|  |  |
| --- | --- |
|  | **2014** |
|  | **FPT Software**  **Le Sy Cuong (FSU1.BU16)** |

|  |
| --- |
| **OVERVIEW ABOUT AUTOSAR SPECIFICATION OF LIN DRIVER** |
| For new engineer join to ITG group |

Contents

[1. WHAT IS AUTOSAR? 4](#_Toc404325532)

[1.1 Software architecture 4](#_Toc404325533)

[1.2 Freescale AUTOSAR Software 5](#_Toc404325534)

[1.3 Autosar Specifications 6](#_Toc404325535)

[2. V&V FRAMEWORK: BEART and MAF 6](#_Toc404325536)

[2.1 BEART commands 6](#_Toc404325537)

[2.1.1 Parse (Phân tích cú pháp) 6](#_Toc404325538)

[2.1.2 Generate (Đưa ra kết quả) 7](#_Toc404325539)

[2.1.3 Build 8](#_Toc404325540)

[2.1.4 Execute tests 9](#_Toc404325541)

[2.1.5 Generating test report 12](#_Toc404325542)

[2.2 MAF 14](#_Toc404325543)

[2.2.1 14](#_Toc404325544)

[3. LIN DRIVER 14](#_Toc404325545)

[3.1 Architecture Layer 14](#_Toc404325546)

[3.2 Basic concepts 0](#_Toc404325547)

[3.3 Properties 0](#_Toc404325548)

[3.3.1 Signal manager 0](#_Toc404325549)

[3.3.2 Frame 2](#_Toc404325550)

[3.3.3 Data transport 5](#_Toc404325551)

[3.3.4 Schedule table 6](#_Toc404325552)

[3.4 Task behavior model 7](#_Toc404325553)

[3.4.1 Master task state machine 7](#_Toc404325554)

[3.4.2 Slave task state machine 7](#_Toc404325555)

[3.5 Network management 8](#_Toc404325556)

[3.5.1 Slave communication state diagram 9](#_Toc404325557)

[3.5.2 Wake up 9](#_Toc404325558)

[3.5.3 Go to sleep 10](#_Toc404325559)

[3.6 Status management 11](#_Toc404325560)

[3.6.1 Concept 11](#_Toc404325561)

[3.6.2 Event triggered frames 11](#_Toc404325562)

[3.6.3 Reporting to the cluster 11](#_Toc404325563)

[3.7 Appendices 11](#_Toc404325564)

[3.8 API 12](#_Toc404325565)

[3.8.1 Cấu trúc file code 12](#_Toc404325566)

[3.8.2 Imported types 12](#_Toc404325567)

[3.8.3 Type definitions 13](#_Toc404325568)

[3.8.4 Lin\_Init 15](#_Toc404325569)

[3.8.5 Lin\_CheckWakeup 15](#_Toc404325570)

[3.8.6 Lin\_GetVersionInfo 16](#_Toc404325571)

[3.8.7 Lin\_SendFrame 16](#_Toc404325572)

[3.8.8 LinGoToSleep 17](#_Toc404325573)

[3.8.9 Lin\_GoToSleepInternal 17](#_Toc404325574)

[3.8.10 Lin\_Wakeup 18](#_Toc404325575)

[3.8.11 Lin\_GetStatus 18](#_Toc404325576)

[4. CC, CQ 21](#_Toc404325577)

[4.1 ClearCase 21](#_Toc404325578)

[4.2 CQ? 21](#_Toc404325579)

[5. OVERVIEW OF GROUP OPERATION 21](#_Toc404325580)

[6. FOLDER TREE 22](#_Toc404325581)

# 0. Note

* Các phần mềm cài đặt:
* **T32**: Driver cho DEBUG **Lauterbach.** Dẫn driver cab usb nối với lauterbach tới thư mục t32 trong ổ C.
* Kiểu biến **Volatile**: dùng khi các biến thay đổi giá trị không báo trước, compiler luôn load lại giá trị biến này khi thực hiện lệnh.
* Kiểu biến **Static:** dùng để giữ giá trị của biến sau mỗi lần gọi.
* Kiểu biến **Extern:** tham chiếu của một biến cùng tên nào đó đã được định nghĩa bên ngoài. Biến được tham chiếu phải được khai báo ở cấp độ cao nhất (toàn cục) và có thể nằm trong một file khác.

# 1. WHAT IS AUTOSAR?

**AUTOSAR** (**AUT**omotive **O**pen **S**ystem **AR**chitecture) is an open and [standardized](http://en.wikipedia.org/wiki/Standardized) [automotive software](http://en.wikipedia.org/wiki/Automotive_software) architecture, jointly developed by automobile manufacturers, suppliers and tool developers 🡪 create and establish open standards for automotive E/E (Electrics/Electronics) architectures that will provide a basic [infrastructure](http://en.wikipedia.org/wiki/Infrastructure) to assist with developing [vehicular](http://en.wikipedia.org/wiki/Vehicular) [software](http://en.wikipedia.org/wiki/Software), user interfaces and management for all [application domains](http://en.wikipedia.org/wiki/Application_domain). This includes the standardization of basic systems functions, [scalability](http://en.wikipedia.org/wiki/Scalability) to different [vehicle](http://en.wikipedia.org/wiki/Vehicle) and platform variants, [transferability](http://en.wikipedia.org/wiki/Transferability) throughout the network, [integration](http://en.wikipedia.org/wiki/System_integration) from multiple suppliers, [maintainability](http://en.wikipedia.org/wiki/Maintainability) throughout the entire [product life-cycle](http://en.wikipedia.org/wiki/Product_lifecycle_%28marketing%29) and [software updates](http://en.wikipedia.org/wiki/Software_update) and [upgrades](http://en.wikipedia.org/wiki/Software_upgrade) over the vehicle's lifetime as some of the key goals.

## Software architecture

MCAL (**M**icro**C**ontroller **A**rchitecture **L**ayer) gồm có 6 lớp chính:

* **Microcontroller (MC)**: Phần cứng
* **Microcontroller Abstraction Layer**: (Lớp trừu tượng MC) là lớp phần mềm thấp nhất bao gồm các driver nội, điều khiển truy nhập tới các noại vi nội của MC và các thiết bị nhớ ngoài. Giúp cho lớp phần mềm cao hơn độc lập với MC.
* **ECU Abstraction Layer**: Là driver các giao diện của MCAL, cung cấp API cho truy nhập các ngoại vi và thiết bị ngoại trừ port pins... Giúp các lớp phần mềm trên độc lập với lớp phần cứng.
* **Services Layer**: Lớp cao nhất trong các lớp phần mềm cơ bản.
* **AUTOSAR Runtime Environment (RTE)**: Cung cấp dịch vụ truyền thông cho lớp phần mềm ứng dụng; giúp các thành phần phần mềm AUTOSAR độc lập khi mapping tới ECU.
* **Application Layer**: Lớp ứng dụng điều khiển hoạt động của MC.

