Bài 1.

Khóa học Nâng cao Kỹ năng Kiểm thử Ô tô: Lý thuyết và Thực hành Giao thức Mạng CAN

1. Giới thiệu về Khóa học

- **Người giảng dạy**: Khóa học được hướng dẫn bởi một kỹ sư kiểm thử cao cấp trong lĩnh vực ô tô.
- **Mục tiêu của khóa học**: Khóa học này rất quan trọng cho những ai muốn nâng cao kỹ năng trong lĩnh vực kỹ thuật ô tô, đặc biệt là kiểm thử. Khóa học giúp học viên phát triển các kỹ năng cần thiết để ứng tuyển vào các vị trí công việc trong lĩnh vực ô tô trên toàn thế giới.

2. Cấu trúc Khóa học

- **Hai phần chính**:
    - **Phần 1: Lý thuyết**: Cung cấp kiến thức cơ bản về giao thức mạng CAN (Controller Area Network), là nền tảng để hiểu các khía cạnh kỹ thuật trong phần hai.
    - **Phần 2: Thực hành kỹ thuật**: Hướng dẫn sử dụng các công cụ của Vector, phần mềm và cấu hình phần cứng Vector. Học viên sẽ học cách cấu hình phần mềm Vector CANoe để đọc và gửi các thông điệp CAN trên mạng CAN.

3. Ứng dụng Thực tế và Cơ hội Nghề nghiệp

- **Tìm kiếm công việc trên LinkedIn**: Khóa học này được thiết kế để giúp học viên đủ khả năng ứng tuyển vào các vị trí công việc yêu cầu kỹ năng về mạng CAN. Khi tìm kiếm công việc liên quan đến CAN trên LinkedIn, học viên sẽ thấy rằng loại công việc này được yêu cầu ở nhiều quốc gia trên toàn thế giới.
    - **Yêu cầu công việc**: Các công việc này đòi hỏi kỹ thuật viên phải có khả năng quản lý môi trường thử nghiệm, thiết lập cấu hình CANoe chính xác, và xử lý các hệ thống tự động hóa.

4. Công cụ và Phần mềm Liên quan

- **Phần mềm CANoe**: Phần mềm này cho phép kiểm soát hành vi của nhiều đơn vị điều khiển điện tử (ECU) kết nối trong mạng, do đó có tính linh hoạt cao hơn.
- **Phần mềm Analyzer**: Chỉ cho phép kiểm soát hành vi của một ECU duy nhất, phù hợp với các tác vụ đơn giản hơn.
- **ECU (Electronic Control Unit)**: Là đơn vị điều khiển điện tử, ví dụ như ECU của hệ thống động cơ chịu trách nhiệm điều khiển việc di chuyển của xe, hoặc ECU của hệ thống phanh chịu trách nhiệm điều khiển quá trình phanh xe.

5. Cấu trúc và Chức năng của Mạng Điện trong Xe

- **Mạng điện trong xe**: Một chiếc xe có thể có nhiều mạng CAN khác nhau, mỗi mạng chịu trách nhiệm truyền tải thông tin và điều khiển các hệ thống khác nhau trong xe.
    - **Mạng CAN của động cơ**: Điều khiển việc di chuyển của xe, có thể là quan trọng nhất.
    - **Mạng CAN của hệ thống phanh**: Điều khiển hệ thống phanh, có thể tự động kích hoạt phanh khi phát hiện chướng ngại vật thông qua tín hiệu từ camera.
- **Chức năng của các nút mạng đặc biệt (Gateway Nodes)**: Trong các xe hiện đại, có thể có từ hai đến bốn đường mạng CAN khác nhau kết nối với nhau. Các nút mạng đặc biệt này đóng vai trò như các cổng vật lý, mở hoặc đóng kết nối truyền thông giữa các mạng CAN, đảm bảo dữ liệu có thể được trao đổi một cách hiệu quả giữa các mạng khác nhau.

6. Cấu trúc và Hoạt động của Hệ thống CAN trong Xe Điện

- **ECU động cơ**: Chịu trách nhiệm điều khiển động cơ và các bánh xe của xe. Đây có thể là ECU quan trọng nhất trong hệ thống, vì nó quyết định đến sự vận hành chính của xe.
- **ECU hệ thống phanh**: Chịu trách nhiệm điều khiển hệ thống phanh. Khi hệ thống camera phát hiện chướng ngại vật, nếu người lái không phanh kịp thời, ECU này sẽ tự động kích hoạt phanh để tránh tai nạn. Điều này tương đương với việc có một người lái xe cực kỳ cẩn thận, luôn phanh xe đúng lúc.

7. Phần tiếp theo của khóa học

- **Bài học tiếp theo**: Sẽ đi sâu vào chi tiết giao thức mạng CAN và cách thức truyền thông hoạt động trong hệ thống.

Bài 2.

Giao thức Mạng CAN: Tầng Vật lý và Chức năng của Transceiver

1. Giới thiệu về Giao thức Mạng CAN

- **Mục đích của Mạng CAN**: Mạng CAN (Controller Area Network) là hệ thống truyền thông được thiết kế để phục vụ việc liên lạc trong xe ô tô và trong các ứng dụng công nghiệp.
    - **Giao thức CANopen**: Khi được áp dụng trong công nghiệp, giao thức CAN được gọi là CANopen.
    - **Truyền thông thời gian thực**: Mạng CAN cho phép các thiết bị giao tiếp với nhau trong thời gian thực mà không cần một máy chủ trung tâm.

2. Đặc điểm của Mạng CAN

- **Không yêu cầu hệ thống địa chỉ**: Khác với Ethernet, mạng CAN không sử dụng địa chỉ mạng cụ thể cho từng nút (node). Thay vào đó, mạng CAN sử dụng cơ chế lọc mặt nạ (mask filter) để xác định thông điệp cần xử lý.
- **Giao thức dựa trên thông điệp**: Mạng CAN là một giao thức hoàn toàn dựa trên thông điệp, nghĩa là mỗi thông điệp được xác định thông qua trường định danh (identifier field) chứa trong nội dung của nó. Các nút trong mạng sẽ quyết định liệu thông điệp đó có liên quan hay không dựa trên định danh này.
- **Tính năng tránh va chạm (Collision Avoidance)**: Mạng CAN có tích hợp cơ chế tránh va chạm, giúp đảm bảo rằng các thông điệp được truyền tải mà không bị xung đột trên bus.

3. Tầng Vật lý của Mạng CAN

- **Cấu trúc dây dẫn**: Tầng vật lý của mạng CAN chỉ bao gồm hai dây điện chính là CAN H (CAN High) và CAN L (CAN Low).
    - **Dây xoắn đôi (Twisted Pair)**: Hai dây này được xoắn lại với nhau để giảm thiểu nhiễu điện từ (EMI), đảm bảo tín hiệu được truyền ổn định.
    - **Trọng lượng nhẹ**: Việc sử dụng ít dây điện hơn giúp giảm trọng lượng tổng thể của hệ thống dây dẫn, phù hợp với yêu cầu giảm trọng lượng xe nhằm tiết kiệm nhiên liệu và giảm khí thải.

4. Hoạt động của Transceiver trong Mạng CAN

- **Chuyển đổi tín hiệu điện áp**: Sự chênh lệch điện áp giữa CAN H và CAN L được transceiver chuyển đổi thành các giá trị bit nhị phân (bit 0 hoặc bit 1).
    - **Tín hiệu dominant**: Khi chênh lệch điện áp lớn, transceiver sẽ gửi tín hiệu logic 0 (dominant).
    - **Tín hiệu recessive**: Khi chênh lệch điện áp nhỏ hoặc bằng 0, transceiver sẽ gửi tín hiệu logic 1 (recessive).

- **Bắt đầu khung thông điệp**: Khung thông điệp CAN bắt đầu khi có sự chuyển đổi từ chênh lệch điện áp bằng 0 sang chênh lệch điện áp lớn. Sự chuyển đổi này được phát hiện bởi transceiver thông qua bộ so sánh (comparator) và khởi động quá trình truyền thông điệp. Bit đầu tiên của khung thông điệp luôn có giá trị 0, biểu thị sự bắt đầu của khung.
- **Truyền và nhận tín hiệu**:
    - **Pha nhận tín hiệu**: Trong pha nhận tín hiệu, sự chênh lệch điện áp giữa CAN H và CAN L được đưa vào bộ so sánh. Nếu chênh lệch điện áp lớn, transceiver sẽ gửi tín hiệu logic 0 (dominant); nếu chênh lệch nhỏ, sẽ gửi tín hiệu logic 1 (recessive).
    - **Pha truyền tín hiệu**: Trong pha truyền tín hiệu, transceiver nhận tín hiệu logic từ các tầng trên (như tầng OSAI) và điều khiển điện áp trên dây CAN để truyền tín hiệu ra mạng, đảm bảo các thông điệp được gửi đến đúng các đơn vị điều khiển điện tử (ECU) khác trong hệ thống.

5. Kết thúc và Kết nối Phần cứng Mạng CAN

- **Điện trở kết thúc (Termination Resistor)**: Ở cuối mỗi dây dẫn, có một điện trở kết thúc có giá trị 120 Ohm để tránh hiện tượng phản xạ tín hiệu, đảm bảo rằng tín hiệu không bị chồng chéo và truyền thông được thực hiện ổn định.
    - **Lý do chọn 120 Ohm**: Giá trị này phù hợp với trở kháng danh định của đường truyền mạng CAN, giúp cân bằng dòng điện và đảm bảo truyền thông không bị nhiễu.
- **Cấu hình phần cứng**: Mỗi nút mạng trong mạng CAN được biểu diễn bằng một hình chữ nhật nét đứt, với hai dây dẫn CAN H và CAN L được chia sẻ giữa các nút. Các dây dẫn này kết nối với các transceiver để chuyển đổi tín hiệu giữa các tầng vật lý và logic.

6. Kết luận và Phần tiếp theo của khóa học

- **Kết luận**: Tầng vật lý của mạng CAN và hoạt động của transceiver đóng vai trò quan trọng trong việc đảm bảo truyền thông giữa các nút mạng được thực hiện một cách chính xác và hiệu quả. Transceiver là thiết bị chịu trách nhiệm chính trong việc chuyển đổi các tín hiệu điện áp thành các giá trị số để xử lý tiếp theo.
- **Bài học tiếp theo**: Sẽ đi sâu vào chi tiết cấu trúc của khung thông điệp CAN và cách thức xây dựng nó.

Bài 3.

Cấu trúc Khung Thông điệp CAN: Các Thành Phần và Chức Năng Chi Tiết

1. Giới thiệu về Quá trình Xây dựng Khung Thông điệp CAN

- **Quá trình chuyển đổi tín hiệu**: Trong mạng CAN, sự chênh lệch điện áp giữa hai dây CAN H (CAN High) và CAN L (CAN Low) được chuyển đổi thành các giá trị bit nhị phân (bit 0 hoặc bit 1). Việc chuyển đổi này được thực hiện bởi transceiver, một thiết bị chuyển đổi các tín hiệu điện thành các giá trị số tương ứng trên đường Orix.
- **Bắt đầu khung thông điệp**: Khung thông điệp CAN bắt đầu khi có sự chuyển đổi từ chênh lệch điện áp bằng 0 sang chênh lệch điện áp lớn. Transceiver sử dụng bộ so sánh (comparator) để phát hiện sự chuyển đổi này và khởi động quá trình truyền thông điệp. Bit đầu tiên trong khung thông điệp luôn có giá trị 0, biểu thị sự bắt đầu.

2. Trường Tranh chấp (Arbitration Field)

- **Chức năng của trường tranh chấp**: Trường này xác định định danh (identifier) của một thông điệp cụ thể. Trong mạng CAN, đây là phần quan trọng nhất để xác định xem một thông điệp có liên quan đến một nút mạng cụ thể hay không.
- **Độ dài của ID**:
    - **ID 11 bit**: Sử dụng khi số lượng thông điệp trên mạng ít hơn 2^11 (2048 thông điệp). Đây là độ dài ID thường gặp trong các mạng CAN ứng dụng thực tế.
    - **ID 29 bit**: Sử dụng trong các mạng CAN chẩn đoán, khi cần xử lý nhiều thông điệp hơn.
- **Ứng dụng thực tế**:
    - **CAN Application Messages**: Là các thông điệp trao đổi giữa các nút điều khiển trong thời gian thực khi xe đang hoạt động.
    - **CAN Diagnostic Messages**: Là các thông điệp trao đổi giữa hệ thống kiểm tra và cổng OBD (On-Board Diagnostics) của xe trong quá trình kiểm tra và bảo dưỡng.

3. Trường Kiểm soát (Control Field)

- **Chức năng của trường kiểm soát**: Trường này xác định lượng dữ liệu mà thông điệp muốn truyền tải đến các nút nhận. Nó chứa thông tin về số lượng bit dữ liệu sẽ được truyền đi trong trường dữ liệu.
- **Ví dụ**: Trong một số trường hợp, trường kiểm soát có thể chỉ có một bit để biểu thị lượng dữ liệu rất nhỏ.

4. Trường Dữ liệu (Data Field)

- **Chức năng của trường dữ liệu**: Đây là phần cốt lõi của khung thông điệp, nơi chứa các thông tin quan trọng của xe, chẳng hạn như tốc độ xe, vị trí bàn đạp phanh, hoặc các thông tin liên quan khác.
- **Độ dài của trường dữ liệu**:
    - **Độ dài chuẩn**: 8 bit, tương ứng với 2^8 = 256 tổ hợp thông tin khác nhau có thể được mã hóa.
    - **Độ dài mở rộng**: Có thể dao động từ 8 bit đến 64 bit, tùy thuộc vào giao thức CAN được sử dụng (chuẩn hoặc mở rộng).

5. Trường Kiểm tra Dư thừa Tuần hoàn (CRC Field)

- **Chức năng của trường CRC**: Trường này đảm bảo rằng dữ liệu trong khung thông điệp không bị lỗi trong quá trình truyền. Nó thực hiện chức năng kiểm tra lỗi bằng cách tính toán giá trị CRC (Cyclic Redundancy Check) dựa trên tất cả các bit của thông điệp.
- **Cách tính toán CRC**:
    - **Thuật toán đa thức (polynomial algorithm)**: Sử dụng một thuật toán để tính toán giá trị CRC dựa trên tất cả các bit từ bit bắt đầu khung đến bit dữ liệu cuối cùng.
    - **So sánh CRC**: Giá trị CRC được tính toán độc lập bởi cả ECU gửi và ECU nhận. Nếu hai giá trị CRC này không khớp, điều đó có nghĩa là đã xảy ra lỗi trong quá trình truyền tải dữ liệu.
    - **Hành động khi phát hiện lỗi**: Khi phát hiện lỗi, ECU nhận sẽ bỏ qua thông điệp đó và không xử lý thông tin chứa trong nó.
- **Ví dụ về sai lệch dữ liệu**: Nếu một bit dữ liệu được gửi từ ECU gửi khác với bit mà ECU nhận thấy, thì giá trị CRC được tính toán bởi ECU nhận sẽ khác với giá trị CRC của ECU gửi, dẫn đến việc thông điệp bị loại bỏ.

6. Kết luận và Phần tiếp theo của khóa học

- **Kết luận**: Khung thông điệp CAN bao gồm nhiều trường khác nhau, mỗi trường có chức năng và độ dài cụ thể, nhằm đảm bảo rằng các thông điệp được truyền tải một cách chính xác và hiệu quả trên mạng CAN.
- **Bài học tiếp theo**: Sẽ đi sâu vào chi tiết của tầng liên kết dữ liệu (Data Link Layer) trong mạng CAN.

Bài 4.

Ảnh hưởng của Độ dài Bus và Cơ chế Xử lý Thông điệp trong Tầng Liên kết Dữ liệu Mạng CAN

1. Ảnh hưởng của Độ dài Bus đến Tốc độ Bit

- **Quan hệ giữa độ dài bus và tốc độ bit**: Khi độ dài của bus CAN tăng lên, tốc độ bit cho phép sẽ giảm xuống. Điều này xảy ra vì hệ thống truyền thông điện tử bị ảnh hưởng bởi nhiều hiện tượng nhiễu, chẳng hạn như hiện tượng phản xạ và nhiễu điện tử từ các tín hiệu khác trong môi trường vật lý nơi dây dẫn CAN được đặt.
    - **Ứng dụng trong thực tế**:
        - **Trong ô tô**: Độ dài dây dẫn CAN thường dưới 40 mét, nên tốc độ bit 1000 kbps là phù hợp cho các ứng dụng trong xe.
        - **Trong công nghiệp**: Yêu cầu dây dẫn dài hơn 1 km có thể làm giảm tốc độ bit và ảnh hưởng đến khả năng truyền thông.

