4.CẤU TRÚC KHUNG DỮ LIỆU VÀ KHUNG YÊU CẦU

4.1. Các loại khung truyền của giao thức CAN

Giao thức CAN sử dụng 4 loại khung (frame) khác nhau để truyền tải dữ liệu và điều khiển. Bốn loại khung này gồm:

- Khung dữ liệu (Data frame) là khung mang dữ liệu từ một bộ truyền đến các bộ nhận. Khung này có vùng để mang các byte dữ liệu.
- Khung yêu cầu hay khung điều khiển (Remote frame) là khung được truyền từ một Node để yêu cầu Node khác truyền khung dữ liệu có ID (IDENTIFIER) trùng với khung yêu cầu.
- Khung báo lỗi (Error frame) là khung được truyền bởi bất kỳ Node nào khi Node đó phát hiện lỗi bus.
- Khung báo quá tải (Overload frame) được sử dụng để tạo thêm độ trễ giữa giữa các khung dữ liệu hoặc khung yêu cầu. Mỗi Node trong bus CAN có thể truyền bất kỳ khi nào nếu phát hiện bus rảnh, nêu một Node nhận quá nhiều dữ liệu, nó có thể dùng khung này để ngăn sự truyền tiếp theo.

Chỉ có khung dữ liệu và khung yêu cầu là có ID, cơ chế phân xử sẽ áp dụng cho hai loại khung này khi chúng được truyền trên bus.

Khung dữ liệu và khung yêu cầu có hai định dạng khác nhau là định dạng chuẩn (Standard) và định dạng mở rộng (Extended).

- Định dạng khung chuẩn sử dụng ID có độ dài 11 bit
- Định dạng khung mở rộng sử dụng ID có độ dài 29 bit

Chuẩn CAN Specification 2.0-Part A (gọi tắt 2.0A) chỉ quy định sử dụng loại khung chuẩn. Chuẩn CAN Specification 2.0-Part B (gọi tắt 2.0B) sử dụng cả loại khung chuẩn và khung mở rộng. Như vậy, khi sử dụng CAN controller, bạn nên quan tâm đến việc nó tương thích chuẩn nào.

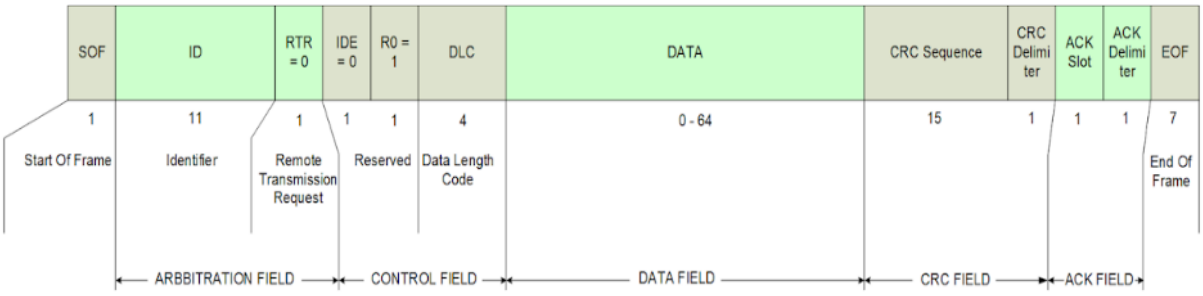
Ngoài ra, chuẩn CAN còn một loại định dạng thứ 5 là khoảng liên khung (Interframe Spacing). Nó có vai trò tạo khoảng ngăn cách giữa các khung truyền trên bus CAN.

