1.1. Lịch sử phát triển của chuẩn CAN

Controller Area Network (CAN hay CAN Bus) là một giao thức truyền thông nối tiếp rất hiệu quả cho các ứng dụng điều khiển thời gian thực với mức độ an toàn cao.

Ý tưởng của mạng CAN được nhóm kỹ sư tại GmbH Robert Bosch, Đức, nghiên cứu từ đầu thập niên 1980. Họ đã nghiên cứu thị trường cho một công nghệ bus mới dùng trong xe ô tô mà có thể cho phép đưa thêm nhiều chức năng vào nữa mà số lượng dây nối không quá lớn.

Lịch sử phát triển của CAN được tóm lược như sau:

· 1983: Bắt đầu dự án phát triển mạng trên xe hơi trong nội bộ hãng Bosch.
· 1986: Chính thức giới thiệu giao thức CAN.
· 1987: Những chíp điều khiển CAN đầu tiên xuất hiện ở nhà sản xuất linh kiện bán dẫn Intel và Philips.
· 1991: Bosch xuất bản thông số kỹ thuật CAN 2.0.
· 1992: Thành lập nhóm các nhà sử dụng và sản xuất CAN quốc tế: Hội CAN tự động hóa (CiA). Hội CiA xuất bản giao thức Lớp ứng dụng CAN (CAN Application Layer, CAL). Những chiếc xe Mercedes-Benz đầu tiên được trang bị CAN xuất hiện.
· 1993: Xuất bản tiêu chuẩn ISO 11898.
· 1994: CiA tổ chức Hội nghị CAN quốc tế lần thứ nhất (iCC). Allen-Bradley giới thiệu giao thức DeviceNet.
· 1995: Xuất bản Tiêu chuẩn ISO 11898 sửa đổi (định dạng khung mở rộng). CiA xuất bản giao thức CANopen.

Giải thích:

· CiA (CAN in Automation): Là nhóm các nhà sản xuất và sử dụng trên thế giới phát triển và hỗ trợ CANopen và các giao thức CAN cơ bản lớp cao hơn khác. Đây là tổ chức phi lợi nhuận được thành lập từ năm 1992 để cung cấp thông tin về công nghệ CAN cơ bản, sản xuất và tiếp thị. Có khoảng 500 công ty là thành viên của tổ chức phi lợi nhuận này và có trụ sở chính đặt tại Nuremberg, Đức.
· DeviceNet và CANopen: là hai giao thức lớp cao hơn cho CAN, dùng trong tự động.

Hiện nay CAN được ứng dụng trong rất nhều lĩnh vực khác nhau như: Xe ôtô, tàu khách và tàu hàng, hệ thống điện tử hàng hải, điện tử máy bay và hàng không, tự động hóa nhà máy, điều khiển máy công nghiệp, tự động hóa tòa nhà, thang máy (thang nâng và thang trượt), thiết bị phụ tùng y tế,….với tốc độ bit có thể lên đến 1 Mbit/s.

Các phân tích trong phần lý thuyết này là cho CAN 2.0B. Lý thuyết về CAN 2.0A tương tự CAN 2.0B chỉ khác là CAN 2.0A không có định dạng khung ‘mở rộng’ mà chỉ truyền và nhận định dạng khung chuẩn.

1.2. Các khái niệm cơ bản trong CAN:

1.2.1. Các đặc tính của CAN:

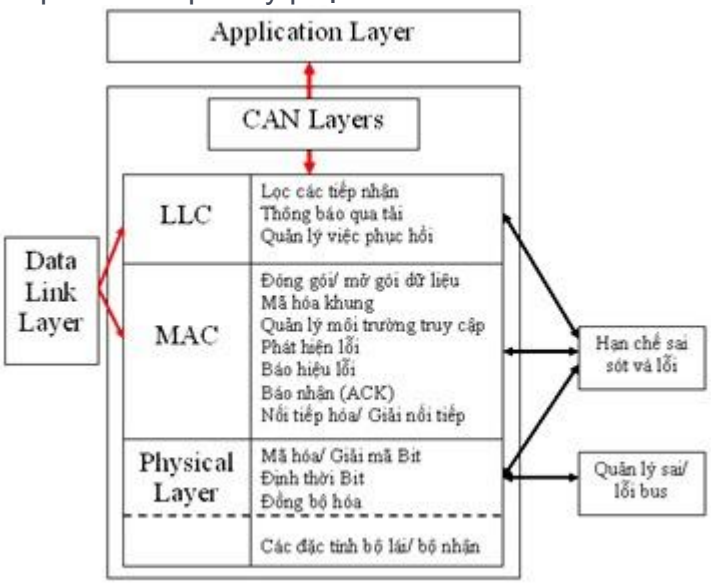
· Các thông điệp có mức độ ưu tiên khác nhau.
· Đảm bảo về thời gian trễ.

· Tính linh hoạt của cấu hình.
· Sự thu nhận đa điểm có đồng bộ về thời gian.
· Tính bền vững của dữ liệu mở rộng hệ thống.
· Phát hiện và báo hiệu lỗi.

1.2.2. Cấu trúc các lớp của CAN theo mô hình tham khảo OSI:

· Giao thức CAN thực thi trong hai lớp là lớp vật lý (Physical Layer) và lớp liên kết dữ liệu (Data Link Layer) còn các giao thức lớp cao hơn cho CAN thuộc lớp ứng dụng (Application Layer).
· Lớp vật lý định nghĩa cách thức mà các tín hiệu được truyền thực tế trên đường truyền. Ở lớp này, các vấn đề về định thời bit, mã hóa bit và đồng bộ sẽ được xử lý. Ngoại trừ, các đặc tính Lái/Nhận (Driver/Receiver) của lớp vật lý không được định nghĩa với mục đích cho phép môi trường truyền và sự thực thi mức tín hiệu được tối ưu cho mỗi ứng dụng.
· Lớp liên kết dữ liệu:

- Lớp phụ MAC (Medium Access Control) trình bày phần lõi của giao thức CAN. Nó biểu diễn các thông điệp nhận từ lớp phụ LLC (Logical Link Control) và cho phép các thông điệp được truyền từ lớp vật lý đến lớp phụ này. Lớp phụ MAC chịu trách nhiệm về các khung thông điệp, sự phân xử (Arbitration), ACK (Acknowledgment), phát hiện và báo hiệu lỗi. Lớp phụ MAC được giám sát bởi một đối tượng quản lý gọi là “Hạn chế lỗi” (Fault Confinement), nó là một cơ chế tự kiểm tra để phân biệt các nhiễu ngắn (các nhiễu tác động ngẫu nhiên và không có tính chất lâu dài) với các lỗi cố định (các lỗi có tính chất lâu dài).
- Lớp phụ LLC (Logical Link Control) liên quan đến việc lọc thông điệp, thông báo quá tải và quản lý phục hồi.



Các lớp ứng dụng CAN trong mô hình Open System Interconnection (OSI)

1.2.3. Một số đặc điểm cần lưu ý:

- *Thông điệp (Message):* Thông tin trên bus CAN được gửi dưới dạng các thông điệp được định dạng cố định khác nhau và được giới hạn về độ dài. Khi bus rảnh thì mỗi đơn vị kết nối có thể bắt đầu truyền một thông điệp mới.
- *Tuyến thông tin (Information Routing):*

Trong một hệ thống CAN, một nút CAN (CAN node) không được thiết lập theo kiểu mỗi nút sử dụng một thông tin về cấu hình hệ thống (nghĩa là không định địa chỉ trạm) mà mang các đặc điểm quan trong sau đây:

➢Tính linh động của hệ thống (System Flexibility): Các nút CAN có thể được thêm vào mạng CAN mà không yêu cầu bất cứ sự thay đổi nào về phần mềm và phần cứng ở mỗi nút và lớp ứng dụng.

➢Định tuyến thông điệp (Message Routing): Nội dung của một thông điệp được "đặt tên" bằng một định danh (IDENTIFIER). Định danh này không chỉ ra đích mà thông điệp sẽ được gửi đến. Định danh chỉ mô tả ý nghĩa và mức độ ưu tiên của dữ liệu để các nút trong mạng tự quyết định xem có thực thi thông điệp này hay không thông qua cơ chế lọc thông điệp (Message Filtering ở lớp LLC).

➢Thực thi đa điểm (Multicast): Kết quả của cơ chế lọc thông điệp là nhiều nút có thể nhận và thực hiện theo cùng một thông điệp.

➢Tính bền vững dữ liệu (Data Consistency): Một mạng CAN đảm bảo một thông điệp có thể được chấp nhận cùng lúc bởi nhiều nút hoặc không nút nào. Dữ liệu của hệ thống đạt được điều đó thông qua cơ chế thực thi đa điểm và điều khiển lỗi.

- *Tốc độ bit (Bit rate):* Tốc độ bit của CAN có thể khác nhau trong các hệ thống khác nhau nhưng trong một hệ thống cho trước thì tốc độ bit đồng nhất và cố định. Tốc độ bit còn tùy thuộc vào chiều dài đường truyền. Tốc độ tối đa có thể lên đến 1 Mbit/s. Sau đây là bảng một số thông số thực tế:

Tốc độ bit (Bit Rate – kbit/s)	Chiều dài bus (Bus Length - m)	Thời gian bit danh định (Nominal Bit-time - µs)
1000	30	1
800	50	1.25
500	100	2
250	250	4
125	500	8
62.5	1000	20
20	2500	50
10	5000	100

Bảng thông số tốc độ bit và chiều dài bus thực tế

- *Tính ưu tiên (Priorities):* IDENTIFIER sẽ định nghĩa một mức ưu tiên cố định cho thông điệp trong suốt quá trình truy cập bus.
- *Yêu cầu dữ liệu từ xa (Remote data request):* Bằng cách gửi một khung yêu cầu (Remote Frame), một nút có thể yêu cầu một nút khác gửi một khung dữ liệu (Data Frame) đáp ứng yêu cầu đó. Khung dữ liệu và khung yêu cầu tương ứng phải có IDENTIFIER giống nhau.
- *Multimaster:* Khi bus rảnh, Mỗi đơn vị đều có thể truyền thông điệp nhưng đơn vị nào truyền thông điệp có mức ưu tiên cao hơn sẽ có lợi khi truy cập bus, tức thông điệp đó được truyền trước.
- *Sự phân xử (Arbitration):*

➢Bất cứ khi nào bus rảnh, mỗi đơn vị đều có thể bắt đầu truyền thông điệp. Như vậy, nếu có hai hay nhiều đơn vị truyền thông điệp trong cùng một thời gian thì sẽ có xung đột khi truy cập bus. Để tránh xung đột này, cơ chế phân xử từng bit trong vùng IDENTIFIER được sử dụng để đảm bảo không có thông tin nào bị mất.

➢Nếu một khung dữ liệu và một khung yêu cầu có IDENTIFIER giống nhau được bắt đầu cùng một thời gian thì khung dữ liệu chiếm ưu thế hơn khung yêu cầu.

- *Tính an toàn (Safety):* Để có được sự an toàn cao cho việc truyền/nhận dữ liệu thì giao thức có áp dụng các biện pháp phát hiện lỗi, báo hiệu lỗi, tự kiểm tra được thực thi cho

mỗi nút CAN.
➢Phát hiện lỗi (Error detection): sử dụng cơ chế giám sát (Monitoring – so sánh mức bit được truyền với mức bit trên bus), kiểm tra độ dư vòng (CRC – Cyclic Redundancy Check), Chèn bit (Bit stuffing), kiểm tra khung thông điệp.
➢Các đặc tính của sự phát hiện lỗi: Tất cả các lỗi toàn thể được phát hiện, tất cả các lỗi cục bộ ở bộ truyền được phát hiện, phân biệt được đến 5 lỗi ngẫu nhiên trong 1 thông điệp, các lỗi nhóm với độ dài nhỏ hơn 15 được phát hiện, các lỗi với số lẻ lần được phát hiện.
➢Xác xuất lỗi không phát hiện được nhỏ hơn: ***(Tốc độ lỗi thông điệp)×4.7×10-11***
➢Hạn chế lỗi (Fault Confinement): Các nút CAN cho phép phân biệt các nhiễu ngắn và các sai sót cố định. Các nút hỏng được tắt.
•□*Sự kết nối (Connection):* Kết nối truyền tin nối tiếp CAN là một bus với số đơn vị kết nối đến bus là không giới hạn (theo lý thuyết). Trên thực tế, tổng số đơn vị kết nối đến bus bị giới hạn do thời gian trễ và các tải trên đường bus.
•□*Giá trị bus (Bus values):* Khi hoạt động, bus có thể mang một trong hai giá trị logic là 'dominant' (mức thấp – bit 0) hay 'recessive' (mức cao – bit 1). Trong cùng một thời điểm nếu 'dominant' và 'recessive' cùng được truyền lên bus thì giá trị bus sẽ là 'dominant' (do thực thi wire-AND sẽ được trình bày sau).
•□*Báo nhận (Acknowledgment):* Tất cả các bộ nhận sẽ kiểm tra tính chắc chắn của thông điệp được nhận và sẽ báo là nhận đúng hoặc sai.