2. Cấu trúc và Xử lý Thông điệp trong Tầng Liên kết Dữ liệu

- **Cấu trúc của một thông điệp**:
    - **Thông điệp**: Bao gồm nhiều tín hiệu, với mỗi tín hiệu được chứa trong trường dữ liệu (data field) của khung chính (mainframe).
    - **Trường dữ liệu (payload)**: Là nơi chứa các tín hiệu của thông điệp. Số lượng tín hiệu trong một thông điệp có thể chỉ là một hoặc nhiều tín hiệu khác nhau.
    - **Tối ưu hóa băng thông**: Để tiết kiệm băng thông, cần tối ưu hóa bằng cách đưa nhiều tín hiệu nhất có thể vào một khung CAN. Điều này giúp truyền thông hiệu quả hơn và giảm tải cho mạng.
- **Định danh và Địa chỉ Thông điệp**:
    - **ID của thông điệp**: Mỗi thông điệp được xác định thông qua một ID, và dựa trên ID này, các nút (nodes) có thể định địa chỉ và xử lý thông điệp tương ứng.
    - **Cơ chế truyền tải**: Chỉ có một nút duy nhất có thể truyền thông điệp trên bus tại một thời điểm. Nếu có nhiều nút cố gắng truyền cùng lúc, các khung CAN có thể bị chồng chéo, gây ra lỗi. Điều này được ngăn chặn bằng cơ chế tránh va chạm (collision avoidance), sẽ được phân tích trong bài học tiếp theo.
- **Cơ chế Lọc Thông điệp (Filter Mask)**:
    - **Cách hoạt động**: Tất cả các nút trên bus đọc thông điệp hiện tại và quyết định xem thông điệp có liên quan hay không dựa trên ID.
    - **Cơ chế lọc**: Thông điệp không chứa địa chỉ đích cụ thể, mà chỉ có một số định danh chung. Dựa trên giá trị định danh này, các nút có thể xác định xem thông điệp cần được xử lý hay bỏ qua.
    - **Kiểm tra lỗi CRC**: Để đảm bảo truyền tải thông điệp không có lỗi, mỗi thông điệp được kiểm tra bằng cơ chế CRC (Cyclic Redundancy Check). Nếu thông điệp có lỗi, nút nhận sẽ loại bỏ nó.

3. Ví dụ về Cơ chế Lọc Thông điệp

- **Thiết lập giá trị mask và filter**:
    - **Mask**: Được thiết lập với tất cả các bit là 1, ngoại trừ hai bit cuối cùng.
    - **Filter**: Được thiết lập với tất cả các bit là 0.
    - **Hoạt động**: Nút mạng sẽ lọc thông điệp dựa trên phép nhân nhị phân giữa ID của thông điệp và giá trị mask. Nếu kết quả khớp với giá trị filter, thông điệp sẽ được xử lý; nếu không, thông điệp sẽ bị bỏ qua.
- **Ví dụ cụ thể**:
    - **Các thông điệp có ID 1, 2, 3**: Sau khi thực hiện phép nhân nhị phân với giá trị mask, kết quả khớp với giá trị filter, nên thông điệp được xử lý.
    - **Thông điệp có ID 4**: Kết quả phép nhân không khớp với giá trị filter, nên thông điệp bị loại bỏ.

4. Kết luận và Phần tiếp theo của khóa học

- **Kết luận**: Tầng liên kết dữ liệu của mạng CAN chịu trách nhiệm xử lý và truyền tải các thông điệp dựa trên các cơ chế lọc và kiểm tra lỗi. Sự tối ưu hóa việc sử dụng băng thông và cơ chế lọc giúp đảm bảo mạng CAN hoạt động hiệu quả và đáng tin cậy.
- **Bài học tiếp theo**: Sẽ tập trung vào việc giải thích cơ chế tránh va chạm (collision avoidance) trên bus CAN.

Bài 5.

Cơ chế Tránh Va chạm trong Mạng CAN: Hoạt động và Nguyên lý

1. Giới thiệu về Cơ chế Tránh Va chạm

- **Mục đích**: Cơ chế tránh va chạm trong mạng CAN được thiết kế để đảm bảo rằng khi nhiều nút (nodes) muốn truyền thông điệp cùng lúc, chỉ một nút được phép tiếp tục truyền, từ đó tránh được tình trạng xung đột dữ liệu trên bus CAN.

2. Hoạt động Cơ bản của Cơ chế Tránh Va chạm

- **Giai đoạn Bắt đầu Khung (Start of Frame)**:
    - **Tình huống cụ thể**: Giả sử có ba nút (Node 1, Node 2, và Node 3) cùng muốn bắt đầu truyền một khung thông điệp trên bus CAN. Các nút này sẽ cùng gửi tín hiệu chuyển đổi từ trạng thái thụ động (recessive) sang trạng thái ưu thế (dominant).
    - **Tín hiệu ưu thế (dominant)**: Tín hiệu ưu thế được tạo ra khi có sự chênh lệch điện áp lớn giữa CAN H và CAN L. Giá trị logic 0 biểu thị cho tín hiệu này, và đây là tín hiệu sẽ được ưu tiên trên bus.
    - **Giám sát tín hiệu**: Mỗi nút sẽ luôn giám sát tín hiệu trên bus để xác định liệu tín hiệu mà mình gửi ra có khớp với tín hiệu trên bus hay không.
- **Giai đoạn Truyền tín hiệu và Xử lý Va chạm**:
    - **Bước đầu tiên**: Tất cả ba nút đều gửi tín hiệu ưu thế (logic 0) và quan sát thấy rằng trên bus cũng có tín hiệu ưu thế. Do đó, cả ba nút tiếp tục truyền phần còn lại của khung thông điệp.
    - **Tín hiệu thụ động (recessive)**: Trong hai bit tiếp theo, cả ba nút đều gửi tín hiệu thụ động (logic 1) và nhận thấy rằng tín hiệu trên bus khớp với tín hiệu đã gửi, nên tiếp tục truyền.
    - **Xảy ra va chạm**:
        - Tại bit thứ sáu, Node 1 gửi tín hiệu thụ động trong khi Node 2 và Node 3 gửi tín hiệu ưu thế. Trên bus, tín hiệu ưu thế sẽ thắng thế và Node 1 phát hiện ra rằng tín hiệu mà nó gửi ra không khớp với tín hiệu trên bus. Điều này dẫn đến Node 1 tự động chuyển sang chế độ chỉ lắng nghe (listen-only mode) và dừng việc truyền thông điệp.
        - Node 2 và Node 3 tiếp tục truyền thông điệp. Tại bit thứ bảy, Node 3 gửi tín hiệu thụ động trong khi Node 2 gửi tín hiệu ưu thế. Tương tự, Node 3 phát hiện ra sự không khớp và chuyển sang chế độ chỉ lắng nghe. Node 2 tiếp tục truyền các trường còn lại của khung thông điệp (trường kiểm soát, trường dữ liệu).

3. Kết quả của Cơ chế Tránh Va chạm

- **Kết quả cuối cùng**: Chỉ có một nút (Node 2) tiếp tục truyền khung thông điệp trên bus CAN, trong khi các nút khác đã dừng lại do phát hiện sự không khớp giữa tín hiệu gửi và tín hiệu trên bus. Cơ chế này đảm bảo rằng chỉ có một thông điệp được truyền trên bus tại một thời điểm, tránh xung đột và đảm bảo truyền thông hiệu quả.

4. Kết luận và Phần tiếp theo của Khóa học

- **Kết luận**: Cơ chế tránh va chạm là một thành phần quan trọng trong mạng CAN, giúp đảm bảo rằng chỉ có một thông điệp được truyền trên bus tại một thời điểm, từ đó tránh được các xung đột tín hiệu. Điều này đảm bảo tính ổn định và hiệu quả của mạng CAN trong việc truyền thông tin.
- **Bài học tiếp theo**: Sẽ tập trung vào cách cấu hình hệ thống dây dẫn của mạng CAN.

Bài 6.

Summary of CAN network protocol theory from csselectronics.com

If you want to have a deeper introduction to the CAN protocol, before diving into the live CANOe sessions please download the following tutorial:

https://www.csselectronics.com/pages/confirmation-page-ultimate-guide

To download the tutorial, Click on "download now

Bài 7.

Hệ thống Dây dẫn Mạng CAN trong Xe Thực tế: Cấu trúc, Kết nối và Ứng dụng

1. Giao tiếp với ECU qua Cổng OBD

- **Cổng OBD (On-Board Diagnostics)**: Là cổng giao tiếp duy nhất để kết nối và giao tiếp với các đơn vị điều khiển điện tử (ECU) trên xe thực tế.
    - **Giai đoạn hậu mãi (post-market phase)**: Trong giai đoạn này, chỉ có thể tương tác với ECU thông qua giao tiếp chẩn đoán CAN, sử dụng cổng OBD màu đen.
    - **Giai đoạn tiền sản xuất và phát triển (pre-production phase)**: Trong giai đoạn này, cần phải đọc và ghi được các thông điệp ứng dụng CAN của tất cả các nút mạng trên xe, không chỉ giới hạn ở các thông điệp chẩn đoán.

2. Thông tin cần biết về Mạng CAN trong Xe

- **Số lượng đường CAN**: Cần xác định chính xác số lượng đường CAN mà xe sử dụng để theo dõi và kiểm soát hiệu quả.
- **Tốc độ truyền thông**: Biết rõ tốc độ truyền thông của các đường CAN để cấu hình chính xác.
- **Vị trí của dây CAN Low và CAN High**: Xác định vị trí chính xác của các dây dẫn CAN Low và CAN High để đảm bảo kết nối đúng cách.

3. Kết nối và Theo dõi Mạng CAN

- **Theo dõi nhiều đường CAN**:
    - **Trường hợp 1: Theo dõi ba đường CAN**: Sử dụng một cổng OBD để kết nối với xe. Bên còn lại sử dụng ba cổng DB9 để kết nối với phần cứng Vector VN1640L.
    - **Trường hợp 2: Theo dõi một đường CAN**: Kết nối cổng OBD với xe và sử dụng chỉ một cổng DB9 để kết nối với phần cứng Vector. Hai cổng DB9 còn lại không sử dụng.
    - **Trường hợp 3: Theo dõi hai đường CAN**: Kết nối cổng OBD với xe và sử dụng hai cổng DB9 để kết nối với phần cứng Vector, để trống cổng DB9 còn lại.
    - **Giới hạn của phần cứng Vector VN1640L**: Có thể theo dõi tối đa ba đường CAN, vì cổng DB9 thứ tư được dành riêng cho kết nối liên kết (link communication).

4. Sơ đồ Chân cắm của Cổng DB9 và OBD

- **Chân cắm của cổng DB9**:
    - **Chân số 2**: Kết nối với tín hiệu CAN Low.
    - **Chân số 3**: Kết nối với tín hiệu CAN High.

- **Chân cắm của cổng OBD**:
    - **Chân số 6**: Kết nối với tín hiệu CAN High.
    - **Chân số 14**: Kết nối với tín hiệu CAN Low.
    - **Kết nối dây dẫn**: Các chân cắm của cổng DB9 sẽ được nối tắt với các chân tương ứng của cổng OBD để đảm bảo tín hiệu truyền thông đúng.

5. Thiết lập trong Giai đoạn Tiền sản xuất trên Bàn kiểm tra (Test Bench)

- **Kết nối DB9 trong giai đoạn tiền sản xuất**:
    - **Cả hai đầu dây dẫn đều sử dụng cổng DB9**: Khi làm việc trên bàn kiểm tra trong giai đoạn tiền sản xuất, cả hai đầu của dây dẫn có thể sử dụng cổng DB9 với kết nối 1:1.
    - **Kết nối một đầu với phần cứng Vector**: Một đầu cổng DB9 sẽ nối với dây CAN Low, CAN High và đất (ground), đầu kia sẽ nối với phần cứng Vector.
    - **Số lượng cổng DB9**: Số lượng cổng DB9 cần thiết sẽ tương ứng với số lượng đường CAN mà bạn muốn theo dõi.

6. Kết luận và Phần tiếp theo của khóa học

- **Kết luận**: Hệ thống dây dẫn và kết nối của mạng CAN trên xe là yếu tố quan trọng để đảm bảo rằng quá trình kiểm tra và theo dõi mạng CAN được thực hiện chính xác và hiệu quả. Hiểu rõ về cấu trúc, sơ đồ chân cắm, và cách kết nối là cần thiết để đảm bảo hoạt động của hệ thống.
- **Bài học tiếp theo**: Sẽ hướng dẫn cách cài đặt và cấu hình môi trường Vector CANoe.

Bài 8.

Hướng dẫn Cài đặt và Cấu hình Phần mềm Vector CANoe

1. Cài đặt Phần mềm Vector CANoe

- **Tải phần mềm**: Đầu tiên, bạn cần tải phiên bản dùng thử của phần mềm Vector CANoe từ liên kết cung cấp trong phần tài nguyên của bài học.
    - **Chọn phiên bản**: Cuộn xuống trang và tải về phiên bản mới nhất phù hợp với hệ điều hành của bạn.
    - **Quá trình cài đặt**: Sau khi nhấp vào phiên bản mong muốn, nhấp vào nút "Download Now" và làm theo các bước hướng dẫn để cài đặt phần mềm.
    - **Giới hạn của phiên bản dùng thử**: Phiên bản dùng thử cho phép quản lý tối đa hai mạng CAN và một mạng LIN, nhưng điều này đủ cho các mục đích học tập trong khóa học này.

2. Chuẩn bị Phần cứng và Phần mềm

- **Phần cứng cần thiết**: Khi làm việc cho một công ty, họ sẽ cung cấp cho bạn phiên bản phần mềm đầy đủ, không bị giới hạn, cùng với phần cứng Vector VN1640L và tất cả các dây dẫn cần thiết.
    - **Vector VN1640L**: Đây là phần cứng mà bạn sẽ kết nối với máy tính để giám sát các mạng CAN thực tế trên xe.
    - **Không cần tùy chọn đặc biệt**: Trong khóa học này, bạn không cần bất kỳ tùy chọn đặc biệt nào của Vector.

3. Mở và Cấu hình Phần mềm Vector CANoe

- **Mở phần mềm**: Sau khi cài đặt, bạn mở phần mềm CANoe.
    - **Cấu hình ban đầu**: Khi mở phần mềm lần đầu, bạn sẽ thấy một cấu hình trống không có nút nào bên trong. Bạn có thể lưu cấu hình này bằng cách nhấp vào "File" -> "Save As", đặt tên và nhấp "Save".
    - **Kết nối phần cứng**: Nếu phần cứng Vector VN1640L đã được kết nối với cổng USB của máy tính, bạn cần cấu hình các kênh phần cứng.
        - **Chọn phần cứng**: Nhấp vào cửa sổ "Hardware", chọn phần cứng Vector, và phần mềm sẽ nhận diện rằng VN1640L đã được kết nối qua cổng USB.
        - **Cấu hình kênh phần cứng**: Các kênh CAN1 và CAN2 là các kênh ứng dụng tương ứng với mỗi mạng CAN vật lý đã được kết nối.

4. Cấu hình Chi tiết của Kênh Phần cứng

- **Truy cập bảng điều khiển (Control Panel)**:
    - **Mở Control Panel**: Truy cập Control Panel của máy tính và nhấp vào "Vector Hardware".

    - **Chọn thiết bị**: Nhấp vào biểu tượng "+" cạnh VN1640L để mở rộng các tùy chọn.
    - **Liên kết kênh phần cứng và ứng dụng**:
        - **Kênh 2 của phần cứng**: Được liên kết với kênh ứng dụng CAN1 trong phần mềm CANoe.
        - **Kênh 3 của phần cứng**: Được liên kết với kênh ứng dụng CAN3 trong phần mềm CANoe.
        - **Kênh 4 của phần cứng**: Được liên kết với kênh ứng dụng CAN2 trong phần mềm CANoe.
- **Cấu hình các kênh ứng dụng**: Đối với bài học này, chúng ta sẽ chỉ sử dụng kênh ứng dụng CAN1 để giám sát một mạng CAN vật lý duy nhất. Sau này, khi cần giám sát đồng thời hai mạng CAN vật lý, chúng ta sẽ sử dụng thêm kênh ứng dụng CAN2.

5. Mã hóa các kênh phần cứng và ứng dụng

- **Pinout của đầu nối DB9**:
    - **Pin 2**: Kết nối với tín hiệu CAN Low.
    - **Pin 7**: Kết nối với tín hiệu CAN High.
    - **Pin 3**: Kết nối với ground (b-minus).
- **Bản đồ kênh**: Sau khi cấu hình các kênh phần cứng trong Control Panel, bạn phải thực hiện bản đồ kênh trong phần mềm CANoe, nơi bạn liên kết kênh phần cứng với kênh ứng dụng tương ứng.

6. Tạo Mạng CAN trong Phần mềm CANoe

- **Thêm Mạng CAN**:
    - **Thêm mới**: Nhấp chuột phải vào "CAN Networks", sau đó chọn "Add" và đặt tên cho mạng CAN mới.
    - **Các biểu tượng tự động xuất hiện**: Sau khi thêm, các biểu tượng như Nodes, Generators, Replay Blocks, Databases, và Application Channels sẽ tự động xuất hiện.
    - **Nodes**: Đây là các nút hoặc ECU, chẳng hạn như ECU phanh, ECU động cơ, hoặc ECU HDMI.
- **Liên kết Kênh Ứng dụng**: Kênh ứng dụng CAN1 được liên kết với kênh phần cứng tương ứng để giám sát các thông điệp truyền tải qua mạng CAN.

7. Kết luận và Phần tiếp theo của Khóa học

- **Kết luận**: Sau khi cài đặt và cấu hình phần mềm CANoe, bạn sẽ sẵn sàng để thiết lập và giám sát các mạng CAN.
- **Bài học tiếp theo**: Sẽ hướng dẫn cách thiết lập cơ sở dữ liệu mạng CAN và các tính năng liên quan.

Bài 9.

