eSCOPE®
Electronic Lab Scope
Training

Diagnostic Trouble Code (DTC) by the WIRE

by Bernie Thompson

The triode was invented by Lee De Forest in the early1900s. The triode was built on vacuum tube technology and it was the first electrical amplifier, which gave us the ability to turn an electric circuit on or off by current flow. No longer was a mechanical switch needed in order to control an electric circuit, but voltage could now be used in the operational control of the circuit. This new control strategy would enable the start of modern electronics.

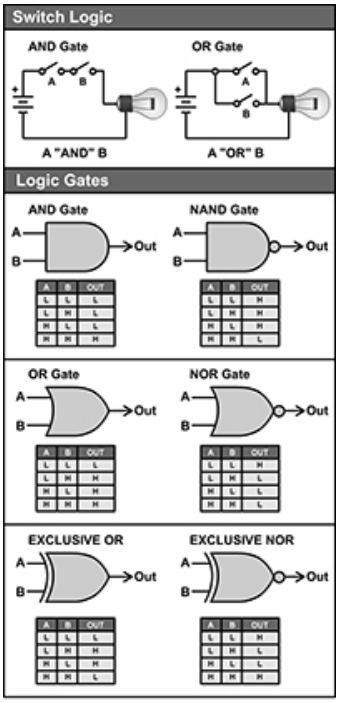


fig 1

The use of the triode to switch voltage from a low potential to that of a high potential was instrumental in the construction of the early electric telephones, radios, and calculators. In the early 1950s, a new technology based on semiconductor construction would replace the triode with the transistor in cases where low power amplification was needed. These transistors, when fully saturated, would enable a circuit to change states by turning it on or off, which then could be used to control an electric circuit or store information in an electric circuit. By allowing the on-off states of these transistors to represent a “low (0)” usually near zero voltage, and a “high (1)” at source voltage, information can be transmitted or stored in a circuit. This is accomplished by Boolean logic which is the logic of digital numbers “0” and “1”. The electric circuit can be set up so that one or more logic inputs can be processed to a single logic output. This electronic circuit is referred to as a logic gate (figure 1). There are several logic gate configurations such as; AND gate, NAND gate, OR gate, NOR gate, EXCLUSIVE OR gate, EXCLUSIVE NOR gate, and there are many other configurations of gates in use as well.

In order to process data, many logic circuits would be used in conjunction with one another. These “0s” and “1s” can set up a logical decision circuit that can convey information in a digital format. All modern microprocessors use digital logic circuits in order to process data. Each “0”or “1” is called a bit and it is the maximum amount of information that can be used or stored by a device that only has two possible states; off or on, “0”or “1”, no or yes, false or true. In computers, these two states are binary digits and they are designated as “0” and “1’. It maybe hard to understand how using just two states of operation could be used for the storage and transmission of information; however, you are already aware of such systems. One early use of digital binary bit encoding to send information in an electrical format was the telegraph which used Morse code. Morse code uses a binary bit of “dot” or “dash” to convey data. With just two states of operation [“dot”(0) or “dash”(1)], Morse code allows very complex information to be transmitted or stored.

In modern computers, the encoding scheme to program the microprocessor can vary with many different languages and they are seen by the programmer as the source codes. However when the source code is compiled to be used by the microprocessor, they are all based on “0” and “1”. In computers when four bit’s are combined to convey information; it is called a nibble, if eight bit’s are used it is called a byte. Computers use a binary code or base-2 system to convey information. The base-2 system makes for very long encoded messages or bit width so programmers came up with a way to shorten this encoding based on the hexadecimal system (figure 2). Instead of a count system based on 10 such as the decimal system, the hexadecimal system uses a count system of 16 to convey information. The hexadecimal system uses numbers 0-9 and letters A-F to encode information. The technician will see this hexadecimal encoding when reading mode 6 data from the engine control module.

