ISO-15031 Protocol

Introduction

ISO-15031 Protocol consists of a number of parts taken together to provide a coherent self-consistent set of specifications to facilitate emissions-related diagnostics. Parts 2 through 7 are based on SAE-recommended practices. This part of ISO 15031 is based on SAE J2012: MAR99 (Recommended Practice for the Diagnostic Trouble Code Definitions).

The ISO 15031-1 Protocol provides an introduction to the series of International Standards. Most automobile manufacturers equip at least a portion of their product line with some on-board diagnostic (OBD) capability. These systems provide an implication as to the general location of the diagnosed malfunction. This information is provided through an alphanumeric code.

ISO 15031-6 (Diagnostic trouble code definitions)

This part of ISO 15031 provides recommended uniformity for the alphanumeric trouble codes. It further provides guidance for the uniform messages associated with these codes. It specifies several sections addressing format, structure, messages and a few examples, and is applicable to electrical/electronic systems diagnostics of motor vehicles.

DTC General specifications:

The recommended DTCs consist of a three-digit numeric code preceded by an alphanumeric designator. The alphanumeric designators are "B0", "B1", "B2", "B3", "C0", "C1", "C2", "C3", "P0", "P1", "P2", "P3", "U0", "U1", "U2", "U3", corresponding to four sets of body, four sets of chassis, four sets of powertrain and four sets of network trouble codes. The code structure itself is partially open-ended. A portion of the available numeric sequences (portions of "B0", "C0", "P0", "P2", "P3", "U0" and "U3") is reserved for uniform codes assigned by this or future updates.

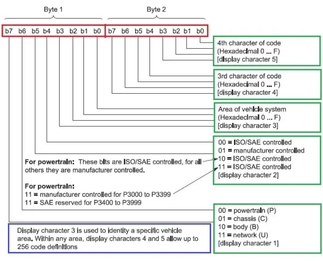
System	Code categories	Hex value	Appendix
Body	B0xxx – B3xxx	8xxx – Bxxx	B
Chassis	C0xxx – C3xxx	4xxx – 7xxx	C
Powertrain	P0xxx – P3xxx	0xxx – 3xxx	P
Network	U0xxx – U3xxx	Cxxx – Fxxx	U

Detailed specifications of the DTC format structure are specified in Clause 5. Most circuit, component, or system diagnostic trouble codes that do not support a sub-fault strategy are specified by four basic categories:

1. Circuit/open.
2. Range/performance.
3. Circuit low.
4. Circuit high.

DTC Format structure:

The diagnostic trouble code consists of an alphanumeric designator, B0-B3 for the body, C0-C3 for chassis, P0-P3 for powertrain, and U0-U3 for network communication, followed by three characters. The assignment of the proper alpha designator should be determined by the area most appropriate for that function. In most cases, the alpha designator will be implied since diagnostic information will be requested from the particular controller. However, this does not imply that all codes supported by the particular controller shall have the same alphanumeric designator. The codes are structured as in Figure below:



Structure of diagnostic trouble codes

EXAMPLE 1: The 2-byte DTC as a data bus value $9234 would be displayed to technicians as the manufacturer controlled body code B1234 as:

ISO-15031 Protocol 2 byte diagnostic trouble code structure

2-byte diagnostic trouble code structure

EXAMPLE 2 The 3-byte DTC as a data bus value $923400 would be displayed to technicians as the manufacturer controlled body code B1234-00. See Annex D for the DTC Low Byte (Failure Type Byte) definitions. The low byte shall be displayed in the hexadecimal format, e.g. $1A shall be displayed as 1A.

ISO-15031 Protocol 3 byte diagnostic trouble code structure

3-byte diagnostic trouble code structure

The Codes have been specified to indicate a suspected trouble or problem areas, and are intended to be used as a directive to the proper service procedure. To minimize service confusion, fault codes should not be used to indicate the absence of problems or the status of parts of the system (e.g. powertrain system O.K., or MIL activated), but should be confined to indicate areas in need of service attention. The ranges have been expanded beyond 100 numbers by using the hexadecimal base 16 number system.

ISO/SAE controlled DTC:

ISO/SAE-controlled diagnostic trouble codes are those codes where industry uniformity has been achieved. These codes were felt to be common enough across most manufacturers' applications that a common number and fault message could be assigned. All unspecified numbers in each grouping are the ISO/SAE reserved for future growth. Although service procedures may differ widely amongst manufacturers, the fault being indicated is common enough to be assigned a particular fault code. Codes in this area are not to be used by manufacturers until they have been approved by the ISO/SAE.

Manufacturer controlled DTC:

Areas within each alpha designator have been made available for the manufacturer-controlled DTCs. These are the fault codes that will not generally be used by a majority of the manufacturers due to basic system differences,
implementation differences, or diagnostic strategy differences. Each vehicle manufacturer or supplier who designs and specifies diagnostic algorithms, software, and diagnostic trouble codes are strongly encouraged to remain consistent across their product line when assigning codes in the manufacturer-controlled area. The codes where possible, the same groupings should be used as in the ISO/SAE controlled area, i.e. 100's and 200's for fuel and air metering, 300's for ignition system or misfire, etc. While each manufacturer has the ability to define the controlled DTCs to meet its specific controller algorithms, all DTC words shall meet ISO 15031-2.

