LIN
Module Introduction DOCUMENT

Project Code: <Code of the project>

Document Code: <07ae-BM/PM/HDCV/FSOFT> – <1.4>

RECORD OF CHANGE

No	Effective Date	Version	Change Description	Reason	Reviewer	Approver
1	11/06/2017	1.0	Initial version	NA		
2	16/07/2017	1.1	Update chapter 2.3, 2.4, 2.6			
3	22/07/2017	1.2	Update chapter 1.6, 1.7, 2.4			
4	24/07/2017	1.3	Update chapter 2.4			
5	30/07/2017	1.4	Update chapter 2.1, 2.6			
6	08/08/2017	1.5	Update chapter 2.3, 2,6			
7	11/08/2017	1.6	Update chapter	Update arccording to Dat Bui's finding	Dat Bui	
8	14/08/2017	1.7	Update chapter 2.3, 2.4.4	Update arccording to Luong Tran's finding	Luong Tran	
9						
10						

MỤC LỤC

PHỤ LỤC HÌNH ẢNH 5

1. GIỚI THIỆU CHUNG TÀI LIỆU..... 7

1.1 Mục đích. 7

1.2 Các từ viết tắt. 8

1.3 Tài liệu tham khảo. 9

1.4 Lịch sử ra đời giao thức LIN. 10

1.5 LIN trong mô hình OSI 7 lớp. 13

1.6 Mô tả hoạt động của mạng nội bộ LIN. 14

1.7 Phần cứng IC transceiver và đường bus. 19

2. GIAO THỨC VÀ CƠ CHẾ HOẠT ĐỘNG CỦA LỚP DRIVER. 21

2.1 Giao thức. 21

2.1.1 Khung truyền. 21

2.1.2 Trường Break. 23

2.1.3 Trường đồng bộ dữ liệu Sync. 23

2.1.4 Trường PID. 23

2.1.5 Trường dữ liệu. 26

2.1.6 Trường checksum. 26

2.1.7 Lập lịch. 27

2.2 Giới thiệu các chức năng. 29

2.2.1 Các đặc tính/ thông số cơ bản. 29

2.2.2 Khái niệm cơ bản. 29

2.2.3 Các kiểu dữ liệu. 31

2.2.4 Các hàm API. 31

2.3 Các nền tảng phần cứng (IPVs). 32

2.3.1 Phần cứng LINFLEXD. 33

2.3.2 Phần cứng ESCI. 41

2.3.3 Phần cứng SCI. 44

2.3.4 Phần cứng LPUART. 48

2.4 Cách thức thực hiện hàm. 51

2.4.1 IP Vault LINFLEXD. 52

2.4.2 IP Vault ESCI. 68

2.4.3 IP Vault SCI. 80

2.4.4 IP Vault LPUART. 91

2.5 Sequence điển hình. 104

2.6 TIPS &Tricks. 105

2.6.1 Phân tích khung truyền trên thiết bị đo Oscilloscope. 105

2.6.2 Cách tạo các trường hợp lỗi truyền nhận bằng MAF Lite. 107

2.6.3 Đo thời gian thực thi hàm driver. 107

2.6.4 Lắp đặt mạch phần cứng. 108

2.6.5 Tốc độ baudrate sai. 109

2.7 Q&A. 110

2.7.1 Câu hỏi 1. 110

2.7.2 Câu hỏi 2. 110

3. PHỤ LỤC. 111

PHỤ LỤC HÌNH ẢNH

Hình 1.1 Một hệ thống điện tử trên ô tô điển hình. 10

Hình 1.2 Biểu đồ chi phí của các giao thức trên tổng số node. 11

Hình 1.3 Kết hợp hai giao thức CAN& LIN trong hệ thống điện tử trên ô tô. 11

Hình 1.4 Một số ứng dụng của LIN trên mạng điện tử ô tô. 12

Hình 1.5 Phân cấp của module LIN trong mô hình OSI 7 lớp. 13

Hình 1.6 Mô tả hoạt động của mạng LIN. 14

Hình 1.7 Vai trò của node master trong mạng. 15

Hình 1.8 Ví dụ hoạt động của khung truyền. 16

Hình 1.9 Các chế độ hoạt động của một node. 17

Hình 1.10 Quy trình gửi tín hiệu wakeup khi có lỗi. 17

Hình 1.11 Cấu tạo phần cứng một node LIN. 19

Hình 1.12 Hình ảnh mạng LIN vật lý. 19

Hình 1.13 Mức điện áp trên đường Bus LIN. 19

Hình 1.14 Hình ảnh transceiver LIN MC33662. 20

Hình 2.1 Mô hình mạng truyền thông LIN…… 21

Hình 2.2 Thành phần header trong khung truyền. 22

Hình 2.3 Thành phần response trong khung truyền. 22

Hình 2.4 Cấu trúc trường bắt đầu break trong khung truyền LIN. 23

Hình 2.5 Cấu trúc byte đồng bộ Sync trong khung truyền LIN. 23

Hình 2.6 Cấu trúc trường PID trong khung truyền LIN. 23

Hình 2.7 Cấu trúc của 1 byte dữ liệu trong khung truyền LIN. 26

Hình 2.8 Hai kiểu checksum. 26

Hình 2.9 Thời gian một khung truyền LIN. 27

Hình 2.10 Timing thực tế của khung truyền. 28

Hình 2.11 Lập lịch cho mạng LIN. 28

Hình 2.12 Các trạng thái của tác vụ Master. 29

Hình 2.13 Các trạng thái của tác vụ Slave. 30

Hình 2.14 Thời gian tối đa của khung truyền. 31

Hình 2.15 Cấu tạo phần cứng LinFlexD trên platform RaceRunner Ultra. 35

Hình 2.16 Hai trường hợp lỗi timeout. 37

Hình 2.17 Cấu tạo phần cứng eSCI trên platform Cobra55. 41

Hình 2.18 Cấu tạo phần cứng bên truyền SCI trên platform Carcassone 44

Hình 2.19 Cấu tạo phần cứng bên nhận SCI trên platform Carcassone. 45

Hình 2.20 Thanh ghi trên phần cứng SCI. 46

Hình 2.21 Cấu tạo phần cứng bên phát Tx - LPUART trên platform S32K14X. 48

Hình 2.22 Cấu tạo phần cứng bên thu Rx - LPUART trên platform S32K14X. 49

Hình 2.23 Cấu trúc các hàm API trong driver. 51

Hình 2.24 Hình ảnh một khung truyền LIN trên thiết bị đo oscilloscope. 105

Hình 2.25 Hình ảnh khung truyền LIN trên thiết bị đo Analyzer. 106

Hình 2.26 Giao diện trực tuyến tính toán giá trị byte checksum. 106

Hình 2.27 Sơ đồ kết nối phần cứng. 108

Hình 2.28 Hình ảnh mạch IC Tranceiver LIN. 109

1. GIỚI THIỆU CHUNG TÀI LIỆU.

1.1 Mục đích.

Tài liệu training về giao tiếp LIN trình bày tổng quan về khái niệm của giao thức và cung cấp kiến thức khá chi tiết về cấu trúc một số phần cứng cũng như cách thực hiện xây dựng driver theo chuẩn Autosar.

Đối tượng mà tài liệu này hướng đến là những Developer và Validator có kinh nghiệm làm việc với các module khác của team **MCAL_Development_Standard**, có nền tảng hiểu biết về các mảng kiến thức sau đây:

- EUnit, BEART (Tài liệu tham khảo: BEART and EUnit User Manual).
- UML design.
- Kiến thức basic về các nền tảng phần cứng ARM, PPC.
- Kiến thức lập trình C, coding convention (Misra, Compiler warning…).
- Đọc, hiểu các requirement Autosar, Non-Autosar.
- Cấu trúc các file code driver và thư mục test.
- …

Cách thức trình bày của tài liệu gồm có 3 phần, người đọc nên tìm hiểu lần lượt từng phần để tiếp cận từ tổng quan đến chi tiết các vấn đề liên quan đến giao tiếp LIN:

- **Chương 1:** Giới thiệu sơ lược về quá trình ra đời và phương thức hoạt động giao thức.
- **Chương 2:** Trình bày cụ thể về khung truyền, cấu trúc phần cứng và cách thức thực hiện xây dựng bộ driver. Bên cạnh đó, tài liệu cũng đề cập một số kinh nghiệm thực tiễn đáng chú ý.
- **Chương 3:** Sau khi đã nắm vững khái niệm tổng quan về LIN, người đọc nên tìm hiểu một số tài liệu về các test suite, test case.

1.2 Các từ viết tắt.

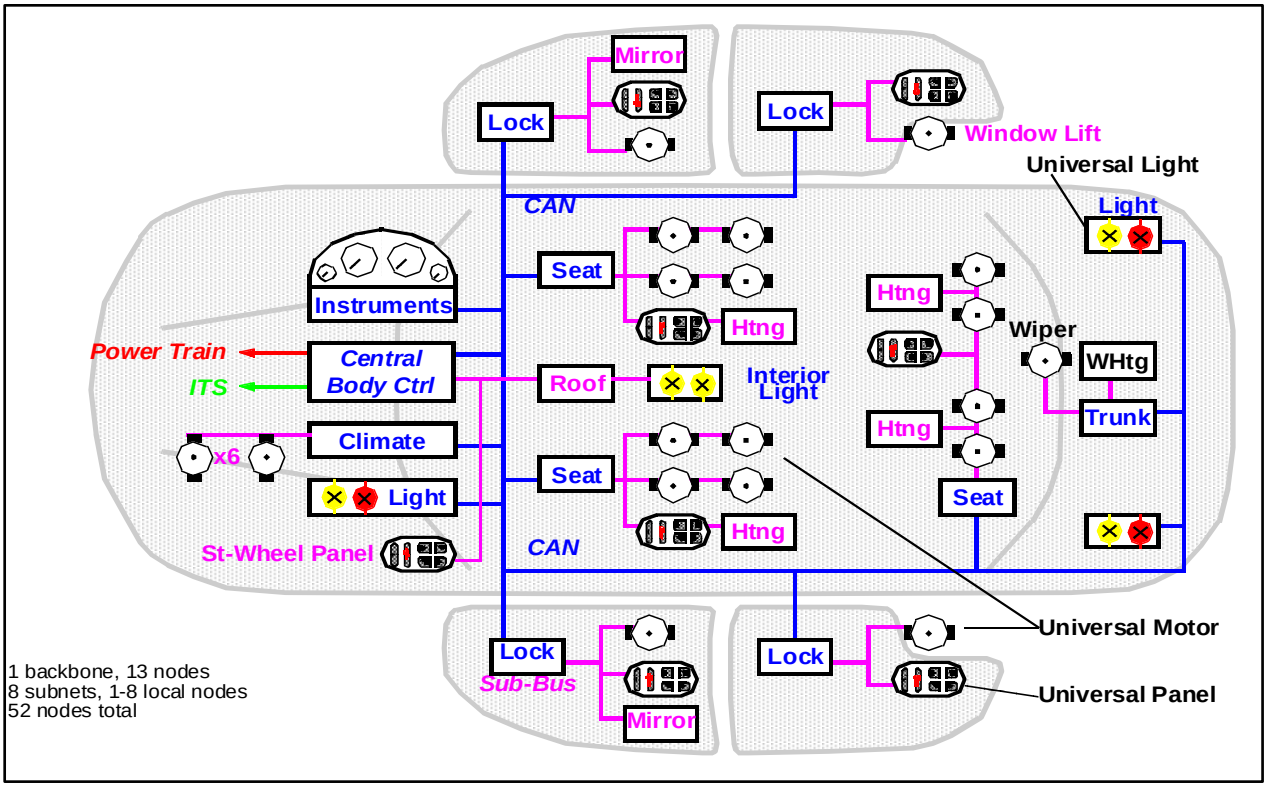
Từ viết tắt	Mô tả	Chú thích
AUTOSAR	Automotive Open System Architecture	
BDR	Baud Rate (bps)	Tốc độ truyền (đơn vị: bit trên giây)
CAN	Controller Area Network	
ECU	Electronic Control Unit	Bộ điều khiển điện tử
ESCI	Enhanced Serial Communication Interface	
ID	LIN Identifier Field	
IPV	IP Vault	
IPW	IP Wrapper	
LIN	Local Interconnect Network	
LPUART	Low Power Universal Asynchronous Receiver Transmitter	
LSB	Least Significant Bit	Bit trọng số thấp nhất
MAF	Measurement Automation Framework	Hệ thống đo lường tự động
MCAL	MicroController Abstraction Layer	
MSB	Most Significant Bit	Bit trọng số cao nhất
PDU	Protocol Data Unit	Đơn vị dữ liệu giao thức (bao gồm: PID, độ dài dữ liệu và dữ liệu)
RX	Receiver	Bộ nhận
SCI	Serial Communication Interface	
SDU	Service Data Unit	Dữ liệu được truyền trong PDU
TX	Transmission	Bộ truyền
UML	Unified Modeling Language	Ngôn ngữ mô hình hóa thống nhất

1.3 Tài liệu tham khảo.

Thứ tự	Tiêu đề
<01>	**LIN Specification Package** Revision 2.1, November 24, 2006
<02>	**LIN Specification Package** Revision 2.2A, December 31, 2010
<03>	**Specification of LIN Driver**, V1.5.0 R4.0 Rev 3
<04>	**LIN Training**, Miguel Agnesi Meléndez
<05>	**LIN protocol description**, Motorola

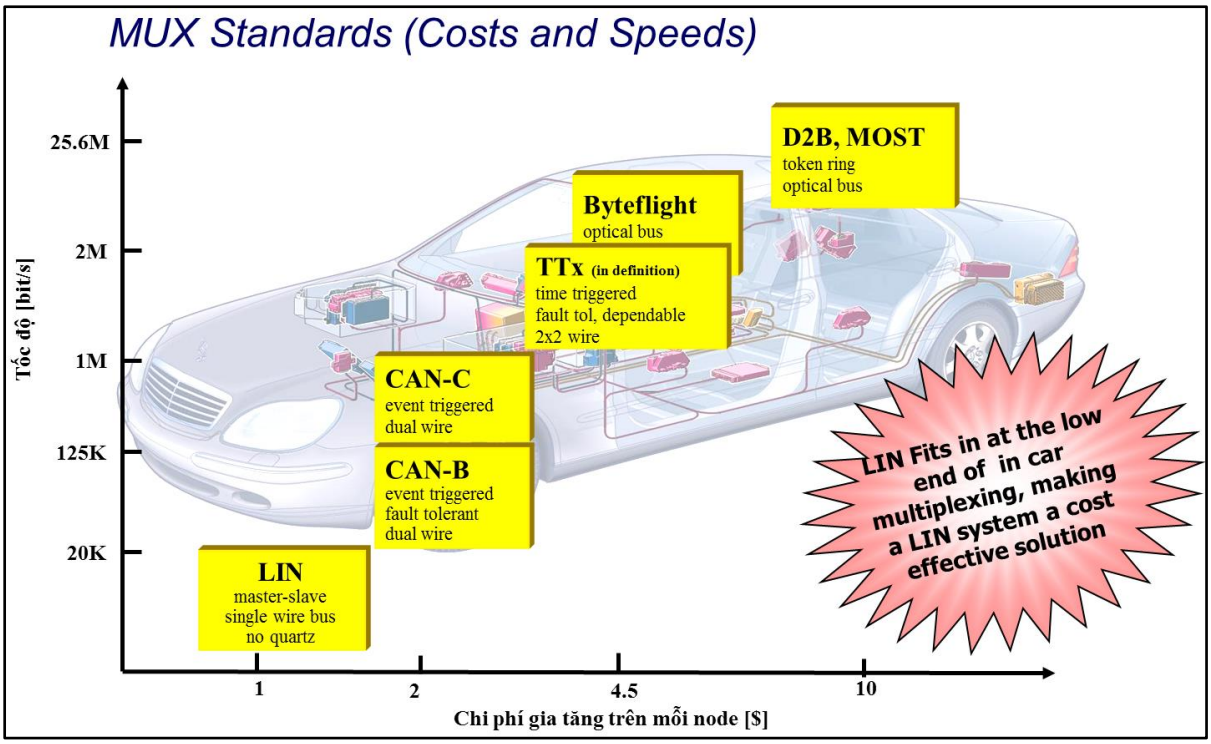
1.4 Lịch sử ra đời giao thức LIN.

Ngày nay, hệ thống điện tử trên ô tô thường sử dụng CAN (bên cạnh SPI, FlexRay) là giao thức chính trong kết nối các bộ điều khiển điện tử (ECU) với nhau. Tuy nhiên, càng ngày số lượng ECU càng nhiều hơn và đồng thời tỷ lệ thuận với số lượng giao tiếp giữa chúng với nhau.

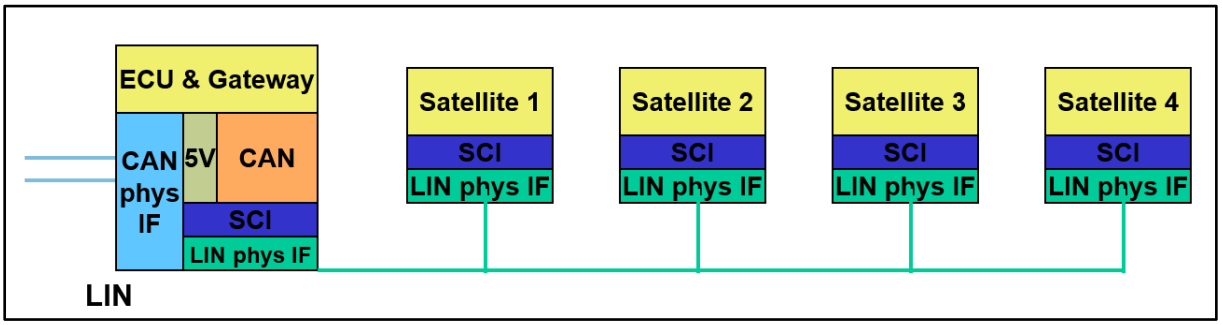


Hình 1.1 Một hệ thống điện tử trên ô tô điển hình.

Hình trên mô tả một mạng các phần tử điện tử điển hình có cấu trúc phân cấp với những ECU được kết nối với nhau qua bus CAN là giao tiếp chính của mạng. Tuy nhiên, giá thành sản phẩm xe hơi sẽ quá cao nếu sử dụng CAN trong mọi kết nối. Trong khi đó, ta cũng nhận thấy có nhiều các thành phần cơ điện tử nhỏ kết nối thông tin cục bộ với nhau với yêu cầu không quá cao về tốc độ truyền thông (dưới 20 Kbps). Do vậy, một giải pháp cân bằng giữa yếu tố kinh tế và yếu tố kỹ thuật (tốc độ truyền thông giữa các phần tử) là sử dụng kết hợp CAN với một giao thức mới có tên là LIN mà vẫn đảm bảo độ tin cậy.



Hình 1.2 Biểu đồ chi phí của các giao thức trên tổng số node.



Hình 1.3 Kết hợp hai giao thức CAN& LIN trong hệ thống điện tử trên ô tô.

Vào cuối những năm 1990, hiệp hội LIN được thành lập bởi 5 nhà sản xuất ô tô châu Âu (BMW, Volkswagen Group, Audi Group, Volvo Cars, Mercedes-Benz) với sự hỗ trợ về kỹ thuật từ phía hai nhà sản xuất chip là Mentor Graphics (trước đây là Volcano Automotive Group) và Freescale (trước đây là Motorola). Vào tháng 11/2002, phiên bản đầy đủ đầu tiên của bản đặc tả kỹ thuật LIN được phát hành là phiên bản 1.3. Tháng 9/2003, phiên bản 2.0 đã được phát hành với những cập nhật liên quan đến mở rộng khả năng cấu hình và bổ sung đáng kể các tính năng chẩn đoán và giao diện công cụ. Tính tới thời điểm hiện tại, tháng 08/2017 phiên bản mới nhất của LIN spec là 2.2A.

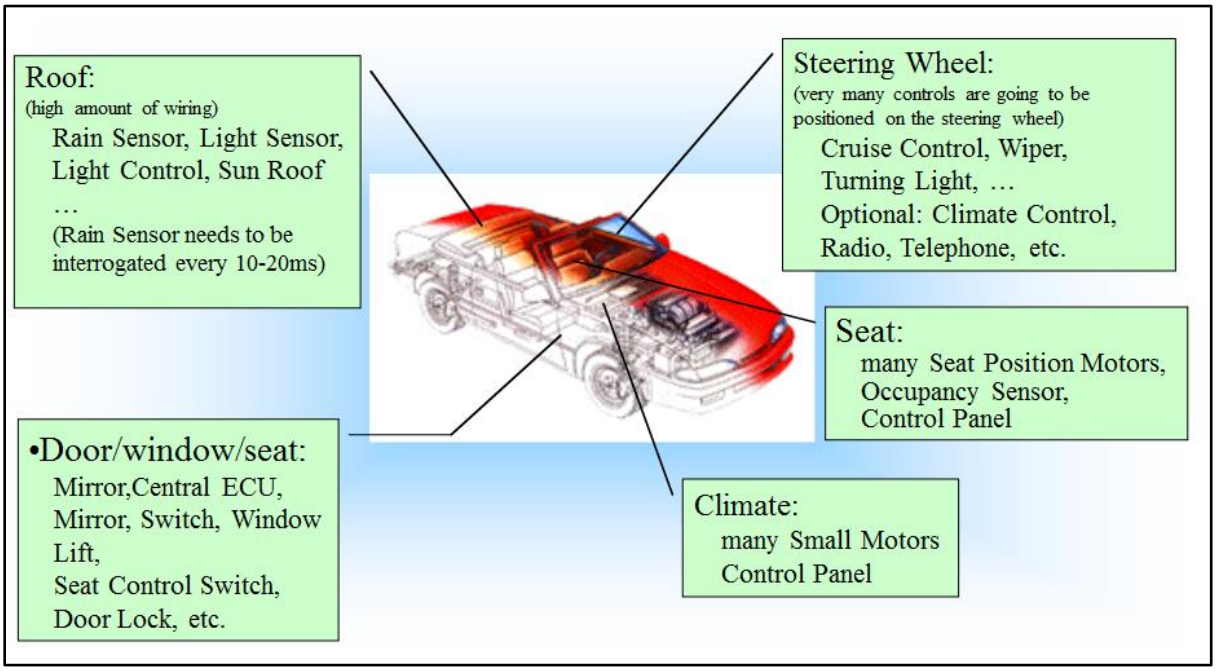
Bảng 1.1 Các phiên bản Spec LIN.

#	Phiên bản	Ngày xuất bản

1	LIN 1.0	01-07-1999
2	LIN 1.1	06-03-2000
3	LIN 1.2	17-11-2000
4	LIN 1.3	13-12-2002
5	LIN 2.0	16-09-2003
6	**LIN 2.1**	**24-11-2006 (Đang áp dụng cho MCAL)**
7	LIN 2.2A	31-12-2010 (Đang áp dụng cho SDK)
8	ISO17987	Đang thực hiện

Hiện nay, giao tiếp LIN được ứng dụng làm mạng con trong các trường hợp sau:

- Điều chỉnh gương.
- Điều chỉnh ghế ngồi.
- Kiểm soát cần gạt nước mưa.
- Kiểm soát mái che.
- Hệ thống khóa xe.
- Hệ thống điều hòa.
- …

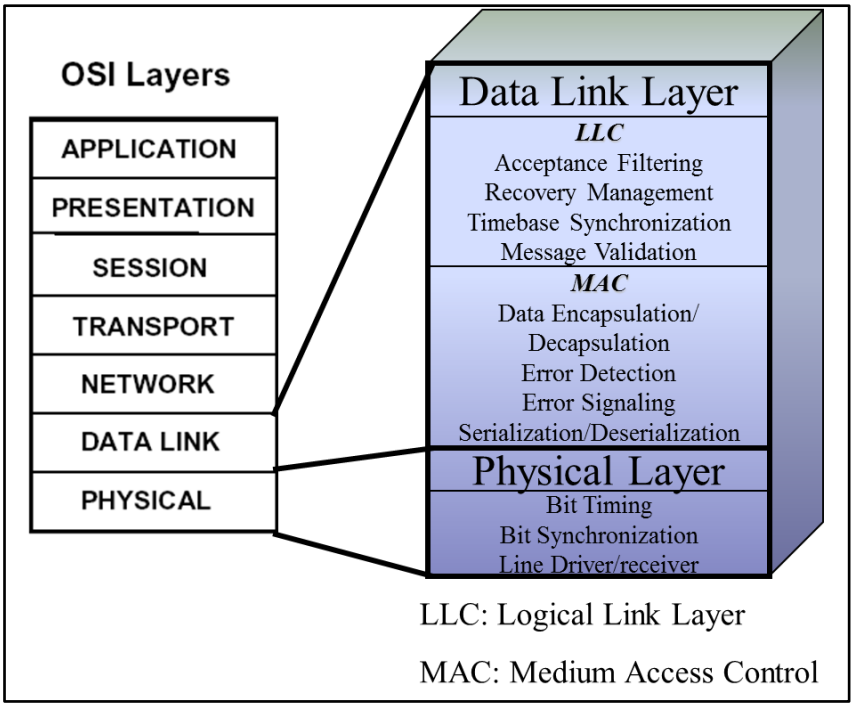


Hình 1.4 Một số ứng dụng của LIN trên mạng điện tử ô tô.

1.5 LIN trong mô hình OSI 7 lớp.

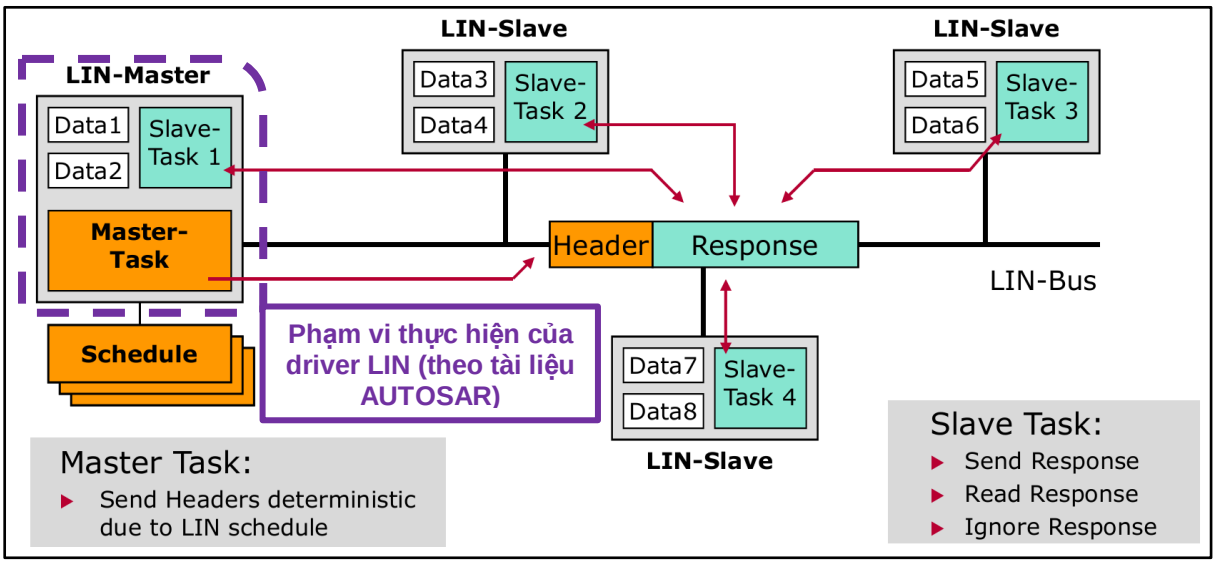
Giao thức LIN trong mô hình OSI 7 lớp được định nghĩa trên 2 lớp: **Lớp liên kết dữ liệu** (Data Link) và **Lớp vật lý** (Physical). Tiếp theo, ta bắt đầu tìm hiểu LIN bằng việc trả lời 2 câu hỏi định hướng:

- **Lớp liên kết dữ liệu:** Các node tham gia mạng có những vai trò và hành vi như thế nào? Câu hỏi này được trả lời trong phần 1.6.
- **Lớp vật lý:** Phần cứng hoạt động và tín hiệu truyền trên bus như thế nào?
  Nhóm MCAL thực hiện xây dựng bộ driver LIN dựa trên lớp vật lý, do đó câu trả lời sẽ được trình bày chi tiết ở chương 2.



Hình 1.5 Phân cấp của module LIN trong mô hình OSI 7 lớp.

1.6 Mô tả hoạt động của mạng nội bộ LIN.

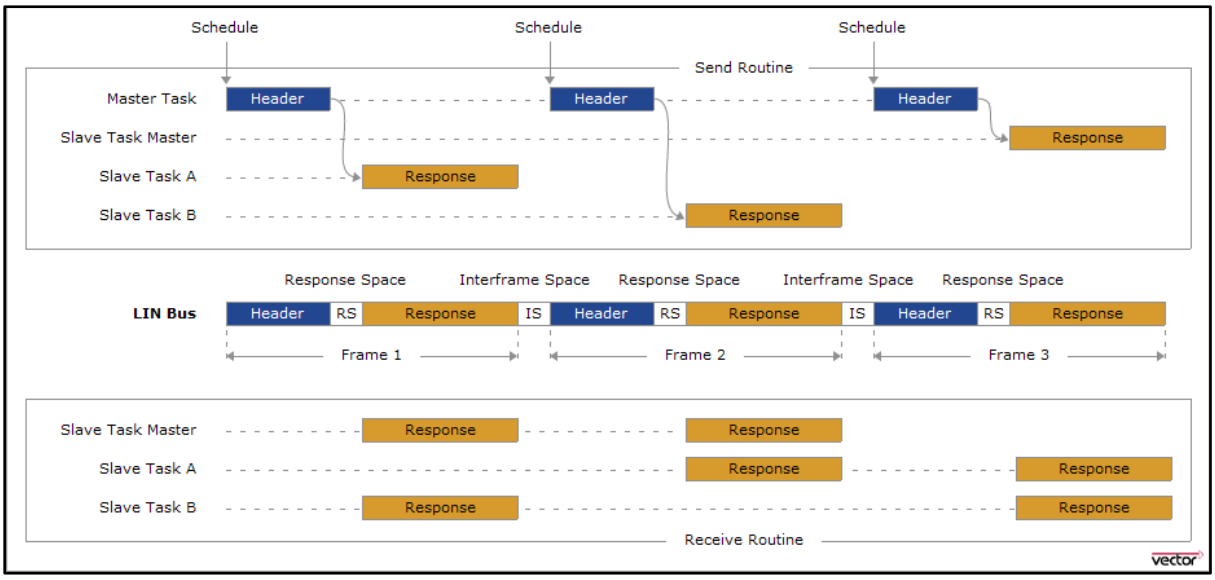


Hình 1.6 Mô tả hoạt động của mạng LIN.

Trong mạng nội bộ LIN, các node không có vai trò ngang hàng với nhau do có kiến trúc Master – Slave. Node Master có vai trò quyết định quy luật hoạt động của cả mạng thông qua nguyên tắc Token (PID – một thành phần trong khung truyền). Các node trong mạng truyền dữ liệu với nhau với định dạng khung truyền gồm hai phần là header (chứa thông tin về khởi tạo, đồng bộ baudrate, kiểu khung truyền) và response (chứa dữ liệu và checksum). Định dạng chi tiết về khung truyền sẽ được trình bày trong chương 2.

Node Master luôn là node khởi tạo mọi hoạt động truyền hoặc nhận dữ liệu trong mạng với nhiệm vụ gửi ra đường bus tiêu đề khung (header). Các thời điểm tạo ra các header này được lập lịch sẵn và lưu trong bảng lập lịch (schedule table). Trong cấu trúc header có chứa 1 byte PID – kiểu khung truyền - quy định hành vi phản hồi của các node slave. Khi node slave phát hiện một header xuất hiện trên đường bus, nó căn cứ vào giá trị PID để chọn 1 trong 3 cách xử lý:

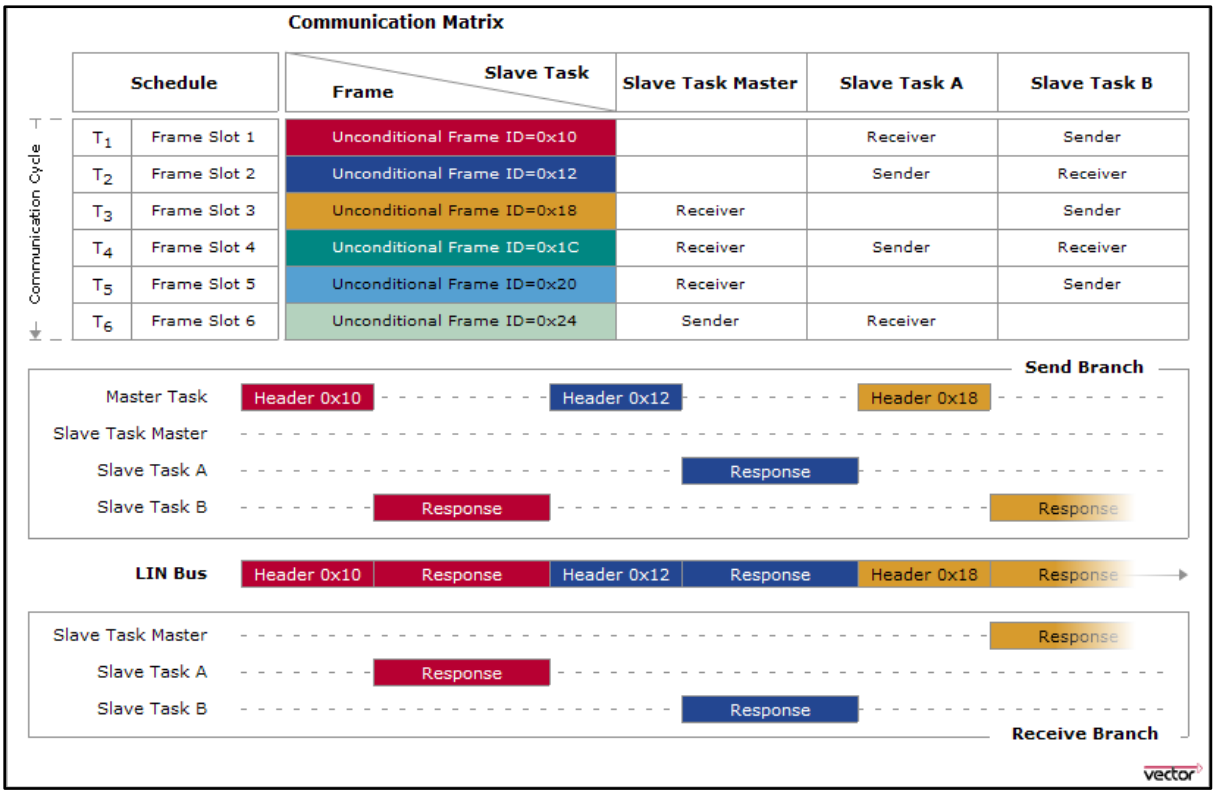
- Gửi lại phần response.
- Nhận phần response chứa dữ liệu, hoặc:
- Bỏ qua header, không phản hồi cũng như không nhận phần response.



Hình 1.7 Vai trò của node master trong mạng.

Sau đây, xin được trình bày một ví dụ đơn giản về hoạt động truyền thông trong mạng LIN có 3 node, gồm 1 master node và 2 node slave A và B. Bảng lập lịch cho một chu kì truyền nhận bao gồm 6 frame slot từ T1 đến T6 ứng 6 giá trị PID tương ứng. Tác vụ được cấu hình sẵn cho từng node, cụ thể như sau:

- **Slave gửi cho Slave**: ID = 0x10 ứng với frame slot đầu tiên, node slave B sẽ gửi dữ liệu đến node slave A. Tương tự cho trường hợp ID = 0x12, slave node A gửi dữ liệu cho slave node B.
- **Slave gửi cho Master**: ID = 0x18, node slave B gửi dữ liệu response lên đường bus LIN và master nhận.
- **Master gửi cho Slave**: ID = 0x24, node master gửi cả hai phần header và dữ liệu response lên bus, Slave node nhận dữ liệu.



Hình 1.8 Ví dụ hoạt động của khung truyền.

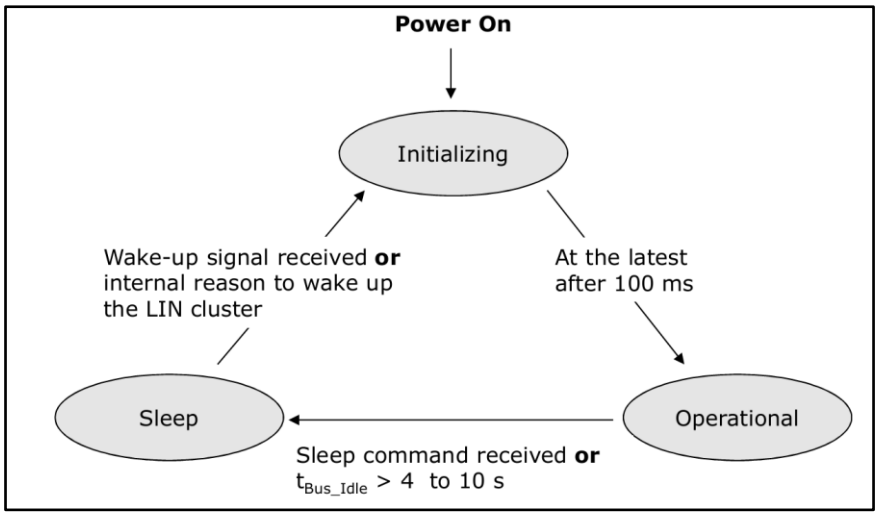
- **Quản lý hoạt động mạng:**

Bên cạnh quá trình truyền nhận dữ liệu, giao thức LIN cũng định nghĩa phương thức quản lý mạng LIN với hai hoạt động chính: wakeup và sleep mạng. Một node có 3 chế độ hoạt động: Khởi tạo, hoạt động và ngủ (tiết kiệm năng lượng).

Trạng thái Khởi tạo (Initializing): Trạng thái này được diễn ra sau khi reset hoặc sau khi wakeup. Node sẽ thực hiện khởi tạo cấu hình cần thiết và sau đó bắt đầu chuyển sang trạng thái hoạt động. Thời gian diễn ra trạng thái khởi tạo trong cả quá trình hoạt động của mạng LIN là không đáng kể khi chưa đến 100ms.

Trạng thái hoạt động (Operational): Trạng thái giao thức LIN (truyền và nhận các khung).

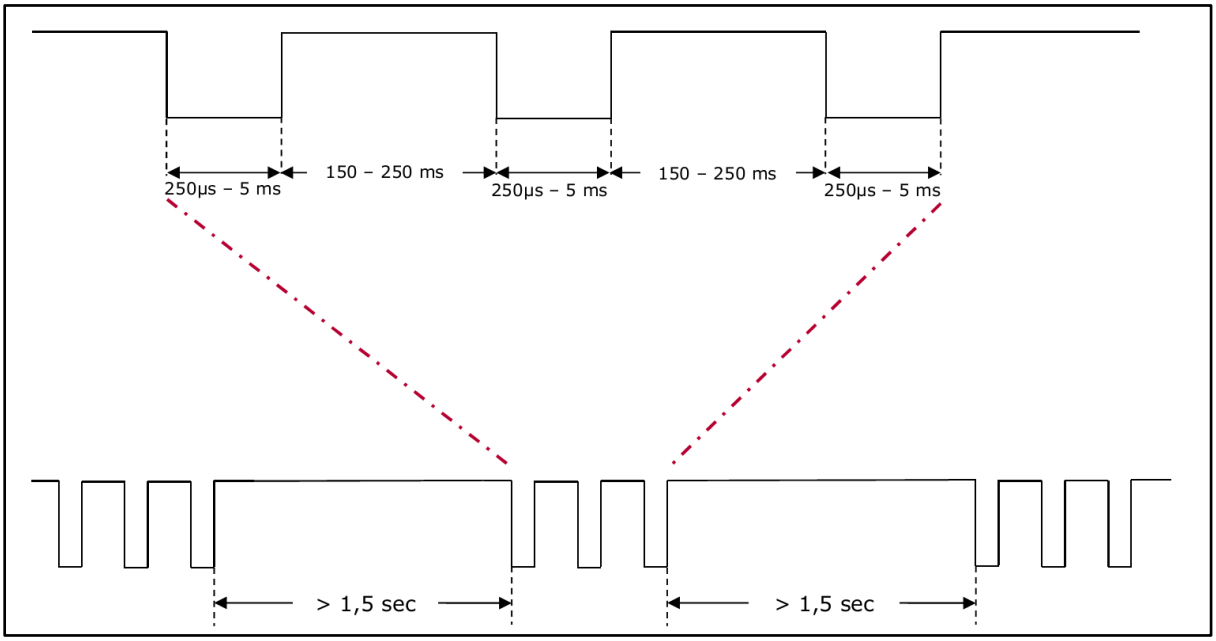
Trạng thái ngủ Bus (Bus sleep mode): Mức độ hoạt động của bus bị giảm xuống, chỉ còn tín hiệu wakeup có thể được truyền đi.



