### Giới thiệu và Tổng quan chức năng

A computer screen shot of a computer

Description automatically generated

Tài liệu này xác định chức năng, API và cấu hình của mô-đun CAN Driver trong AUTOSAR Basic Software (gọi là "mô-đun Can" trong tài liệu này).

Mô-đun Can là một phần của tầng thấp nhất, thực hiện việc truy cập phần cứng và cung cấp một API độc lập với phần cứng cho tầng trên.

Tầng trên duy nhất có quyền truy cập vào mô-đun Can là mô-đun CanIf (xem thêm SRS\_SPAL\_12092).

Mô-đun Can cung cấp dịch vụ để khởi tạo truyền và gọi hàm gọi lại của mô-đun CanIf để thông báo sự kiện, độc lập với phần cứng. Hơn nữa, nó cung cấp dịch vụ để kiểm soát hành vi và trạng thái của CAN controllers thuộc cùng một CAN Hardware Unit.

Nhiều CAN controllers có thể được kiểm soát bởi một mô-đun Can duy nhất miễn là chúng thuộc cùng một CAN Hardware Unit

Để có mô tả chi tiết hơn về CAN controller và CAN Hardware Unit, xem chương Viết tắt và biểu đồ trong [5].

### Từ viết tắt và viết tắt

|  |  |
| --- | --- |
| **Từ viết tắt và viết tắt** | **Mô tả** |
| CAN controller | Bộ điều khiển CAN phục vụ chính xác một kênh vật lý. |
| CAN Hardware Unit | Bộ phần cứng CAN có thể bao gồm một hoặc nhiều bộ điều khiển CAN cùng loại và một hoặc nhiều vùng CAN RAM. Bộ phần cứng CAN nằm trên chip hoặc thiết bị bên ngoài. Đơn vị phần cứng CAN được đại diện bởi một trình điều khiển CAN. |
| CAN L-PDU | Đơn vị dữ liệu giao thức lớp liên kết dữ liệu. Bao gồm Mã định danh, Độ dài dữ liệu và Dữ liệu (SDU). (xem [20]) |
| CAN L-SDU | Đơn vị dữ liệu dịch vụ lớp liên kết dữ liệu. Dữ liệu được vận chuyển bên trong L-PDU. (xem [20]) |
| DLC | Mã độ dài dữ liệu (một phần của thông báo CAN mô tả độ dài SDU) |
| Hardware Object | Đối tượng phần cứng CAN được định nghĩa là bộ đệm PDU bên trong CAN RAM của bộ phần cứng CAN/bộ điều khiển CAN. Đối tượng Phần cứng được định nghĩa là bộ đệm L-PDU bên trong RAM CAN của Đơn vị Phần cứng CAN. |
| Hardware Receive Handle (HRH) | Xử lý nhận phần cứng (HRH) được xác định và cung cấp bởi Trình điều khiển CAN. Mỗi HRH thường chỉ đại diện cho một đối tượng phần cứng. HRH có thể được sử dụng để tối ưu hóa việc lọc phần mềm. |
| Hardware Transmit Handle (HTH) | Bộ điều khiển truyền phần cứng (HTH) được xác định và cung cấp bởi Trình điều khiển CAN. Mỗi HTH thường chỉ đại diện cho một hoặc nhiều đối tượng phần cứng được cấu hình như nhóm bộ đệm truyền phần cứng. |
| Inner Priority Inversion | Việc truyền L-PDU có mức ưu tiên cao bị ngăn chặn bởi sự hiện diện của L-PDU có mức ưu tiên thấp đang chờ xử lý trong cùng một đối tượng phần cứng truyền. |
| ISR | Dịch vụ ngắt thường xuyên |
| L-PDU Handle | Tay cầm L-PDU được xác định và đặt bên trong lớp mô-đun Canlf. Thông thường, mỗi bộ điều khiển đại diện cho một L-PDU, là một cấu trúc cố định chứa thông tin dành cho quá trình xử lý Tx/Rx. |
| MCAL | Lớp trừu tượng của vi điều khiển |
| Outer Priority Inversion | Khoảng cách thời gian xảy ra giữa hai L-PDU truyền liên tiếp. Trong trường hợp này, L-PDU có mức ưu tiên thấp hơn từ nút khác có thể ngăn việc gửi L-PDU có mức ưu tiên cao hơn của chính nó. Ở đây, L-PDU có mức ưu tiên cao  hơn không thể tham gia phân xử trong quá trình truy cập mạng vì L-PDU có mức ưu tiên thấp hơn đã thắng trong quá trình phân xử. |
| Physical Channel | Kênh vật lý đại diện cho giao diện từ bộ điều khiển CAN đến Mạng CAN. Các kênh vật lý khác nhau của thiết bị phần cứng CAN có thể truy cập các mạng khác nhau. |
| Priority | Mức độ ưu tiên của CAN L-PDU được thể hiện bằng Mã định danh CAN. Giá trị số của mã định danh càng thấp thì mức độ ưu tiên càng cao. |
| SFR | Đăng ký chức năng đặc biệt. Thanh ghi phần cứng điều khiển hành vi của bộ điều khiển. |
| SPAL | Lớp trừu tượng ngoại vi tiêu chuẩn |