Diagnostic trouble code descriptions:

Diagnostic Trouble Codes or OBD2 Trouble Codes are codes that the car's OBD system uses to notify you about an issue. Each code corresponds to a fault detected in the car. When the vehicle detects an issue, it will activate the corresponding trouble code. A vehicle stores the trouble code in its a memory when it detects a component or system that's not operating within acceptable limits. The code will help you to identify and fix the issue within the car.

To implement the DTC, The total vehicle functionality is divided into 4 categories as Body, Chassis, Powertrain, & Network & Vehicle Integration.

1. **Powertrain systems:** The Body (**B**-codes) systems category covers functions that are, generally, inside of the passenger compartment. These functions provide the vehicle occupants with assistance, comfort, convenience, and safety.
2. Body systems: The body systems category covers functions that are, generally, inside of the passenger compartment. These functions provide the vehicle occupants with assistance, comfort, convenience, and safety. Each specified trouble code has been assigned a description to indicate the component or system area that was determined to be at fault.
3. Chassis systems: The chassis systems category covers functions that are, generally, outside of the passenger compartment. These functions typically include mechanical systems such as brakes, steering, and suspension. Each specified trouble code has been assigned a description to indicate the component or system area that was determined to be at fault.
4. Network and vehicle integration systems: The network communication and vehicle integration systems category covers functions that are shared among computers and/or systems on the vehicle. Each specified trouble code has been assigned a description to indicate the component or system area that was determined to be at fault.

Diagnostic trouble code naming Convention:

Search Your Idea Here...
Piest Forum - Android App Link
Gift A Cup Of Coffee To PiEmbSystech
JOIN TELEGRAM FOR TECH. DISCUSSION

Recent Posts
Understanding Inheritance in CPP Programming Language
Objects of a Class in CPP Language
Classes in CPP Programming Language
How Crystal Oscillators Work in Microcontrollers: Ensuring Precise Timing and Stable Performance
Download and Install Turbo C++ Compiler

Archives
Select Month

Embedded Research Forum

Table Of Contents
8051 Microcontroller
8085 Microprocessor
8086 Microprocessor
About Us
Account
Adaptive AUTOSAR
Advanced driver assistance systems (ADAS)
Arduino
ARINC Protocol
ARM Microcontroller
Artificial Intelligence
Assembly Language
Automotive Architecture
Automotive BAP Protocol
Automotive ECU
Automotive Protocols
Automotive Safety
AUTOSAR
AUTOSAR DCM
AVR Microcontroller
Basic Electronics
Basic Understanding Of VLSI
BlueTooth Protocol
Boot Loader
ByteFlight Protocol
C Plus Plus (CPP) Tutorial
C-Language
CAN Protocol
CAN-FD Protocol
CAN-TP Protocol
CanaLyzer
Canoe
CAPL Language
ChibiOS/RT
CMSIS-RTOS
Contact Us
Contiki RTOS
Cookie Policy
CPU Design
Dashboard
Disclaimer
DMC
DoCAN Protocol
DoIP Protocol
DSRC Protocol
eCos RTOS
Edit
Embedded Linux
EtherNet Protocol
FlexRay Protocol
FlexRay Transport Protocol (ISO 10681-2)
Free RTOS
Guest Post
Home
HTTP (Hypertext Transfer Protocol): An Overview of the Internet's Most Widely Used Protocol
Hw/Sw Interface
I2C Protocol
INTEGRITY Operating System
IoT
ISO-15031 Protocol
K-Line Protocol
KWP-2000 Protocol
LIN Protocol
Linux Basics
Linux Device Driver
Linux IPC
Linux Kernel
Linux System Architecture
Login
Long Range Wide Area Network (LoRaWAN) Protocol
Mastering MIPI I3C Protocol: A Comprehensive Guide to Efficient Communication Between Devices
Mbed OS: Arm's Open-Source OS for IoT Devices
Message Queuing Telemetry Transport (MQTT) Protocol
Microcontroller
MODBUS Protocol
MOST Protocol
Motor Design
Nucleus RTOS
OBD-II
Operating System
Order Received
OSAL
OSEK
Payment
Power Electronics
PowerPC Processor
Privacy Policy
QNX RTOS
Raspberry-Pi
Register
Resistor
RIOT Operating System
Robotics
RT-Thread RTOS
RTLinux RTOS
RTOS Concept
SAE J1708 Protocol
SAE-J1939 Protocol
SENT Protocol
SPI Communication Protocol: A Comprehensive Guide to Serial Peripheral Interface
Subscription
Terms and Conditions
Thank You
ThreadX RTOS: A Lightweight and Scalable RTOS for Embedded Devices
TinyOS
Transmission Control Protocol (TCP/IP)
UART Protocol
uC/OS RTOS
UDS Protocol
Understanding of Internet Protocol (IP) - The Backbone of the Internet
USB Protocol
User Datagram Protocol (UDP)
V2X Communication
VxWorks RTOS: A High-Performance Real-Time Operating System for Embedded Systems
Wi-Fi Protocol
Windows OS: An Evolution in GUI Based OS
XBEE Protocol
XCP Protocol
Zephyr RTOS

Automotive Electronics
Computer Science
Electronics Technology
Linux System
Programming Language
C
C++
Python
Robotics Technology
VLSI