Hình 1.9 Các chế độ hoạt động của một node.

Quay lại với các hoạt động quản lý mạng:

- **Hoạt động Wakeup:** Bất kỳ node nào trong cụm LIN đang ở trạng thái ngủ có thể gửi một yêu cầu wakeup, bằng cách truyền tín hiệu wakeup. Tín hiệu wakeup được phát ra bằng cách gửi một xung xuống (mức logic 0) có độ dài từ 250us đến 5ms. Trong trường hợp node master không gửi một trường break (bắt đầu một khung truyền) hoặc một node tạo ra xung wakeup không nhận được xung wakeup từ các node khác sau thời gian 150ms đến 250ms thì node ban đầu tiếp tục gửi một xung wakeup khác. Trong trường hợp node slave truyền một xung wakeup cùng thời điểm với node master gửi trường break thì slave sẽ xử lý trường break này và hoạt động bình thường. Sau 3 lần gửi yêu cầu wakeup thì node tạo ra xung wakeup phải chờ ít nhất 1.5s trước khi tiếp tục gửi.

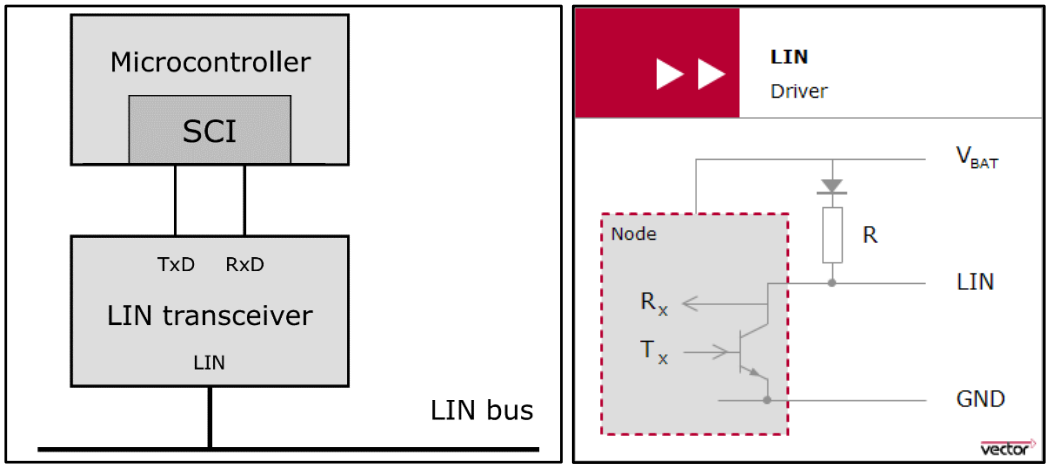


Hình 1.10 Quy trình gửi tín hiệu wakeup khi có lỗi.

- **Hoạt động Sleep**: Node master điều khiển mạng LIN vào trạng thái ngủ (tiết kiệm năng lượng) bằng cách gửi lệnh Go_To_Sleep là một khung truyền với dữ liệu 8 byte (0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF). Tuy nhiên, trường hợp bus LIN không có hoạt động truyền nhận nào diễn ra (định nghĩa là thay đổi mức điện áp trên bus) trong khoảng thời gian ngắn nhất là 4s đến 10s thì các node sẽ tự động chuyển trạng thái sang ngủ.

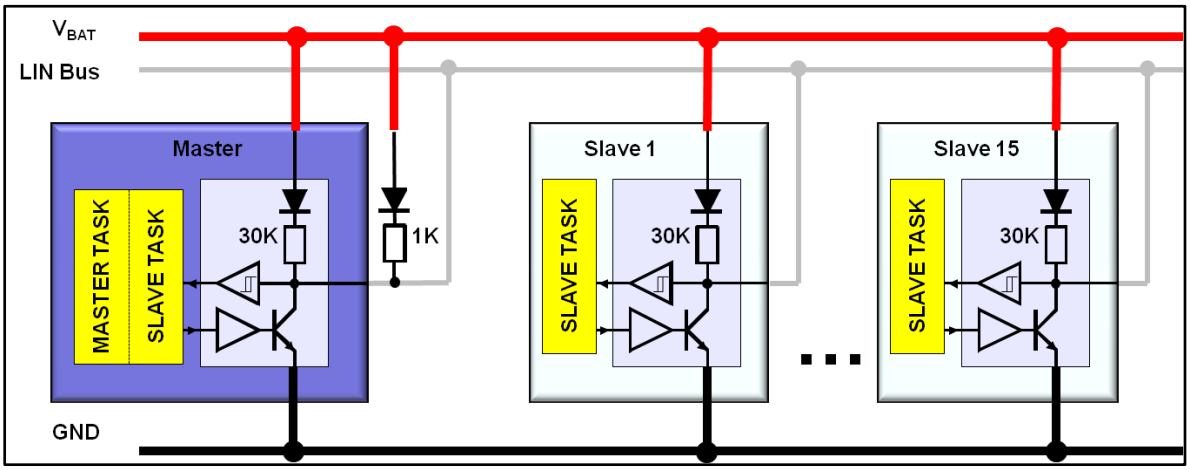
1.7 Phần cứng IC transceiver và đường bus.

Một node phần cứng (Master hoặc Slave) bao gồm vi điều khiển kết nối với IC transceiver LIN (MC33661, MC33662…) thông qua hai tín hiệu truyền Tx và nhận Rx.



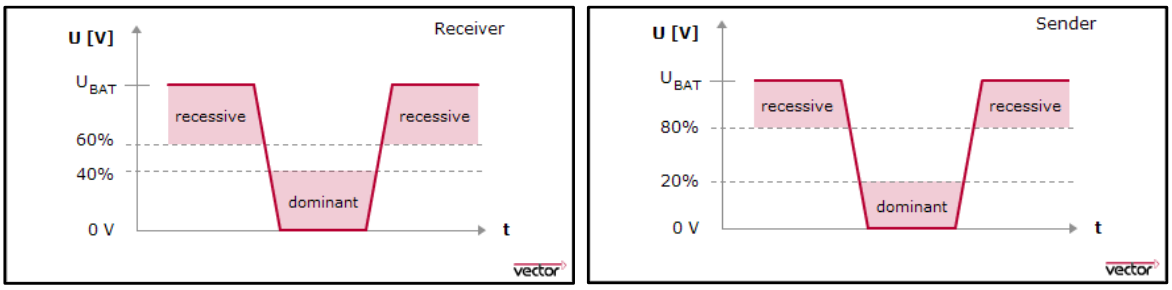
Hình 1.11 Cấu tạo phần cứng một node LIN.

Đường bus gồm có hai dây là 1 dây GND và 1 dây tín hiệu (mức chuẩn là 12V).



Hình 1.12 Hình ảnh mạng LIN vật lý.

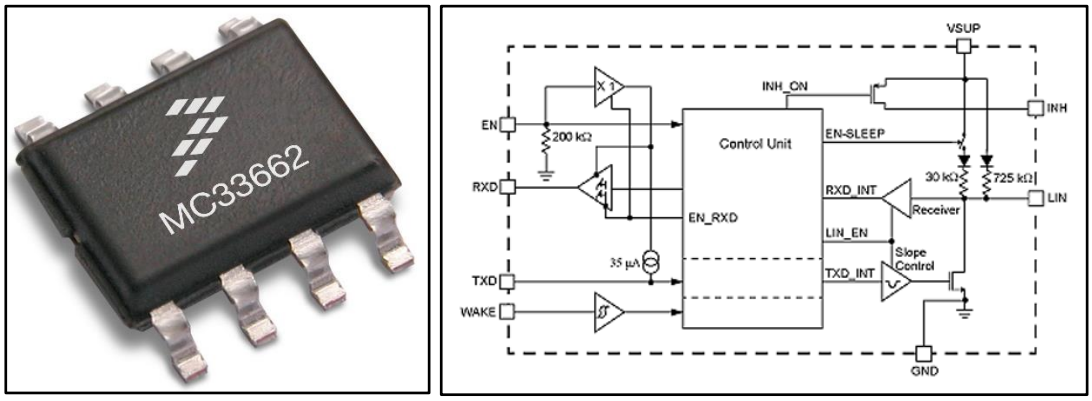
Tín hiệu trên đường bus LIN:



Hình 1.13 Mức điện áp trên đường Bus LIN.

- Ngưỡng điện áp mức “1” (Recessive threshold): 60% V_{SUP} khi nhận và khi truyền là 80% V_{SUP}
- Ngưỡng điện áp mức “0” (Dominant threshold): 40% V_{SUP} khi nhận và khi truyền là 20% V_{SUP}

Các IC transceiver này có nhiệm vụ chuyển đổi hai tín hiệu truyền Tx và nhận Rx (mức điện áp 3.3V hoặc 5V) sang một dây tín hiệu Bus mức 12V.

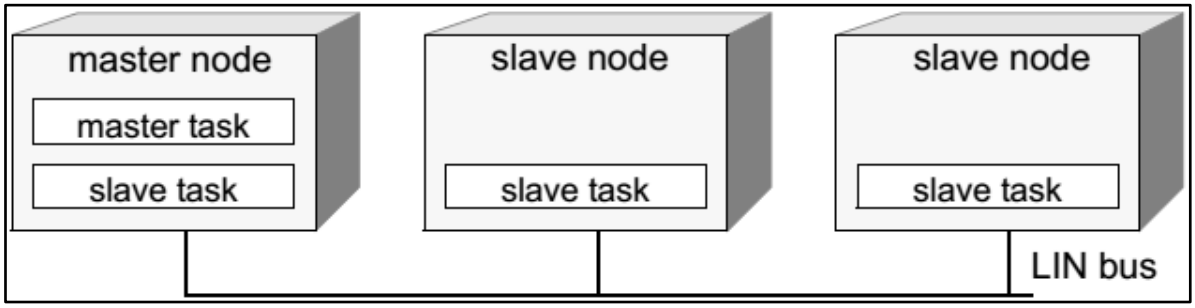


Hình 1.14 Hình ảnh transceiver LIN MC33662.

2. GIAO THỨC VÀ CƠ CHẾ HOẠT ĐỘNG CỦA LỚP DRIVER.

2.1 Giao thức.

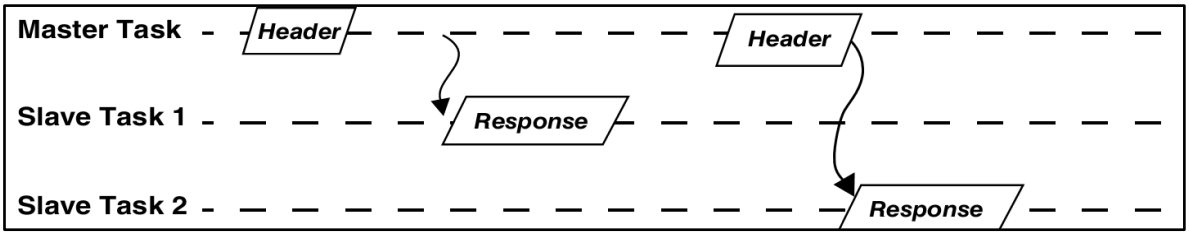
Khái niệm: LIN là giao thức mạng nội bộ sử dụng trong truyền thông giữa các thiết bị điện tử trên ô tô. Hình thái topo mạng của LIN là mô hình dạng BUS, truyền nhận nối tiếp bao gồm một node master và tối đa lên đến 15 node slave.



Hình 2.1 Mô hình mạng truyền thông LIN.

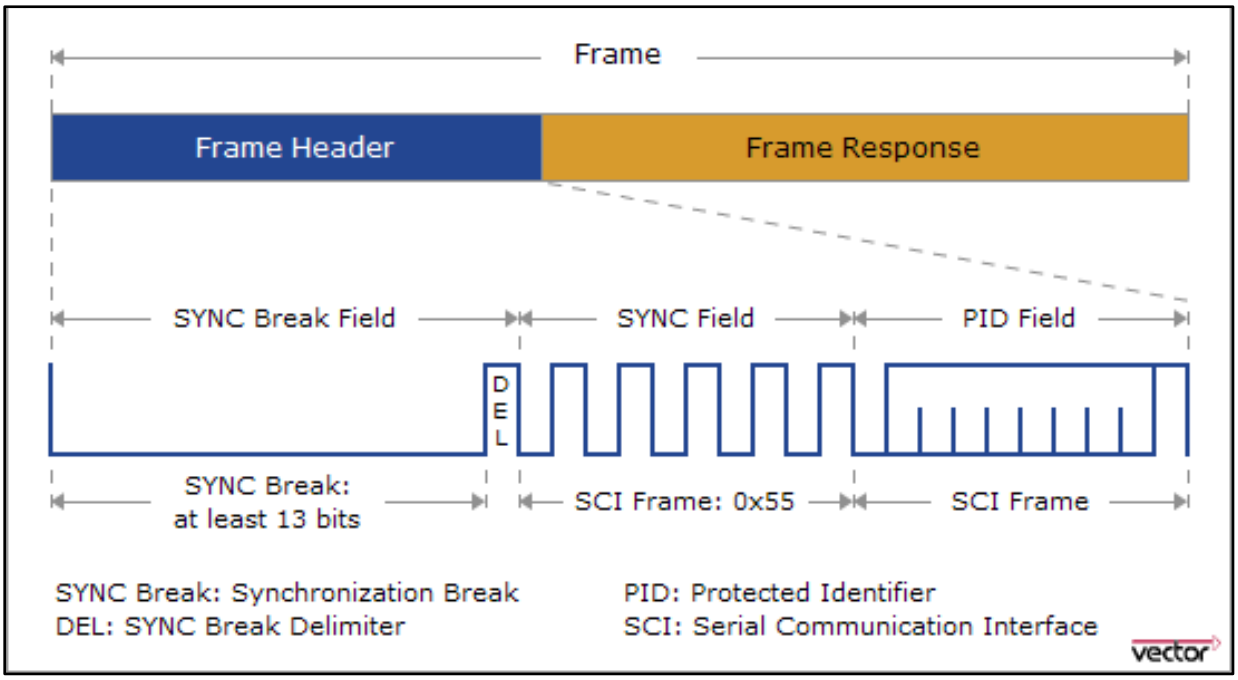
- **Node master**: thực hiện hai tác vụ master và slave.
- **Node slave**: duy nhất một tác vụ slave.

2.1.1 Khung truyền.

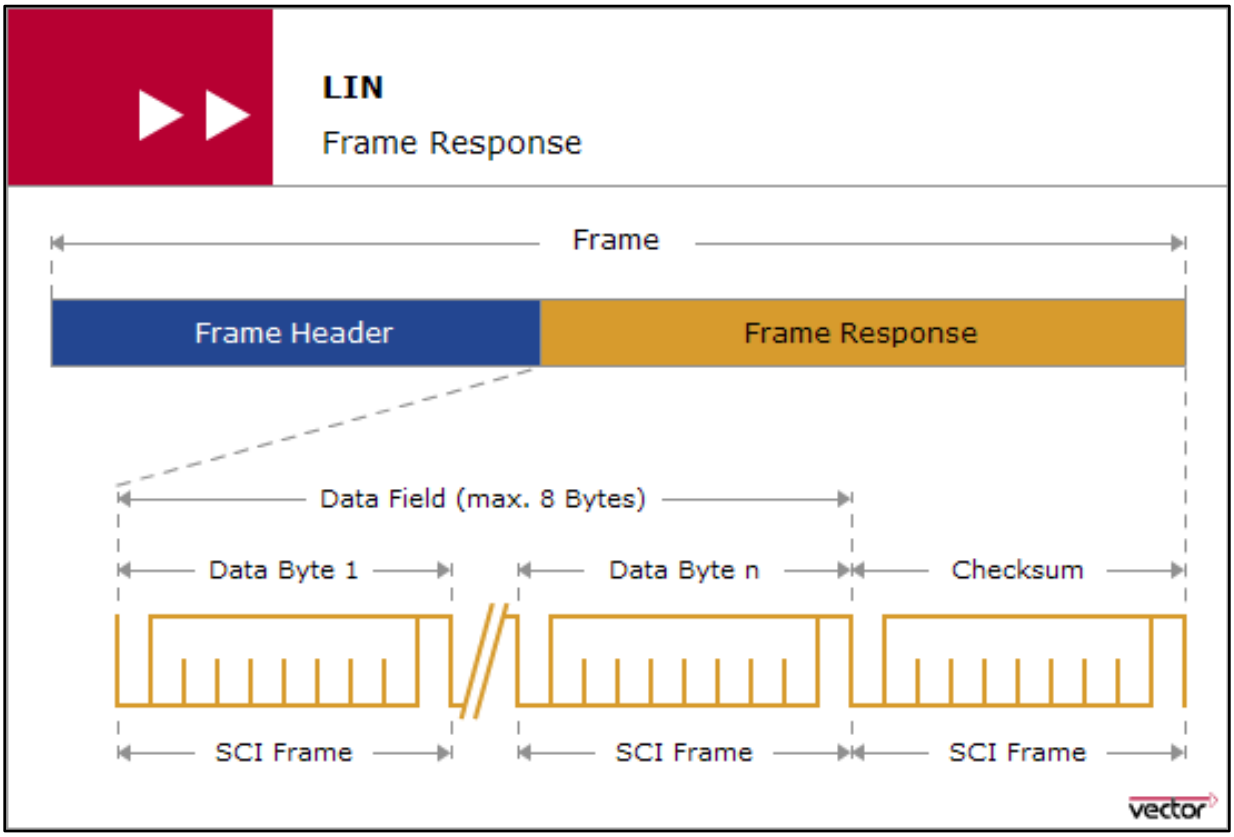


Một khung truyền bao gồm hai phần: **phần tiêu đề (header)** được tạo bởi **tác vụ master** và **phần phản hồi (response)** của **tác vụ slave**:

- Phần tiêu đề: bao gồm 3 trường là break, byte đồng bộ Sync và byte PID.
- Phần phản hồi: bao gồm 2 trường là trường dữ liệu và byte checksum.



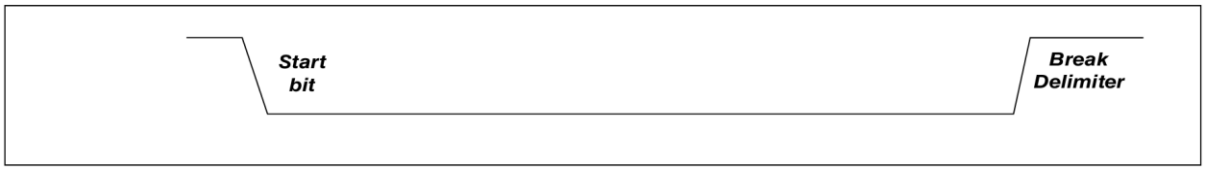
Hình 2.2 Thành phần header trong khung truyền.



Hình 2.3 Thành phần response trong khung truyền.

2.1.2 Trường Break.

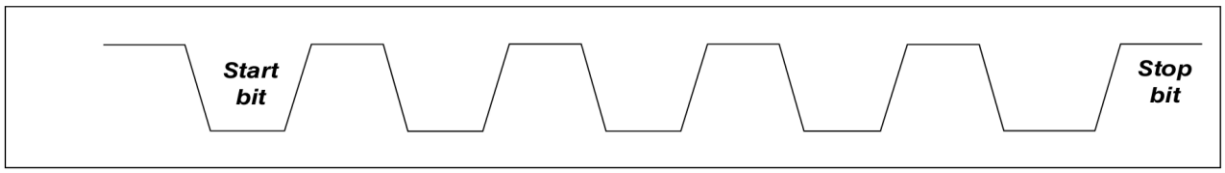
Trường break báo hiệu sự bắt đầu của một khung truyền mới được tạo ra bởi tác vụ master và có độ tối thiểu độ dài 13 bit xung thấp (đã bao gồm start bit) và xung cao break delimiter có độ dài 1 đến 2 bit.



Hình 2.4 Cấu trúc trường bắt đầu break trong khung truyền LIN.

2.1.3 Trường đồng bộ dữ liệu Sync.

Trường đồng bộ dữ liệu Sync là 1 byte có giá trị dữ liệu 0x55 được truyền theo kiểu LBS (bit có trọng số thấp nhất được truyền đi đầu tiên), dùng trong đồng bộ tốc độ truyền (baudrate) giữa master và các slave node.



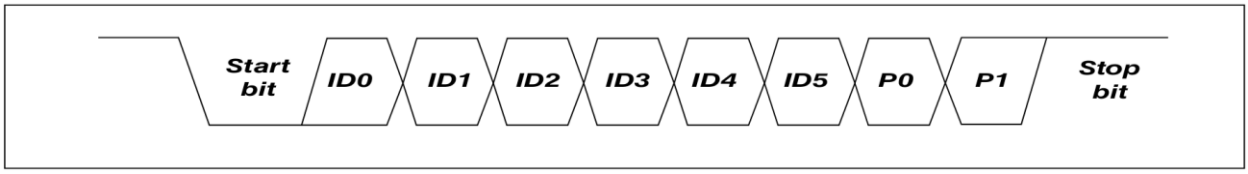
Hình 2.5 Cấu trúc byte đồng bộ Sync trong khung truyền LIN.

2.1.4 Trường PID.

Trường PID, viết đầy đủ là **P**rotected **ID**entifier, là một byte gồm có 2 phần là **Identifier** (6 bit, tính từ bit thứ 0-5) và **Identifier parity** (2 bit P0 và P1), trong đó 2 bit P0 và P1 được tính theo công thức:

P0 = (**ID0 ^ ID1 ^ ID2 ^ ID4**)

P1 = ~ (**ID1 ^ ID3 ^ ID4 ^ ID5**)



Hình 2.6 Cấu trúc trường PID trong khung truyền LIN.

Lưu ý, trường PID có **chức năng định dạng khung truyền** (frame) trong mạng giao tiếp LIN, không phải định dạng các node vật lý.

Bảng 2.1 Các giá PID có valid.

ID[0..5] Dec	ID[0..5] Hex	P0 = ID0⊕ID1⊕ID2⊕ID4	P1 = ¬ ID1⊕ID3⊕ID4⊕ID5	ID-Field 7 6 5 4 3 2 1 0	ID-Field Dec	ID-Field Hex
0	0x00	0	1	1 0 0 0 0 0 0 0	128	0x80
1	0x01	1	1	1 1 0 0 0 0 0 1	193	0xC1
2	0x02	1	0	0 1 0 0 0 0 1 0	66	0x42
3	0x03	0	0	0 0 0 0 0 0 1 1	3	0x03
4	0x04	1	1	1 1 0 0 0 1 0 0	196	0xC4
5	0x05	0	1	1 0 0 0 0 1 0 1	133	0x85
6	0x06	0	0	0 0 0 0 0 1 1 0	6	0x06
7	0x07	1	0	0 1 0 0 0 1 1 1	71	0x47
8	0x08	0	0	0 0 0 0 1 0 0 0	8	0x08
9	0x09	1	0	0 1 0 0 1 0 0 1	73	0x49
10	0x0A	1	1	1 1 0 0 1 0 1 0	202	0xCA
11	0x0B	0	1	1 0 0 0 1 0 1 1	139	0x8B
12	0x0C	1	0	0 1 0 0 1 1 0 0	76	0x4C
13	0x0D	0	0	0 0 0 0 1 1 0 1	13	0x0D
14	0x0E	0	1	1 0 0 0 1 1 1 0	142	0x8E
15	0x0F	1	1	1 1 0 0 1 1 1 1	207	0xCF
16	0x10	1	0	0 1 0 1 0 0 0 0	80	0x50
17	0x11	0	0	0 0 0 1 0 0 0 1	17	0x11
18	0x12	0	1	1 0 0 1 0 0 1 0	146	0x92
19	0x13	1	1	1 1 0 1 0 0 1 1	211	0xD3
20	0x14	0	0	0 0 0 1 0 1 0 0	20	0x14
21	0x15	1	0	0 1 0 1 0 1 0 1	85	0x55

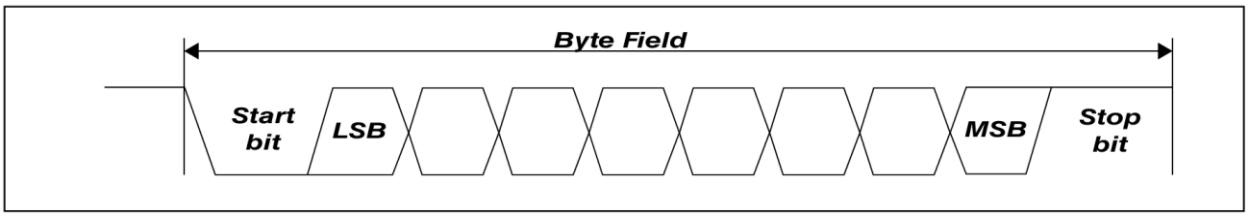
ID[0..5] Dec	Hex	P0 = ID0⊕ID1⊕ID2⊕ID4	P1 = ¬ ID1⊕ID3⊕ID4⊕ID5	ID-Field 7 6 5 4 3 2 1 0	ID-Field Dec	Hex
22	0x16	1	1	1 1 0 1 0 1 1 0	214	0xD6
23	0x17	0	1	1 0 0 1 0 1 1 1	151	0x97
24	0x18	1	1	1 1 0 1 1 0 0 0	216	0xD8
25	0x19	0	1	1 0 0 1 1 0 0 1	153	0x99
26	0x1A	0	0	0 0 0 1 1 0 1 0	26	0x1A
27	0x1B	1	0	0 1 0 1 1 0 1 1	91	0x5B
28	0x1C	0	1	1 0 0 1 1 1 0 0	156	0x9C
29	0x1D	1	1	1 1 0 1 1 1 0 1	221	0xDD
30	0x1E	1	0	0 1 0 1 1 1 1 0	94	0x5E
31	0x1F	0	0	0 0 0 1 1 1 1 1	31	0x1F
32	0x20	0	0	0 0 1 0 0 0 0 0	32	0x20
33	0x21	1	0	0 1 1 0 0 0 0 1	97	0x61
34	0x22	1	1	1 1 1 0 0 0 1 0	226	0xE2
35	0x23	0	1	1 0 1 0 0 0 1 1	163	0xA3
36	0x24	1	0	0 1 1 0 0 1 0 0	100	0x64
37	0x25	0	0	0 0 1 0 0 1 0 1	37	0x25
38	0x26	0	1	1 0 1 0 0 1 1 0	166	0xA6
39	0x27	1	1	1 1 1 0 0 1 1 1	231	0xE7
40	0x28	0	1	1 0 1 0 1 0 0 0	168	0xA8
41	0x29	1	1	1 1 1 0 1 0 0 1	233	0xE9
42	0x2A	1	0	0 1 1 0 1 0 1 0	106	0x6A
43	0x2B	0	0	0 0 1 0 1 0 1 1	43	0x2B
44	0x2C	1	1	1 1 1 0 1 1 0 0	236	0xEC
45	0x2D	0	1	1 0 1 0 1 1 0 1	173	0xAD
46	0x2E	0	0	0 0 1 0 1 1 1 0	46	0x2E
47	0x2F	1	0	0 1 1 0 1 1 1 1	111	0x6F
48	0x30	1	1	1 1 1 1 0 0 0 0	240	0xF0
49	0x31	0	1	1 0 1 1 0 0 0 1	177	0xB1
50	0x32	0	0	0 0 1 1 0 0 1 0	50	0x32
51	0x33	1	0	0 1 1 1 0 0 1 1	115	0x73
52	0x34	0	1	1 0 1 1 0 1 0 0	180	0xB4
53	0x35	1	1	1 1 1 1 0 1 0 1	245	0xF5
54	0x36	1	0	0 1 1 1 0 1 1 0	118	0x76
55	0x37	0	0	0 0 1 1 0 1 1 1	55	0x37
56	0x38	1	0	0 1 1 1 1 0 0 0	120	0x78
57	0x39	0	0	0 0 1 1 1 0 0 1	57	0x39
58	0x3A	0	1	1 0 1 1 1 0 1 0	186	0xBA
59	0x3B	1	1	1 1 1 1 1 0 1 1	251	0xFB
60[a]	*0x3C*	*0*	*0*	*0 0 1 1 1 1 0 0*	*60*	*0x3C*
61[b]	*0x3D*	*1*	*0*	*0 1 1 1 1 1 0 1*	*125*	*0x7D*

ID[0..5] Dec	Hex	P0 = ID0⊕ID1⊕ID2⊕ID4	P1 = ¬ ID1⊕ID3⊕ID4⊕ID5	ID-Field 7 6 5 4 3 2 1 0	ID-Field Dec	Hex
62[c]	0x3E	1	1	1 1 1 1 1 1 1 0	254	0xFE
63[d]	0x3F	0	1	1 0 1 1 1 1 1 1	191	0xBF

Các khung truyền được chia làm 4 kiểu sau: Unconditional frame, Event triggered frame, Sporadic frame và Diagnostic frames dựa trên giá trị của ID (6 bit 0->5). Để hiểu về các kiểu khung truyền này tham khảo mục **2.3.3 FRAME TYPES** trong tài liệu **LIN Physical Layer Spec Revision 2.1 November 24, 2006**.

2.1.5 Trường dữ liệu.

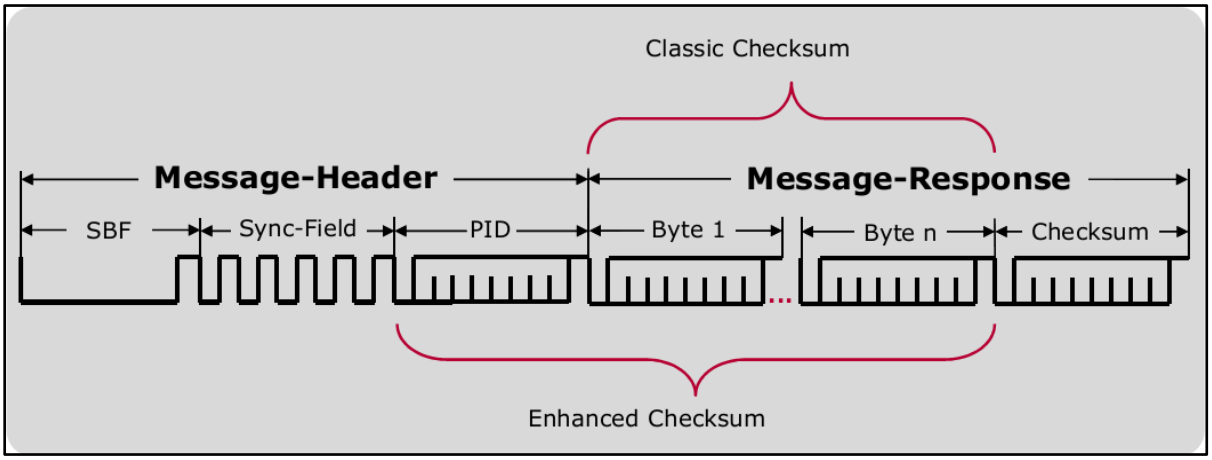
Mỗi byte dữ liệu được truyền theo kiểu LSB kèm với 1 start bit (0b) ở đầu byte tiếp theo là 8 bit dữ liệu và 1 stop bit (1b) ở cuối cùng của byte. Trường dữ liệu có tối đa là 8 byte dữ liệu.



Hình 2.7 Cấu trúc của 1 byte dữ liệu trong khung truyền LIN.

2.1.6 Trường checksum.

Trường checksum là trường cuối cùng trong cấu trúc khung truyền, bao gồm nghịch đảo 8 bit của tổng có dư của các byte dữ liệu (kiểu classic) hoặc của các byte dữ liệu và trường PID (kiểu enhanced).



Hình 2.8 Hai kiểu checksum.

- Kiểu Classic Checksum áp dụng cho các phiên bản cũ: LIN 1.1, LIN 1.2 và LIN 1.3.
- Kiểu Enhanced Checksum áp dụng cho các phiên bản mới hơn: LIN 2.0, LIN 2.1 và LIN 2.2.

Ví dụ, tính giá trị byte checksum kiểu classic với bản tin có 4 byte dữ liệu:

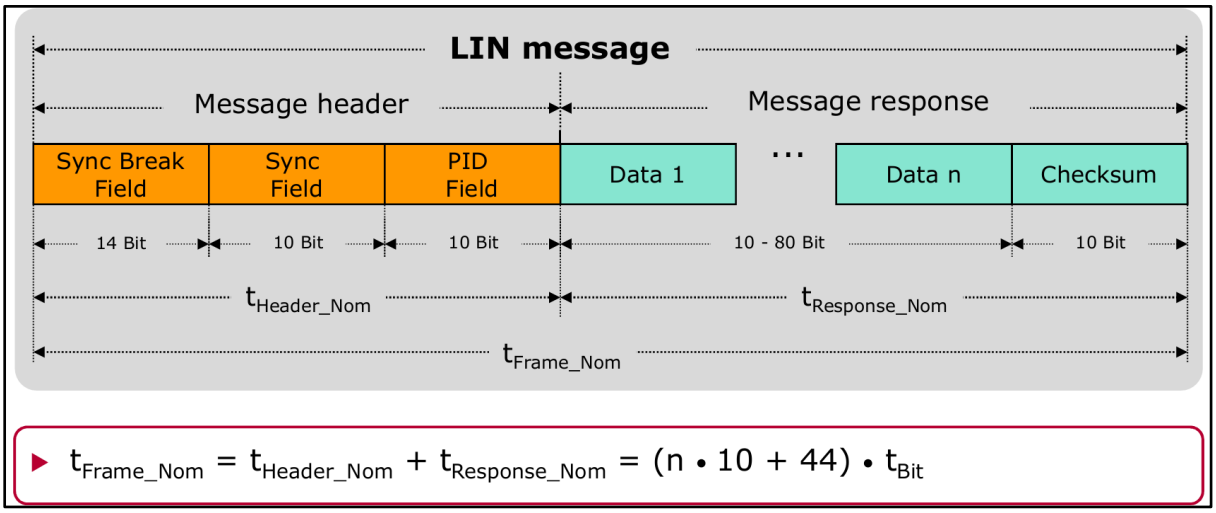
Data0 = 0x1A; Data1 = 0x4C; Data2 = 0xAA; Data3 = 0xF4

Kết quả: **Checksum = 0xF9**

Bảng 2.2 Ví dụ tính byte checksum.

Dữ liệu	Hex	Dư	D7	D6	D5	D4	D3	D2	D1	D0
0x1A	0x1A		0	0	0	1	1	0	1	0
+ 0x4C = **Add Carry**	0x66 0x66	0	0 0	1 1	1 1	0 0	0 0	1 1	1 1	0 0
+ 0xAA = **Add Carry**	0x10 0x11	1	0 0	0 0	0 0	1 1	0 0	0 0	0 0	0 1
+0xF4 = **Add Carry**	0x05 0x06	1	0 0	0 0	0 0	0 0	0 0	1 1	0 1	1 0
Nghịch đảo	0xF9		1	1	1	1	1	0	0	1
0x06 + 0xF9	0xFF		1	1	1	1	1	1	1	1

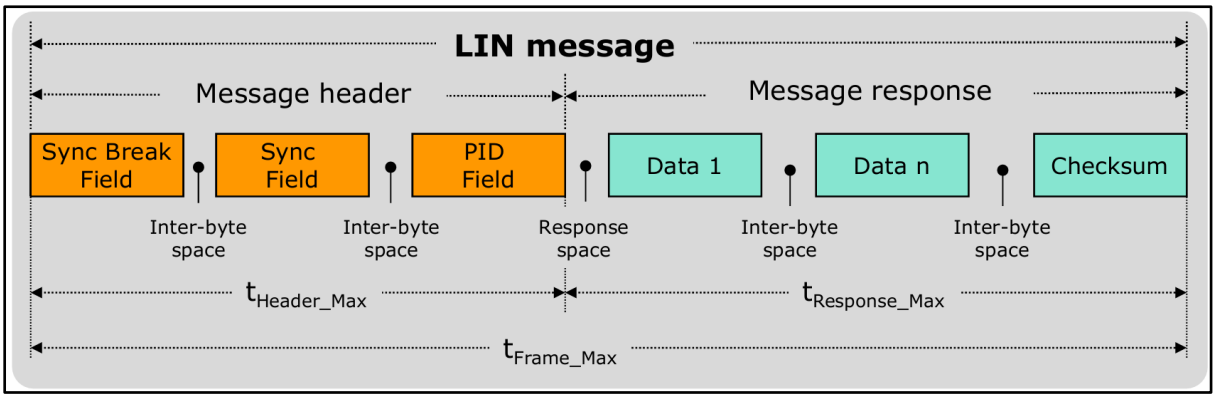
2.1.7 Lập lịch.



Hình 2.9 Thời gian một khung truyền LIN.

Việc thiết lập một bảng lịch được dựa trên độ rộng thời gian của các khung truyền. Trong thực tế, khung truyền có cấu trúc với hầu hết các trường dữ liệu là cố định, ngoại trừ trường dữ liệu thay đổi độ dài từ 1 đến 8 byte. Do đó, chúng ta có thể tính thời gian chuẩn khi truyền một khung như công thức trên, với n là số byte dữ liệu và tBit là thời gian của 1 bit.

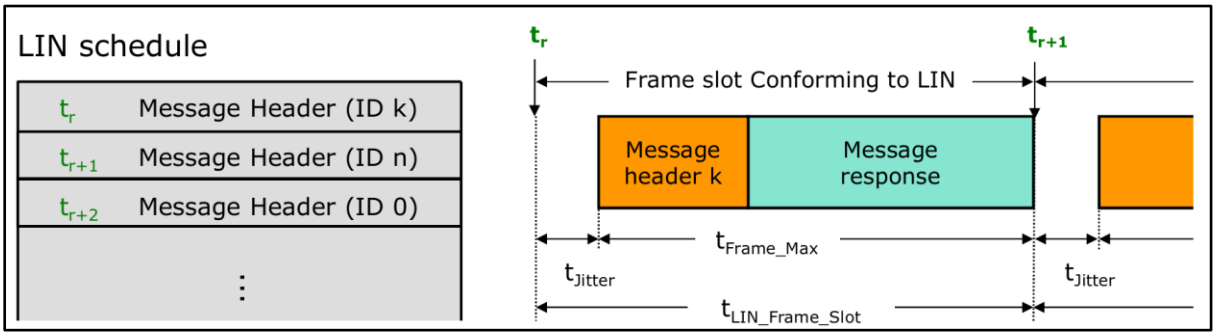
Do các slave node thường có hiệu suất thấp nên thường có trễ truyền xảy ra, theo giao thức có thể dự trữ thời gian (time reserve) tới 40% để truyền một khung truyền.



▶ $t_{Frame_Max} = 1.4 \bullet t_{Frame_Nom} = [1.4 \bullet (n \bullet 10 + 44)] \bullet t_{Bit}$

Hình 2.10 Timing thực tế của khung truyền.

- Lập bảng hoạt động của mạng:



Hình 2.11 Lập lịch cho mạng LIN.

Độ rộng một slot khung truyền:

tLIN_Frame_Slot = 1.4 • tFrame_Nom + tJitter = tFrame_Max + tJitter

Trong đó, tJitter là khoảng thời gian an toàn giữa hai khung liên tiếp, tránh xảy ra hai khung chồng lên nhau. Do quá trình truyền nhận thông tin trong mạng đã được lên lịch sẵn nên không xảy ra hiện tượng quá tải. Mỗi time slot có độ dài khác nhau phụ thuộc vào độ dài phần response (cụ thể là số byte dữ liệu).

