# Overview of DTC Handling in the Global B VIP

## Introduction

Diagnostic Trouble Codes (DTCs) are error codes which indicate the presence of certain fault conditions in an ECU. Each ECU in the vehicle has its own set of DTCs which are specific to the purpose, functionality, and connectivity of that ECU. For ECUs adhering to the AUTOSAR software architecture, the supported DTCs are defined as part of the ODX file, which is integrated into the software as part of the System Extract.

This document is intended to provide an introduction and overview of how DTCs are handled in the Global B VIP.

## Some Definitions

The following terms are useful to know as they relate to DTCs:

* **DTC Code**: A code used to represent a specific type of fault condition. This code has two representations: SAE and hexadecimal. For example, the Lost Communication with Body Control Module DTC has the SAE code U014000 and the hexadecimal code 0xC14000. The DTC code is often separated in a 2-byte “code” portion (0xC140) and a 1-byte “failure type byte” (FTB) portion (0x00).

|  |  |  |
| --- | --- | --- |
| **Representation** | **Code** | **FTB** |
| SAE | U0140 | 00 |
| Hexadecimal | 0xC140 | 0x00 |

* **Fault Condition**: A condition within the CSM that is abnormal and which negatively affects its performance and/or functioning. For example, a microphone open circuit and an antenna short-to-ground condition are examples of fault conditions which may exist in the CSM.
* **Test**: A check performed by the CSM which is intended to detect the presence of a fault condition. For example, the CSM may regularly read the current draw on a particular GPIO line to see if it drops below a certain threshold which would indicate an open circuit condition.
* **Test Passed**: A test has shown that a fault condition does not currently exist.
* **Test Failed**: A test has shown that a fault condition does currently exist.
* **Diagnostic Monitor**: A software component which performs tests to monitor for fault conditions.
* **Enable Conditions**: A set of conditions, all of which must be satisfied for a DTC to be enabled. For example, Power Mode Run or Propulsion for 5 seconds, and Voltage in Range are examples of enable conditions.
* **Debouncing**: A technique used to damp out transient effects and momentary blips and for confirming that a fault condition exists.
* **DTC\_Triggered Message**: A CAN message that is sent periodically by the CSM which contains the DTC Code and Failure Type Byte (FTB) of the last DTC set. For the CSM, this message has Arb ID 0x4EE.
* **Confirmed DTC**: Refers to a DTC which occupies a place in a memory storage slot. The CSM currently has 8 available storage slots.
* **Aging**: Refers to the process by which a DTC is “healed” when the fault condition which generated the DTC is no longer present. Specifically, an aging DTC will be cleared after 40 operation cycles have been completed.
* **Operation Cycle**: Essentially an ignition cycle. A new operation cycle starts when the power mode transitions from Off to Run/Propulsion or from Accessory to Run/Propulsion. The current operation cycle ends when the power mode transitions from Accessory/Run/Propulsion to Off.
* **Extended Data Record**: Metadata associated with a DTC. An Extended Data Record consists of the Timestamp, Occurrence Counter, and Aging Cycle Counter.

## Types of DTCs

There are two types of DTCs in the VIP:

1. **Internal (VIP-monitored) DTCs**: These are DTCs for which the VIP functions as the Diagnostic Monitor. There are two subtypes of internal DTCs:
   1. Loss of Comm (LoC) DTCs. These DTCs indicate when the CSM has lost communication with another ECU on the CAN bus. The VIP will monitor a set of supervised CAN signals on the CAN bus, and if it fails to receive one or more of these signals within a certain time period, it will set a Loss of Com DTC.
   2. All Other DTCs.
2. **External (SoC-Monitored) DTCs**: These are DTCs for which the SoC functions as the Diagnostic Monitor. For external DTCs, the VIP relies on the SoC to inform it when a test has passed or failed. The SoC uses IPC 0x05 for this purpose. See the IPC Message Specification for more detail.

## Software Components

There are a number of components which are responsible for handling in the Global B VIP. Each component plays a role. This can initially be quite confusing as the functionality is distributed among various software components and it can be hard to figure out what component does what. Here is a summary of which components are involved and what their role is in handling DTCs:

At the Application Layer:

1. **dtc\_list.h**. This is an auto-generated file that contains the list of all DTC definitions, generated from the CalDef database using a script. If the DTC does not have a DTC\_MASK calibration defined, you can add a DTC definition manually to the bottom of this file.
2. **SWC\_DTC\_Handling**. In general, this component is responsible for implementing the Diagnostic Monitors for the DTCs. However, in the case of the Loss of Com DTCs, the Diagnostic Monitors are actually implemented by the SUM\_ERRH component. For the Loss of Com DTCs, the SWC\_DTC\_Handling component is simply responsible for passing the GetEventStatus request to the DEM in the SWC\_Read\_DTC\_Status function. This function is called when a Read DTC Status Request IPC Message (tag 0x04) is received from the SoC.
3. **IoHwAb\_XX**. There are actually several components represented here: IoHwAb\_30, IoHwAb\_ExtAmp, IoHwAbs\_ElmosIC, etc. These components directly interface with various hardware peripherals and are responsible for the actual detection of hardware-based fault conditions. For example, IoHwAbs\_ElmosIC periodically reads an error register from the ELMOS E522-40 chip over I2C to determine if it has detected an open load or overcurrent condition on one of the antennas. If a fault is detected, the IoHwAb component informs SWC\_DTC\_Handling.
4. **SUM\_SSM**. This component monitors the status of the supervised signals for a particular ECU. If it detects a change in status of one or more of the signals (e.g. it has not received the signal in the expected period of time), it notifies the SUM\_ERRH component of the event. See GB8002 for more information.
5. **SUM\_ERRH**. This component is an error aggregator. For the Loss of Com DTCs, this component also implements the Diagnostic Monitors. When the SUM\_SSM notifies the SUM\_ERRH of a change in signal status, the SUM\_ERRH lets the DEM know that the DTC status has changed. See GB8002 for more information. This component also performs the DTC\_MASK calibration checks, but only for the Loss of Com DTCs. The calibration checks for the other DTCs are performed in SWC\_DTC\_Handling. This component also contains some logic for handling the sending of the DTC\_Triggered CAN message.
6. **SUM\_ERRH\_Adapter.c**. In this file, the DTC\_MASK calibrations for the Loss of Com DTCs are initialized. When a new Loss of Com DTC is added, a new entry needs to be added in this file.
7. **DEM\_SUMS\_Adapter**. This component sits between the DEM and the SUM\_ERRH. When the DEM triggers the UdsStatusChanged event for a DTC, the DEM\_SUMS\_Adapter handles it by doing two things: (1) informing SUM\_ERRH of the event (so that the DTC\_Triggered CAN message can be sent), and (2) sending the appropriate IPC message to the SoC to inform it of the change in DTC status.
8. **Cdd\_J6, Cdd\_J6\_cbk, and J6\_prv\_Diagnostics**. These components handle the sending and receiving of IPC messages to/from the SoC.

At the BSW Layer:

1. **DEM**. This component stores the “golden” copy of the DTC’s status in non-volatile memory. It triggers the UdsStatusChanged event when the status of a DTC changes. It also handles things such as DTC confirmation, aging, enable conditions, event priority, Extended Data Records, and the Operation Cycle definition. See the [AUTOSAR DEM software specification](https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_SWS_DiagnosticEventManager.pdf) for more details.
2. **NVRAM**. This component manages the non-volatile memory (i.e. EEPROM), where the DTC status is stored.
3. **DCM**. This component implements the UDS protocol as defined in ISO 14229-1. See the [AUTOSAR DCM software specification](https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_SWS_DiagnosticCommunicationManager.pdf) for more details.
4. **COM**. This component is responsible for handling and processing CAN signals. This component is responsible for detecting signal timeout events, which are used to the detect fault conditions for Loss of Com DTCs.

Here is a very crude diagram depicting the interconnections between components as they relate to the handling of the Loss of Com DTCs. In general, the components communicate with each other via the AUTOSAR RTE, with exception of DEM, DCM, and NVRAM, which communicate with each other via direct C API calls. Note that not all connections are depicted here.

## IPC Messages for DTCs

The SoC can read the status of a DTC from the VIP using the Read DTC Status Request IPC message (0x04). The VIP will respond with an IPC 0x04 response with the current status. The function which handles the processing of IPC 0x04 is the **SWC\_Read\_DTC\_Status()** function in SWC\_DTC\_Handling.

The SoC updates the status of an external DTC using the Update DTC Status Request IPC message (0x05). If all of the enable conditions are satisfied, the VIP will update the status of the DTC as specified by the SoC. It will also respond with an IPC 0x05 message regardless of whether the DTC status was changed. IPC 0x05 requests are processed in the **ProcessExternalDiagnosticDTCResponse()** and **MonitorExtDtcStatus()** functions in SWC\_DTC\_Handling. IPC 0x05 responses are generated in the **ProcessExternalDiagnosticDTCResponse()** and **MonitorExtDtcStatus()** functions in SWC\_DTC\_Handling and in the **CBEventUdsStatusChanged\_DTC\_0xXXXXXX\_DemCallbackEventStatusChanged\_CallbackEventUdsStatusChanged()** callback functions in DEM\_SUMS\_Adapter.

If the VIP changes the status of a DTC without receiving an IPC 0x05 message from the SoC, the VIP will send a DTC Changed Notification IPC message (0x06) to the SoC. This message is handled within the **CBEventUdsStatusChanged\_DTC\_0xXXXXXX\_DemCallbackEventStatusChanged\_CallbackEventUdsStatusChanged()** callback functions in DEM\_SUMS\_Adapter.

See the IPC message specification for more details about these messages.