Hướng dẫn Cài đặt và Mô phỏng Mạng CAN/LIN với Phần mềm Vector CANoe

1. Giới thiệu về CANoe và Quá trình Tải xuống

- **CANoe là gì?**: CANoe là một công cụ chuyên nghiệp được sử dụng trong việc mô phỏng, phát triển, kiểm thử và phân tích các mạng ô tô, bao gồm các mạng CAN và LIN. CANoe được nhiều nhà sản xuất ô tô (OEM) trên toàn thế giới sử dụng để phát triển các mạng ECU hoàn chỉnh.
    - **CANdb++**: Đây là công cụ đi kèm với CANoe, dùng để tạo và quản lý cơ sở dữ liệu CAN, hỗ trợ việc phát triển các mạng CAN.
    - **Ngôn ngữ CAPL**: Hành vi vào/ra của các nút mạng được mô tả bằng ngôn ngữ CAPL, trong khi cấu trúc khung tin (frame) được xác định bằng CANdb++ cho các thông điệp CAN và bằng tập tin LDF cho các khung LIN.
- **Tải xuống CANoe**:
    - **Liên kết tải xuống**: Phiên bản demo của CANoe có thể được tải xuống miễn phí từ liên kết: https://vector.com/vi_downloadcenter_en.html.
    - **Lựa chọn sản phẩm**: Sau khi truy cập trang, dưới phần "Products and Topics", chọn "CANoe". Tiếp theo, dưới phần "Categories", chọn "Demos" và nhấp vào nút "Show results".
    - **Chọn phiên bản**: Một bảng hiển thị các sản phẩm có sẵn sẽ xuất hiện. Chọn phiên bản mới nhất và nhấp vào "Continue".
    - **Nhập thông tin cá nhân**: Nhập các thông tin cá nhân cần thiết và nhấp vào "Continue" để tiến hành tải xuống.

2. Quá trình Tải xuống và Cài đặt CANoe

- **Chọn sản phẩm và tải xuống**: Nhấp vào liên kết sản phẩm đã chọn (trong trường hợp này là CANoe 10.0 SP5) để bắt đầu tải xuống phần mềm CANoe.
    - **Yêu cầu hệ điều hành**: Lưu ý rằng CANoe chỉ hoạt động trên các phiên bản Windows 64-bit.
- **Bắt đầu cài đặt**: Sau khi tải xuống hoàn tất, nhấp đúp vào tập tin đã tải để bắt đầu cài đặt phiên bản demo của phần mềm.
    - **Đường dẫn trích xuất**: Khi được hỏi, bạn có thể để mặc định đường dẫn trích xuất và nhấp vào "Ok".
    - **Cập nhật thành phần hệ thống**: Quá trình cài đặt sẽ yêu cầu cập nhật một số thành phần hệ thống. Nhấp vào "Next >" để bắt đầu cài đặt các thành phần này (có thể mất một vài phút).

3. Lựa chọn Phiên bản Cài đặt và Tùy chọn Mạng

- **Cài đặt phiên bản demo**: Chọn cài đặt phiên bản demo, đây là phiên bản bạn có thể sử dụng mà không cần giấy phép từ Vector.
    - **Lựa chọn tùy chọn cài đặt**: Chọn cài đặt CANoe và ít nhất chọn các tùy chọn CAN và LIN. Bạn cũng có thể chọn tất cả các tùy chọn nếu muốn mô phỏng một số lượng lớn các giao thức mạng.
    - **Tên công ty**: Trường "company name" có thể để trống hoặc bạn có thể nhập tên công ty (ví dụ: "Politecnico di Torino").
- **Xác nhận cài đặt**: Nhấp vào "Next >" và xác nhận với "Yes" rằng bạn muốn cài đặt phiên bản demo. Nhấp lại vào "Next >" và quá trình cài đặt sẽ bắt đầu, cài đặt từng thành phần một.
- **Hoàn tất cài đặt**:
    - **Chọn cài đặt hoàn chỉnh**: Khi được hỏi, chọn tùy chọn "Complete" để cài đặt đầy đủ tất cả các thành phần cần thiết.
    - **Kết thúc cài đặt**: Sau khi quá trình cài đặt hoàn tất, nhấp vào "Finish".

4. Khởi chạy Phần mềm và Các Giới hạn của Phiên bản Demo

- **Khởi chạy CANoe**: Bạn có thể khởi chạy CANoe từ menu Start của Windows, tìm kiếm trong thư mục “Vector CANoe x.x (64 bit)”.
- **Giới hạn của phiên bản demo**:
    - **Số lượng nút mô phỏng**: Phiên bản demo chỉ giới hạn mô phỏng tối đa 4 nút.
    - **Không hỗ trợ phần cứng thực tế**: Phiên bản demo không cho phép sử dụng phần cứng thực tế.
    - **Simulink block set**: Khối Simulink để tích hợp với Matlab không được cài đặt, mặc dù tích hợp vẫn được hỗ trợ.
    - **LDF Explorer**: Công cụ chỉnh sửa tập tin mô tả LIN (LDF explorer) không được cài đặt, do đó bạn phải tự viết các tập tin LDF cần thiết.
    - **Giới hạn kiểm thử nút**: Các kiểm thử trên các nút bị giới hạn.
    - **Quản lý thời gian**: Tất cả quản lý thời gian trong phiên bản demo được thực hiện bằng các bộ hẹn giờ của Windows, có thể không phản ánh đúng thời gian thực.

Bài 10.

Hướng dẫn Bước đầu Thiết lập Mạng CAN trong Vector CANoe

1. Giới thiệu về Thiết lập Mạng CAN

- **Bài học trước**: Trong bài học trước, chúng ta đã tạo một mạng CAN mới.
- **Bài học hiện tại**: Bây giờ chúng ta sẽ thêm một nút (node) mới vào mạng CAN.

2. Cách Thêm Nút Mạng trong CANoe

- **Hai phương pháp để thêm nút**:
  - **Phương pháp 1**: Nhấp chuột phải vào "Nodes" dưới cây mạng CAN và chọn "Insert Network Node".
  - **Phương pháp 2 (Được khuyến nghị)**: Tạo nút trực tiếp từ cơ sở dữ liệu. Sau khi tạo cơ sở dữ liệu, nhấp chuột phải vào biểu tượng cơ sở dữ liệu dưới mạng CAN và chọn "Add".

3. Tạo Cơ sở Dữ liệu Mạng CAN

- **Công cụ CANdb++**: Để tạo cơ sở dữ liệu, truy cập cửa sổ "Tools" ở phần trên cùng của CANoe và chọn "CANdb++ Editor". Đây là một plugin của CANoe cho phép bạn tạo các thành phần cần thiết cho mạng CAN, bao gồm các nút, tín hiệu, thông điệp và thiết lập các thuộc tính liên quan.
- **Các bước tạo cơ sở dữ liệu**:
  - **Tạo cơ sở dữ liệu**: Nhấp vào "File" và chọn "Create Database". Chọn mẫu mặc định và đặt tên cho cơ sở dữ liệu, sau đó lưu vào thư mục trên máy tính của bạn.
  - **Cây cơ sở dữ liệu**: Cây cơ sở dữ liệu có nhiều trường như "ECUs", "Network Nodes", "Messages" và "Signals". Trong bài học này, chúng ta tập trung vào việc thêm nút mạng và thông điệp.

4. Thêm Nút Mạng Mới

- **Các bước thêm nút mạng**:
  - **Thêm nút mới**: Nhấp chuột phải vào "Network Nodes" và chọn "New". Một cửa sổ sẽ xuất hiện, bạn cần đặt tên cho nút, ví dụ: "Engine Control Node".
  - **Thiết lập ban đầu**: Để mặc định các trường địa chỉ (address fields) và các thông số khác.

- **Cửa sổ cấu hình**: Cửa sổ cấu hình của nút có các mục con như "Map Signals" (chứa các tín hiệu mà ECU cần truyền), "Map Receive Signals" (chứa các tín hiệu mà ECU cần nhận), và "Transmit Messages" (chứa các thông điệp mà ECU cần truyền).

5. Tạo và Liên kết Thông điệp với Nút Mạng

- **Tạo thông điệp mới**:
  - **Thêm thông điệp**: Nhấp chuột phải vào biểu tượng "Messages" và chọn "New". Đặt tên cho thông điệp, ví dụ: "Engine RPM".
  - **Thiết lập ID**: Đặt ID cho thông điệp, ví dụ: "0x25", và chọn kiểu chuẩn (standard), với độ dài ID là 11 bit.
- **Tạo tín hiệu**:
  - **Thêm tín hiệu**: Nhấp chuột phải vào biểu tượng "Signals" và chọn "New". Đặt tên cho tín hiệu, ví dụ: "RPM (round per minute)".
  - **Thiết lập thuộc tính**: Chọn độ dài tín hiệu là 1 byte (8 bit). Chọn thứ tự byte là Intel (byte ít quan trọng nhất đứng trước). Chọn kiểu giá trị là signed (cho phép giá trị âm). Đặt giá trị tối thiểu là 0 và giá trị tối đa là 6000, phù hợp với dải RPM của động cơ.
- **Liên kết tín hiệu với thông điệp**: Sau khi tạo tín hiệu, mở lại cửa sổ thông điệp "Engine RPM" và thêm tín hiệu vào thông điệp bằng cách nhấp vào "Add" và chọn tín hiệu.
- **Liên kết thông điệp với nút mạng**: Mở cửa sổ "Engine Control Node", chọn mục "Transmit Messages" và thêm thông điệp "Engine RPM" vào.

6. Tổng Kết và Các Bước Tiếp Theo

- **Tóm tắt**: Trong bài học này, chúng ta đã tạo một nút mạng, tạo thông điệp "Engine RPM", tạo tín hiệu "RPM", sau đó liên kết tín hiệu với thông điệp và thông điệp với nút mạng. Dù chỉ có một tín hiệu trong thông điệp, quy trình này vẫn có thể thực hiện một cách hiệu quả.
- **Bài học tiếp theo**: Chúng ta sẽ học cách thêm nhiều thông điệp và tín hiệu hơn vào các nút mạng.

Bài 11.

Thêm Thông điệp "Engine Status" và Tín hiệu "Ignition Status" trong Mạng CAN

1. Giới thiệu về Bài học và Nút Mạng

- **Bài học trước**: Trong bài học trước, chúng ta đã xác định và liên kết thông điệp "Engine RPM" với nút "Engine Control Node" trong cơ sở dữ liệu mạng.
- **Bài học hiện tại**: Bài học này sẽ tập trung vào việc thêm một thông điệp mới có tên "Engine Status" và tín hiệu "Ignition Status" vào mạng CAN.

2. Tạo Thông điệp "Engine Status"

- **Các bước tạo thông điệp**:
    - **Thêm thông điệp mới**: Nhấp chuột phải vào biểu tượng "Messages" trong cây mạng và chọn "New".
    - **Đặt tên thông điệp**: Đặt tên cho thông điệp là "Engine Status".
    - **Thiết lập ID**: Đặt ID của thông điệp là 0x20. ID này cao hơn so với ID của thông điệp "Engine RPM" (0x25), điều này có nghĩa là thông điệp "Engine Status" sẽ có độ ưu tiên cao hơn khi cả hai được truyền đồng thời từ nút "Engine Control Node".

3. Giải thích về Độ Ưu tiên trong Mạng CAN

- **Độ ưu tiên dựa trên ID**:
    - **Quy tắc ưu tiên**: Trong mạng CAN, thông điệp có ID thấp hơn sẽ có độ ưu tiên cao hơn trong quá trình truyền tải. Điều này rất quan trọng trong việc thiết kế mạng CAN, nơi các thông điệp quan trọng hơn cần được truyền tải nhanh hơn.
    - **Quyết định độ ưu tiên**: Độ ưu tiên của các thông điệp trong mạng CAN thường được quyết định cùng với khách hàng trong ngành công nghiệp ô tô mà bạn đang làm việc.

4. Tạo và Cấu hình Tín hiệu "Ignition Status"

- **Tạo tín hiệu mới**:

- **Thêm tín hiệu**: Nhấp chuột phải vào biểu tượng "Signals" trong cây mạng và chọn "New".
- **Đặt tên tín hiệu**: Đặt tên cho tín hiệu là "Ignition Status".
- **Thiết lập thuộc tính**:
  - **Độ dài**: Chọn độ dài tín hiệu là 1 bit, vì tín hiệu này có thể được coi là giá trị Boolean với hai trạng thái: "On" (1) hoặc "Off" (0).
  - **Đơn vị**: Có thể nhập "Boolean" trong trường đơn vị.
  - **Giá trị tối thiểu và tối đa**: Đặt giá trị tối thiểu là 0 và giá trị tối đa là 1, phù hợp với đặc tính của tín hiệu Boolean.
- **Hoàn tất tạo tín hiệu**: Sau khi hoàn thành các bước này, nhấp vào "OK" để hoàn tất tạo tín hiệu.

5. Liên kết Tín hiệu với Thông điệp

- **Liên kết tín hiệu "Ignition Status" với thông điệp "Engine Status"**:
  - **Mở cửa sổ thông điệp "Engine Status"**: Chuyển đến cửa sổ của thông điệp "Engine Status" và thêm tín hiệu "Ignition Status" vào bằng cách nhấp vào mục "Signals" và chọn tín hiệu, sau đó nhấp "OK".
  - **Xác nhận liên kết**: Kiểm tra để đảm bảo rằng tín hiệu "Ignition Status" đã được liên kết đúng cách với thông điệp "Engine Status".

6. Mục đích của Thông điệp và Tín hiệu

- **Mục tiêu truyền tải**: Hai thông điệp "Engine RPM" và "Engine Status" được xác định trong nút "Engine Control Node" nhằm gửi thông tin đến nút "HMI Cluster Node". Các thông tin này sẽ được hiển thị cho người lái trên cụm đồng hồ (HMI Cluster).

7. Kết luận và Bài học Tiếp theo

- **Tóm tắt**: Trong bài học này, chúng ta đã thêm thông điệp "Engine Status" và tín hiệu "Ignition Status" vào mạng CAN. Chúng ta cũng đã liên kết tín hiệu với thông điệp và chuẩn bị để gửi các thông tin này đến nút "HMI Cluster Node".
- **Bài học tiếp theo**: Sẽ tập trung vào việc định nghĩa nút mạng "HMI Cluster Node" và các bước tiếp theo trong việc thiết lập mạng CAN.

Bài 12.

Thêm Nút "HMI Cluster Node" và Tạo Thông điệp "Engine Status" và "Acknowledge Engine Status" trong Mạng CAN

1. Giới thiệu về Nút "HMI Cluster Node"

- **Mục đích của HMI Cluster Node**: Nút "HMI Cluster" (Human Machine Interface) được sử dụng để hiển thị thông tin động cơ, như số vòng quay động cơ (RPM) và trạng thái động cơ trên màn hình tương tác với người lái.
- **Tầm quan trọng của giao diện HMI**: HMI liên quan đến tất cả các tương tác mà người lái có thể thực hiện trên xe, bao gồm cả việc đọc các thông tin quan trọng như trạng thái động cơ và RPM.

2. Thêm Nút "HMI Cluster Node" vào Mạng

- **Các bước thêm nút**:
    - **Thêm nút mới**: Nhấp chuột phải vào biểu tượng "Network Nodes" và chọn "New".
    - **Đặt tên**: Đặt tên cho nút là "Cluster HMI Node".
    - **Thiết lập địa chỉ**: Để mặc định trường địa chỉ và nhấp "OK" để thêm nút vào cây mạng.

3. Tạo Thông điệp "Acknowledge Engine Status"

- **Mục đích của thông điệp**: Thông điệp "Acknowledge Engine Status" được sử dụng để gửi phản hồi từ nút "HMI Cluster" đến nút "Engine Control Node", xác nhận rằng thông điệp "Engine Status" đã được nhận đúng cách.
- **Các bước tạo thông điệp**:
    - **Thêm thông điệp mới**: Nhấp chuột phải vào biểu tượng "Messages" và chọn "New".
    - **Đặt tên**: Đặt tên cho thông điệp là "Acknowledge Engine Status".
    - **Thiết lập ID**: Đặt ID là 0x30. ID này cao hơn so với ID của thông điệp "Engine Status" (0x25), đảm bảo rằng thông điệp "Engine Status" sẽ được truyền trước.

4. Tạo và Liên kết Tín hiệu "Engine Message Received"

- **Tạo tín hiệu**:
    - **Thêm tín hiệu mới**: Nhấp chuột phải vào biểu tượng "Signals" và chọn "New".
    - **Đặt tên tín hiệu**: Đặt tên tín hiệu là "Engine Message Received".
    - **Thiết lập thuộc tính**:
        - **Giá trị tối thiểu và tối đa**: Đặt giá trị tối thiểu là 0 và tối đa là 1, tương tự như tín hiệu "Ignition Status".
        - **Kiểu giá trị**: Chọn "Unsigned" và đơn vị là "Boolean".
    - **Liên kết tín hiệu với thông điệp**: Mở cửa sổ thông điệp "Acknowledge Engine Status" và thêm tín hiệu này vào.

5. Liên kết Thông điệp với Nút "HMI Cluster Node"

- **Liên kết thông điệp với nút**: Mở cửa sổ nút "HMI Cluster Node", vào mục "Transmit Messages" và thêm thông điệp "Acknowledge Engine Status".
- **Xác nhận liên kết**: Kiểm tra lại để đảm bảo rằng thông điệp đã được liên kết đúng cách với nút "HMI Cluster Node".