#### Priority Inversion - Đảo ngược ưu tiên

A diagram of a data link

Description automatically generated

"Nếu chỉ có một bộ đệm truyền đơn được sử dụng, có thể xảy ra sự đảo ngược ưu tiên nội bộ. Do ưu tiên thấp, một thông điệp được lưu trữ trong bộ đệm phải chờ đợi cho đến khi 'lưu lượng trên bus ổn định'. Trong thời gian chờ đợi này, thông điệp này có thể ngăn cản một thông điệp có ưu tiên cao hơn được tạo ra bởi cùng một vi điều khiển từ việc được truyền qua bus."

A data link system

Description automatically generated with medium confidence

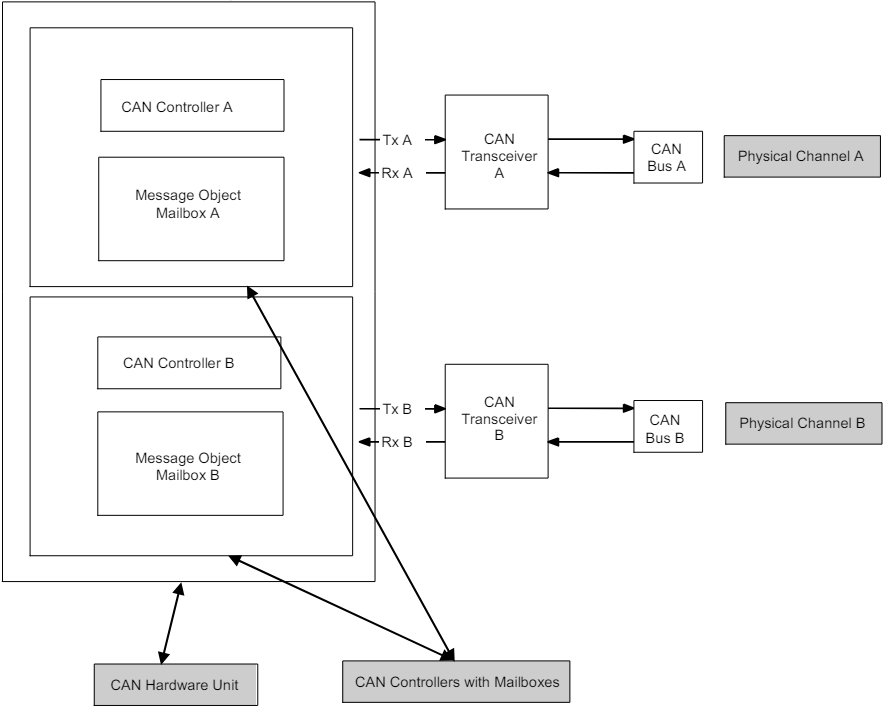
"Vấn đề của sự đảo ngược ưu tiên bên ngoài có thể xảy ra trong một số cài đặt CAN. Hãy giả sử rằng một nút CAN muốn truyền một gói tin của các thông điệp liên tiếp với ưu tiên cao, được lưu trữ trong các bộ đệm thông điệp khác nhau. Nếu khoảng thời gian giữa các thông điệp này trên mạng CAN dài hơn so với khoảng trống tối thiểu được xác định bởi tiêu chuẩn CAN, một nút thứ hai có thể bắt đầu truyền một thông điệp ưu tiên thấp hơn. Khoảng trống tối thiểu giữa các khung tin trên mạng CAN được xác định bởi trường Intermission, bao gồm 3 bit không quyết định. Một thông điệp, đợi trong quá trình truyền của một thông điệp khác, được bắt đầu trong khoảng thời gian Bus Idle, sớm nhất là trong bit tiếp theo sau trường Intermission. Ngoại lệ là một nút có thông điệp truyền đợi sẽ giải thích một bit động ở bit thứ ba của Intermission như một bit Start-of-Frame và bắt đầu truyền với bit nhận dạng đầu tiên mà không cần truyền trước một bit SOF. Thời gian xử lý nội bộ của một mô-đun CAN phải đủ ngắn để gửi các thông điệp liên tiếp với khoảng trống tối thiểu giữa các khung để tránh sự đảo ngược ưu tiên bên ngoài dưới tất cả các tình huống đã đề cập."

#### CAN Hardware Unit

CAN Hardware Unit kết hợp một hoặc nhiều CAN controllers, có thể được đặt trên chip hoặc là các thiết bị độc lập bên ngoài cùng loại, với

Hardware Objects chung hoặc riêng lẻ.

Hình dưới đây thể hiện một CAN Hardware Unit bao gồm hai CAN controllers kết nối với hai Kênh Vật lý:



### Related documentation

### Ràng buộc và giả định

#### 4.1 Giới hạn

Một bộ điều khiển CAN luôn tương ứng với một kênh vật lý. Được phép kết nối các kênh vật lý ở phía bus. Bất kể mô-đun CanIf sẽ xử lý

các bộ điều khiển CAN liên quan một cách riêng biệt.