4.2. Khung dữ liệu (Data frame)

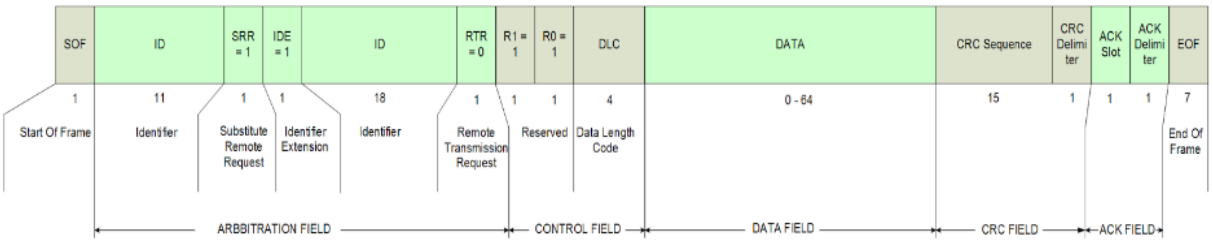
Một khung dữ liệu bao gồm bảy vùng bit khác nhau theo thứ tự là:

- 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)

Mỗi vùng có chức năng nhiệm vụ khác nhau, số lượng bit khác nhau và có quy định cụ thể về mức logic và ý nghĩa các bit. Trong hình minh họa có chú thích rõ số lượng bit từng vùng.



Hình 1. Định dạng khung dữ liệu chuẩn (Standard Data frame)



Hình 2. Định dạng khung dữ liệu mở rộng (Extended Data Frame)

4.2.1 Vùng bắt đầu khung SOF (Start of frame)

SOF đánh dấu sự bắt đầu của một khung dữ liệu hoặc khung yêu cầu. Nó chỉ bao gồm một bit dominant (logic 0).

Một Node chỉ được cho phép truyền khi bus rảnh, vấn đề khi nào bus rảnh sẽ được trình bày sau. Khi bus rảnh thì trạng thái trên bus đang là recessive (logic 1). Lúc này, nếu có một Node 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 Node trên bus được đồng bộ theo cạnh này. Nghĩa là, các Node bắt đầu tính toán tính toán thời gian để lấy mẫu các bit trên bus CAN.

Để dễ hình dung, bạn có thể liên tưởng đến hoạt động của UART, UART cũng bắt đầu truyền một khung của nó bằng 1 bit START có mức logic 0. Bên nhận phát hiện cạnh xuống của bit START sẽ bắt đầu tính toán để lấy mẫu các giá trị bit của khung truyền.

Việc đồng bộ theo cạnh gọi là đồng bộ cứng (Hard Synchronization).

4.2.2 Vùng phân xử (Arbitration field)

Định dạng vùng phân xử là khác nhau đối với dạng khung chuẩn và dạng khung mở rộng.

- Định dạng chuẩn: vùng phân xử có độ dài 12 bit, bao gồm 11 bit IDENTIFIER và 1 bit RTR. Vùng này có quy định khác nhau đối với chuẩn 2.0A và 2.0B.

- Định dạng mở rộng: vùng phân xử có độ dài 32 bit, bao gồm có 29 bit ID, 1 bit SRR, 1 bit IDE và 1 bit RTR

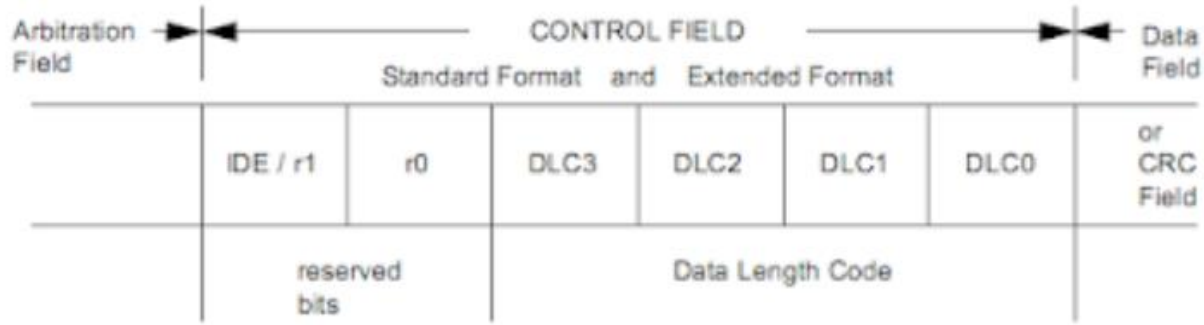
IDENTIFIER (ID)

