Engineering*, Addison-Wesley Publishing, Reading, Massachusetts, 1995.

All Breath Testing Machines Are Defectively Designed ON PURPOSE

Every Breath Testing Machine in widespread use is designed so that the biological sample is discarded after processing¹⁹, and so that all intermediate data concerning the measurements made by the machine, are also discarded. It is possible that intermediate calculations are saved by the software, for some period of time, but this capability would likely only be discovered through an examination of the source code.

The decision to discard the breath sample is a “design choice” made by the various state agencies that define how the equipment is to be deployed (some breath test machines have a discharge connection which could be used to capture the breath sample, but few states have decided to “utilize” this feature²⁰). This design decision to discard what could be exculpatory evidence was implemented by the manufacturers of the breath testing machines, who respond to their customers. Customers of breath testing machines are “law enforcement”, and these “customers” have shown little or no interest in deploying the containers necessary collect and to preserve the breath samples. After all, the best that could come from such preservation is that the machine was functioning correctly.

Urine testing and blood testing result in the collection of a sample, and that sample is either retained, in whole (as in urine enzyme screens) or in part (as in retaining a portion of the drawn blood for later testing). Every form of forensic science has as a cornerstone the ability to repeat the test, given a sufficient supply of the sample (often the test requires a portion of the collected sample, and is destructive by its very nature). Breath testing stands alone in the forensic sciences, as the only forensic testing method that prohibits verification and validation by virtue of the design of the process.

Most Breath Testing Equipment is equipped with design features that interpret the various stages of a human breath (from the air in the mouth, to shallow lung air, to deep lung air). These different treatments are implemented through the software in the various machines. Not coincidentally, the wet bath simulators that “verify” the correct operation of the machines supply a stream of air with alcohol and any contaminants in a constant concentration, unlike the human breath. The use of these traditional simulators fails to exercise the critical logic employed by most machines in order to detect the different phases of the human breath. The only time that the sensors, and the use of the data recorded by the sensors, are used is during the test of a human subject’s breath. If the results are wrong, the subject may go to jail.

In addition to destroying the sample and all calculations used to compute the answer, the Intoxilyzer 5000 provides error feedback to the operator through tones that are generated

¹⁹ The manufacturers at best provide a plumbing connection through which the breath sample is discharged from the machine, and modifications to discharge a sample of the last air in the chamber through a custom discharge port. No facility to trap the sample is provided internal to the machine, nor are supplies and training provided by the manufacturers. Any solution is left to the “after market” suppliers of accessories for the various machines.

²⁰ New Hampshire and Colorado are believed to use the ToxTrap, a facility that captures a sample of the air that is present at the conclusion of a breath test. Oklahoma and Arizona have case law that refers to the ToxTrap, though the author believes that only New Hampshire and Colorado require the preservation of a sample using the ToxTrap silica gel breath capture product.

and intended to communicate problems. The tones are not self-documenting (who can remember what each of the tones means), and some of the tones are generated during the course of the normal operation of the machine, during the testing of a breath sample..

When law enforcement observes a test which they know to be defective, state regulations, and company policies of the manufacturers of Breath Testing Machines, provide no mechanism for recording or reporting the error to the manufacturer. Since no data is captured which would facilitate the detection of, and the correction of, error conditions, errors in the field will continue unless and until that same error is coincidentally observed by the manufacturer. The log of what tests are administered by the manufacturer, in order to certify that each model of their manufacture is working properly, has never been produced (to my knowledge).

All Breath Testing Machines Fail to Provide a Reporting Mechanism for Failures

Keep in mind that the manufacturers of these machines view law enforcement as their customers, and their customers do NOT want to provide any ammunition for diligent defense attorneys to use in establishing the innocence of their clients. If the machine produces a “number” over the per se limit, then most law enforcement officials are convinced that their suspicions concerning the sobriety of a citizen must be correct., and that the citizen is guilty.

It should not be surprising that many states elect not to log anything about the breath tests.²¹ Those that do log, log only the final results, and not all of the intermediate values that go into computing each discrete sample of breath. The software, as represented by the source code, selects which samples are to be used in computing the reported contents of alcohol. Select the wrong samples, and you may not have accurate results. Make adjustments to the alcohol value in an improper way, and the results are wrong.