6. Xác định Các Thông điệp mà Mỗi Nút Nhận

- **Quan trọng của việc nhận thông điệp**: Ngoài việc truyền thông điệp, mỗi nút trong mạng cũng cần biết các thông điệp mà nó phải nhận để hoạt động chính xác.
- **Các bước xác định thông điệp nhận**:
    - **Mở cửa sổ nút**: Mở cửa sổ của nút "HMI Cluster Node", vào mục "Map Receive Signals" và thêm các tín hiệu "Ignition Status" và "RPM" mà nút cần nhận từ "Engine Control Node".
    - **Xác nhận thông điệp**: Mở cửa sổ thông điệp "Engine RPM" và "Engine Status", kiểm tra mục "Receiver" để đảm bảo rằng các thông điệp này đã được liên kết với nút "HMI Cluster Node".

7. Kết luận và Bài học Tiếp theo

- **Tóm tắt**: Chúng ta đã hoàn thành việc thêm nút "HMI Cluster Node", tạo và liên kết các thông điệp "Engine Status" và "Acknowledge Engine Status", cùng với tín hiệu liên quan. Điều này đảm bảo rằng các thông tin quan trọng sẽ được hiển thị đúng cách trên giao diện HMI của xe.

- **Bài học tiếp theo**: Sẽ tập trung vào cấu hình chi tiết các trường của thông điệp trong cơ sở dữ liệu, chuẩn bị cho quá trình mô phỏng thực tế với phần cứng Vector.

Bài 13.

Thiết lập Thuộc tính Chu kỳ và Loại Truyền của Thông điệp trong CANoe

1. Giới thiệu về Thiết lập Thuộc tính Thông điệp

- **Mục tiêu của bài học**: Trong bài học này, chúng ta sẽ thiết lập các thuộc tính quan trọng cho các thông điệp đã tạo trong cơ sở dữ liệu mạng CAN, bao gồm chu kỳ thời gian (cycle time) và phương thức truyền (transmission method).
- **Các vấn đề gặp phải**: Một số thuộc tính như "transmission method" và "cycle time" trong cửa sổ định nghĩa của thông điệp hiện đang bị mờ và không thể chỉnh sửa trực tiếp. Để khắc phục, chúng ta sẽ tìm hiểu cách thiết lập các thuộc tính này thông qua việc sử dụng thuộc tính do người dùng định nghĩa (user-defined attributes).

2. Thiết lập Thuộc tính Chu kỳ Thời gian (Cycle Time)

- **Cách tiếp cận**: Để tìm giải pháp cho việc chỉnh sửa chu kỳ thời gian, ta sử dụng tài liệu hướng dẫn của CANoe và tìm thông tin liên quan đến "cycle time" thông qua tìm kiếm Google.
    - **Phát hiện trong tài liệu**: Giá trị của chu kỳ thời gian được liên kết với một thuộc tính gọi là "GenMsgCycleTime" trong thiết lập mặc định. Để chỉnh sửa giá trị này, ta cần định nghĩa thuộc tính do người dùng định nghĩa với tên tương ứng.
- **Tạo Thuộc tính "GenMsgCycleTime"**:
    - **Tạo thuộc tính**: Mở công cụ CANdb++ Editor, chọn "View" -> "Attribute Definitions", nhấp chuột phải vào cửa sổ trắng và chọn "New" để tạo thuộc tính mới.
    - **Thiết lập thuộc tính**:
        - **Loại đối tượng**: Chọn "Message".
        - **Loại giá trị**: Chọn "Integer".
        - **Tên thuộc tính**: Đặt tên thuộc tính là "GenMsgCycleTime".
        - **Giá trị mặc định**: Đặt giá trị mặc định là 100 milliseconds, đơn vị là milliseconds.
        - **Giá trị tối thiểu và tối đa**: Đặt giá trị tối thiểu là 0 và tối đa là 1000 milliseconds.
    - **Kiểm tra thuộc tính**: Đảm bảo rằng thuộc tính đã được định nghĩa đúng theo tài liệu hướng dẫn.

3. Thiết lập Thuộc tính Loại Truyền (Transmission Type)

- **Phát hiện trong tài liệu**: Ngoài thuộc tính chu kỳ thời gian, thuộc tính "GenMsgSendType" cũng cần được định nghĩa để chỉ định loại truyền của thông điệp (ví dụ: truyền theo chu kỳ hoặc truyền một lần).
- **Tạo Thuộc tính "GenMsgSendType"**:
    - **Tạo thuộc tính**: Tương tự như với "GenMsgCycleTime", nhưng với loại giá trị là "Enumeration".
    - **Thiết lập giá trị**:
        - **Giá trị đầu tiên**: Đặt giá trị đầu tiên là 0, tương ứng với kiểu truyền một lần (one-shot).
        - **Giá trị thứ hai**: Đặt giá trị thứ hai là 1, tương ứng với kiểu truyền theo chu kỳ (cyclic).
    - **Kiểm tra thuộc tính**: Đảm bảo rằng các giá trị đã được định nghĩa đúng theo tài liệu.

4. Áp dụng và Kiểm tra Thuộc tính

- **Áp dụng thuộc tính**: Sau khi định nghĩa các thuộc tính, áp dụng chúng vào các thông điệp đã tạo bằng cách mở cửa sổ thông điệp, vào mục "Attributes" và chỉnh sửa giá trị của "GenMsgCycleTime" và "GenMsgSendType".
- **Kiểm tra và Lưu cơ sở dữ liệu**: Sau khi áp dụng, lưu và kiểm tra cơ sở dữ liệu để đảm bảo các giá trị đã được cập nhật chính xác. Nếu giá trị không được cập nhật ngay lập tức, có thể cần phải đóng và mở lại cơ sở dữ liệu hoặc cấu hình CANoe.

5. Kết luận và Bài học Tiếp theo

- **Tóm tắt**: Trong bài học này, chúng ta đã thành công trong việc thiết lập và áp dụng các thuộc tính chu kỳ thời gian và loại truyền cho các thông điệp trong mạng CAN. Điều này đảm bảo rằng các thông điệp sẽ được truyền đúng cách trong quá trình mô phỏng.
- **Bài học tiếp theo**: Sẽ hướng dẫn cách nhập cơ sở dữ liệu đã tạo vào trong cấu hình mô phỏng của CANoe, chuẩn bị cho quá trình mô phỏng thực tế.

Bài 14.

Nhập Cơ sở Dữ liệu Mạng CAN và Thiết lập Cấu hình Mạng trong CANoe

1. Giới thiệu về Nhập Cơ sở Dữ liệu

- **Mục tiêu**: Sau khi hoàn thành việc tạo cơ sở dữ liệu cho mạng CAN (gọi là `database_one`), bước tiếp theo là nhập cơ sở dữ liệu này vào cấu hình mạng CAN trong phần mềm CANoe.
- **Phương pháp nhập cơ sở dữ liệu**: Có hai cách để nhập cơ sở dữ liệu vào mạng CAN trong CANoe:
    - **Cách 1**: Nhấp chuột phải vào biểu tượng "Databases" dưới cây "CAN Network One", chọn "Add" và chọn cơ sở dữ liệu đã tạo (`database_one`). Tuy nhiên, cách này không hiển thị các nút mạng trong cửa sổ bên trái.
    - **Cách 2 (Khuyến nghị)**: Sử dụng "Import Wizard" để nhập cơ sở dữ liệu cùng với các nút mạng.

2. Sử dụng "Import Wizard" để Nhập Cơ sở Dữ liệu

- **Các bước thực hiện**:
    - **Mở Import Wizard**: Nhấp chuột phải vào biểu tượng "Databases", chọn "Import Wizard".
    - **Chọn cơ sở dữ liệu**: Trong cửa sổ "Import Wizard", chọn cơ sở dữ liệu `database_one`.
    - **Chọn các nút mạng**: Đảm bảo chọn tất cả các nút mạng có sẵn trong cửa sổ bên trái và di chuyển chúng sang cửa sổ bên phải bằng cách nhấp vào mũi tên.
    - **Hoàn thành quá trình nhập**: Sau khi nhập, các nút mạng sẽ hiển thị trong cửa sổ "CAN Network One". Lưu lại cấu hình toàn bộ.

3. Tìm hiểu về Giao diện "Node Panel"

- **Truy cập Node Panel**: Nhấp vào biểu tượng Node Panel màu xanh để mở giao diện quản lý các nút mạng.
    - **Thông tin trong Node Panel**: Node Panel chứa tất cả các thông điệp (messages) và tín hiệu (signals) đã được xác định trong cơ sở dữ liệu.

    - **Ví dụ về "Engine Control Node"**: Tại nút "Engine Control Node", bạn sẽ thấy thông điệp "Engine Status" với tín hiệu "Ignition Status" và thông điệp "Engine RPM" với tín hiệu "RPM (round per minute)".
    - **Giá trị của tín hiệu**:
        - **Signal Name**: Tên của tín hiệu.
        - **Physical Value**: Giá trị thực tế của tín hiệu.
        - **Raw Value**: Giá trị nguyên của tín hiệu, được hiển thị dưới dạng thập phân.
        - **Symbolic Value**: Giá trị tượng trưng của tín hiệu.
- **Thiếu giá trị trong Symbolic Value**: Khi mở rộng "Symbolic Value Enumeration", không thấy giá trị nào hiển thị. Điều này yêu cầu một quá trình thiết lập trong các bài học tiếp theo để gán giá trị tượng trưng cho các tín hiệu.

4. Thiết lập và Gán Giá trị Tượng Trưng

- **Khó khăn trong việc nhớ giá trị vật lý**: Trong một mạng CAN thực tế với hàng ngàn tín hiệu, việc nhớ tất cả các giá trị vật lý là không khả thi. Ví dụ, giá trị vật lý của trạng thái "Engine On" có thể là 1, và "Engine Off" có thể là 0.
- **Giải pháp**: Bài học tiếp theo sẽ hướng dẫn cách gán giá trị tượng trưng cho các tín hiệu để dễ dàng quản lý và kiểm soát chúng trong quá trình mô phỏng.

5. Kết luận và Bài học Tiếp theo

- **Tóm tắt**: Trong bài học này, chúng ta đã nhập cơ sở dữ liệu `database_one` vào cấu hình CANoe và hiểu về giao diện Node Panel, nơi chứa các thông điệp và tín hiệu của mạng CAN. Đồng thời, chúng ta đã nhận ra sự cần thiết của việc gán giá trị tượng trưng cho các tín hiệu để dễ dàng quản lý trong mạng lớn.
- **Bài học tiếp theo**: Sẽ tập trung vào việc thiết lập và chạy mô phỏng thực tế bằng phần cứng Vector và cấu hình đã tạo trong CANoe.

Bài 15.

Khởi chạy Mô phỏng và Kiểm tra Kết nối Mạng CAN trong CANoe

1. Giới thiệu về Khởi chạy Mô phỏng

Trong bài học này, chúng ta sẽ thực hiện khởi chạy mô phỏng mạng CAN trong phần mềm CANoe, đồng thời kiểm tra kết nối phần cứng với mạng thực tế. Để bắt đầu, ta cần cắm phần cứng Vienne (VAN) vào cổng USB Type-C mà không kết nối bất kỳ đầu nối DB9 nào. Điều này có nghĩa là mạng CAN thực tế chưa được kết nối.

2. Khởi chạy Mô phỏng mà Không Kết Nối Mạng CAN Thực Tế

- **Bước đầu tiên**: Nhấn nút "Start Measurement" màu vàng ở góc trên bên trái của cửa sổ CANoe để bắt đầu mô phỏng.
- **Kiểm tra Trace**: Chuyển sang phần "Trace" ở góc dưới bên trái của cửa sổ CANoe và mở một phiên làm việc mới trong "Trace".
- **Kết quả khi không có DB9 kết nối**: Phần cứng Vienne cố gắng gửi một khung (frame) nhưng do không có kết nối DB9, khung này không được truyền trên kênh phần cứng. Điều này gây ra lỗi "Bit Error", cho thấy rằng thông điệp không thực sự xuất hiện trên mạng vì không có kết nối vật lý.

3. Kết nối Mạng CAN Thực Tế với Phần Cứng Vienne

- **Kết nối thực tế**: Cắm đầu nối DB9 vào kênh phần cứng Channel 2 và Channel 4 của Vienne, nhưng không cấp nguồn cho nút mạng thực tế để ít nhất Vienne có thể thấy điện trở 120 Ohm.
- **Kiểm tra Trace sau khi kết nối DB9**: Sau khi cắm DB9, lỗi "Bit Error" biến mất nhưng xuất hiện lỗi "Stuff Error" do nút mạng thực tế chưa được cấp nguồn. Điều này xảy ra khi kênh phần cứng đã được kết nối nhưng không có tín hiệu CAN hợp lệ do thiếu điện áp 12V cung cấp cho nút mạng.

4. Cấp Nguồn và Kiểm Tra Tin Nhắn CAN Thực Tế

- **Cấp nguồn cho nút mạng**: Bật nguồn cung cấp 12V cho nút mạng thực tế. Bây giờ, các tin nhắn CAN thực tế sẽ xuất hiện trên kênh "CAN 2 Application Channel".
- **Vấn đề về Bitrate**: Mặc dù kênh phần cứng Channel 2 đã kết nối, nhưng chỉ có tin nhắn từ kênh "CAN 2 Application Channel" được hiển thị vì có sự khác biệt về bitrate giữa cấu hình CANoe và mạng thực tế.
- **Điều chỉnh Bitrate**: Để đồng bộ hóa, cần điều chỉnh bitrate của kênh "CAN 1 Application Channel" trong phần cứng về giá trị phù hợp với mạng thực tế (125

kbps). Sau khi điều chỉnh, tin nhắn từ kênh "CAN 1 Application Channel" cũng sẽ được hiển thị mà không còn lỗi.

5. Kết luận và Bài học Tiếp theo

Bài học này đã hướng dẫn quy trình khởi chạy mô phỏng mạng CAN trong CANoe và cách kiểm tra kết nối vật lý với mạng thực tế. Chúng ta cũng đã xử lý các vấn đề liên quan đến lỗi bit và lỗi đồng bộ bitrate. Trong bài học tiếp theo, chúng ta sẽ khám phá chi tiết các trường trong Trace để hiểu rõ hơn về cách chúng hoạt động trong quá trình mô phỏng.

Bài 16.

Hiển thị và Tương tác với Tin nhắn trên Mạng CAN

1. Giới thiệu về Hiển thị Tin nhắn trên Mạng CAN

Trong bài học này, chúng ta đã xem cách các tin nhắn được truyền qua mạng CAN và nhận thấy rằng để mạng hoạt động, cần phải kết nối các nút mạng thực tế và cung cấp nguồn điện cho chúng. Khi điều này được thực hiện, chúng ta có thể thấy các tin nhắn của mình trong ứng dụng CANoe.

2. Hiển thị Tin nhắn từ Các Nút Mạng

- **Hiển thị Tin nhắn trong Trace**: Các tin nhắn từ nút mạng thực tế được hiển thị dưới dạng các giá trị số thập lục phân (hexadecimal numbers). Điều này xảy ra khi cơ sở dữ liệu chưa được định nghĩa đầy đủ cho các tin nhắn này, khiến chúng ta chỉ thấy các giá trị số mà không có thông tin chi tiết.
- **So sánh Tin nhắn từ Ứng dụng và Tin nhắn Thực tế**: Tin nhắn từ ứng dụng CANoe được định nghĩa và hiển thị rõ ràng với ID cụ thể và các trường dữ liệu, trong khi tin nhắn từ nút mạng thực tế chỉ hiển thị các giá trị số thập lục phân.

3. Các Trường Dữ liệu Quan trọng trong Trace

- **Time**: Trường này hiển thị thời gian khi mỗi tin nhắn được gửi. Nếu nguồn cung cấp bị tắt, thời gian sẽ dừng và lỗi sẽ xuất hiện. Khi nguồn được bật lại, tin nhắn sẽ tiếp tục được gửi trên mạng.
- **Event Type**: Thông thường, loại sự kiện (event type) là "Frame", biểu thị đây là khung tin nhắn. Nếu có lỗi, như khi mạng bị ngắt kết nối, lỗi sẽ được hiển thị thay vì khung tin nhắn.
- **Data Length Code (DLC)**: Mã độ dài dữ liệu này chỉ định số byte dữ liệu trong tin nhắn. Ví dụ, một tin nhắn có DLC là 1 byte sẽ chứa hai số thập lục phân (hexadecimal), mỗi số đại diện cho 4 bit.

4. Tương Tác với Tin nhắn trên Mạng

- **Thay đổi Giá trị Tin nhắn**: Có thể thay đổi giá trị của các tin nhắn trên mạng, chẳng hạn như bật/tắt động cơ bằng cách gửi giá trị vật lý 1 hoặc 0. Tuy nhiên, khi cố gắng thay đổi giá trị này trong bài học, thao tác không thành công, điều này sẽ được giải thích chi tiết trong bài học tiếp theo.

5. Kết luận và Bài học Tiếp theo

Bài học này đã hướng dẫn cách hiển thị và phân tích tin nhắn trên mạng CAN trong CANoe, đồng thời cố gắng tương tác và thay đổi các giá trị tin nhắn trên mạng. Trong bài học tiếp theo, chúng ta sẽ tìm hiểu lý do tại sao các thay đổi giá trị không hoạt động như mong đợi và cách khắc phục vấn đề này.

Bài 17.

