9. AUTOSAR PORT INTERFACES

Khái niệm về cổng (Ports)

. **Cổng**: Điểm kết nối để giao tiếp giữa các thành phần phần mềm.

. **Trách nhiệm**: Chuyển tải thông tin giữa các thành phần.

Các loại cổng

1. **Cổng cung cấp (Provider Port - P-Port)** . **Vai trò**: Dùng để gửi thông tin từ một thành phần phần mềm.
2. **Cổng nhận (Receiver Port - R-Port)** . **Vai trò**: Dùng để nhận thông tin từ các thành phần khác.
3. **Cổng cung cấp và nhận (Provider Receiver Port - PR-Port)** . **Vai trò**: Có tính chất của cả cổng cung cấp và cổng nhận, dùng để gửi và nhận cùng một thông tin từ một thành phần.

Các loại giao diện cổng

1. **Giao diện gửi nhận (Sender Receiver Interface)**

   . **Vai trò**: Truyền dữ liệu giữa các cổng.

   . **Dữ liệu**: Các kiểu dữ liệu nguyên thủy hoặc phức tạp như integer, float, cấu trúc, mảng, v.v.

   . **Cách hoạt động**:

   - Thành phần cung cấp sẽ có cổng cung cấp định nghĩa với giao diện gửi nhận.
   - Thành phần nhận sẽ có cổng nhận với cùng định nghĩa giao diện gửi nhận.

2. **Giao diện khách hàng máy chủ (Client Server Interface)**

   . **Vai trò**: Truyền hoặc truy cập các dịch vụ từ các thành phần khác.

   . **Cách hoạt động**:

   - Thành phần có hàm gọi sẽ là máy chủ, và sẽ có cổng cung cấp.
   - Thành phần gọi hàm sẽ là khách hàng, và sẽ có cổng nhận.

3. **Giao diện dữ liệu không bay hơi (NV Data Interface)**

   . **Vai trò**: Giao tiếp các giao diện NVM giữa các thành phần.

   . **Cách hoạt động**:

   - Để ghi dữ liệu vào khối NVM, thành phần ứng dụng sẽ có cổng cung cấp và thành phần NV block sẽ có cổng nhận.
   - Để khôi phục dữ liệu, thành phần NV block sẽ có cổng cung cấp và các thành phần khác sẽ có cổng nhận.

4. **Giao diện tham số (Parameter Interface)**

   . **Vai trò**: Trao đổi các tham số hiệu chỉnh giữa các thành phần.

   . **Cách hoạt động**: Được dùng để chia sẻ các tham số hiệu chỉnh từ thành phần phần mềm tham số đến các thành phần khác.

5. **Giao diện chuyển đổi chế độ (Mode Switch Interface)**

   . **Vai trò**: Dùng khi có các trạng thái hệ thống được định nghĩa trước.

   . **Ví dụ**: Định nghĩa trạng thái đèn xe có thể là OFF, ON với High beam, hoặc ON với Low beam.

   . **Cách hoạt động**:

   - Thành phần quyết định chế độ sẽ là cổng cung cấp.
   - Thành phần thực hiện hành động dựa trên quyết định sẽ là cổng nhận.

6. **Giao diện kích hoạt (Trigger Interface)**

   . **Vai trò**: Kích hoạt chức năng trên thành phần khi xảy ra sự kiện.

   . **Cách hoạt động**:

   - Thành phần quyết định kích hoạt sẽ là cổng cung cấp.
   - Thành phần chờ kích hoạt từ bên ngoài sẽ là cổng nhận.

Tóm tắt và hướng đi tiếp theo

. **Lưu ý các biểu tượng độc đáo của mỗi giao diện cổng**: Chúng ta sẽ sử dụng các biểu tượng này để đại diện cho cổng và các giao diện của chúng.

. **Tiếp tục khóa học**: Tìm hiểu chi tiết hơn về giao diện gửi nhận và khách hàng máy chủ, vì chúng là các trường hợp sử dụng chính trong phần mềm.