Liên quan đến thông số thời gian khung truyền tối đa **tFrame_Max**, trong driver có thực hiện tính năng có tên Disable_Timeout_Frame cho phép chờ phần response sau khoảng thời gian **tFrame_Max.**

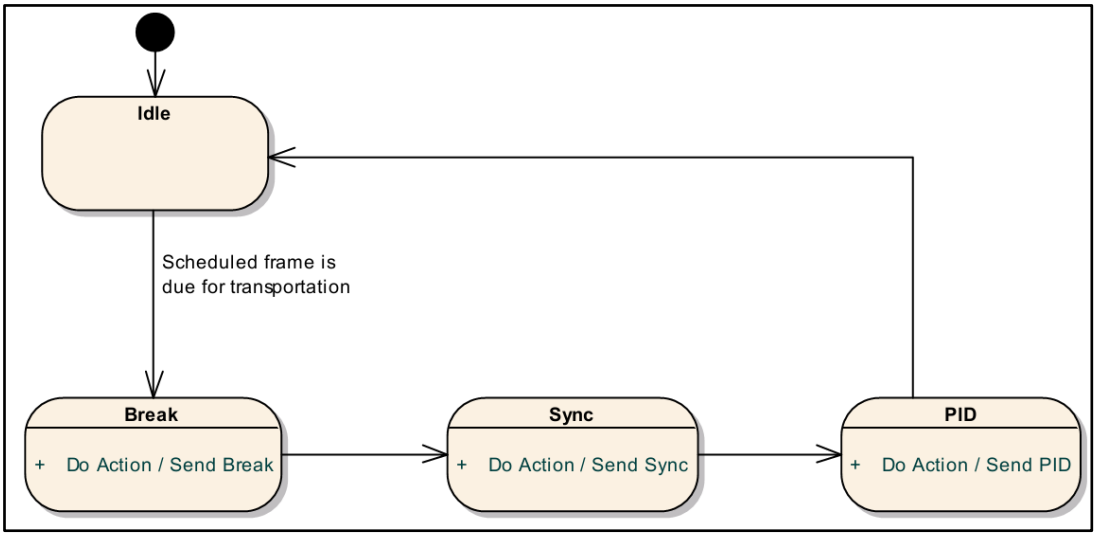
2.2 Giới thiệu các chức năng.

2.2.1 Các đặc tính/ thông số cơ bản.

- Chi phí triển khai thấp, do chỉ sử dụng 1 đường bus tín hiệu (tiêu chuẩn ISO 9141 nâng cao).
- Tốc độ truyền nhận từ 1Kbit/s đến 20 Kbit/s (giới hạn vì lý do EMI).
- Mô hình 1 Master, nhiều Slave (tối đa 15 Slave).
- Giảm chi phí triển khai do tận dụng phần cứng giá rẻ UART/ SCI/ GPIO/ FLEXIO/ ESCI phổ biến trên hầu hết các dòng vi điều khiển hiện nay.
- Tự đồng bộ đường truyền mà không cần clock thạch anh, bộ dao động nội. Vì vậy, đặc điểm này cũng giảm chi phí đáng kể triển khai nền tảng phần cứng.
- Do không dùng clock, vì vậy sai số xảy ra cho phép trong khoảng +/- 14%.
- Thời gian trễ được đảm bảo khi truyền tín hiệu (có khả năng dự đoán thời gian truyền khung dữ liệu).
- Điện áp bus: 9 - 18V (chuẩn: 12V).
- Độ dài tối đa đường Bus: 40m (https://en.wikipedia.org/wiki/Local_Interconnect_Network).

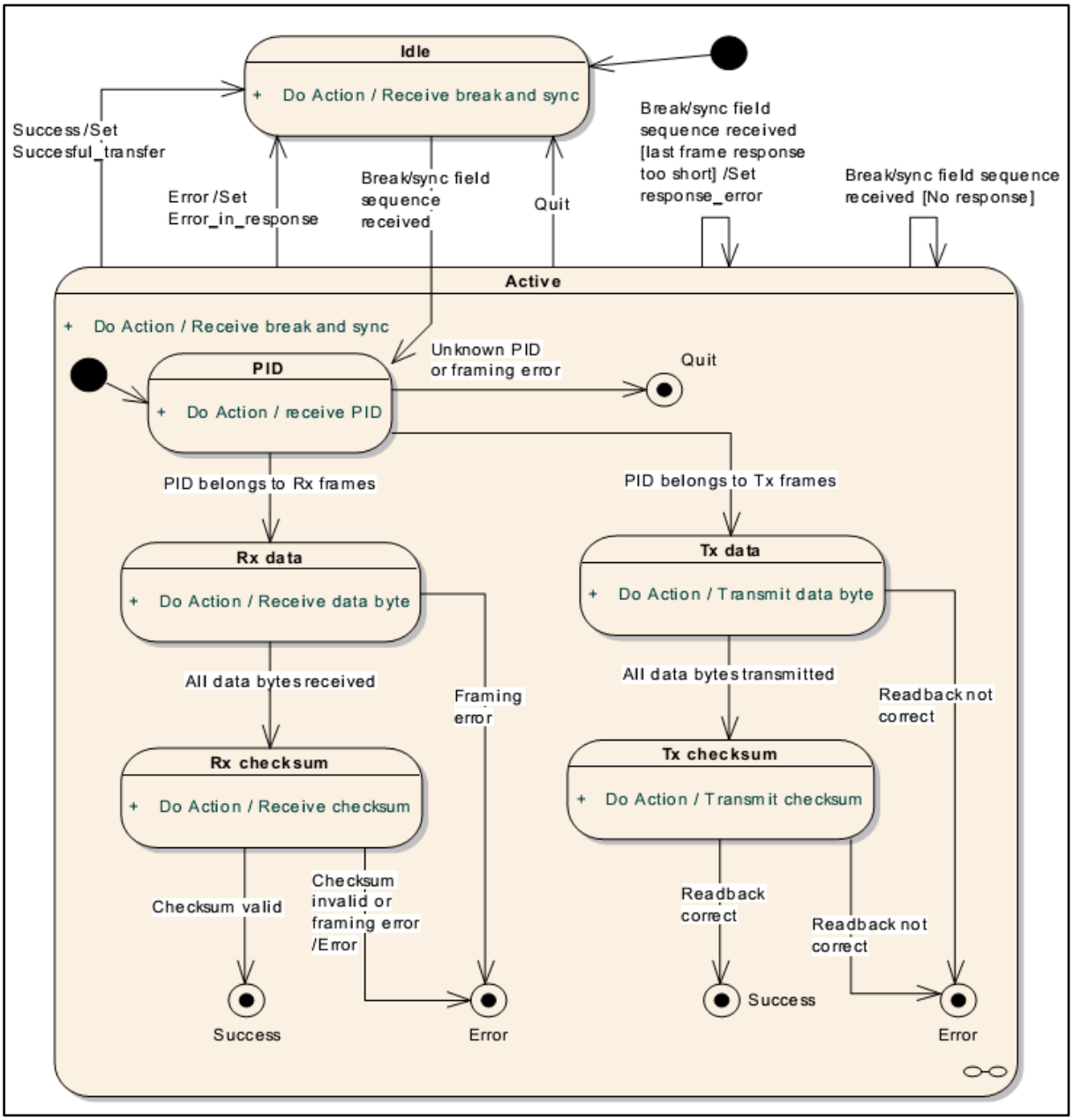
2.2.2 Khái niệm cơ bản.

- **Tác vụ Master:**
  - Xác định thứ tự và mức độ ưu tiên của bản tin.
  - Có quyền kiểm soát toàn bộ Bus và giao thức. Master kiểm soát thông báo nào vào thời điểm nào sẽ được chuyển qua bus, đồng thời nó cũng phát hiện và xử lý lỗi khi xảy ra bằng nhiệm vụ gửi 3 trường: Sync Break, đồng bộ Sync và PID.
  - Phục vụ như là một tham chiếu với clock cơ sở của nó (cần thiết có clock ổn định).
  - Nhận tín hiệu báo thức Wakeup từ các node Slave.



Hình 2.12 Các trạng thái của tác vụ Master.

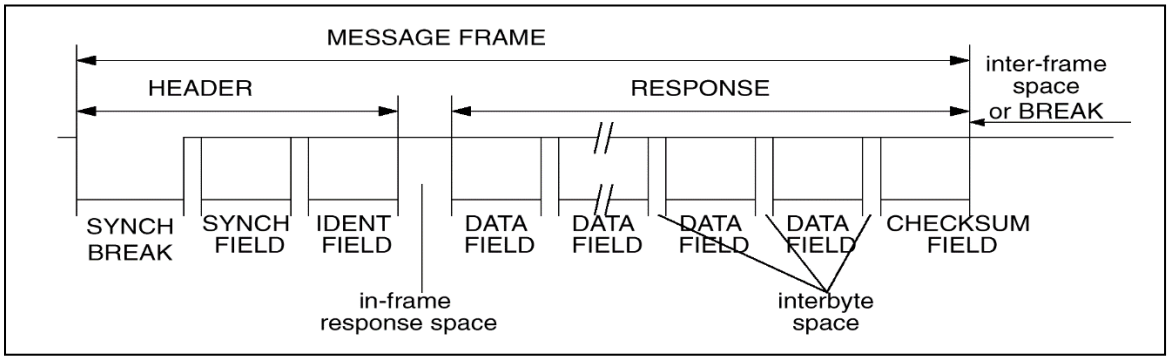
- **Tác vụ Slave:**
  - Là một trong 2 đến 16 node phần tử trên đường Bus.
  - Nhận hoặc truyền dữ liệu khi xác định được một frame có ID tương ứng (đã quy định sẵn) được gửi từ Master. Trong trường hợp truyền dữ liệu, slave sẽ gửi từ 1 đến 8 byte dữ liệu và 1 byte checksum.
  - Node Master có thể thực hiện được cả tác vụ slave.



Hình 2.13 Các trạng thái của tác vụ Slave.

- **Cách tiếp cận thời gian (Time Triggered Approach):**

Trong mạng LIN, độ dài dữ liệu mỗi khung truyền đã xác định sẵn (fix cứng) do đó ta có thể tính thời gian tối thiểu để truyền khung truyền đó. Giao thức LIN cũng quy định mỗi khung truyền được thực hiện trong thời gian tối đa 140% khung truyền tối thiểu, tức là trong trường hợp không có trễ. Từ đó, ta tính được thời điểm tối thiểu để bắt đầu truyền 2 khung liên tiếp và dễ dàng thiết lập mạng chạy theo chế độ tự động.



Hình 2.14 Thời gian tối đa của khung truyền.

2.2.3 Các kiểu dữ liệu.

Tham khảo phần **8.2 Type definitions** (trang 34), tài liệu Specification of LIN Driver V1.5.0 R4.0 Rev 3.

2.2.4 Các hàm API.

Tham khảo phần **8.3 Function definitions** (trang 37), tài liệu Specification of LIN Driver V1.5.0 R4.0 Rev 3.

2.3 Các nền tảng phần cứng (IPVs).

Hiện tại, driver LIN hỗ trợ trên 4 nền tảng phần cứng khác nhau tương ứng với 4 IPV: **LinFlexD**, **eSCI**, **SCI** và **LPUART**. Bốn phần cứng có một số đặc điểm khác nhau vì vậy dẫn đến cách implement trong driver cũng khác nhau, cụ thể như sau:

Thứ tự	Phần cứng	LINFLEXD	ESCI	SCI	LPUART
1	Break length (bit)	(11 -> 23), 36, 50	10, 13	10, 13	10, 13
2	Thanh ghi dữ liệu (byte)	LINFlexD_BDRM - 4 LINFlexD_BDRL - 4	eSCI_LRR – 1 eSCI_LTR – 1	SCIDRL – 1	LPUART_DATA - 1
3	DMA	2/16 kênh (LINFlexD[0], LINFlexD[1])	2/6 kênh (eSCI_A, eSCI_B)	Không	3/3 kênh (LPUART0, LPUART1, LPUART2)
4	Timer	Có	Có	Không	Không
5	Chế độ Sleep	Có	Stop	Không	Không
6	Cờ wakeup	Có	Có	Không	Không
7	Tính Checksum	Có	Có	Không	Không
8	Tính PID	Có	Có	Không	Không

(1) Số độ dài trường Break của LinFlexD hỗ trợ là 15 case, các phần cứng còn lại là 2 case 10bit và 13bit. Điều này dẫn đến số lượng option cấu hình break length trong file Lin.xdm là khác nhau giữa các IPVault.

(2) Số lượng byte trong các thanh ghi dữ liệu mà mỗi phần cứng hỗ trợ không ảnh hưởng đến việc thực hiện code driver, nó chỉ khác nhau trong quá trình thực thi code trên vi điều khiển. Đối với phần cứng LinFlexD, do có 8 byte dữ liệu nên khi gửi một khung truyền chỉ cần duy nhất một lần vào ngắt để xử lý truyền. Các phần cứng còn lại có thanh ghi dữ liệu 1 byte, do đó cơ chế xử lý truyền/ nhận là vào ngắt mỗi lần truyền hoặc nhận 1 byte.

(3) Tính năng truyền nhận sử dụng DMA có trên 3 phần cứng: LinFlexD, eSCI, LPUART và đã được implement trên IPV_ESCI. Đây là tính năng nâng cao không có requirement từ AUTOSAR, vì vậy developer có thể cân nhắc để thêm vào driver nếu cần thiết.

(4) Phần cứng timer được sử dụng trong tính năng đo thời gian truyền/ nhận header và response. Theo tài liệu LIN Spec 2.1 (phần 2.3.2 FRAME LENGTH), độ dài một frame tối đa là 140% khung tiêu chuẩn

(xem lại 2.1.7 Lập lịch) do đó nếu như thời gian truyền nhận vượt ngưỡng thì status trả về sẽ là LIN_RX_NO_RESPONSE hoặc LIN_RX_ERROR. Liên quan đến tính năng này, bên khách hàng của NXP có request thêm tính năng Disable_Frame_Timeout tức là nếu timeout truyền frame vượt quá 140% frame tiêu chuẩn thì vẫn tiếp tục chờ tín hiệu từ slave cho đến khi xuất hiện thêm một frame mới hoặc hết thời gian chạy test.

(5) Sleep: là chế độ hoạt động hiệu năng thấp trong giao thức LIN, ở mode này sẽ hạn chế tiêu thụ năng lượng bằng cách không có các hoạt động truyền nhận frame, chỉ duy nhất có tính năng truyền và nhận tín hiệu wakeup. Tuy nhiên, thực chất chỉ có hai phần cứng hỗ trợ chế độ này là LinFlexD, eSCI khi có các bit enable Sleep (LINCR1_SLEEP, LCR1_LRES). Hai phần cứng SCI và LPUART không có chế độ hoạt động này nên khi thực hiện dev hàm Lin_GoToSleep và Lin_GoToSleepInternal, ta mô phỏng nó bằng cách ngắt tất cả các ngắt truyền/ nhận, ngắt lỗi ngoại trừ ngắt dectect xung (xung wakeup).

(6) Wakeup: Tương ứng với chế độ Sleep là trạng thái wakeup, LinFlexD và eSCI có chế độ sleep nên tồn tại cờ wakeup để thoát chế độ ngủ (LINSR_WUF, LCR1_WU) và hỗ trợ tạo request xung wakep (LINCR2_WURQ, LCR1_WU). Hai phần cứng LPUART và SCI sử dụng một biến software để mô phỏng cờ wakeup; trong khi đó gửi request wakeup bằng cách gửi một byte dữ liệu (đáp ứng độ rộng xung wakeup từ 250us - 5ms).

(7) Phần cứng LinFlexD và eSCI hỗ trợ tính tự động giá trị byte checksum để truyền đi trong trường hợp khung truyền có kiểu Master_Response và kiểm tra lại dữ liệu đã nhận trong trường hợp Slave_Response. Phần cứng SCI, LPUART thì không có chức năng tự động tính byte checksum vì vậy, ta cần phải dev thêm hàm software (Lin_SCI_ChecksumCalc, Lin_LPUART_ChecksumCalc) để thực hiện chức năng này.

(8) Tương tự như (7), phần cứng SCI, LPUART không hỗ trợ tính tự động byte PID nên ta phải dev hàm software (Lin_SCI_PIDCalc, Lin_LPUART_PIDCalc).

2.3.1 Phần cứng LINFLEXD.

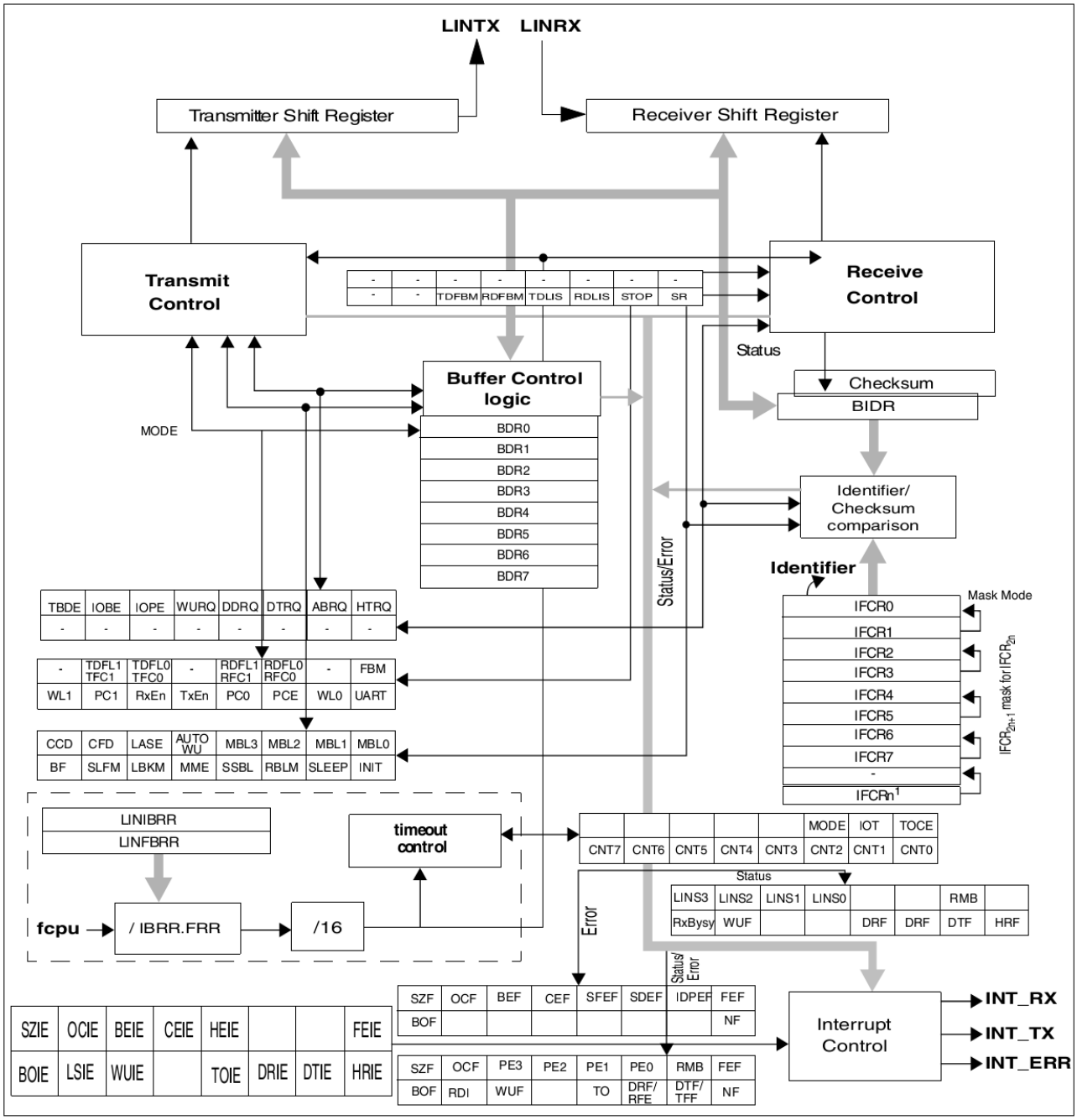
Các platform có phần cứng LinFlexD: Treerunner (IMXVS4), Panther (MPC574XP), Calypso (MPC574XG), Rainier (MPC574XR), Halo (MAC57DXXX), Racerunner Ultra& Skinny (S32R27X).

- **Mô tả cơ chế hoạt động của phần cứng:**

Phần cứng LINFlex lấy xung nhịp hoạt động từ clock của CPU (tùy từng vi điều khiển mà có đầu ra peripheral khác nhau) qua bộ chia tần **IBRR.FRR (= LDIV)** được tính theo công thức sau: **Baud rate = LIN_CLK/(16 x LDIV)** hay: **LDIV = LIN_CLK/(16 x Baudrate)**. Để cấu hình tốc độ cho khung truyền LIN thì ta dùng công thức sau để tính toán và ghi giá trị cấu hình vào hai thanh ghi IBRR và FRR. Bộ chia tần cũng là đầu vào của khối timeout control - đếm thời gian xử lý khung truyền – dùng trong các trường hợp kiểm tra lỗi timeout của header và response.

- Bộ điều khiển truyền (Transmit Control): Điều khiển các thanh ghi CR1, CR2 và UARTCR điều khiển hoạt động truyền dữ liệu.
- Bộ điều khiển nhận (Receive Control): Cấu hình các trường checksum, PID thông qua thanh ghi BIDR.
- Bộ điều khiển ngắt (Interrupt Control): Bao gồm các thanh ghi enable/ disable các cờ ngắt LINIER, thanh ghi trạng thái ngắt lỗi LINESR và thanh ghi trạng thái LINSR. Bộ điều khiển ngắt được tác động từ 2 bộ điều khiển truyền và nhận, các ngắt của Lin chia làm 3 loại ngắt: Ngắt Tx, Rx và ngắt lỗi.
- Các thanh ghi buffer/ dịch: Dữ liệu trước khi truyền được các bộ điều khiển truyền ghi vào thanh ghi buffer sau đó đưa ra thanh ghi dịch (transmiter shift register) đẩy trực tiếp ra chân vi điều khiển Tx. Ngược lại dữ liệu nhận được từ bus sẽ chuyển từ thanh ghi dịch (receive shift register) đến thanh ghi buffer.

Lưu ý: Các thanh ghi IFCRn chỉ dùng trong mode Slave.



Hình 2.15 Cấu tạo phần cứng LinFlexD trên platform RaceRunner Ultra.

Phần quan trọng khi nhắc đến giao tiếp LIN, đó là tính năng phát hiện các lỗi trong quá trình truyền thông. Nền tảng LINFlexD là phần cứng hỗ trợ đầy đủ những dạng lỗi truyền nhận:

- **Header error:** Lỗi xảy trong quá trình nhận header, vì vậy nó chỉ xảy ra trên các node slave. Do đó, header error không được implement trong driver của team MCAL. Header error xảy ra khi 1 trong 3 trường hợp sau xảy ra:

- *Sync Del error (SDEF):* Nếu dấu phân cách sync delimiter có độ rộng nhỏ hơn 1 bit time (trong spec 2.1 đề xuất là 2 bit) thì được xem xét là ngắn và node slave sẽ hủy bỏ đồng bộ đối với header hiện tại. Ngắt lỗi nhận header sẽ được bật lên nếu bit enable ngắt LINEIER_HEIE được set.
- *Sync field error (SFEF):* Trường Sync xảy ra lỗi nhận

Trường hợp 1: Bit LINCR1_LASE = 1. Lỗi sai lệch được kiểm tra bằng cách so sánh tốc độ hiện tại (bộ dao động của slave) với trường sync đã nhận (liên quan đến bộ dao động của master). Sự kiểm tra này trên cơ sở đo lường giữa sườn xuống đầu tiên và sườn xuống cuối cùng của trường Sync. Gọi độ sai lệch này là D:

Nếu: D > 14.0625% thì xảy ra ngắt lỗi SFEF.

Ngược lại, nếu D < 14.84375% thì không có lỗi.

Trong khoảng: 14.0625% < D < 14.84375% thì phụ thuộc vào dephasing giữa của đường RDI và LIN_CLK.

Ngoài ra, lỗi overflow trong quá trình đo trường Sync dẫn đến tràn thanh ghi chia.

Trường hợp 2: Bit LINCR1_LASE = 0. Trong trường hợp này, giá trị trường Sync là 0x55 được xem như một byte dữ liệu thông thường. Nếu slave nhận được byte sync có giá trị 0x55 là thành công.

Xảy ra bất kì sứ không nhất quán của trường Sync thì slave ngay lập tức sẽ thoát khỏi trạng thái Sync_Field và khung truyền sẽ bị hủy. Cờ ngắt lỗi SFEF bị bật.

- *Identifier Parity error (IDPEF):* Như đã trình bày trong mục 2.1.4 về cấu trúc trường PID, byte này có 2 bit parity để kiểm tra lỗi truyền của PID. Hai bit này được tính toán tự động bằng phần cứng từ giá trị 6 bit D0 – D5, nếu có lỗi xảy ra thì bit LINCR2_IOPE sẽ bật lên và slave sẽ thoát khỏi trạng thái xác định PID.
- **Bit error:** Cờ này được dùng trong chế độ truyền khi giá trị đọc lại (readback) từ bus LIN không khớp với giá trị truyền. Việc kiểm tra lỗi trên từng bit được đảm bảo nếu như trễ truyền nhỏ hơn (1 bit – 6 x chu kì LIN_CPU). Lưu ý, bit error không được kiểm tra trong khi truyền trường Break.

Ví dụ, nếu baudrate = 20 Kbit/s thì 1-bit time = 50 µs

LIN_CLK = 80 MHz, 6 x chu kì LIN_CLK = 75 ns

Do đó, (1-bit time) - (6 x chu kì LIN_CLK) = 49.925 µs

- Nếu dấu cách break delimiter không được xác định bởi chính master trong vòng 1 bit time sau khi truyền do trễ truyền hoặc lỗi bus thì cờ bit lỗi LINCR2_IOBE sẽ bật lên, PID sẽ không được thay thế trong thanh ghi BIDR.

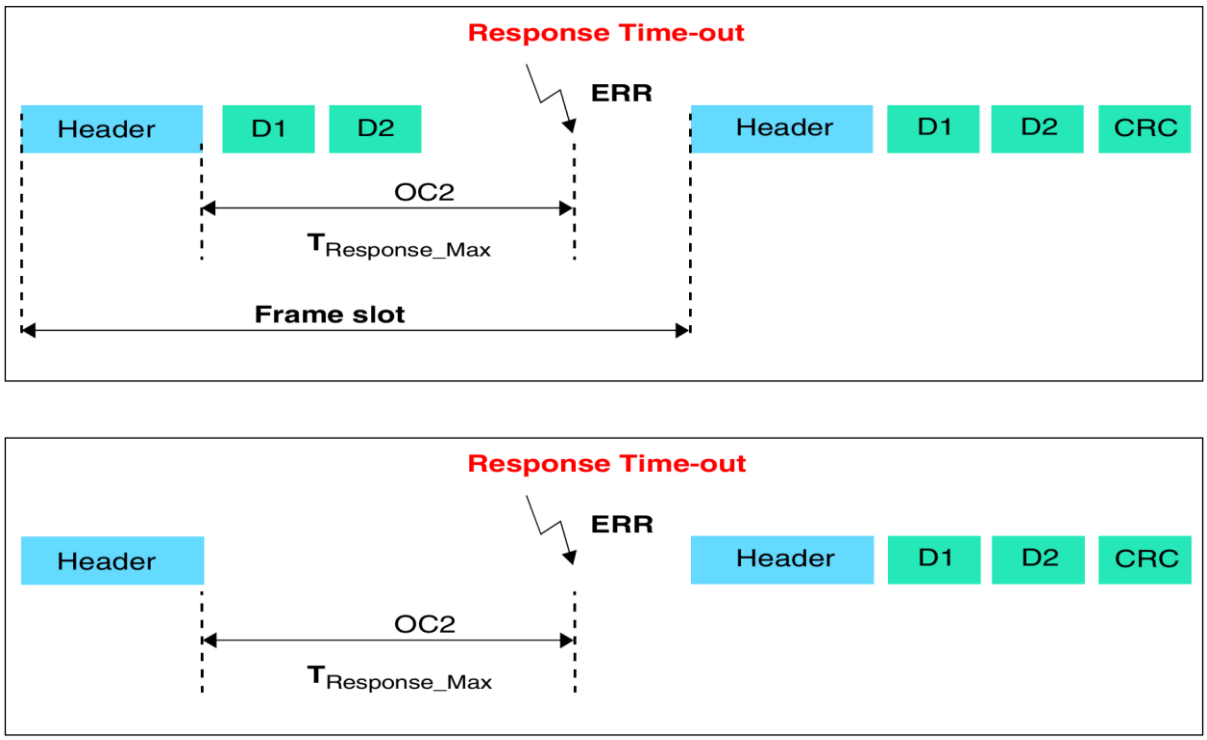
- Tương tự, sườn xuống của byte dữ liệu không được xác định sau khoảng 1 bit time bởi master thì bit lỗi cũng bị bật lên. Do đó, dữ liệu, byte checksum trong thanh ghi BDR và CFR tương ứng không được đảm bảo.

- **Framing error:** Lỗi truyền này xảy ra khi stop bit của byte đang nhận hiện tại (byte Sync, byte PID, các byte dữ liệu, byte checksum) có giá trị là mức thấp 0b. Phần cứng LinFlexD sẽ hủy bỏ khung truyền hiện tại và trả lại trạng thái Idle. Một ngắt được bật nếu như bit LINEIER_FEIE đã được set. Byte dữ liệu xảy ra lỗi framing vẫn được chuyển tới buffer nhưng cờ hoàn thành nhận dữ liệu DRF sẽ không được set.
- **Checksum error:** Lỗi này xảy ra nếu như giá trị byte checksum được tính toán tự động bởi phần cứng không khớp với giá trị của byte checksum nhận được. Phần cứng LinFlexD sẽ hủy bỏ khung truyền hiện tại và trả lại trạng thái Idle. Một ngắt được bật nếu như bit LINEIER_CEIE đã được set.
- **Overrun error:** Khi bộ đệm bản tin đã đầy (RMB được set), việc tiếp nhận bản tin tiếp theo sẽ dẫn đến lỗi overrun và bản tin sẽ bị mất. Phần cứng sẽ báo hiệu điều kiện overrun bằng cách set bit BOF. Tin nhắn bị mất phụ thuộc vào set bit RBLM:

- RBLM = 0: bản tin cũ trong thanh ghi buffer sẽ bị ghi đè bằng bản tin mới nhất.
- RBLM = 1: bản tin mới nhất sẽ bị hủy bỏ và bản tin cũ nhất sẽ vẫn tồn tại trong phần mềm.

Trong trường hợp slave, nếu thanh ghi buffer không được phát hành (RMB không được reset) trước khi nhận PID tiếp theo và bit RBLM được set thì PID cùng với dữ liệu bị hủy bỏ.

- **Timeout error:**



Hình 2.16 Hai trường hợp lỗi timeout.

Cơ chế timeout response:

- *Chế độ Master:*

Trường LINOCR_OC2 được load giá trị Nominal_time_out + LINFlexD_LINTCSR[CNT] ở cuối stop bit của trường PID (nominal_time_out = 1.4 × ((DFL + 2) × 10 bit time). Lỗi timeout xảy ra khi giá trị trường counter LINTCSR_CNT bằng giá trị trường LINOCR_OC2 đã được set từ ban đầu khi truyền khung. Trường hợp không có dữ liệu nhận được hoặc phần response không hoàn thành thì lỗi timeout đều set.

- *Chế độ Slave:* Phần này không được implement trong driver do không nằm trong phạm vi tài liệu Autosar.

Trường hợp định dạng nhận được quản lý bởi một bộ lọc, thì việc thực hiện cũng tương tự như trên chế độ master.

Trường hợp định dạng nhận không được quản lý bởi bộ lọc thì giá trị DFL phải được cập nhật bởi phần mềm sau khi trường PID được nhận. Cuối trường PID, trường LINOCR_OC2 được thiết lập giá trị 36 (khoảng thời gian response lớn nhất có thể, nếu tính 8 byte dữ liệu thì: 40% x (10 bit time x (8 byte dữ liệu + 1 byte checksum))). Cuối byte dữ liệu đầu tiên, LINOCR_OC2 sẽ được thiết lập lại với giá trị DFL đã được cập nhật.

Cơ chế timeout header:

- *Chế độ Master:* Header thì được tạo bởi master node do đó chỉ có 2 trường hợp xảy ra: không có lỗi trên bus và timing là chính xác (độ dài của header là chuẩn) hoặc có lỗi xảy ra trên bus và sẽ có cờ ngắt lỗi được bật như Bit error. Do vậy, đối với chế độ master thì header timeout không có ý nghĩa và nó bị vô hiệu hóa.
- *Chế độ Slave:* Gía trị thiết lập vào trường HTO là 45.

header_nominal = 13 + 2 + 10 + 10 = 35 Tbit

header_max = 1.4 × Header_nominal = 49 Tbit

Tính độ lệch 14% clock, header_max sẽ bằng: 49 × 1.14 = 56 Tbit

Nếu bộ đếm bắt đầu sau trường break 11 Tbit (break duration), giá trị của HTO là 56 –11 = 45 Tbit.

Stuck at zero timeout error:

Nếu mức điện áp trên bus LIN ở mức thấp ít nhất 100 bit thì bit LINESR_SZF sẽ được set. Cùng một xung thấp kéo dài thì bit LINESR_SZF sẽ bật lên với 87 bit.

- **Noise:**

Trong quá trình nhận dữ liệu, mỗi bit sẽ được lấy mẫu 16 lần và phần cứng sẽ so sánh giá trị điện áp tại các lần lấy mẫu thứ 8, 9 và 10. Nếu bất kì 1 trong 3 mẫu có giá trị khác thì cờ lỗi sẽ được bật lên. Khi: UARTCR_OSR = 8 thì các mẫu thứ 2, 3 và 4 sẽ được dùng để xác định lỗi nhiễu noise.

Bảng 2.3 Các thanh ghi trong phần cứng LINFLEXD.

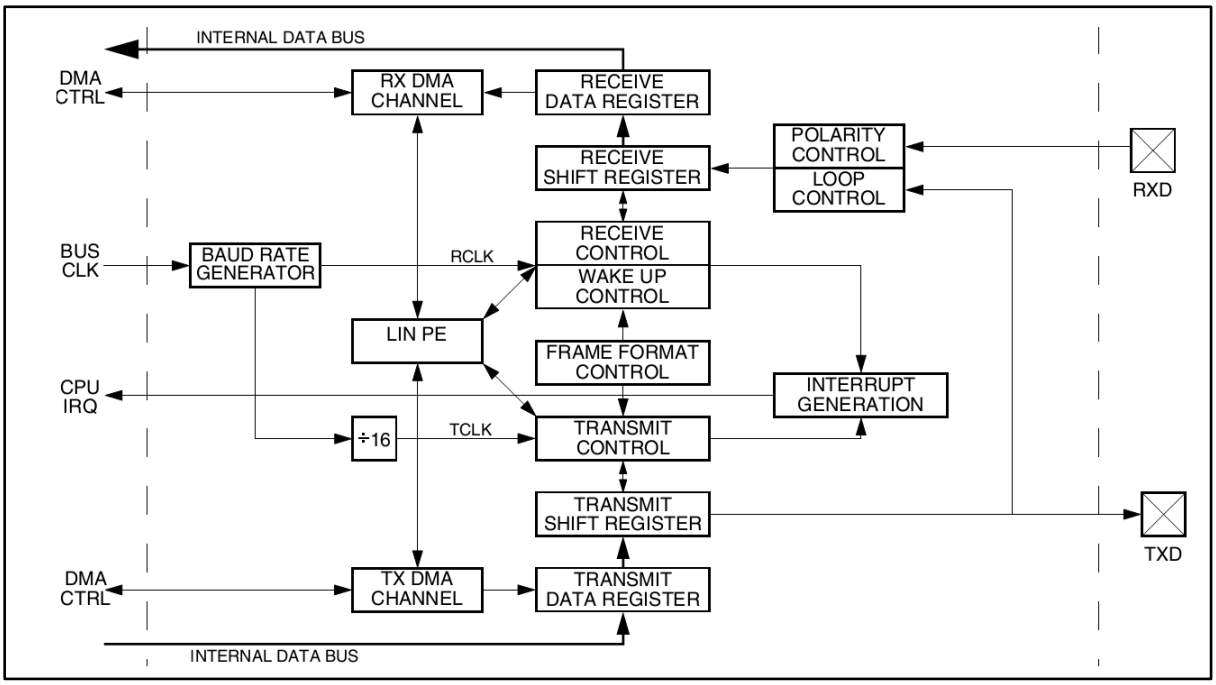
Thanh ghi	Chức năng	Mô tả
LINCR1	Cấu hình	Cấu hình các chế độ hoạt động và tính năng của LinFlexD
LINIER	Cấu hình	Gồm các bit cho phép ngắt
LINSR	Trạng thái	Các bit trạng thái
LINESR	Trạng thái	Gồm các cờ ngắt lỗi
UARTCR	Cấu hình	Điều khiển chế độ UART
UARTSR	Trạng thái	Các cờ trạng thái trong chế độ UART (chế độ LIN không sử dụng)
LINTCSR	Cấu hình	Gồm các bit điều khiển và trạng thái của tính năng timeout
LINOCR	Cấu hình	Chứa giá trị so sánh với trường LINTCSR[CNT]
LINTOCR	Cấu hình	Chứa giá trị timeout response và header
LINFBRR	Cấu hình	Chứa giá trị phần phân số của tốc độ truyền
LINIBRR	Cấu hình	Chứa giá trị phần nguyên của tốc độ truyền
LINCFR	Cấu hình	Tùy chọn phần cứng hoặc phần mềm tính giá trị byte checksum
LINCR2	Điều khiển	Gồm các bit trạng thái liên quan đến các hoạt động của buffer dữ liệu
BIDR	Cấu hình	Gồm các bit cung cấp thông tin về định dạng khung truyền
BDRL	Dữ liệu	Gồm 4 byte đầu tiên của buffer dữ liệu
BDRM	Dữ liệu	Gồm 4 byte sau của buffer dữ liệu
IFER	Cấu hình	Bật/ tắt bộ lọc cụ thể
IFMI	Chỉ số	Chỉ đọc bởi RAM
IFMR	Cấu hình	Cấu hình chế độ các bộ lọc
IFCRn	Cấu hình	Chỉ dùng trong chế độ slave
GCR	Cấu hình	Thiết lập định dạng các byte trong khung truyền, thiết lập lại giá trị các thanh ghi
UARTPTO	Cấu hình	Chứa giá trị timeout thiết lập sẵn trong chế độ UART (chế độ LIN không sử dụng)
UARTCTO	Trạng thái	Chứa giá trị timeout hiện tại trong chế độ UART (chế độ LIN không sử dụng)
DMATXE	Cấu hình	Cho phép dùng giao diện truyền DMA (Driver chưa hỗ trợ)
DMARXE	Cấu hình	Cho phép dùng giao diện nhận DMA (Driver chưa hỗ trợ)

2.3.2 Phần cứng ESCI.

Platform có phần cứng eSCI: Cobra55 (MPC5777C).

- **Mô tả cơ chế hoạt động của phần cứng:** Từ bus clock phần cứng eSCI lấy clock đưa vào bộ tạo baudrate (Baudrate generator) chia làm 2 đường RCLK vào bộ điều khiển nhận và TCLK (1/16 Bus clock) vào bộ điều khiển truyền.

- **Bộ điều khiển truyền**: Tham khảo phần 49.5.6.3 LIN TX Frame generation (MPC5777C Reference Manual, Rev. 2, 06/2014), eSCI cung cấp hai chế độ tạo khung truyền Tx: chế độ điều khiển bằng CPU và chế độ điều khiển DMA. Trong chế độ điều khiển bởi CPU, lần lượt ghi dữ liệu vào thanh ghi eSCI_LTR đồng thời tạo một ngắt để xử lý truyền. Trong chế độ điều khiển DMA, khi dữ liệu mới cần truyền thì phần cứng eSCI tạo một yêu cầu truyền DMA và bộ điều khiển DMA sẽ phát dữ liệu.
- **Bộ điều khiển nhận**: Tương tự như phần truyền dữ liệu, bộ điều khiển nhận cũng có hai chế độ nhận được điều khiển bởi CPU hoặc DMA. Khi nhận dữ liệu bằng CPU một cờ ngắt nhận được set lên và phần mềm có thể đọc từ eSCI_LRR. Trong chế độ nhận dữ liệu điều khiển bởi DMA cũng tương tự như bộn truyền.