Thiết lập và Tương tác với Tín hiệu trên Mạng CAN trong CANoe

1. Giới thiệu về Vấn đề Không Thể Thay Đổi Giá Trị Tín Hiệu

Trong bài học trước, chúng ta đã phát hiện rằng không thể thay đổi giá trị của các tín hiệu được định nghĩa từ lớp tương tác (interaction layer). Để giải quyết vấn đề này, chúng ta đã tìm thấy một hướng dẫn trên StackOverflow, trong đó chỉ ra rằng để lớp tương tác hoạt động hiệu quả, cần phải thiết lập các thuộc tính cần thiết cho mỗi tin nhắn được định nghĩa trong cơ sở dữ liệu CANoe.

2. Thiết lập Thuộc tính cho Các Tin Nhắn

- **Thiết lập các thuộc tính**: Để lớp tương tác hoạt động, cần phải thiết lập đúng các thuộc tính trong bảng thuộc tính, bao gồm tên, loại đối tượng, loại giá trị, giá trị mặc định và phạm vi giá trị. Đặc biệt, các thuộc tính dưới phần "Interaction Layer" là quan trọng nhất để lớp tương tác hoạt động.
- **Chỉnh sửa các thuộc tính của tin nhắn**: Sau khi thiết lập các thuộc tính, bạn cần điều chỉnh chúng trong phần "Attributes" của mỗi cửa sổ tin nhắn. Ví dụ, thuộc tính "Gen Sky L Support" cần được đặt là "Yes" để lớp tương tác có thể hỗ trợ thay đổi giá trị tín hiệu.

3. Kiểm Tra và Chỉnh Sửa Giá Trị Tín Hiệu

- **Thay đổi giá trị tín hiệu**: Khi thử thay đổi giá trị của tín hiệu "RPM round per min" trong lớp tương tác, giá trị đã được cập nhật chính xác trong phần "CAN Trace". Tuy nhiên, đối với tín hiệu "Ignition Status", thay đổi này không thành công, và giá trị tín hiệu vẫn giữ nguyên.
- **Vấn đề về độ dài dữ liệu**: Nếu cố gắng đặt giá trị lớn hơn 255 cho tín hiệu "RPM round per min", giá trị này không thể được gửi vì độ dài tín hiệu chỉ là 1 byte (8 bit), giới hạn giá trị ở 255. Để giải quyết, bạn có thể tăng độ dài của tín hiệu hoặc sử dụng hệ số nhân (scaling factor).

4. Cập Nhật Chu Kỳ Gửi Tin Nhắn

- **Chu kỳ gửi tin nhắn**: Một vấn đề khác là tin nhắn "Engine Status" chỉ được gửi một lần khi khởi động mô phỏng, mặc dù nó được định nghĩa là tin nhắn tuần hoàn. Nguyên nhân là thuộc tính "Gen MSG Send Type" của tin nhắn này không được đặt là

"Cyclic". Khi thuộc tính này được chỉnh sửa và đặt lại, tin nhắn đã được gửi đúng chu kỳ.

- **Khởi động lại mô phỏng**: Sau mỗi lần thay đổi thuộc tính hoặc cấu hình cơ sở dữ liệu, cần phải khởi động lại mô phỏng và cơ sở dữ liệu để các thay đổi được áp dụng chính xác.

5. Kết luận và Bài học Tiếp theo

Trong bài học này, chúng ta đã giải quyết vấn đề không thể thay đổi giá trị tín hiệu trong lớp tương tác bằng cách thiết lập đúng các thuộc tính cần thiết. Chúng ta cũng đã tìm hiểu cách cấu hình chu kỳ gửi tin nhắn đúng cách và tầm quan trọng của việc khởi động lại mô phỏng sau khi thực hiện các thay đổi trong cơ sở dữ liệu. Bài học tiếp theo sẽ tiếp tục khám phá các vấn đề tương tác và cấu hình chi tiết hơn trong môi trường CANoe.

Bài 18.

Bảng Giá Trị Tín Hiệu và Mô phỏng Giá Trị Tượng Trưng trong CANoe

1. Giới thiệu về Mô phỏng Giá Trị Tượng Trưng

Trong bài học này, chúng ta tập trung vào việc liên kết các giá trị thô của tín hiệu với các ý nghĩa cụ thể trong cơ sở dữ liệu. Khi đọc một tập hợp các tin nhắn CAN trong phần "CAN Trace", chúng ta cần hiểu được ý nghĩa của mỗi giá trị tín hiệu trong các tin nhắn đó.

2. Tạo Bảng Giá Trị cho Tín Hiệu

- **Tạo Bảng Giá Trị**: Để gán ý nghĩa cho các giá trị của tín hiệu, chúng ta tạo một bảng giá trị (Value Table) trong cơ sở dữ liệu bằng cách nhấn vào "View" và sau đó chọn "Value Tables".
- **Ví dụ về Tạo Bảng Giá Trị**: Đầu tiên, chúng ta tạo bảng giá trị cho tín hiệu "Ignition Status" và đặt tên là "Ignition Status". Sau đó, thêm hai dòng bằng cách nhấn "Add" và gán mô tả ý nghĩa cho từng giá trị. Ví dụ: "Engine is off" cho giá trị 0 và "Engine is on" cho giá trị 1.

3. Liên Kết Bảng Giá Trị với Tín Hiệu

- **Liên kết bảng giá trị**: Sau khi tạo bảng giá trị, bạn có thể liên kết nó với tín hiệu bằng cách chuyển đến phần định nghĩa tín hiệu và chọn bảng giá trị đã tạo trong mục "Value Table". Sau đó, nhấn "OK" để xác nhận.
- **Áp dụng bảng giá trị cho nhiều tín hiệu**: Bảng giá trị vừa tạo có thể được áp dụng cho nhiều tín hiệu nếu các tín hiệu đó có ý nghĩa mô tả tương tự. Ví dụ, tín hiệu "Engine Message Received" cũng có thể sử dụng bảng giá trị của "Ignition Status" vì chúng có mô tả tương tự.

4. Kiểm Tra và Mô phỏng Giá Trị Tượng Trưng

- **Kiểm tra mô phỏng**: Sau khi liên kết bảng giá trị với tín hiệu, bạn cần khởi động lại mô phỏng trong CANoe và kiểm tra kết quả trên "CAN Trace". Lúc này, các mô tả giá trị sẽ được hiển thị bên cạnh tín hiệu tương ứng.
- **Cập nhật tín hiệu**: Trong quá trình mô phỏng, bạn có thể thay đổi giá trị của tín hiệu và kiểm tra kết quả trong "CAN Trace". Khi thay đổi tín hiệu "Engine Message Received", giá trị tín hiệu và mô tả sẽ được cập nhật chính xác.

5. Ý Nghĩa của Bảng Giá Trị Tượng Trưng

- **Tầm quan trọng của giá trị tượng trưng**: Liên kết giá trị tượng trưng rất quan trọng đối với kỹ sư thiết kế mạng CAN, vì đây là một dạng từ điển chung giữa nhà cung cấp và khách hàng. Điều này giúp cả hai bên hiểu rõ ý nghĩa của các tín hiệu trong mạng CAN.
- **Dễ dàng thay đổi**: Khi khách hàng yêu cầu thay đổi ý nghĩa của tín hiệu, bạn chỉ cần thay đổi mô tả trong bảng giá trị mà không cần thay đổi toàn bộ cơ sở dữ liệu, giúp tiết kiệm thời gian và đảm bảo tính nhất quán.

6. Kết luận và Bài học Tiếp theo

Bài học này đã giúp chúng ta hiểu cách liên kết các giá trị tín hiệu với ý nghĩa cụ thể thông qua bảng giá trị và cách kiểm tra mô phỏng trong CANoe. Trong bài học tiếp theo, chúng ta sẽ bắt đầu tìm hiểu về ngôn ngữ C trong bối cảnh ứng dụng CAN.

Bài 19.

Ngôn ngữ CAPL trong CANoe

1. Tổng quan về Ngôn ngữ CAPL

Ngôn ngữ CAPL (CAN Access Programming Language) là một ngôn ngữ lập trình cấp cao được sử dụng trong CANoe, một công cụ mô phỏng và phát triển mạng CAN/LIN phổ biến trong ngành công nghiệp ô tô. CAPL được yêu cầu rộng rãi trong các công việc liên quan đến kỹ thuật ô tô, đặc biệt là khi kết hợp với phần mềm CANoe của Vector.

CAPL là một dạng mã giả (pseudo-code) nhưng dễ sử dụng và dễ hiểu hơn ngôn ngữ C. Các hàm trong CAPL được thiết kế sẵn, dễ dàng sử dụng và thân thiện với người dùng, giúp đơn giản hóa quá trình lập trình và tương tác với mạng CAN.

2. Cấu trúc và Thành phần của CAPL Editor

Mỗi node (nút) được định nghĩa trong mạng CAN đều có một CAPL Editor liên kết với nó. Để mở CAPL Editor, bạn cần nhấp vào biểu tượng bút chì ở góc dưới bên trái của hình chữ nhật đại diện cho node.

3. Các phần chính trong CAPL Editor:

- **Phần biến (Variables Section):** Tại đây, bạn có thể định nghĩa các biến cần thiết cho logic của mã CAPL.
- **Phần thư viện (Include Section):** Cho phép bạn nhập các thư viện bên ngoài, ví dụ như các thư viện DLL để tự động hóa các tác vụ. Các thư viện này cung cấp khả năng tương tác toàn diện với mạng CAN, bao gồm việc gửi và nhận tin nhắn, thay đổi giá trị của tín hiệu, v.v.
- **Các hàm hệ thống (System Functions):** Các hàm này giúp bạn tương tác với mạng CAN.
- **Các đối tượng giá trị (Value Objects):** Các đối tượng này tương ứng với các giá trị trong mạng CAN, chẳng hạn như tin nhắn, và chúng đã được giới thiệu trong phần định nghĩa thuộc tính của cơ sở dữ liệu.
- **Hàm Callback (Callback Functions):** Đây là các hàm được gọi tự động khi một sự kiện cụ thể xảy ra, ví dụ như khi một tin nhắn CAN được nhận hoặc khi một tín hiệu thay đổi.

4. Mục tiêu và Ứng dụng của CAPL trong CANoe

Mục tiêu chính của ứng dụng CANoe là cho phép kỹ sư kiểm tra và mô phỏng mạng CAN một cách toàn diện. Bạn cần có khả năng mô phỏng tin nhắn đến một ECU nhất định khi ECU này không có sẵn trên xe thực tế. Điều này đặc biệt quan trọng trong giai đoạn sản xuất trước hoặc khi bạn muốn ngắt kết nối một ECU thực và mô phỏng tất cả các tin nhắn mà các nút khác trong mạng mong đợi.

5. Loại bỏ và Mô phỏng các Node Thực tế

Trong một số trường hợp, khi một nút thực đã có sẵn trên xe, bạn cần loại bỏ nút đó khỏi mô phỏng để tránh xung đột giữa các tin nhắn. Để loại bỏ một nút khỏi mô phỏng, bạn chỉ cần nhấp vào nút đó và nhấn phím cách (spacebar) trên bàn phím. Nút sẽ chuyển sang màu xám nhạt, cho biết rằng nút thực đang hoạt động và không được mô phỏng.

Ngược lại, nếu hình chữ nhật của node chuyển sang màu xám đậm, điều đó có nghĩa là node đang được mô phỏng bởi phần mềm CANoe.

6. Ví dụ Thực tế về Sử dụng CAPL

Giả sử bạn có một băng thử với một nút điều khiển động cơ thực tế (ECU) và một nút cụm HMI (Human Machine Interface) không có sẵn. Khi nút điều khiển động cơ gửi một tin nhắn trạng thái động cơ, nó mong đợi một phản hồi từ nút HMI. Tuy nhiên, vì HMI thực tế không có sẵn, phần mềm CANoe sẽ mô phỏng phản hồi này bằng cách sử dụng mã CAPL.

Nhờ đó, nút điều khiển động cơ thực sự "nghĩ" rằng nút HMI đang hoạt động và tiếp tục gửi tin nhắn bình thường, không nhận thấy sự thiếu vắng của HMI trên băng thử.

7. Kết luận và Bài học Tiếp theo

Trong bài học này, chúng ta đã tìm hiểu về CAPL, cách mở và sử dụng CAPL Editor, cũng như cách mô phỏng các tín hiệu và tin nhắn trên mạng CAN. Trong bài học tiếp theo, chúng ta sẽ đi sâu hơn vào các script ngôn ngữ CAPL, hiểu rõ cách chúng được sử dụng để mô phỏng và kiểm tra mạng CAN trong CANoe.

Bài 20.

Sử dụng ngôn ngữ CAPL trong CANoe

1. Giới thiệu về CAPL Editor và các phần chính

Để bắt đầu sử dụng CAPL Editor, bạn cần chuyển đến cửa sổ mô phỏng CANoe và nhấp vào biểu tượng bút chì trên hình chữ nhật đại diện cho node (nút) mà bạn muốn chỉnh sửa. Trong CAPL Editor, có một số phần chính bạn cần biết:

- **Include Section:** Đây là phần bạn có thể nhập các thư viện bên ngoài nếu cần, mặc dù trong phạm vi khóa học này, chúng ta sẽ không sử dụng phần này.
- **Variable Section:** Phần này cho phép bạn định nghĩa các biến cần thiết cho logic của script. Các biến này có thể là biến tin nhắn, biến bộ đếm thời gian hoặc biến số nguyên.

2. Ví dụ thực tế: Logic của tin nhắn xác nhận trạng thái động cơ

Trong ví dụ này, chúng ta sẽ viết một script để xử lý logic của tin nhắn "Acknowledge Engine Status" (xác nhận trạng thái động cơ) được gửi từ node HMI Cluster sau khi node điều khiển động cơ gửi tin nhắn "Engine Status" trên mạng CAN.

3. Phân tích mã CAPL

Đầu tiên, chúng ta sẽ định nghĩa hai biến kiểu tin nhắn, bao gồm:

- `AcknowledgeEngineStatus`: Biến này đại diện cho tin nhắn "Acknowledge Engine Status".
- `EngineStatus`: Biến này đại diện cho tin nhắn "Engine Status" đến từ node điều khiển động cơ.

```
message AcknowledgeEngineStatus; // Định nghĩa biến tin nhắn Acknowledge Engine Status

message EngineStatus; // Định nghĩa biến tin nhắn Engine Status
```

Tiếp theo, chúng ta sử dụng hàm `on message` để phát hiện khi tin nhắn "Engine Status" được gửi trên mạng CAN. Khi tin nhắn này được phát hiện, chúng ta sẽ kiểm tra giá trị của tín hiệu "Ignition Status" bên trong tin nhắn.

```
on message EngineStatus {
    if (this.IgnitionStatus == 0) {

setSignal(AcknowledgeEngineStatus.EngineMessageReceived, 0);
// Đặt tín hiệu EngineMessageReceived bằng 0
    } else {

setSignal(AcknowledgeEngineStatus.EngineMessageReceived, 1);
// Đặt tín hiệu EngineMessageReceived bằng 1
    }
}
```

- **`this.IgnitionStatus == 0`**: Kiểm tra xem tín hiệu "Ignition Status" có bằng 0 hay không. Nếu có, tín hiệu "Engine Message Received" sẽ được đặt bằng 0.
- **`setSignal`**: Hàm này được sử dụng để đặt giá trị cho tín hiệu "Engine Message Received" trong tin nhắn "Acknowledge Engine Status".

4. Kiểm tra Script và Kết quả

Sau khi viết script, bạn cần lưu các thay đổi và bắt đầu mô phỏng trong CANoe. Khi bạn thay đổi giá trị của tín hiệu "Ignition Status", giá trị của tín hiệu "Engine Message Received" sẽ tự động cập nhật theo, chứng tỏ script của bạn hoạt động đúng.

5. Ứng dụng Thực tế

Trong tình huống thực tế, tín hiệu "Ignition Status" sẽ được gửi từ một node ECU thực tế trong mạng CAN. Tuy nhiên, trong ví dụ này, chúng ta đã kích hoạt nó thủ công từ bảng tương tác (Interaction Panel). Phương pháp này cũng sẽ hoạt động nếu node điều khiển động cơ là một node thật trên băng thử (test bench).

6. Kết luận

Bài học này đã giới thiệu cách viết một script đơn giản bằng ngôn ngữ CAPL để xử lý logic của tin nhắn trên mạng CAN. Chúng ta đã thấy cách xác định các biến, sử dụng các hàm callback để phát hiện tin nhắn, và cách đặt giá trị cho các tín hiệu trong tin nhắn. Trong bài

học tiếp theo, chúng ta sẽ tiếp tục với các chức năng CAPL khác để xử lý các tình huống phức tạp hơn.

Bài 21.

Sử dụng sự kiện `setTimer` trong ngôn ngữ CAPL

1. Giới thiệu về việc thay đổi giá trị tín hiệu theo chu kỳ

Trong bài học này, chúng ta sẽ học cách thay đổi giá trị của một tín hiệu CAN theo một chu kỳ định sẵn bằng cách sử dụng sự kiện `setTimer` trong ngôn ngữ CAPL. Đặc biệt, chúng ta sẽ tập trung vào tín hiệu "Engine Message Received" trong tin nhắn "Acknowledge Engine Status" từ node HMI Cluster.