Whichever method of encoding or language is chosen to operate the computer on, it is just a set of instructions that will be executed by the central processing unit (CPU). The CPU contains thousands and thousands of transistors and logic circuits that have become packaged in a very small design known as the integrated circuit (IC). This package of transistors allows for logical decision circuits to operate with the encoded messages known as the program. These logical decision circuits, when operating with the encoded messages, will have a predictable outcome. Since the outcome can be predicted, a logic path can be written to obtain a desired outcome. The CPU's task is to execute a sequence of stored instructions, layer by layer, as indicated by the program. The program will have the primary instruction set running layer by layer with many subroutines, loops, conditions, and exceptions running layer by layer at the same time. The CPU gets the instructions, decodes the instruction and carries out the instructions all based on the physical layer of the IC and the process scheduling from the program control flow. The computers IC and program instruction set will be based on which system the controller is in operation of; such as, the engine management system, wheel antilock control system, air conditioning control system, etc. Regardless of the type of control system that is used, the program will work with the CPU to carry out its tasks. The first task is to initiate its base program that will control and operate the device. In one of the subroutines, the CPU will do a self test on its internal circuits. This self test is based on what the programmer decided was necessary to check for the operation of the device. This is important because the programmer is checking the circuit based on an analysis of a good circuit and what might fail within that circuit. To accomplish this, a set of instructions are written that allows set points or thresholds to be above or below a set value for a set time period. If these predetermined set points are broken, the code in the program will show that this is equal to "true" and the instructions for this outcome will be to activate the warning lamp and set a diagnostic trouble code (DTC). If this test is to check the CPU, the DTC may read "internal failure" and the diagnostic trouble tree would instructed you to replace the control unit. What needs to be understood here is that the instructions did not account for someone putting in an extra fuse or relay in a spare location or possibly a short circuit. This additional circuit now allows a power to be applied to the CPU that can change the internal voltage on the circuit that the program is looking at, thus setting a false DTC.

HEX	DEC	BINARY
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

fig 2

Once the subroutine self test for the internal circuit has run and passed, the program will initiate the next subroutine to check the basic circuits of the system that it controls. Each one of these subroutines will be labeled with the circuit that is to be tested. One example of this is a P0122, a program label that has an instruction set that will check the powertrain throttle position sensor (TPS) for a voltage that is less than 0.2volts for one second. If the voltage is below 0.2volts for one second, the program will equal "true" and the DTC label P0122 will be stored. The program instruction will be written to check each circuit that the programmer deems important. The circuit tests will have set points against time that are assigned to each of these individual circuits. These set points will be programmed at values that are above and below the operational voltage range of the circuit. The purpose of these initial tests is to find a circuit that is in a gross failure at system start. If a circuit breaks the set point over the correct time period, the program will be equal to "true" and will carry out the correct program instructions. These instructions may be to set this DTC in a pending column or to set a mature DTC and turn on the warning lamp. What is important here is to understand that a circuit that is within its operational range may not set a DTC. One example of this is if the TPS has unwanted resistance in the ground circuit that causes the sensed voltage to rise above the clear flood mode set point (figure 3). This is a mode to shut the injector pulse down during cranking so the spark plugs can dry. However if the engine is not flooded, this unwanted condition will cause a no start condition. The program instructions would be; if the crank signal equals "true", and if the TPS voltage is greater than 4.0 volts equals "true", disable fuel injector pulse. The CPU acts on the data that are present. It has no way to check the driver intent other than the actual TPS voltage, and that TPS voltage is in range.

When writing the program, the programmer never assumed the TPS signal would fail. The program is written on the assumption that the system is working correctly. There will be no DTC set for a problem such as this because the CPU carried out its instruction set correctly. Once the subroutine self test has run to completion, the tests may be suspended until the next key cycle or they may be run in a continuous loop. The base program will be running to control the device, taking inputs through logic circuits that have program instructions. These instructions are based on algorithms so the proper

outputs can be obtained. During the control of the device, DTC subroutines will be run to check the operation of the circuits, sensors and actuators. In order to obtain the highest probability of a successful DTC subroutine test, enabling criteria will be used, which is a way of controlling the outcome of the test by controlling the variables. Just as in a laboratory to obtain consistent results, the test must hold the variables to a minimum and it must have a set procedure that will be used to accomplish this.