- Đối với 2.0A, chỉ có định dạng chuẩn, 11 bit ID được đánh thứ tự từ ID-10 đến ID-0, ID-0 là bit LSB. 7 bit MSB, từ ID-10 đến ID-4 không được cùng là recessive (logic 1).
- Đối với 2.0B:
    - Định dạng chuẩn: 11 bit ID được đánh số thứ tự từ ID-28 đến ID-18. Việc quy định thứ tự này là vì bản 2.0B có thêm khung mở rộng được thêm vào 18 bit ID từ ID-17 đến ID-0. 7 bit MSB, từ ID-28 đến ID-22 không được cùng là recessive (logic 1).
    - Định dạng mở rộng: chia 29 bit ID làm 2 vùng
- Base ID (ID cơ sở): 11 bit từ ID-28 đến ID-18, tương ứng với vị trí của 11 bit ID ở khung chuẩn)
    - Extended ID (ID mở rộng): 18 bit từ ID-17 đến ID-0
    - Bit RTR (Remote Transmission Request)
    - Đây là bit dùng để phân biệt khung dữ liệu và khung yêu cầu
    - Bit này luôn bằng 0 (bit dominant) đối với khung dữ liệu
    - Bit này luôn bằng 1 (bit recessive) đối với khung yêu cầu
    - Vị trí bit này luôn ngay sau ID. Chú ý, bit RTR là nằm sau bit ID-18 với khung chuẩn và sau ID-0 với khung mở rộng
- Bit SRR (Substitute Remote Request)
    - Bit này chỉ có ở khung mở rộng
    - Đây là bit recessive (bit 1)
    - So sánh với khung chuẩn, vị trí của bit này trùng với vị trí của bit RTR nên có tên gọi là bit thay thế (Subsitute) cho bit RTR ở khung chuẩn
    - Giải sử có hai Node cùng truyền, một Node truyền khung dữ liệu chuẩn, một Node truyền khung dữ liệu mở rộng có ID giống nhau thì Node truyền

khung chuẩn sẽ thắng phân xử vì đến vị trí sau Base ID, khung chuẩn là bit RTR = 0, còn khung mở rộng là bit SRR = 1. Như vậy, khung chuẩn chiếm ưu thế hơn so với khung mở rộng khi có Base ID như nhau.

- Bit IDE (IDentifier Extension)
    - Đây chính là bit phân biệt giữa loại khung chuẩn và khung mở rộng.
    - IDE = 0 (dominant) thì là khung chuẩn
    - IDE = 1 (recessive) thì là khung mở rộng
- Bit này thuộc:
    - Vùng phân xử nếu là khung mở rộng
    - Vùng điều khiển nếu là khung chuẩn
- Dữ liệu trên bus CAN được truyền nối tiếp, các Node nhận từng bit, khi một Node nhận xong Base ID thì sẽ nhận tiếp bit tiếp theo, bit này có thể là SRR (nếu là khung mở rộng) hoặc RTR (nếu là khung chuẩn) vì đến đây Node chưa xác định được loại khung. Node sẽ xác định như sau:
    - Nếu bit ngay sau Base ID là bit dominant (bit 0) thì đây chắc chắn là khung dữ liệu định dạng chuẩn
    - Nếu bit ngay sau Base ID là bit recessive (bit 1) thì đây là khung Remote định dạng chuẩn hoặc khung mở rộng (khung dữ liệu hoặc khung yêu cầu). Node sẽ nhận tiếp bit tiếp theo là bit IDE để phân biệt
        - Nếu IDE = 0 thì bit trước đó là bit RTR và đây là khung yêu cầu dạng chuẩn
        - Nếu bit IDE = 1 thì bit trước đó là bit SRR và đây là khung mở rộng nhưng chưa biết là khung dữ liệu hay khung yêu cầu. Node sẽ nhận tiếp Extended ID và đến bit RTR mới phân biệt được.