Hình 2.17 Cấu tạo phần cứng eSCI trên platform Cobra55.

Tiếp theo xin trình bày đến phần dectect lỗi trong quá trình truyền nhận.

- **Physical Bus Error Detection:**

Sau khi bắt đầu truyền một frame, chân tín hiệu Rx ở mức 0 trong ít nhất 31 chu kì clock lấy mẫu thì cờ ngắt lỗi IFSR2_PBERR được bật lên.

- **Unrequested Activity Detection:**

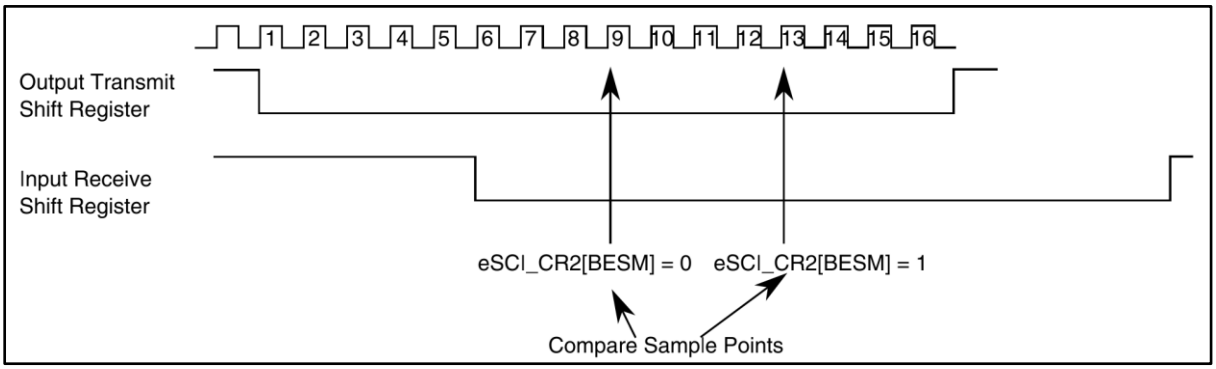
Nếu chân tín hiệu Rx nhận được một byte dữ liệu nhưng được yêu cầu từ Slave, đồng thời cũng không phải xung wakeup hay break thì cờ IFSR1_BERR sẽ được bật lên.

- **Standard Bit Error Detection:** Điều kiện là bit CR2_FBR = 0

Sau mỗi byte truyền thì phần cứng eSCI sẽ so sánh dữ liệu đã truyền và dữ liệu nhận được so sánh với nhau, nếu xảy ra sai lệch thì bit lỗi IFSR1_BERR sẽ được bật.

- **Fast Bit Error Detection:** Điều kiện là bit CR2_FBR = 1

Tương tự như trên nhưng dữ liệu truyền và nhận được so sánh tại các mẫu được lấy trên từng bit. Nếu eSCI_CR2[BESM] = 1 thì phần cứng so sánh tại mẫu thứ 9, ngược lại thì so sánh tại mẫu thứ 13.



- **Slave-Not-Responding-Error Detection:**

Đây là tính năng kiểm tra timeout một frame truyền với kiểu Slave_Response, nếu sau thời gian tối đa Tframe_max mà Master không nhận được dữ liệu từ Slave thì cờ IFSR2_STO sẽ được bật lên.

- **Checksum Error Detection:** Điều kiện LTR_CSE = 1

Nếu giá trị của byte checksum nhận được từ Slave không khớp với giá trị byte checksum mà phần cứng eSCI tự tính thì cờ lỗi IFSR2_CKERR được bật lên.

- **CRC Error Detection:** Trong spec của LIN không định nghĩa CRC vì vậy lỗi này không được implement trong driver.

Khi giá trị của 2 byte CRC nhận được và phần cứng tự tính không khớp nhau thì cờ IFSR2_CRC sẽ bật lên.

- **Overflow Detection:**

Khi eSCI nhận được một byte dữ liệu tiếp theo, nhưng dữ liệu trước đó chưa được đọc từ thanh ghi eSCI_LRR hoặc kênh DMA thì dữ liệu mới sẽ bị mất. Đồng thời, cờ IFSR2_OVFL được bật.

Bảng 2.4 Các thanh ghi trong phần cứng ESCI.

Thanh ghi	Phân loại	Mô tả chức năng
ESCI_BRR	Cấu hình	Cấu hình tốc độ truyền/ nhận (baudrate) dữ liệu.
ESCI_CR1	Cấu hình	Cấu hình các chức năng chính, chế độ chạy của LIN.
ESCI_CR2	Điều khiển	Cấu hình các chức năng, cho phép ngắt và các bit điều khiển hoạt động truyền nhận
ESCI_SDR	Cấu hình	Thanh ghi dữ liệu
ESCI_IFSR1	Trạng thái	Gồm các cờ ngắt truyền/ nhận và các cờ ngắt lỗi
ESCI_IFSR2	Trạng thái	Gồm các cờ ngắt liên quan đến truyền/ nhận và xác định lỗi truyền.
ESCI_LCR1	Cấu hình	Gồm các bit điều khiển hoạt động, cấu hình phần cứng LIN
ESCI_LCR2	Cấu hình	Cho phép hoạt động các ngắt trên thanh ESCI_IFSR2
ESCI_LTR	Cấu hình	Tham số các trường header
ESCI_LRR	Dữ liệu	Dữ liệu nhận
ESCI_LPR	Cấu hình	Cung cấp đa thức CRC cho việc tạo và xử lý khung LIN nâng cao có CRC.
ESCI_CR3	Điều khiển	Thanh ghi này được sử dụng để điều khiển các định dạng khung và tạo ra bit ERR trong

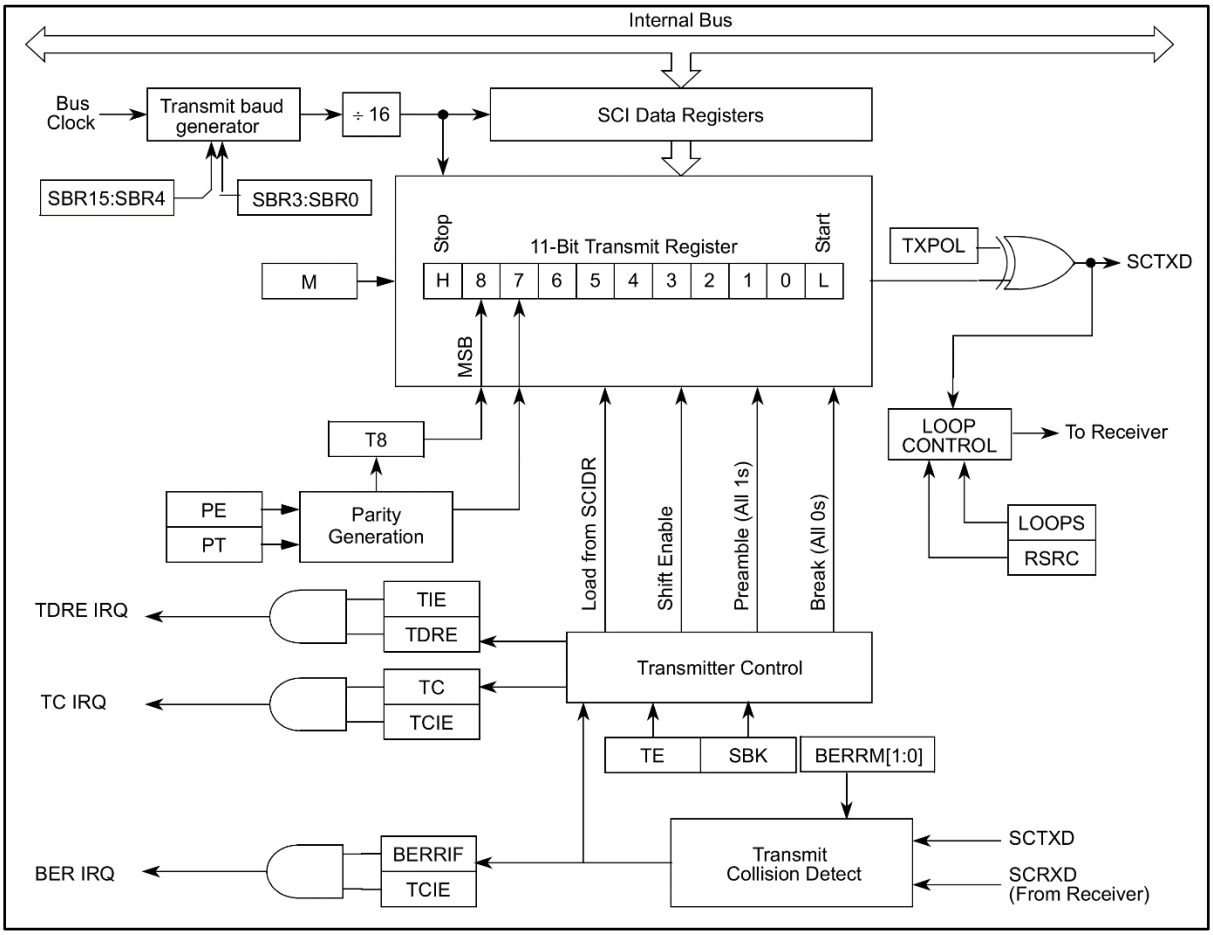
2.3.3 Phần cứng SCI.

Hiện tại, các platform có phần cứng SCI là Carcassonne (MC9S12ZVM) và Hearst (MC9S12ZVC). Phần cứng SCI bao gồm có 11 thanh ghi.

Mô tả cơ chế hoạt động của phần cứng:

- **Khối truyền:**

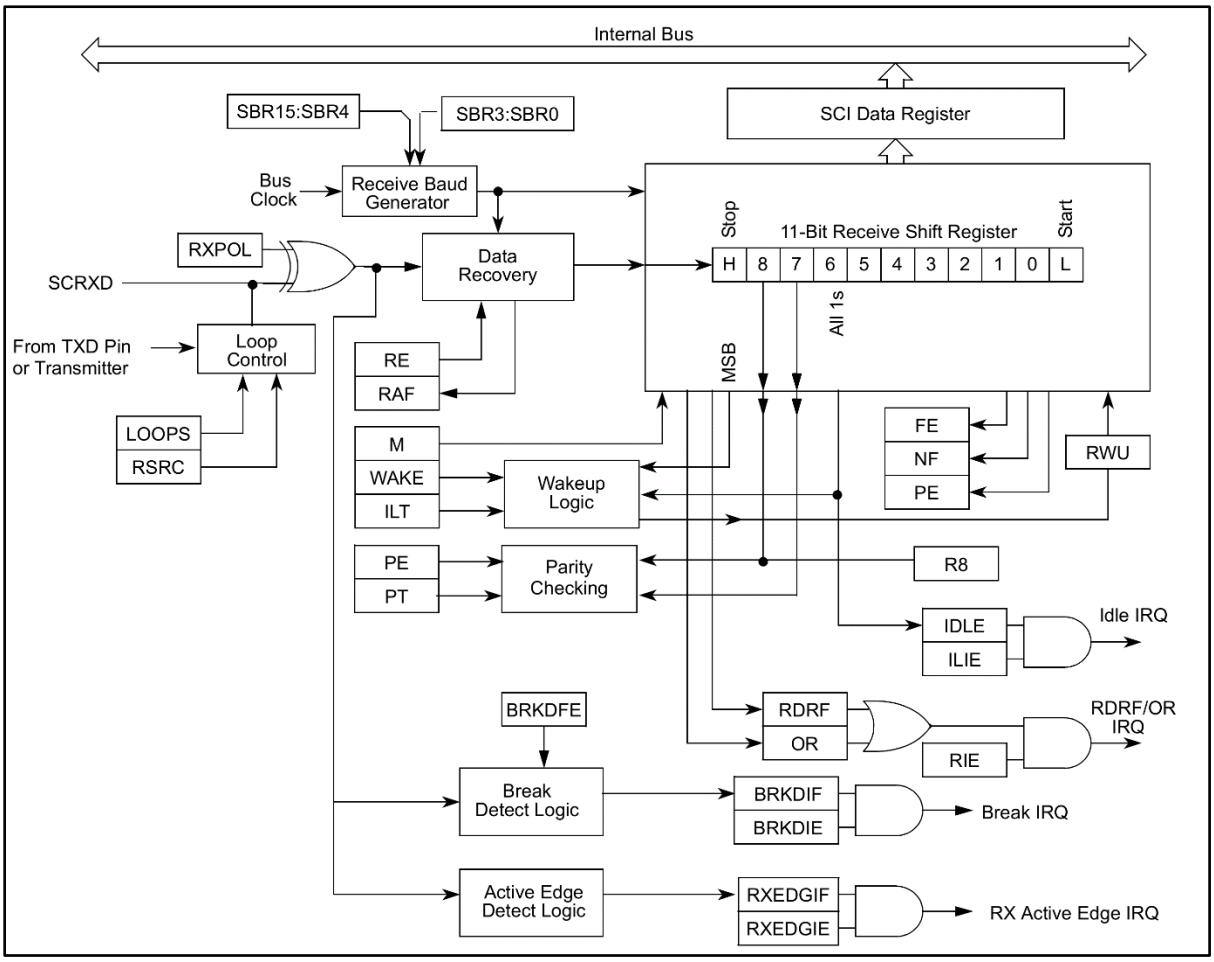
- Clock của module được lấy từ bus clock của vi điều khiển, qua bộ Transmit baud generator tính toán tốc độ truyền của khung theo công thức **SCI baud rate = SCI bus clock / (SCIBR[15:0]).**
- Bộ điều khiển truyền (Transmitter Control): Đây là bộ phận quan trọng nhất khối truyền, chịu trách nhiệm điều khiển quá trình load dữ liệu từ thanh ghi SCI_data vào thanh ghi dịch, các ngắt truyền…
- Bộ lặp (Loop Control): Dùng khi debug, nối thông chân tín hiệu Tx với Rx.
- Bộ phát hiện lỗi truyền (Transmitter Collision Detect): Xảy ra lỗi truyền thì bật cờ ngắt lỗi.



Hình 2.18 Cấu tạo phần cứng bên truyền SCI trên platform Carcassone

- **Khối nhận:**

- Clock module: từ bộ Receive Baud Generator
- Tín hiệu từ chân Rx được đưa đến thanh ghi dịch rồi đến thanh ghi dữ liệu.



Hình 2.19 Cấu tạo phần cứng bên nhận SCI trên platform Carcassone.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0000 SCIBDH[1]	R/W	SBR15	SBR14	SBR13	SBR12	SBR11	SBR10	SBR9	SBR8
0x0001 SCIBDL[1]	R/W	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x0002 SCICR1[1]	R/W	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x0000 SCIASR1[2]	R	RXEDGIF	0	0	0	0	BERRV	BERRIF	BKDIF
	W								
0x0001 SCIACR1[2]	R	RXEDGIE	0	0	0	0	0	BERRIE	BKDIE
	W								
0x0002 SCIACR2[2]	R	IREN	TNP1	TNP0	0	0	BERRM1	BERRM0	BKDFE
	W								
0x0003 SCICR2	R/W	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x0004 SCISR1	R	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
	W								
0x0005 SCISR2	R	AMAP	0	0	TXPOL	RXPOL	BRK13	TXDIR	RAF
	W								
0x0006 SCIDRH	R	R8	T8	0	0	0	Reserved	Reserved	Reserved
	W								
0x0007 SCIDRL	R	R7	R6	R5	R4	R3	R2	R1	R0
	W	T7	T6	T5	T4	T3	T2	T1	T0

1.These registers are accessible if the AMAP bit in the SCISR2 register is set to zero.

2,These registers are accessible if the AMAP bit in the SCISR2 register is set to one.

= Unimplemented or Reserved

Hình 2.20 Thanh ghi trên phần cứng SCI.

Bảng 2.5 Các thanh ghi trong phần cứng SCI.

Thanh ghi	Phân loại	Mô tả chức năng
SCI_BDH, **SCI_BDL**	Cấu hình	Cấu hình tốc độ truyền/ nhận (baudrate) dữ liệu.
SCI_CR1	Cấu hình	Cấu hình các chức năng chính, chế độ chạy của LIN.
SCI_ASR1	Trạng thái	Gồm các cờ ngắt bit lỗi, nhận dạng xung break, nhận dạng sườn xung lên/ xuống.
SCI_ACR1, **SCI_ACR2**	Cấu hình	Gồm các bit enable cờ ngắt của thanh ghi SCIASR1 và quy định chế độ xác định bit lỗi…
SCI_CR2	Điều khiển	Gồm các bit enable các cờ ngắt truyền/ nhận.
SCI_SR1	Trạng thái	Gồm các cờ ngắt liên quan đến truyền/ nhận và xác định lỗi truyền.
SCI_SR2	Trạng thái	Gồm các bit thiết lập chế độ, dạng dữ liệu truyền; trong đó có bit AMAP khá đặc biệt dùng để chuyển đổi quyền truy nhập các thanh ghi có cùng địa chỉ.
SCI_DRH, SCI_DRL	Dữ liệu	Lưu trữ dữ liệu dữ liệu truyền/ nhận.

Bảng 2.6 Mô tả sơ bộ chức năng các thanh ghi của phần cứng SCI.

Lưu ý: Do hạn chế phần cứng, driver xây dựng trên nền tảng SCI không có chức năng sau:

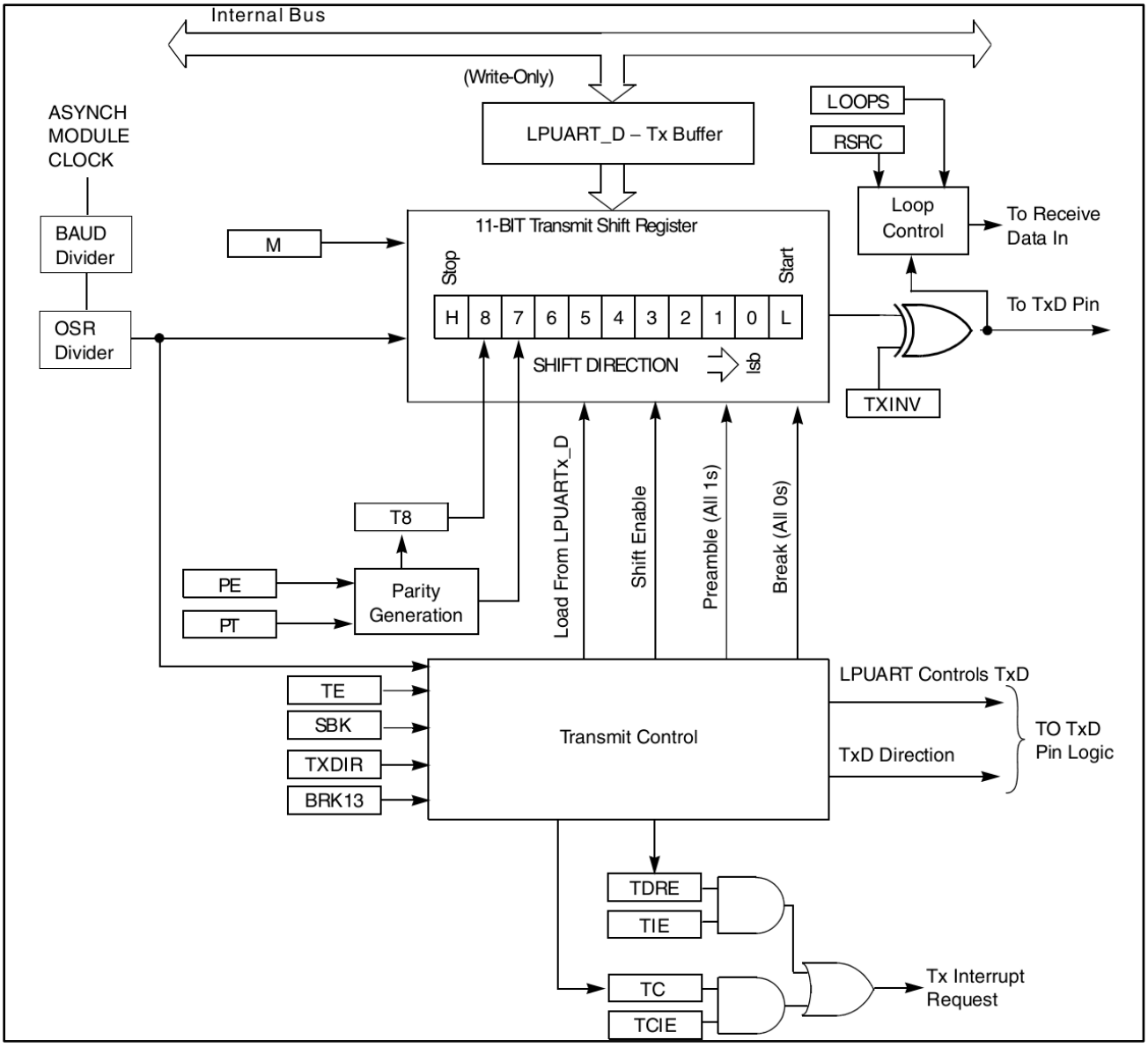
Disable Frame TimeOut: SCI không có thanh ghi chuyên dụng để đo thời gian truyền/ nhận một khung truyền. Do vậy, hàm Lin_GetStatus() không trả về trạng thái LIN_RX_NO_RESPONSE.

2.3.4 Phần cứng LPUART.

Platform có phần cứng LPUART: S32K14X.

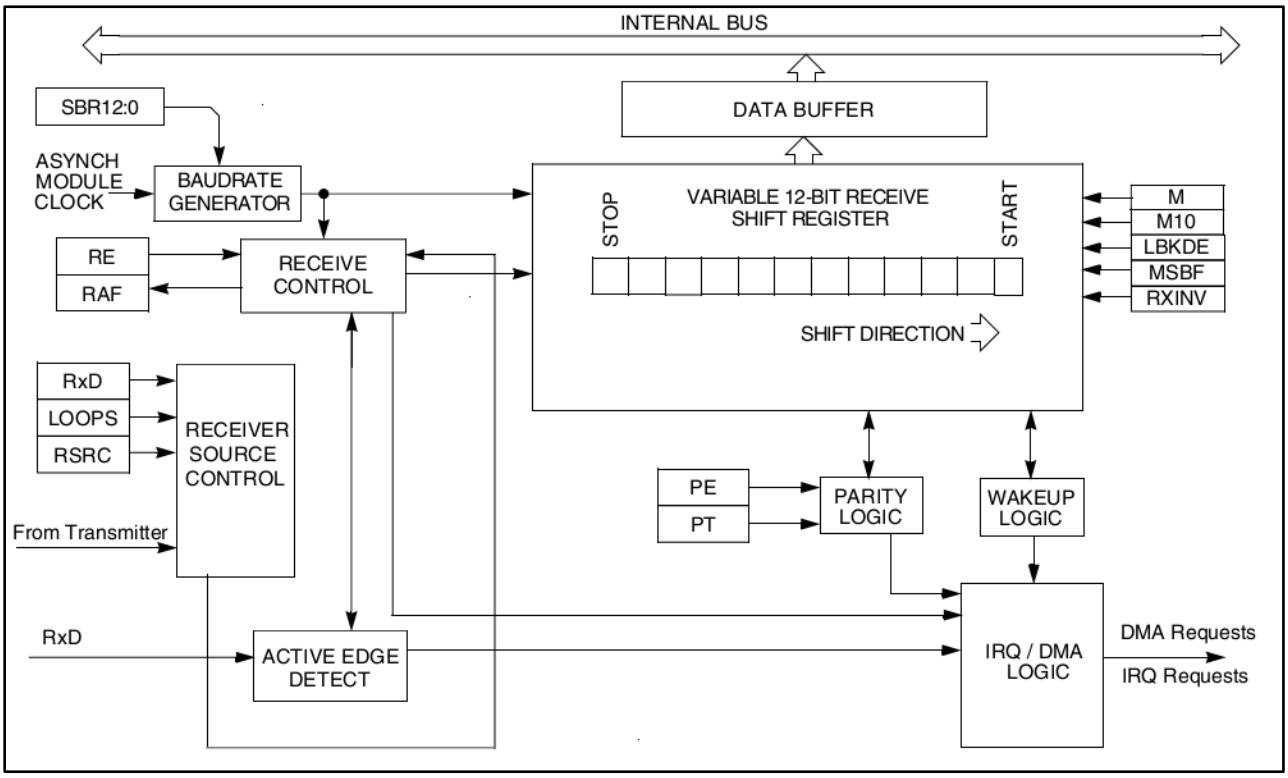
- **Mô tả cơ chế hoạt động của phần cứng:** Tương tự như phần cứng SCI.

- Khối phát:



Hình 2.21 Cấu tạo phần cứng bên phát Tx - LPUART trên platform S32K14X.

- Bộ nhận:



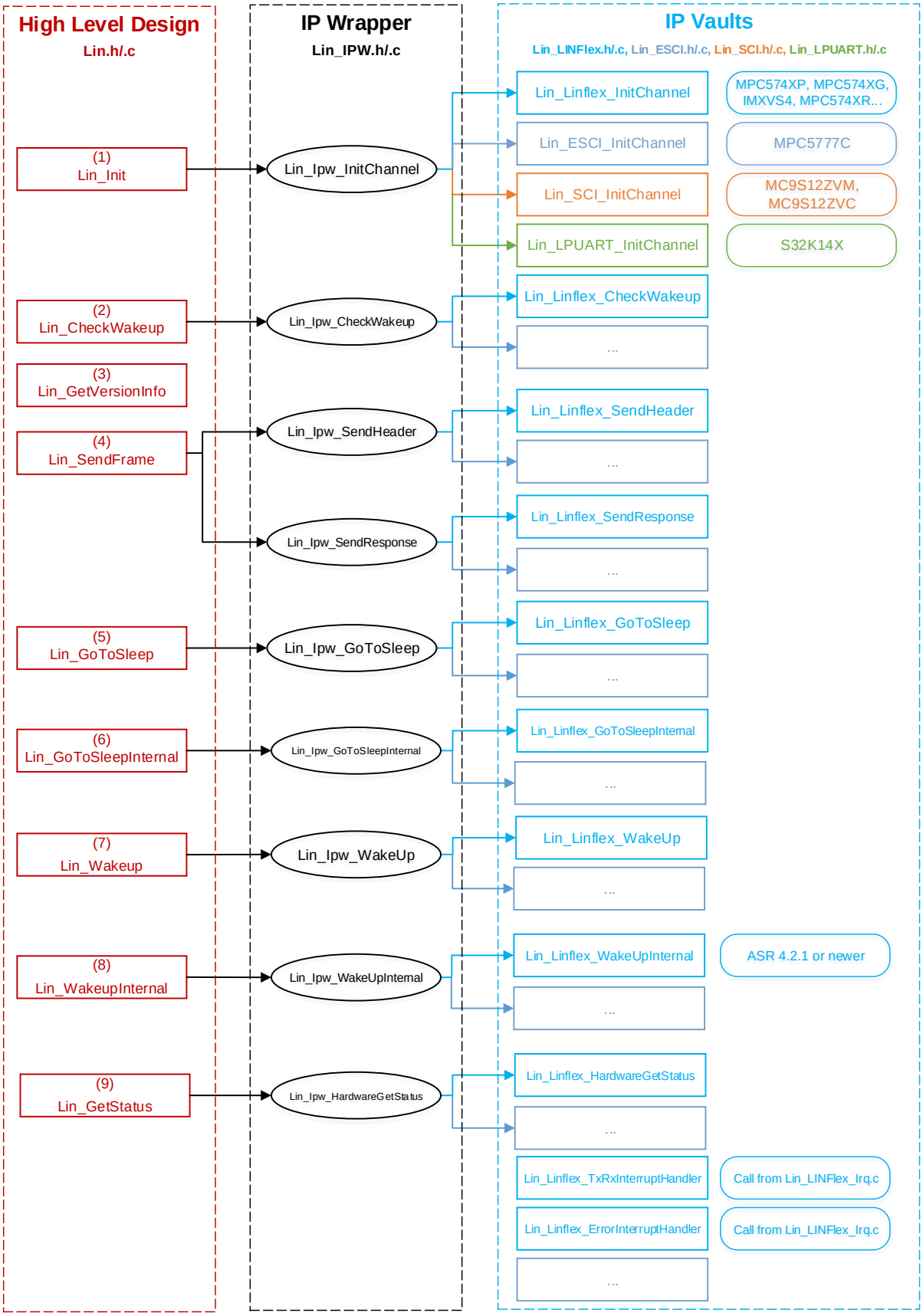
Hình 2.22 Cấu tạo phần cứng bên thu Rx - LPUART trên platform S32K14X.

Bảng 2.7 Các thanh ghi trong phần cứng LPUART.

Thanh ghi	Phân loại	Mô tả chức năng
LPUART_VERID	Trạng thái	Thông tin phiên bản phần cứng
LPUART_PARAM	Trạng thái	Số lượng word trong FIFO
LPUART_ GLOBAL	Trạng thái	Khởi tạo lại giá trị toàn bộ các thanh ghi
LPUART_ PINCFG	Cấu hình	Chọn kiểu trigger
LPUART_BAUD	Cấu hình	Cấu hình tốc độ truyền nhận, các định dạng byte trong khung truyền
LPUART_STAT	Trạng thái	Gồm các cờ ngắt liên quan đến truyền/ nhận và xác định lỗi truyền.
LPUART_CTRL	Cấu hình	Điều khiển các tính năng tùy chọn của hệ thống LIN
LPUART_DATA	Dữ liệu	Lưu trữ dữ liệu dữ liệu truyền/ nhận.
LPUART_MATCH		Chế độ LIN không sử dụng
LPUART_MODIR		Chế độ LIN không sử dụng

LPUART_FIFO		Chế độ LIN không sử dụng
LPUART_WATER		Chế độ LIN không sử dụng

2.4 Cách thức thực hiện hàm.



Hình 2.23 Cấu trúc các hàm API trong driver.

Theo tài liệu Autosar, driver được xây dựng có 3 lớp:

- **Lớp HLD** (High Level Design): Khai báo và thực hiện các hàm API như Autosar đã định nghĩa theo từng requirement, cụ thể LIN có 9 API. Xem lại phần 2.2.4.
- **Lớp IP Wrapper**: Kết nối giữa 2 lớp HLD và IP Vault, có nhiệm vụ rename các hàm ở lớp IP Vault để phù hợp với cách gọi tên từ lớp trên cùng. Lớp này không có code thực thi bất kì tính năng nào.
- **Lớp IP Vault**: Khai báo và thực hiện các hàm tác động mức thanh ghi. Lưu ý, không có các requirement cụ thể cho các hàm ở lớp IP Vault này. Vì vậy, các developer khi xây dựng một driver mới căn cứ vào cấu tạo phần cứng để viết các function cho phù hợp các API lớp HLD.

Chi tiết về cấu trúc các file trong driver tham khảo phần **5.1.2 Header file structure**, tài liệu **Specification of LIN Driver V1.5.0 R4.0 Rev 3**.

2.4.1 IP Vault LINFLEXD.

2.4.1.1 Hàm Lin_Linflex_InitChannel.

Mô tả: Hàm này khởi tạo tất cả các thanh ghi phần cứng cần thiết để bắt đầu chức năng LIN trên kênh đã chọn.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void

Bước 1. Trạng thái khởi tạo (Init) bằng cách set bit LINCR1_INIT.

Bước 2. Thoát khỏi chế độ Sleep bằng cách xóa bit LINCR1_SLEEP.

Bước 3. Cho phép phần cứng LinFlexD chạy theo chế độ hoạt động giao thức LIN, xóa bit UARTCR_UART.

Bước 4. Cấu hình chip chạy ở chế độ master node, thiết lập bit: LINCR1_MME.

Bước 5. Thiết lập chip tự tính toán giá trị byte checksum bằng phần cứng, set bit LINCR1_CCD.

Bước 6. Cho phép gửi byte checksum sau khi dữ liệu đã được gửi xong, set bit LINCR1_CFD.

Bước 7. Vô hiệu hóa tính năng LIN Auto Synchronization, xóa bit LINCR1_LASE.

Bước 8. Disable chế độ Loop Back (nối tắt chân Tx với Rx), xóa bit LINCR1_LBKM.

Bước 9. Cho phép không khóa thanh ghi buffer, dữ liệu mới có thể ghi đè lên dữ liệu cũ, xóa bit LINCR1_LBKM.

Bước 10. Chỉ cho phép xóa bit Sleep bởi phần mềm, xóa bit LINCR1_AWUM.

Bước 11. Khi xảy ra các lỗi truyền hoặc nhận, không reset lại trường status LINSR_LINS, xóa bit LINCR2_IOBE.

Bước 12. Cấu hình break delimeter có độ dài 2 bit (theo LIN Spec 2.1), set bit LINCR2_TBDE.

Bước 13. Cấu hình độ dài của trường Break, truyền giá trị từ con trỏ config vào trường LINCR1_MBL.

Bước 14. Cấu hình tốc độ của khung truyền bằng cách ghi giá trị vào 2 thanh ghi Baudrate LINFBRR, LINIBRR.

Bước 15. Cấu hình chế độ timeout cho khung truyền bằng cách set bit LINTCSR_MODE.

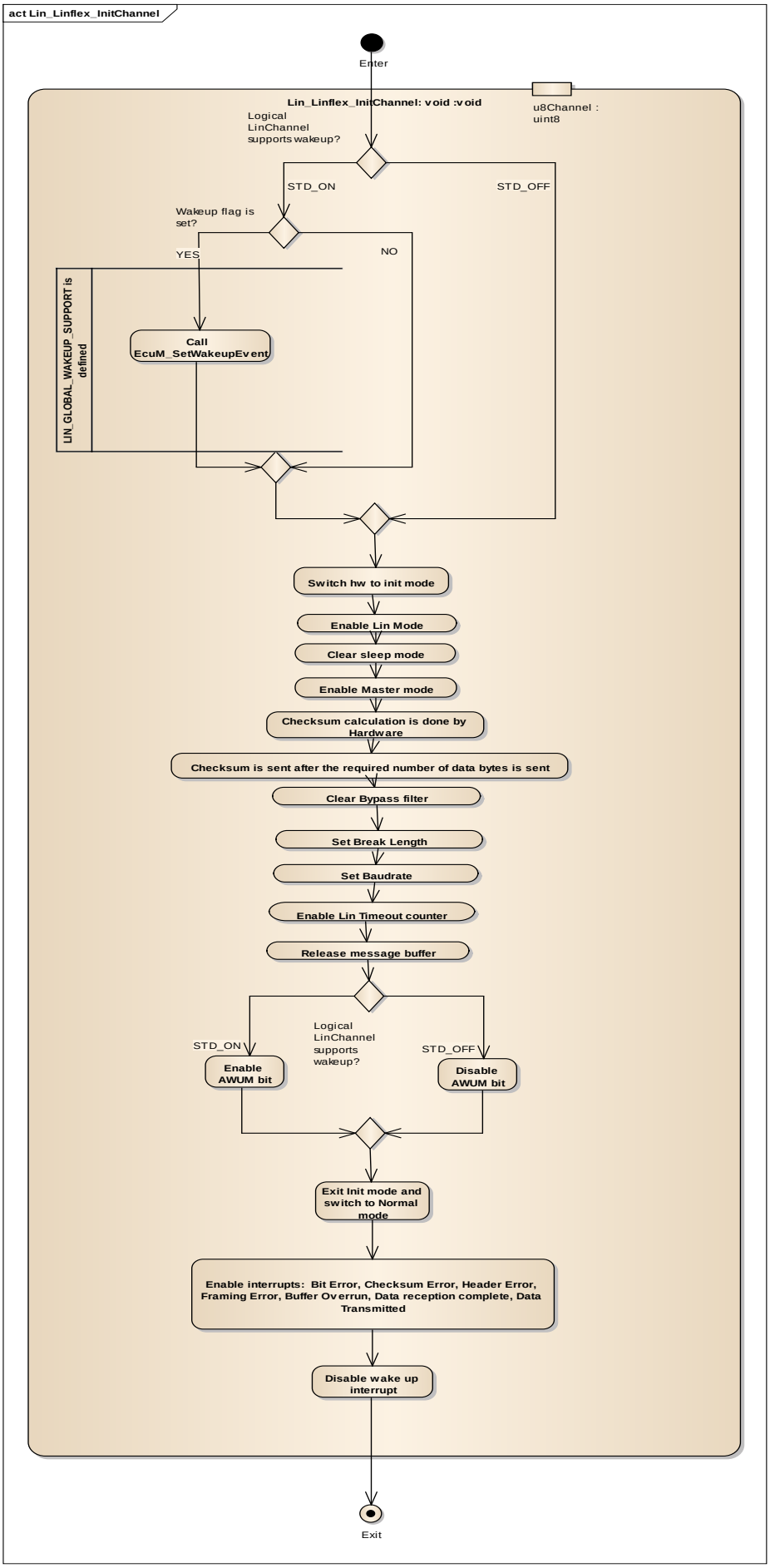
Bước 16. Khi xảy ra timeout trong quá trình frame, không cho phép reset trạng thái trường LINS (trong thanh ghi LINSR), xóa bit LINTCSR_IOT.

Bước 17. Khi xảy ra timeout trong quá trình frame, cho phép cờ LINTCSR_OCF được set lên.

Bước 18. Thiết lập thanh ghi dữ liệu đệm (buffer) sẵn sàng để đọc bằng phần mềm, set bit LINSR_RMB.

Bước 19. Thoát khỏi chế độ khởi tạo, xóa bit LINCR1_INIT.

Bước 20. Disable/ Enable các ngắt trên thanh ghi LINIER_HEIE.


act Lin_Linflex_InitChannel
Enter
Lin_Linflex_InitChannel: void :void
u8Channel : uint8
Logical LinChannel supports wakeup?
STD_ON
STD_OFF
Wakeup flag is set?
YES
NO
LIN_GLOBAL_WAKEUP_SUPPORT is defined
Call EcuM_SetWakeupEvent
Switch hw to init mode
Enable Lin Mode
Clear sleep mode
Enable Master mode
Checksum calculation is done by Hardware
Checksum is sent after the required number of data bytes is sent
Clear Bypass filter
Set Break Length
Set Baudrate
Enable Lin Timeout counter
Release message buffer
Logical LinChannel supports wakeup?
STD_ON
STD_OFF
Enable AWUM bit
Disable AWUM bit
Exit Init mode and switch to Normal mode
Enable interrupts: Bit Error, Checksum Error, Header Error, Framing Error, Buffer Overrun, Data reception complete, Data Transmitted
Disable wake up interrupt
Exit

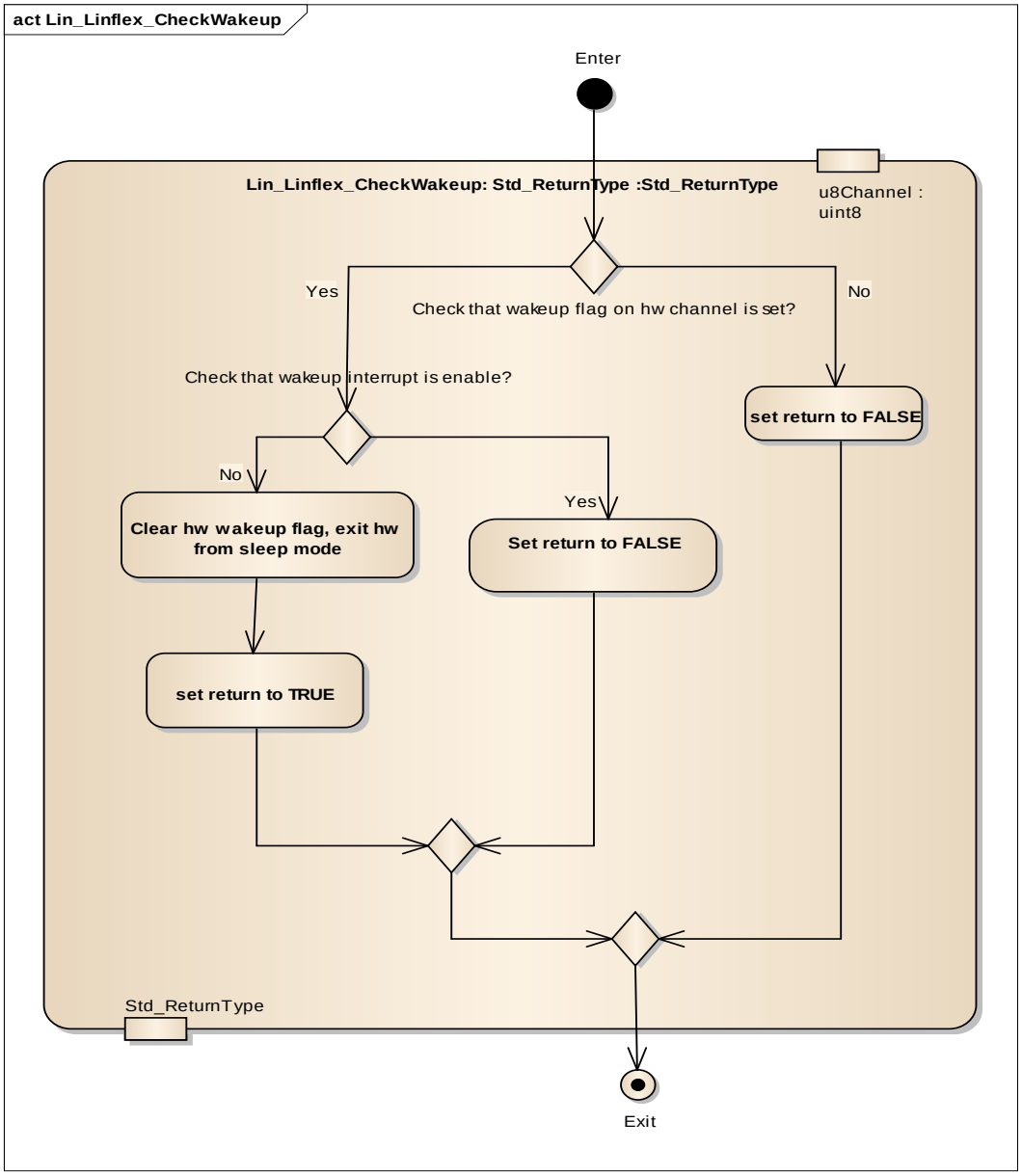
2.4.1.2 Hàm Lin_Linflex_CheckWakeup.