The enabling criteria accomplish this by only allowing the DTC subroutine to run under certain conditions. Once all the conditions have been met, the DTC subroutine is allowed to run. One example of this would be a small leak detected in EVAP system when running a DTC P0442. The enabling criteria might read as; fuel tank level greater than 15% and less than 85% equal “true”, ambient temperature greater than 30°F and less than 95°F equal “true”, BARO greater than 70kPa equal “true”. Since the test is dealing with a pressure in the fuel containment system, all criteria must be controlled that can affect the pressure changes within the system. If the fuel tank is full, the non-liquid area within the fuel tank is very small and may show a pressure change that is not an actual leak and thus set a false DTC. If the fuel tank level is empty, the, the non-liquid area within the fuel tank is very large and may not show a pressure change of an actual leak. In this case, a DTC is not set that should have been set. If the ambient temperature is lower than 30°F, the gaseous phase above the fuel could be contracting and show a pressure change that is not an actual leak, thus setting a false DTC. If the ambient temperature is higher than 95°F, the gaseous phase above the fuel could be expanding and may cover up an actual leak. In this case a DTC is not set that should have been set. If the BARO is below 70kPa (high elevation) the pressure between the inside of the fuel containment system and the atmosphere will not be enough to change the sensed pressure inside the tank if a leak is present and may not set a DTC that should have been set. As you can see, the enabling criteria are a way to control the results of the DTC subroutine so the best possible conditions are present during the testing sequence.

It will be important to look at the enabling criteria at the beginning of the trouble tree for the DTC. This will show which sensors are being used to allow the DTC subroutine to run. If a sensor reading is inaccurate, it can allow the test to run at the wrong time or may not run the test at all. In the example of the P0422 the sensors that are used would be; fuel level sensor, intake air temperature, and manifold absolute pressure sensor. If the fuel level sensor misread and showed ¾ of a tank when the actual fuel tank was full it would allow the DTC subroutine to run and would set a false P0422.

When a DTC is set, it will be necessary to check a wiring diagram to see which wires are present at the controller. In order to set a DTC, the CPU must be able to check the outcome of the DTC subroutine. To accomplish this, a circuit must be wired to the control unit that the programmer can use to check the outcome of the test. This can be done with a direct test or an indirect test. In a direct test, the circuit that is to be tested can be monitored by an electronic device such as an analog-to-digital converter (A/D). In this type of test, the programmer can write code that can check the voltage of the circuit directly with an A/D converter. The A/D converter changes an analog voltage to a digital code comprised of “0s”and”1s”. The CPU can be programmed to understand the sequence of “0s”and”1s” on a parallel bus or can be read by a serial converter that can process this data and send it on a serial line to the CPU.

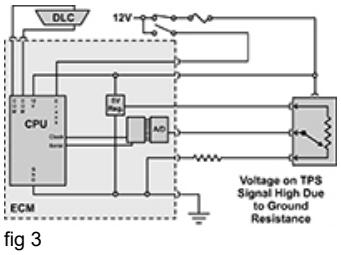


fig 3

The CPU can then use this information to check whether the voltage in the circuit had the correct change that was anticipated. One example of a direct test (figure 4) would be if a transmission solenoid was commanded to be activated. Once the program set points for a shift had been met the CPU instructions would turn on the transistor driver for the solenoid. The program instructions then check to see whether the voltage state of the solenoid control circuit had changed to the expected value. If the voltage value did not change within the program set points for a stated time period, a DTC would be stored and the appropriate instruction set would be carried out. In order for a direct test to occur, the circuit board must be designed with the physical layer of the A/D converter to monitor the circuit. This adds an extra expense to the controller that may not be needed in all of the circuits. In an example of an indirect transmission solenoid activation test (figure 5), the CPU would command the transistor driver on and then the CPU would check the input speed sensor and compare it to the output speed sensor. If the solenoid activation was completed, then the ratio between the input and output speed sensor would change to a known factor. If this factor did not change within the correct set

points, a DTC would be stored and the appropriate instruction set would be carried out. With the indirect design, the physical layer to check the input and output speed sensor is already on the circuit board so the expense of the controller is less. With this type of circuit design, it will be important to understand that a scantool will only display the commanded state, it cannot show whether that state actually occurred.