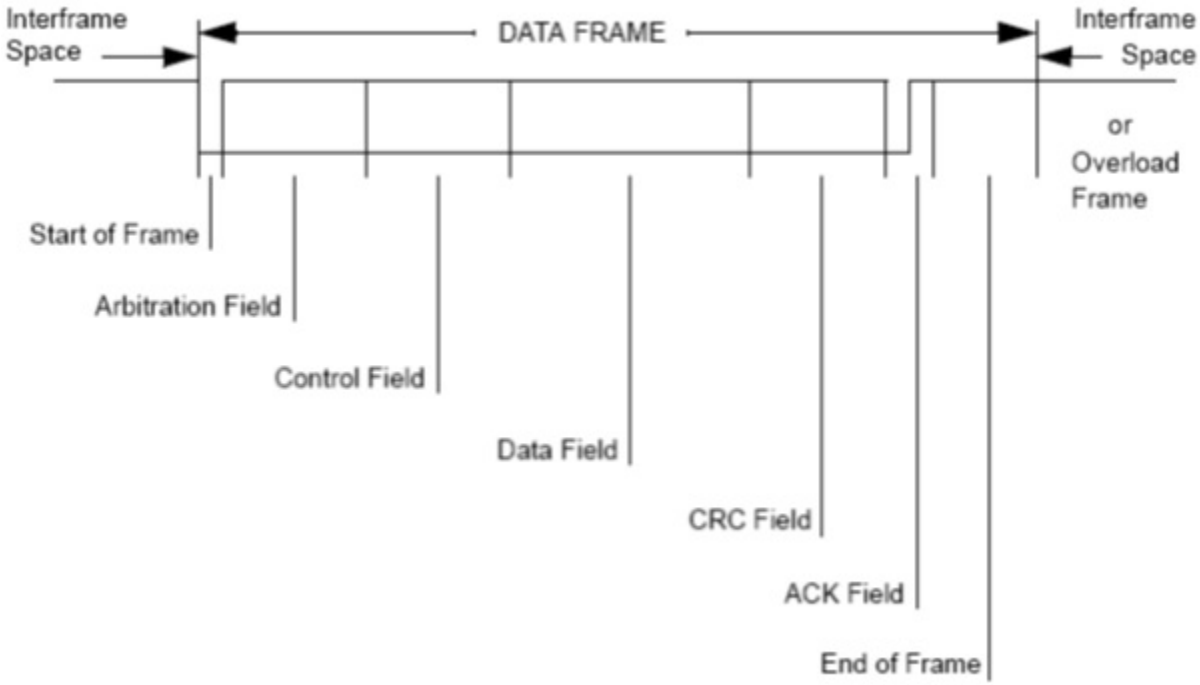
1.3. Các loại khung truyền trong giao thức CAN:
1.3.1. Định dạng khung:
Có hai định dạng khung khác nhau với độ dài IDENTIFIER khác nhau là:
•□Định dạng khung chuẩn (Standard Frames): IDENTIFIER dài 11 bit.
•□Định dạng khung mở rộng (Extended Frames): IDENTIFIER dài 29 bit.
Trong đó, CAN 2.0A chỉ sử dụng định dạng khung chuẩn. CAN 2.0B sử dụng được cả hai loại định dạng khung trên.
1.3.2. Các loại khung truyền:
Sự chuyển thông điệp được biểu hiện và điều khiển bởi bốn loại khung truyền khác nhau:

- Khung dữ liệu (Data Frame): mang dữ liệu từ bộ truyền đến bộ nhận.
- Khung yêu cầu (Remote Frame): được truyền bởi một đơn vị bus để đòi hỏi đơn vị bus khác truyền khung dữ liệu có IDENTIFIER giống khung yêu cầu đã gửi.
- Khung lỗi (Error Frame): được truyền bởi một đơn vị khi phát hiện ra lỗi bus.
- Khung quá tải (Overload Frame): được sử dụng để cung cấp một độ trễ cao hơn giữa khung dữ liệu (hoặc khung yêu cầu) trước với khung dữ liệu (hoặc khung yêu cầu) kế tiếp sau đó.

Chú ý: Các khung dữ liệu và các khung yêu cầu có thể sử dụng định dạng chuẩn hoặc định dạng mở rộng tùy vào phiên bản CAN. Chúng được ngăn cách với nhau bằng một khoảng liên khung (interframe space).

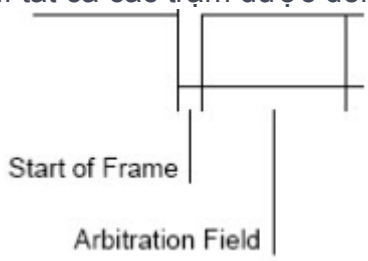
Cấu trúc khung dữ liệu

Một khung dữ liệu bao gồm bảy vùng bit khác nhau theo thứ tự là: Vùng bắt đầu khung (Start of frame – SOF), vùng phân xử (Arbitration field), vùng điều khiển (Control field), vùng dã liệu (Data field), vùng kiểm tra (CRC field), vùng báo nhận (ACK field), vùng kết thúc khung (End of frame – EOF).
Các vùng trên có số lượng bit và nhiệm vụ khác nhau. Khi truyền dữ liệu, bit có trọng số cao MSB (most significant bit) được truyền trước. Khung chuẩn và khung mở rộng khác nhau về số lượng bit ở vùng phân xử.



Cấu trúc khung dữ liệu

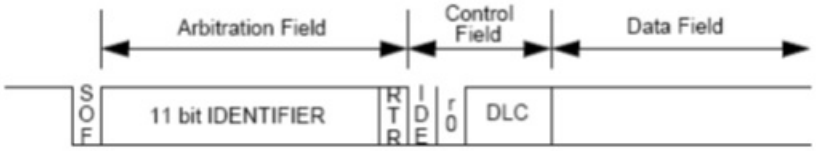
Vùng bắt đầu khung SOF (Start of frame):
· SOF đánh dấu sự bắt đầu của một khung dữ liệu. nó chỉ bao gồm một bit dominant.
· Một trạm chỉ được cho phép truyền khi bus rảnh (Bus idle). Khi bus rảnh thì trạng thái trên bus đang là recessive. Lúc này, nếu có một trạm bắt đầu truyền thì nó sẽ truyền SOF trước. Ngay khi xuất hiện cạnh xuống (chuyển từ recessive thành dominant) của SOF thì tất cả các trạm được đồng bộ theo cạnh này.



Vùng Start Of Frame (SOF)

Vùng phân xử (Arbitration field):
Định dạng vùng phân xử là khác nhau đối với dạng khung chuẩn và dạng khung mở rộng:
· Với định dạng khung chuẩn: vùng phân xử có độ dài 12 bit, bao gồm 11 bit IDENTIFIER và 1 bit RTR.

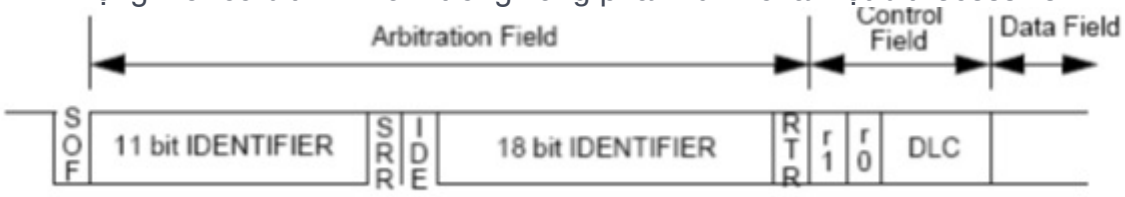
- IDENTIFIER (phần định danh – ID): gồm 11 bit tương ứng với 11 bit IDENTIFIER cơ bản (Base ID) của định dạng khung mở rộng. Các bit này được truyền từ ID-28 đến ID-18 với ID-18 là bit có trọng số thấp nhất LSB (least significant bit). Bảy bit có trọng số cao nhất (ID-28 đến ID-22) không được phép cùng là recessive.
- Bit RTR (Remote Transmission Request): có ý nghĩa là bit yêu cầu truyền từ xa. Trong khung dữ liệu, bit RTR là dominant. Trong khung yêu cầu, nó là bit recessive.



Vùng phân xử của định dạng khung chuẩn

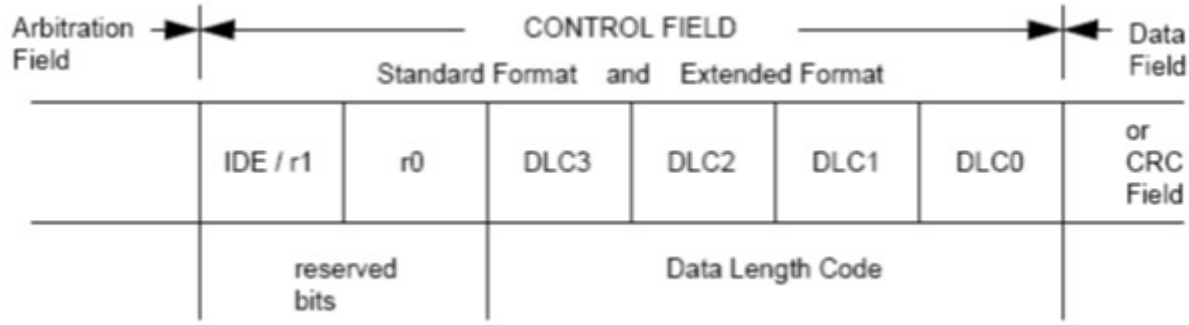
· Với định dạng khung mở rộng: vùng có độ dài 32 bit, bao gồm 29 bit IDENTIFIER, 1 bit SRR, 1 bit IDE, 1 bit RTR.

- IDENTIFIER: Gồm 29 và được chia làm hai phần là IDENTIFIER cơ bản (Base ID) với 11 bit và IDENTIFIER mở rộng (Extended ID) với 18 bit. Trong đó, Base ID là phần quyết định mức ưu tiên cơ bản của khung dữ liệu mở rộng.
- Bit RTR: như khung chuẩn
- Bit SRR (Substitute Remote Request): là bit recessive. Vị trí của nó tương ứng với vị trí của bit RTR trong khung dữ liệu chuẩn. Khi có sự xung đột xảy ra giữa khung dữ liệu chuẩn và khung dữ liệu mở rộng mà Base ID của khung dữ liệu mở rộng lại giống IDENTIFIER của khung chuẩn thì khung dữ liệu chuẩn sẽ chiếm ưu thế hơn khung dữ liệu mở rộng. Có điều này là do giá trị bus được xác định theo phương pháp AND-wire nên khi so sánh bit RTR của khung chuẩn (bit dominant) và bit SRR của khung mở rộng (bit recessive) thì bit RTR của khung chuẩn sẽ chiếm đường bus.
- Bit IDE (Identifier Extension): Nằm ngay sau bit SRR và chỉ có khung mở rộng mới có bit IDE nằm trong vùng phân xử. Nó là một bit recessive.



Vùng phân xử của định dạng khung mở rộng

Vùng điều khiển (Control field):



Vùng điều khiển của khung dữ liệu

· Vùng điều khiển bao gồm 6 bit. Định dạng của vùng này là khác nhau đối với khung chuẩn và khung mở rộng.

- Với định dạng khung chuẩn: Bit đầu tiên là bit IDE được truyền dạng dominant, bit thứ hai là bit dành riêng r0 thì tùy ý. Cuối cùng là 4 bit DLC (Data Length Code) chỉ ra độ dài dữ liệu trong khung dữ liệu.
- Với định dạng khung mở rộng: Hai bit đầu tiên là hai bit dành riêng r1 và r0. Hai bit này thường gửi dạng dominant nhưng các bộ nhận cho phép hai bit này có thể tùy ý là dominant hoặc recessive. Cuối cùng là 4 bit DLC.

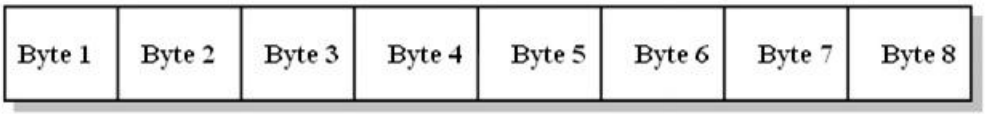
· DCL (Data Length Code) là mã 4 bit chỉ độ dài dữ liệu. Nó chỉ ra số byte dữ liệu chứa trong vùng dữ liệu. Giá trị thấp nhất là 0 (tức không có byte dữ liệu nào trong khung dữ liệu) và giá trị cao nhất là 8 (tức có 8 byte dữ liệu trong khung dữ liệu).

Số byte dữ liệu	Mã chỉ độ dài dữ liệu			
	DLC3	DLC2	DLC1	DLC0
0	d	d	d	d
1	d	d	d	r
2	d	d	r	d
3	d	d	r	r
4	d	r	d	d
5	d	r	d	r
6	d	r	r	d
7	d	r	r	r
8	r	d	d	d

Bảng mã chỉ độ dài dữ liệu DLC

· Trong bảng 1-2, d = ‘dominant’, r = ‘recessive’. Các giá trị khác không cần sử dụng.

Vùng dữ liệu (Data field):



Vùng dữ liệu

· Vùng này chứa số byte dữ liệu từ {0,1,2,….,8} tùy vào mã chỉ độ dài DLC ở vùng điều khiển nằm ngay trước đó. Mỗi byte dữ liệu bao gồm 8 bit và bit MSB được truyền trước.

Vùng kiểm tra (CRC field):

· Vùng kiểm tra hay vùng CRC gồm 16 bit và được chia làm hai phần là phần

tuần tự CRC (CRC Sequence) và phần phân cách CRC (CRC Delimiter)

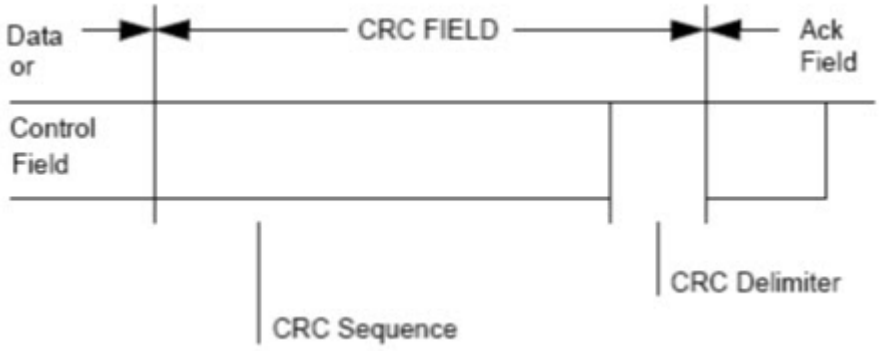
- CRC Sequence: gồm 15 bit CRC tuần tự. Mọi tính toán cho CRC sequence thực chất là phép chia đa thức (hay chia bằng các bit nhi phân) và đều dùng modulo-2. Ta thực hiện tính toán cho CRC sequence như sau:
- Đa thức sinh (đa thức chia) là:

  H(X) = X^15 + X^14 + X^10 + X^8 + X^7 + X^4 + X^3 + 1
- Đa thức của chuỗi bit được kiểm tra CRC (đa thức bị chia) nằm từ vùng SOF đến vùng dữ liệu, đa thức chia không tính đến các bit được chèn thêm (quy luật chèn thêm bit sẽ được nói sau trong phần mã hóa bit). Gọi đa thức bị chia là: Z(X)
- Dịch trái đa thức bị chia 15 bit, tức là thực hiện phép nhân đa thức sau:

  X^15.Z(X)
- Lấy đa thức bị chia đã được dịch trái chia cho đa thức sinh:

$$\frac{X^{15}.Z(X)}{H(X)}$$

- Gọi số dư của phép chia trên là R(X), R(X) chính là 15 bit tuần tự chứa trong CRC sequence.
- Mã kiểm tra CRC phù hợp nhất cho các khung mà chuỗi bit được kiểm tra có chiều dài dưới 127 bit, mã này thích hợp cho việc phát hiện các trường hợp sai nhóm (burst error). Ở đây, tổng bit từ vùng SOF đến vùng dữ liệu tối đa là 83 bit (khung định dạng chuẩn) và 103 bit (khung định dạng mở rộng).
- Bộ nhận cũng sẽ tính toán CRC như bộ truyền khi đã nhận dữ liệu và so sánh kết quả đó với CRC sequence mà nó đã nhận được, nếu khác nhau tức là đã có lỗi, nếu giống nhau tức là đã nhận đúng.
- CRC delimiter: theo ngay sau CRC sequence, nó là một bit recessive làm nhiệm vụ phân cách vùng CRC với vùng ACK.