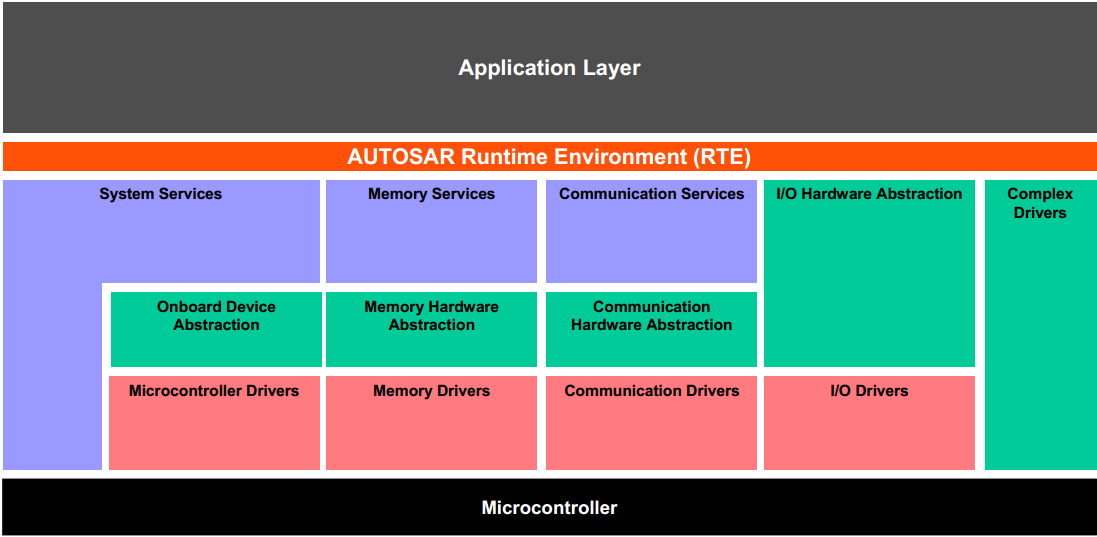


Figure 1.1 MCAL Architecture General

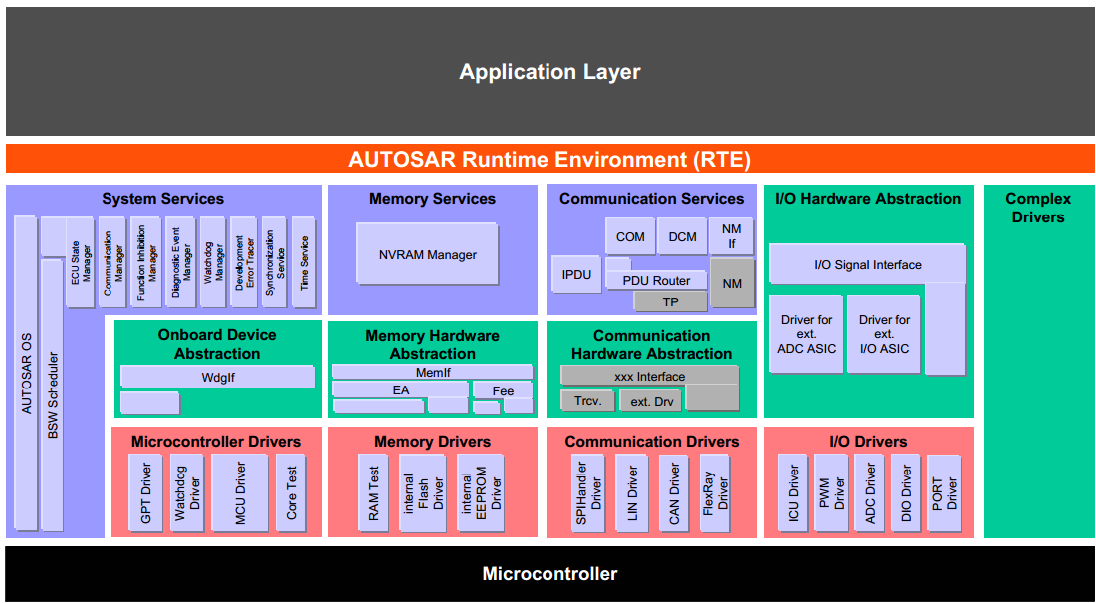


Figure 1.2 MCAL Architecture Detail

## 1.2 Freescale AUTOSAR Software

Trong mỗi lớp phần mềm cơ bản bao gồm có 4 kiểu:

* I/O: Truy nhập chuẩn đến các cảm biến, thi hành và ngoại vi trên mạch ECU.
* Memory: Truy nhập đến các thiết bị nhớ (ngoài, trong).
* Communication: truy nhập đến hệ thống mạng phương tiện, hệ thống truyền thông trong mạch ECU và phần mềm trong ECU.
* Các module phụ thuộc:
* MCU: Phần cứng của LIN phụ thuộc vào clock hệ thống, prescaler và PLL.
* PORT: Cấu hình chân port được sử dụng trong driver LIN kiểu in/out.
* DET (Development Error Tracer): Trong chế độ phát triển, sai sót phát triển báo cáo module LIN trong suốt hàm Det\_ReportError của module DET.
* DEM (diagnostic Event Manager): Báo cáo lỗi sản phẩm module LIN tới DEM.
* OS: LIN driver sử dụng ngắt nên phụ thuộc vào OS, nơi cấu hình nguồn ngắt.
* LIN driver users: Giao diện LIN

## 1.3 Autosar Specifications

# 2. V&V FRAMEWORK: BEART and MAF

## 2.1 BEART commands

Hiểu là một Makefile và các lệnh thực hiện để tạo ra được kết quả mục tiêu. Những lệnh này luôn thực hiện một trong các thao tác:

**- Parse**

**- Generate**

**- Build**

**- Run**

**- Report**

### 2.1.1 Parse (Phân tích cú pháp)

Parse là bước đầu tiên trong vòng lặp lệnh BEART, phân tích cú pháp các file code nguồn và tìm kiếm tất cả các **test case** và **test suites.**

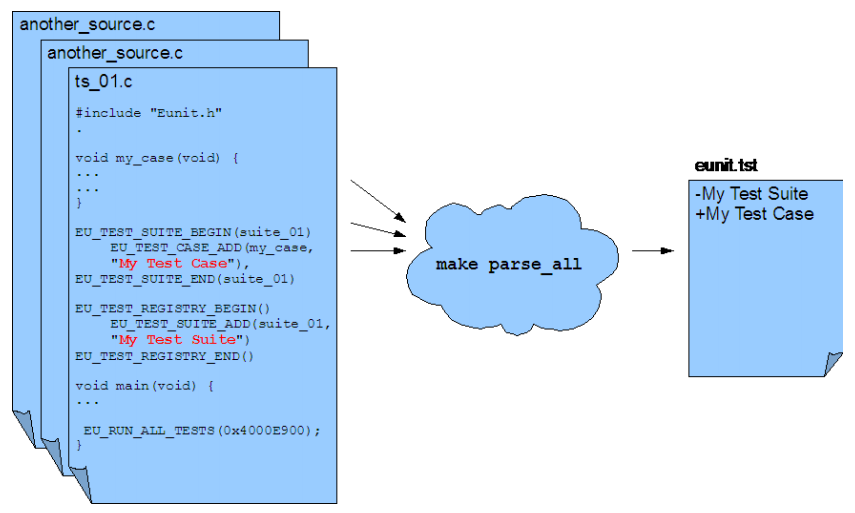


Figure 2.1 Parsing sourse files for test cases

**- Đầu vào:** File nguồn test suit (**\*.c**)

**- Kết quả:** File text (VD: **eunit.tst**) chứa **danh sách** tất cả các **test case** và **test suite**

**- Lệnh:**

|  |
| --- |
| **+ make parse\_<test\_name>** : Phân tích file có tên <test\_name>  **+ make parse\_all** : Phân tích tất cả các file code nguồn |

### 2.1.2 Generate (Đưa ra kết quả)

Bước thực hiện chuẩn theo AUTOSAR , thay thế các tham số trong các file nguồn (\*.c, \*.h).

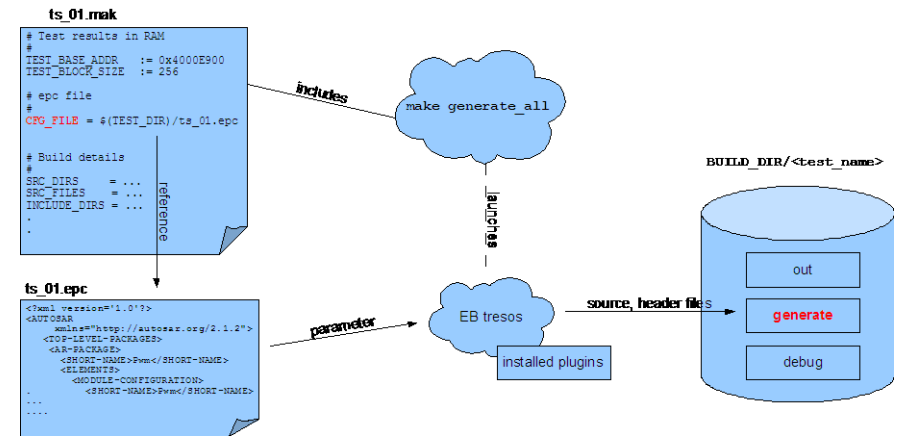


Figure 2.2 Generate step in AR test group

**- Đầu vào:**

+ File lệnh thực hiện: \*.mak (VD: **ts\_01.mak**)

+ File cấu hình AR: \*.epc – Thành phần test (VD: **ts\_01.epc**), lưu trong **drivers/…/test.** Trong file epc chuẩn, có các công cụ: **M4**- thay thế các macro với trường hợp nhiều cấu hình được cho phép hoạt động; **SED** – thay thế id plugin…

+ File định nghĩa: \*.xdm – thành phần của plugin (VD: EB tresos)

**- Kết quả:**

+ Các file nguồn **\*.c, \*.h** đã generate trong thư mục: **BUILD\_DIR/<test\_name>**

**- Lệnh:**

|  |
| --- |
| + **make generate\_(all|<test\_name>)** : Thay thế các thông số trong code nguồn |

### 2.1.3 Build

Kết quả đầu ra của bước này là file test để load vào trong các thiết bị mục tiêu.

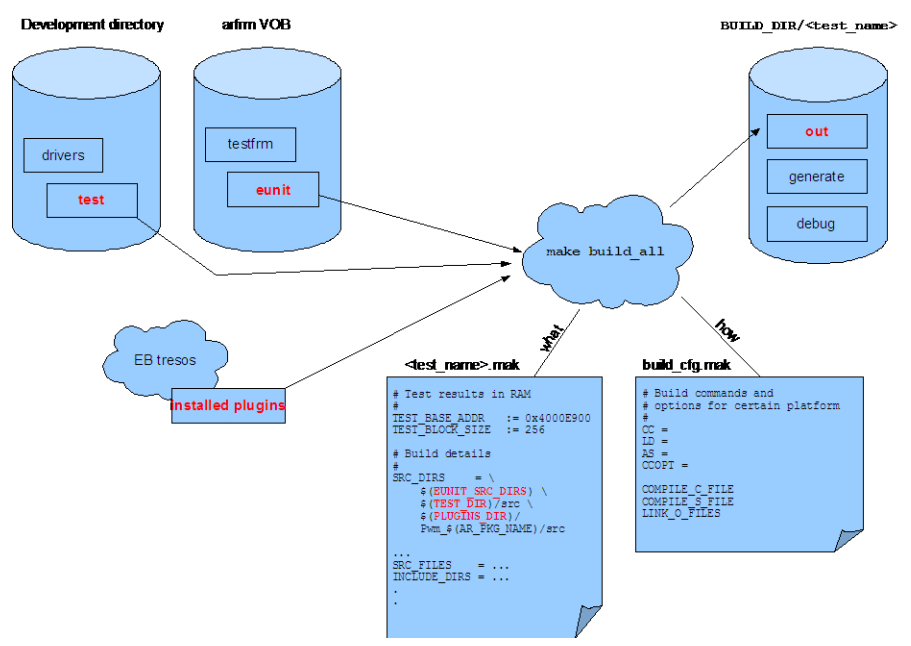


Figure 2.3 Building step application from commonly used sources

**- Đâu vào:**

+ Development directory (test)