2. Cập nhật mã CAPL cho tín hiệu chu kỳ

Đầu tiên, chúng ta cần mở CAPL Editor và chỉnh sửa mã CAPL mà chúng ta đã viết trong bài học trước. Phần đầu tiên của điều kiện `if` sẽ giữ nguyên, vì khi tín hiệu "Ignition Status" bằng 0, tín hiệu "Engine Message Received" phải luôn bằng 0.

```
if (this.IgnitionStatus == 0) {
    setSignal(AcknowledgeEngineStatus.EngineMessageReceived, 0);
}
```

Tuy nhiên, khi tín hiệu "Ignition Status" bằng 1, chúng ta sẽ cần thay đổi tín hiệu "Engine Message Received" bằng 1 trong một khoảng thời gian nhất định, sau đó nó sẽ trở về 0, ngay cả khi tín hiệu "Ignition Status" vẫn bằng 1.

```
else {
    setTimer(timerPeriod, 1000); // Bắt đầu đếm ngược thời gian 1000ms
    setSignal(AcknowledgeEngineStatus.EngineMessageReceived, 1); // Đặt tín hiệu bằng 1 trong khoảng thời gian nhất định
}
```

3. Định nghĩa biến Timer và sử dụng sự kiện `on timer`

Chúng ta cần định nghĩa một biến `timer` để sử dụng trong sự kiện `setTimer`. Sau khi định nghĩa biến này, chúng ta sẽ sử dụng nó trong hàm `setTimer` để đặt giá trị cho tín hiệu sau một khoảng thời gian xác định.

```
variables {
    timer timerPeriod; // Định nghĩa biến Timer
}

on timer timerPeriod {
    setSignal(AcknowledgeEngineStatus.EngineMessageReceived,
0); // Khi thời gian đếm ngược kết thúc, đặt tín hiệu trở về 0
}
```

4. Thực hiện kiểm tra và mô phỏng

Sau khi viết mã, hãy lưu các thay đổi và chuyển đến cửa sổ mô phỏng CANoe. Để theo dõi tín hiệu theo thời gian, chúng ta sẽ sử dụng cửa sổ Graphics trong CANoe, nơi hiển thị sự thay đổi của các tín hiệu trong thời gian thực.

- **Thêm tín hiệu vào cửa sổ Graphics**: Chúng ta thêm các tín hiệu "Ignition Status" và "Engine Message Received" vào cửa sổ này.
- **Cấu hình cột và chế độ hiển thị**: Cấu hình để hiển thị giá trị thô và giá trị tương ứng của tín hiệu dưới dạng đồ thị thời gian thực.

5. Kiểm tra mã trong mô phỏng

Sau khi cấu hình xong, khởi động mô phỏng và thay đổi tín hiệu "Ignition Status" sang giá trị "Engine On". Quan sát cửa sổ đồ thị, bạn sẽ thấy tín hiệu "Engine Message Received" ban đầu sẽ chuyển sang giá trị "Engine On" nhưng không trở về "Engine Off" sau khoảng thời gian 1 giây như dự kiến.

6. Kết luận và chuẩn bị cho bài học tiếp theo

Kết quả thu được từ bài học này là chúng ta đã viết mã CAPL để thay đổi giá trị tín hiệu theo một chu kỳ định sẵn, nhưng có một lỗi cần khắc phục khi tín hiệu không trở về giá trị mong muốn. Trong bài học tiếp theo, chúng ta sẽ tìm hiểu cách giải quyết vấn đề này và làm rõ thêm về các tính năng của CAPL để hiểu rõ hơn về cách hoạt động của các sự kiện callback trong ngôn ngữ lập trình C.

Bài 22.

Hàm `on timer` trong CAPL

1. Vấn đề trong mã CAPL hiện tại

Trong bài học trước, chúng ta đã gặp vấn đề với đoạn mã CAPL khi cố gắng đặt tín hiệu "Engine Message Received" thành giá trị 1 chỉ trong 1 giây. Tuy nhiên, đoạn mã này không hoạt động như mong đợi vì sự kiện `on message` của tín hiệu "Engine Status" được gọi lại mỗi 200ms (theo chu kỳ gửi tin nhắn). Điều này dẫn đến việc hàm `setTimer` bị khởi động lại liên tục trước khi bộ đếm thời gian hiện tại kết thúc, do đó ngăn cản sự kiện `on timer` được kích hoạt đúng cách.

2. Hoạt động của sự kiện `on message` và `on timer`

Mỗi khi tin nhắn "Engine Status" được gửi trên mạng, sự kiện `on message` tương ứng sẽ được gọi. Điều này xảy ra cứ mỗi 200ms một lần, theo chu kỳ gửi của tin nhắn đó. Nếu tín hiệu "Ignition Status" có giá trị là 1, hàm `setTimer` sẽ được kích hoạt, và bộ đếm thời gian sẽ bắt đầu đếm ngược từ 1000ms.

Tuy nhiên, vì sự kiện `on message` được gọi lại sau mỗi 200ms, nó sẽ liên tục đặt lại và khởi động lại bộ đếm thời gian trước khi bộ đếm này kịp hết hạn. Điều này dẫn đến việc hàm `on timer` không bao giờ được kích hoạt đúng cách.

3. Hiểu rõ hơn về cách hoạt động của các sự kiện trong CAPL

- **Sự kiện `on message`**: Được kích hoạt mỗi khi một tin nhắn CAN được phát hiện trên mạng. Trong trường hợp của chúng ta, tin nhắn "Engine Status" có chu kỳ 200ms, do đó sự kiện này sẽ được gọi lại mỗi 200ms.
- **Sự kiện `on timer`**: Được kích hoạt sau khi bộ đếm thời gian hết hạn. Tuy nhiên, nếu bộ đếm thời gian bị khởi động lại trước khi hết hạn, sự kiện này sẽ không được kích hoạt.

4. Kết luận

Trong bài học này, chúng ta đã hiểu rõ hơn về lý do tại sao đoạn mã CAPL hiện tại không hoạt động như mong muốn. Vấn đề chính là do sự kiện `on message` được gọi quá thường xuyên, dẫn đến việc bộ đếm thời gian bị đặt lại liên tục trước khi nó kịp hết hạn, do đó sự kiện `on timer` không được kích hoạt.

5. Hướng dẫn tiếp theo

Trong bài học tiếp theo, chúng ta sẽ học cách viết lại đoạn mã này để giải quyết vấn đề đã nêu, đảm bảo rằng tín hiệu "Engine Message Received" được đặt giá trị 1 chỉ trong 1 giây như yêu cầu.

Bài 23.

Sử Dụng Sự Kiện `on timer` và `on key` Để Điều Khiển Tín Hiệu CAN

1. Mục Tiêu Của Bài Học Trong bài học này, chúng ta sẽ học cách sử dụng đúng sự kiện `on timer` để gửi một tín hiệu CAN với giá trị cụ thể trong một khoảng thời gian nhất định. Cụ thể, chúng ta sẽ đặt tín hiệu `engine_message_received` bằng 1 trong một giây, sau đó đặt lại tín hiệu này về 0. Hoạt động này sẽ được kích hoạt mỗi khi nhấn một phím trên bàn phím.

2. Thiết Lập Sự Kiện `on key`

- Chúng ta bắt đầu bằng cách thêm sự kiện `on key` vào mã CAPL. Sự kiện này sẽ được kích hoạt mỗi khi nhấn một phím trên bàn phím.
- Khi phím được nhấn, đầu tiên chúng ta đặt tín hiệu `engine_message_received` bằng 1.
- Đồng thời, chúng ta sẽ khởi động bộ đếm thời gian bằng cách sử dụng hàm `setTimer`, trong đó chỉ định tên của bộ đếm thời gian và khoảng thời gian (ở đây là 1 giây).

3. Thiết Lập Sự Kiện `on timer`

- Khi bộ đếm thời gian hết hạn, sự kiện `on timer` sẽ được gọi. Trong sự kiện này, chúng ta sẽ đặt lại tín hiệu `engine_message_received` về 0.

4. Thực Thi Mã CAPL

- Sau khi mã CAPL được thiết lập, chúng ta chuyển sang cửa sổ mô phỏng CANoe và bắt đầu mô phỏng.
- Khi nhấn một phím bất kỳ trên bàn phím, tín hiệu `engine_message_received` sẽ được đặt bằng 1 trong 1 giây và sau đó tự động trở về 0 sau khi bộ đếm thời gian hết hạn.

5. Kết Quả Quan Sát

- Khi tín hiệu `engine_message_received` được kích hoạt và duy trì ở mức cao trong 1 giây, tín hiệu sẽ được hiển thị trong cửa sổ đồ thị (graphics window) dưới dạng một xung vuông, có giá trị cao trong 1 giây và sau đó trở về giá trị ban đầu.

- Điều này đảm bảo rằng tín hiệu được kích hoạt và tắt đúng theo yêu cầu, không bị ảnh hưởng bởi các sự kiện khác như trong mã CAPL trước đây.

6. Điều Chỉnh Thời Gian

- Nếu muốn thay đổi thời gian tín hiệu duy trì ở mức cao từ 1 giây thành 2 giây, chỉ cần thay đổi giá trị trong hàm `setTimer` từ 1000 milliseconds thành 2000 milliseconds. Khi thực hiện thay đổi này, bạn sẽ thấy tín hiệu duy trì mức cao trong 2 giây trong cửa sổ đồ thị.

7. Kết Luận

- Qua bài học này, chúng ta đã học cách sử dụng sự kiện `on timer` và `on key` để điều khiển tín hiệu CAN. Phương pháp này giúp tạo ra một xung vuông cho tín hiệu một cách chính xác và dễ dàng hơn rất nhiều so với việc thực hiện thủ công thông qua giao diện tương tác.
- Điều này đặc biệt quan trọng trong các ứng dụng thực tế, nơi mà tín hiệu phải tuân thủ chính xác thời gian để tránh phát sinh lỗi trong hệ thống điều khiển.

8. Hướng Dẫn Tiếp Theo

- Trong bài học tiếp theo, chúng ta sẽ học cách lập trình trực quan, cho phép thay đổi giá trị tín hiệu CAN thông qua việc nhấn một nút chuyển đổi (switch key).

Bài 24.

Lập Trình Trực Quan Trong CANoe

1. Mục Tiêu Dự Án Nhỏ Trong bài học này, chúng ta sẽ bắt đầu với một dự án nhỏ liên quan đến lập trình trực quan trong CANoe. Mục tiêu của dự án là tạo ra một giao diện điều khiển đơn giản, trong đó có các thành phần như công tắc khởi động động cơ, bàn đạp phanh, bàn đạp ga và đồng hồ tốc độ hiển thị tốc độ động cơ (RPM).

2. Mô Tả Giao Diện

- **Phần Phải**: Đặt công tắc khởi động động cơ.
- **Phần Trái**: Đặt bàn đạp phanh và bàn đạp ga.
- **Phần Dưới**: Đặt đồng hồ tốc độ (speedometer) hiển thị tốc độ động cơ dưới dạng RPM.

3. Kịch Bản Ứng Dụng

- Trong bối cảnh này, chúng ta giả lập môi trường thử nghiệm khi chỉ có cụm đồng hồ thực (cluster ECU) và phải giả lập node điều khiển động cơ (engine control node).
- Khi nhấn bàn đạp ga, tốc độ vòng quay của động cơ (RPM) sẽ tăng lên, và khi nhấn bàn đạp phanh, tốc độ vòng quay sẽ giảm xuống, mô phỏng hoạt động của một chiếc xe thực tế.
- Cụm đồng hồ thực sẽ tính toán và hiển thị tốc độ xe dựa trên RPM của động cơ, tỷ lệ truyền của hộp số và bán kính bánh xe.

4. Khung Dự Án Ban Đầu

- Trong phần này, ta sẽ khởi động mô phỏng và nhấn bàn đạp ga.
- Tuy nhiên, không có gì xảy ra do mã CAPL được viết trong cluster ECU, nhưng cluster này đã bị tắt, dẫn đến việc mã CAPL không được thực thi.
- Để khắc phục, chúng ta cần chuyển toàn bộ mã CAPL sang engine control node, node đã được kích hoạt trong mô phỏng.

5. Chuyển Mã CAPL Sang Engine Control Node

- Mở trình chỉnh sửa CAPL (CAPL editor) của engine control node bằng cách nhấp vào biểu tượng bút chì trên node này.
- Tạo một instance mới cho mã CAPL và đặt tên cho mã này.

- Sao chép và dán mã đã viết vào CAPL editor liên kết với engine control node, sau đó biên dịch mã.

6. Kết Quả Mô Phỏng

- Sau khi sửa mã và chạy lại mô phỏng, khi nhấn bàn đạp ga, giá trị RPM sẽ tăng lên và khi nhấn bàn đạp phanh, giá trị RPM sẽ giảm xuống, tương ứng với hoạt động của xe trong thực tế.

7. Kết Luận

- Bài học này chủ yếu tập trung vào việc thiết lập cơ bản cho một dự án nhỏ liên quan đến lập trình trực quan trong CANoe.
- Trong bài học tiếp theo, chúng ta sẽ đi sâu vào việc viết mã CAPL để thực hiện các thao tác này một cách chi tiết hơn.

8. Hướng Dẫn Tiếp Theo

- Trong bài học tiếp theo, chúng ta sẽ học cách viết mã CAPL để điều khiển hoạt động của các thành phần trong mô phỏng, bao gồm việc xử lý các tín hiệu và các sự kiện liên quan.

Bài 25.

Lập Trình Trực Quan Trong CANoe: Tạo Giao Diện Điều Khiển Và Tương Tác Với Các Biến Môi Trường

1. Tạo Biến Môi Trường

- Trước khi tạo giao diện điều khiển, người dùng cần tạo bốn biến môi trường (environment variables) trong cơ sở dữ liệu, bao gồm: Accelerator, Brake, RPM of the motor, và Switch Motor Button.
- **Tạo biến Accelerator và Brake**: Định nghĩa giá trị tối thiểu là 0 và tối đa là 1, vì chỉ có hai trạng thái là nhấn hoặc không nhấn. Biến được thiết lập là "unrestricted" để có thể đọc và ghi.
- **Tạo biến RPM of the Motor**: Định nghĩa giá trị tối thiểu là 0x0000 và tối đa là 0x1770, tương ứng với giá trị thập phân là 6000 RPM.
- **Tạo biến Switch Motor Button**: Biến này được kích hoạt với giá trị 1 khi công tắc động cơ được nhấn, và trở về 0 khi công tắc không nhấn.

2. Thiết Kế Giao Diện Trong Panel Designer

- Người dùng mở Panel Designer trong CANoe để tạo các phần tử cần thiết cho giao diện điều khiển.
- **Thêm các công tắc**: Dùng toolbox ở bên phải của cửa sổ để kéo và thả các phần tử như công tắc (switch) vào panel. Mỗi công tắc được liên kết với biến môi trường tương ứng bằng cách kéo và thả biến môi trường đó vào công tắc.
- **Thêm đồng hồ đo tốc độ (speedometer)**: Tìm phần tử đồng hồ đo (analog gauge) trong toolbox và thả nó vào panel. Đặt giá trị tối thiểu là 0 và tối đa là 6000 RPM, sau đó liên kết nó với biến môi trường RPM of the motor.

3. Viết Mã CAPL

- Mã CAPL được viết trong CAPL Editor liên kết với Engine Control Node, do cụm điều khiển (Cluster ECU) đã bị tắt trong mô phỏng.
- **Khai báo các biến và bộ đếm thời gian**: Hai bộ đếm thời gian (timer) và các biến integer được khai báo để lưu trữ giá trị RPM của động cơ.
- **Sự kiện thay đổi giá trị biến môi trường**: Sử dụng hàm `on EnvVar` để phát hiện khi giá trị của biến môi trường `Switch Motor Button` thay đổi, và dùng `if` để kiểm tra xem biến này có giá trị là 1 hay không. Nếu có, tín hiệu `ignition status` sẽ được đặt giá trị tương ứng.

- **Điều khiển tăng RPM khi nhấn bàn đạp ga**: Khi biến môi trường `Accelerator` có giá trị 1 (bàn đạp ga được nhấn), giá trị RPM tăng 50 đơn vị mỗi 300 ms. Nếu người dùng tiếp tục nhấn bàn đạp, bộ đếm thời gian sẽ được kích hoạt lại, kiểm tra liên tục và tăng giá trị RPM.
- **Điều khiển giảm RPM khi nhấn phanh**: Chiến lược tương tự được áp dụng cho biến môi trường `Brake`, nhưng thay vì tăng, giá trị RPM sẽ giảm đi 50 đơn vị mỗi 300 ms khi bàn đạp phanh được nhấn.

4. Kiểm Tra Mô Phỏng

- Người dùng chạy mô phỏng, nhấn công tắc động cơ để bật động cơ. Khi nhấn bàn đạp ga, giá trị RPM sẽ tăng lên, và khi nhấn bàn đạp phanh, giá trị RPM sẽ giảm xuống. Các thay đổi được hiển thị trên đồng hồ đo RPM trong giao diện.
- Nếu giữ phanh liên tục, RPM sẽ giảm về 0. Khi nhấn và thả ga, mô phỏng hoạt động đúng như mong đợi.

5. Kết Luận

- Bài học này đã chỉ rõ cách tạo giao diện điều khiển đơn giản trong CANoe và cách sử dụng mã CAPL để tương tác với các biến môi trường, mô phỏng hoạt động của một hệ thống động cơ và cụm điều khiển.
- Ở bài học tiếp theo, người dùng sẽ được hướng dẫn cách tạo các log và gateway để tiếp tục quản lý các tín hiệu trong hệ thống CAN.

Bài 26.

Gateway Trong CANoe

1. Giới Thiệu Về Gateway Trong CAN

- **Mục tiêu của bài học**: Giải thích cách thực hiện gateway trong mạng CAN để phản chiếu (mirror) các thông điệp từ một đường CAN sang một đường CAN khác. Đây là kỹ năng quan trọng khi trong thực tế có thể cần phản chiếu thông điệp từ một đường CAN sang đường CAN khác để đảm bảo hoạt động đồng bộ của các thiết bị trên cùng một hệ thống.