Vùng kiểm tra (hay vùng CRC)

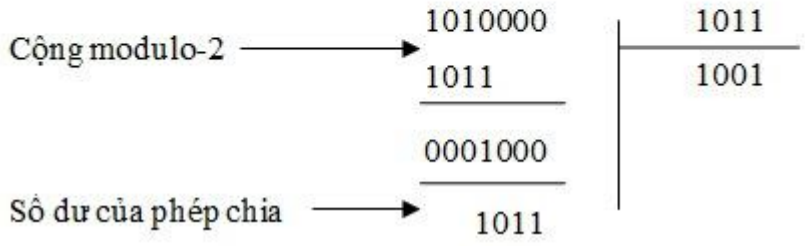
· Một ví dụ đơn giản về tính toán CRC (ở ví dụ này đa thức sinh được lấy khác để tiện tính toán nhưng với bất kỳ đa thức sinh hay chuỗi dữ liệu nào thì cách làm cũng như nhau) :

- Cho chuỗi dữ liệu là (MSB)1010 (LSB).
- Đa thức bị chia tương ứng là: Z(X) = X3 + X
- Cho đa thức sinh là: H(X) = X3 + X + 1.
- Chuỗi bit tương ứng là: (MSB) 1011 (LSB)
- Dịch trái 3 bit (dịch bằng trọng số cao nhất của đa thức sinh):

  X3.Z(X) = X3(X3 + X) = X6 + X4
- Vậy chuỗi bit tương ứng sau khi dịch là:

  (MSB) 1010000 (LSB)
- Thực hiện phép chia như sau:

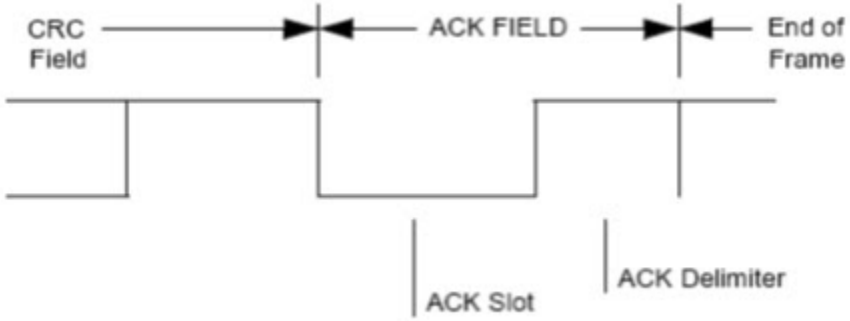
$$\frac{X^3.Z(X)}{H(X)} \Leftrightarrow \frac{1010000}{1011}$$



- Cuối cùng ta có số dư là **11**, tức **R(X) = X + 1**

Vùng báo nhận (ACK field):

· Vùng báo nhận hay vùng ACK có độ dài 2 bit và bao gồm hai phần là 'khe ACK' (ACK slot) và phần phân cách ACK (ACK delimiter)



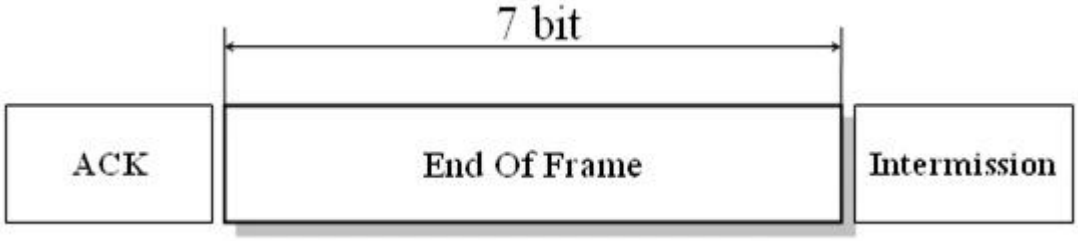
Vùng ACK

· ACK slot: có độ dài 1 bit, một trạm truyền dữ liệu sẽ thiết lập bit này là recessive. Khi một bộ nhận nhận chính xác giá trị thông điệp (không lỗi và so sánh CRC sequence trùng khớp) thì nó sẽ báo lại cho bộ truyền bằng cách đặt bit ACK slot là dominant.

· ACK delimiter: có độ dài 1 bit, nó luôn là một bit recessive. Như vậy, ta thấy rằng ACK slot luôn được đặt giữa hai bit recessive là CRC delimiter và ACK delimiter.

Vùng kết thúc khung EOF (End Of Frame):

· Vùng EOF là vùng thông báo kết thúc một khung dữ liệu và phân cách giữa các khung dữ liệu hay các khung yêu cầu. Vùng này gồm 7 bit recessive.

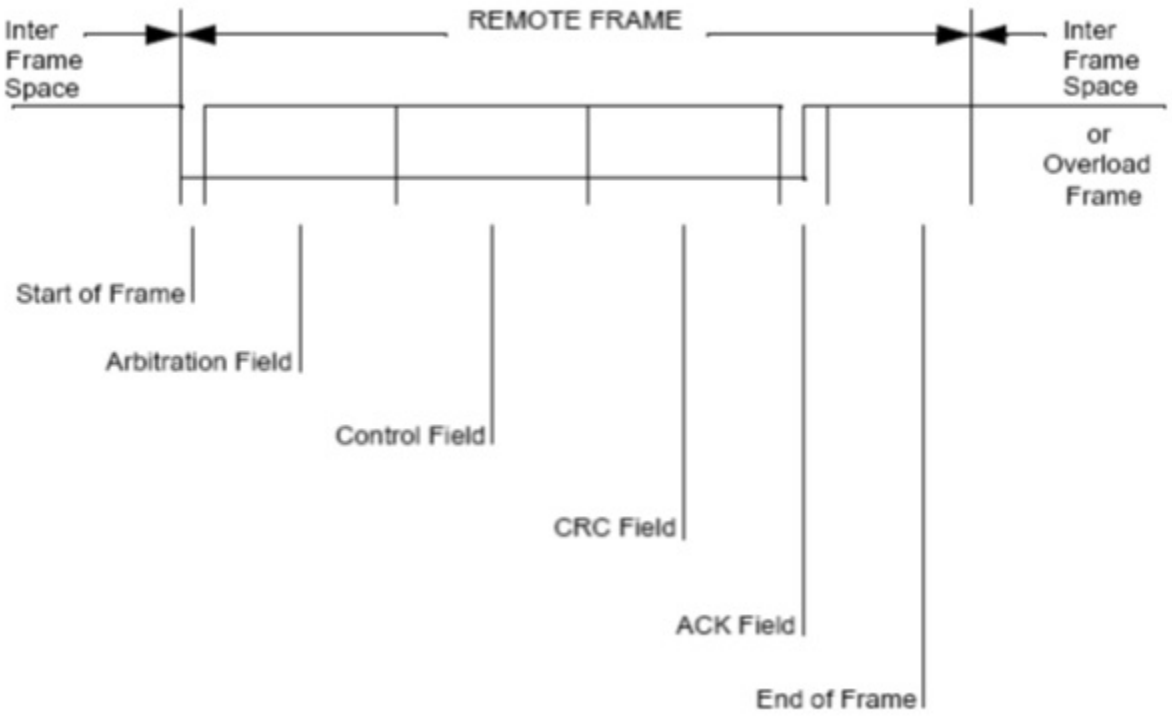


Vùng EOF

Còn tiếp (to be continued ...)

Khung yêu cầu (Remote Frame)

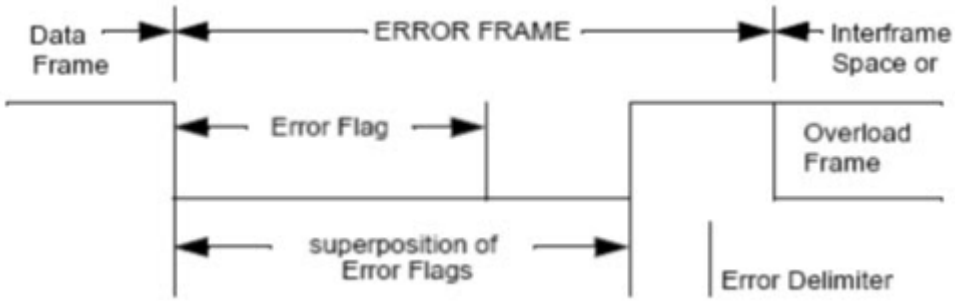
- Một trạm hoạt động như một bộ nhận mong muốn một dữ liệu đã biết, để nhận dữ liệu này từ nút nguồn thì bộ nhận sẽ gửi một khung yêu cầu (còn gọi là khung điều khiển). Khung yêu cầu có tác dụng thông báo cho nút nguồn biết để nút này truyền dữ liệu.
- Một khung yêu cầu có thể là định dạng chuẩn hay định dạng mở rộng. Dù thuộc định dạng nào thì khung điều khiển cũng gồm sáu vùng bit khác nhau là: Vùng bắt đầu khung (Start Of Frame), vùng phân xử (Arbitration Field), vùng điều khiển (Control Field), vùng kiểm tra (CRC Field), vùng ACK (ACK Field), vùng kết thúc khung (End Of Frame).
- Cấu tạo và ý nghĩa các vùng trong khung điều khiển giống như ở khung dữ liệu ngoại trừ một số điểm sau đây là khác so với khung dữ liệu:
    - Khung yêu cầu không có vùng dữ liệu (Data Field). Chính vì vậy, khung yêu cầu không phụ thuộc vào giá trị của mã chỉ độ dài dữ liệu DLC, DLC có thể nhận bất kỳ giá trị nào từ 0 đến 8. Nhưng giá trị DLC của khung yêu cầu lại chỉ ra độ dài dữ liệu trong khung dữ liệu tương ứng sẽ trả về, tức DLC của khung dữ liệu trả về giống DLC của khung yêu cầu đã gửi.
    - Bit RTR (thuộc vùng phân xử) của khung yêu cầu là bit recessive. Điều này khác với khung dữ liệu vì ở khung dữ liệu nó là một bit dominant.



Cấu trúc một khung yêu cầu (Remote Frame)

Khung lỗi (Error Frame)

Khung lỗi gồm hai vùng khác nhau. Vùng đầu tiên (superposition of Error Flags) là sự chồng chập của các cờ lỗi từ các trạm khác nhau trên bus. Tiếp theo đó là vùng phân cách khung lỗi (Error delimiter)



Cấu trúc khung lỗi (Error frame)

Cờ lỗi (Error Flag): Có hại dạng cờ lỗi là cờ lỗi chủ động (Active Error Flag) và cờ lỗi bị động (Passive Error Flag), trong đó:

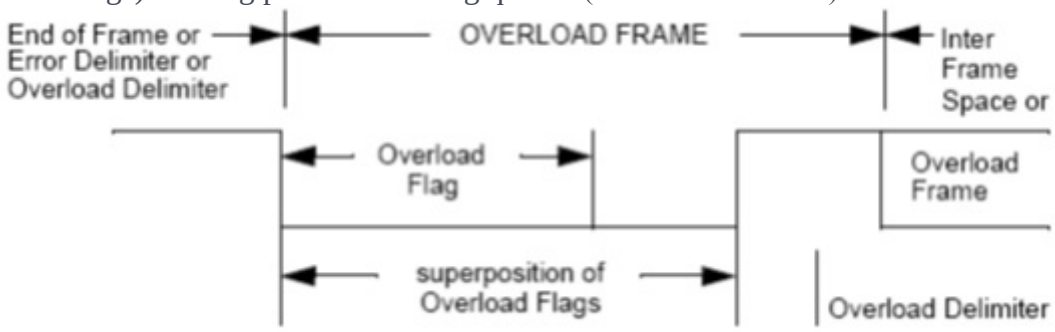
- Cờ lỗi chủ động gồm 6 bit dominant liên tiếp.
- Cờ lỗi bị động gồm 6 bit recessive liên tiếp. Cờ lỗi này có thể bị ghi đè bởi các bit dominant từ các nút khác (vì chế độ AND-wire)

Để kết thúc chính xác một khung lỗi, một nút ‘error passive’ (tức là nút rơi vào trạng thái bị động do lỗi, đây là một trong các trạng thái lỗi sẽ được mô tả trong phần các trạng thái

lỗi) có thể phải cần bus rảnh (bus idle) trong ít nhất 3 thời gian bit (bit time) nếu đây là một lỗi cục bộ xảy ra ở một bộ nhận 'error passive'.
Một trạm 'error active' (tức là trạm vẫn ở trạng thái chủ động khi lỗi) khi phát hiện điều kiện lỗi sẽ báo hiệu bằng cách gửi một cờ lỗi chủ động. Dạng của cờ lỗi vi phạm luật chèn bit (Bit Stuffing, sẽ được trình bày sau) được dùng cho tất cả các vùng từ SOF đến CRC delimiter hay phá hủy định dạng cố định của vùng ACK hay EOF. Kết quả là tất cả các trạm khác phát hiện một điều kiện lỗi và chúng sẽ bắt đầu truyền một cờ lỗi.
Như vậy, trên thực tế, dãy bit dominant có thể được giám sát (hay lắng nghe) trên bus từ sự chồng chập của các cờ lỗi khác nhau được truyền từ các trạm riêng lẻ. Tổng độ dài của dãy này thay đổi trong phạm vi tối thiểu là 6 bit, tối đa là 12 bit.
Một trạm 'error passive' phát hiện một điều kiện lỗi sẽ cố gắng báo hiệu bằng cách gửi một cờ lỗi bị động. Trạm 'error passive' chờ cho đến khi có đủ 6 bit recessive liên tiếp.
Phần phân cách khung lỗi (Error delimiter): gồm 8 bit recessive. Sau khi truyền một cờ lỗi, trạm sẽ gửi đi bit recessive và giám sát bus cho đến khi nó phát hiện một bit recessive. Ngay sau đấy, trạm bắt đầu gửi tiếp 7 bit recessive.

Khung báo quá tải (Overload Frame)

Khung quá tải bao gồm hai vùng là vùng chồng chập của các cờ lỗi (superposition of overload flags) và vùng phân cách khung quá tải (overload delimiter).



Cấu trúc khung quá tải (Overload Frame)

Các điều kiện qua tải:
[1] Do điều kiện nội của bộ nhận cần có một độ trễ cho khung dữ liệu hay khung yêu cầu kế tiếp.
[2] Phát hiện một bit dominant ở bit đầu tiên hoặc bit thứ hai của vùng Intermission (vùng ngừng trong khoảng liên khung).
[3] Nếu một nút CAN lấy mẫu được một bit dominant ở bit thứ tám (bit cuối cùng) của vùng phân cách khung lỗi (thuộc khung lỗi) hoặc của vùng phân cách khung qua tải (thuộc khung qua tải), nó sẽ bắt đầu truyền một khung quá tải (nút CAN không xác định đây là một lỗi, vì vậy không truyền khung lỗi trong trường hợp này). Các bộ đếm lỗi (Error Counters) sẽ không được tăng thêm.
Một khung quá tải được gửi do điều kiện quá tải [1] thì chỉ được phép bắt đầu truyền trong thời gian bit thứ nhất của một vùng Intermission. Trong khi đó, Khung quá tải do điều kiện [2] và [3] gây ra được truyền ngay sau khi phát hiện bit dominant vi phạm.
Tối đa chỉ có hai khung quá tải được phát sinh với mục đích tạo trễ cho khung dữ liệu hoặc khung yêu cầu kế tiếp.