+ Arfrm VOB (eunit)

+ EB tresos (Đã cài các plugin)

+ <test\_name>.mak: Các link file nguồn, code

+ build\_cfg.mak: cách thực thi

**- Kết quả:** trong mục out, đuôi **\*.elf**

**- Lệnh:**

|  |
| --- |
| + **make build\_(all|<test\_name>)** : Chạy build code nguồn |

### 2.1.4 Execute tests

Nạp chương trình sau build vào trong thiết bị mục tiêu,

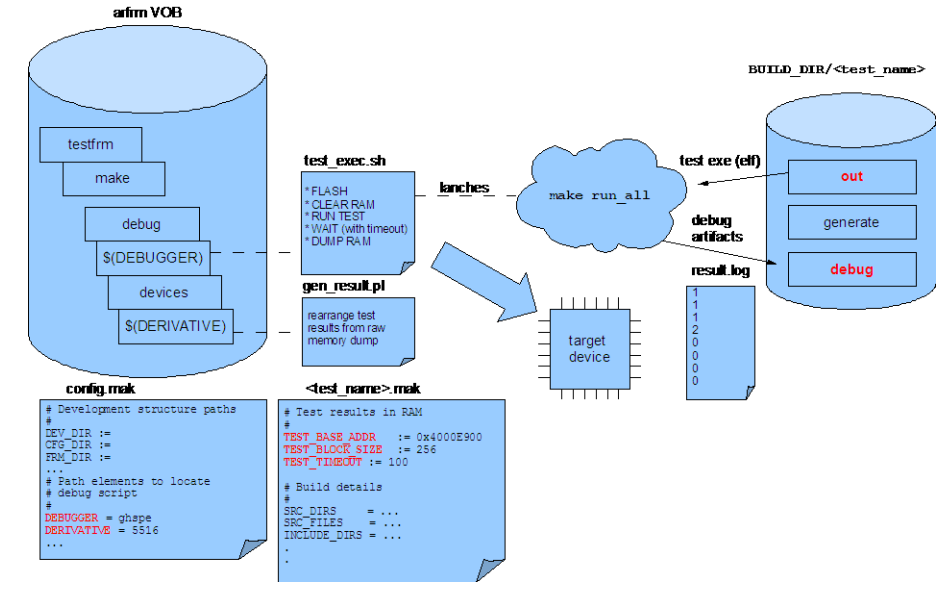


Figure 2.4 Executing test application on emedded target

**- Đầu vào:**

+ **test\_exec.sh; gen\_result.pl**

+ **\*.elf**: file sau build

+ **config.mak, <test\_name>.mak:** link, cách thực thi

**- Đầu ra:**

+ File kết quả: **result.log**

* **Execution flow:**

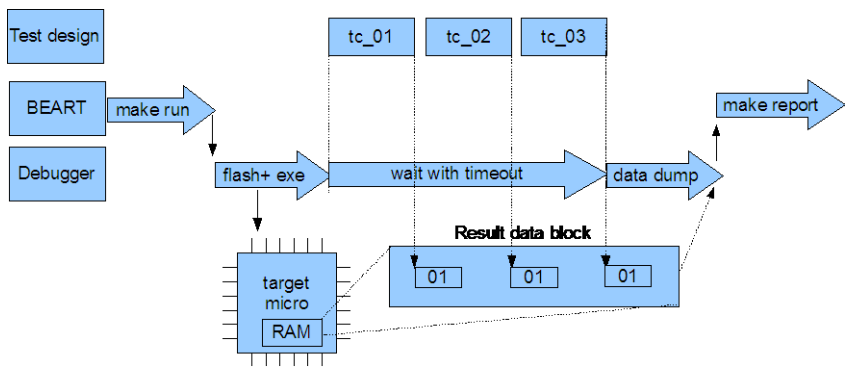


Figure 2.5 Execution flow in normal EUnit mode

Trong hành vi EUnit chuẩn, TS và TC được lập lịch thực hiện lần lượt trong thiết bị và kết quả test được lưu trong RAM của thiết bị. Dấu hiệu chính khi kết thúc quá trình này là SW breakpoint.

* Execution flow for external storage:

Sử dụng khi muốn lưu trữ dữ liệu kết quả test ở thiết bị ngoài, cần kết nối thiết bị test và thiết bị lưu trữ ngoài trước khi lên lịch TS và TC.