Another example of a direct or indirect circuit test would be a P0135 oxygen sensor heater circuit fault. If the oxygen sensor heater circuit is wired directly to the control unit, an A/D converter with a basic shunt circuit will be used to check directly the current of the circuit (figure 6). This test circuit will allow a voltage drop to occur that the A/D converter can read and that is directly proportional to the current flow. Now the programmer can write an instruction set that will convert this voltage from the A/D converter to an amperage reading. If this voltage is not within the program set point a DTC will be stored. In the case of an indirect reading for the oxygen sensor heater circuit, there will not be a heater circuit wire connected to the control unit (figure 7). The heater circuit power wire will be supplied directly from the ignition circuit or a relay and the heater circuit ground will be connected directly to the ground plain.

When you check a wiring diagram and see that the wiring for the associated DTC is not directly connected to the control unit, you must look for the wire that could convey the information to the CPU directly. In order for the programmer to write instructions for the DTC subroutine, a circuit must be connected to the control unit that can be used to obtain the information for the DTC. With the P0135, the CPU has a circuit inside the control unit that applies a bias voltage to the oxygen sensor signal wire. When the zirconium dioxide oxygen sensor sensing bulb is cold, the resistance is greater than 100,000,000 ohms and when this sensing bulb is heated to 700°F, the resistance drops to less than 100 ohms. This sensing bulb resistance change can be used to check the operation of the heater circuit by applying a regulated voltage to a very large resistor inside the control unit that is connected in series to the sensing circuit. This creates a voltage divider circuit. The resistor inside the control unit is fixed and the sensing bulb has a variable resistance that changes when heated. If the heater circuit is working, it will heat the sensing blub thus changing the resistance of the bulb. As the resistance of the sensing bulb drops, so does
the voltage between the resistor and the sensing bulb. By checking the voltage change between the fixed resistor and the sensing blub, over a set time period, the programmer can write an instruction set that can run the DTC subroutine. This can check the oxygen sensor heater circuit without a direct connection to the CPU. It will be important to understand how these systems operate so they can be repaired quickly and accurately.

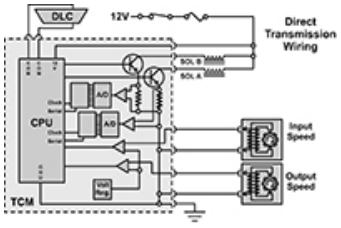


fig 4

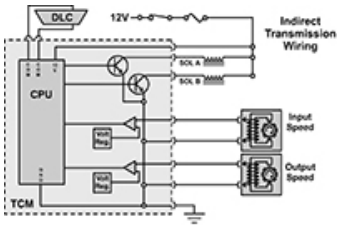
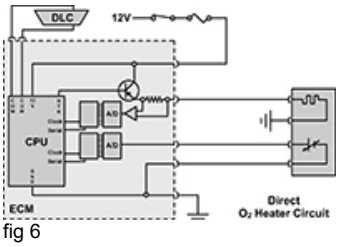
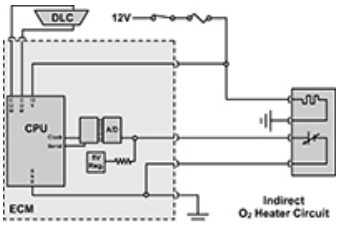


fig 5


DLC
12V
CPU
A/D
A/D
ECM
Direct
O2 Heater Circuit
fig 6


DLC
12V
CPU
A/D
ECM
Indirect
O2 Heater Circuit
fig 7