The law enforcement community has agreed on some standardization concerning what information should be captured and reported. Software programs are available that collect the parameters of defined data fields from the machines within a particular state. These programs load the data to a centralized machine²². Not all states have implemented this kind of logging.

²¹ There are notable exceptions. At least Florida and South Carolina are believed to make available on the Internet data captured with every breath test administered, with the results available for statistical analysis.

²² CMI sells a program called COBRA which systematically contacts each machine within the jurisdiction, over a modem, and uploads the values stored concerning recent breath tests administered.

Why the Breath Test Manufacturers will Fight the Release of Source Code

Companies with significant market share have everything to lose if it becomes known that significant errors exist in the design of their machine²³. They may have actual knowledge of such errors, which they have not corrected in their installed base of machines, due to of extreme the expense of doing a “recall”. The political blow-back of a “recall” could be devastating to the reputation of the manufacturer of a breath test machine, and law enforcement officials want no part of such a process.

Despite claims to the contrary, source code has been provided to some of the state agencies. In Wisconsin, at least between 1986 and 1994, the source code for the Intoxilyzer was kept at the state laboratories, but was the subject of a contractual agreement under which the source code was only to be examined at the state facility, with no copies made.²⁴ Claims made by CMI, that the source code has never been supplied to any stated agencies, is simply false.

Because of the potential risks, most of the companies with market share will fight any requests to examine the source code. Even when a court orders the production of the source code, companies will refuse to produce the source code.²⁵ The companies that have written the software for the breath testing machines have kept the process a secret, and combining the secrecy of development with a feedback process that does not facilitate the reporting of errors, *the defect density is likely to be higher than industry norms*. The companies that make the devices recognize their vulnerability, and will do just about anything to prevent the defense bar from understanding the defects in their machines.

How the Breath Test Manufacturer will Fight the Release of Source Code

First of all, the state directors of the organizations that oversee the selection and calibration of the machines have been trained to report that they do not have any skills in evaluating the design and functionality of software, and that they do not need to see the software to know that the machine is working correctly. They are partly correct, in that they would not recognize software if it fell on their head, but they are wrong in contending that because they do not have the skills, that therefore access to the source code is meaningless. In large part, many of these titular heads of breath testing departments of state crime laboratories are incompetent, and the companies that make the machines have trained them on how they can keep their jobs, in spite of their incompetence.

²³ The notable exception appears to be National Patent, the manufacturer of the Datamaster machine. Reports indicate that with the proper subpoena, they will produce their source code.

²⁴ Source: Mary McMurray, ethosinc@aol.com. Disclosed in a telephone call with the author on March 27, 2007.

²⁵ In *New Jersey v Chun*, the New Jersey Supreme Court ordered the production of the source code so that it could be examined in the course of a hearing designed to evaluate the science of the Drager 7110. Drager refused to turn over the source code. In Florida, CMI has refused to provide the source code for the Intoxilyzer 5000.

Their “hide the ball” argument seems to say that if they are the experts for the state, and if they do not need the software, then the defense does not need the software. The idea that they are incompetent does not seem to enter the equation.

The manufacturers will make this argument, and many fair-minded judges will recognize that it is a false proposition. The harder the manufacturers fight to hide their source code, the more certain the tenacious members of the DUI-DWI defense bar confirm that the manufacturers of these machines have a **LOT** to hide. Then the legal wrangling ensues.

The state will usually report that they do not have the software or the “source code”, and cannot be forced to produce what they do not have. They conveniently omit the fact that if the state requested the software, they would get it.

The next line of defense for the manufacturers is that the software is part of the overall design, and is a **trade secret**. Trade secret issues are discussed infra.

The state will sometimes contend that copyright law prevents the copying and disclosure of source code, and sometimes they will assert that their manuals cannot be produced because they cannot waive the manufacturer’s copyright. In an earlier hearing in the 1980s, CMI contended that the extraction of the machine code from the EPROMs in an Intoxilyzer constituted a criminal violation of the copyright statutes. Defense attorneys should point out that this sounds like obstructing justice, or simply unethical conduct, for it is, in fact, both.