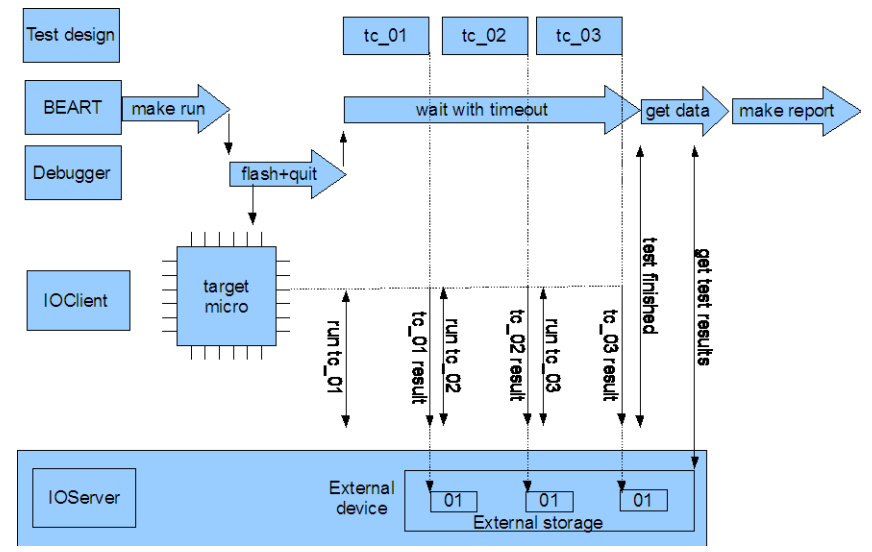


Figure 2.6 Execution floow in external strage EUnit mode

### 2.1.5 Generating test report

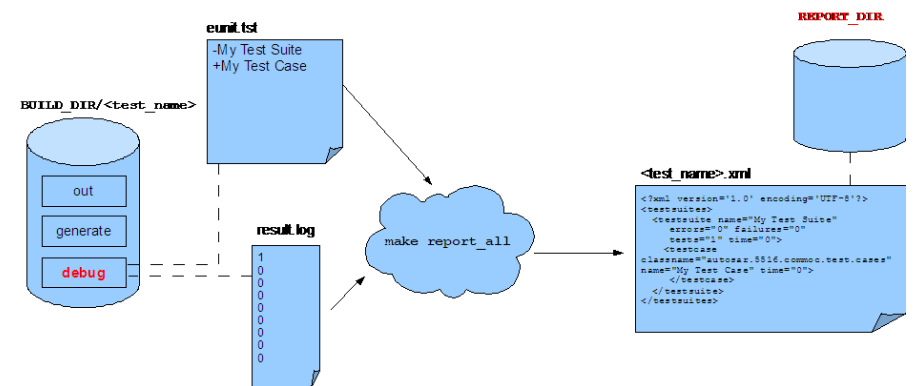


Figure 2.7 Gerarating test report

**- Đầu vào:**

+ **eunit.tst** (Parse)

+ **result.log** (sau build)

**- Đầu ra:**

+ File kết quả: <**test\_name>.xml**

## 2.2 MAF

### 2.2.1

# 3. LIN DRIVER

## 3.1 Architecture Layer

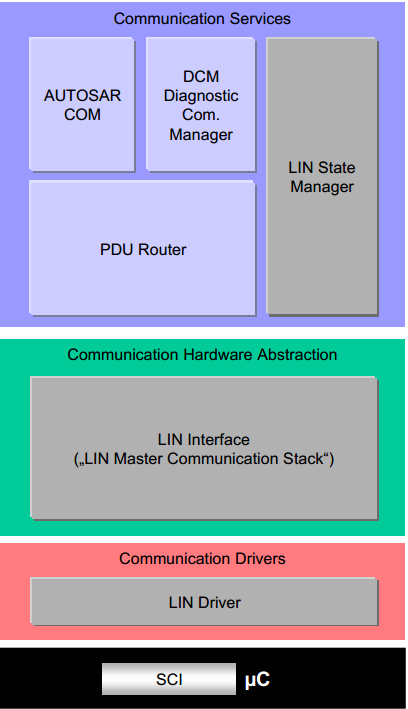


Figure 5.1 LIN Communication Architecture

LIN (Local Interconnect Network) cung cấp giao diện cố định tới mạng LIN, ẩn giao thức và đặc tính bản tin từ ứng dụng.

* LIN 2.0 phục vụ hàng đợi truyền thông với:
  + Quản lý bảng lịch trình cho khung truyền LIN và thao tác yêu cầu để chuyển đổi tới các bảng lịch khác.
  + Giao thức chuyển tin.
  + Có giao diện Wakeup và Sleep.
* Đặc điểm cơ bản:
  + Thực hiện giao thức LIN và thích nghi phần cứng riêng biệt.
  + Hỗ trợ cả UART và LIN.

LIN (mạng kết nối cục bộ) là giao thức mạng nối tiếp sử dụng để truyền thông giữa các thành phần trên phương tiện (ô tô). Mạng quảng bá LIN bao gồm một master và có thể lên đến 16 slave, tất cả các bản tin được khở tạo bởi master với nhiều nhất một slave trả lời với một bản tin xác nhận. Vì vậy, không có các xung đột xảy ra khi truyền thông giữa các master và slave.

## 3.2 Basic concepts

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Acronym** | **Desciption** | **Mô tả** |
|  | AUTOSAR | Automotive Open System Arrchitecture | Kiến trúc hệ thống mở tự động của ô tô |
|  | COM | Communication | Truyền thông |
|  | ECU | Electronic Control Unit | Đơn vị điều khiển điện tử |
|  | EcuM | ECU Manager | Người quản lý ECU |
|  | DEM | Diagnostic Event Manager | Người quản lý sự kiện chẩn đoán |
|  | DET | Development Error Tracer | Người phát hiện lỗi phát triển |
|  | ISR | Interrupt Service Routine | Thủ tục phục vụ ngắt |
|  | LIN | Local Interconnect Network | Mạng kết nối nội bộ cục bộ |
|  | MCAL | MicroController Astraction Layer | Lớp mô tả vi điều khiển |
|  | MCU | MicroController Unit | Đơn vị vi điều khiển |
|  | OS | Operating System | Hệ thống hoạt động |
|  | PDU | Protocol Data Unit. Consists of Identifier, data length and Data (SDU) | Giao thức dữ liệu đơn vị, bao gồm định dạng, chiều dài dữ liệu và dữ liệu |
|  | PID | Protected ID | Địa chỉ bảo vệ |
|  | PLL | Phase-Locked Loop | Vòng khóa pha |
|  | RX | Receiver | Bộ nhận |
|  | SCI | Serial Communication Interface | Giao diện truyền thông nối tiếp |
|  | SDU | Service Data unit | Đơn vị dữ liệu phục vụ |
|  | SFR | Special Function Register | Thanh ghi chức năng đặc biệt |
|  | SPAL | Standard Peripheral Abstraction Layer | Lớp trừu tượng ngoại vi chuẩn |
|  | SRS | Software Requirement Specification | Đặc điểm kỹ thuật yêu cầu phần mềm |
|  | SWS | Software Specificetion | Đặc điểm kỹ thuật phần mềm |
|  | TP | Transport Layer | Lớp chuyển phát |
|  | TX | Transmission | Sự truyền |
|  | UART | Universal Asynchronous Receiver Transmitter | Truyền nhận không đồng bộ phổ biến |
|  | XML | Extensible Markup Language | Ngôn ngữ đánh dấu bằng thẻ có thể mở rộng |
|  |  |  |  |

## 3.3 Properties

Tài liệu đặc điểm kỹ thuật giao thức LIN đề cập đến hành vi trong bus và trong các node. Một node (mặc định là master) sẽ kết nối đến hơn một bus LIN phải được thao tác bởi lớp cao hơn.

### 3.3.1 Signal manager

Tín hiệu được truyền tải trong trường dữ liệu của các khung truyền.