Mô tả: Dùng kiểm tra điều kiện wakeup khi bit enable wakeup bị vô hiệu hóa.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** Std_ReturnType

Bước 1. Kiểm tra điều kiện nếu cờ wakeup LINSR_WUF được set và bit cho phép ngắt LINIER_WUIE bị disable thì xóa cờ wakeup bằng cách ghi 1 vào bit LINSR_WUF; đồng thời xóa bit sleep LINCR1_SLEEP. Return lại status TRUE.

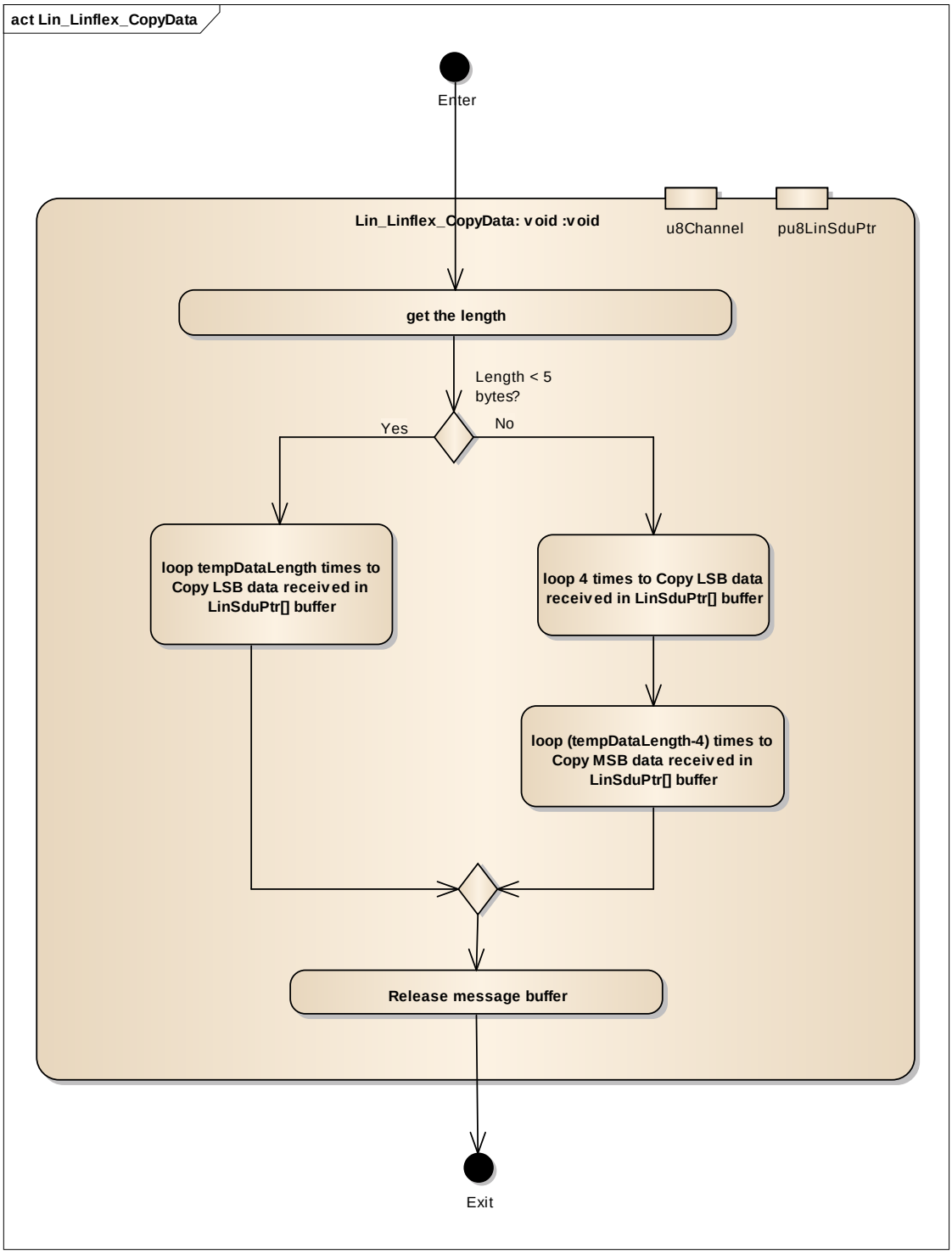
Bước 2. Các trường hợp còn lại thì return FALSE.



2.4.1.3 Hàm Lin_Linflex_CopyData.

Mô tả: Copy dữ liệu nhận được từ buffer trong trường hợp slave response.

- **Input:** u8Channel (địa chỉ kênh vật lý), con trỏ pu8LinSduPtr trỏ đến buffer hoặc bộ nhớ phần cứng SCI nơi có SDU được lưu trữ.
- **Output:** void



2.4.1.4 Hàm Lin_Linflex_SendHeader.

Mô tả: Khởi tạo truyền phần tiêu đề (header) của khung truyền LIN trên Bus bằng cách sử dụng thông tin cấu hình được lưu trữ trên con trỏ pPduInfoPtr. Nếu frame truyền là loại MASTER_RESPONSE thì phần tiêu đề này sẽ không truyền lên Bus mà toàn bộ frame (bao gồm cả tiêu đề) sẽ được truyền trong hàm Lin_LINFlex_SendReponse.

- **Input:** u8Channel (địa chỉ kênh vật lý), pPduInfoPtr (Con trỏ đến PDU chứa PID, mô hình Checksum, loại phản hồi Master/ Slave response, con trỏ dữ liệu Dl và SDU)
- **Output:** Trạng thái Std_ReturnType truyền phần tiêu đề thành công hay không?

Bước 1. Dừng toàn bộ các hoạt động truyền/ nhận đang diễn ra (hủy bỏ các frame truyền trước đó), bằng cách set bit LINCR2_ABRQ.

Bước 2. Reset các trạng thái lỗi: Lin_au8LinChFrameErrorStatus.

Bước 3. Update trạng thái hoạt động của frame truyền: Lin_au8LinChFrameStatus.

Bước 4. Xóa bit LINSR_RMB để chuẩn bị đọc dữ liệu từ buffer nhận.

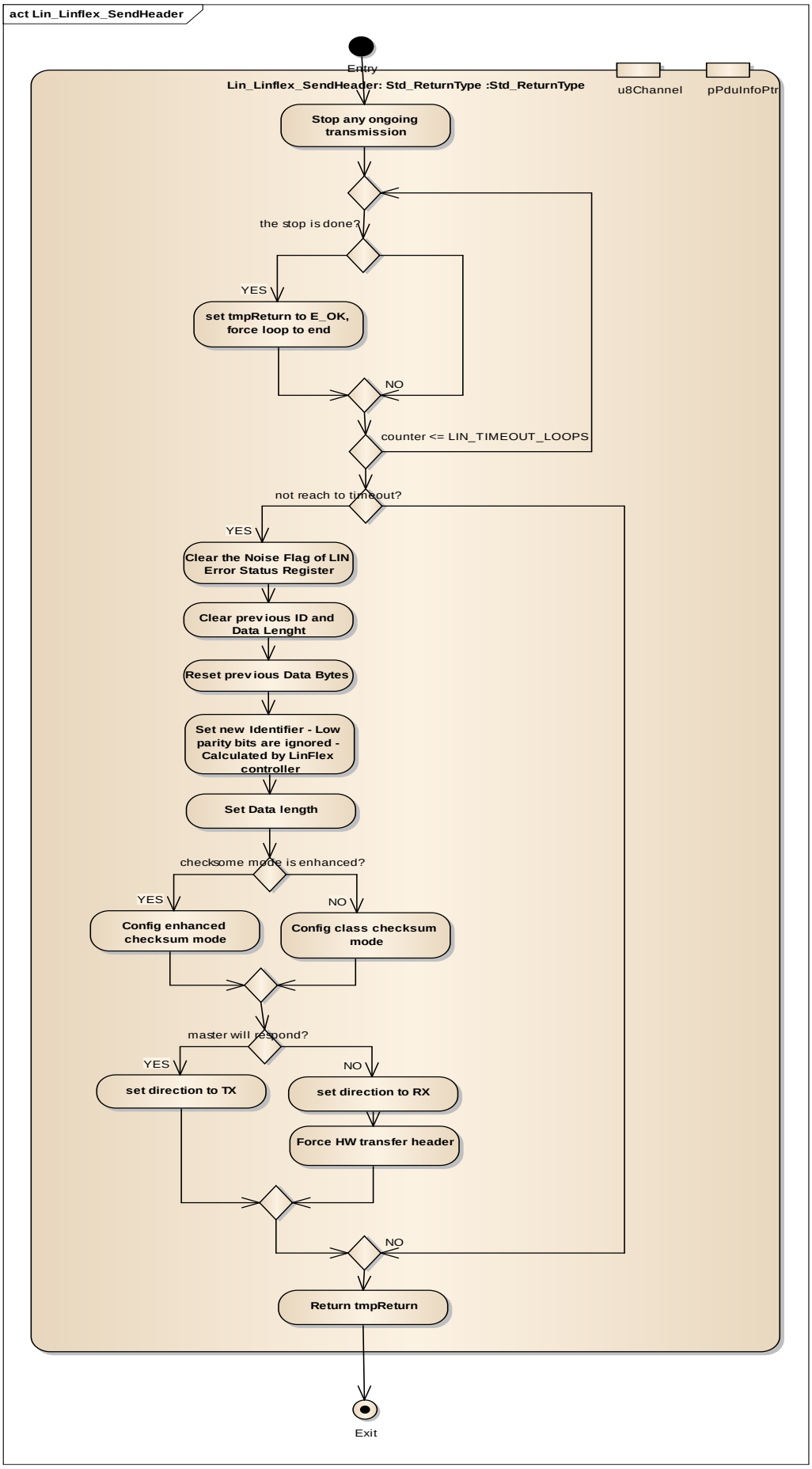
Bước 5. Xóa cờ ngắt lỗi Noise LINESR_NF.

Bước 6. Xóa giá trị các trường: **ID, độ dài dữ liệu DFL** trên thanh ghi BIDR.

Bước 7. Xóa giá trị các byte dữ liệu trên hai thanh ghi: BDRL, BDRM.

Bước 8. Thiết lập các trường: ID, DFL, kiểu checksum và loại phản hồi dữ liệu trên thanh ghi BIDR.

Bước 9. Nếu frame truyền là loại SLAVE_RESPONSE: Tạo yêu cầu ngắt truyền tiêu đề, set bit LINCR2_HTRQ.


act Lin_Linflex_SendHeader
Entry
Lin_Linflex_SendHeader: Std_ReturnType :Std_ReturnType
u8Channel
pPduInfoPtr
Stop any ongoing transmission
the stop is done?
YES
set tmpReturn to E_OK, force loop to end
NO
counter <= LIN_TIMEOUT_LOOPS
not reach to timeout?
YES
Clear the Noise Flag of LIN Error Status Register
Clear previous ID and Data Lenght
Reset previous Data Bytes
Set new Identifier - Low parity bits are ignored - Calculated by LinFlex controller
Set Data length
checksome mode is enhanced?
YES
NO
Config enhanced checksum mode
Config class checksum mode
master will respond?
YES
NO
set direction to TX
set direction to RX
Force HW transfer header
NO
Return tmpReturn
Exit

2.4.1.5 Hàm Lin_Linflex_SendResponse.

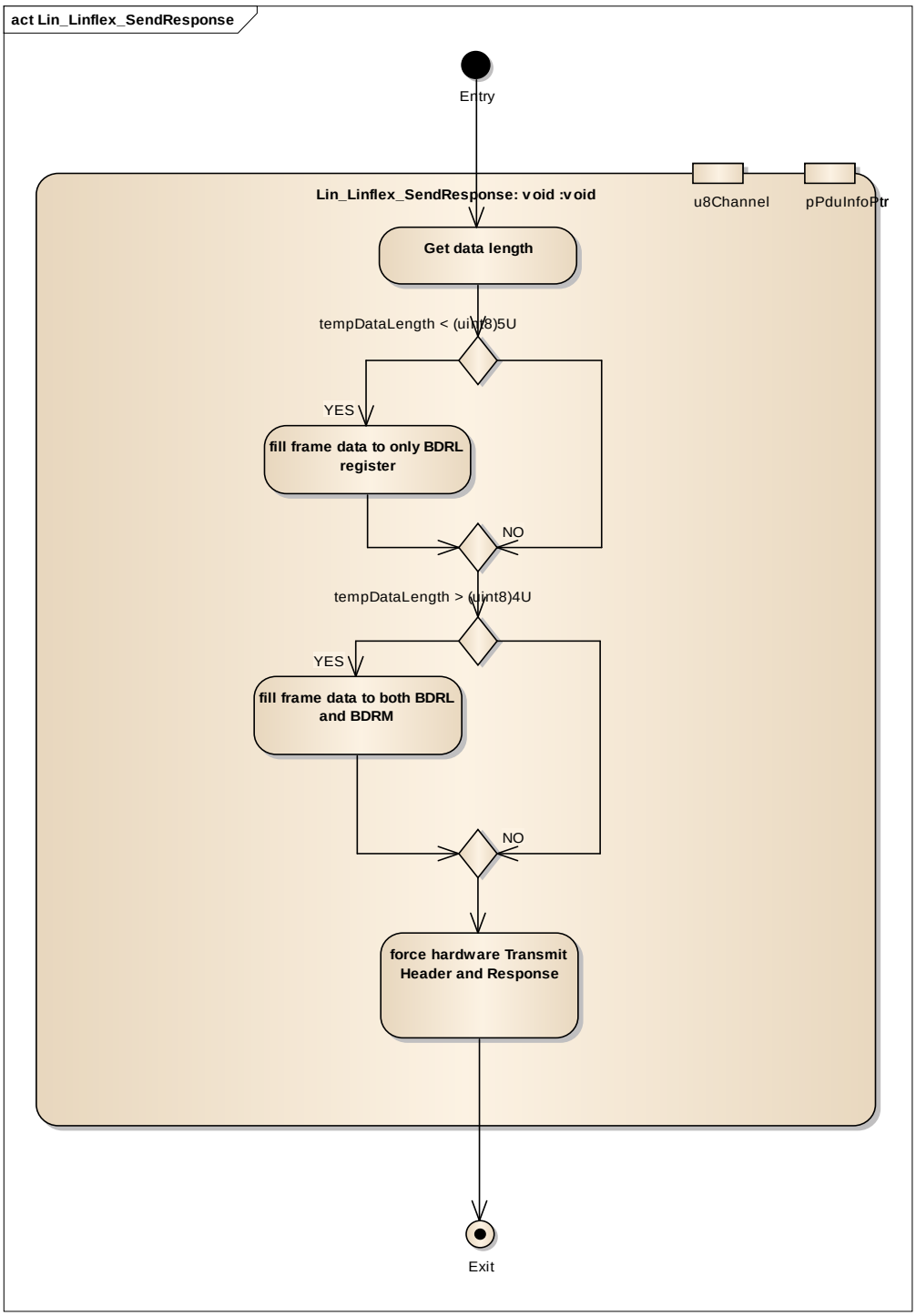
Mô tả: Sử dụng khi frame truyền là loại MASTER_RESPONSE, khởi tạo truyền hai phần header và phần dữ liệu (response) của khung truyền LIN trên kênh bằng cách sử dụng thông tin được lưu trữ trên con trỏ pPduInfoPtr.

- **Input:** u8Channel (địa chỉ kênh vật lý), pPduInfoPtr (Con trỏ đến PDU chứa PID, mô hình Checksum, loại phản hồi Master/ Slave response, con trỏ dữ liệu Dl và SDU)
- **Output:** Trạng thái Std_ReturnType truyền phần tiêu đề thành công hay không?

Bước 1. Đọc số byte dữ liệu phải truyền đến BUS bằng cách đọc giá trị trường BIDR_DFL.

Bước 2. Ghi giá trị các byte dữ liệu vào 2 thanh ghi: BDRL (4 byte đầu), BDRM (4 byte cuối).

Bước 3. Tạo yêu cầu ngắt truyền, set bit LINCR2_HTRQ.



2.4.1.6 Hàm Lin_Linflex_GoToSleep.

Mô tả: Hàm này dừng bất kỳ quá trình truyền nhận nào đang diễn ra và bắt đầu truyền lệnh ngủ với khung truyền: ID = 0x3C và dữ liệu 8 byte = {0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** Trạng thái Std_ReturnType return khung truyền có thành công hay không?

Bước 1. Dừng toàn bộ các hoạt động truyền/ nhận đang diễn ra (hủy bỏ các frame truyền trước đó), set bit LINCR2_ABRQ.

Bước 2. Xóa bit LINSR_RMB để chuẩn bị đọc dữ liệu từ buffer nhận.

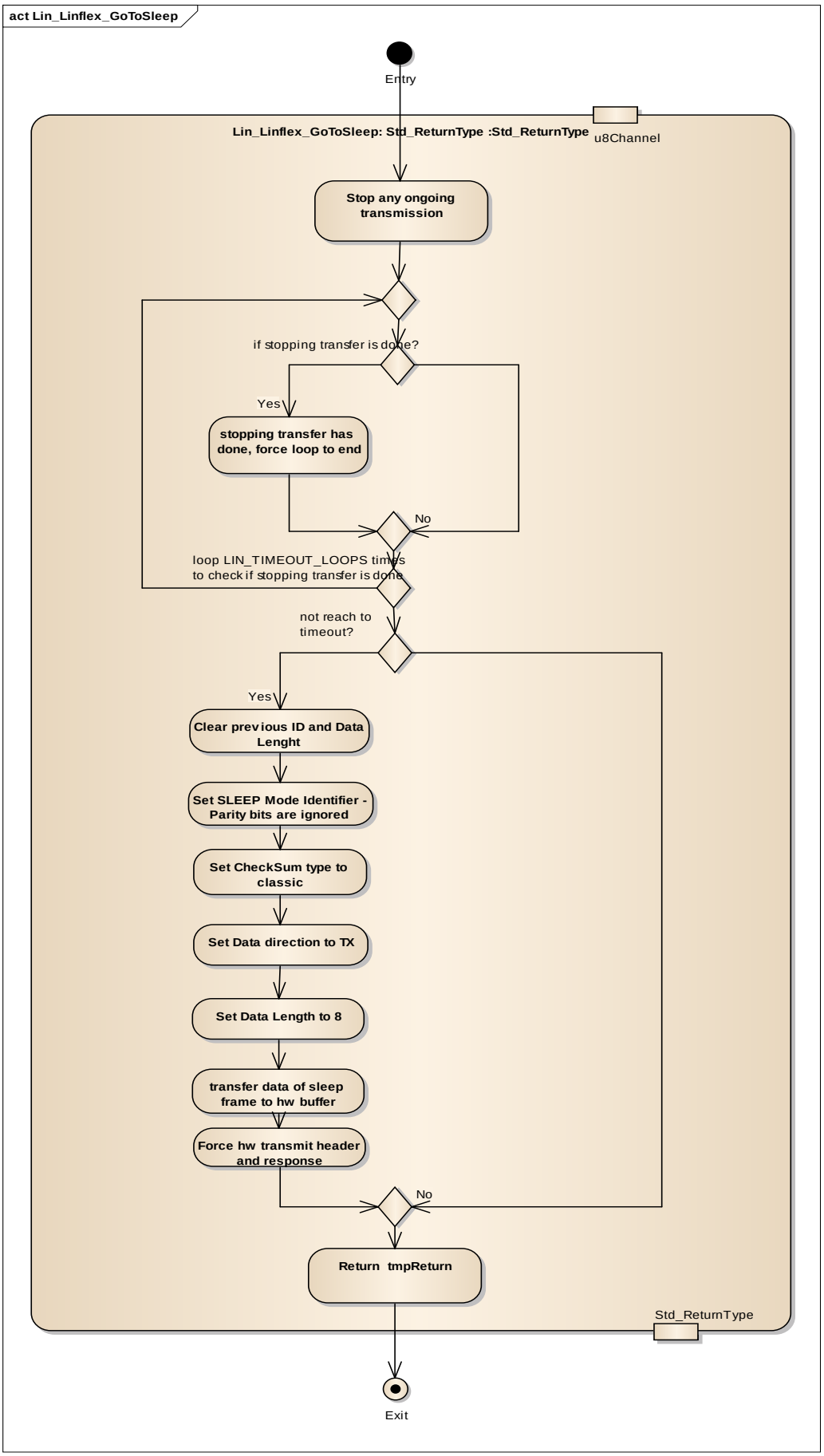
Bước 3. Xóa giá trị cũ của các trường: **ID, độ dài dữ liệu DFL** trên thanh ghi BIDR.

Bước 4. Thiết lập các trường: ID, DFL, kiểu checksum và loại phản hồi dữ liệu trên thanh ghi BIDR.

Bước 5. Xóa giá trị của các byte dữ liệu trên hai thanh ghi: BDRL, BDRM.

Bước 6. Ghi 8 byte dữ liệu: (0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF) vào 2 thanh ghi: BDRL, BDRM.

Bước 7. Tạo yêu cầu ngắt truyền, set bit LINCR2_HTRQ.


act Lin_Linflex_GoToSleep
Entry
Lin_Linflex_GoToSleep: Std_ReturnType :Std_ReturnType
u8Channel
Stop any ongoing transmission
if stopping transfer is done?
Yes
stopping transfer has done, force loop to end
No
loop LIN_TIMEOUT_LOOPS times to check if stopping transfer is done
not reach to timeout?
Yes
Clear previous ID and Data Lenght
Set SLEEP Mode Identifier - Parity bits are ignored
Set CheckSum type to classic
Set Data direction to TX
Set Data Length to 8
transfer data of sleep frame to hw buffer
Force hw transmit header and response
No
Return tmpReturn
Std_ReturnType
Exit

2.4.1.7 Hàm Lin_Linflex_GoToSleepInternal.

Mô tả: Hàm này dừng bất kỳ quá trình truyền nào đang diễn ra và đặt kênh vào chế độ ngủ (sau đó phần cứng LIN đi vào chế độ hoạt động giảm năng lượng).

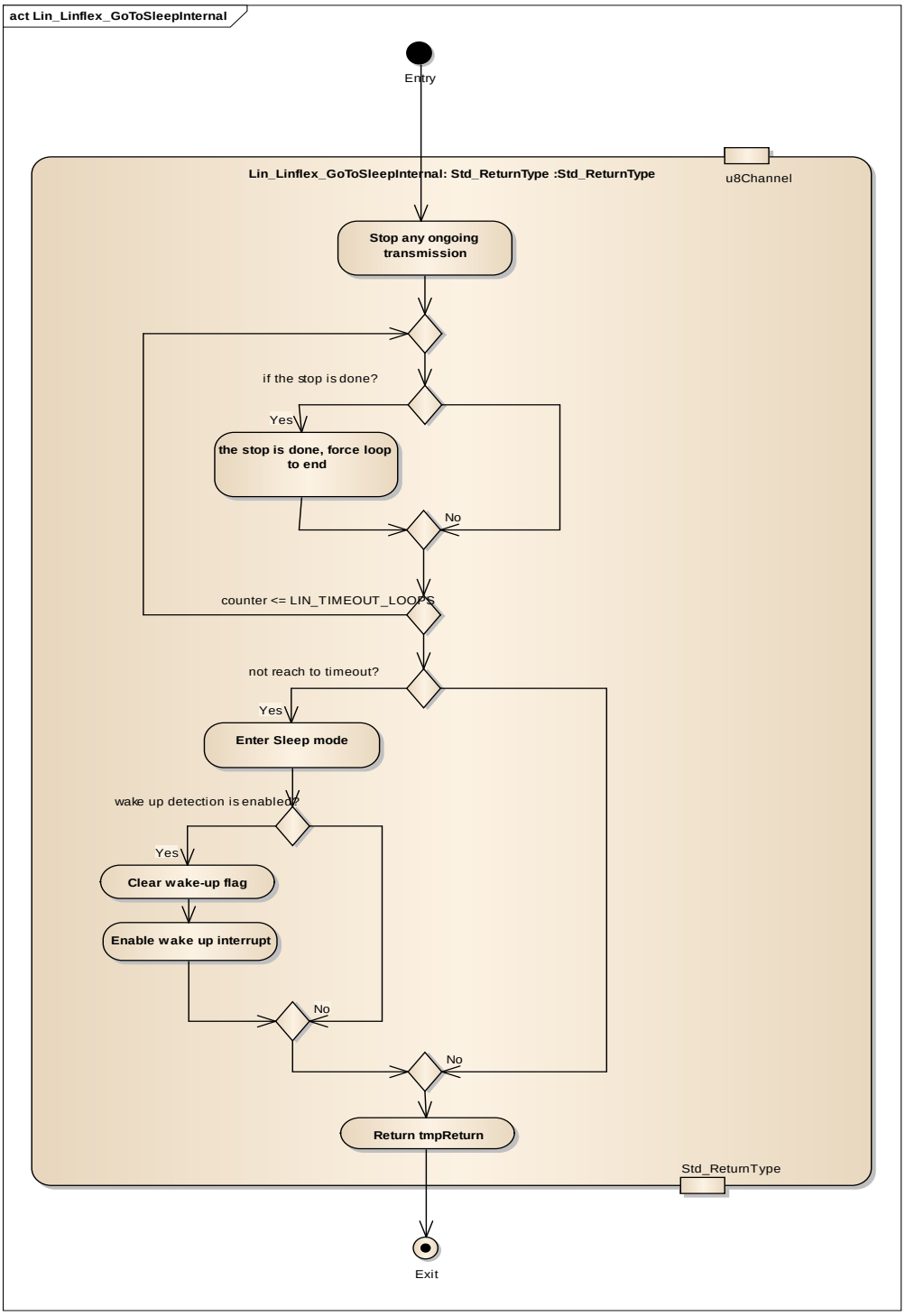
- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** Trạng thái Std_ReturnType return thực hiện hàm có thành công hay không?

Bước 1. Dừng toàn bộ các hoạt động truyền/ nhận đang diễn ra (hủy bỏ các frame truyền trước đó), set bit LINCR2_ABRQ.

Bước 2. Xóa bit LINSR_RMB để chuẩn bị đọc dữ liệu từ buffer nhận.

Bước 3. Thiết lập kênh vào chế độ ngủ bằng cách set bit LINCR1_SLEEP.

Bước 4. Kích hoạt tính năng phát hiện xung wakeup bằng cách xóa cờ wakeup LINSR_WUF và set bit LINIER_WUIE.



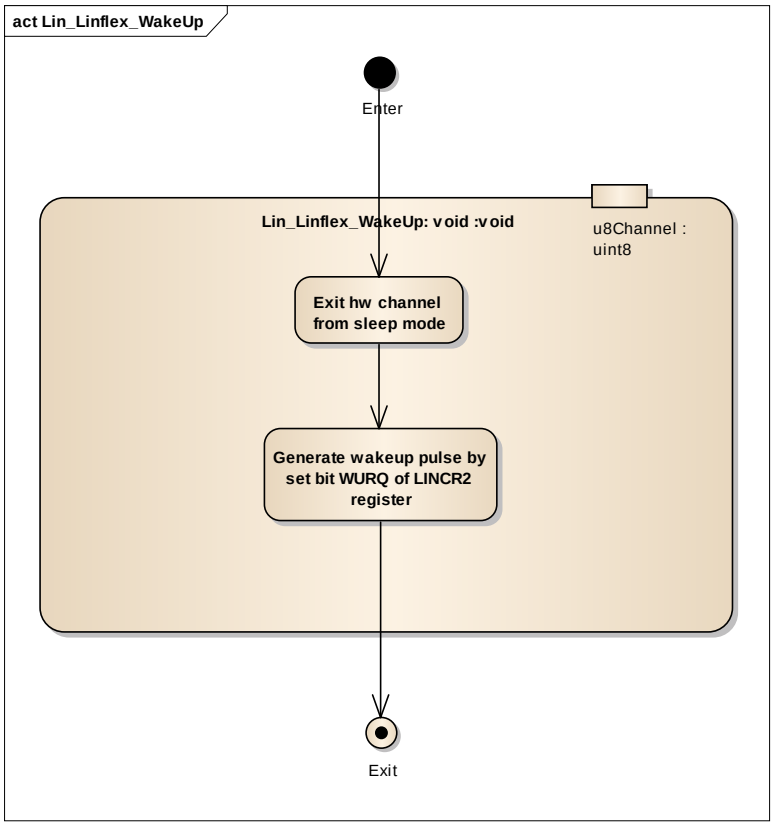
2.4.1.8 Hàm Lin_Linflex_WakeUp.

Mô tả: Hàm này sẽ gửi một tín hiệu wakeup đến đường bus LIN và đặt kênh LIN ở trạng thái hoạt động LIN_CH_OPERATIONAL.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void

Bước 1. Thoát khỏi chế độ ngủ, xóa bit LINCR1_SLEEP.

Bước 2. Gửi một yêu cầu wakep (phần cứng tự động gửi một xung thấp có độ rộng từ 150us – 250ms) lên đường bus LIN bằng cách set bit LINCR2_WURQ.

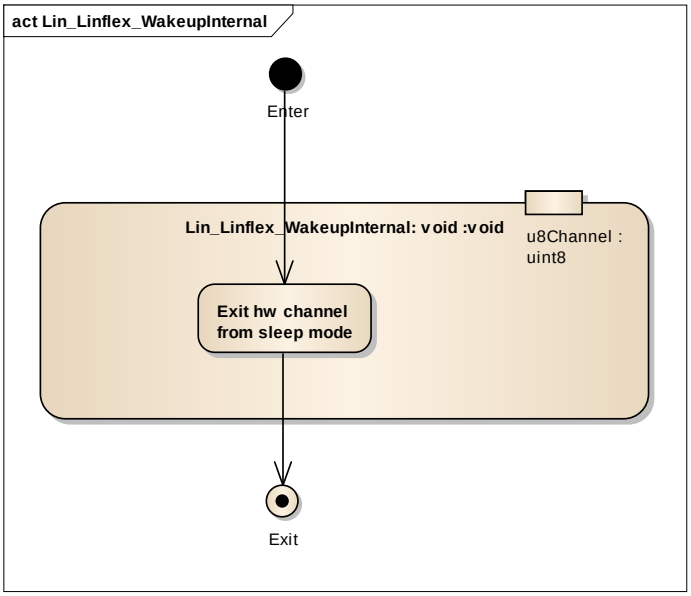


2.4.1.9 Hàm Lin_Linflex_WakeupInternal.

Mô tả: Hàm này sẽ đặt kênh LIN ở trạng thái hoạt động LIN_CH_OPERATIONAL, nhưng không gửi tín hiệu wakeup đến bus LIN.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void

Bước 1. Thoát khỏi chế độ ngủ, xóa cờ LINCR1_SLEEP.



2.4.1.10 Hàm Lin_Linflex_HardwareGetStatus.

Mô tả: Hàm này trả về trạng thái của quá trình truyền, nhận khung truyền hoặc chế độ hoạt động hiện tại. Nếu quá trình nhận phần phản ứng Slave đã thành công thì hàm này cung cấp một con trỏ tới bộ đệm, nơi dữ liệu được lưu trữ.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** Con trỏ pu8LinSduPtr trỏ tới bộ đệm hoặc bộ nhớ phần cứng được ánh xạ nơi SDU hiện tại được lưu trữ

Các bước thực hiện xem trong UML design hoặc tham chiếu driver.

2.4.1.11 Hàm Lin_Linflex_TxRxInterruptHandler.

Mô tả: Hàm này sẽ xử lý tất cả các ngắt RX và TX trên kênh định tuyến.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void

Bước 1. Đọc giá trị của thanh ghi trạng thái LINSR. Các bước tiếp theo có cấu trúc rẽ nhánh if-else.

Bước 2. Nếu cờ wakeup LINSR_WUF được set và trạng thái của module là Sleep thì chuyển trạng thái của module sang LIN_CH_OPERATIONAL.

Bước 3. Nếu cờ ngắt LINSR_DRF (Data Reception Completed) được set

Bước 4. Nếu cờ ngắt LINSR_DTF (Data Transmission Completed) được set:

- Xóa LINSR_DTF.
- Nếu đang trong quá trình truyền lệnh ngủ thì thực hiện xóa cờ wakeup LINSR_WUF, bit enable LINIER_WUIE và bật cờ ngủ LINCR1_SLEEP. Đồng thời, chuyển các biến trạng thái của kênh truyền sang state sleep.
- Nếu trong trường hợp truyền khung dữ liệu thì chuyển trạng thái kênh truyền sang hoàn thành.

Bước 5. Nếu các cờ ngắt bên trên không được set thì reset toàn bộ cờ ngắt.

2.4.1.12 Hàm Lin_Linflex_ErrorInterruptHandler.

Mô tả: Hàm này sẽ xử lý tất cả các ngắt trên kênh định tuyến.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** Trạng thái Std_ReturnType return thực hiện hàm có thành công hay không?

Bước 1. Đọc giá trị thanh ghi trạng thái lỗi LINESR. Các bước tiếp theo có cấu trúc rẽ nhánh if-else.

Bước 2. Nếu cờ ngắt LINESR_BEF được set lên:

- Xóa bit LINESR_BEF.
- Kiểm tra lỗi truyền/ nhận xảy ra ở trạng thái nào?
- Dừng toàn bộ các frame đang truyền, set bit LINCR2_ABRQ.
- Chuyển biến trạng thái lỗi frame Lin_au8LinChFrameErrorStatus thành LIN_BIT_ERROR.
- Nếu kiểu frame là lệnh sleep, set biến bIsTxErrorOnSleepCmd = TRUE.

Bước 3. Nếu cờ ngắt LINESR_CEF được set lên:

- Xóa bit LINESR_CEF.
- Chuyển biến trạng thái lỗi frame Lin_au8LinChFrameErrorStatus thành LIN_CHECKSUM_ERROR.

Bước 4. Nếu cờ ngắt LINESR_OCF được set lên:

- Xóa bit LINESR_OCF.
- Chuyển biến trạng thái lỗi frame Lin_au8LinChFrameErrorStatus thành LIN_TIMEOUT_ERROR.
- Xóa LINIER_OCIE.
- Nếu kiểu frame là lệnh sleep, set biến bIsTxErrorOnSleepCmd = TRUE.

Bước 5. Nếu cờ ngắt LINESR_FEF được set lên:

- Xóa bit LINESR_FEF.
- Chuyển biến trạng thái lỗi frame Lin_au8LinChFrameErrorStatus thành LIN_FRAMING_ERROR.

Bước 6. Nếu cờ ngắt LINESR_BOF được set lên:

- Xóa bit LINESR_BOF.
- Chuyển biến trạng thái lỗi frame Lin_au8LinChFrameErrorStatus thành LIN_BUFFER_OVER_RUN_ERROR.

Bước 7. Nếu bit bIsTxErrorOnSleepCmd = TRUE, thực hiện:

- Xóa cờ wakeup: LINSR_WUF.
- Enable bit LINIER_WUIE.
- Thiết lập module vào trạng thái ngủ, set bit LINCR1_SLEEP.
- Chuyển trạng thái các biến toàn cục sang state ngủ.

2.4.2 IP Vault ESCI.

2.4.2.1 Hàm Lin_ESCI_InitChannel.

Mô tả: Hàm này khởi tạo tất cả các thanh ghi phần cứng cần thiết để bắt đầu chức năng LIN trên kênh đã chọn.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void

Bước 1. Nếu cấu hình có tùy chọn support wakeup và cờ IFSR2_LWAKE bật thì tiến hành wakeup thông qua hàm EcuM_SetWakeupEvent.

Bước 2. Cho phép phần cứng ESCI hoạt động, xóa bit CR2_MDIS.

Bước 3. Bật chế độ LIN cho phần cứng ESCI, set bit LCR1_LIN.

Bước 4. Reset giá trị các trạng thái LIN, set bit LCR1_LRES.

Bước 5. Thiết lập tốc độ truyền baudrate bằng cách ghi giá trị từ Lin_pChannelConfigPtr[] vào thanh ghi ESCI_BRR.

Bước 6. Cho phép chức năng truyền và nhận của ESCI hoạt động, set hai bit CR1_TE và CR1_RE.

Bước 7. Trong chế độ hoạt động có hỗ trợ DMA, cho phép DMA dừng truyền khi xảy ra lỗi Bit hoặc lỗi bus bằng cách set bit CR2_BSTP.

Bước 8. Cho phép ngắt lỗi bit, set bit CR2_BERRIE.

Bước 9. Thiết lập độ dài của trường break là 13 bit, set bit CR2_BRCL.

Bước 10. Cho phép chế độ lấy mẫu xác định lỗi bit nhanh (Fast Bit Error Detection Sample), set bit CR2_BESM.

Bước 11. Dừng truyền ngay khi xảy ra bit error, set bit CR2_BESTP.

Bước 12. Cho phép ngắt lỗi frame (Frame Error) set bit CR2_FEIE.

Bước 13. Reset state machine bằng cách xóa bit LCR1_LRES.

Bước 14. Cho phép ngắt cờ lỗi parity, set bit LCR1_PRTY.

Bước 15. Nếu cấu hình tính năng: LIN_DISABLE_FRAME_TIMEOUT == STD_OFF cần set bit LCR1_STIE, ngược lại xóa set này.

Bước 16. Cho phép phần cứng phát hiện lỗi bus vật lý, set bit LCR1_PBIE.