Cờ báo quá tải (Overload flag)

- Cờ bao gồm 6 bit dominant. Nhìn chung, cờ báo quá tải thể hiện giống cờ lỗi chủ động.
- Dạng cờ báo quá tải làm mất định dạng cố định của vùng Intermission. Như vậy, tất cả các trạm khác sẽ phát hiện ra điều kiện qua tải và chúng sẽ truyền một cờ báo quá tải. Trong trường hợp có một bit dominant được phát hiện trong thời gian của bit thứ ba của vùng Intermission thì các nút sẽ hiểu bit đó như một SOF.

Vùng phân cách khung quá tải (Overload delimiter)

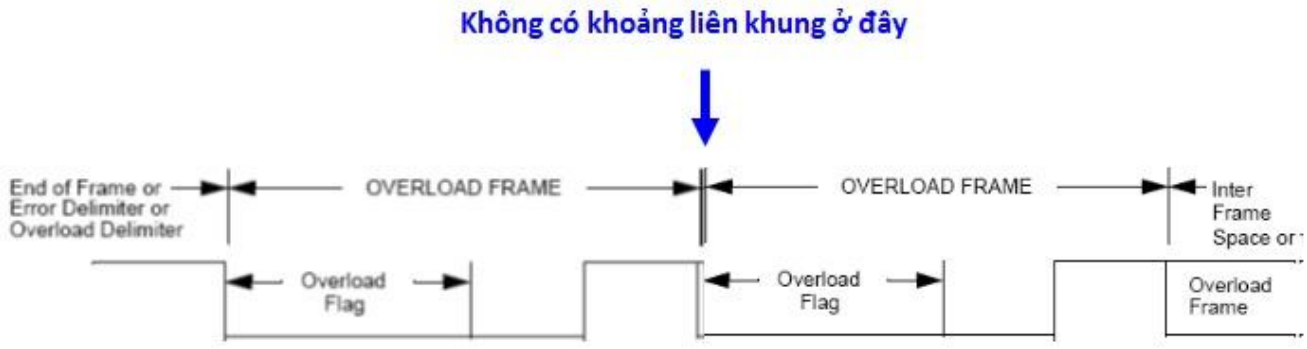
- Vùng này bao gồm 8 bit recessive.
- Vùng phân cách khung quá tải cũng giống như dạng của của vùng phân cách khung lỗi. Sau khi đã truyền một cờ báo quá tải, trạm sẽ truyền tiếp một bit recessive và giám sát bus cho đến khi phát hiện một sự chuyển đổi từ bit dominant thành bit recessive. Ngay tại điểm thời gian này, các trạm khác trên bus đã hoàn thành việc gửi cờ báo quá tải của nó và các trạm sẽ bắt đầu cùng truyền tiếp 7 bit recessive tiếp theo.

Khoảng liên khung (Interframe Space)

Các khung dữ liệu và các khung điều khiển được tách biệt với nhau bằng một vùng bit gọi là khoảng liên khung. Các khung trước đó có thể là bất kỳ loại khung nào như: khung dữ liệu, khung yêu cầu, khung báo lỗi hay khung quá tải.

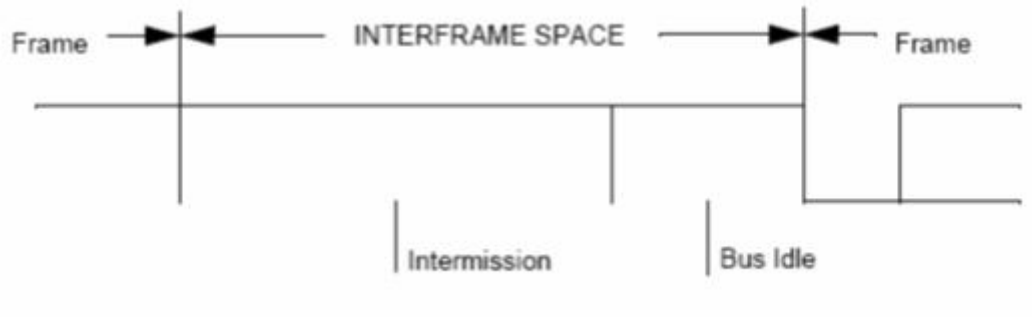
Các khung quá tải và các khung lỗi không dược phép theo sau một khoảng liên khung (tức không có khoảng liên khung đứng trước hai loại khung này).

Nhiều khung quá tải liên tiếp không được phân cách bởi khoảng liên khung mà các khung quá tải này sẽ được phát liên tục nối tiếp với nhau.



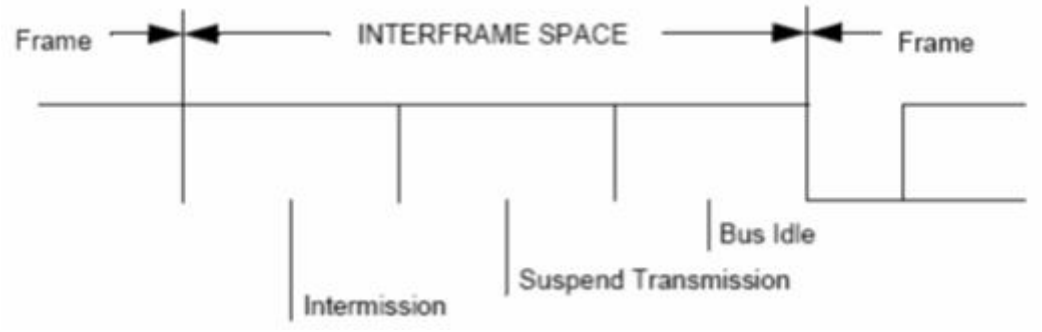
Giữa các khung quá tải không có khoảng liên khung

Cấu trúc khoảng liên khung dành cho các trạm không ‘error passive’ hay các trạm là bộ nhận của thông điệp trước đó như sau:



Khoảng liên khung dành cho các trạm không ‘error passive’

Cấu trúc khoảng liên khung dành cho các trạm ‘error passive’, nó là bộ truyền của thông điệp trước đó như sau:



Khoảng liên khung cho trạm ‘error passive’

Đặc điểm của các vùng trong khoảng liên khung được trình bày sau đây
Intermission:

- Vùng gồm 3 bit recessive.
- Trong suốt khoảng thời gian xảy ra vùng này, chỉ cho phép duy nhất hoạt động báo hiệu quá tải nếu có điều kiện quá tải xảy ra. Không trạm nào được phép bắt đầu hoạt động truyền một khung dữ liệu hay khung yêu cầu.
- Một điểm quan trọng cần lưu ý, Nếu một nút CAN có một thông điệp đang chờ để truyền và nó lấy mẫu được một giá trị bit dominant ở bit thứ 3 của intermission thì nó sẽ hiểu đó như là một bit Start Of Frame (SOF). Như vậy, nút này sẽ bắt đầu truyền thông điệp của nó ngay sau khi phát hiện bit dominant trên nhưng nó không truyền SOF nữa mà sẽ truyền bắt đầu truyền từ bit đầu tiên của IDENTIFIER (thuộc vùng phân xử). Hiển nhiên lúc này nút không trở thành bộ nhận.

Bus Idle:

- Bus idle được thiết lập ở recessive.
- Chu kỳ của Bus idle có độ dài tùy ý. Lúc này bus ở trạng thái tự do (trạng thái rảnh), tức không có bất kỳ dữ liệu nào được truyền trên bus. Một trạm khi có thông điệp cần truyền thì có thể truy cập bus trong khoảng thời gian này.

- Một thông điệp đang trì hoãn truyền do một thông điệp khác đang được truyền trên bus thì có thể được bắt đầu truyền ngay ở bit đầu tiên sau vùng Intermission.
- Trong thời gian Bus idle, nếu một bit dominant được phát hiện thì nó được hiểu là SOF.

Suspend Transmission (vùng tạm ngừng truyền):

- Sau khi một trạm ‘error passive’ đã truyền xong một thông điệp, nó gửi 8 bit recessive theo sau vùng Intermission trước khi bắt đầu truyền thông điệp tiếp theo hay chuyển bus đến trạng thái rảnh.
- Nếu trong thời gian này, một trạm khác khởi động truyền thì trạm ‘error passive’ sẽ trở thành bộ nhận thông điệp đó.

Định nghĩa về bộ truyền/ bộ nhận (Transmitter/ Receiver)

Không giống như các giao thức bus truyền nối khác (như I2C, SPI). Trong một bus CAN không có khái niệm master/slave mà các nút trong mạng, về nguyên tắc, đều có vai trò như nhau. Nghĩa là bất cứ khi nào thấy bus rảnh (IDLE) thì các nút đều có thể truyền thông điệp của mình trên bus. Vì vậy, trong mạng CAN chỉ có khái niệm nút truyền/nút nhận.

- Bộ truyền (Transmitter): Một đơn vị là nơi khởi nguồn của thông điệp thì được gọi là bộ truyền của thông điệp đó. Một đơn vị được giữ là bộ truyền cho đến khi bus rảnh hay đơn vị đó bị thua trong quá trình phân xử (quá trình phân xử là quá trình xảy ra khi có nhiều đơn vị cùng truyền thông điệp trong cùng một thời gian).
- Bộ nhận (Receiver): Một đơn vị được gọi là bộ nhận của thông điệp nếu nó không là bộ truyền của thông điệp khác và bus không rảnh.

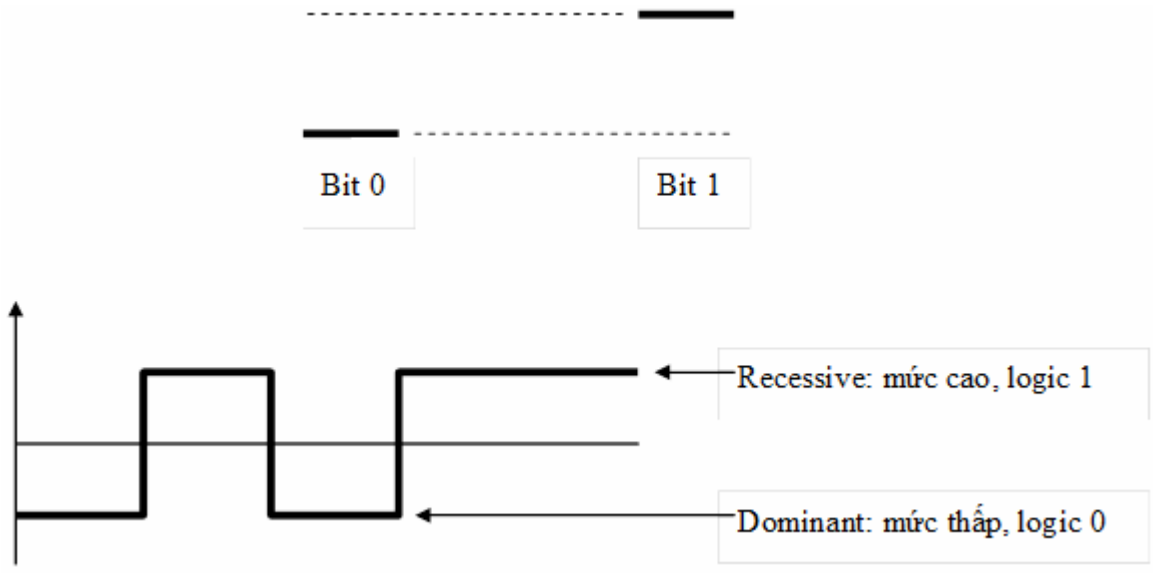
Như vậy, nút truyền là nút chiếm được bus trước các nút khác hoặc thắng trong sự phân xử khi có nhiều nút truyền thông điệp cùng lúc. Ngược lại nút nhận là nút "chậm" hơn các nút khác, không kịp chiếm bus khi bus rảnh hoặc thu trong sự phân xử.

Một nút được gọi là "nút nhận" không có nghĩa là nó phải lấy dữ liệu của khung thông điệp đang truyền trên bus mà nó chỉ giám sát thông điệp đang được nút khác truyền trên bus và kiểm tra xem đó có phải là thông điệp của mình không. Nếu đúng, nút nhận sẽ lưu lại thông điệp này, nếu không đúng, nút nhận sẽ loại bỏ thông điệp này.

Mã hóa bit được sử dụng trong giao thức CAN

- Mã hóa loại Non-Return-to-Zero (NRZ)

Phương pháp mã hóa được sử dụng trong giao thức CAN là phương pháp NRZ mà cụ thể hơn là NRZ-L (Non-Return-to-Zero Level). Đây là dạng mã hóa theo mức, tức là mỗi trạng thái dominant và recesive sẽ được quy định mang một mức áp khác nhau. Cụ thể, dominant là mức thấp tương ứng với logic 0, recessive là mức cao tương ứng với logic 1

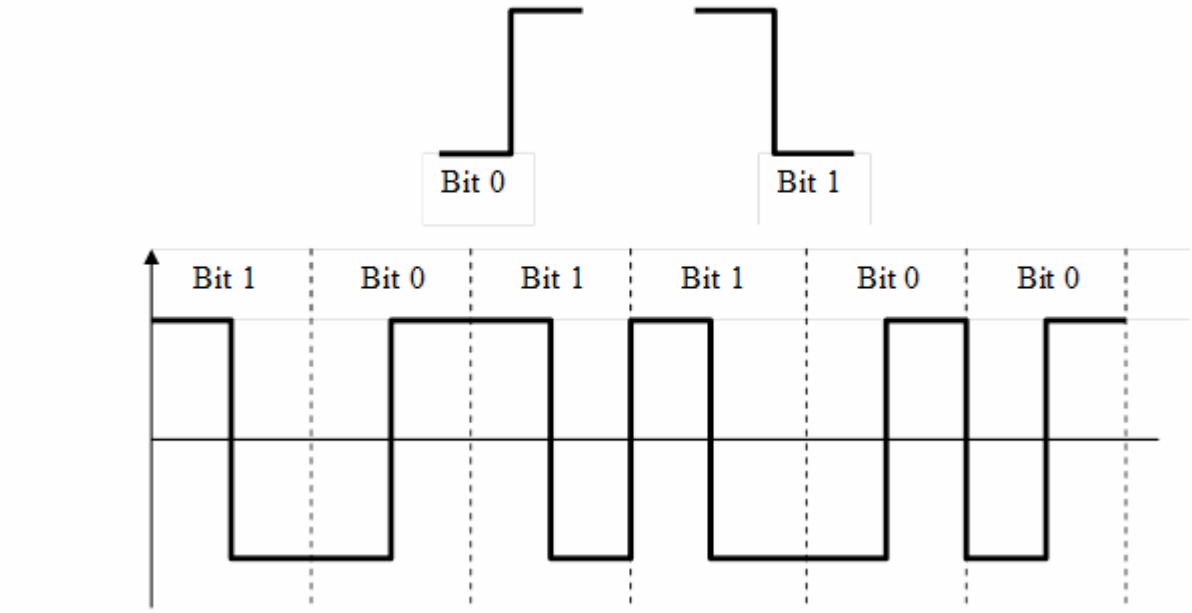


Mã hóa Non-Return-to-Zero

Đây là loại mã hóa đơn giản, thông dụng nhưng lại khó khăn cho vấn đề đồng bộ khi có nhiều bít 0 hay bit 1 xuất hiện liên tiếp. Để có được sự đồng bộ cao hơn người ta còn có thể sử dụng mã hóa Manchester.

