Một mô-đun Can cung cấp truy cập vào một CAN Hardware Unit có thể bao gồm nhiều CAN Controllers

### Trạng thái của Can HW Units

Can Driver có 2 trạng thái là: CAN\_UNINIT và CAN\_READY

* Hàm Can\_Init phải thay đổi trạng thái của mô-đun thành CAN\_READY, sau khi khởi tạo tất cả các bộ điều khiển bên trong HW Unit.

Thực hiện:

Các thiết lập thanh ghi phần cứng có ảnh hưởng đến tất cả các CAN Controller bên trong HW Unit chỉ có thể được thiết lập trong hàm Can\_Init

Mô-đun ECU State Manager phải gọi Can\_Init tối đa một lần trong quá trình chạy.

+ Mỗi CAN Controller phải được khởi động riêng lẻ bằng cách gọi hàm **Can\_SetControllerMode (CAN\_CS\_STARTED**)

### Điều khiển trạng thái của CanController

CanController cos 4 trạng thái:

UNINIT, STOPPED, STARTED và SLEEP

Có hai cách để kích hoạt các thay đổi trạng thái của CAN Controller bằng các sự kiện bên ngoài [sự kiện trên bus (hardware)].:

* Bus-off event
* HW wakeup event

Các sự kiện này được chỉ ra bằng một ngắt hoặc bằng một bit trạng thái được đặt trong hàm **Can\_MainFunction\_BusOff** hoặc **Can\_MainFunction\_Wakeup.**

Khi kích hoạt sẽ gửi thông báo cho CanIF với hàm gọi lại tương ứng. Trạng thái phần mềm sau đó được thay đổi bên trong hàm gọi lại này.

### Qúa trình chuyển đổi trạng thái

Một chuyển đổi trạng thái được kích hoạt bởi phần mềm với hàm Can\_SetControllerMode với chuyển đổi cần thiết làm tham số. Một chuyển đổi trạng thái thành công được kích hoạt bởi phần mềm được thông báo bằng hàm gọi lại (CanIf\_ControllerModeIndication). Việc theo dõi xem trạng thái yêu cầu đã đạt được là phần của một mô-đun lớp trên và không phải là phần của mô-đun Can.

#### Can\_SetControllerMode

Phần mềm có thể kích hoạt một chuyển đổi trạng thái của CAN Controllers với hàm Can\_SetControllerMode. Tùy thuộc vào phần cứng CAN, việc thay đổi cài đặt thanh ghi để chuyển sang một trạng thái mới của CAN Controllers có thể chỉ xảy ra sau một khoảng trễ. Mô-đun Can thông báo cho lớp trên (CanIf\_ControllerModeIndication) sau một chuyển đổi trạng thái thành công về trạng thái mới. Việc theo dõi xem trạng thái yêu cầu đã đạt được là phần của một mô-đun lớp trên và không phải là phần của mô-đun Can.

[**SWS\_Can\_00398**] ⌈ Hàm Can\_SetControllerMode phải sử dụng hệ thống dịch vụ GetCounterValue để theo dõi thời gian chờ để tránh chặn các hàm.⌋ ()

[**SWS\_Can\_00370**] ⌈ Hàm Can\_Mainfunction\_Mode phải kiểm tra 1 flag của thanh ghi trạng thái CAN cho đến khi cờ báo hiệu rằng thay đổi có hiệu lực và thông báo cho lớp trên với hàm CanIf\_ControllerModeIndication về một chuyển đổi trạng thái thành công với thông tin về CAN Controllers tương ứng với abstract CanIf ControllerId.⌋ ()

A diagram of a flowchart

Description automatically generated

### Chuyển đổi trạng thái bởi HW Events

|  |
| --- |
| **Can\_Write**  Trong AUTOSAR, hàm Can\_Write được sử dụng để gửi một khung dữ liệu (data frame) qua một bộ điều khiển CAN cụ thể.  Khi hàm Can\_Write được gọi, nó sẽ thực hiện các công việc sau:   1. Kiểm tra tính hợp lệ của các tham số đầu vào:    * Xác minh xem các tham số như HandleId, PduInfo và những tham số khác có hợp lệ không.    * Nếu có bất kỳ tham số nào không hợp lệ, hàm sẽ trả về một mã lỗi thích hợp. 2. Lấy thông tin cấu hình tương ứng với HandleId:    * HandleId xác định một kênh truyền dữ liệu (data communication channel) cụ thể.    * Thông tin cấu hình bao gồm các chi tiết như bộ điều khiển CAN được sử dụng, ID khung CAN, loại khung (CAN 2.0 hoặc CAN-FD), v.v. 3. Đẩy khung dữ liệu vào hàng đợi truyền (Transmit Queue) của bộ điều khiển CAN tương ứng:    * Khung dữ liệu sẽ được xây dựng dựa trên thông tin cấu hình và dữ liệu được cung cấp trong PduInfo.    * Nếu hàng đợi truyền đầy, hàm có thể trả về mã lỗi CAN\_BUSY. 4. Nếu khung dữ liệu được đẩy vào hàng đợi truyền thành công, hàm sẽ trả về mã thành công CAN\_OK.   Sau khi hàm Can\_Write kết thúc, việc truyền khung dữ liệu thực tế trên bus CAN sẽ do bộ điều khiển CAN xử lý dựa trên hàng đợi truyền và các quy tắc điều khiển truy cập bus CAN.  Khi Can\_Write gửi xong, nó sẽ xác nhận bằng hàm CanIf\_TxConfirmation() |

Khi nhận L-PDU, module Can sẽ gọi hàm RX indication CanIf\_RxIndication với ID, HoH, Abstract CanIf ControllerId trong tham số Mailbox.

[SWS\_Can\_00489] ⌈ Trình điều khiển CAN phải hỗ trợ các bộ điều khiển mà triển khai một FIFO phần cứng. Kích thước của FIFO được cấu hình thông qua "CanHwObjectCount".⌋ ()

chỉ là một ví dụ.

[**SWS\_Can\_00299**] ⌈ Module Can sẽ sao chép L-SDU vào bộ đệm bóng sau khi nhận, nếu bộ đệm RX không thể được bảo vệ (khóa) bởi Phần cứng CAN chống ghi đè bởi một thông điệp mới được nhận.⌋ ()

[**SWS\_Can\_00300**] ⌈ Module Can sẽ sao chép L-SDU vào bộ đệm bóng, nếu Phần cứng CAN không có sẵn toàn cầu.⌋ ()

Quá trình RX hoàn chỉnh (bao gồm sao chép vào tầng đích, ví dụ: COM) được thực hiện trong ngữ cảnh của ngắt RX hoặc trong ngữ cảnh của Can\_MainFunction\_Read.

[**SWS\_Can\_00012**] ⌈ Module Can sẽ đảm bảo rằng cả các ISR lẫn hàm Can\_MainFunction\_Read không thể bị gián đoạn bởi chính nó. Bộ đệm Phần cứng CAN (hoặc bóng) luôn nhất quán, vì nó được ghi và đọc theo chuỗi trong một hàm duy nhất không bị gián đoạn bởi chính nó.⌋ (SRS\_Can\_01059)