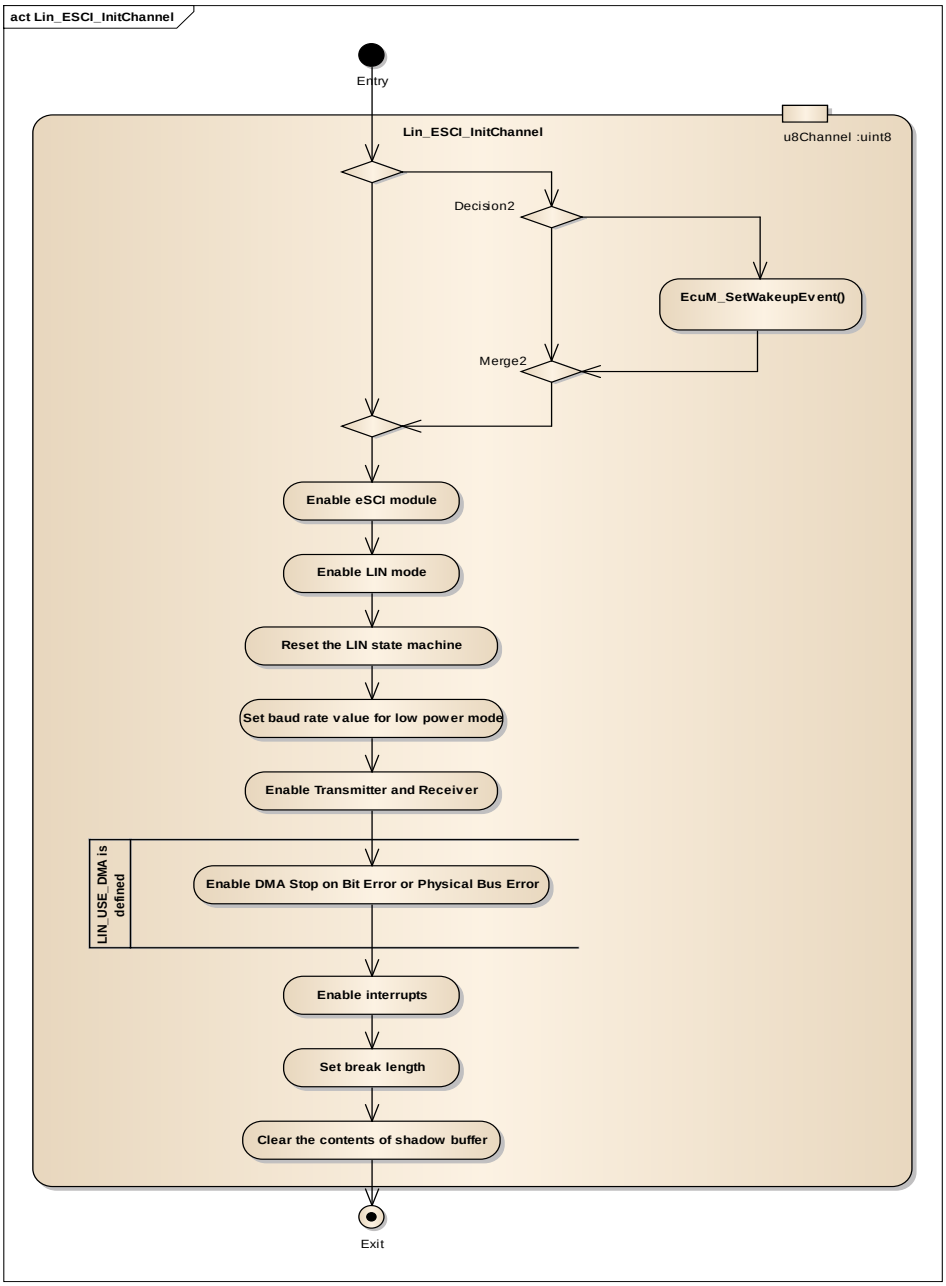
Bước 17. Cho phép phần cứng phát hiện lỗi checksum, set bit LCR1_CKIE.

Bước 18. Vô hiệu hóa ngắt lỗi CRC, xóa bit LCR1_CIE.

Bước 19. Cho phép phần cứng phát hiện lỗi overflow, set bit LCR2_OFIE.

Bước 20. Vô hiệu hóa ngắt wakeup, xóa bit LCR1_WUIE.

Bước 21. Xóa dữ liệu trong biến buffer Lin_ESCI_pChShadowBuffPtr[].



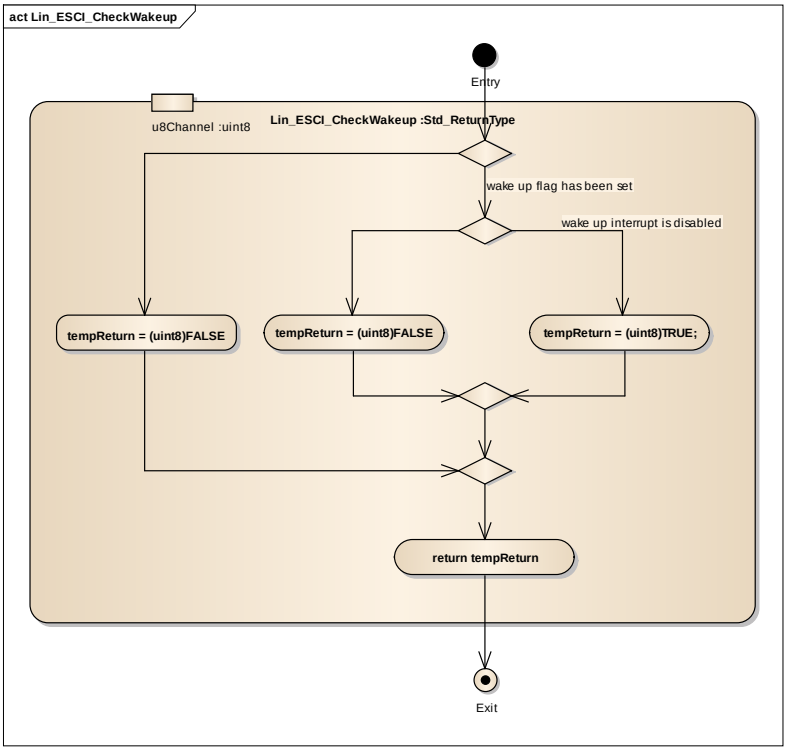
2.4.2.2 Hàm Lin_ESCI_CheckWakeup.

Mô tả: Hàm này xác định nếu kênh LIN được wakeup bởi đường truyền bus LIN. API này được sử dụng khi chức năng wakeup kênh LIN bị vô hiệu hóa (ngắt wakeup bị tắt). Nó kiểm tra cờ wakeup từ kênh LIN trong chế độ ngủ và có chức năng wakeup bị tắt.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void

Bước 1. Nếu cờ wakep IFSR2_LWAKE được bật và bit enable LCR1_WUIE bị vô hiệu hóa, return lại lại là TRUE và xóa cờ wakeup IFSR2_LWAKE.

Bước 2. Các trường hợp còn lại return lại FALSE.

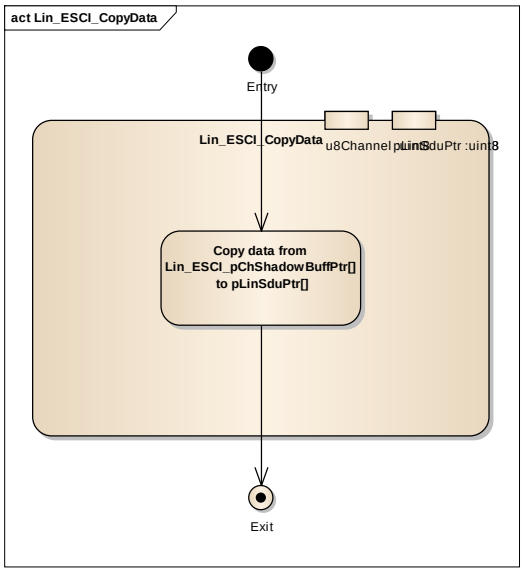


2.4.2.3 Hàm Lin_ESCI_CopyData.

Mô tả: Hàm này sao chép dữ liệu nhận được trong bộ đệm được cung cấp bởi tham số con trỏ pu8LinSduPtr.

- **Input:** u8Channel (địa chỉ kênh vật lý), con trỏ pu8LinSduPtr trỏ đến buffer hoặc bộ nhớ phần cứng SCI nơi có SDU được lưu trữ.
- **Output:** void

Bước 1. Copy dữ liệu nhận được từ buffer đến pLinSduPtr[]



2.4.2.4 Hàm Lin_ESCI_SendHeader.

Mô tả: Khởi tạo truyền phần tiêu đề (header) của khung truyền LIN trên Bus bằng cách sử dụng thông tin cấu hình được lưu trữ trên con trỏ pPduInfoPtr. Nếu frame truyền là loại MASTER_RESPONSE thì phần tiêu đề này sẽ không truyền lên Bus mà toàn bộ frame (bao gồm cả tiêu đề) sẽ được truyền trong hàm Lin_LINFlex_SendReponse.

- **Input:** u8Channel (địa chỉ kênh vật lý), pPduInfoPtr (Con trỏ đến PDU chứa PID, mô hình Checksum, loại phản hồi Master/ Slave response, con trỏ dữ liệu Dl và SDU)
- **Output:** Trạng thái Std_ReturnType truyền phần tiêu đề thành công hay không?

Bước 1. Dừng mọi hoạt động truyền nhận đang xảy ra, xóa 2 bit điều khiển CR1_RIE, CR1_TIE và 2 bit ngắt truyền, nhận LCR1_RXIE và LCR1_TXIE.

Bước 2. Vô hiệu hóa chế độ DMA, xóa 2 bit CR2_TXDMA và CR2_RXDMA.

Bước 3. Vô hiệu hóa ngắt điều khiển frame, xóa bit LCR1_FCIE.

Bước 4. Hủy bỏ khung truyền đang xảy ra.

Bước 5. Thiết lập các biến trạng thái

Bước 6. Reset trạng thái, set bit LCR1_LRES.

Bước 7. Xóa cờ lỗi Noise, xóa IFSR1_NF.

Bước 8. Tính giá trị byte PID và gán vào Lin_ESCI_pChShadowBuffPtr[]

Bước 9. Gán giá trị các trường độ dài dữ liệu Dl, Checksum vào Lin_ESCI_pChShadowBuffPtr[]

Bước 10. Xóa cờ ngắt IFSR1_RDRF

Bước 11. Cho phép ngắt nhận hoạt động, set bit CR1_RIE.

Bước 12. Reset trạng thái LIN, xóa bit LCR1_LRES

Bước 13. Cho phép phần cứng ESCI hoạt động truyền/ nhận, set hai bit CR1_RE và CR1_TE.

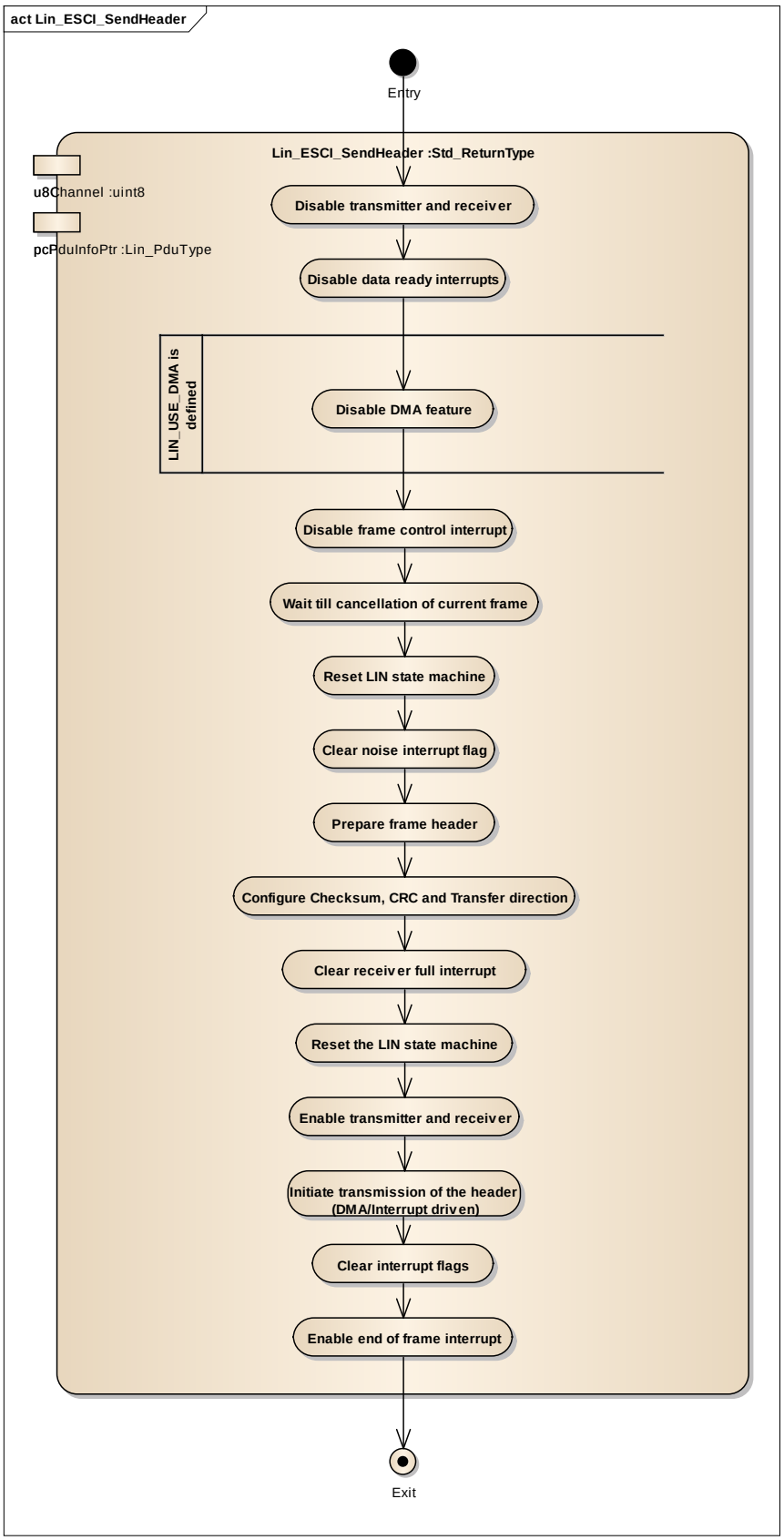
Bước 14. Trong chế độ truyền/ nhận có hỗ trợ DMA, ghi các giá trị dữ liệu vào DMA channel.

Bước 15. Nếu loại frame khác LIN_MASTER_RESPONSE, tiến hành xóa cờ ngắt nhận IFSR2_RXRDY và cho phép enable LCR1_RXIE.

Bước 16. Nếu loại frame là LIN_SLAVE_RESPONSE, reset các biến buffer và cho phép ngắt truyền set bit LCR1_TXIE.

Bước 17. Xóa các cờ trạng thái lỗi: IFSR1_NF, IFSR1_FE, IFSR1_BERR, IFSR2_STO, IFSR2_PBERR, IFSR2_CKERR, IFSR2_OVFL, IFSR2_UREQ, IFSR2_FRC.

Bước 18. Bắt đầu truyền phần tiêu đề khung bằng yêu cầu ngắt LCR1_FCIE.


act Lin_ESCI_SendHeader
Entry
Lin_ESCI_SendHeader :Std_ReturnType
u8Channel :uint8
pcPduInfoPtr :Lin_PduType
Disable transmitter and receiver
Disable data ready interrupts
LIN_USE_DMA is defined
Disable DMA feature
Disable frame control interrupt
Wait till cancellation of current frame
Reset LIN state machine
Clear noise interrupt flag
Prepare frame header
Configure Checksum, CRC and Transfer direction
Clear receiver full interrupt
Reset the LIN state machine
Enable transmitter and receiver
Initiate transmission of the header (DMA/Interrupt driven)
Clear interrupt flags
Enable end of frame interrupt
Exit

2.4.2.5 Hàm Lin_ESCI_SendResponse.

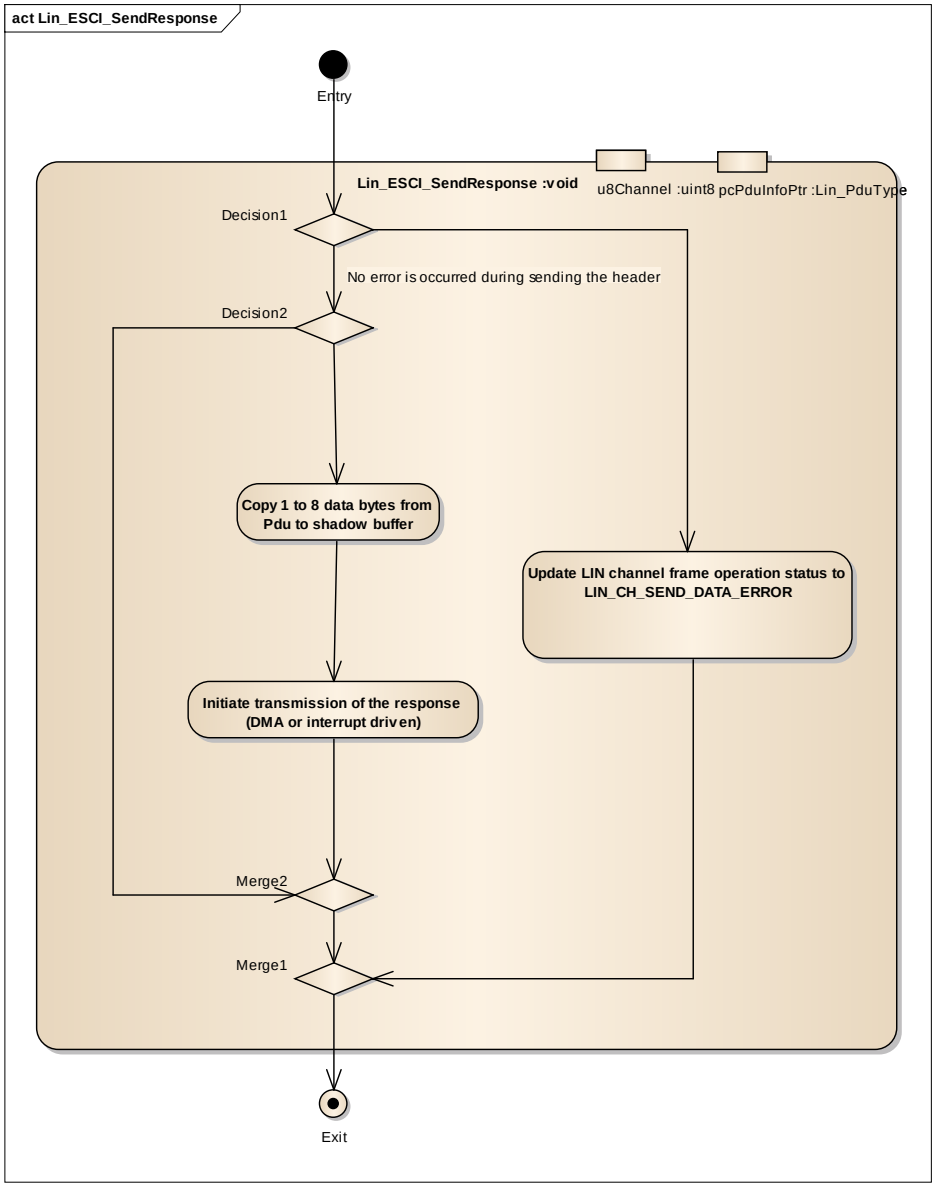
Mô tả: Khởi tạo truyền phần dữ liệu (response) của khung truyền LIN trên Kênh bằng cách sử dụng thông tin được lưu trữ trên con trỏ pPduInfoPtr.

- **Input:** u8Channel (địa chỉ kênh vật lý), pPduInfoPtr (Con trỏ đến PDU chứa PID, mô hình Checksum, loại phản hồi Master/ Slave response, con trỏ dữ liệu Dl và SDU)
- **Output:** Trạng thái Std_ReturnType truyền phần tiêu đề thành công hay không?

Bước 1. Nếu xảy ra lỗi truyền phần tiêu đề thì dừng truyền phần phản hồi.

Bước 2. Copy dữ liệu từ con trỏ cấu hình vào buffer.

Bước 3. Cho phép ngắt truyền set bit LCR1_TXIE.



2.4.2.6 Hàm Lin_ESCI_GoToSleep.

Mô tả: Hàm này dừng bất kỳ quá trình truyền nhận nào đang diễn ra và bắt đầu truyền lệnh ngủ với khung truyền: ID = 0x3C và dữ liệu = (0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF).

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** Trạng thái Std_ReturnType return khung truyền có thành công hay không?

Bước 1. Dừng mọi hoạt động truyền nhận đang xảy ra, xóa 2 bit điều khiển CR1_RIE, CR1_TIE và 2 bit ngắt truyền, nhận LCR1_RXIE và LCR1_TXIE.

Bước 2. Vô hiệu hóa chế độ DMA, xóa 2 bit CR2_TXDMA và CR2_RXDMA.

Bước 3. Vô hiệu hóa ngắt điều khiển frame, xóa bit LCR1_FCIE.

Bước 4. Chờ kết thúc quá trình truyền hoặc nhận, chờ cờ IFSR1_TACT được xóa.

Bước 5. Reset state machine bằng cách xóa bit LCR1_LRES 4 lần.

Bước 6. Cho phép truyền và nhận, set 2 bit CR1_TE và CR1_RE.

Bước 7. Thiết lập các buffer dữ liệu.

Bước 8. Xóa cờ ngắt frame, xóa bit IFSR2_FRC.

Bước 9. Cho phép ngắt LCR1_FCIE.

2.4.2.7 Hàm Lin_ESCI_GoToSleepInternal.

Mô tả: Hàm này dừng bất kỳ quá trình truyền nào đang diễn ra và đặt Kênh vào chế độ ngủ (sau đó phần cứng LIN đi vào chế độ hoạt động giảm năng lượng).

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** Trạng thái Std_ReturnType return thực hiện hàm có thành công hay không?

Bước 1. Dừng mọi hoạt động truyền nhận đang xảy ra, xóa 2 bit điều khiển CR1_RIE, CR1_TIE và 2 bit ngắt truyền, nhận LCR1_RXIE và LCR1_TXIE.

Bước 2. Vô hiệu hóa chế độ DMA, xóa 2 bit CR2_TXDMA và CR2_RXDMA.

Bước 3. Vô hiệu hóa ngắt điều khiển frame, xóa bit LCR1_FCIE.

Bước 4. Reset trạng thái, set bit LCR1_LRES.

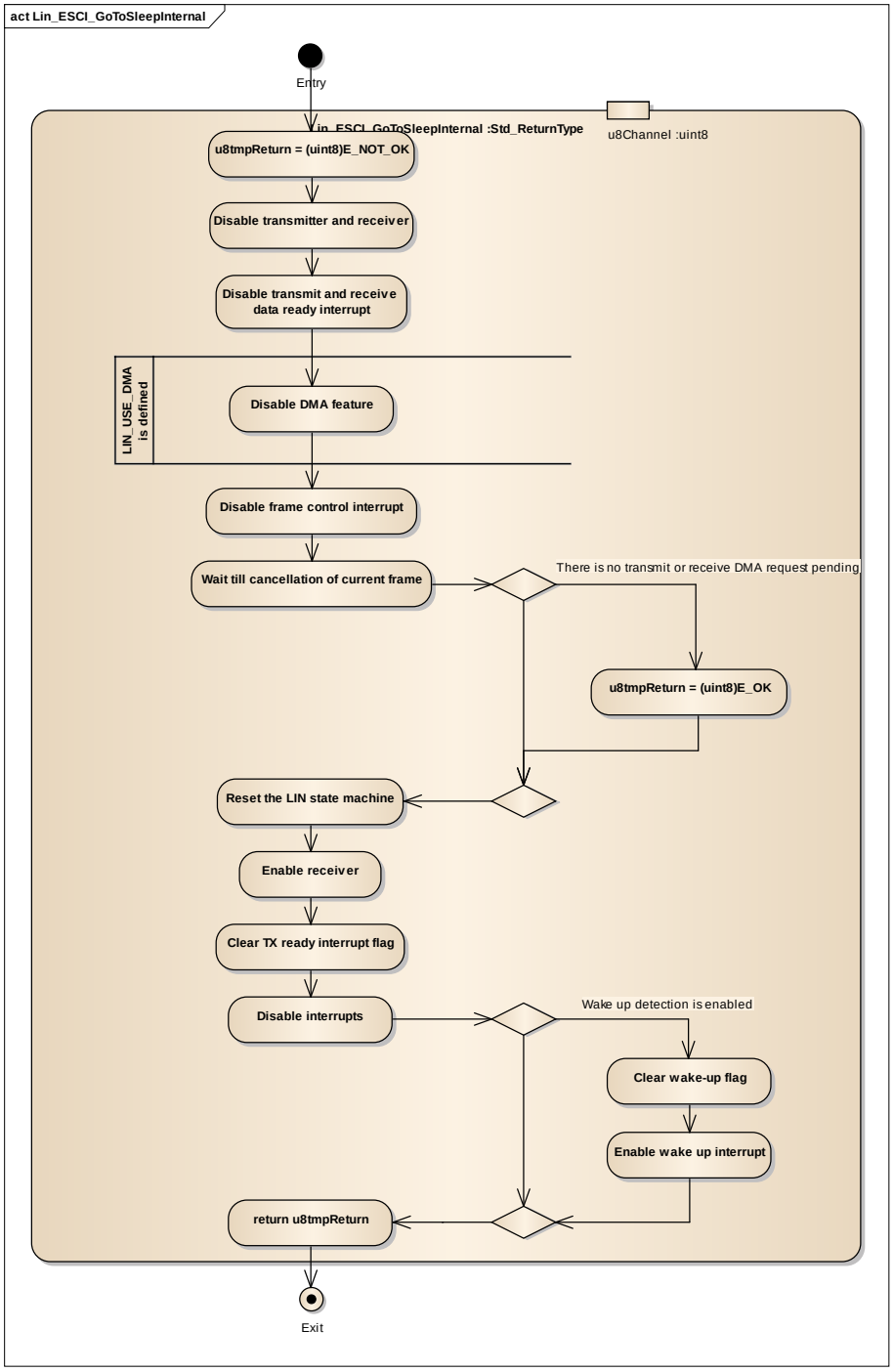
Bước 5. Cho phép ngắt nhận hoạt động, set bit CR1_RE.

Bước 6. Vô hiệu hóa ngắt nhận, xóa cờ truyền IFSR2_TXRDY và LCR1_TXIE.

Bước 7. Vô hiệu hóa ngắt lỗi frame, xóa IFSR1_FE và CR2_FEIE.

Bước 8. Vô hiệu hóa ngắt lỗi noise, xóa IFSR1_NF và CR2_NFIE.

Bước 9. Bật tính năng detect xung wakeup xóa cờ IFSR2_LWAKE và set bit LCR1_WUIE.

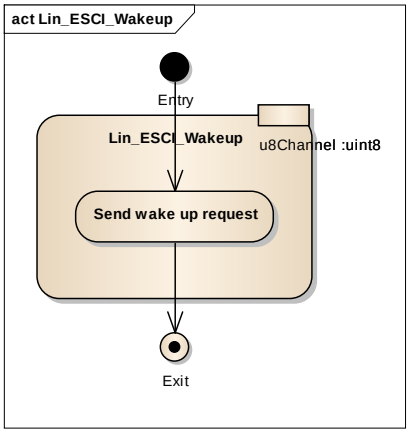


2.4.2.8 Hàm Lin_ESCI_WakeUp.

Mô tả: Hàm này sẽ gửi một tín hiệu wakeup đến đường bus LIN và đặt kênh LIN ở trạng thái hoạt đồn LIN_CH_OPERATIONAL.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void

Bước 1. Gửi xung wakeup lên bus LIN, set bit LCR1_WU.



2.4.2.9 Hàm Lin_ESCI_HardwareGetStatus.

Mô tả: Hàm này trả về trạng thái của quá trình truyền, nhận khung truyền hoặc chế độ hoạt động hiện tại. Nếu quá trình nhận phần phản ứng Slave đã thành công thì hàm này cung cấp một con trỏ tới bộ đệm, nơi dữ liệu được lưu trữ.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** Con trỏ pu8LinSduPtr trỏ tới bộ đệm hoặc bộ nhớ phần cứng được ánh xạ nơi SDU hiện tại được lưu trữ

2.4.2.10 Hàm Lin_ESCI_InterruptHandler.

Mô tả: Hàm này sẽ xử lý tất cả các ngắt RX và TX trên kênh định tuyến.

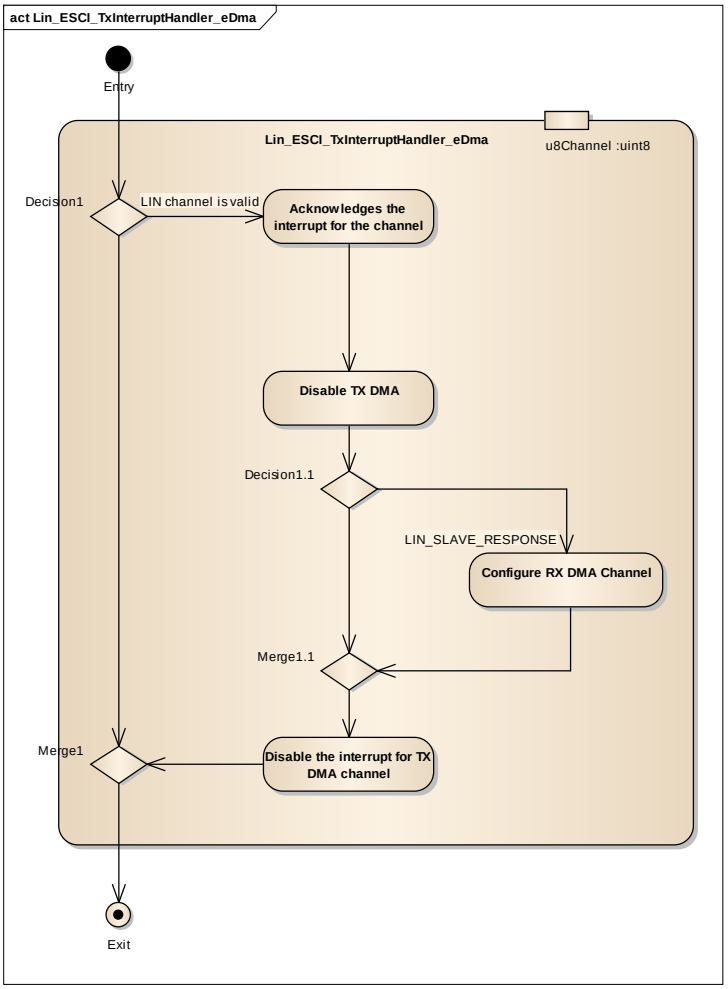
- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void

2.4.2.11 Hàm Lin_ESCI_TxInterruptHandler_eDma.

Mô tả: Hàm này sẽ xử lý tất cả các ngắt RX và TX trên kênh định tuyến.

- **Input:** u8Channel (địa chỉ kênh vật lý)

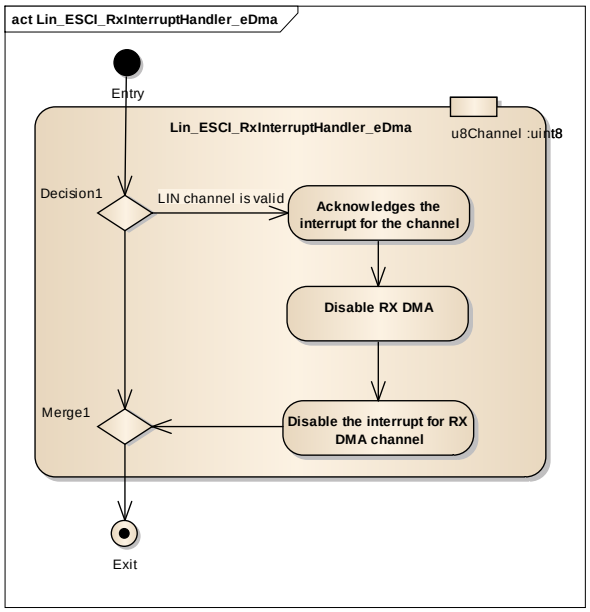
- **Output:** void



2.4.2.12 Hàm Lin_ESCI_RxInterruptHandler_eDma.

Mô tả: Hàm này sẽ xử lý tất cả các ngắt RX và TX trên kênh định tuyến.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void



2.4.3 IP Vault SCI.

2.4.3.1 Hàm Lin_SCI_InitChannel.

Mô tả: Hàm này khởi tạo tất cả các thanh ghi phần cứng cần thiết để bắt đầu chức năng LIN trên kênh đã chọn.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void

Bước 1. Chọn tùy chọn phần cứng SCI tính giá trị baudrate theo công thức: SCI Baud Rate = SCI Bus Clock/ (SBR[15:0]), xóa bit ACR2_IREN.

Bước 2. Theo chuẩn LIN, chọn tùy chọn độ dài của trường break là 13 bit, set bit SR2_BRK13.

Bước 3. Chọn kiểu định dạng byte trong khung truyền là 8 bit độ dài, xóa bit CR1_M.

Bước 4. Cấu hình phần cứng SCI hoạt động trong chế độ Wait của vi điều khiển, xóa bit CR1_SCISWA.

Bước 5. Khởi tạo giá trị baudrate bằng cách ghi vào trường SCI_BDH.

Bước 6. Xóa các cờ trạng thái bằng cách đọc lần lượt thanh ghi SCI_SR1 và SCI_DRL.

Bước 7. Cho phép ngắt cờ phát hiện lỗi bit, set bit ACR1_BERRIE.

Bước 8. Chọn chế độ lấy mẫu khi kiểm tra lỗi truyền bit tại time stick thứ 9, set hai bit BERRM[1:0] = 01b.

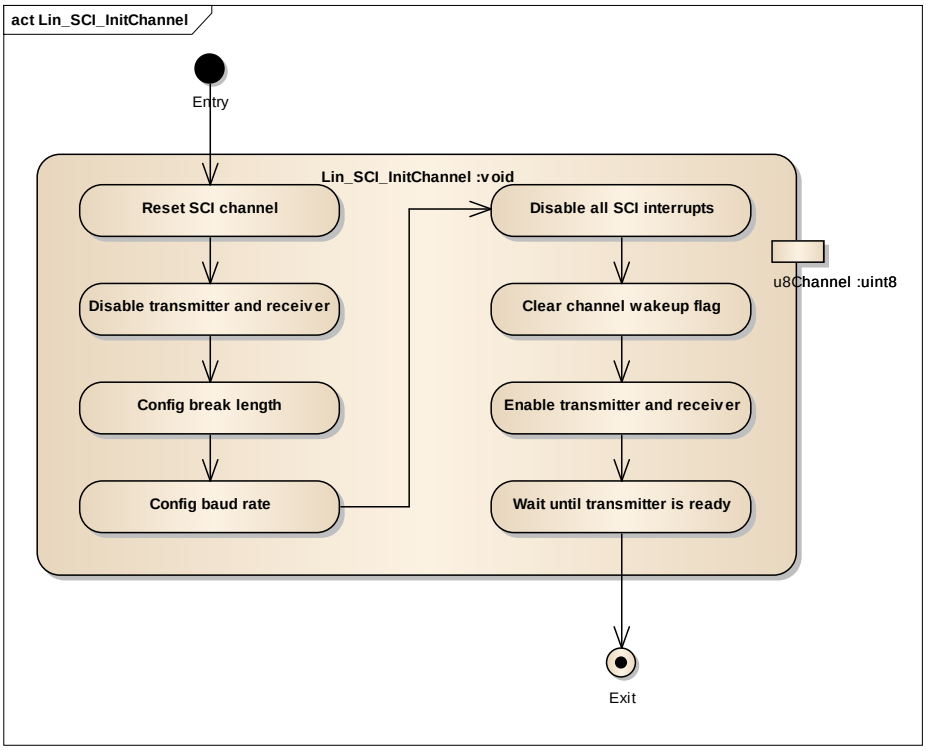
Bước 9. Xóa các cờ trạng thái SCI_ASR1.

Bước 10. Vô hiệu hóa chức năng truyền và nhận của SCI, xóa hai bit CR2_TE và CR2_RE.

Bước 11. Xóa biến cờ wakeup.

Bước 12. Cho phép chức năng truyền và nhận của SCI hoạt động, set hai bit CR2_TE và CR2_RE.

Bước 13. Xóa bit SR2_RXPOL.



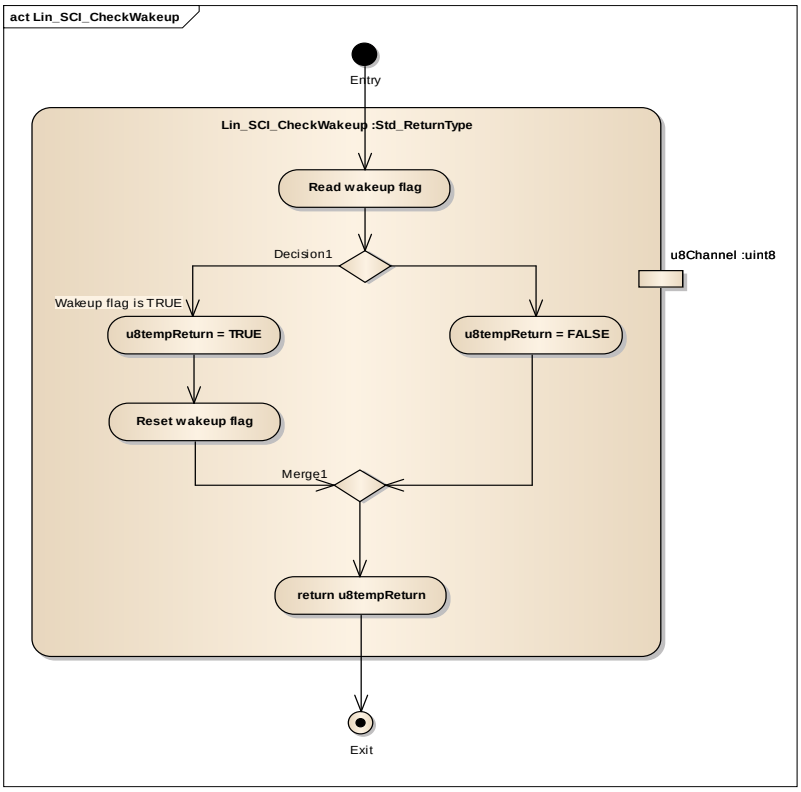
2.4.3.2 Hàm Lin_SCI_CheckWakeup.

Mô tả: Hàm này xác định nếu kênh LIN được wakeup bởi đường truyền bus LIN. API này được sử dụng khi chức năng wakeup kênh LIN bị vô hiệu hóa (ngắt wakeup bị tắt). Nó kiểm tra cờ wakeup từ kênh LIN trong chế độ ngủ và có chức năng wakeup bị tắt.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void

Bước 1. Đọc biến cờ wakeup: Lin_Sci_au8WakeupFlag[].

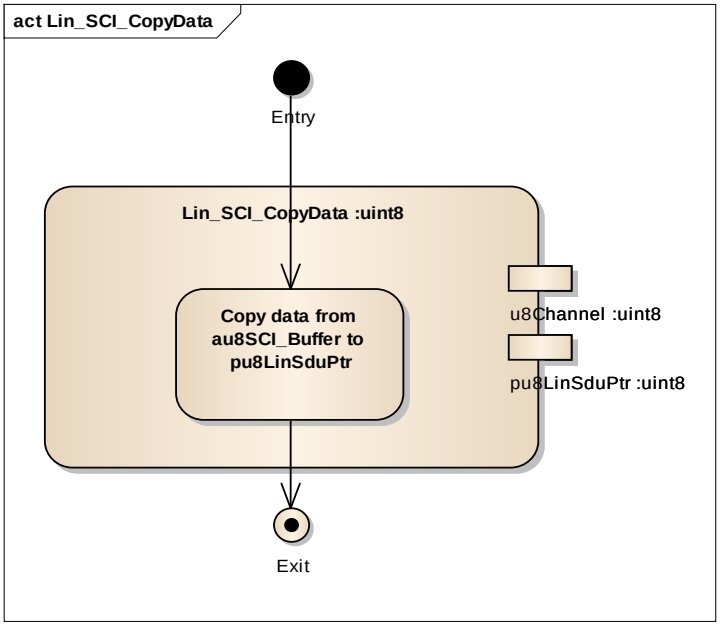
Bước 2. Reset giá trị biến này nếu đang được set.



2.4.3.3 Hàm Lin_SCI_CopyData.

Mô tả: Hàm này sao chép dữ liệu nhận được trong bộ đệm được cung cấp bởi tham số con trỏ pu8LinSduPtr.