- Mã hóa Manchester

Mã hóa Manchester là loại mã hóa có sự chuyển mức tại điểm giữa mỗi bit. Cụ thể, bit 1 là sự biến đổi trạng thái từ mức cao xuống mức thấp, bit 0 là sự biến đổi trạng thái từ mức thấp lên mức cao.

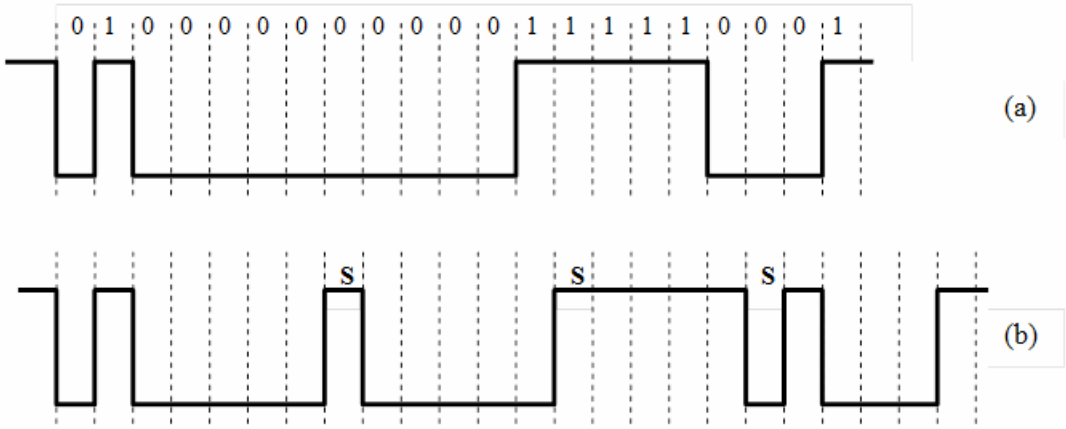


Mã hóa Manchester

Loại mã hóa này đạt được sự đồng bộ tốt hơn NRZ chính là nhờ sự đảo mức tại điểm giữa mỗi bit.

Quy luật chèn bit (Bit Stuffing)

Loại mã hóa được dùng chính thức trong giao thức CAN là NRZ. Chính vì sử dụng loại mã hóa này nên để đảm bảo tốt vấn đề đồng bộ khi xuất hiện quá nhiều bit 0 hay bit 1 liên tiếp ta phải tuân thủ theo luật chèn bit như sau: Bất cứ khi nào một bộ truyền phát hiện có 5 bit liên tiếp có giá trị giống nhau trong chuỗi bit được truyền thì nó sẽ tự động thêm một bit (gọi là bit chèn hay bit chỉnh) vào ngay sau 5 bit kia và bit chèn này đảo mức so với 5 bit trước đó.
Tại bộ nhận, bit chèn sẽ được loại bỏ để lấy thông tin chính xác được truyền.
Sau đây là một ví dụ về luật chèn bit



Luật chèn bit (a) Chuỗi bit cần truyền (b) Chuỗi bit được truyền với S là vị trí bit chèn

Quy luật chèn bit chỉ sử dụng cho các vùng: SOF, vùng phân xử (Arbitration Field), vùng điều khiển (Control Field), vùng dữ liệu (Data Field và CRC Sequence (15 bit đầu tiên của vùng kiểm tra CRC).
Các vùng còn lại của khung dữ liệu hay khung điều khiển (như CRC delimiter, vùng ACK, EOF) được giữ cố định và không áp dụng luật chèn bit. Tương tự như vậy. khung lỗi và khung quá tải là hai loại khung có định dạng cố định và không áp dụng luật chèn bit.

Tỉ số bit sử dụng trên bit chèn (User-bit/ Stuff-bit)

Đầu tiên khi xem luật chèn bit, ta nghĩ rằng tỉ số “bit sử dụng / bit chèn” được ước lượng là 5:1. Nhưng thực tế trong trường hợp xấu nhất, số bit được chèn tối đa được tính toán như sau:

s_max = (n – 1) : 4 với n là số bit trong vùng được phép chèn

Ví dụ, ta tính toán trong trường hợp xấu nhất cho khung dữ liệu định dạng chuẩn:

Số bit SOF:	1
Số bit vùng phân xử:	12
Số bit vùng điều khiển:	6
Số bit vùng dữ liệu:	$8 \times DLC$
Số bit CRC Sequence:	15

Tổng: $n = (34 + 8 \times DLC)$

Sau khi tính toán ta có bảng giá trị cụ thể sau (sau khi tính chỉ lấy phần nguyên):

dlc	s_max
0	8
1	10
2	12
3	14
4	16
5	18
6	20
7	22
8	24

Bảng tính số bit chèn tối đa cho khung dữ liệu định dạng chuẩn

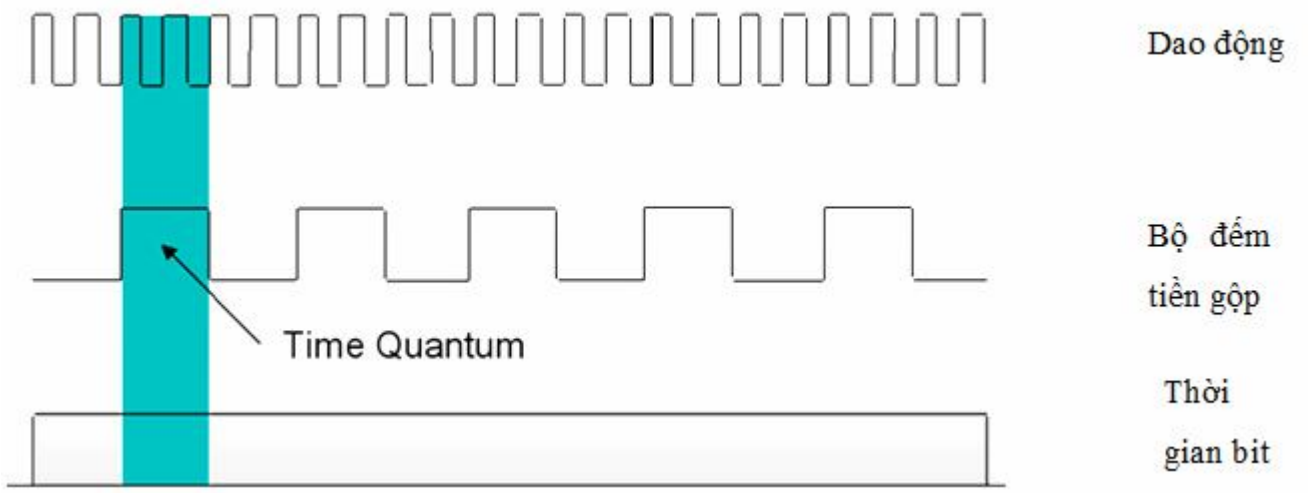
Định thời bit (Bit Timing)

- **Tốc độ bit danh định:**

Tốc độ bit danh định (Nominal Bit Rate) là số bit được truyền trong một giây bằng bộ truyền lý tưởng và không có sự tái đồngbộ xảy ra.
Tốc độ bit danh định có thể hiểu là tốc độ bit cao nhất mà đường truyền có thể đạt được trên lý thuyết.

- **Lượng tử thời gian:**

Lượng tử thời gian (Time Quantum) là một đơn vị cố định về thời gian được suy ra từ chu kỳ dao động.
Thông qua một bộ đếm tiền gộp lập trình được (programmable prescaler) có giá trị nguyên trong tầm từ 1 đến 32. Lượng tử thời gian nhỏ nhất (Minimum Time Quantum) tương ứng bằng một chu kỳ dao động, vậy lượng tử thời gian được tính như sau:
Time Quantum (Tq) = m × Minimum Time Quantum với m = [1;32] là giá trị bộ đếm tiền gộp.



Mô tả lượng tử thời gian

- **Thời gian bit danh định:**

Thời gian bit danh định (Nominal Bit Time) được định nghĩa như sau:

$$\text{Nominal Bit Time} = \frac{1}{\text{Nominal Bit Rate}}$$

Thời gian bit danh định được chia thành 4 đoạn không riêng biệt không phủ chồng lên nhau là: đoạn đồng bộ (Synchronization Segment) ký hiệu là Sync_seg, đoạn lan truyền (Propagation Time Segment) ký hiệu là Prop_seg, đoạn đệm pha thứ nhất (Phase Buffer Segment 1) ký hiệu là Phase_seg1, đoạn đệm pha thứ hai (Phase Buffer Segment 2) ký hiệu là Phase_seg2.

Thời gian bit danh định

- **Đặc điểm của các đoạn trong thời gian bit:**
- Sync_seg: có độ dài 1 Tq (Tq là lượng tử thời gian). Phần này trong thời gian bit được sử dụng để đồng bộ các nút khác nhau trên bus. Một cạnh bit xuất hiện sẽ được mong đợi nằm trong đoạn này.
- Prop_seg: có độ dài từ 1 đến 8 Tq. Phần này dùng để bù cho thời gian trễ vật lý trong mạng. nó bằng hai lần tổng của thời gian lan truyền tín hiệu (tprop_time), trễ so với ngõ vào (**t**in_delay) và trễ lái ngõ ra (tout_delay).

$$t_{prop\_seg} = 2 \times (t_{prop\_time} + t_{in\_delay} + t_{out\_delay})$$

- Phase_seg1: có độ dài từ 1 đến 8 Tq. Phần này để bù cho lỗi pha của cạnh bit và có thể được kéo dài hơn khi tái đồng bộ.
- Phase_seg2: có độ dài là tối đa của phase_seg1 và thời gian xử lý thông tin.

Thời gian xử lý thông tin (Information processing time) có độ dài nhỏ hơn hoặc bằng 2 Tq. Đây là khoảng thời gian cần thiết để một nút có thể nhận dạng được bit (tính từ khi lấy mẫu).
Điểm lấy mẫu (Sample poit): là điểm mà tại đó mức bus được đọc và dịch ra giá trị bit tương ứng. Nó được định vị tại điểm cuối của Phase_seg1.
Tổng số lượng tử thời gian được lập trình cho một thời gian bit phải nằm trong khoảng từ 8 Tq đến 25 Tq.

Vấn đề đồng bộ

Trong CAN, chúng ta thường mong rằng các đơn vị điều khiển không sử dụng các dao động khác nhau cho các CPU cục bộ và phương tiện thông tin của nó. Vì vậy, tần số dao động của một linh kiện CAN luôn hướng đến CPU cục bộ và được xác định bằng các điều kiện của đơn vị điều khiển. Để nhận được tốc độ bit mong muốn, việc lập trình cho thời gian bit là cần thiết. Trong trường hợp sự thực thi của CAN được thiết kế để sử dụng mà không có CPU cục bộ thì thời gian bit không thể lập trình được. Nói cách khác, phương tiện đó cho phép chọn một dao động bên ngoài, đó là cách phương tiện được điều chỉnh cho thích hợp với tốc độ bit như vậy việc lập trình có thể bỏ qua cho các thành phần này.
Việc xác định điểm lấy mẫu thông qua định nghĩa thời gian bit và đồng bộ là cần thiết để các nút khác nhau có thể lấy mẫu chính xác bit đó.

Một ví dụ về quy định thời gian bit

Trong ví dụ trên, ‘Phase buffer seg 1’ có thể được kéo dài thêm và ‘Phase buffer seg 2’ có thể rút ngắn bớt khi xảy ra sự tái đồng bộ bit.

Đồng bộ cứng (Hard Synchronization)

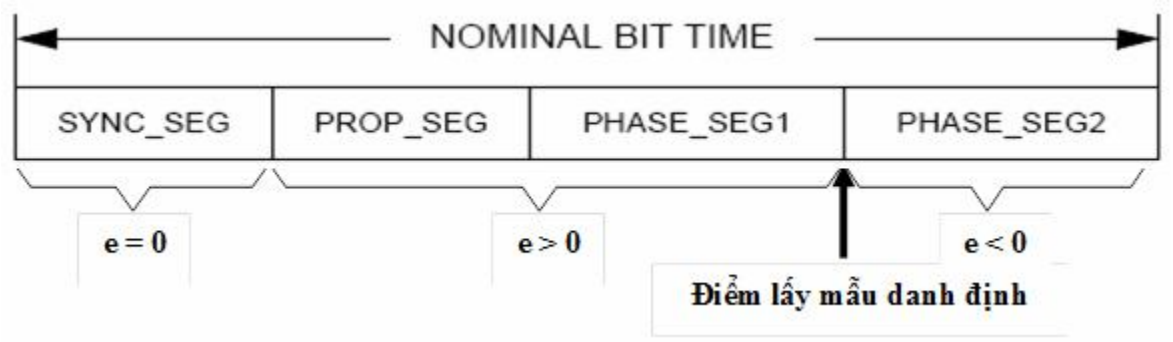
- Trong thời gian một bit, đồng bộ tác động theo cạnh bit, một đồng bộ cứng sẽ xảy ra khi cạnh bit nằm trong vùng Sync_seg. Khi có đồng bộ cứng xảy ra thì một bit sẽ được tái bắt đầu với Sync_seg.
- Trong một hệ thống bus CAN, đồng bộ cứng xảy ra khi xuất hiện một cạnh chuyển từ recessive ở trạng thái bus rảnh xuống dominant. Lúc này tất cả các nút sẽ hiểu đó là SOF.

Độ rộng bước tái đồng bộ

- Độ rộng bước tái đồng bộ (Resynchronization Jump Width – RJW) là khoảng thời gian được thêm vào ở Phase_seg1 hoặc cắt bớt ở Phase_seg2 khi xảy ra sự tái đồng bộ bit.
- Độ rộng bước tái đồng bộ được lập trình trong khoảng: [1 Tq đến min(4, Phase_seg1)]
- Thông tin về xung nhịp có thể được suy ra từ sự chuyển tiếp của một giá trị bit thành một giá trị bit khác. Đặc tính giới hạn số bit tối đa các bit liên tiếp có giá trị giống nhau (xem luật chèn bit) cung cấp khả năng tái đồng bộ cho đơn vị bus trong suốt quá trình truyền một khung. Độ dài tối đa giữa hai sự chuyển tiếp có thể được dùng cho sự tái đồng bộ là 29 thời gian bit (Bit time).