The federal copyright statutes contemplate occasions when copying is allowed. The relevant legal principle is referred to as “fair use”. Without fair use, bizarre scenarios arise which make no sense. For example, a prosecutor recites or quotes in a brief a portion of a work protected by copyright, and the court reporter places the words spoken in the record. Is the court reporter now a criminal for having placed the words down on paper? Is the prosecutor a criminal for having “performed” or “read” the work, even if the performance was from memory? *No* and *No*.

Patent law is often thrown out by the manufacturers and their as a barrier to production. The prosecutor will sometimes report that patent law is Federal law, and that this state court does not have the jurisdiction to decide a patent dispute. Carried to the extreme, the court would be unable to hear any case dealing with an automobile, since all automobiles have some patents that protect the car or components of the car.

Expect any subpoena of a manufacturer’s employee to bring the “source code” to Court to be the subject of a motion to quash. Be prepared to have to litigate in the manufacturer’s home state, which is generally a hostile environment. A federal case, for example of a serviceman charged with DUI-DWI on a military base, would eradicate this bogus defense for the manufacturers, as a federal subpoena would have to be quashed in a federal court, not the friendly state court venues that the companies have become accustomed to.

Lastly, the Prosecutor will claim that the software is not relevant to the operation of the machine. They will point out that the officers are not trained on the software, and that the officers do not manipulate the software, ergo it is irrelevant. The stupidity and arrogance of this circular logic needs no elucidation.

Estimating Defect Density in Source Code

Even without the source code, it is possible to make some statements about the *density of defects* in software. The *defect density* is a measure of the number of defective statements in a number of lines of source code. Usually the measure is stated as “Defects per Thousand Lines of Source Code”, sometimes abbreviated KLOC for thousand (“K”) Lines Of Code. Studies of defect density have confirmed that the defect density has improved as better languages and better techniques have been developed to author and examine software. On average, 25 software defects exist for every 1,000 lines of code.²⁶

Be careful if you introduce this concept, as a Judge may decide that you do not need the source code to establish your points. This “concession”, however, is not the answer to the quest for the truth about the machines. You may be able to establish that the machines are unreliable, without the source code, but you will not be able to establish defects that relate to your specific client.

If the number of lines of code in the source code can be ascertained, the industry averages can be applied to estimate the number of defects. The estimated number of defects is calculated by multiplying the number of lines of code by 25, and dividing that product by 1,000.

The number of lines in the source code has been disclosed in testimony for the Drager 7110 device, which has 53,774 lines of code that print out on 896 pages.²⁷ Applying the formula that utilizes the industry average, it is reasonable to expect 1,344 defects in the software for the Drager 7110, if it conforms to the industry norms and is “average”.

For the Intoxilyzer 5000 EN, three memory chips hold the machine language for the machine. The first two 8K chips are full, and the second chip is nominally half full, for the purposes of estimating the number of machine instructions. The number of bytes of information is thus $8,192 + 8,192 + 8,192/2$, or 20,480 bytes. Assuming that the instruction set in those memory chips require two bytes for the average instruction, then the source code consists of approximately 10,000 instructions. Assuming further that the source is written in assembly language, where one line of code exists for each instruction, then the Intoxilyzer 5000 EN source code contains approximately 250 defects.

The Drager machine is likely the most complex, and likely has the largest number of lines of source code; and the Intoxilyzer 5000 likely has the smallest number of lines of source code, based on the age of the machine and the size and number of chips dedicated to storing the programs that operate the machine.

What you need to know about Patent law

United States Patent law provides the public with rights, with respect to inventions which were once covered by a patent, or which now are covered by a valid patent. While the rights of the Patent holder, against all infringers, is enforced in Federal Court. The rights

²⁶ The Science of Debugging, Telles and Hseih, Coriolis Technology Press, 2001, page 57

²⁷ Direct examination, under oath, of STEPHEN B. SEIDMAN, page 6, lines 4-6 of volume 16T, Supreme Court of New Jersey, docket 58,879, New Jersey v. Jane H. Chun et al., October 3, 2006 Hearing.

of the Public do not need to be enforced in a Federal court, as the patent holder has obligations as to disclosure which if not discharged, can result in criminal charges and fines of up to a million dollars, these remedies would be sought in Federal Court, in the event of non-compliance by the patent holder.