2. Tạo Mạng CAN Mới Và Cơ Sở Dữ Liệu

- **Tạo mạng CAN mới**: Do cấu hình hiện tại chỉ có một mạng CAN, cần tạo thêm một mạng CAN mới để chứa đơn vị điều khiển túi khí (airbag control unit). Đơn vị này cần biết tốc độ động cơ để có thể kích hoạt túi khí kịp thời trong trường hợp xảy ra tai nạn.
- **Tạo cơ sở dữ liệu**: Mở một cơ sở dữ liệu mới, chọn mẫu CAN và đặt tên là `Database 2`.
- **Tạo nút điều khiển túi khí**: Trong `Database 2`, tạo nút điều khiển túi khí và đặt tên là `Airbag Control Unit`.
- **Tạo thông điệp và tín hiệu**: Tạo thông điệp `Airbag Status` với các tín hiệu tương ứng như `Airbag Status Charge 1`, `Airbag Status Charge 2`, và `Airbag Status Charge 3`. Các tín hiệu này có chiều dài là 1 bit để biểu thị trạng thái kích hoạt của túi khí.
- **Tạo bảng giá trị**: Tạo bảng giá trị `Airbag Status Activation` với các giá trị miêu tả `Airbag Deactivated` và `Airbag Activated` cho tín hiệu túi khí.

3. Cấu Hình Nút Gateway

- **Tạo lại nút điều khiển động cơ**: Nút `Engine Control Node` cũng cần được tạo trong `Database 2` để nó có thể giao tiếp trên cả hai mạng CAN. Đặt tên nút giống như trong `Database 1` để đảm bảo đồng bộ.
- **Tạo thông điệp và tín hiệu cho nút điều khiển động cơ**: Tạo thông điệp `Engine Velocity` với tín hiệu `Engine RPM` để phản chiếu thông tin tốc độ động cơ từ `Database 1` sang `Database 2`.

- **Liên kết thông điệp và tín hiệu**: Liên kết thông điệp `Airbag Status` với nút điều khiển túi khí và thông điệp `Engine Velocity` với nút điều khiển động cơ trong `Database 2`.
- **Nhập thuộc tính từ `Database 1`**: Sử dụng tùy chọn `Import Attribute Definitions` để nhập các thuộc tính từ `Database 1` vào `Database 2`.

4. Viết Mã CAPL Để Thực Hiện Gateway

- **Mở CAPL Editor**: Mở trình soạn thảo CAPL của `Engine Control Node` trong `Network 2` và bắt đầu viết mã để phản chiếu thông điệp.
- **Lấy giá trị tín hiệu từ `Database 1`**: Sử dụng lệnh `this.` để lấy giá trị của tín hiệu `RPM` từ `Network 1`.
- **Thiết lập tín hiệu trong `Network 2`**: Sử dụng hàm `setSignal` để thiết lập giá trị của tín hiệu `Engine RPM` trong `Network 2` bằng với giá trị lấy từ `Network 1`.
- **Kiểm tra và sửa lỗi**: Trong quá trình biên dịch mã, nếu gặp lỗi do trùng tên tín hiệu, cần đổi tên tín hiệu để tránh xung đột. Đổi tên tín hiệu trong `Network 2` thành `Round Per Minute Network 2` để phân biệt.

5. Kiểm Tra Và Chạy Mô Phỏng

- **Chạy mô phỏng**: Sau khi chạy mô phỏng, kiểm tra cửa sổ console để xác nhận rằng các tín hiệu được thiết lập đúng cách. Kiểm tra trong `Trace Window` để đảm bảo tín hiệu `Engine Velocity` được gửi ra đúng cách trên `Network 2`.
- **Xử lý lỗi**: Nếu gặp lỗi liên quan đến các thuộc tính, cần đảm bảo rằng các thuộc tính của thông điệp trong `Database 2` giống với `Database 1`.

6. Kết Luận

- **Tóm tắt**: Bài học đã chỉ rõ cách thiết lập một gateway trong CANoe để phản chiếu thông điệp giữa hai mạng CAN khác nhau. Đây là kỹ năng quan trọng cho các kỹ sư kiểm thử trong việc đảm bảo đồng bộ hóa các thông điệp trên các mạng CAN khác nhau.
- **Hướng dẫn tiếp theo**: Bài học tiếp theo sẽ hướng dẫn cách lấy log của toàn bộ các đường CAN cùng với việc quay video để giám sát và phân tích chi tiết hơn.

Bài 27.

Ghi Lại Dữ Liệu Trong CANoe

1. Giới Thiệu Về Quá Trình Ghi Log Trong CANoe

- **Tầm quan trọng của việc ghi log**: Ghi lại log của một mạng CAN là một công việc rất quan trọng. Bạn có thể thu thập toàn bộ cấu trúc của mạng CAN trong một chiếc xe thực trong suốt chu trình lái thử và sau đó phân tích tất cả các tín hiệu của tất cả các đường CAN thực trong văn phòng khi hoàn thành chu trình lái thử. Điều này giúp bạn có thể truy xuất giá trị của từng tín hiệu vào mỗi thời điểm cụ thể một cách chính xác.

2. Bắt Đầu Quá Trình Ghi Log

- **Cấu hình ghi log**: Để ghi lại log, bạn cần nhấp đúp vào ô vuông bên trái của mục `Logging`, sau đó chọn thư mục đích và đặt tên cho tệp log. Định dạng tệp nên chọn là BLF.
- **Bắt đầu ghi log**: Sau khi thiết lập xong, nhấn vào nút `Start Measurement` và xác nhận. Bạn sẽ thấy nhãn `Logging Active` xuất hiện ở góc dưới cùng. Trong `Trace Window`, tất cả các thông điệp hiện tại trên `CAN Network 1` và `CAN Network 2` sẽ được ghi lại dưới dạng `Frames`.

3. Thay Đổi Giá Trị Tín Hiệu Và Ghi Log

- **Thay đổi giá trị tín hiệu**: Khi bạn thay đổi giá trị của một tín hiệu cụ thể (ví dụ: thay đổi trạng thái của tín hiệu `Ignition Status` thành `Engine On`), thay đổi này sẽ được ghi lại trong log.
- **Dừng quá trình ghi log**: Khi quá trình ghi log hoàn thành, nhấn nút `Stop Measurement`. Tất cả các khung CAN sẽ được lưu trữ trong tệp mà bạn đã đặt tên (ví dụ: `Log_1.blf`).

4. Phát Lại Dữ Liệu Log

- **Chuyển sang chế độ ngoại tuyến**: Để phát lại dữ liệu đã ghi, chuyển sang chế độ `Offline` bằng cách nhấp vào công tắc bên trái trong cấu hình và chọn tệp log BLF đã ghi trước đó.

- **Phát lại dữ liệu log**: Trong `Trace Window`, nhấn nút `Start` để bắt đầu phát lại dữ liệu. Lưu ý rằng bạn phải vô hiệu hóa mục ghi log hiện tại để tránh xung đột.
- **Xem dữ liệu trên đồ thị**: Trong cửa sổ đồ thị, bạn có thể thêm các tín hiệu từ cả `CAN Network 1` và `CAN Network 2` để xem lại xu hướng tín hiệu trong quá trình ghi log.

5. Ghi Lại Video Cùng Với Dữ Liệu Log

- **Cấu hình ghi hình**: Bạn có thể cấu hình để quay video cùng với việc ghi log bằng cách sử dụng một camera kết nối với PC. Cấu hình video bao gồm việc chọn nguồn video (camera), nguồn âm thanh (microphone), và các thông số thời gian trước và sau khi kích hoạt.
- **Phát lại video cùng dữ liệu log**: Khi phát lại, bạn có thể đồng bộ hóa cửa sổ video với dữ liệu log để xem lại những gì đã xảy ra trên màn hình cụm đồng hồ của xe tại một thời điểm cụ thể.

6. Kết Luận

- **Tóm tắt**: Bài học này đã hướng dẫn chi tiết cách ghi lại và phát lại dữ liệu log của mạng CAN trong CANoe, đồng thời tích hợp việc ghi hình để phân tích toàn diện. Đây là kỹ năng quan trọng giúp bạn phân tích và khắc phục sự cố trong quá trình kiểm thử hệ thống CAN.
- **Hướng dẫn tiếp theo**: Bài học tiếp theo sẽ hướng dẫn cách kích hoạt các thay đổi tín hiệu trên mạng CAN theo những cách khác nhau.

Bài 28.

Interactive Generator Block trong CANoe

1. Giới Thiệu Interactive Generator Block

- **Mục đích**: Interactive Generator (IG) Block là một công cụ mạnh mẽ trong CANoe, cho phép bạn gửi các thông điệp CAN một cách độc lập, không phụ thuộc vào việc các thông điệp đó có được định nghĩa trong các cơ sở dữ liệu đã tạo hay không. Điều này rất hữu ích trong các trường hợp cần kiểm tra hoặc mô phỏng các tình huống đặc biệt trên mạng CAN.

2. Thêm Interactive Generator Block vào Mạng CAN

- **Cách thêm**: Để thêm IG Block, bạn phải dừng mô phỏng, sau đó nhấp chuột phải vào một trong các mạng CAN và chọn "Insert CAN Interactive Generator". Sau khi thêm, bạn mở block này để cấu hình.
- **Chức năng**: IG Block có khả năng xuất bản bất kỳ thông điệp CAN nào mà bạn cung cấp, không phụ thuộc vào mạng CAN mà nó được đặt. Điều này có nghĩa là các thông điệp có thể được gửi trên cả mạng CAN1 và CAN2.

3. Tránh Trùng Lặp Thông Điệp Trên Mạng

- **Cảnh báo**: Khi sử dụng IG Block, bạn phải đảm bảo rằng các thông điệp mà nó gửi ra không trùng lặp với các thông điệp từ các nút (node) khác trên cùng một mạng. Nếu không, các thông điệp sẽ bị gửi hai lần trên mạng, gây ra hiện tượng trùng lặp và nhiễu tín hiệu.

4. Ví Dụ Gửi Thông Điệp Từ IG Block

- **Thêm thông điệp vào IG Block**: Bạn có thể thêm thông điệp vào IG Block bằng cách chọn "Add Frame from Database" và chọn thông điệp mong muốn từ cơ sở dữ liệu tương ứng.
- **Gửi thông điệp**: Khi đã thêm thông điệp, bạn có thể cấu hình để gửi thông điệp theo chu kỳ bằng cách chọn tùy chọn "Periodic" và thiết lập thời gian chu kỳ theo ý muốn.

5. Tránh Gửi Thông Điệp Từ Nhiều Nguồn

- **Tắt các nút xung đột**: Nếu bạn đã cấu hình IG Block để gửi một thông điệp cụ thể, hãy đảm bảo rằng bạn đã tắt các nút trên mạng mà cũng gửi cùng thông điệp đó, tránh gây xung đột tín hiệu trên mạng.
- **Kiểm tra trong Can Trace**: Sau khi bắt đầu mô phỏng, bạn có thể kiểm tra các thông điệp được gửi từ IG Block trong cửa sổ Can Trace. Nếu có hiện tượng nhấp nháy hoặc thay đổi giá trị tín hiệu bất thường, điều đó có thể do trùng lặp thông điệp từ nhiều nguồn.

6. Khả Năng Độc Lập Của IG Block

- **Gửi thông điệp trên các mạng khác nhau**: Một điểm mạnh của IG Block là khả năng gửi thông điệp đến các mạng khác nhau, không phụ thuộc vào việc nó được đặt trong mạng nào. Ví dụ, bạn có thể đặt IG Block trong mạng CAN2 nhưng vẫn gửi được thông điệp thuộc mạng CAN1.

7. Kết Luận

- **Tóm tắt**: IG Block là một công cụ linh hoạt và mạnh mẽ, cho phép bạn kiểm soát hoàn toàn việc gửi các thông điệp CAN trên mạng. Việc sử dụng IG Block cần phải cẩn thận để tránh trùng lặp thông điệp, và cần phải biết cách tắt các nút không cần thiết để tránh xung đột.
- **Hướng dẫn tiếp theo**: Bài học tiếp theo sẽ giới thiệu thêm một phương pháp khác để gửi thông điệp trên mạng CAN, giúp bạn nắm vững mọi khả năng trong việc quản lý thông điệp trên mạng.

Bài 29.

Signal Generator và Lọc CAN Trace trong CANoe

1. Giới Thiệu Về Signal Generator

- **Mục đích**: Signal Generator là một công cụ trong CANoe cho phép bạn gửi các tín hiệu với các xu hướng (trend) tùy chỉnh, như sóng sin, sóng vuông hoặc bất kỳ hàm tùy chỉnh nào. Công cụ này đặc biệt hữu ích khi bạn cần mô phỏng các tín hiệu với các đặc tính phức tạp hoặc chu kỳ dài.

2. Thêm Tín Hiệu Vào Signal Generator

- **Cách thực hiện**: Để sử dụng Signal Generator, bạn cần mở phần cấu hình của nó trong CANoe. Phần này nằm dưới mục "Simulation" ở phía bên phải của cửa sổ. Sau khi mở Signal Generator, nhấp vào biểu tượng tín hiệu ở góc trên bên trái của cửa sổ để thêm tín hiệu.
- **Ví dụ**: Tôi đã thêm tín hiệu "RPM round per min" và đặt biên độ mặc định là 500 đơn vị, chu kỳ là 10 giây để dễ dàng quan sát xu hướng trên cửa sổ đồ thị. Đơn vị đo lường sẽ được thừa kế từ đơn vị đã được định nghĩa trong cơ sở dữ liệu.

3. Bắt Đầu Mô Phỏng Với Signal Generator

- **Cấu hình**: Nếu bạn để phần "active" và "auto start" được chọn, tín hiệu sẽ được gửi ngay lập tức sau khi bạn nhấn nút bắt đầu mô phỏng.
- **Kết quả**: Khi bắt đầu mô phỏng, bạn có thể thấy tốc độ động cơ trong thông điệp liên quan của cả mạng CAN1 và CAN2. Nếu bạn chuyển sang cửa sổ đồ thị, bạn sẽ thấy xu hướng hình sin của tín hiệu đã đề cập.

4. Sự Khác Biệt Giữa Signal Generator và Interactive Generator Block

- **Sự khác biệt**: Sự khác biệt chính giữa Signal Generator và Interactive Generator Block (IG Block) là với Signal Generator, nút CAN gửi thông điệp phải được mô phỏng trong mạng, trong khi với IG Block, nút CAN có thể ở bất kỳ trạng thái nào.
- **Lợi ích của Signal Generator**: Công cụ này cho phép bạn mô phỏng tín hiệu CAN với các xu hướng tùy chỉnh rất linh hoạt như sóng sin, sóng vuông hoặc các hàm tùy chỉnh khác.

5. Lọc Thông Điệp Trong CAN Trace

- **Cách lọc**: Để lọc các thông điệp trong CAN Trace, bạn cần mở cửa sổ thiết lập đo lường, dừng mô phỏng và nhấp chuột phải vào phía bên trái của hộp "trace", sau đó chọn "Insert a Channel Filter".
- **Áp dụng bộ lọc**: Sau khi thêm bộ lọc, bạn có thể áp dụng "block filter" hoặc "pass filter" để chỉ xem các thông điệp từ một kênh CAN cụ thể.
- **Ví dụ**: Nếu bạn muốn chỉ xem các thông điệp trên mạng CAN1, bạn có thể áp dụng bộ lọc "pass filter" cho CAN1 và "block filter" cho CAN2. Khi khởi động lại mô phỏng, bạn sẽ chỉ thấy các thông điệp trên kênh CAN1.

6. Kết Luận

- **Tóm tắt**: Bài học này đã giới thiệu cách sử dụng Signal Generator để tạo tín hiệu tùy chỉnh và cách lọc các thông điệp trong CAN Trace để dễ dàng quản lý và phân tích. Đây là những công cụ mạnh mẽ giúp bạn kiểm soát tốt hơn quá trình mô phỏng và phân tích mạng CAN.
- **Hướng dẫn tiếp theo**: Bài học tiếp theo sẽ hướng dẫn cách quản lý Replay Block trong CANoe, một công cụ quan trọng khác giúp bạn tái hiện lại các thông điệp đã thu thập trước đó.

Bài 30.

Replay Block Trong CANoe

1. Giới Thiệu Replay Block

- **Chức năng**: Replay Block trong CANoe cho phép bạn tái hiện lại bất kỳ log nào đã thu thập trước đó lên một mạng CAN thực. Điều này rất hữu ích trong việc kiểm tra các bộ điều khiển (ECU) trên bàn thử nghiệm, khi mà bạn muốn tái tạo toàn bộ các tín hiệu từ một phương tiện thực tế lên mạng CAN chỉ có một ECU duy nhất.

2. Thao Tác Thực Hiện Replay Block

- **Tạo Replay Block**: Để tạo Replay Block, nhấp chuột phải vào một mạng CAN trong cấu hình CANoe và chọn "Insert Replay Block".
- **Ứng dụng thực tế**: Giả sử bạn muốn tái hiện các tín hiệu từ bộ điều khiển động cơ (ECU) lên cụm đồng hồ hiển thị (HMI). Để làm điều này, bạn cần thu thập một log chứa tất cả các thông điệp đang truyền trên các kênh ứng dụng CAN1 và CAN2.