Một số ít đơn vị phần cứng CAN hỗ trợ khả năng kết hợp nhiều bộ điều khiển CAN bằng cách sử dụng CAN RAM, để mở rộng số lượng đối tượng thông điệp cho một CAN controller. Các bộ điều khiển CAN kết hợp này được xử lý như một bộ điều khiển bởi mô-đun Can.

Mô-đun Can không hỗ trợ các khung CAN từ xa.

[SWS\_Can\_00237] ⌈ Mô-đun Can không được phép truyền các thông điệp được kích hoạt bởi yêu cầu truyền từ xa.⌋ (SRS\_Can\_01147)

[SWS\_Can\_00236] ⌈ Mô-đun Can phải khởi tạo CAN HW để bỏ qua bất kỳ yêu cầu truyền từ xa nào.⌋ (SRS\_Can\_01147)

#### Ứng dụng trong các lĩnh vực xe hơi

Mô-đun Can có thể được sử dụng cho bất kỳ ứng dụng nào, nơi giao thức CAN được sử dụng.

### Dependencies to other modules

#### Static Configuration

Các phần tử cấu hình mô tả trong chương 10 có thể được các mô-đun BSW khác tham chiếu đến cho cấu hình của chúng.

#### Driver Services

[SWS\_Can\_00238] ⌈ Nếu CAN controller nằm trên chip, mô-đun Can không được phép sử dụng bất kỳ dịch vụ của các trình điều khiển khác.⌋ (SRS\_BSW\_00005)

[SWS\_Can\_00239] ⌈ Hàm Can\_Init phải khởi tạo tất cả các tài nguyên phần cứng trên chip được sử dụng bởi CAN controller. Ngoại lệ duy nhất cho điều này là cấu hình chân I/O kỹ thuật số (của các chân được sử dụng bởi CAN), được thực hiện bởi trình điều khiển cổng.⌋ (SRS\_BSW\_00377)

[SWS\_Can\_00240] ⌈ Mô-đun Mcu (SPAL xem [8]) phải cấu hình các thiết lập của bộ đăng ký được 'chia sẻ' với các mô-đun khác.⌋ ()

Mẹo triển khai: Mô-đun Mcu phải được khởi tạo trước khi khởi tạo mô-đun Can.

[SWS\_Can\_00242] ⌈ Nếu CAN controller sử dụng, mô-đun Can phải sử dụng các dịch vụ của các trình điều khiển MCAL khác (ví dụ: SPI).⌋ (SRS\_BSW\_00005)

Mẹo triển khai: Nếu mô-đun Can sử dụng các dịch vụ của các trình điều khiển MCAL khác (ví dụ: SPI), phải đảm bảo rằng các trình điều khiển này đang hoạt động trước khi khởi tạo mô-đun Can. Thứ tự khởi tạo của các trình điều khiển khác nhau được chỉ định một phần trong [7].

[SWS\_Can\_00244] ⌈ Mô-đun Can phải sử dụng các API đồng bộ của các trình điều khiển MCAL cơ sở và không được cung cấp các hàm gọi lại có thể được gọi bởi các trình điều khiển MCAL.⌋ ()

Do đó, loại kết nối giữa vi điều khiển và Đơn vị Phần cứng CAN chỉ ảnh hưởng đến triển khai và không ảnh hưởng đến API.

#### System Services

[SWS\_Can\_00280] ⌈ Trong các trường hợp phần cứng đặc biệt, mô-đun Can phải đánh giá sự kiện của phần cứng.⌋ ()

[SWS\_Can\_00281] ⌈ Mô-đun Can phải sử dụng OsCounter được cung cấp bởi dịch vụ hệ thống để phát hiện timeout trong trường hợp phần cứng không phản ứng trong thời gian mong đợi (sự cố phần cứng) để ngăn chặn vòng lặp vô hạn.⌋ ()

Mẹo triển khai: Thời gian chặn của hàm mô-đun Can đang chờ phản ứng của phần cứng phải ngắn hơn chu kỳ kích hoạt của hàm chính CAN (tức là Can\_MainFunction\_Read), vì các chức năng chính CAN không thể được sử dụng cho mục đích đó.

#### Can module Users

[SWS\_Can\_00058] ⌈ Mô-đun Can tương tác với các mô-đun khác (ví dụ: Tracer Lỗi Mặc định (DET), Quản lý Trạng thái Ecu (ECUM)) với mô-đun CanIf một cách trực tiếp. Tài liệu này không bao giờ xác định nguồn gốc thực sự của một yêu cầu hoặc điểm đến thực sự của một thông báo. Trình điều khiển chỉ nhìn thấy mô-đun CanIf như là nguồn gốc và điểm đến.⌋ (SRS\_SPAL\_12092)

#### File structure

[SWS\_Can\_00436] ⌈ Can\_GeneralTypes.h phải chứa tất cả các loại và hằng số được chia sẻ giữa các mô-đun CAN AUTOSAR Can, CanIf và CanTrcv.⌋ ()