Patent law exists to advance the collective body of knowledge. Patent holders are given a monopoly in their invention for a period of time, after which the invention becomes part of the body of knowledge available to all.²⁸ Since Congress has this power, the states do not – Patent protection can only be granted through federal statutes.

Patent law exists to advance science, not to help a company make more money. The underpinnings of this are articulated in Article I, Section 8, paragraph 8 of the United States Constitution.²⁹ Once the limited time of protection expires, the body of scientific knowledge increases, with the invention now available for anyone to make and practice. Companies like Xerox at one time were the only producers of machines, but once the patents became property of the public, many competitors entered the business. This is exactly as the Patent process anticipated.

When a Patent is granted, one fundamental requirement of patent law is that the applicant must disclose everything they know about the “best mode” of practicing the invention.³⁰ An applicant is expressly prohibited from holding back some information, so that when the patent expires they are still the only person able to make and practice the invention. Maintaining a trade secret in some portions of the invention is not permitted.

Breath test companies will often claim that they did not patent the software, just the hardware. Such a claim must either admit that the software contains nothing of significance (in which case there is nothing to protect), or that it does contain significant information, in which case they were obligated to disclose it. The “Claims” of a patent spell out what specifically is new, and what is therefore protected. Whether the vendor included claims relating to the software is irrelevant as to the obligation to disclose the best way to practice the invention. The bright line test: If the machine would work without the software, then the software is not part of the invention. The machine will not function without the software.

Failure to disclose the best mode has severe consequences. A company that intentionally withholds information, while enjoying the protection of a patent, is practicing a monopoly without the protection of a Patent as a license to practice that monopoly. Such a company would be a common monopolist, subject to criminal and civil penalties, under the Sherman antitrust statutes, section 2, as well as state statutes that prohibit monopoly practices. Every patent application is accompanied by a Declaration, signed under oath

²⁸ The United States Constitution contains the enabling language for patent protection, in article I, Section 8, Paragraph 8, stating: “The Congress Shall have the power ... To promote the progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries”.

²⁹ Id.

³⁰ 35 USC 112 first paragraph states “The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same, and shall set forth the best mode contemplated by the inventor of carrying out his invention.”

by both the patent attorney and all of the inventors, which acknowledges that all understand their obligation to comply with the Patent statutes, and they must explicitly acknowledge that the failure to comply could result in incarceration of up to ten years in a federal prison.

Most manufacturers, such as CMI and Drager, now have or did have patent protection for their machines. These are fairly easy to research, one way is with Google, at: <http://www.google.com/patents>, with links to the US Patent office. The CMI patents are 4,268,751 (expired) and 4,587,427³¹ (expired). Drager has a number of US patents as well. The software, as it existed at the time the machine was patented, must either be insignificant or the companies have an obligation to disclose it.

Prosecutors will try to convince a Judge that the court cannot litigate the validity of the patent, and while that is true, that has nothing to do with the affirmative obligations that companies have to disclose the science behind their once patented inventions. They cannot legally claim that any part of the invention, as now or once patented, is a trade secret, or protected by copyright.

What you need to know about Copyright law

At one time, copyright law existed in both state and federal forums. A 1978 federal statute provided preemption of state rights³², eliminating state copyright protection for subject matter which qualified a work for federal copyright protection.

In addition, modern law has eliminated the requirement that a copyright statement appear on all distributed copies³³. Works created prior to March 1, 1989, e.g. the software for

³¹ Note the drawings of the invention in Figures 1a and 1b at the Patent Office, you will see a drawing of the model 5000 machine as part of the patent.

³² 17 USC 301(a) preempted state copyright protection from January 1, 1978 onwards, and stated: “(a) On and after January 1, 1978, all legal or equitable rights that are equivalent to any of the exclusive rights within the general scope of copyright as specified by [section 106](#) in works of authorship that are fixed in a tangible medium of expression and come within the subject matter of copyright as specified by [sections 102](#) and [103](#), whether created before or after that date and whether published or unpublished, are governed exclusively by this title. Thereafter, no person is entitled to any such right or equivalent right in any such work under the common law or statutes of any State.”