Chúng ta sẽ đi sâu hơn vào hai giao diện này và các cơ chế giao tiếp chi tiết mà Autosar cung cấp trong các phần tiếp theo của khóa học.

10. SENDER RECEIVER INTERFACE

Khái niệm cơ bản

Giao diện gửi nhận (Sender Receiver Interface) là mô hình phân phối thông tin không đồng bộ, nơi một sender (người gửi) phân phối thông tin tới một hoặc nhiều receiver (người nhận). Đây là giao tiếp không đồng bộ, nghĩa là người gửi không bị chặn và không mong đợi hoặc nhận được phản hồi từ người nhận.

Đặc điểm chính

1. **Giao tiếp không đồng bộ**:
    - Người gửi chỉ cung cấp thông tin và người nhận tự quyết định khi nào và cách sử dụng nó.
    - Hạ tầng truyền thông (RTE) chịu trách nhiệm phân phối thông tin tới người nhận.
2. **Người gửi không biết danh tính hoặc số lượng người nhận**:
    - Giao tiếp một chiều, không cần biết ai nhận hoặc có bao nhiêu người nhận.

Các loại giao tiếp

1. **Giao tiếp một-một (One-to-One Communication)**:
    - Một thành phần gửi và một thành phần nhận.
2. **Giao tiếp một-nhiều (One-to-Many or 1:M Communication)**:
    - Một thành phần gửi và nhiều thành phần nhận.
3. **Giao tiếp nhiều-một (Many-to-One or M:1 Communication)**:
    - Nhiều thành phần gửi và một thành phần nhận.
4. **Giao tiếp nhiều-nhiều (Many-to-Many or M:1 Communication)**:
    - Không được hỗ trợ trong Autosar. Không cho phép có nhiều hơn một người gửi và nhiều hơn một người nhận cho cùng một dữ liệu.

Cấu hình giao diện gửi nhận

1. **Tạo giao diện gửi nhận**:
    - Xác định dữ liệu sẽ được truyền trong biến data prototype.
    - Tên ngắn của biến data prototype xác định tên của giao diện được truyền.
2. **Thuộc tính bổ sung**:
    - Đơn vị, phương pháp tính toán, v.v., có thể được cung cấp từ kiểu dữ liệu Application-primitive hoặc Application-Record trong giao diện gửi nhận.
3. **Gửi nhiều dữ liệu trên cùng một cổng**:
    - Để gửi nhiều dữ liệu trên cùng một cổng, có thể xác định nhiều biến data prototype trong cùng một giao diện gửi nhận.
4. **Tạo cổng trên thành phần**:
    - Liên kết giao diện gửi nhận đã tạo với cổng trên thành phần.

Ví dụ về cấu hình

1. **Thiết lập cổng cung cấp (Provider Port)** trên thành phần muốn gửi dữ liệu.
2. **Thiết lập cổng nhận (Receiver Port)** trên thành phần muốn nhận dữ liệu.
3. **Kết nối**: Liên kết cổng cung cấp với cổng nhận.

Minh họa

- **One-to-One**: Thành phần A gửi dữ liệu tới thành phần B.
- **One-to-Many**: Thành phần A gửi dữ liệu tới thành phần B, C, và D.
- **Many-to-One**: Thành phần A và B gửi dữ liệu tới thành phần C.

Kết luận

. **Giao diện gửi nhận** là một trong những giao diện chính được sử dụng trong Autosar, cho phép phân phối thông tin không đồng bộ giữa các thành phần phần mềm.

Trong các phần tiếp theo, chúng ta sẽ đi sâu vào các ví dụ cụ thể và cấu hình chi tiết của giao diện gửi nhận cũng như các giao diện khác trong Autosar.

11. CLIENT SERVER INTERFACE

Khái niệm cơ bản

Giao diện khách hàng-máy chủ (Client Server Interface) được sử dụng để gọi hàm hoặc dịch vụ giữa các thành phần phần mềm. Khi chúng ta nói về dịch vụ, có thể hiểu đơn giản là một hàm trên một module mà có thể được gọi từ các module khác.