Lỗi Pha của một cạnh bit

- Lỗi pha của một cạnh bit được xác định dựa vào vị trí của cạnh bit trong một thời gian bit được đo bằng lượng tử thời gian (Tq).
- Ký hiệu của lỗi pha được định nghĩa như sau:
    - e = 0 nếu cạnh bit nằm trong vùng sync_seg (nghĩa là không có lỗi pha và xảy ra đồng bộ cứng).
    - e > 0 nếu cạnh bit nằm sau sync_seg và trước điểm lấy mẫu danh định (nghĩa là lỗi pha dương và có thể xảy ra sự tái đồng bộ bit).
    - e < 0 nếu cạnh bit nằm sau điểm lấy mẫu danh định (hay trong phase_seg2) của bit trước đó (nghĩa là lỗi pha âm và có thể xảy ra sự tái đồng bộ bit).

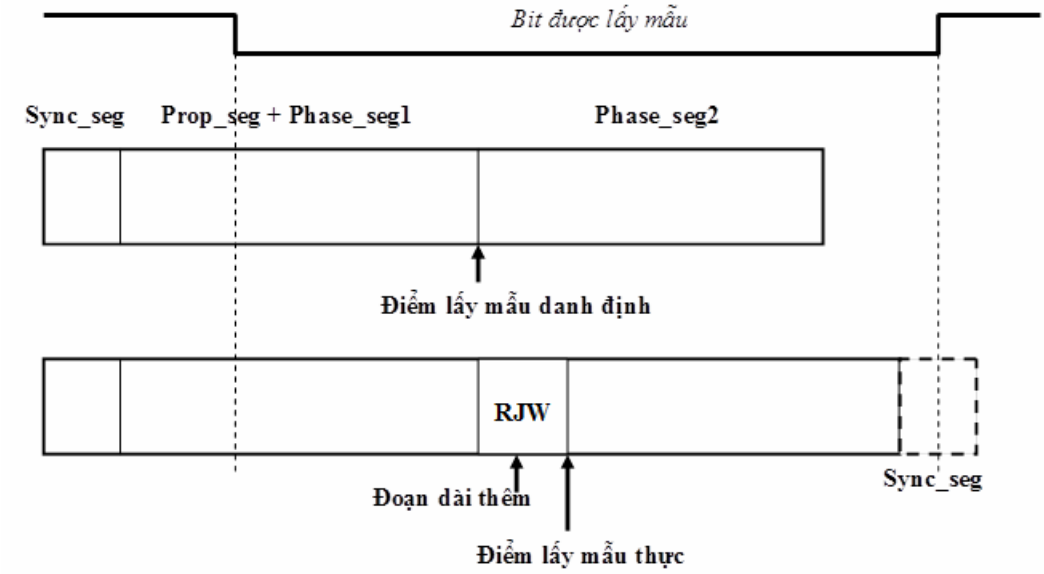


Mô tả lỗi pha dựa trên vị trí cạnh bit

Sự tái đồng bộ

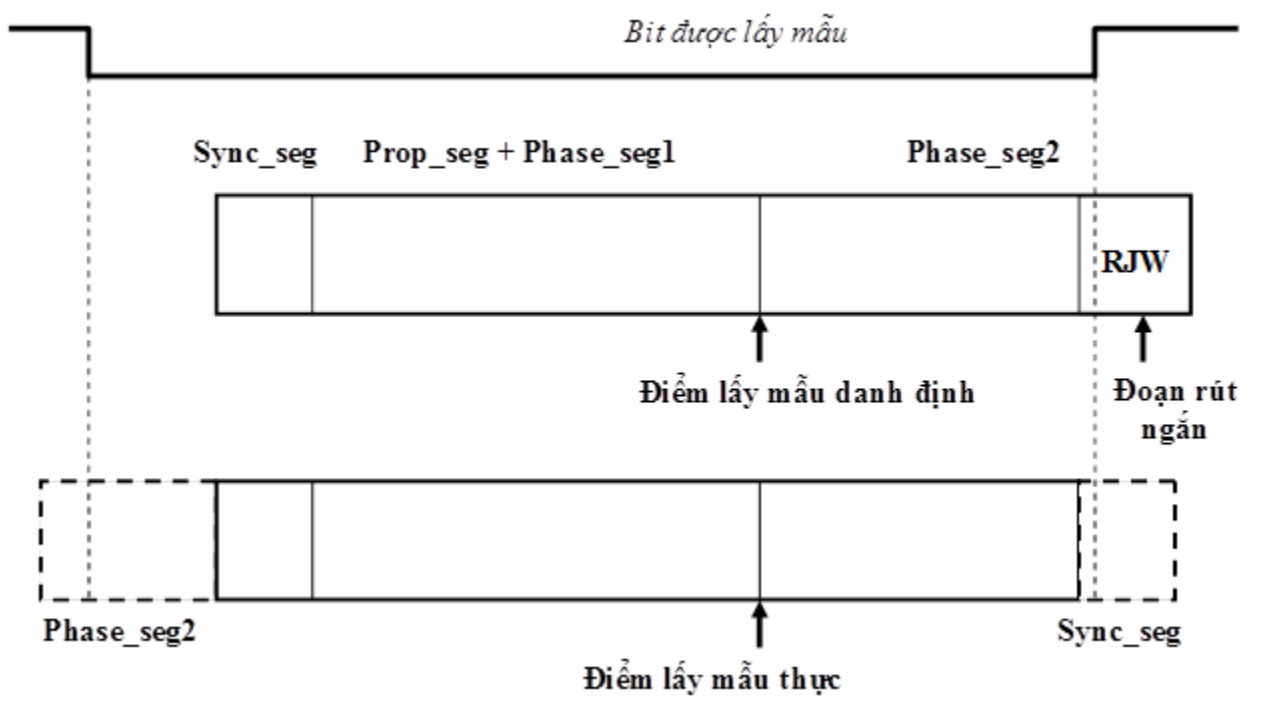
Sự tái đồng bộ (Resynchronization) xảy ra khi có lỗi pha và lỗi pha này vượt qua tầm của 'độ rộng bước tái đồng bộ' RJW. Sự tái đồng bộ được thực hiện như sau:

Nếu lỗi pha dương thì Phase_seg1 được kéo dài thêm một đoạn đúng bằng RJW.



Tái đồng bộ khi xảy ra lỗi pha dương

Nếu lỗi pha âm thì Phase_seg2 được rút ngắn một đoạn đúng bằng RJW.



Tái đồng bộ khi xảy ra lỗi pha âm

Như vậy, nếu xảy ra lỗi pha nhưng lỗi này vẫn nhỏ hơn hoặc bằng giá trị RJW đã được lập trình thì vẫn xảy ra đồng bộ cứng (lấy mẫu tại điểm lấy mẫu danh định) chứ không xảy ra tái đồng bộ.

Các quy tắc đồng bộ

Sự đồng bộ cứng và sự tái đồng bộ là hai dạng của sự đồng bộ (Synchronization) . Chúng tuân theo các quy tắc sau đây:

- [1] Chỉ có duy nhất một sự đồng bộ xảy ra trong một thời gian bit. Như vậy, trong một thời gian bit chỉ xảy ra hoặc là đồng bộ cứng, hoặc là tái đồng bộ.
- [2] Một cạnh chỉ được sử dụng để đồng bộ nếu giá trị được phát hiện ở điểm lấy mẫu trước (giá trị bus được đọc trước đó) khác giá trị bus ngay sau cạnh đó. Như vậy, sự đồng bộ chỉ thực hiện được khi xuất hiện sự chuyển ‘recessive thành dominant’ hoặc ‘dominant thành recessive’. Chính vì vậy, khi xuất hiện những chuỗi bit liên tiếp có giá trị giống nhau kéo dài thì có thể gây ra sự mất đồng bộ. CAN khắc phụ điều này bằng ‘luật chèn bit’.
- [3] Sự đồng bộ cứng được thực hiện bất cứ khi nào có sự chuyển ‘recessive thành dominant’ trong suốt thời gian bus rảnh (Bus Idle).
- [4] Tất cả các cạnh chuyển ‘recessive thành dominant’ khác thi hành theo luật [1] và [2] sẽ được sử dụng cho sự tái đồng bộ. Ngoài ra, nếu chỉ có các cạnh chuyển ‘recessive thành dominant’ được dùng cho tái đồng bộ thì khi một nút truyền một bit dominant sẽ không thực thi tái đồng bộ khi một lỗi pha được xác định.

Xử lý lỗi (Error Handling)

- Tính hợp lệ của thông điệp

Tại bộ truyền và bộ nhận thông điệp, thời gian mà tại đó thông điệp được xác định là hợp lệ là khác nhau.

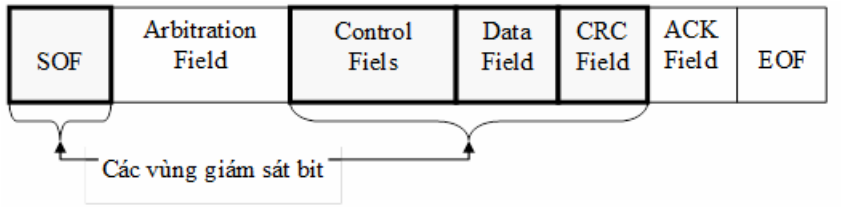
Bộ truyền (Transmitter): Thông điệp là hợp lệ nếu không có bất kỳ lỗi nào cho đến khi truyền xong EOF. Nếu xuất hiện lỗi, Sự truyền lại sẽ tự động bắt đầu cùng với mức ưu tiên của nó. Để có thể cạnh tranh truy cập bus với các thông điệp khác, sự truyền lại được bắt đầu ngay khi bus rảnh.

Bộ nhận (Receiver): Thông điệp là hợp lệ nếu không có bất kỳ lỗi nào cho đến bit kế cuối của EOF. Với bit cuối cùng của EOF, bộ nhận xử lý nó dạng 'don't care' (tùy định), tức nếu là bit dominant cũng không ảnh hưởng và không xem là lỗi.

- Phát hiện lỗi

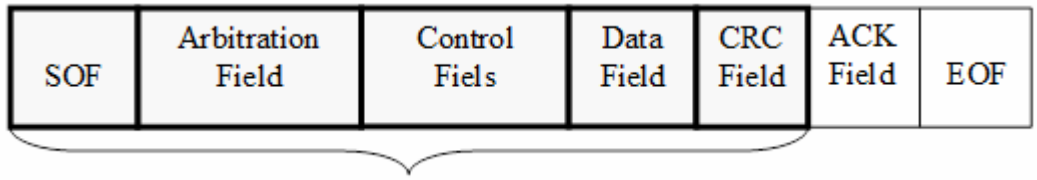
Có 5 loại lỗi khác nhau (chúng không loại trừ lẫn nhau)

Lỗi bit (Bit Error): Một đơn vị đang gửi một bit trên bus sẽ giám sát bus đó. Một lỗi bit được phát hiện ở thời gian bit đó khi giá trị bit được giám sát khác với giá trị bit được gửi. Ngoại trừ, khi gửi một bit recessive trong chuỗi bit được chèn của vùng phân xử (Arbitration field) và trong khe ACK (ACK slot) thì sẽ không coi là lỗi bit nếu giám sát thấy đó là bit dominant. Một bộ truyền gửi một cờ lỗi bị động và phát hiện ra một bit dominant thì vẫn không coi đó là một lỗi bit.



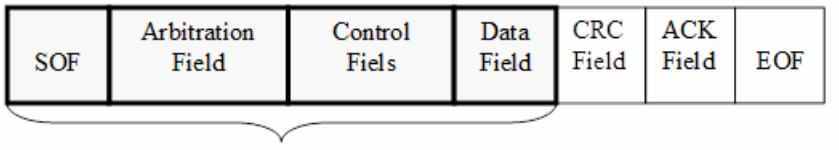
Những vùng được giám sát bit

Lỗi chèn (Stuff Error): Lỗi chèn xuất hiện khi phát hiện thấy có 6 bit liên tiếp bằng mức nhau trong vùng thông điệp được mã hóa theo luật chèn bit. Bit lỗi ở đây là bit thứ 6 (bit chèn).



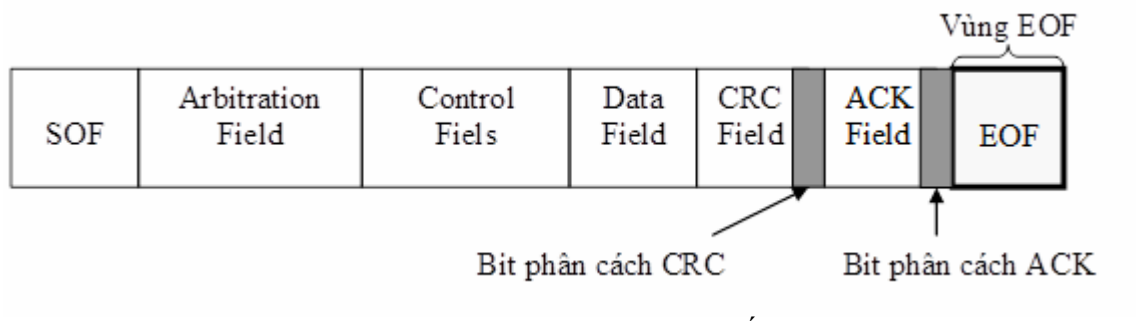
Những vùng áp dụng luật chèn bit

Lỗi CRC (CRC Error): Như đã mô tả trong phần các loại khung truyền, CRC Sequence trong vùng CRC là kết quả của quá trình tính toán CRC của bộ truyền. Các bộ nhận cũng sẽ tính CRC giống như bộ truyền khi nó nhận được thông điệp. Một lỗi CRC xuất hiện khi kết quả tính toán của bộ nhận không giống CRC Sequence mà nó nhận được.



Những vùng được tính CRC

Lỗi dạng (Form Error): Một lỗi dạng được phát hiện khi một vùng bit được định dạng cố định có một hay nhiều bit vi phạm. Chú ý, một bộ nhận phát hiện một bit dominant ở bit cuối của EOF sẽ không xử lý nó như một lỗi dạng).



Những vị trí bit được định dạng cố định

Lỗi ACK (Acknowledgment Error): một lỗi ACK được phát hiện khi một bộ truyền không giám sát thấy bất kỳ một bit dominant nào trong khe ACK (ACK slot).

- Báo hiệu lỗi (Error Signalling)

Một trạm phát hiện một điều kiện lỗi sẽ báo hiệu lỗi này bằng cách truyền một cờ lỗi (Error Flag). Nếu một nút 'error active', nó sẽ truyền một cờ lỗi chủ động. Nếu một nút 'error passive', nó sẽ truyền cờ lỗi bị động.
Khi có bất kì một lỗi bit, lỗi chèn, lỗi dạng hay lỗi ACK nào được phát hiện bởi một trạm, sự truyền cờ lỗi sẽ được bắt đầu ở trạm tương ứng tại bit kế tiếp.
Khi một lỗi CRC được phát hiện, Sự truyền cờ lỗi sẽ bắt đầu tại bit sau bit phân cách ACK (sau ACK delimiter), trừ khi một cờ lỗi do điều kiện lỗi khác gây ra được sẵn sàng gửi.