³³ (a) Effect of Omission on Copyright. -- With respect to copies ... publicly distributed by authority of the copyright owner before the effective date of the Berne Convention Implementation Act of 1988, the omission of the copyright notice described in sections 401 through 403 from copies ... publicly distributed by authority of the copyright owner does not invalidate the copyright in a work if –

(1) the notice has been omitted from no more than a relatively small number of copies ... distributed to the public; or

(2) registration for the work has been made before or is made within five years after the publication without notice, and a reasonable effort is made to add notice to all copies ... that are distributed to the public in the United States after the omission has been discovered; or

(3) the notice has been omitted in violation of an express requirement in writing that, as a condition of the copyright owner's authorization of the public distribution of copies or failure to affix proper notice to certain copies of a work will not result in forfeiture of copyright protection if, within five years after the first publication of a copy without notice, the work is registered at the Copyright Office and a "reasonable

the Intoxilyzer 5000, would be in the public domain and would not be afforded copyright protection unless the formalities of the law, e.g. registration, were followed, and the required copyright notice was affixed to virtually all of the works released.

Software delivered in semiconductor chips presents two interesting sections of copyright law, namely the limitation on exclusive rights for software, which permits copying for archival, maintenance, and repair purposes³⁴ and that portion of the copyright statutes designed to protect semiconductor chip products, which likely do not apply to the EPROMs used to distribute the software that makes the machines operate, but WOULD apply to any ROMs delivered, since the ROMs would have a unique manufacturing process that would produce parts that could not be written more than once.³⁵

State employees have been trained by the manufacturers of the machines that test breath to refuse to turn over manuals and any materials they possess, under the guise that these materials are copyright protected. Aside from the exception that exists for court proceedings³⁶, it is well accepted that it is permissible to copy works for the purposes of “criticism”, “comment” and for “research”.³⁷ The copies of documentation, and of source code, are being made for these purposes. If a person were to obtain copies for judicial purposes, and then go into business selling copies they obtained, then the purpose of copying would not be protected under the fair use doctrine.

The refusal to sell documents to defense attorneys can be addressed by requiring the government’s witness to provide their copy of the documents, and order replacements for themselves from the manufacturer. Such is one remedy, if the government’s employees will not make the copies.

Lastly, federal government documents are exempt from copyright, and never enjoy copyright protection.³⁸

effort" is made by the author to add notice to all copies distributed to the public "after the omission has been discovered." 17 U.S.C. 405(a)(2);

³⁴ 17 USC § 117. Limitations on exclusive rights: Computer programs (a) MAKING OF ADDITIONAL COPY OR ADAPTATION BY OWNER OF COPY. — Notwithstanding the provisions of [section 106](#), it is not an infringement for the owner of a copy of a computer program to make or authorize the making of another copy or adaptation of that computer program provided...” See (a) for archival permission, (c) for maintenance and repair, (d) for definitions.

³⁵ 17 USC §§ 901-914, referred to as the “Protection of Semiconductor Chip Products”.

³⁶ Can you imagine what would happen if a prosecutor spoke text from a work protected by copyright, and the Court Reporter “copied” those words into a written record and a Judge “copied” those words into his decision? Without an exception for judicial matters, the Prosecutor, Court Reporter and Judge would all be “Guilty” of copyright infringement.

³⁷ 17 USC § 107. “Limitations on exclusive rights: Fair use. Notwithstanding the provisions of [sections 106](#) and [106A](#), **the fair use of a copyrighted work, including such use by reproduction in copies** or phonorecords or by any other means specified by that section, **for purposes such as criticism, comment**, news reporting, teaching (including multiple copies for classroom use), scholarship, **or research, is not an infringement of copyright.**” (emphasis added).

³⁸ 17 USC § 105. “Subject matter of copyright: United States Government works. Copyright protection under this title is not available for any work of the United States Government, but the United States Government is not precluded from receiving and holding copyrights transferred to it by assignment, bequest, or otherwise.”

What you need to know about Trade Secret law

The prosecutor may tell you that you cannot see the source code because it is a trade secret. The government will have no problem using the software to calculate the amount of alcohol in your client's breath and/or blood, they just will not let you know how that quantity was actually calculated.

The defense generally begins by claiming that the government is not in possession of the source code, and you cannot have what they do not possess. In most states, the prosecution is obligated to use good faith efforts to obtain evidence that is theirs for the asking – police reports and investigatory reports are in this category. Source code may fall outside of this obligation, but you can be sure if it was helpful to the government, a phone call would be all that was needed.