* **Kiểu tín hiệu**: Một tín hiệu có một trong hai kiểu là giá trị vô hướng hoặc mảng byte:
* Giá trị vô hướng: có độ dài từ 1 ÷ 16 bit; tín hiệu vô hướng 1 bit được gọi là kiểu boolean, tín hiệu có kích thước từ 2 bit trở lên thì là dạng số nguyên không dấu.
* Một mảng byte: mảng từ 1 ÷ 8 byte.

Mỗi tín hiệu có chính xác một publissher, các node khác có thể theo dõi tín hiệu này. Tất cả các tín hiệu này đều có giá trị khởi tạo, giá trị khởi tạo này có hiệu lực cho đến khi node ghi một giá trị mới tới chính publisher signal. Gía trị khởi tạo cho tín hiệu theo dõi có hiệu lực đến khi một giá trị cập nhật được nhận từ một node khác.

* **Tính nhất quán**: Tín hiệu vô hướng ghi hoặc đọc phải hoạt động atomic (nguyên tử) …
* **Đóng gói tín hiệu**: Tín hiệu được truyền đầu tiên với LSB ([Least Significant Bit](http://en.wikipedia.org/wiki/Least_significant_bit)) và MSB ([Most Significant Bit](http://en.wikipedia.org/wiki/Least_significant_bit)) cuối cùng. Không hạn chế trong việc đóng gói tín hiệu vô hướng quá hạn byte. Mỗi byte trong một mảng byte được map tới một byte khung đơn bắt đầu với byte dữ liệu số thấp nhất. Nhiều tín hiệu có thể đóng gói trong một frame miễn là không chồng lên nhau.Các tín hiệu sẽ thực thi hiệu quả hơn nếu kích thước không lớn hơn hạn mức.
* Tín hiệu được đóng gói trong nhiều frame miễn là publisher của tín hiệu giống nhau. Nếu node nhận được một tín hiệu đóng gói trong nhiều frame thì giá trị tín hiệu cuối cùng có hiệu lực.
* **Truyền/ nhận tín hiệu**: Thời điểm truyền nhận tín hiệu cần được định nghĩa để giúp công cụ thiết kế và test có thể phân tích tín hiệu. Điều này có nghĩa rằng tất cả các thực thi sẽ làm như ta dự kiến.
* **Nhận tín hiệu:**

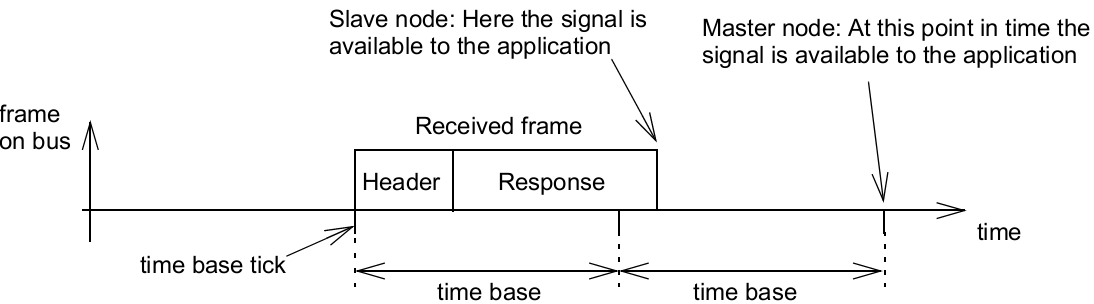


Figure 5.2Timing of signal reception

**Master node** – cập nhật định kì tín hiệu nhận tại các thời điểm time base, kế tiếp sau độ dài frame lớn nhất.

**Slave node** – sau khi checksum cho frame nhận có hiệu lực, node slave sẽ cập nhạt tín hiệu nhận của nó.

* **Phát tín hiệu:**

**Master node** – trước khi frame truyền được khởi tạo

**Slave node** – khi ID cho frame được nhận

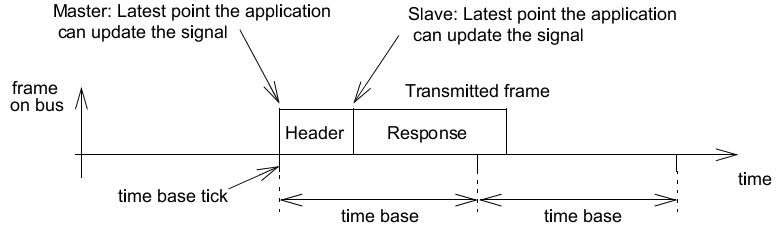


Figure 5.3 Timing of signal tranmission

### 3.3.2 Frame

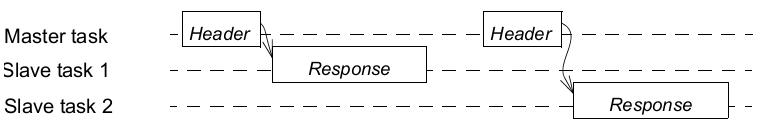


Figure 5.4 LIN Process

Các response bao gồm hai thành phần:

* Trường dữ liệu.
* Trường checksum – kiểm tra lỗi.

#### 3.3.2.1 Frame structure:

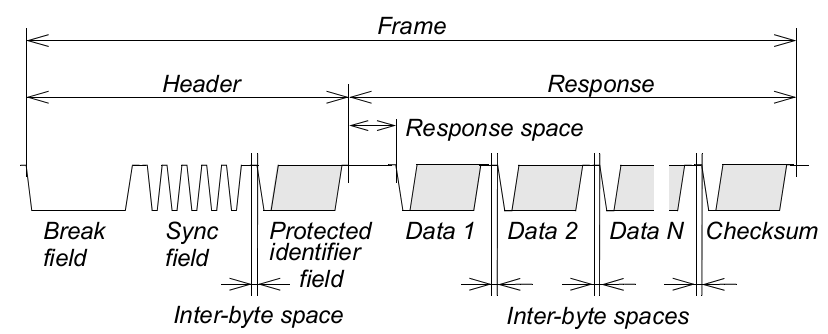


Figure 5.5 Frame structure

* **Header**: bắt đầu tại sườn xuống của break field và kết thúc sau stop bit của trường PID (Proctected Identifier).
* **Response**: bắt đầu cuối stop bit cùa PID field và kết thúc sau stop bit của checksum field.
* **Note**: **Inter-byte spaces** là thời gian giữa thời điểm kết thúc stop bit của field trước và start bit của field kế tiếp.
* Mỗi field (ngoại trừ break field) sẽ truyền trường byte như hình dưới:

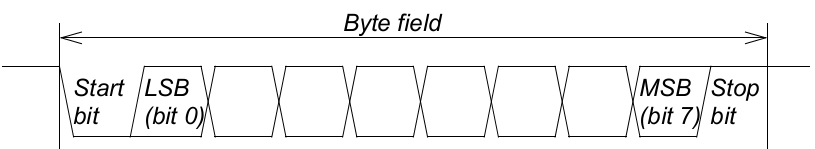


Figure 5.6 Struture of a byte field

LSB được gửi đầu tiên và cuối cùng là MSB, start bit được mã hóa với giá trị 0 và stop bit với giá trị 1.