- **Server**: Là nhà cung cấp dịch vụ.
- **Client**: Là người sử dụng dịch vụ.

Cách hoạt động

1. **Client**: Khởi tạo giao tiếp, yêu cầu server thực hiện dịch vụ và truyền tham số nếu cần.
2. **Server**: Chờ yêu cầu từ client, thực hiện dịch vụ được yêu cầu và trả về phản hồi cho yêu cầu của client.

Ví dụ

Giả sử có một thành phần chịu trách nhiệm thực hiện các phép toán như cộng hai số. Nó có một hàm gọi là "Sum" nhận giá trị đầu vào là X và Y, cộng chúng lại và trả về kết quả dưới dạng tham số thứ ba "S".

- **Server**: Thành phần chứa hàm "Sum", sẽ tạo một cổng cung cấp (provider port).
- **Client**: Thành phần cần sử dụng hàm "Sum", sẽ tạo một cổng nhận (receiver port).

Loại gọi hàm

1. **Gọi hàm đồng bộ (Synchronous Call)**:
    - Client bị chặn cho đến khi server trả về kết quả.
    - Sử dụng khi client cần kết quả ngay lập tức để xử lý tiếp.
2. **Gọi hàm không đồng bộ (Asynchronous Call)**:
    - Client không bị chặn, chỉ kích hoạt hàm và tiếp tục phần code tiếp theo, kết quả sẽ được xử lý sau.
    - Sử dụng khi client không cần kết quả ngay lập tức hoặc không có kết quả trả về từ server.

Cấu hình giao diện khách hàng-máy chủ

1. **Tạo giao diện khách hàng-máy chủ**:
    - Tạo giao diện và định nghĩa các thao tác server bên trong, mô tả các thao tác server.
2. **Định nghĩa thao tác server**:
    - Phải chứa định nghĩa các tham số với hướng của tham số như IN, OUT hoặc INOUT.

    - Phản ánh cấu hình giao diện khách hàng-máy chủ cho hàm "Sum" với các tham số và hướng của chúng đến hàm server.
3. **Lỗi ứng dụng**:
    - Có thể xác định các lỗi mà server có thể trả về trong trường hợp có vấn đề.
4. **Cổng cung cấp và nhận**:
    - Tạo cổng cung cấp cho server và cổng nhận cho client.
    - Ánh xạ giao diện khách hàng-máy chủ này vào các cổng tương ứng.

Ví dụ minh họa

- **One-to-One**: Thành phần A (client) gọi hàm "Sum" trên thành phần B (server).
- **One-to-Many**: Thành phần B (server) cung cấp hàm "Sum" cho các thành phần A, C, và D (clients).

Kết luận

- **Client Server Interface** là một trong những giao diện chính được sử dụng trong Autosar để thực hiện các dịch vụ hoặc gọi hàm giữa các thành phần phần mềm.
- **Cấu hình**: Bao gồm tạo giao diện, định nghĩa thao tác server, tạo cổng cung cấp và nhận, và ánh xạ giao diện này vào các cổng.

Trong các phần tiếp theo của khóa học, chúng ta sẽ xem xét chi tiết hơn về các ví dụ và cấu hình cụ thể của giao diện khách hàng-máy chủ.

12. PORT INTERFACE (EXAMPLE)

Yêu cầu sử dụng

1. Lấy tốc độ xe từ cảm biến tốc độ bên ngoài.
2. Nếu tốc độ vượt quá giá trị hiệu chỉnh, thực hiện các hành động sau:
    - Ngắt kim phun ngay lập tức và dừng xe.
    - Ghi lại lỗi thông qua module chẩn đoán.
3. Tính toán tốc độ tối đa mà xe đã đạt được và lưu dữ liệu này khi tắt máy.
4. Khôi phục lại dữ liệu khi ECU được bật lên trong chu kỳ lái tiếp theo.