Unlike Patent and Copyright law, Trade Secret is state law, not federal law. Each state has their own statutes concerning the protection of trade secrets. Forty two states³⁹ have laws modeled after the Uniform Trade Secrets Act⁴⁰, seven states protect trade secrets under common law⁴¹, and two states⁴² have statutes that are not common law nor based on the Uniform Trade Secrets Act.

The Uniform act contemplates a process for dealing with judicial action that involves alleged trade secrets⁴³, and that mechanism is a protective order.

³⁹ These 42 states are: Alaska 9/2/88 Alaska Statutes § 45.50.910 et seq., Arizona 4/11/90 Arizona R.S. § 44-401 et seq., Arkansas 3/21/81 Arkansas C.A. § 4-75-601 et seq., California 1/1/85 Cal. Civ. Code § 3426 et seq., Colorado 7/1/86 Colo. R.S. § 7-74-101 et seq., Connecticut 6/23/83 Conn. G.S. § 35-50 et seq. (Amended 1997), Delaware 4/15/82 6 Del. C. § 2001 et seq. (Amended 1997), District of Columbia 3/16/89 D.C. Code § 48-501 et seq., Florida 10/1/88 Fla. S. § 688.001 et seq., Georgia 7/1/90 Ga. C.A. § 10-1-760 et seq., Hawaii 7/1/89 Haw. R.S. § 482B-1 et seq., Idaho 1981 Idaho C. § 48-801 et seq., Illinois 1/1/88 765 ILCS § 1065/1 et seq., Indiana 2/25/82 Ind. C. § 24-2-3-1 et seq., Iowa 1990 90 Acts, ch 1201 Section 550.1 et seq., Kansas 7/1/81 K.S.A. § 60-3320 et seq., Kentucky 4/6/90 Ky. R.S. § 365.880 et seq., Louisiana 7/19/81 La. R.S.A. § 51:1431 et seq., Maine 5/22/87 Me. R.S.A. § 1541 et seq., Maryland 7/1/89 Md. Comm.L., § 11-1201 et seq., Michigan 12/30/98, MCL 445.1901 et seq., MSA Sections 19.901 et seq., Minnesota 1/1/81 M.S.A. § 325C.01 et seq., Missouri 8/28/95 Mo. Rev. Stat. Sec. 417.450 et seq., Mississippi 7/1/90 M.C.A. § 75-26-1 et seq., Montana 1985 Mont. C.A. § 30-14-401 et seq., Nebraska 7/9/88 Neb. R.S. § 87-501 et seq., Nevada 3/5/87 Nev. R.S.A. § 600A.010 et seq., New Hampshire 1/1/90 N.H. R.S.A. § 350-B:1 et seq., New Mexico 4/3/89 N.M. S.A. § 57-3A-1 et seq., North Carolina 1981 N.C. G.S. § 66-152 et seq., North Dakota 7/1/83 N.D. Cent. C. § 47-25.1-01 et seq., Ohio 7/22/94 R.C. § 1333.61 et seq., Oklahoma 1/1/86 Okl. S.A. tit. 78 § 85 et seq., Oregon 1/1/86 OR. R.S. § 646.461 et seq., Rhode Island 7/1/86 R.I. G.L.A. § 6-41-1 et seq., South Carolina 6/15/92 S.C. C.A. § 39-8-1 et seq. (replaced 5/21/97 S.C. Code Ann. Section 39-8-10), South Dakota 7/1/88 S.D. Comp. L.A. § 37-29-1 et seq., Utah 5/1/89 Utah C.A. § 13-24-1 et seq., Vermont 7/1/96 Ch. 143 Section 4601 et seq., Virginia 7/1/86 Va. C.A. § 59.1-336 et seq., Washington 1/1/82 Wash. R.C.A. § 19.108.010 et seq., West Virginia 7/1/86 W. Va. C.A. § 47-22-1 et seq., Wisconsin 4/24/86 Wis. S. § 134.90

⁴⁰ Available on the internet at: <http://www.law.upenn.edu/bll/ulc/fnact99/1980s/utsa85.htm>

⁴¹ Missouri, New Jersey, New York, Pennsylvania, Tennessee, Texas and Wyoming

⁴² Massachusetts and Alabama, coincidentally two states that now use the Drager 7110 breath testing machine.