Các quy định về trạng thái lỗi

- Các trạng thái lỗi

Một nút CAN có thể rơi vào một trong ba trạng thái sau:

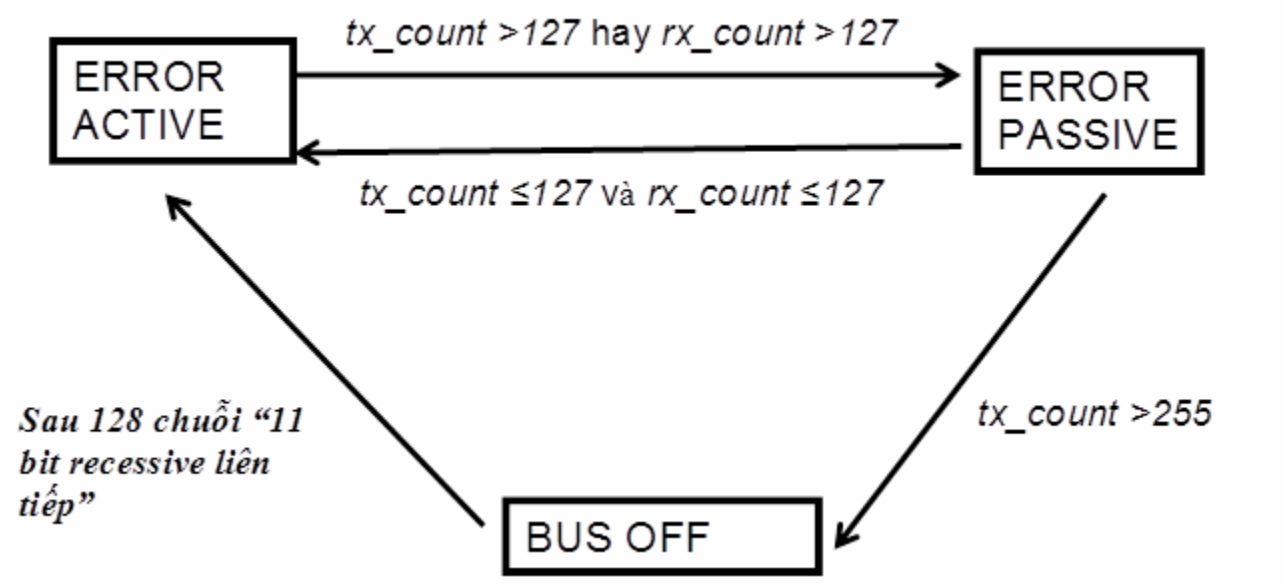
- 'Error active': tạm gọi là trạng thái chủ động với lỗi. Một đơn vị vẫn xử lý bình thường với thông tin trên bus như truyền hay nhận dữ liệu và sẽ gửi một cờ lỗi chủ động khi một lỗi được phát hiện.
- 'Error passive': tạm gọi là trạng thái bị động khi có lỗi. Một đơn vị vẫn xử lý bình thường trên bus thông tin và khi một lỗi được phát hiện nó sẽ gửi một cờ lỗi bị động. Nhưng sau khi truyền, một đơn vị 'error passive' sẽ phải chờ trong một khoảng thời gian (đó là khoảng ngưng truyền tạm thời Suspend Transmission – xem phần khoảng liên khung) trước khi bắt đầu sự truyền tiếp theo.
- 'Bus off': tạm gọi là trạng thái tắt bus. Khi đơn vị xuất hiện những vấn đề nghiêm trọng khi truyền thông điệp, nó sẽ không được phép tác dụng lên bus (không truyền và nhận bất cứ thông điệp nào). Nó sẽ tái hoạt động khi được reset.

- Quy định về xác định trạng thái lỗi

Để biết khi nào một đơn vị trên bus rơi vào trạng thái nào thì một đơn vị sẽ dựa vào giá trị của hai bộ đếm lỗi là bộ đếm lỗi truyền (Transmit Error Count – Tx_count) và bộ đếm lỗi nhận (Receive Error Count – Rx_count). Các khoảng giá trị của chúng được quy định như sau:

- 'Error active' nếu Tx_count ≤ 127 và Rx_count ≤ 127.
- 'Error passive' nếu Tx_count > 127 hoặc Rx_count > 127 nhưng Tx_count ≤ 255.
- 'Bus off' nếu Tx_count > 255.

Hình sau đây sẽ minh họa đầy đủ về sự thay đổi giữa các trạng thái này:



Sự chuyển đổi giữa các trạng thái lỗi

Cấu trúc và nguyên tắc hoạt động của CAN

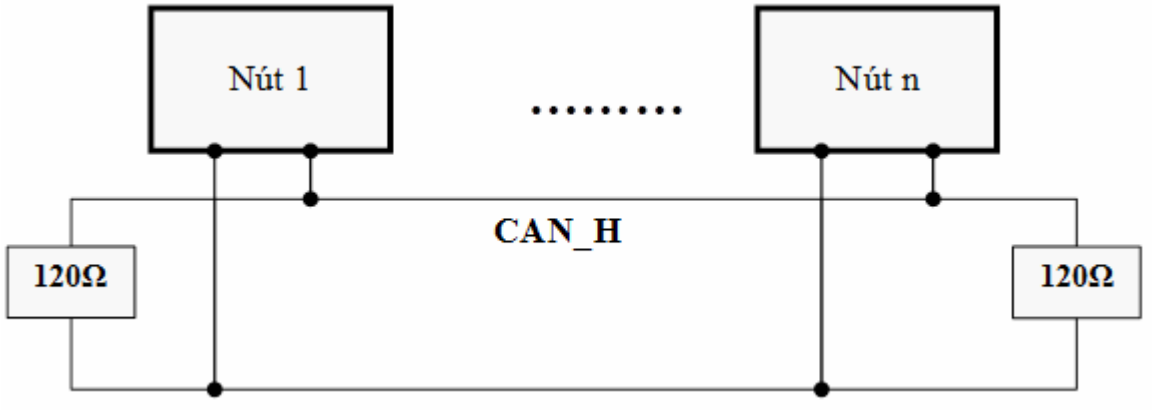
Phần này trình bày về cách kết nối các nút trong một bus CAN, cấu trúc một nút CAN, mức áp trên bus CAN và cách truyền nhận thông điệp trên bus.

- Cấu trúc liên kết (thuộc lớp vật lý):

Như phần đầu đã trình bày, giao thức CAN được quy định trong hai lớp là lớp vật lý (Physical Layer) và lớp liên kết dữ liệu (Data Link Layer). Trong đó lớp liên kết dữ liêu gồm hai lớp con là MAC (medium Access Control) và LLC (Logical Link Control).

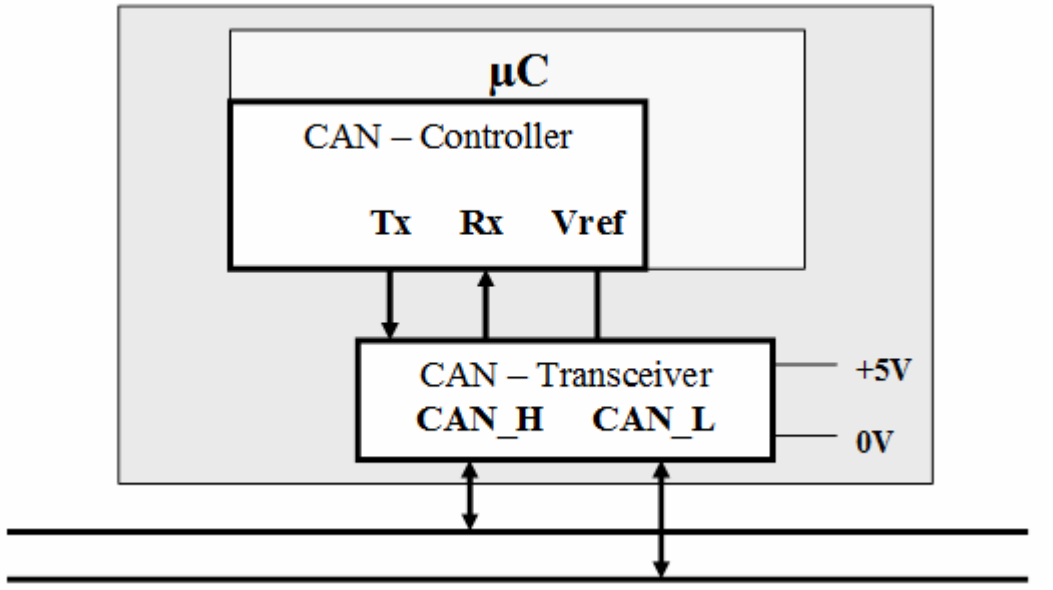
Theo chuẩn ISO 11898 – 2 về tốc độ cao, cấu trúc mạng CAN là một cấu trúc bus gồm hai dây riêng biệt được gọi là CAN_H và CAN_L với tầm điện áp từ -2V trên CAN_L đến +7V trên CAN_H. Tùy vào mỗi ứng dụng mà hai dây này có thể là dây xoắn kép hay cáp quang. Điện trở đặc tính của đường dây là 120Ω và điện trở đầu cuối đặt giữa các điểm cuối của đường dây là 120Ω. Theo chuẩn này, tốc độ dữ liệu có thể lên tới 1Mbit/s với vhiều dài bus trên lý thuyết là 40 m.

Trễ lan truyền đặc trưng danh định của đường bus hai dây được chỉ định là 5 ns/m. Để đạt được sự tương thích thì tất cả các nút trong mạng phải sử dụng định thời bit giống nhau.



Đường bus CAN

Trong đó một nút CAN được mô tả như sau, một nút CAN yêu cầu phải có một 'vi điều khiển' (microcontroller - µC) kết nối với một 'bộ điều khiển CAN' (CAN – Controller). Bộ điều khiển CAN sẽ được kết nối với 'bộ chuyển đổi CAN' (CAN – Transceiver) thông qua một đường ra dữ liệu nối tiếp (Tx) và một đường vào dữ liệu nối tiếp (Rx). Đường Vref là điện áp ra tham khảo cung cấp một mức điện áp danh định bằng 0.5×Vcc = 0.5×5 = 2.5V.

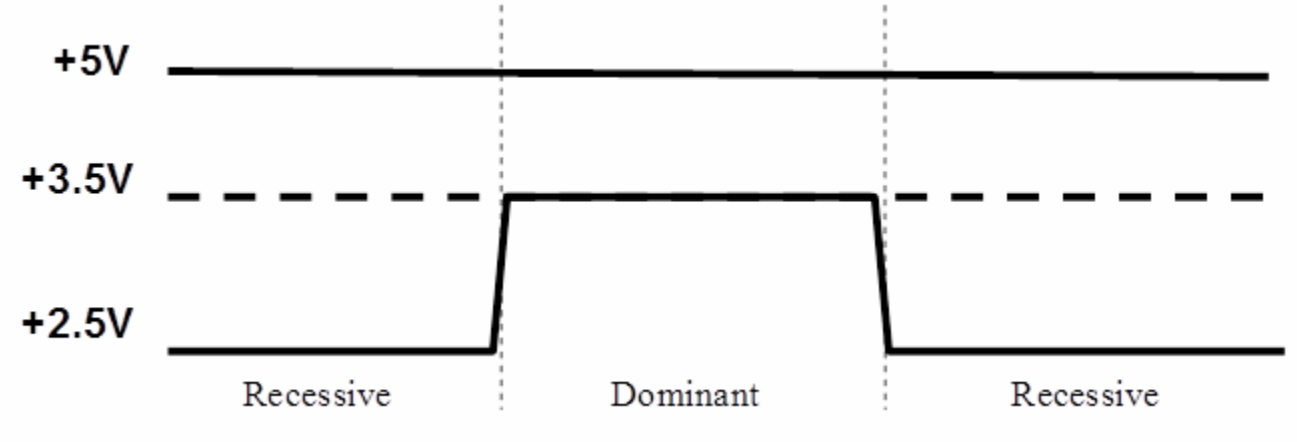


Cấu trúc một nút CAN

- Mức áp trên bus CAN

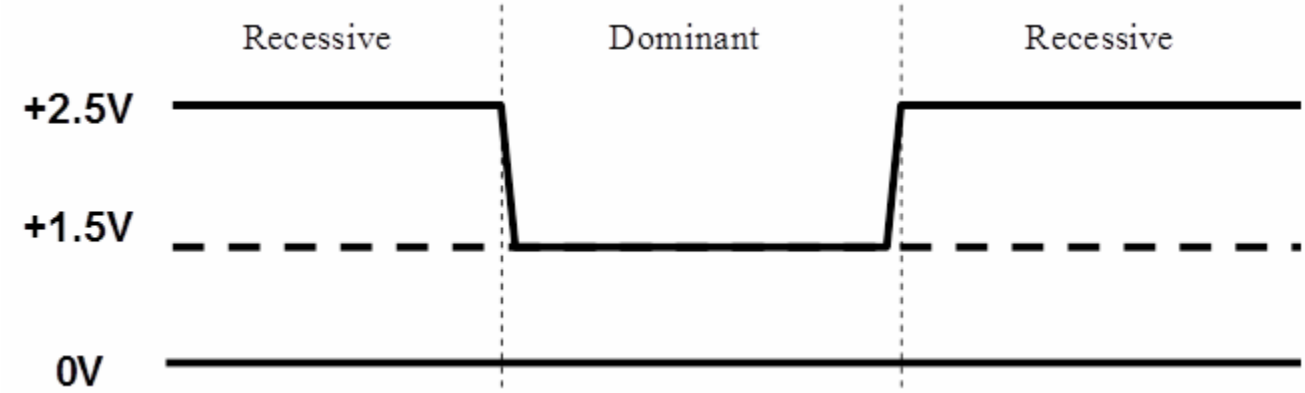
Theo chuẩn ISO 11898 thì mức áp trên bus CAN trong tần từ -2V với CAN_L đến +7V với CAN_H nhưng được sử dụng phổ biến ngày nay là 0V với CAN_L và +5V với CAN_H khi dùng ở tốc tộ cao 1Mbit/s.
Đường CAN_H mức áp +5V khi ở trạng thái nghỉ và sẽ sụt áp còn +3.5V khi

đang hoạt động. Lúc này, mức +3.5V được quy định là mức ‘dominant’ và +2.5V được quy định là mức ‘recessive’.