4.2.3 Vùng điều khiển (Control Field)



Hình 3. Định dạng vùng điều khiển

Định dạng vùng này ở khung chuẩn và mở rộng là khác nhau.

- Khung chuẩn gồm IDE, r0 và DLC (Data Length Code).
- Khung mở rộng gồm r1, r0 và DLC.

  Bit IDE đã trình bày ở trên.

Bit r1 và r0:

- Đây là hai bit dự trữ.
- Tuy hai bit này phải được truyền là bit recessive (bit 1) bởi bộ truyền nhưng bộ nhận không qua tâm đến giá trị 2 bit này. Bộ nhận có thể nhận được các tổ hợp 00, 01, 10 hoặc 11 của r1 và r0 nhưng không coi đó là lỗi mà bỏ qua và nhận thông điệp bình thường.

Mã độ dài dữ liệu DLC (Data Length Code)

- Có độ dài 4 bit quy định số byte của vùng dữ liệu của khung dữ liệu
- Chỉ được mang giá trị từ 0 đến 8 tương ứng với vùng dữ liệu có từ 0 đến 8 byte dữ liệu. Như vậy, khung dữ liệu có thể không có byte dữ liệu nào khi DLC = 0.
- Giá trị lớn hơn 8 không được phép sử dụng

Number of Data Bytes	Data Length Code			
	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

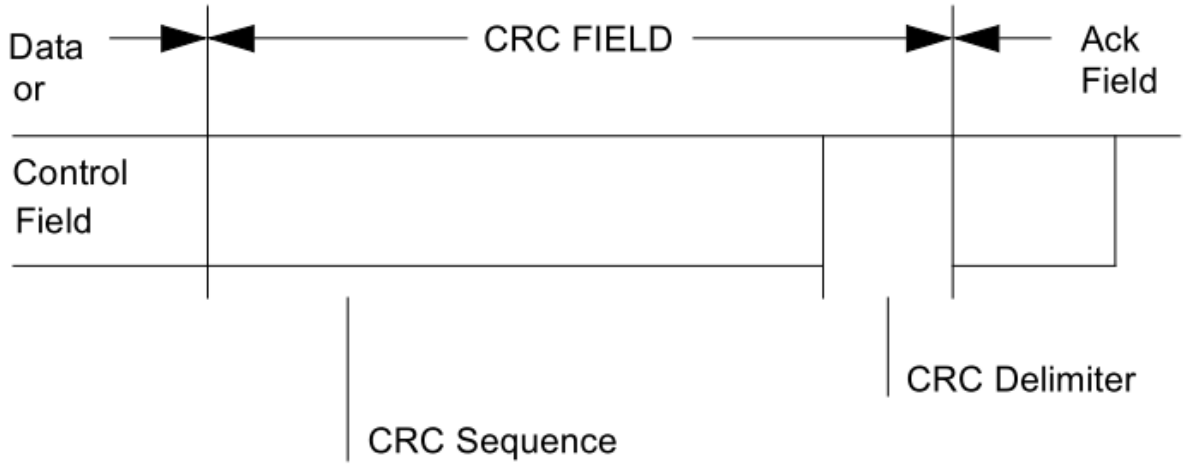
Hình 4. Bảng giá trị của DLC (r = recessive = 1, d = dominant = 0)

4.2.4 Vùng dữ liệu (Data field)

Vùng này có độ dài từ 0 đến 8 byte tùy vào giá trị của DLC ở vùng điều khiển. Mỗi byte có 8 bit dữ liệu với bit MSB được truyền trước.