⁴³ “SECTION 5. PRESERVATION OF SECRECY. In an action under this [Act], a court shall preserve the secrecy of an alleged trade secret by reasonable means, which may include granting protective orders in connection with discovery proceedings, holding in-camera hearings, sealing the records of the

Trade secrets and patents are mutually exclusive remedies, and companies can elect to employ one or the other, but not both, to protect their intellectual property. The Prefatory Note to the Uniform Trade Secret Act, which set forth the context in which the Act is presented, starts with the following:

“A valid patent provides a legal monopoly for seventeen years in exchange for public disclosure of an invention. If, however, the courts ultimately decide that the Patent Office improperly issued a patent, an invention will have been disclosed to competitors with no corresponding benefit. In view of the substantial number of patents that are invalidated by the courts, many businesses now elect to protect commercially valuable information through reliance upon the state law of trade secret protection. *Kewanee Oil Co. v. Bicron Corp.*, 416 U.S. 470 (1974), which establishes that neither the Patent Clause of the United States Constitution nor the federal patent laws pre-empt state trade secret protection for patentable or unpatentable information, may well have increased the extent of this reliance.” UTSA, Prefatory Note, Paragraph 1, 1985.

Reverse Engineering is a protected method of discovering a trade secret.⁴⁴ In prior judicial proceedings, CMI has attempted to chill testimony of an expert for the Defense by insinuating that those who reverse engineer the software, by copying out the machine language instructions from the EPROM chips, have engaged in illegal actions based on copyright and trade secret violations. The defense should be ready to counter any such line with warnings that such prosecutorial conduct is “an obstruction of justice”, not grounded on the proper application of law, and is an attempt to chill protected research and reverse engineering of the machine, which are protected by federal and state laws.

What Should You Ask For

Your request should be made in multiple parts, at successive times.

The initial request should be a request to **disclose the computer language** that was used to write the source code that was written by or for the manufacturer’s machine, and **what microprocessor** was used on the machine. Usually, the entire collection of source code will be in a single language, although it is possible that several languages were employed. You should also ask **how many lines of code** there are in the source code for the application software that is in the subject machine, and you should also ask **what has changed in the software** in the past two years, expecting that the state will tell you that nothing has changed. This “first round” of information is not highly contentious, does

action, and ordering any person involved in the litigation not to disclose an alleged trade secret without prior court approval.”

⁴⁴ UTSA, 1985, Comment to Section 1 states: Proper means of discovery include: “Discovery by ‘reverse engineering’, that is, by starting with the known product and working backward to find the method by which it was developed. The acquisition of the known product must, of course, also be by a fair and honest means, such as purchase of the item on the open market for reverse engineering to be lawful...”

not disclose any “secrets”, and provides a foundation for what you would need to do if you got the source code.

With the state’s response to this request, you should now consult with an expert who can help you understand the cost and time required to analyze the source code, if it were produced. Some modern software languages, like C++, have programs designed to automatically detect common defects. Companies now exist that will produce reports that identify defects, based on common mistakes that programmers make⁴⁵.

An expert will want to be able to verify that he has all of the source code associated with a particular machine. To do that, your expert will need to process that source code with the same “Compiler”⁴⁶ or “Assembler”⁴⁷ used by the manufacturer. It is thus important to ask the manufacturer **what compiler or assembler they use to convert the source code into the machine code** delivered with their machine. It is important to ask for the name of the package, the manufacturer, the version designator, and **whether the company has applied the updates that are often delivered by the software compiler or assembler company**. Armed with this information, your expert has a chance to duplicate the environment at the breath test company’s software house, so that theoretically your expert can recreate the **machine code delivered in firmware** in your state’s machine.⁴⁸

Often times machines with embedded processors employ routines that are commercially written and available as “off the shelf” software that can be used to perform common functions. While you may be unable to get the source code for these, if they are employed, they are generally well documented, and if they are not being used correctly they will produce faults and failures. You must ask **whether the application software depends upon the presence of any commercially obtained routines**, and if so, what routines are used, **from what company were they obtained**, and **what version of the utilities was utilized with their source code**.