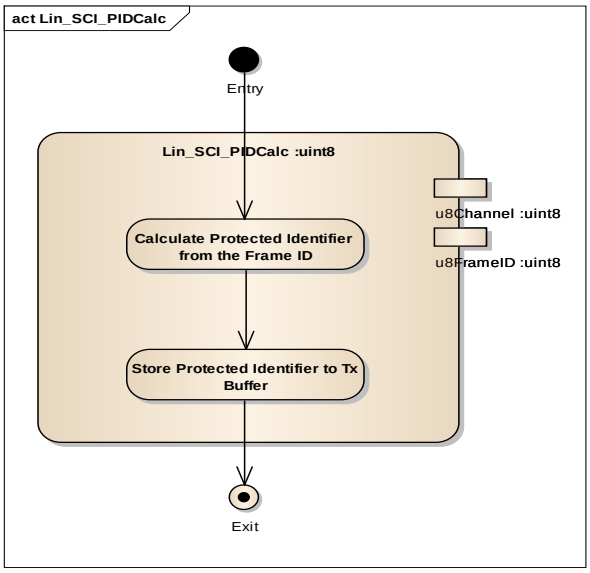
- **Input:** u8Channel (địa chỉ kênh vật lý), con trỏ pu8LinSduPtr trỏ đến buffer hoặc bộ nhớ phần cứng SCI nơi có SDU được lưu trữ.
- **Output:** void



2.4.3.4 Hàm Lin_SCI_PIDCalc.

Mô tả: Phần cứng SCI không hỗ trợ tính tự động trường PID, vì vậy chúng ta cần xây dựng một hàm mềm để thực hiện việc này. Hàm này tính giá trị byte PID từ ID khung truyền (gồm 6 bit D0 -> D5).

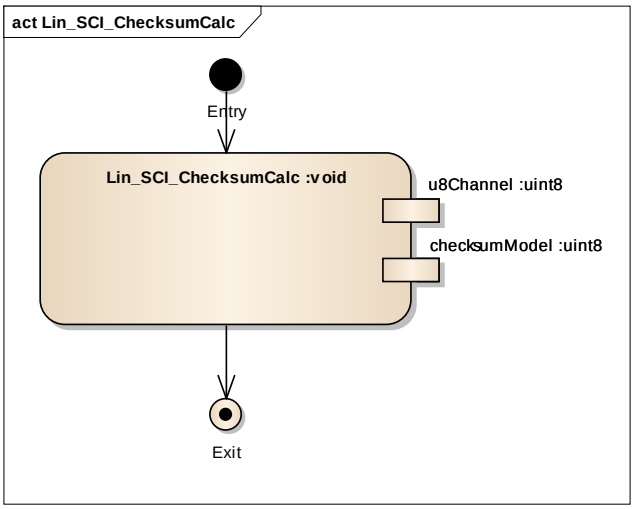
- **Input:** u8Channel (địa chỉ kênh vật lý), ID khung truyền.
- **Output:** void



2.4.3.5 Hàm Lin_SCI_ChecksumCalc.

Mô tả: Phần cứng SCI không hỗ trợ tính tự động trường checksum, vì vậy chúng ta cần xây dựng một hàm mềm để thực hiện việc này.

- **Input:** u8Channel (địa chỉ kênh vật lý), checksumModel (kiểu checksum).
- **Output:** u8Checksum



2.4.3.6 Hàm Lin_SCI_SendHeader.

Mô tả: Khởi tạo truyền phần tiêu đề (header) của khung truyền LIN trên Bus bằng cách sử dụng thông tin cấu hình được lưu trữ trên con trỏ pPduInfoPtr. Nếu frame truyền là loại MASTER_RESPONSE thì phần tiêu đề này sẽ không truyền lên Bus mà toàn bộ frame (bao gồm cả tiêu đề) sẽ được truyền trong hàm Lin_LINFlex_SendReponse.

- **Input:** u8Channel (địa chỉ kênh vật lý), pPduInfoPtr (Con trỏ đến PDU chứa PID, mô hình Checksum, loại phản hồi Master/ Slave response, con trỏ dữ liệu Dl và SDU)
- **Output:** Trạng thái Std_ReturnType truyền phần tiêu đề thành công hay không?

Bước 1. Dừng mọi hoạt động truyền nhận đang xảy ra, xóa 2 bit CR2_RIE, CR2_TIE.

Bước 2. Hủy bỏ dữ liệu trong buffer nhận, đọc thanh ghi SCI_DRL.

Bước 3. Reset các trạng thái lỗi: Lin_au8LinChFrameErrorStatus.

Bước 4. Update trạng thái hoạt động của frame truyền: Lin_au8LinChFrameStatus.

Bước 5. Update biến lưu trữ độ dài khung dữ liệu: Lin_Sci_au8BufferLength[]

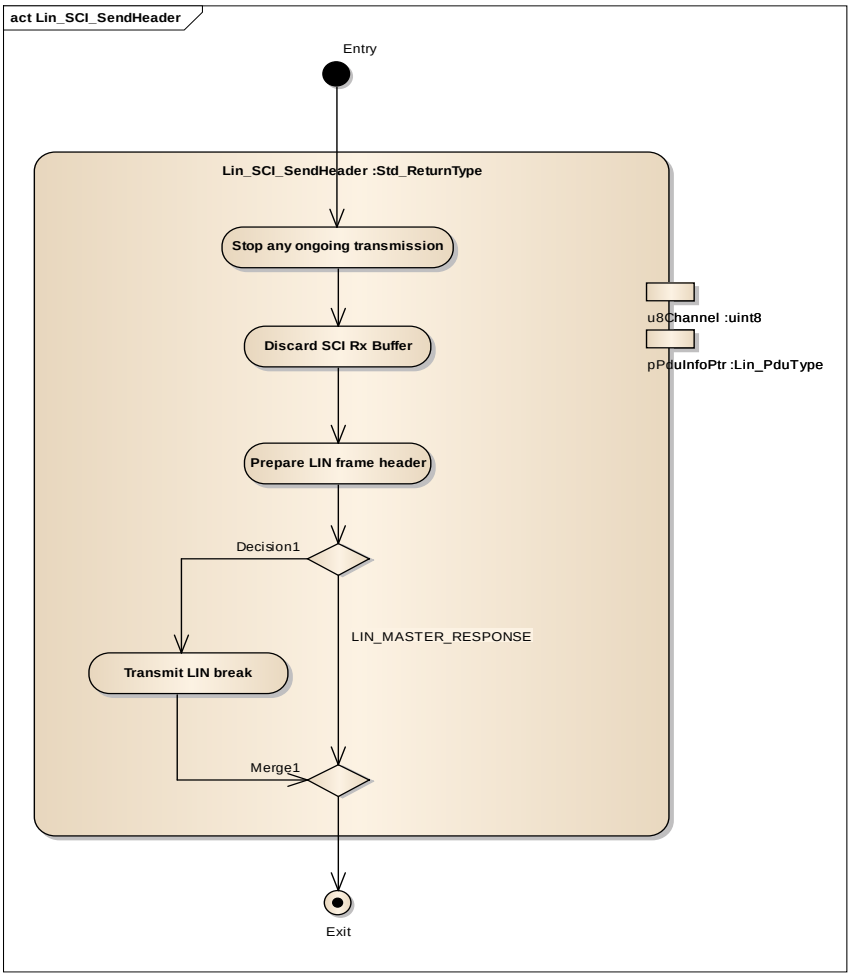
Bước 6. Ghi lần lượt giá trị byte Sync và PID vào mảng 2 chiều Lin_Sci_au8Buffer[][]

Bước 7. Nếu khung truyền là dạng Slave response thì đặt byte dữ liệu truyền tiếp theo là Sync: Lin_Sci_au8Buffer

Bước 8. Cho phép ngắt nhận hoạt động, set bit: CR2_RIE

Bước 9. Truyền trường break bằng cách lần lượt bật và tắt bit CR2_SBK.

Bước 10. Cho phép ngắt truyền hoàn thành hoạt động, set bit CR2_TCIE.



2.4.3.7 Hàm Lin_SCI_SendResponse.

Mô tả: Khởi tạo truyền phần dữ liệu (response) của khung truyền LIN trên Kênh bằng cách sử dụng thông tin được lưu trữ trên con trỏ pPduInfoPtr.

- **Input:** u8Channel (địa chỉ kênh vật lý), pPduInfoPtr (Con trỏ đến PDU chứa PID, mô hình Checksum, loại phản hồi Master/ Slave response, con trỏ dữ liệu Dl và SDU)
- **Output:** Trạng thái Std_ReturnType truyền phần tiêu đề thành công hay không?

Bước 1. Ghi trường dữ liệu và byte checksum vào mảng 2 chiều Lin_Sci_au8Buffer[][].

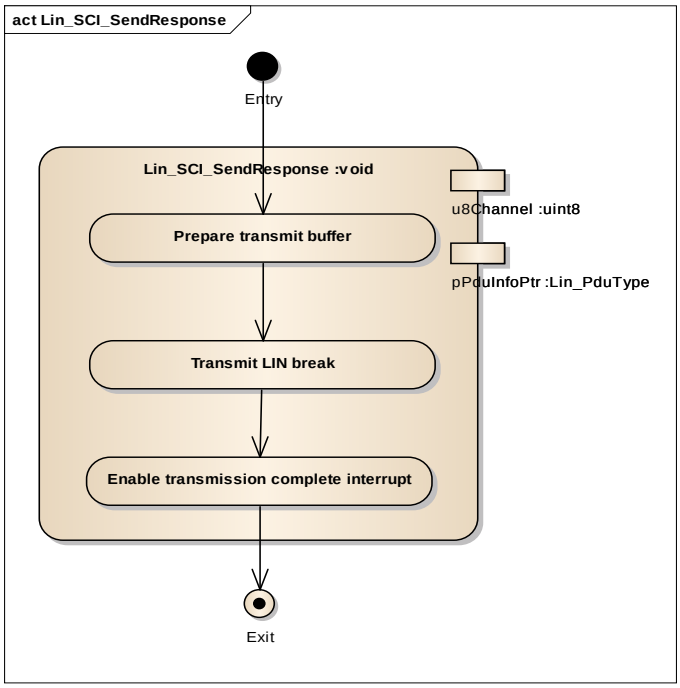
Bước 2. Đặt byte dữ liệu truyền tiếp theo là byte Sync: Lin_Sci_au8Buffer.

Bước 3. Hủy bỏ dữ liệu trong buffer nhận, đọc thanh ghi SCI_DRL.

Bước 4. Cho phép ngắt nhận hoạt động, set bit: CR2_RIE.

Bước 5. Truyền trường break bằng cách lần lượt bật và tắt bit CR2_SBK.

Bước 6. Cho phép ngắt truyền hoàn thành hoạt động, set bit CR2_TCIE.



2.4.3.8 Hàm Lin_SCI_GoToSleep.

Mô tả: Hàm này dừng bất kỳ quá trình truyền nhận nào đang diễn ra và bắt đầu truyền lệnh ngủ với khung truyền: ID = 0x3C và dữ liệu = (0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF).

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** Trạng thái Std_ReturnType return khung truyền có thành công hay không?

Bước 1. Dừng mọi hoạt động truyền nhận đang xảy ra, xóa 3 bit CR2_RIE, CR2_TIE, CR2_TCIE.

Bước 2. Hủy bỏ dữ liệu trong buffer nhận, đọc thanh ghi SCI_DRL.

Bước 3. Ghi giá trị độ dài khung truyền (11 byte) vào biến Lin_Sci_au8BufferLength[] do đã biết trước khung truyền lệnh sleep có 8 byte dữ liệu.

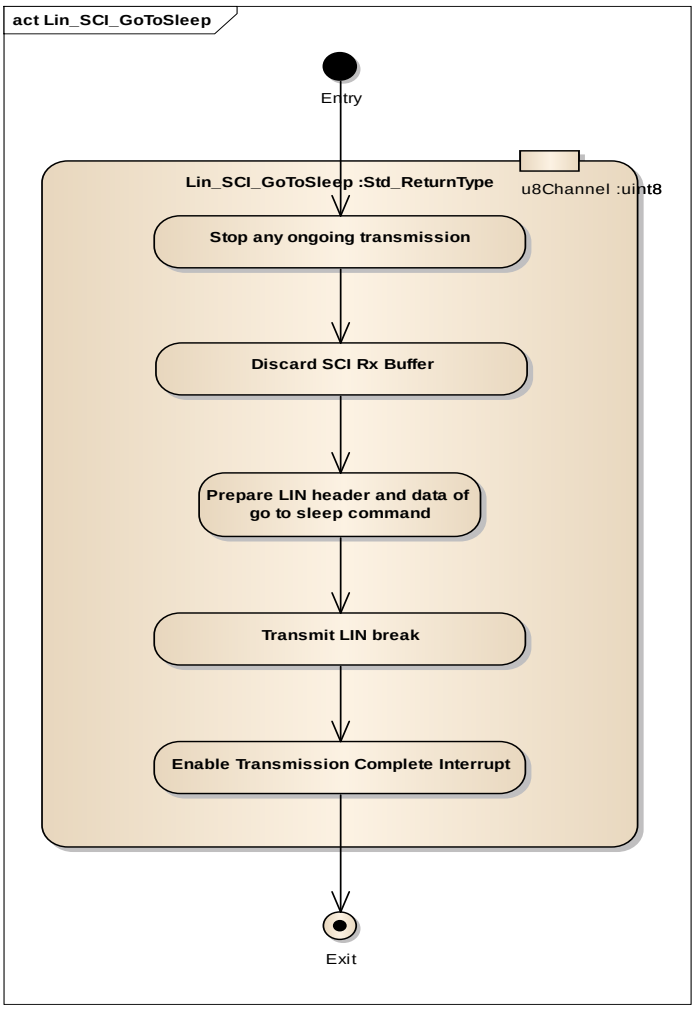
Bước 4. Ghi lần lượt trường sync, PID, dữ liệu, byte checksum vào mảng 2 chiều Lin_Sci_au8Buffer[][].

Bước 5. Cho phép ngắt nhận hoạt động, set bit: CR2_RIE.

Bước 6. Truyền trường break bằng cách lần lượt bật và tắt bit CR2_SBK.

Bước 7. Thiết lập: Lin_au8TransmitHeaderCommand[u8Channel] = LIN_TX_SLEEP_COMMAND.

Bước 8. Cho phép ngắt truyền hoàn thành hoạt động, set bit CR2_TCIE.



2.4.3.9 Hàm Lin_SCI_GoToSleepInternal.

Mô tả: Hàm này dừng bất kỳ quá trình truyền nào đang diễn ra và đặt Kênh vào chế độ ngủ (sau đó phần cứng LIN đi vào chế độ hoạt động giảm năng lượng).

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** Trạng thái Std_ReturnType return thực hiện hàm có thành công hay không?

Bước 1. Dừng mọi hoạt động truyền đang xảy ra, xóa 2 bit CR2_TIE, CR2_TCIE.

Bước 2. Hủy bỏ dữ liệu trong buffer nhận, đọc thanh ghi SCI_DRL.

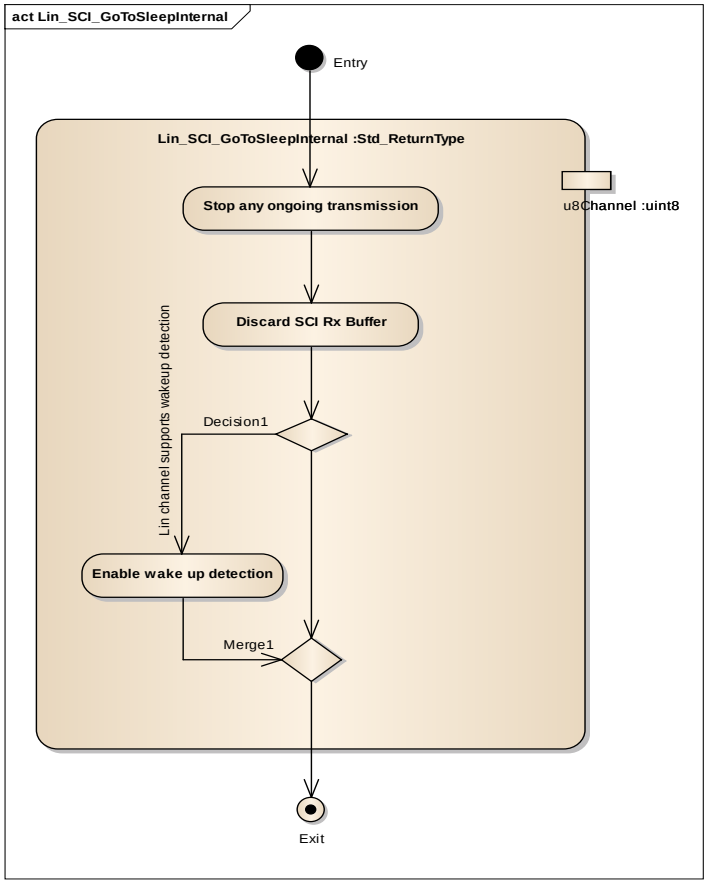
Bước 3. Update trạng thái kênh truyền LIN_CH_SLEEP_STATE.

Bước 4. Reset độ dài khung truyền Lin_Sci_au8BufferLength.

Bước 5. Cho phép phần cứng SCI detect được yêu cầu wakeup bằng cách xóa cờ ngắt ASR1_RXEDGIF và set bit ACR1_RXEDGIE.

Bước 6. Vô hiệu hóa ngắt nhận, xóa bit CR2_RIE.

Bước 7. Cho phép phần cứng SCI hoạt động hoặc không ở chế độ Wait của vi điều khiển, set hoặc xóa bit CR1_SCISWAI.



2.4.3.10 Hàm Lin_SCI_WakeUp.

Mô tả: Hàm này sẽ gửi một tín hiệu wakeup đến đường bus LIN và đặt kênh LIN ở trạng thái hoạt đồn LIN_CH_OPERATIONAL.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void

Bước 1. Tính giá trị của byte wakeup, ghi vào biến u8WakeupByte.

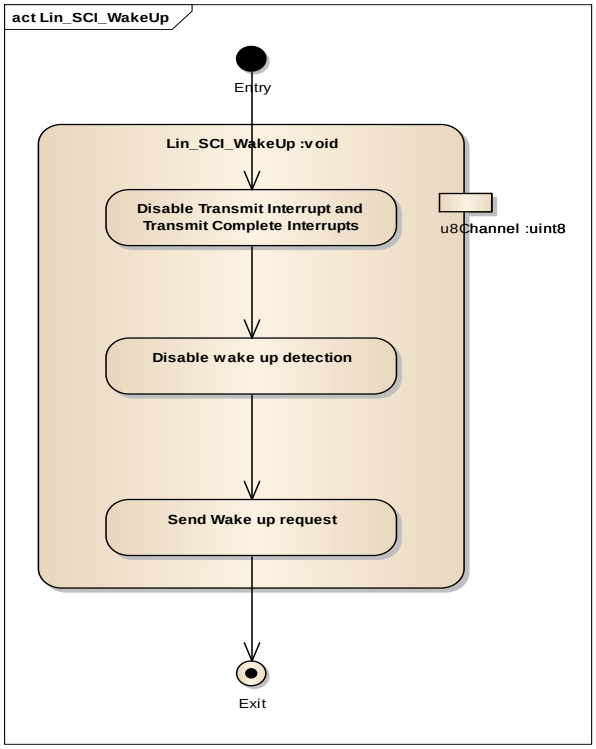
Bước 2. Dừng mọi hoạt động truyền đang xảy ra, xóa 2 bit CR2_TIE, CR2_TCIE.

Bước 3. Cho phép phần cứng SCI hoạt động ở chế độ Wait của vi điều khiển, xóa bit CR1_SCISWAI.

Bước 4. Vô hiệu hóa ngắt detect xung sườn xuống Rx, bằng cách xóa cờ ASR1_RXEDGIF và bit ACR1_RXEDGIE.

Bước 5. Vô hiệu hóa ngắt nhận, xóa bit CR2_RIE.

Bước 6. Truyền xung wakeup bằng cách ghi giá trị byte wakeup u8WakeupByte vào thanh ghi SCI_DRL.



2.4.3.11 Hàm Lin_SCI_WakeupInternal.

Mô tả: Hàm này sẽ đặt kênh LIN ở trạng thái hoạt động LIN_CH_OPERATIONAL mà không gửi tín hiệu wakeup đến bus LIN.

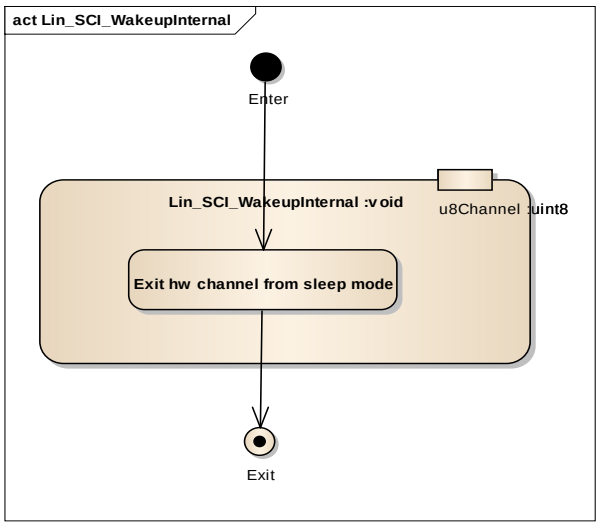
- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void

Bước 1. Dừng mọi hoạt động truyền đang xảy ra, xóa 2 bit CR2_TIE, CR2_TCIE.

Bước 2. Cho phép phần cứng SCI hoạt động ở chế độ Wait của vi điều khiển, xóa bit CR1_SCISWAI.

Bước 3. Vô hiệu hóa ngắt detect xung sườn xuống Rx, bằng cách xóa cờ ASR1_RXEDGIF và bit ACR1_RXEDGIE.

Bước 4. Vô hiệu hóa ngắt nhận, xóa bit CR2_RIE.



2.4.3.12 Hàm Lin_SCI_HardwareGetStatus.

Mô tả: Hàm này trả về trạng thái của quá trình truyền, nhận khung truyền hoặc chế độ hoạt động hiện tại. Nếu quá trình nhận phần phản ứng Slave đã thành công thì hàm này cung cấp một con trỏ tới bộ đệm, nơi dữ liệu được lưu trữ.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** Con trỏ pu8LinSduPtr trỏ tới bộ đệm hoặc bộ nhớ phần cứng được ánh xạ nơi SDU hiện tại được lưu trữ

Các bước thực hiện xem trong UML design hoặc tham chiếu driver.

2.4.3.13 Hàm Lin_SCI_TxRxInterruptHandler.

Mô tả: Hàm này sẽ xử lý tất cả các ngắt RX và TX trên kênh định tuyến.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void

Các bước thực hiện xem trong UML design hoặc tham chiếu driver.

2.4.3.14 Hàm Lin_SCI_RxBuffFullInterruptHandler.

Mô tả: Hàm này sẽ xử lý ngắt Rx full buffer, thực chất đây là 1 nhánh xử lý ngắt trong hàm Lin_SCI_TxRxInterruptHandler().

- **Input:** u8Channel (địa chỉ kênh vật lý), u8LogicChannel (địa chỉ kênh logic)
- **Output:** void

Các bước thực hiện xem trong UML design hoặc tham chiếu driver.

2.4.3.15 Hàm Lin_SCI_IdleLineDetectInterruptHandler.

Mô tả: Hàm này sẽ xử lý ngắt Idle, thực chất đây là 1 nhánh xử lý ngắt trong hàm Lin_SCI_TxRxInterruptHandler().

- **Input:** u8Channel (địa chỉ kênh vật lý), u8LogicChannel (địa chỉ kênh logic)
- **Output:** void

Bước 1. Xóa cờ ngắt CR2_ILIE.

Bước 2. Nếu state hiện tại của kênh SCI là sleep thì xóa bit CR1_SCISWAI.

Bước 3. Cấu hình có support wakeup, gọi hàm EcuM_CheckWakeup và chuyển các biến sang trạng thái hoạt động.

2.4.3.16 Hàm Lin_SCI_ErrorInterruptHandler.

Mô tả: Hàm này sẽ xử lý tất cả các lỗi trên kênh định tuyến.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void

Các bước thực hiện xem trong UML design hoặc tham chiếu driver.

2.4.4 IP Vault LPUART.

Phần cứng SCI và LPUART tương tự nhau, vì vậy chúng ta có thể tham chiếu cách thức thực hiện trên IP Vault SCI.

2.4.4.1 Hàm Lin_LPUART_InitChannel.

Mô tả: Hàm này khởi tạo tất cả các thanh ghi phần cứng cần thiết để bắt đầu chức năng LIN trên kênh đã chọn.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void

Bước 1. Reset các thanh ghi LPUART, set sau đó xóa bit GLOBAL_RST.

Bước 2. Vô hiệu hóa nhận và truyền, xóa 2 bit CTRL_RE và CTRL_TE.

Bước 3. Cấu hình trường break = 13 bit: xóa bit BAUD_SBNS, CTRL_M, CTRL_M7 và set bit STAT_BRK13.

Bước 4. Thiết lập chế độ lấy 16 mẫu 1 bit trong quá trình nhận dữ liệu, ghi giá trị 01111b vào trường BAUD_OSR.

Bước 5. Cấu hình baudrate cho kênh bằng cách ghi giá trị từ con trỏ Lin_pcConfigPtr vào trường BAUD_SBR.

Bước 6. Thiết lập kiểu truyền dữ liệu LSB, xóa bit STAT_MSBF.

Bước 7. Không check lỗi parity, xóa bit CTRL_PE.

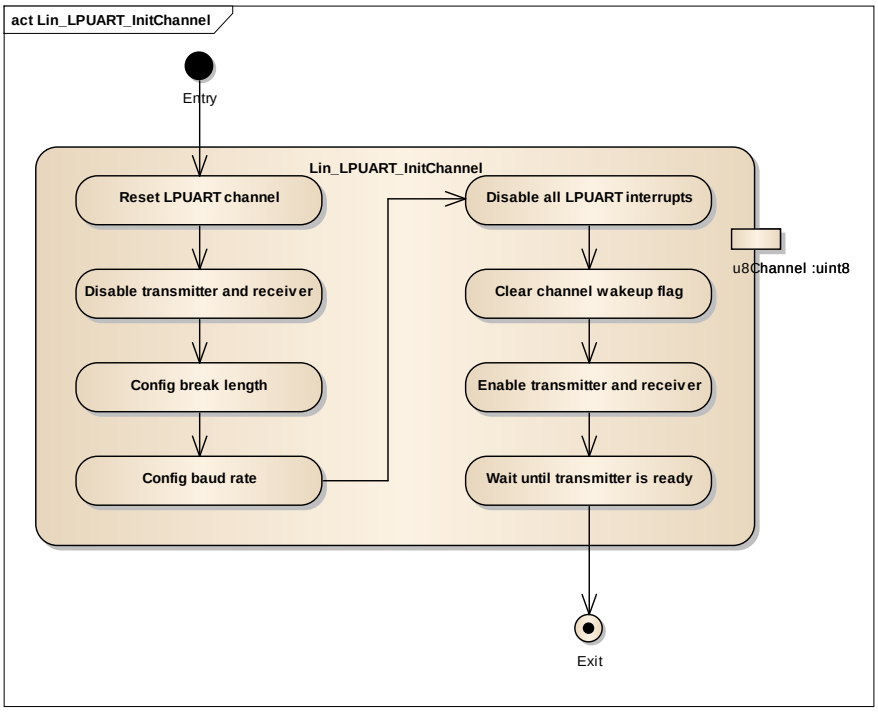
Bước 8. Xóa tất cả các cờ status LPUART_STAT.

Bước 9. Vô hiệu tất cả các ngắt, set các bit (macro LPUART_CTRL_IE_FLAGS_MASK_U32) trong thanh ghi LPUART_CTRL.

Bước 10. Reset cờ wakeup au8LPUART_WakeupFlag.

Bước 11. Cho phép chức năng nhận, truyền 2 set bit CTRL_TE, CTRL_RE

Bước 12. Chờ phần cứng sẵn sàng hoạt động nếu 2 bit STAT_TDRE STAT_TC được set.



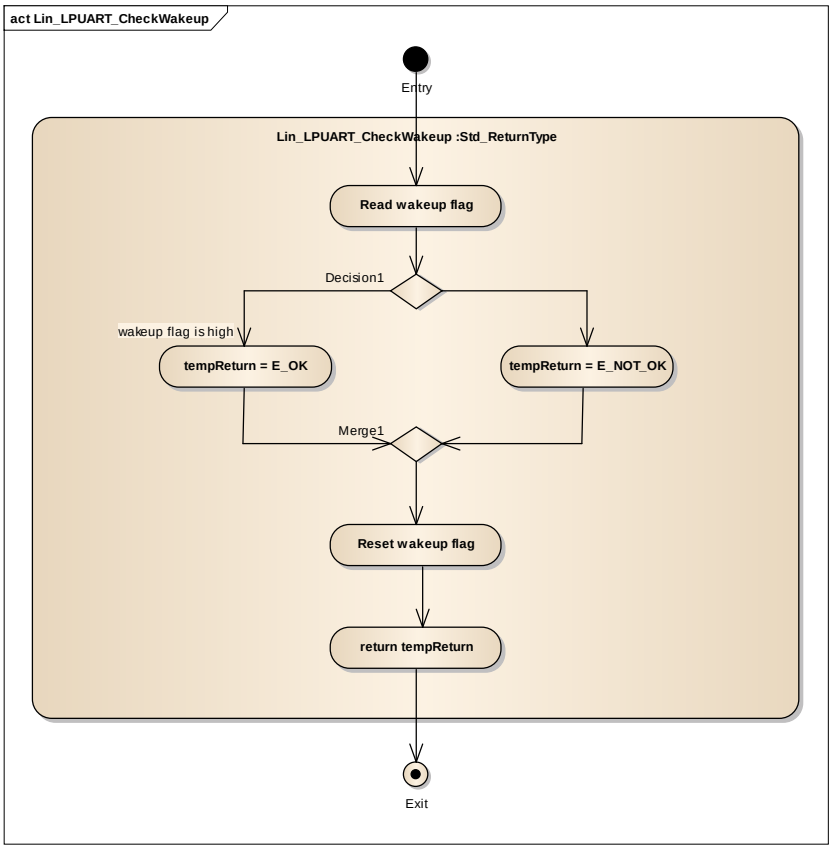
2.4.4.2 Hàm Lin_LPUART_CheckWakeup.

Mô tả: Hàm này xác định nếu kênh LIN được wakeup bởi đường truyền bus LIN. API này được sử dụng khi chức năng wakeup kênh LIN bị vô hiệu hóa (ngắt wakeup bị tắt). Nó kiểm tra cờ wakeup từ kênh LIN trong chế độ ngủ và có chức năng wakeup bị tắt.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void

Bước 1. Đọc biến cờ wakeup: au8LPUART_WakeupFlag[].

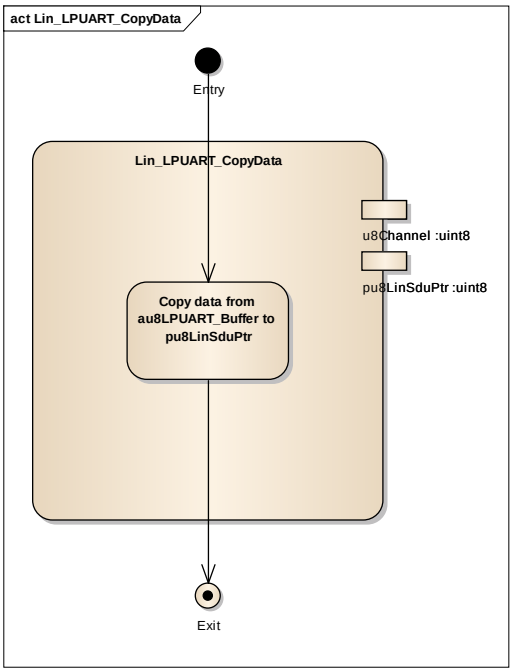
Bước 2. Reset giá trị biến này nếu đang được set.



2.4.4.3 Hàm Lin_LPUART_CopyData.

Mô tả: Hàm này sao chép dữ liệu nhận được trong bộ đệm được cung cấp bởi tham số con trỏ pu8LinSduPtr.

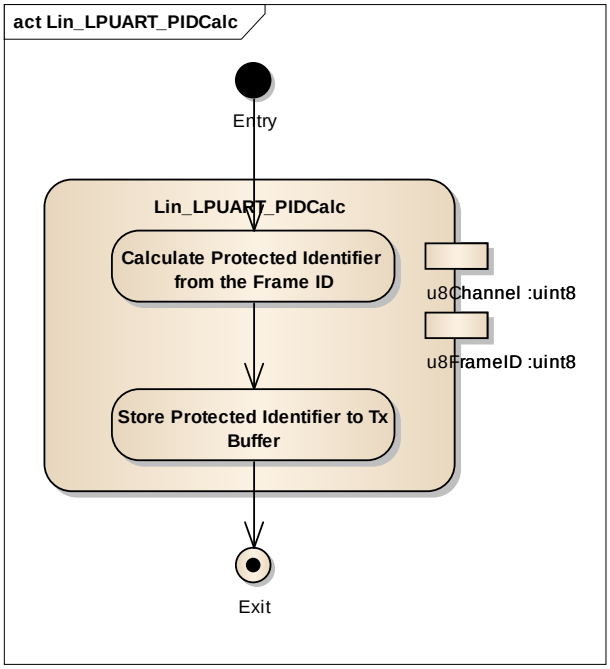
- **Input:** u8Channel (địa chỉ kênh vật lý), con trỏ pu8LinSduPtr trỏ đến buffer hoặc bộ nhớ phần cứng SCI nơi có SDU được lưu trữ.
- **Output:** void



2.4.4.4 Hàm Lin_LPUART_PIDCalc.

Mô tả: Phần cứng LPUART không hỗ trợ tính tự động trường PID, vì vậy chúng ta cần xây dựng một hàm mềm để thực hiện việc này. Hàm này tính giá trị byte PID từ ID khung truyền (gồm 6 bit D0 -> D5).

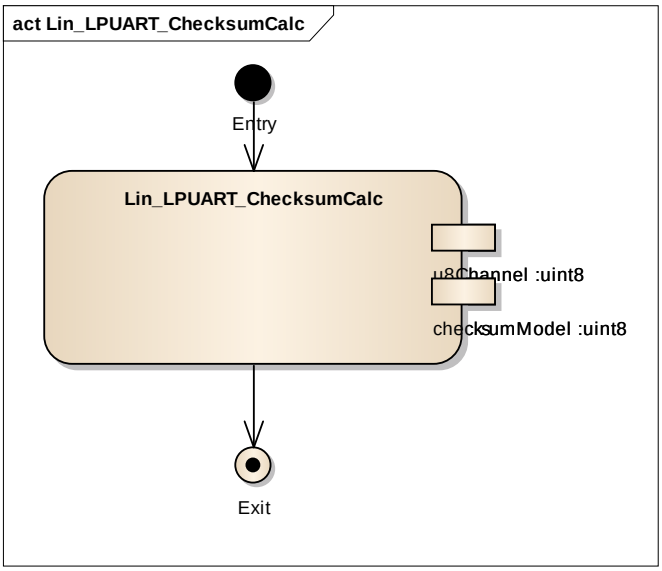
- **Input:** u8Channel (địa chỉ kênh vật lý), ID khung truyền.
- **Output:** void



2.4.4.5 Hàm Lin_LPUART_ChecksumCalc.

Mô tả: Phần cứng LPUART không hỗ trợ tính tự động trường checksum, vì vậy chúng ta cần xây dựng một hàm mềm để thực hiện việc này.

- **Input:** u8Channel (địa chỉ kênh vật lý), checksumModel (kiểu checksum).
- **Output:** u8Checksum



2.4.4.6 Hàm Lin_LPUART_SendHeader.

Mô tả: Khởi tạo truyền phần tiêu đề (header) của khung truyền LIN trên Bus bằng cách sử dụng thông tin cấu hình được lưu trữ trên con trỏ pPduInfoPtr. Nếu frame truyền là loại MASTER_RESPONSE thì phần tiêu đề này sẽ không truyền lên Bus mà toàn bộ frame (bao gồm cả tiêu đề) sẽ được truyền trong hàm Lin_LINFlex_SendReponse.

- **Input:** u8Channel (địa chỉ kênh vật lý), pPduInfoPtr (Con trỏ đến PDU chứa PID, mô hình Checksum, loại phản hồi Master/ Slave response, con trỏ dữ liệu Dl và SDU)
- **Output:** Trạng thái Std_ReturnType truyền phần tiêu đề thành công hay không?

Bước 1. Dừng mọi hoạt động truyền nhận đang xảy ra, xóa 2 bit CTRL_RIE, CTRL_TIE.

Bước 2. Vô hiệu hóa ngắt CTRL_TCIE.

Bước 3. Chờ hủy bỏ khung truyền hiện tại với 2 bit STAT_TC và STAT_TDRE được set.

Bước 4. Hủy bỏ dữ liệu trong buffer nhận, đọc thanh ghi LPUART_DATA.

Bước 5. Reset các trạng thái lỗi: Lin_au8LinChFrameErrorStatus.

Bước 6. Update trạng thái hoạt động của frame truyền: Lin_au8LinChFrameStatus.

Bước 7. Update biến lưu trữ độ dài khung dữ liệu: au8LPUART_BufferLength[].

Bước 8. Ghi lần lượt giá trị byte Sync và PID vào mảng 2 chiều au8LPUART_Buffer[][].

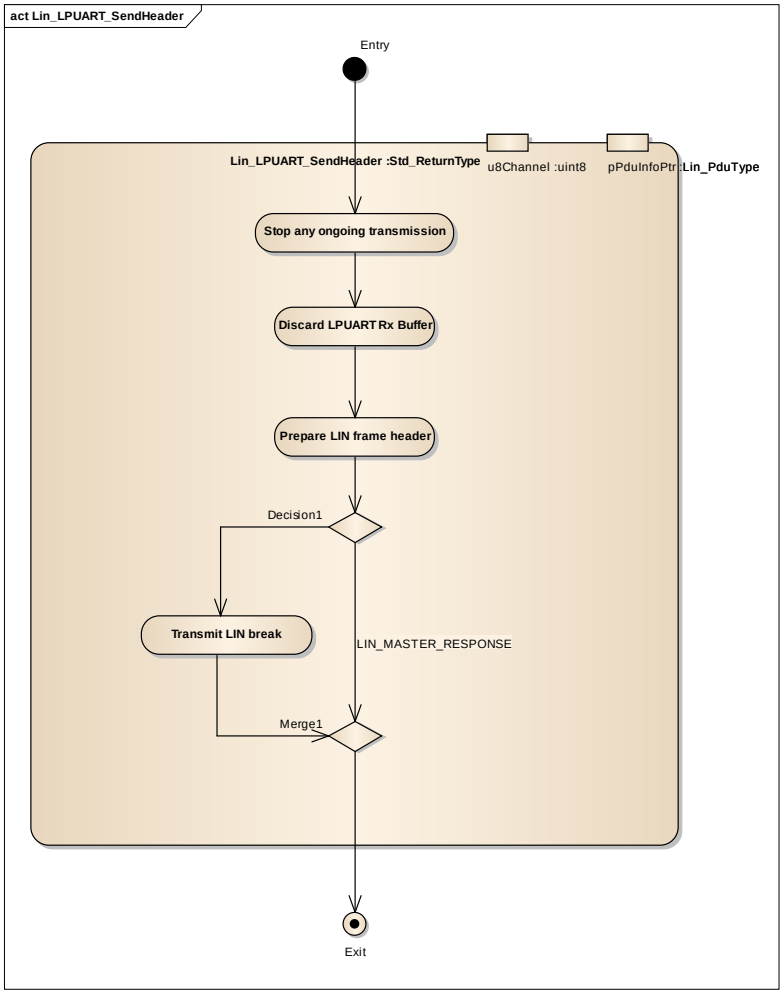
Bước 9. Nếu khung truyền là dạng Slave response thì đặt byte dữ liệu truyền tiếp theo là Sync: au8LPUART_BufferPtr.

Bước 10. Vô hiệu hóa các ngắt lỗi, ghi macro LPUART_CTRL_ONFIE_DISABLE_U32 vào thanh ghi LPUART_CTRL.

Bước 11. Hủy bỏ dữ liệu trong buffer nhận, đọc thanh ghi LPUART_DATA.

Bước 12. Truyền trường break bằng cách lần lượt bật và tắt bit DATA_FRETSC.