4.2.5 Vùng CRC

Vùng kiểm tra hay vùng CRC gồm 16 bit và được chia làm hai phần là chuỗi CRC (CRC Sequence) và phần phân cách CRC (CRC Delimiter).



Hình 5. Vùng CRC

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 trình bày sau). 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:

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 chuỗi CRC.

Sau đây là đoạn mã giả minh họa cách tính toán chuỗi CRC

CRC_RG = 0; // initialize shift registerREPEAT CRCNXT = NXTBIT EXOR CRC_RG(14);

CRC_RG(14:1) = CRC_RG(13:0); // shift left by

CRC_RG(0) = 0; // 1 position

IF CRCNXT THEN

CRC_RG(14:0) = CRC_RG(14:0) EXOR (4599hex);

ENDIF

UNTIL (CRC SEQUENCE starts or there is an ERROR condition)

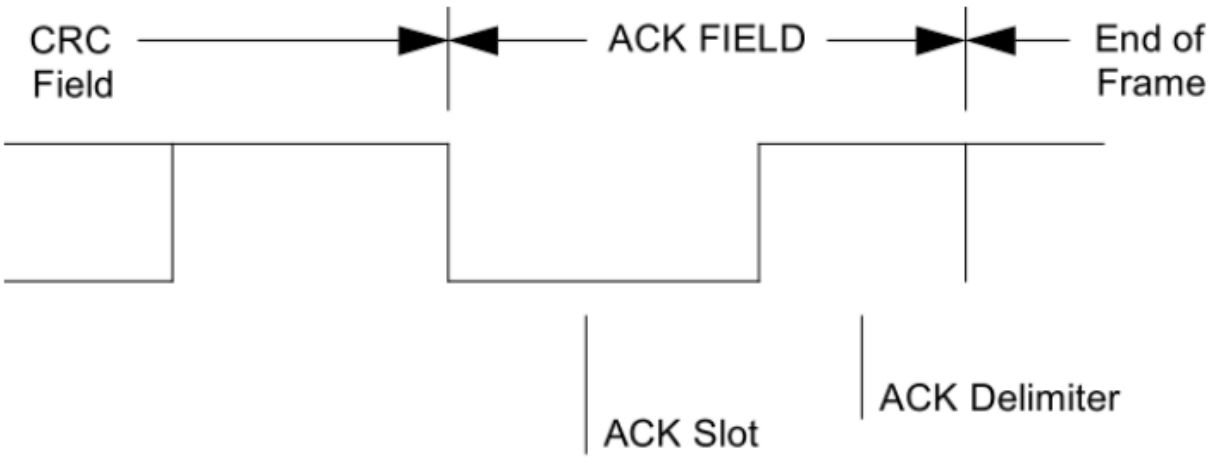
Mã kiểm tra CRC phù hợp nhất cho các khung mà chuỗi bit được kiểm tra có chiều dài dưới 127 bit, mã này thích hợp cho việc phát hiện các trường hợp sai nhóm (burst error). Ở đây, tổng bit từ vùng SOF đến vùng dữ liệu tối đa là 83 bit (khung định dạng chuẩn) và 103 bit (khung định dạng mở rộng).

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 từ vùng SOF đến vùng dữ liệu.

CRC delimiter: theo ngay sau CRC sequence, nó là một bit recessive làm nhiệm vụ phân cách vùng CRC với vùng ACK.

4.2.6 Vùng ACK

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)



Hình 6. Vùng ACK

ACK slot: có độ dài 1 bit, một Node truyền dữ liệu sẽ thiết lập bit này là recessive. Khi một hoặc nhiều Node 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 truyền ra một bit dominant ngay vị trí ACK slot để ghi đè lên bit recessive của bộ truyền.

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.