In addition to the “library” of software routines that may have been employed, sometimes application software to gather together and store the results of a breath test is present. That portion of COBRA on a CMI machine is required to deliver the results of a “saved”

⁴⁵ http://www.reasoning.com/pdf/Reasoning_Inspection_Services.pdf is one such service.

⁴⁶ A Compiler is a software program that accepts as input a set of source code written in a high level language for an application, and produces a set of machine code which, if placed onto an EPROM or set of EPROMs, will cause a computer to execute the instructions set forth in the source code. A high level language is characterized as a language that is capable of presenting commands in a form that is closer to the way that a person presents and thinks of the computation required.

⁴⁷ An Assembler is a software program that accepts as input a set of source code written in assembly language for an application, and produces a set of machine code which, if placed onto an EPROM or set of EPROMs, will cause a computer to execute the instructions set forth in the source code. Assembly language programs require the programmer to select the specific instructions that the computer will execute, at the most atomic level, and to specify the parameters utilized by that specific instruction. The programmer must translate the computations required into the individual machine language instructions, and hand code each of these instructions.

⁴⁸ If your expert compiles all of the source code and the result is 20,000 bytes of machine language, and the machine has 40,000 bytes of instruction in the firmware, then the company has not disclosed all of the source code. In addition, different compilers process the source code differently, and generate the machine language differently. There may be a history of defects in the compiler, which would produce defects in the firmware, *even if the source code was perfect*.

breath test, and the software on the central machine is important to discover, since it will often describe what significance the collected data has (e.g. the machine may supply a variable called “X213”, which the COBRA software might label as “TemperatureOfSample”; knowing how COBRA refers to the data is often very helpful in ascertaining the purpose of the information, and the operation of the machine).

The breath test machine may also contain “systems” software, in the form of an “operating system”. The Intoxilyzer 8000 is believed to utilize an obsolete operating system referred to as RTOS (**Real Time Operating System**). To analyze the source code, the contextual support, provided by the operating system deployed, is an important part of the process. Fortunately, most commercially delivered operating systems have extensive documentation. To access the appropriate documentation, the version of the operating system, and the **formal name of the manufacturer of the operating system must be requested**. This information must be requested from the breath test manufacturer.

The **source code should be requested in electronic format**, delivered on a CDROM or other digital media, or delivered as a text file attachment in an email message. It should be the same source code that would be used by the breath test machine manufacturer, should the manufacturer decide to change the source code. It should be requested in “clear text”, that is it should not be encrypted.⁴⁹

You may be asked to assent to a protective order, which sets conditions upon the use of the source code by your expert. The company will often include onerous terms that will make the analysis work impossible. Conditions like “The source code shall not be processed by any other program” would prevent any automated analysis of the source code, which would seriously hamper the analysis, increase the time required to perform a meaningful analysis, and dramatically increase the cost of an analysis to your client.

Because all breath testing machines used in the United States are or were protected by United States Patent law, a court should allow automated analysis under a protective order that is constructed to protect any reasonable interests of the manufacturer which have not been waived by virtue of their obligation to disclose under Patent law.

Conclusion

The source code is the enabling technology for all breath test machines in use in the United States. Every machine running the same software, as delivered in semiconductor EPROMs by the manufacturer, will faithfully execute any defective software in the same way on every machine in which the software is deployed.

⁴⁹ There exists a class of software tools called “Source Code Obfuscators”, whose sole purpose is to make it difficult to read the source code. Imagine trying to read a legal document in which all of the spaces were removed, so that all of the letters and words were present, but without any natural spaces to assist in processing the text. This is in essence what an obfuscator does, though some add statements or modify comments to mislead the reader, or add instructions that will not invalidate the computations, but which make it difficult to understand what the software is trying to accomplish.

Clients of the DUI-DWI defense bar deserve a defense that is zealous and which questions the science of the machines which often convict, and the questioning of the science requires a methodical and scientific examination of the instructions which, if wrong, will wrongfully convict citizens.

The author can assist the diligent DUI-DWI attorney in framing the requests for information, he will testify at hearings designed to obtain the source code, he can examine the source code produced, and he can finally testify providing an expert opinion concerning the manufacturer's source code, and thus the manufacturer's breath testing machine.