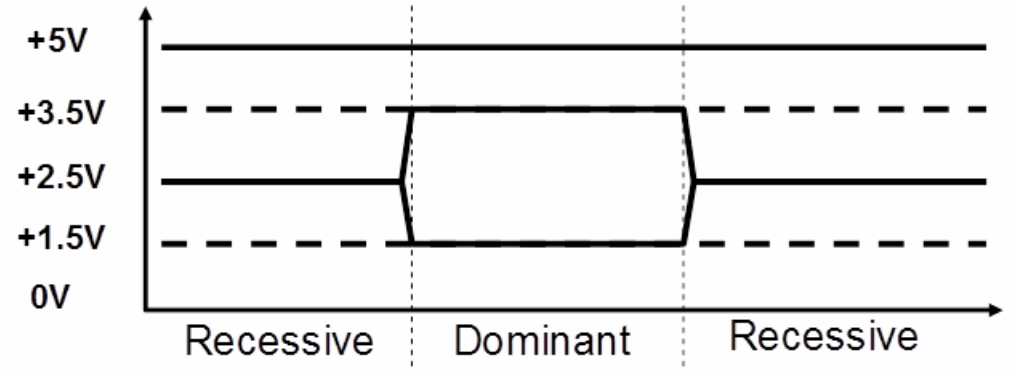
Đường CAN_H

Đường CAN_L có mức áp 0V khi ở trạng thái nghỉ và sẽ tăng lên +1.5V khi hoạt động. Lúc này, mức +1.5V được quy định là mức ‘dominant’ và +2.5V được quy định là mức ‘recessive’.



Đường CAN_L

Việc xác định trạng thái bus dựa vào sự sai lệch áp giữa CAN_H và CAN_L . Cụ thể, nếu áp CAN_H cao hơn CAN_L không quá +0.5V thì trạng thái bus là ‘recessive’, Nếu áp CAN_H cao hơn áp CAN_L tối thiểu là +0.9V thì trạng thái bus là ‘dominant’. Chú ý rằng nếu sai lệch áp giữa hai đường nằm trong đoạn (+0.5V; +0.9V) thì trạng thái trên bus không thể phân biệt được và gây ra sự hiểu sai về dữ liệu.



Xác định trạng thái bus thông qua sai lệch áp

- Cơ chế cấp phát bus (thuộc lớp phụ MAC của lớp liên kết dữ liệu)

Như đã trình bày, lớp phụ MAC có một chức năng quan trọng nhất là 'cấp phát bus' (the bus allocation). CAN cho phép các nút khác nhau được bắt đầu truyền cùng lúc và khi đó trên bus sẽ xảy ra sự xung đột (sự tranh chấp quyền truy cập bus giữa các thông điệp). Để giải quyết vấn đề này, CAN sử dụng một cơ chế gọi là CSMA/CD with Non-Destructive Bitwise Arbitration, tạm dịch là 'cơ chế truy cập đa điểm cảm biến sóng mang có phát hiện xung đột và phân xử từng bit không phá hủy'. Cơ chế này rất thích hợp với hệ thống điều khiển thời gian thực của CAN. Ngoài ra, cơ chế này còn được biết đến dưới một tên khác là CSMA/CD and Arbitration by Message Priority, tạm dịch 'cơ chế truy cập đa điểm cảm biến sóng mang có phát hiện xung đột và phân xử dựa trên mức ưu tiên của thông điệp'. Cơ chế này sẽ được giải thích cụ thể sau đây.

- Giải thích:

CSMA là viết tắt của Carrier Sense Multiple Access: là phương pháp truy cập mạng dùng cho các mô hình mạng chia sẻ để điều khiển truy cập mạng. Các thiết bị kết nốivào mạng sẽ lắng nghe (listen - phát hiện tín hiệu) trước khi truyền. Nếu kênh đang bận, các thiết bị sẽ chờ khi truyền. Truy cập bội (MA - multiple access) chỉ ra rằng nhiều thiết bị có thể kết nối vào và chia sẻ cùng một mạng. Tất cả các thiết bị có quyền truy cập như nhau trên mạng khi đường truyền rảnh. Tuy thiết bị có khả năng nhận biết mạng có đang được sử dụng hay không nhưng vẫn có khả năng là có hai trạm tìm cách truy cập mạng đồng thời, tức là sẽ xảy ra sự xung đột. Trên các mạng lớn, thời gian truyền từ đầu cáp này đến đầu kia là đủ để một trạm có thể truy cập đến cáp đó ngay cả khi đang có một trạm khác truy cập đến. Chính vì vậy cần phải có cơ chế phát hiện ra sự xung đột này.

CD là viết tắt của Collision Detection: Khi xung đột xảy ra, và cả hai thiết bị ngưng truyền, chờ trong một khoảng thời gian ngẫu nhiên, sau đó truyền lại. Phương pháp này xử lý các xung đột, nhưng nếu bus luôn bận, xung đột có thể xảy ra thường xuyên đến nỗi hiệu suất sẽ giảm đi đáng kể. Ước tính rằng lượng giao thông trên mạng phải nhỏ hơn 40% của dung lượng bus để có thể vận hành hiệu quả. Chính vì vậy, cơ chế này không thích hợp cho các hệ thống thời gian thực yêu cầu nhanh và tức thời.

Phân xử từng bit không phá hủy (Non-Destructive Bitwise Arbitration) hay phân xử trên mức ưu tiên của thông điệp (Arbitration by Message Priority) là cơ chế được sử dụng để làm cho CSMA/CD phù hợp với hệ thống thời gian thực. Trong đó, không phá hủy (Non-Destructive) nghĩa là nút thắng trong sự phân xử sẽ trở thành bộ truyền và tiếp tục truyền thông điệp chứ không phải bắt đầu lại từ đầu. Ngược lại, nút thua trong sự phân xử này trở thành bộ nhận và sẽ truyền lại thông điệp của nó khi bus rảnh.

Một đặc điểm quan trọng của thông tin CAN là các khung dữ liệu và khung yêu cầu được định hướng dữ liệu (data – oriented) chứ không định hướng đích đến (destination – oriented). Định hướng đích đến có nghĩa là dữ liệu được gửi đi đã chỉ rõ trạm nào trong mạng được phép nhận nó còn các trạm khác thì không. Còn với CAN, thông điệp được phát chung trên bus và mọi nút đều có thể thấy được thông điệp này còn việc có nhận thông điệp và thực thi nó hay không là tùy vào nút đó quyết định dựa vào 'khoảng cho phép' (scope of interests) được thiết lập cho nút đó.

Xem lại cấu trúc của khung dữ liệu và khung yêu cầu ta thấy mỗi khung khi được truyền đều kèm theo một IDENTIFIER (định danh) chỉ ra mức độ ưu tiên của khung. Khi có sự xung đột xảy ra thì việc so sánh các bit trong vùng phân xử giúp quyết định xem khung nào sẽ chiếm quyền truy cập bus. Nếu tại cùng một thời điểm, hai nút bắt đầu truyền, nút 1 truyền dominant (bit 0), nút 2 truyền recessive (bit 1) thì nút 1 sẽ chiếm ưu thế hơn và trên bus sẽ là dominant (do cơ chế AND-wire). Lúc này, nút 2 thấy bus là dominant khác giá trị bit mà nó truyền lên thì nó sẽ ngưng truyền và lắng nghe dữ liệu trên bus, ta nói nút 2 thua trong sự phân xử. Ngược lại, nút 1 thắng trong sự phân xử sẽ tiếp tục truyền thông điệp của mình. Về cấp độ ưu tiên, ta nói thông điệp nút 1 có mức ưu tiên cao hơn thông điệp nút 2.

Chú ý rằng, bit MSB được truyền trước nên thông điệp có IDENTIFIER càng cao thì có mức ưu tiên càng thấp và ngược lại. Ví dụ như: thông điệp thứ nhất có IDENTIFIER là MSB00010010111LSB (151), thông điệp thứ hai có IDENTIFIER là MSB00010000111LSB (135) thì thông điệp thứ hai có mức ưu tiên cao hơn.

Giữa khung dữ liệu định dạng chuẩn và khung dữ liệu định dạng mở rộng, khi có xung đột xảy ra và IDENTIFIER của khung chuẩn giống Base ID của khung mở rộng thì khung chuẩn sẽ có mức ưu tiên cao hơn vì bit thứ 12 trong vùng phân xử của khung dữ liệu chuẩn là bit RTR mang giá trị dominant, còn bit thứ 12 trong vùng phân xử của khung dữ liệu là bit SRR mang giá trị recessive.

Tương tự như vậy, khi xuất hiện xung đột giữa một khung dữ liệu và một khung yêu cầu và IDENTIFIER của hai khung này giống nhau thì khung dữ liệu sẽ chiếm ưu thế vì bit thứ 12 của khung yêu cầu là bit RTR nhưng lại mang giá trị recessive.

Cơ chế AND-wire được xác định như bảng sau, với cơ chế này, bus sẽ là

dominant khi một nút truyền dominant và bus chỉ là recessive chỉ khi tất cả các nút truyền recessive.

Nút 1	Nút 2	Nút 3	BUS	Chú thích
D	D	D	D	
D	D	R	D	
D	R	D	D	D = 'dominant'
D	R	R	D	
R	D	D	D	R = 'recessive'
R	D	R	D	
R	R	D	D	
R	R	R	R	

Minh họa cơ chế AND-wire

- Lọc thông điệp (thuộc lớp phụ LLC của lớp liên kết dữ liệu)

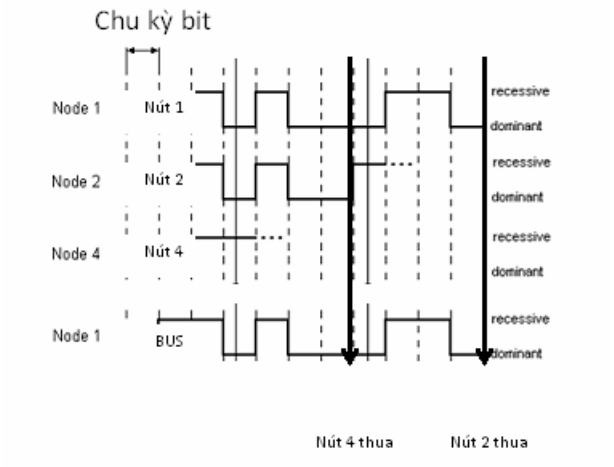
Lọc thông điệp (Message Filtering) là một nhiệm vụ quan trọng nhất của lớp phụ LLC. Một thông điệp luôn được truyền cùng với IDENTIFIER của nó. Các nút khác trong mạng sẽ nhận thông điệp này và đưa nó đến lớp phụ LLC. Tại đây, IDENTIFIER sẽ được so sánh với 'khoảng cho phép' (scope of interests), đây là một thanh ghi đặc biệt chứa các giá trị IDENTIFIER của dữ liệu mà nút đó được phép nhận.

Khoảng cho phép có thể chỉ chứa một giá trị IDENTIFIER hoặc chứa một khoảng IDENTIFIER bằng cách thiết lập giá trị tùy định tại vị trí bit nào đó. Một thông điệp mang IDENTIFIER nằm trong khoảng này thì lớp LLC sẽ cho phép đi qua và chuyển lên lớp cao hơn, còn nếu không LLC sẽ loại bỏ thông điệp đó.

Ví dụ về sự truyền và nhận thông điệp trong CAN

Giả sử ta có bốn nút CAN như sau:

- Nút 1: muốn truyền dữ liệu với IDENTIFIER 01101000110 (838).
- Nút 2: muốn truyền một dữ liệu khác với IDENTIFIER 01101001000 (840) và cho phép nhận dữ liệu với IDENTIFIER 01101000110 (838), nghĩa là 'khoảng cho phép' của nút này chỉ có một giá trị.
- Nút 3: không truyền dữ liệu nào và có khoảng cho phép là 0110100xxxx ('x' hiểu là tùy định, tức tầm giá trị của khoảng cho phép là [832; 847]).
- Nút 4: muốn truyền dữ liệu có IDENTIFIER 01110010010 (914) và khoảng cho phép là 0111001xxx1 (tức tầm của khoảng cho phép là các số lẻ trong đoạn [913; 927])
- Giả sử, nút 1, nút 2, nút 4 cùng truyền dữ liệu của mình trong cùng một thời gian. Như vậy, cả 3 nút đều trở thành bộ truyền và sự xung đột sẽ xuất hiện nên trên bus sẽ xảy ra sự phân xử.

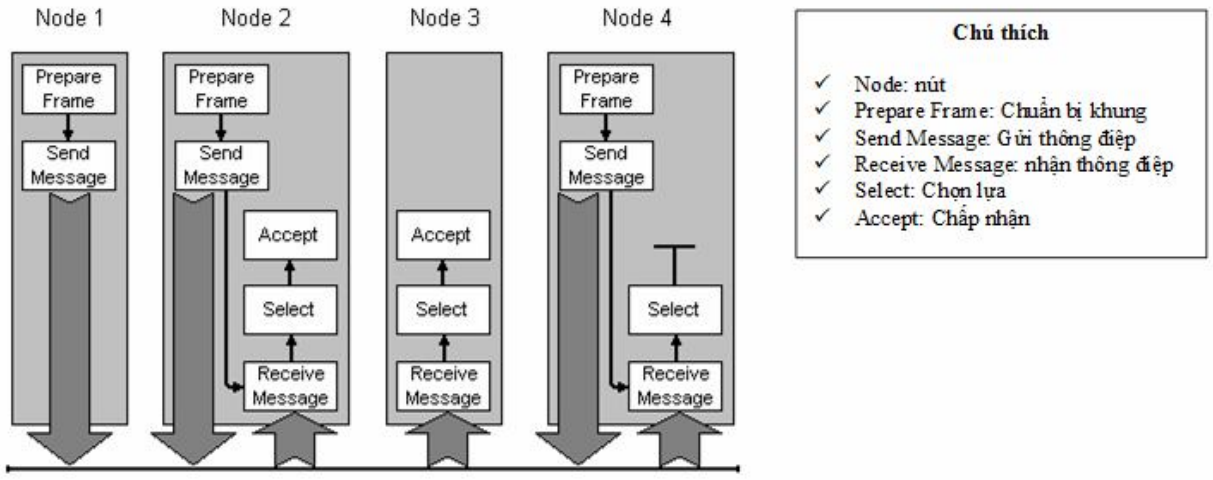


Sự phân xử từng bit khi xảy ra xung đột

Sự phân xử thực hiện như sau, ở ba bit đầu tiên, cả 3 nút đều truyền cùng mức và giá trị bus ở vị trí tương ứng cũng vậy. Đến bit thứ 4, nút 1 và nút 2 truyền dominant nên giá trị bus theo 'AND-wire' là dominant. Cùng lúc này, nút 4 truyền recessive nhưng khi giám sát bus lại thấy là dominant, nó hiểu là có nút khác cũng đang truyền dữ liệu, nó sẽ ngưng truyền và trở thành bộ nhận. Ta nói rằng nút 4 thua trong sự phân xử. Tương tự như vậy, đến bit thứ 8, nút 2 cũng thua trong sự phân xử và trở thành bộ nhận.

Kết quả của cuối cùng của trình phân xử trên là chỉ có nút 1 truyền dữ liệu với IDENTIFIER 838 của nó trên bus và là bộ truyền, các nút còn lại đều trở thành bộ nhận dữ liệu từ nút 1.

Trong 3 nút nhận là nút 2, nút 3, nút 4 thì nút 4 sẽ không nhận dữ liệu này vì 838 không thuộc đoạn [913; 927]. Nút 2 và nút 3 sẽ cho phép dữ liệu qua lớp phụ LLC để lên lớp cao hơn.



Trạng thái truyền/nhận dữ liệu của các nút