3. Thực Hiện Ghi Log

- **Bắt đầu quá trình ghi log**: Kích hoạt logging và bắt đầu mô phỏng để ghi lại tất cả các tín hiệu trong khoảng 40 giây.
- **Lưu file log**: Sau khi dừng việc ghi log, chuyển đổi trạng thái sang offline và xác nhận rằng log đã được lưu vào thư mục đầu vào (input folder). Nếu cần lọc bớt các thông điệp không mong muốn, bạn có thể tạo một log mới và lọc bỏ những thông điệp đó.

4. Lọc Bỏ Thông Điệp Không Mong Muốn

- **Thêm bộ lọc sự kiện**: Để lọc bỏ các thông điệp không mong muốn, bạn cần thêm một sự kiện loại CAN filter trước khi ghi log. Sau đó, chọn các thông điệp cần lọc bỏ bằng cách thêm chúng từ cơ sở dữ liệu và áp dụng bộ lọc.

5. Tái Hiện Log

- **Kiểm tra log đã lọc**: Sau khi log đã được lọc, chuyển sang chế độ offline, thêm log đã lọc vào thư mục đầu vào, và kiểm tra các thông điệp trong can trace. Bạn sẽ thấy rằng chỉ có những thông điệp từ bộ điều khiển động cơ (ECU) được gửi đi, trong khi các thông điệp từ cụm đồng hồ hiển thị (HMI) đã bị loại bỏ.

6. Kết Luận

- **Tóm tắt**: Trong bài học này, bạn đã học cách sử dụng Replay Block để tái hiện log thu thập trước đó trên một mạng CAN thực. Điều này rất quan trọng khi bạn cần tái tạo môi trường thực tế lên một mạng CAN nhỏ hơn hoặc với ít thiết bị hơn, giúp việc kiểm tra và chẩn đoán trở nên hiệu quả hơn.

7. Hướng Dẫn Tiếp Theo

- **Lập kế hoạch tái hiện**: Trong bài học tiếp theo, chúng ta sẽ tìm hiểu cách thêm log đã lọc vào Replay Block để tái hiện lại nó trên một mạng thực, cung cấp cho bạn một cái nhìn hoàn chỉnh về việc tái tạo dữ liệu CAN.

Bài 31.

Replay Block Để Tái Hiện Log Trên Mạng CAN Thực

1. Giới Thiệu và Chuẩn Bị Replay Block

- **Nội dung chính**: Trong bài học này, chúng ta sẽ học cách tái hiện lại file log `log_one_filtered` đã được tạo ở bài học trước trên một mạng CAN thực.
- **Chuẩn bị**: Đầu tiên, di chuyển đến mạng CAN nơi Replay Block đã được thêm vào. Do tất cả các tín hiệu đã có sẵn trong log, ta cần phải vô hiệu hóa hai node hiện tại.

2. Cấu Hình Replay Block

- **Chọn file log**: Nhấp đúp vào Replay Block và chọn `log_one_filtered` làm nguồn (source file) rồi nhấn OK.
- **Chuyển sang chế độ online**: Để tái hiện log trên mạng CAN thực, bạn cần chuyển mô phỏng sang chế độ online và bắt đầu mô phỏng.

3. Kiểm Tra Và Điều Chỉnh Thang Thời Gian

- **Kiểm tra hiển thị**: Chuyển sang cửa sổ đồ thị (graphics window). Nếu không thấy các điểm mẫu hiển thị, cần điều chỉnh thang thời gian.
- **Điều chỉnh thang thời gian**: Nhấp chuột phải vào cửa sổ đồ thị, chọn "Configuration", sau đó chuyển sang mục "Time" và điều chỉnh thang thời gian sao cho phù hợp để thấy rõ các mẫu trong khoảng thời gian.

4. Quan Sát Kết Quả Tái Hiện

- **Kiểm tra trace**: Khi kiểm tra trace, bạn sẽ thấy các thông điệp `engine status` và `engine RPM` được gửi đi đến khoảng giây thứ 43, thời điểm mà ta đã dừng việc ghi log `log_one_filtered`.

5. Tổng Kết

- **Kết luận**: Chúng ta đã đi qua tất cả các yếu tố quan trọng cần biết để có thể đảm nhiệm vị trí kỹ sư kiểm thử ô tô ở cấp độ junior.
- **Bài học tiếp theo**: Trong bài học cuối cùng, chúng ta sẽ làm quen với Vector Analyzer và tổng kết lại toàn bộ khóa học.

Bài 32.

Tổng hợp kiến thức khóa học "Mastering CAN Network"

1. Tổng quan về khóa học

- **Nội dung chính**: Khóa học tập trung vào việc giải thích các khái niệm cơ bản và nâng cao về mạng CAN (Controller Area Network) trong ngành ô tô. Mục tiêu là để tất cả các học viên có thể hiểu rõ các chủ đề đã được đề cập.

2. Giới thiệu về Vector Analyzer

- **So sánh với CANoe**: Vector Analyzer có giao diện và cách thức hoạt động tương tự như Vector CANoe, nhưng đơn giản hơn. Trong Vector Analyzer, bạn chỉ quản lý một node duy nhất và mô phỏng các thông điệp CAN thông qua việc chèn khối Interactive Generator (IG). Không có node mô phỏng hay node thực, tất cả đều được quản lý bằng khối IG.
- **Cách sử dụng**: Bạn có thể đọc các thông điệp trên CAN trace và trên cửa sổ đồ thị (graphics window) giống như trong CANoe.

3. Giới thiệu về chẩn đoán trên xe thực

- **OBD-II và CAN**: Để thực hiện kiểm tra chẩn đoán trên xe thực, cần kết nối đầu nối OBD-II với bảng chẩn đoán của xe. Sử dụng bộ dây cáp để kết nối OBD-II với phần cứng Vienne (Vector), chia đầu còn lại của kết nối thành bốn đầu nối DB9 và kết nối chúng với phần cứng Vienne.
- **Thông điệp chẩn đoán**: Thông điệp chẩn đoán thường là thông điệp CAN với ID loại extended (29 bit). Các thông điệp này có thể được đọc trong CAN trace của CANoe hoặc Analyzer.

4. Tổng quan về các node và cảm biến trong xe

- **Các node chính**: Trong phần thực hành của khóa học, chúng ta đã đề cập đến các node quan trọng như Airbag (túi khí), Cluster HMI (giao diện người dùng của bảng điều khiển), và ECU (điều khiển động cơ).
- **Cảm biến và LIN bus**: Các cảm biến có thể được đặt trên mạng LIN bus. Các cảm biến này giao tiếp với node Master LIN, sau đó node này sẽ chuyển tiếp thông tin lên mạng CAN. Node Master LIN hoạt động như một gateway giữa LIN bus và CAN bus.
- **Ví dụ về cảm biến**: Cảm biến mưa là một ví dụ điển hình. Nó phát hiện mưa và gửi thông tin này dưới dạng thông điệp LIN đến node Master LIN. Node này sau đó sẽ đăng thông tin lên mạng CAN.

5. Kết luận khóa học

- **Mục tiêu đạt được**: Khóa học đã giúp học viên nắm vững các kỹ năng cần thiết để trở thành kỹ sư kiểm thử ô tô. Tất cả các chủ đề đã được giải thích một cách dễ hiểu và chi tiết.
- **Lời cảm ơn**: Giảng viên gửi lời cảm ơn đến các học viên đã tham gia khóa học và hy vọng rằng những kiến thức này sẽ hữu ích trong sự nghiệp của họ.

Bài 33.

Tìm kiếm việc làm trong lĩnh vực ô tô - Các buổi học riêng tư

1. Thành công của khóa học

- **Phổ biến**: Khóa học đã trở thành một trong những khóa học bán chạy nhất. Giảng viên rất tự hào khi có thể giúp đỡ nhiều người tìm được việc làm trong lĩnh vực kiểm thử ô tô.
- **Tác động tích cực**: Sự phổ biến của khóa học cho thấy sự hữu ích và cần thiết của kiến thức mà khóa học mang lại, đặc biệt đối với những ai muốn tìm việc trong ngành ô tô.

2. Các bước tiếp theo đề xuất

- **Mục tiêu tiếp theo**: Để xây dựng một danh mục công việc ở cấp trung-cao cấp khi ứng tuyển vào các vị trí kiểm thử ô tô, học viên cần học cách tự động hóa hàng nghìn trường hợp thử nghiệm (test cases) cho các ECU ô tô.
- **Kiến thức cần thiết**: Để đạt được mục tiêu này, cần phải có kiến thức chi tiết về Python, vì ngôn ngữ này rất quan trọng trong việc tự động hóa các trường hợp thử nghiệm.
- **Khóa học Python đề xuất**: Giảng viên đề xuất học viên nên tham gia khóa học Python hoàn chỉnh trên trang Udemy của giảng viên để bắt đầu học cách tự động hóa các trường hợp thử nghiệm bằng Python. Link khóa học được đề xuất: Khóa học Python hoàn chỉnh.

3. Các buổi học Vector Canoe riêng tư

- **Dịch vụ bổ sung**: Giảng viên sẵn sàng cung cấp các buổi học riêng về Vector Canoe thông qua các buổi học trực tiếp.
- **Liên hệ**: Học viên có thể liên hệ với giảng viên qua LinkedIn để sắp xếp các buổi học riêng. Chi phí sẽ được thông báo dựa trên số lượng người tham gia.

Bài 34.

Q&A

1. Vấn đề không thấy biến môi trường trong CANdb

- **Nguyên nhân**: Do đang sử dụng phiên bản demo của phần mềm.
- **Giải pháp**: Cần phải sử dụng phiên bản đầy đủ của phần mềm để thấy được biến môi trường.

2. Sự khác biệt giữa "CANoe Template" và "CAN Template" trong DB++

- **Thông tin**: Chúng ta luôn nói về "CANoe Template", nhưng thông tin này không quan trọng lắm đối với quá trình học và làm việc.

3. Giá trị kiểu "unsigned" và "signed"

- **Unsigned**: Là giá trị chỉ có thể là số dương.
- **Signed**: Là giá trị có thể là số âm hoặc số dương, nghĩa là dấu +/- của giá trị rất quan trọng.

4. Cần phải định nghĩa thuộc tính (attributes) cho các tín hiệu giống như cách làm cho các thông điệp không?

- **Câu trả lời**: Chúng ta chỉ cần định nghĩa thuộc tính cho các thông điệp, không cần cho các tín hiệu.

5. Sự khác biệt giữa "raw value" và "symbolic value" trong lớp tương tác (interaction layer)

- **Symbolic value**: Là giá trị ở dạng dễ hiểu cho con người, ví dụ như "closed" - "open".
- **Raw value**: Là biểu diễn ở dạng HEX, ví dụ như "0x01" - "0x00".

6. CAPL được sử dụng khi nào?

- **Câu trả lời**: CAPL được sử dụng để chèn thông điệp vào các ECU thực tế trên bàn kiểm thử.

7. Gateway kết nối cho cùng một nút và cùng tên ECU có giống nhau không? Và thông điệp và tín hiệu sẽ có cùng tên hay khác nhau?

- **Câu trả lời**: Các thông điệp và tín hiệu có thể khác nhau giữa hai mạng khác nhau trong đó có cùng một tên ECU.

8. Signal generators chỉ được sử dụng cho các ECU mô phỏng? Làm sao để sử dụng signal generator cho mạng ECU thực tế?

- **Câu trả lời**: Signal generators chỉ được sử dụng cho các ECU mô phỏng. Tuy nhiên, quan trọng là các thông điệp được gửi, chứ không phải ECU gửi thông điệp đó.

9. Tại sao chúng ta sử dụng Replay blocks?

- **Câu trả lời**: Replay block cho phép bạn thu thập tất cả các thông điệp đang di chuyển dọc theo một mạng nhất định và có thể tái tạo các thông điệp đó vào một mạng khác. Ví dụ, bạn có thể thu thập toàn bộ mạng CAN trên xe và tái tạo lại vào một mạng chỉ có một thành phần trên bàn kiểm thử.
- **Câu hỏi phụ**: Khi log cho Replay block, tất cả các ECU cần ở chế độ mô phỏng hay thời gian thực?
    - **Câu trả lời**: Khi bạn tái tạo replay block, các ECU trên bàn kiểm thử nên ở chế độ thời gian thực, nếu không các thông điệp từ ECU thật và từ replay block sẽ chồng lên nhau.

10. Bố trí các ECU trên bàn kiểm thử như thế nào và chúng ta có kiểm thử trong xe thực tế không?

- **Câu trả lời**: Trong xe, tất cả các ECU đều là thật, vì vậy tất cả các ECU trong cấu hình của bạn sẽ ở chế độ thời gian thực và không mô phỏng. Trên bàn kiểm thử, bạn sẽ đặt các ECU không có trên bàn ở chế độ mô phỏng.

Bài 35.

Q&A 2

1. Tình trạng "Bus Off" trong CAN và có mã DTC nào không?

- **Bus Off** là trạng thái lỗi của bộ điều khiển CAN. Chỉ bộ truyền mới chuyển sang trạng thái Bus Off khi bộ đếm lỗi truyền đạt vượt quá 255.
- Có hai cách để phục hồi từ Bus Off:
    1. Tự động sau khi 128 chuỗi liên tiếp của 11 bit 'recessive' đã được theo dõi trên bus.
    2. Một nút có thể bắt đầu khôi phục từ trạng thái Bus Off chỉ khi có yêu cầu từ người dùng.
- Khi một nút phải truyền thông điệp, nó kiểm tra xem thông điệp có được truyền thành công hay không. Nếu không thành công sau 255 lần thử, nút sẽ vào trạng thái Bus Off và dừng truyền. Có thể phục hồi sau khi loại bỏ nhiễu, thường xảy ra sau 200ms.

2. Timeout hoặc lỗi timeout trong CAN là gì?

- Các ECU nhận được lập trình để sau khi một thông điệp không được phát hiện trong một khoảng thời gian nhất định, DTC sẽ được đặt ra.
- Sau khi hết một khoảng timeout (thường là 10-20 lần chu kỳ của thông điệp), nếu không nhận được thông điệp từ ECU, DTC sẽ được kích hoạt.
- Giá trị timeout được đặt trong biến nội bộ của nút nhận và có thể đọc từ nội dung của DTC.

3. Kỹ thuật xử lý lỗi trong CAN là gì?

- Tham khảo liên kết: CAN Protocol Error Types.

4. Điều gì xảy ra khi truyền hai thông điệp có cùng ID trong CAN?

- Cả hai đều có thể được truyền nhưng có thể chênh lệch 2-3ms. Nếu ECU phát hiện tần số truyền không đúng, DTC sẽ được đặt ra.
- Nếu hai thông điệp có ID khác nhau, thông điệp có ID thấp hơn (có độ ưu tiên cao hơn) sẽ được truyền trước.

5. Tại sao CAN luôn sử dụng điện trở đầu cuối 120 ohm mà không phải 100 hoặc 200 ohm?

- Điện trở đầu cuối 120 ohm được sử dụng để hấp thụ năng lượng của tín hiệu CAN, ngăn không cho phản xạ tín hiệu xảy ra từ đầu cáp. Điều này giúp tránh nhiễu và tổn hại tín hiệu.
- Để đạt kết quả tốt nhất, điện trở đầu cuối phải khớp với trở kháng danh định của cáp, thường là 120 ohm cho CAN tốc độ cao.

6. CRC error xảy ra ở cả bộ nhận và bộ truyền hay chỉ một bên trong CAN?

- CRC được tính ở cả hai bên: nút gửi và nút nhận. Nút gửi tính CRC dựa trên nội dung khung và đính kèm vào cuối khung. Nút nhận cũng tính CRC và so sánh với CRC từ bên gửi. Nếu khớp, khung được coi là hợp lệ, ngược lại sẽ báo lỗi khung.

7. Biến môi trường là gì?

- Biến môi trường là những biến mà giá trị của chúng có thể được lấy từ mã CAPL, giúp người dùng Canoe gửi thông điệp CAN khi một sự kiện nhất định xảy ra (ví dụ như nhấn nút).
- Giá trị của biến môi trường có thể được sử dụng để kích hoạt sự kiện gửi thông điệp CAN trong mã CAPL.

8. Làm thế nào để đo độ trễ của một thông điệp CAN?

- Tham khảo liên kết: How can I measure latency in a CAN bus?.

9. Replay block có hoạt động ở chế độ offline không?

- **Replay block** không sử dụng ở chế độ offline vì nó dùng để gửi lại các thông điệp đã được ghi lại (việc ghi lại có thể thực hiện offline) lên một mạng CAN thực tế.

10. Cách tạo console cho các nút như thế nào?

- Console cho các nút là nơi gửi thông điệp CAN từ bảng tương tác (interaction layer panel). Bạn cần gán các thuộc tính đã giải thích trong CANDB++.

11. Có sự khác biệt về tốc độ giữa CAN standard và CAN extended không?

- Không có sự khác biệt về tốc độ giữa CAN standard và CAN extended. Cả hai đều có tốc độ thông thường là 500kb/s cho cả mạng chẩn đoán và mạng ứng dụng.
- Sự khác biệt nằm ở độ dài ID:
    - CAN standard: ID dài 11 bit (ví dụ: 4DA).
    - CAN extended: ID dài 29 bit (ví dụ: 14DA55F1), trong đó 14DA là tiền tố, 55 là địa chỉ ECU nhận và F1 là địa chỉ của thiết bị chẩn đoán.