Bước 13. Cho phép ngắt truyền hoàn thành hoạt động, set bit CTRL_TCIE.



2.4.4.7 Hàm Lin_LPUART_SendResponse.

Mô tả: Khởi tạo truyền phần dữ liệu (response) của khung truyền LIN trên Kênh bằng cách sử dụng thông tin được lưu trữ trên con trỏ pPduInfoPtr.

- **Input:** u8Channel (địa chỉ kênh vật lý), pPduInfoPtr (Con trỏ đến PDU chứa PID, mô hình Checksum, loại phản hồi Master/ Slave response, con trỏ dữ liệu Dl và SDU)
- **Output:** Trạng thái Std_ReturnType truyền phần tiêu đề thành công hay không?

Bước 1. Ghi trường dữ liệu và byte checksum vào mảng 2 chiều au8LPUART_Buffer[][].

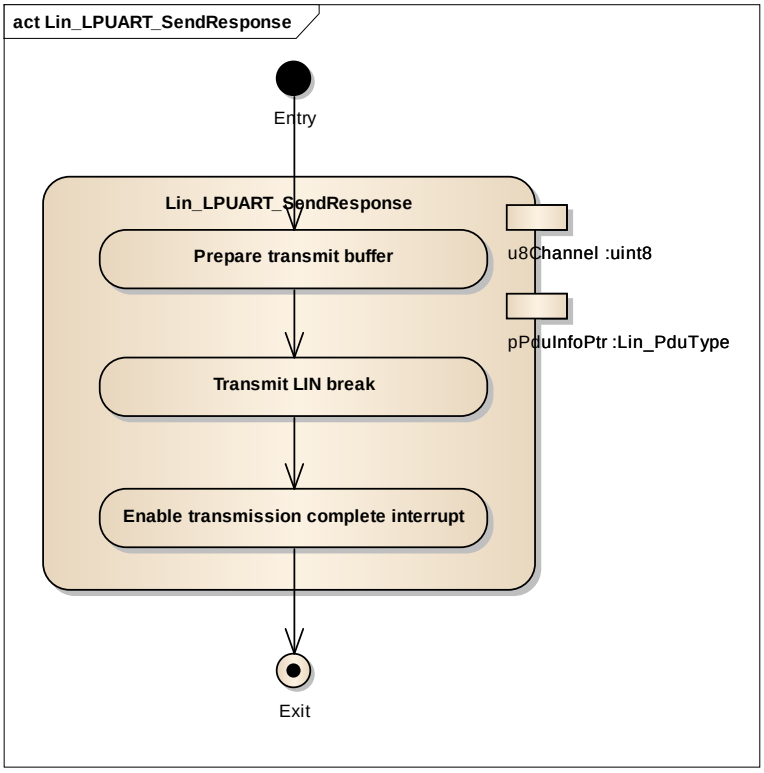
Bước 2. Đặt byte dữ liệu truyền tiếp theo là byte Sync: au8LPUART_BufferPtr

Bước 3. Vô hiệu hóa các ngắt lỗi bằng cách ghi macro LPUART_CTRL_ONFIE_DISABLE_U32 vào thanh ghi LPUART_CTRL.

Bước 4. Hủy bỏ dữ liệu trong buffer nhận, đọc thanh ghi LPUART_DATA.

Bước 5. Truyền trường break bằng cách lần lượt bật và tắt bit DATA_FRETSC.

Bước 6. Cho phép ngắt truyền hoàn thành hoạt động, set bit CTRL_TCIE.



2.4.4.8 Hàm Lin_ LPUART_GoToSleep.

Mô tả: Hàm này dừng bất kỳ quá trình truyền nhận nào đang diễn ra và bắt đầu truyền lệnh ngủ với khung truyền: ID = 0x3C và dữ liệu = (0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF).

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** Trạng thái Std_ReturnType return khung truyền có thành công hay không?

Bước 1. Dừng mọi hoạt động truyền đang xảy ra, xóa 2 bit CTRL_TIE, CTRL_TCIE.

Bước 2. Chờ hủy bỏ khung truyền hiện tại với 2 bit STAT_TC và STAT_TDRE được set.

Bước 3. Hủy bỏ dữ liệu trong buffer nhận, đọc thanh ghi LPUART_DATA.

Bước 4. Ghi giá trị độ dài khung truyền (11 byte) vào biến au8LPUART_BufferLength[] do đã biết trước khung truyền lệnh sleep có 8 byte dữ liệu.

Bước 5. Ghi lần lượt trường sync, PID, dữ liệu, byte checksum vào mảng 2 chiều au8LPUART_Buffer[][].

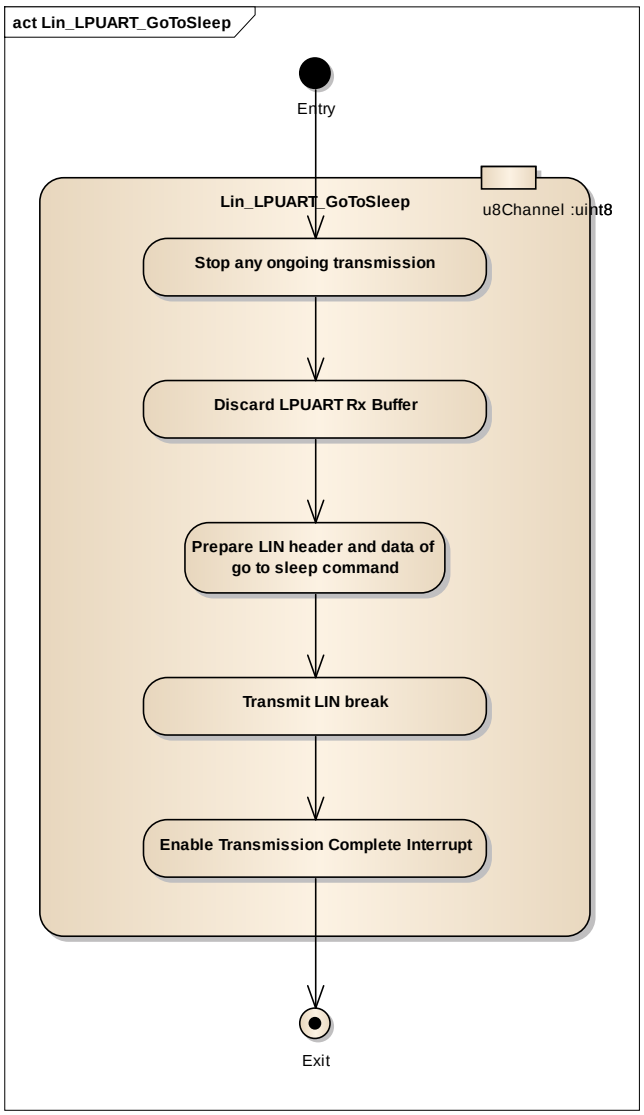
Bước 6. Vô hiệu hóa các ngắt lỗi, ghi macro LPUART_CTRL_ONFIE_DISABLE_U32 vào thanh ghi LPUART_CTRL.

Bước 7. Hủy bỏ dữ liệu trong buffer nhận, đọc thanh ghi LPUART_DATA.

Bước 8. Truyền trường break bằng cách lần lượt bật và tắt bit DATA_FRETSC.

Bước 7. Thiết lập: Lin_au8TransmitHeaderCommand[u8Channel] = LIN_TX_SLEEP_COMMAND.

Bước 8. Cho phép ngắt truyền hoàn thành hoạt động, set bit CTRL_TCIE.



2.4.4.9 Hàm Lin_ LPUART_GoToSleepInternal.

Mô tả: Hàm này dừng bất kỳ quá trình truyền nào đang diễn ra và đặt Kênh vào chế độ ngủ (sau đó phần cứng LIN đi vào chế độ hoạt động giảm năng lượng).

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** Trạng thái Std_ReturnType return thực hiện hàm có thành công hay không?

Bước 1. Dừng mọi hoạt động truyền đang xảy ra, xóa 2 bit CTRL_TIE, CTRL_TCIE.

Bước 2. Chờ hủy bỏ khung truyền hiện tại với 2 bit STAT_TC và STAT_TDRE được set.

Bước 3. Hủy bỏ dữ liệu trong buffer nhận, đọc thanh ghi LPUART_DATA.

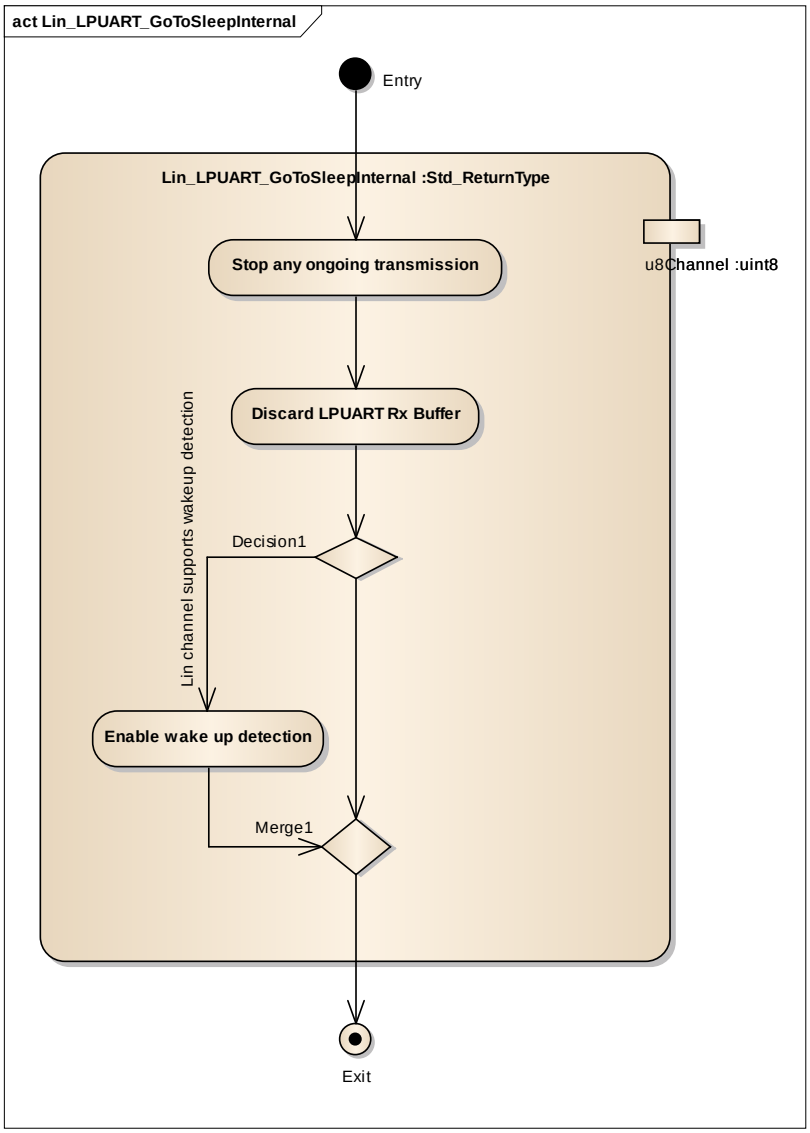
Bước 4. Update trạng thái kênh truyền LIN_CH_SLEEP_STATE.

Bước 5. Reset độ dài khung truyền au8LPUART_BufferLength.

Bước 6. Cho phép phần cứng SCI detect được yêu cầu wakeup bằng cách xóa cờ ngắt STAT_RXEDGIF và set bit BAUD_RXEDGIE.

Bước 7. Vô hiệu hóa ngắt nhận, xóa bit CR2_RIE.

Bước 8. Cho phép phần cứng SCI hoạt động hoặc không ở chế độ Wait của vi điều khiển, set hoặc xóa bit CR1_SCISWAI.



2.4.4.10 Hàm Lin_ LPUART_WakeUp.

Mô tả: Hàm này sẽ gửi một tín hiệu wakeup đến đường bus LIN và đặt kênh LIN ở trạng thái hoạt đồn LIN_CH_OPERATIONAL.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void

Bước 1. Tính giá trị của byte wakeup, ghi vào biến u8WakeupByte.

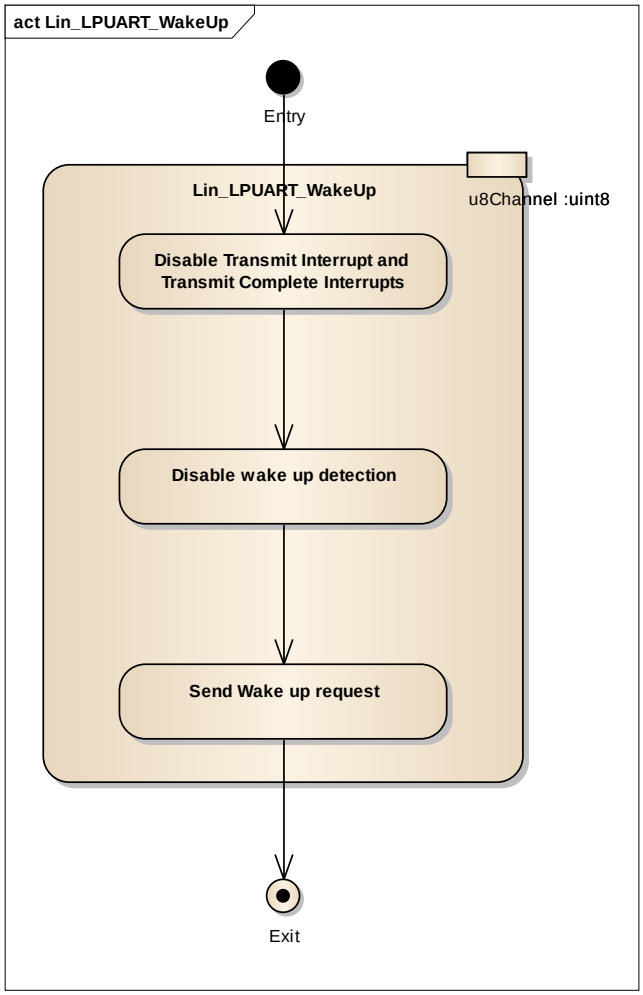
Bước 2. Dừng mọi hoạt động truyền đang xảy ra, xóa 2 bit CTRL_TIE, CTRL_TIE.

Bước 3. Chờ hủy bỏ khung truyền hiện tại với 2 bit STAT_TC và STAT_TDRE được set.

Bước 4. Vô hiệu hóa ngắt detect xung sườn xuống Rx, bằng cách xóa cờ STAT_RXEDGIF và bit BAUD_RXEDGIE.

Bước 5. Vô hiệu hóa ngắt nhận, xóa bit CTRL_RIE.

Bước 6. Truyền xung wakeup bằng cách ghi giá trị byte wakeup u8WakeupByte vào thanh ghi LPUART_DATA.



2.4.4.11 Hàm Lin_ LPUART_WakeupInternal.

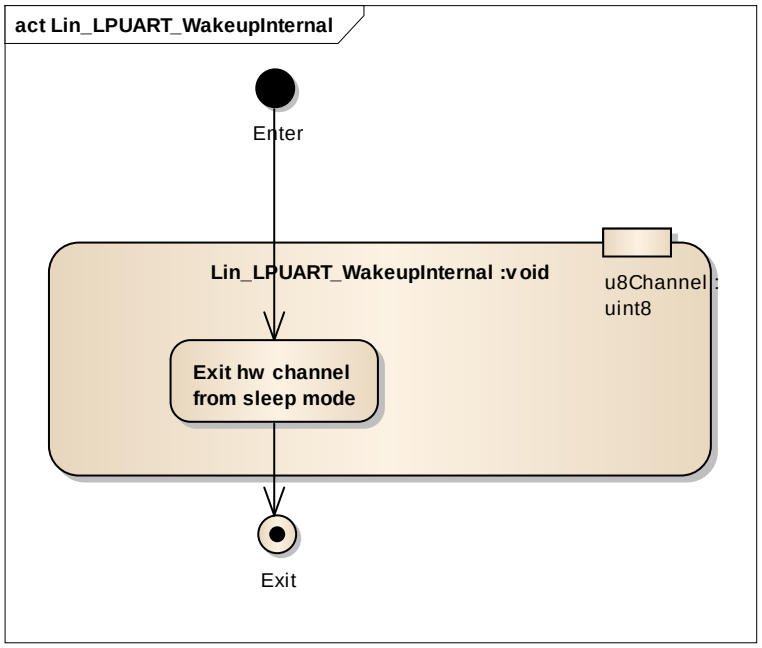
Mô tả: Hàm này sẽ đặt kênh LIN ở trạng thái hoạt động LIN_CH_OPERATIONAL mà không gửi tín hiệu wakeup đến bus LIN.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void

Bước 1. Dừng mọi hoạt động truyền đang xảy ra, xóa 2 bit CTRL_TIE, CTRL_TCIE.

Bước 2. Vô hiệu hóa ngắt detect xung sườn xuống Rx, bằng cách xóa cờ STAT_RXEDGIF và bit BAUD_RXEDGIE.

Bước 3. Vô hiệu hóa ngắt nhận, xóa bit CTRL_RIE.



2.4.4.12 Hàm Lin_ LPUART_HardwareGetStatus.

Mô tả: Hàm này trả về trạng thái của quá trình truyền, nhận khung truyền hoặc chế độ hoạt động hiện tại. Nếu quá trình nhận phần phản ứng Slave đã thành công thì hàm này cung cấp một con trỏ tới bộ đệm, nơi dữ liệu được lưu trữ.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** Con trỏ pu8LinSduPtr trỏ tới bộ đệm hoặc bộ nhớ phần cứng được ánh xạ nơi SDU hiện tại được lưu trữ

2.4.4.13 Hàm Lin_ LPUART_TxRxInterruptHandler.

Mô tả: Hàm này sẽ xử lý tất cả các ngắt RX và TX trên kênh định tuyến.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void

2.4.4.14 Hàm Lin_ LPUART_RxBuffFullInterruptHandler.

Mô tả: Hàm này sẽ xử lý ngắt Rx full buffer.

- **Input:** u8Channel (địa chỉ kênh vật lý), u8LogicChannel (địa chỉ kênh logic)
- **Output:** void

2.4.4.15 Hàm Lin_ LPUART_IdleLineDetectInterruptHandler.

Mô tả: Hàm này sẽ xử lý ngắt Idle.

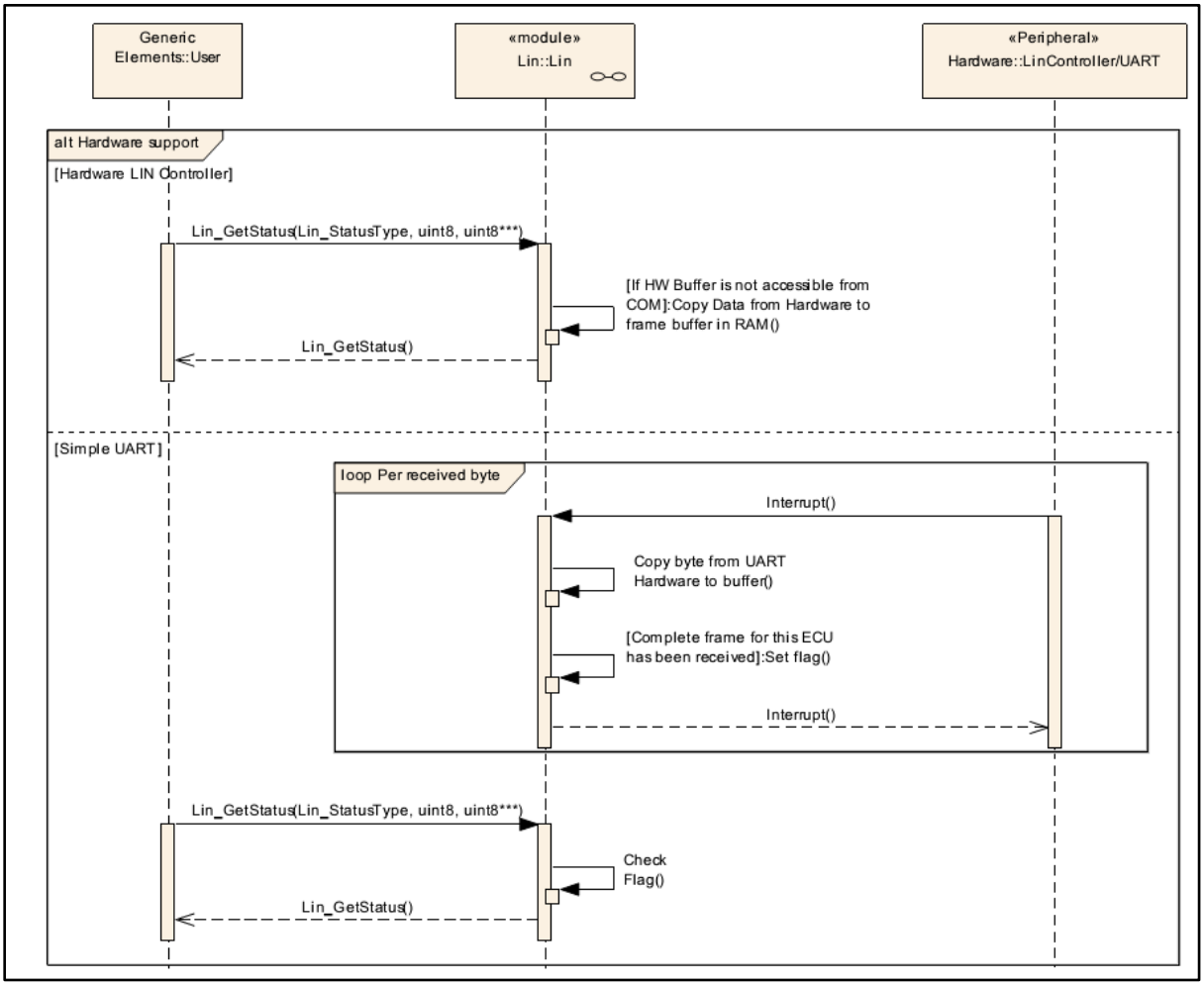
- **Input:** u8Channel (địa chỉ kênh vật lý), u8LogicChannel (địa chỉ kênh logic)
- **Output:** void

2.4.4.16 Hàm Lin_ LPUART_ErrorInterruptHandler.

Mô tả: Hàm này sẽ xử lý tất cả các lỗi trên kênh định tuyến.

- **Input:** u8Channel (địa chỉ kênh vật lý)
- **Output:** void

2.5 Sequence điển hình.



2.6 TIPS &Tricks.

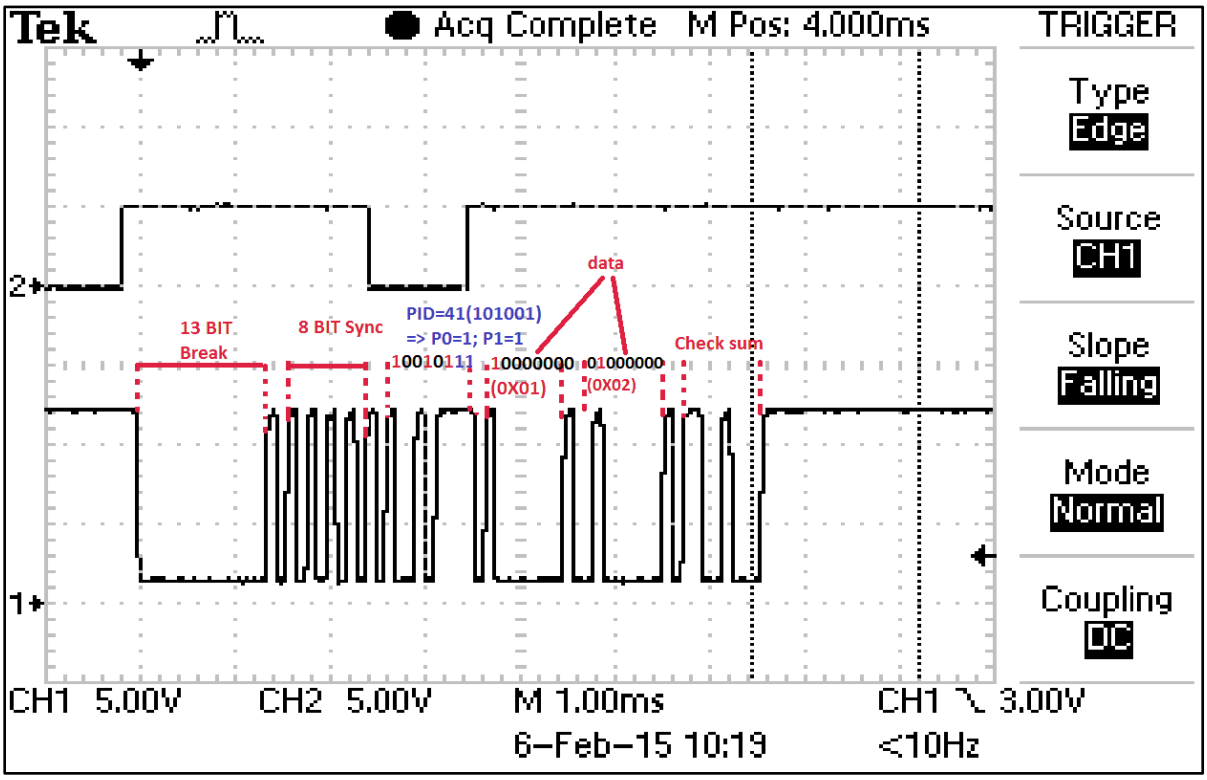
2.6.1 Phân tích khung truyền trên thiết bị đo Oscilloscope.

Khi làm việc với LIN, một trong các kỹ năng cơ bản và quan trọng là chụp dạng khung truyền bằng thiết bị đo oscilloscope hoặc thiết bị Analyzer. Mục đích của việc phân tích khung truyền là:

- Xác định tốc độ truyền trên bus LIN của master/ slave node có chính xác không?
- Lỗi trên khung truyền xuất hiện ở giai đoạn (trường) truyền/ nhận nào?
- Slave node có gửi phần phản hồi response lên bus LIN không?
- Hỗ trợ trong các trường hợp cụ thể khi đang debug…

Lấy ví dụ, một khung truyền có các thông số như bên dưới, lưu ý đọc giá trị các byte trong khung truyền theo thứ tự từ phải sang trái (LBS):

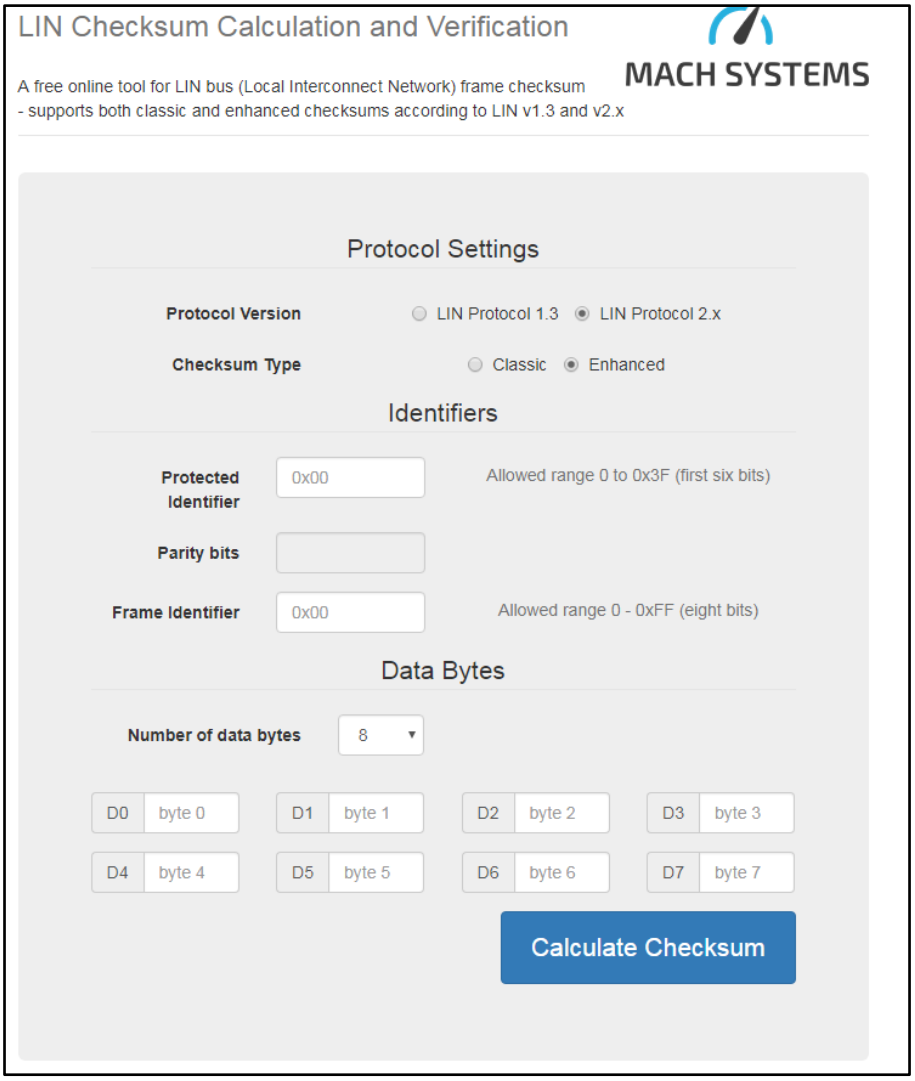
- Tốc độ truyền: 9600 bps. Khi đọc giá trị tốc độ truyền, ta đo giá trị của 1 bit trong trường đồng bộ Sync, sau đó nghịch đảo giá trị.
- Trường Break gồm 13 bit.
- Trường PID có giá trị: 239 (0xE9).
- Trường dữ liệu gồm có 2 byte: 0x01, 0x02.
- Trường checksum kiểu Enhanced, giá trị là: 19 (0x13).



Hình 2.24 Hình ảnh một khung truyền LIN trên thiết bị đo oscilloscope.

Hình 2.25 Hình ảnh khung truyền LIN trên thiết bị đo Analyzer.

- Để thuận tiện trong việc kiểm tra giá trị byte checksum trong khung truyền, ta có thể sử dụng công cụ tính toán trực tuyến theo đường dẫn sau: http://linchecksumcalculator.machsystems.cz/



Hình 2.26 Giao diện trực tuyến tính toán giá trị byte checksum.

2.6.2 Cách tạo các trường hợp lỗi truyền nhận bằng MAF Lite.

Như phần 2.3.1 đã trình bày, trong quá trình truyền nhận frame thường có các lỗi xảy ra như sau: bit error, noise, frame error… Hiện tại, công cụ MAF hỗ trợ tạo ra các lỗi truyền trên bằng cách kéo bus LIN xuống mức tín hiệu thấp.

2.6.3 Đo thời gian thực thi hàm driver.

Khi muốn đo thông số thời gian thực thi (profiling test) của các hàm trong driver trên một số platform không hỗ trợ test profiling như Halo và S32K14X, ta có thể thực hiện bằng các bước sau đây:

Bước 1. Cấu hình một chân GPIO (gọi tắt PinX) PinX ở mức cao.

Bước 2. Đặt hàm set PinX xuống mức thấp ở đầu hàm trong driver.

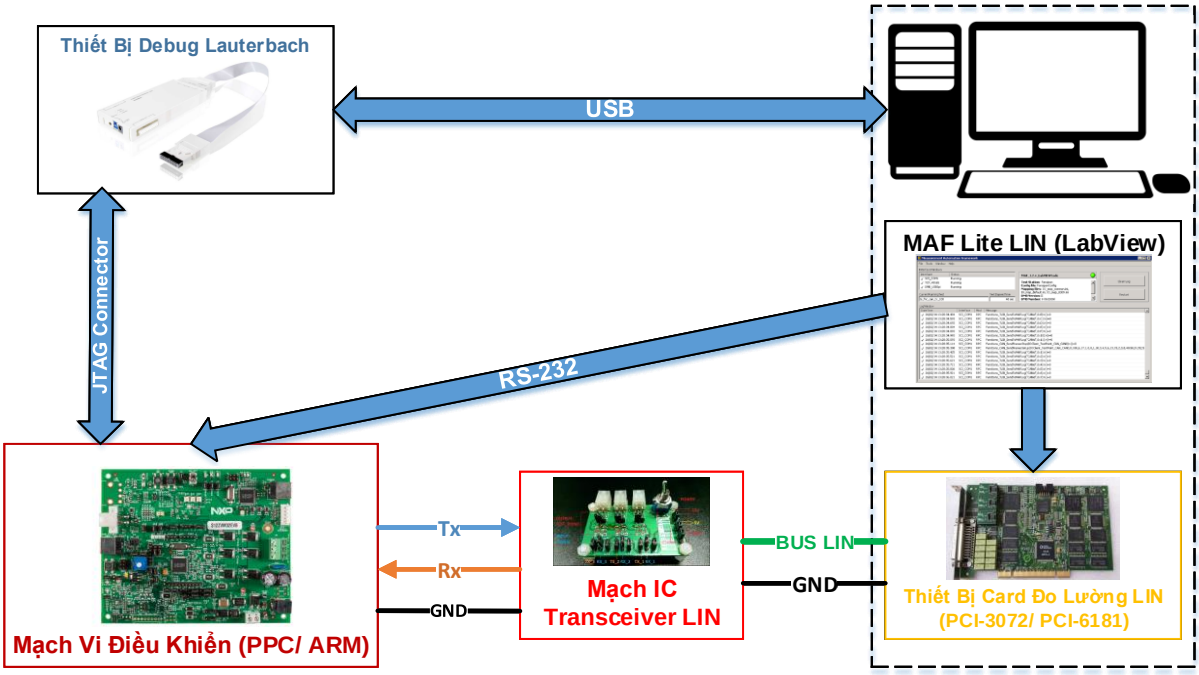
Bước 3. Đặt hàm set PinX lên mức cao ở cuối hàm.

Bước 4. Dùng thiết bị Osciloscope để bắt dạng tín hiệu tại chân PinX.

Bên dưới là ví dụ việc implement trong hàm của driver:

```
REG_WRITE32(PORTX_ADR32, 0xFF);
void function()
{
        /* Pull PinX to low level */
        REG_WRITE32(PORTX_ADR32, 0x00);
        /* Implement code */…
        /* Push PinX to high level */
        REG_WRITE32(PORTX_ADR32, 0xFF);
}
```

2.6.4 Lắp đặt mạch phần cứng.



Hình 2.27 Sơ đồ kết nối phần cứng.

Thực tế, trường hợp IC transceiver LIN trên mạch chính (mother board) hư hỏng thường xảy ra và khó khăn để khò hàn thay thế. Bên cạnh đó, để thuận tiện khi chạy test nhiều kênh một thời điểm, MCAL team đã thiết kế một mạch mini gồm 3 IC transceiver như hình ảnh bên dưới. Các bước cần thiết khi tiến hành lắp đặt mạch phần cứng để chạy test external:

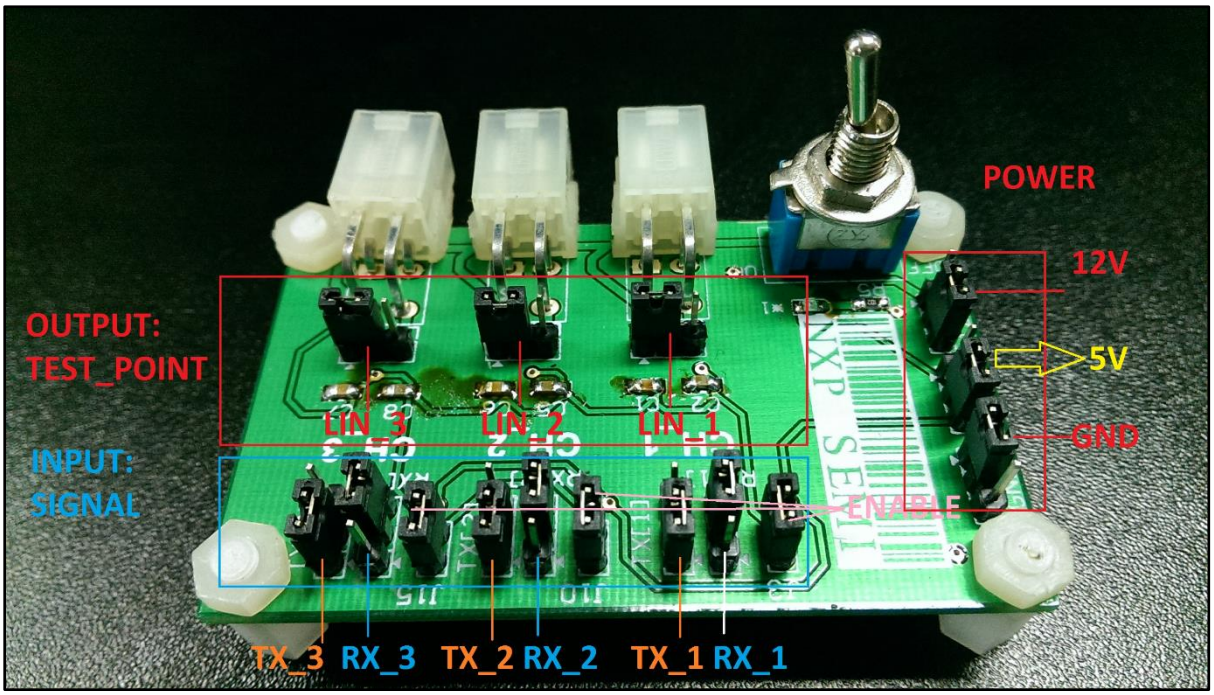
Bước 1. Nối hai chân tín hiệu Tx và Rx của vi điều khiển từ mạch in (tham chiếu file excel wiring của từng platform) đến hai chân Tx, Rx tương ứng trên mạch IC transceiver.

Bước 2. Nối đường bus LIN tương ứng đến các header trên mạch IC transceiver.

Bước 3. Cấp nguồn 12V, chân đất GND cho mạch IC transceiver (lưu ý phải chung đất GND với mother board).

Bước 4. Bật công tắc cấp nguồn cho Lauterbach, mạch vi điều khiển và tiến hành chạy các bài test.

Bên cạnh đó, trên mạch IC transceiver còn có đầu ra điện áp 5V và các chân test point tương ứng với các kênh dùng trong các trường hợp kiểm tra khung truyền bằng Oscilloscope hoặc Analyzer.



Hình 2.28 Hình ảnh mạch IC Tranceiver LIN.

<u>Lưu ý</u>: Stuck at zero timeout error là lỗi hay xảy ra trong quá trình setup mạch chạy test. Nguyên nhân là do chân tín hiệu Rx của vi điều khiển bị nối đất GND, khi xảy ra lỗi này thì ta cần phải kiểm tra:

- IC transceiver đã được cấp nguồn chưa?
- Chân Rx có bị nối tắt với đất không?

2.6.5 Tốc độ baudrate sai.

- **Tốc độ baudrate của kênh LIN bị cấu hình sai:**
- LINFlexD: Hai thanh ghi LINFBRR and LINIBRR.
- LPUART: Hai trường OSR và SBR trong thanh ghi LPUARTx_BAUD.
- eSCI: Thanh ghi BRR.
- SCI: Hai thanh ghi SCIBDH và SCIBDL.
- **Nguồn clock của kênh LIN không đúng, xem lại sys_init() hoặc MCU Driver.**

2.7 Q&A.

2.7.1 Câu hỏi 1.

- **Khung truyền LIN có 1 byte PID, vậy LIN có bao nhiêu kiểu khung truyền (có bao nhiêu giá trị PID có hiệu lực)?**

Trả lời: Như phần 2.1.4 đã trình bày thì PID có 64 giá trị hợp lệ, trong đó có 2 giá trị 63, 64 là reserved. Do đó, lưu ý khi viết test không được gán giá trị PID bất kì vào khung truyền.

2.7.2 Câu hỏi 2.

3. PHỤ LỤC.

[1] Tài liệu tham khảo về bộ test_lin: LIN_Test_Detail.xlsx

[2] Link giới thiệu giao thức LIN: https://elearning.vector.com/vl_lin_introduction_en.html

[3] MPC5777C Reference Manual, Rev. 2, 06/2014

[4] MPC5746C Reference Manual, Rev. 4, 12/2016

[5] MC9S12ZVM Family Reference Manual Rev. 2.11

[6] S32K14x Series Reference Manual, Rev. 3, 03/2017