Chọn giao diện phần mềm dựa trên yêu cầu

1. **Giao tiếp từ thành phần trừu tượng hóa ECU đến thành phần cảm biến/điều khiển**:
    - **Giao diện gửi nhận (Sender Receiver Interface)**:
        - **ECU Abstraction Component**: Cổng cung cấp (Provider Port).
        - **Sensor Actuator Component**: Cổng nhận (Receiver Port).
2. **Giao tiếp từ thành phần cảm biến/điều khiển đến các thành phần ứng dụng khác**:
    - **Giao diện gửi nhận (Sender Receiver Interface)**:
        - **Sensor Actuator Component**: Tính toán tốc độ và gửi thông tin này đến các thành phần ứng dụng khác qua cổng cung cấp.
3. **Lấy giá trị hiệu chỉnh từ thành phần tham số**:
    - **Giao diện tham số (Parameter Interface)**:
        - **Parameter Software Component**: Cung cấp tham số hiệu chỉnh (Provider Port).
        - **Application Software Component**: Nhận tham số hiệu chỉnh (Receiver Port).
4. **Giao tiếp ghi tốc độ tối đa vào thành phần phần mềm khối NVM**:
    - **Giao diện NV (NV Interface)**:
        - **Application Software Component**: Cung cấp dữ liệu tốc độ tối đa (Provider Port).
        - **NV Block Software Component**: Nhận dữ liệu tốc độ tối đa (Receiver Port).
5. **Giao tiếp giữa thành phần phần mềm khối NVM và trình quản lý NVM cơ bản**:
    - **Giao diện khách hàng-máy chủ (Client Server Interface)**:
        - **NV Block Software Component**: Gọi hàm dịch vụ từ NVM manager.
6. **Ghi nhật ký chẩn đoán**:
    - **Giao diện khách hàng-máy chủ (Client Server Interface)**:
        - **Application Software Component**: Gọi hàm dịch vụ từ Diagnostics Manager.
7. **Ngắt kim phun**:
    - **Giao diện gửi nhận (Sender Receiver Interface)** hoặc **Giao diện kích hoạt (Trigger Interface)**:
        - **Application Software Component**: Gửi lệnh ngắt kim phun (Provider Port).
        - **Complex Device Driver**: Nhận lệnh ngắt kim phun (Receiver Port).

Tóm tắt

- **Giao diện gửi nhận (Sender Receiver Interface)**: Dùng cho giao tiếp dữ liệu.
    - Ví dụ: Truyền tốc độ từ cảm biến đến thành phần cảm biến/điều khiển và từ thành phần cảm biến/điều khiển đến các thành phần ứng dụng khác.
- **Giao diện tham số (Parameter Interface)**: Dùng cho giao tiếp tham số hiệu chỉnh.
    - Ví dụ: Lấy giá trị hiệu chỉnh từ thành phần tham số.
- **Giao diện NV (NV Interface)**: Dùng cho giao tiếp với thành phần phần mềm khối NVM.
    - Ví dụ: Ghi tốc độ tối đa vào thành phần phần mềm khối NVM.
- **Giao diện khách hàng-máy chủ (Client Server Interface)**: Dùng cho gọi hàm hoặc dịch vụ từ các module khác.
    - Ví dụ: Ghi nhật ký chẩn đoán và giao tiếp với NVM manager.
- **Giao diện kích hoạt (Trigger Interface)**: Có thể dùng để kích hoạt các hành động thời gian thực.
    - Ví dụ: Ngắt kim phun.

Kết luận

Qua ví dụ này, chúng ta có thể thấy rõ cách lựa chọn loại giao diện phù hợp dựa trên yêu cầu giao tiếp cụ thể. Giao diện gửi nhận thường được sử dụng cho giao tiếp dữ liệu, trong khi giao diện khách hàng-máy chủ được sử dụng cho các cuộc gọi hàm và dịch vụ.