* **Break field**: Sử dụng để bắt đầu một frame mới, luôn được phát ra bởi master node. Trường này có độ dài ngắn nhất là 13 bit (time), break delimiter (dấu ngăn cách) ít nhất là 1 bit. Slave node sử dụng một break detection có độ dài 11 bit time để xác định frame mới.



Figure 5.7 The break field

* **Sync byte field**: Khi một break/ sync field xảy ra liên tục thì quá trình truyền nhận sẽ bị hủy bỏ và quá trình mới được bắt đầu. Sync là một byte có giá trị 0x55.



Figure 5.8 The sync field

* **Protected identifier field**: Trường định dạng bảo vệ có 8 bit gồm hai phần:
* Frame identifier field: bit 0÷5, giá trị từ 0 ÷ 63. 0 ÷ 59 (0x3B) đánh dấu frame; 60 ÷ 61 dùng để nhớ chẩn đoán và cấu hình dữ liệu thường dùng để checksum phiên bản LIN 1.x; 62 ÷ 63 dành cho mở rộng giao thức trong tương lai.
* Parity: Tính toán



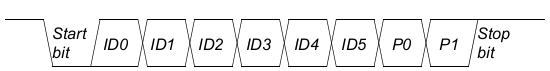


Figure 5.9 The protected identifier field

* **Data**: Độ dài từ 1÷8 byte, nếu độ dài là 1 byte thì truyền như hình 5.6. Độ dài dữ liệu từ 2 byte trở lên thì sẽ sắp xếp truyền byte LSB đầu tiên và MSB cuối cùng.

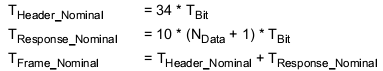


Figure 5.10 The data field

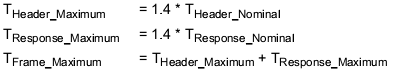
* **Checksum**: Nghịch đảo của 8 bit tổng của **tất cả các byte data** hoặc **trên tất cả data và PID**. Xác định hai loại checksum bởi frame identifier.
* Checksum tất cả byte data: gọi là checksum classic, nếu giá trị lớn hơn 255 thì quy thành 255=> LIN 1.x slave
* Checksum trên tất cả byte data và PID : gọi là checksum nâng cao=> LIN 2.x slave

#### 3.3.2.2 Frame length

* Công thức tính độ dài frame chuẩn:



* + Header: bao gồm 14bit chuẩn của break field, 10 bit sync field (2 bit start và stop bit), 10 bit PID
  + Response: 10 bit trong các gói data, 10 bit của checksum field.
* Độ dài frame tối đa: Độ dài tối đa giữa các byte tối đa thêm 40%



#### 3.3.2.3Frame types

* **Unconditional frame**: Frame identifier có giá trị từ 0 ÷ 59. Một sự truyền (tranfer) luôn được khởi tạo bởi master.

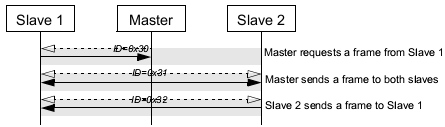


Figure 5.11 Three unconditional frame transfer

* **Event triggered frame**: Khung truyền sườn sự kiện, mục đích là tăng phản hồi của cụm LIN với việc phân công lớn từ độ rộng băng thông tới việc thu các phiếu của nhiều slave node với các sự kiện hiếm xảy ra.
* **Sporadic frame**:
* **Diagnostic frame**: Frame identifier là 60 và 61, gọi là frame phản hồi slave
* **Reserved frame**: Không sử dụng trong LIN 2.x cluster, vì nó là phần mở rộng với frame identifier là 62 và 63.

### 3.3.3 Data transport

Có hai kiểu dữ liệu có thể truyền trong một khung, bản tin tín hiệu và chẩn đoán:

* **Tín hiệu:** tín hiệu có giá trị vô hướng hoặc mảng byte được đóng gói trong trường dữ liệu của một khung truyền. Một tín hiệu luôn xuất hiện ở môt vị trí như nhau của trường dữ liệu cho tất cả khung truyền với các định dạng khung giống nhau.
* **Bản tin chẩn đoán:** Chuyển phát trong các khung truyền với hai định dạng khung truyền chuyên dụng. Sự rõ ràng của trường dữ liệu phụ thuộc vào trường dữ liệu của chính nó như trạng thái của nút truyền thông.

### 3.3.4 Schedule table

Nhiệm vụ master truyền cơ sở header trong một bảng lịch trình, đảm bảo rằng đường bus không bị quá tải.

* **Time definition**:

Thời gian nhỏ nhất sử dụng trong 1 cụm LIN là T\_base bao gồm thời gian thực hiện trong master node và sử dụng để điều khiển timing trong bảng lịch trình.

* Jitter: khác nhau giữa thời gian delay nhỏ và lớn nhất từ time base tick đến điểm bắt đầu (sườn xuống).
* Inter frame space: khoảng thời gian giữa điểm kết thúc frame đến frame kế tiếp.
* **Frame slot**:

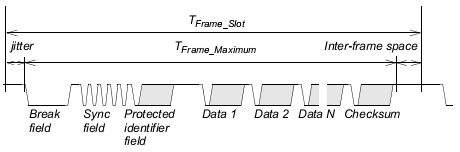


Figure 5.12 Frame slot

* **Schedule table handling**:

## 3.4 Task behavior model

### 3.4.1 Master task state machine

Công việc của master là tạo ra các header chính xác, quyết định frame sẽ truyền và duy trì các thời gian đúng giữa các frame phù hợp trong bảng lịch trình.