4.2.7 Vùng kết thúc khung EOF (End Of Frame)

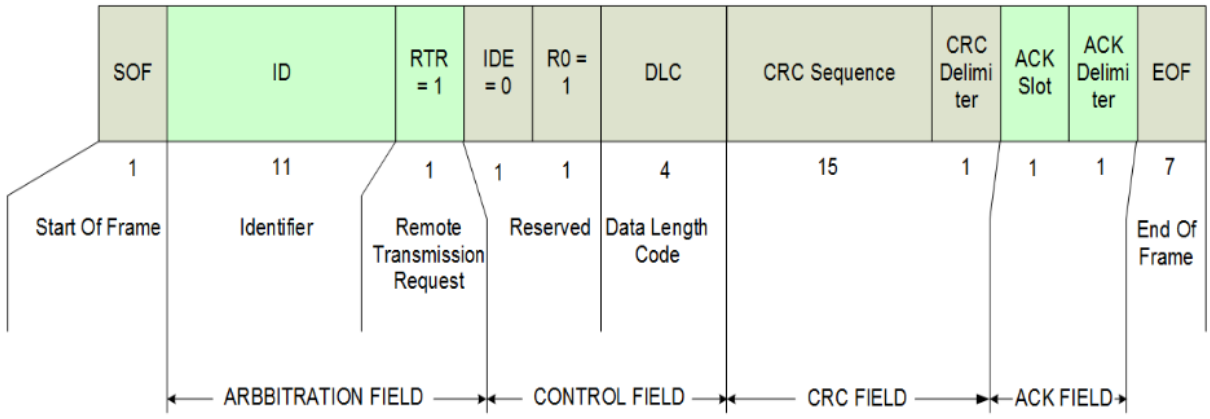
Vùng EOF là vùng thông báo kết thúc một khung dữ liệu hay khung yêu cầu. Vùng này gồm 7 bit recessive.

4.3. Khung yêu cầu (Remote frame)

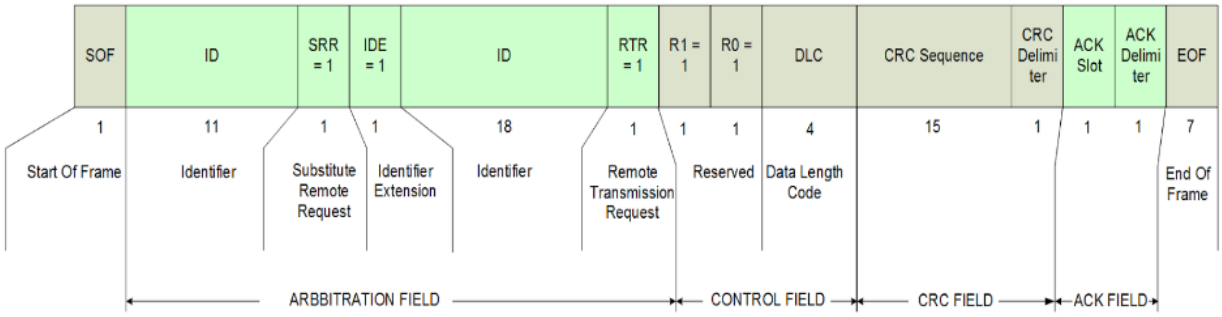
Một Node hoạt động như một bộ nhận đang mong muốn một dữ liệu, để nhận dữ liệu này từ một Node nguồn thì bộ nhận sẽ gửi một khung yêu cầu (còn gọi là khung điều khiển). Khung yêu cầu có tác dụng thông báo cho Node nguồn biết để Node này truyền dữ liệu.

Một khung yêu cầu có thể là định dạng chuẩn hay định dạng mở rộng. Dù thuộc định dạng nào thì khung yêu cầu 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).



Hình 7. Khung yêu cầu dạng chuẩn



Hình 8. Khung yêu cầu dạng mở rộng

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. Và khung dữ liệu trả về có số byte bằng với giá trị DLC này.

- Bit RTR (thuộc vùng phân xử) của khung yêu cầu là bit recessive. Nó khác với khung dữ liệu vì ở khung dữ liệu nó là một bit dominant.