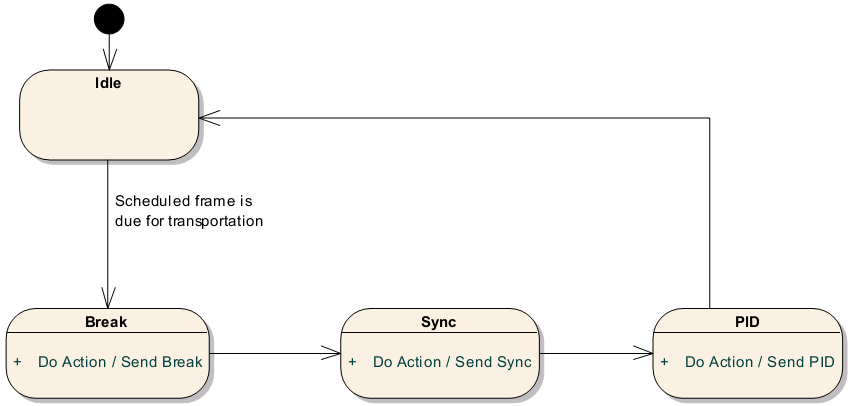


Figure 5.13 Master task state machine

### 3.4.2 Slave task state machine

Đảm bảo truyền các frame trả lời khi nó là pusliher và nhận các frame trả lời khi nó là subsciber. Vì vậy, slave sẽ có 2 mô hình máy trạng thái:

* Nhận biết các trường Break/sync (khi phát)
* Xử lý frame (khi nhận frame)
* Break/Sync field sequence:

Slave nhận chính xác PID, sync.

* Frame processor:

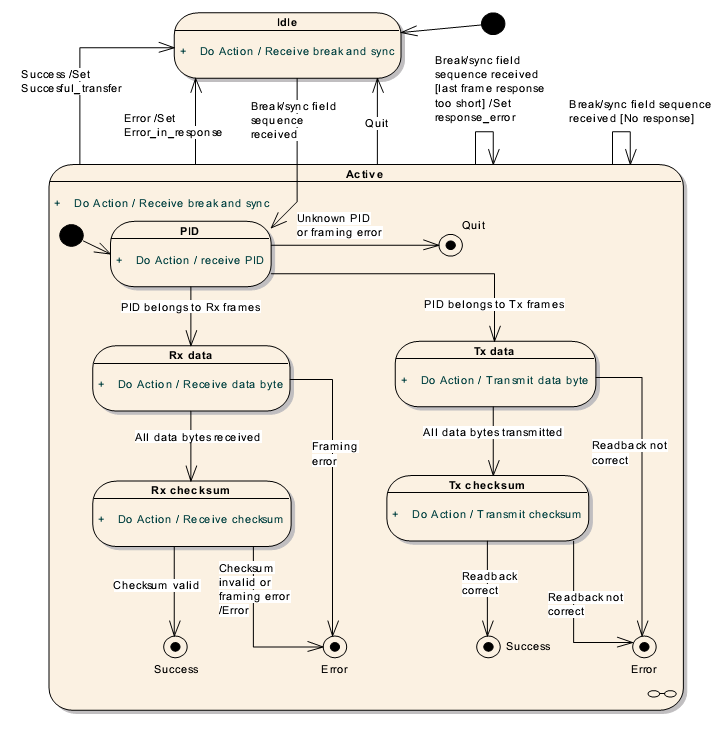


Figure 5.14 The frame processor state machine

## 3.5 Network management

Quản lý mạng trong cụm LIN là việc wake up và sleep cụm.

### 3.5.1 Slave communication state diagram

* Initializing: (Khởi tạo) Trạng thái sau reset và wakeup, slave node khởi tạo sau đó chuyển đến trạng thái operational. Reset và wakeup có thể khác nhau ở trạng thái khởi tạo.
* Operational: Hành vi giao thức transmitting và receiving frames.
* Bus sleep mode: Chỉ có tín hiệu wake up có thể truyền frame trong cluster.

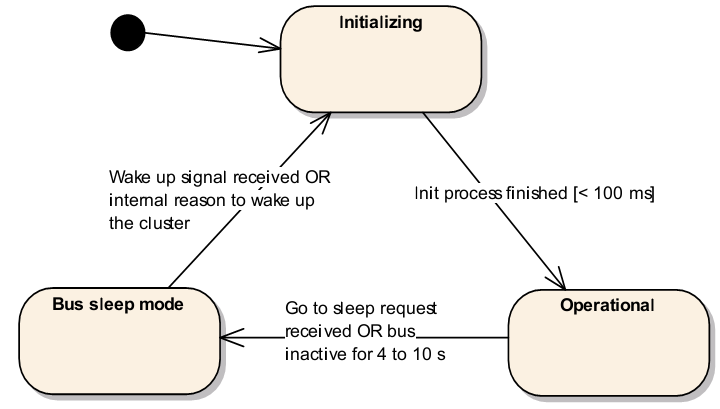


Figure 5.15 Slave mode communication state diagram

### 3.5.2 Wake up

Bất kì một node trong LIN cluster đều có thể yêu cầu một wake up bằng cách truyền một tín hiệu wake up. Master mode có thể gửi tín hiệu wake up bằng cách gửi một break field, có thể frame này không thực hiện được bởi slave node có thể chưa awake.

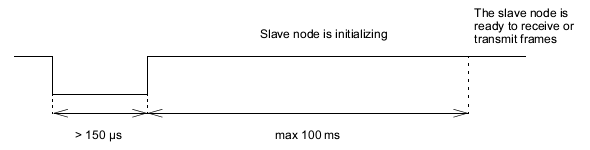
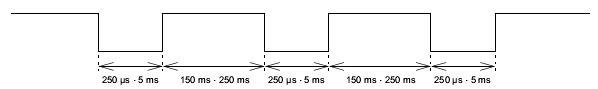


Figure 5.16 Wake up signal reception in slave nodes

Slave node sẽ detect tín hiệu wake up trong vòng >150us; và sẵn sàng để lắng nghe lệnh bus <= 100ms. Nếu slave node truyền tín hiệu wake up lên bus thì nó sẽ sẵn sàng ngay lập tức, không cần detect.



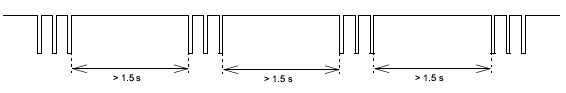


Figure 5.17 One block of wake up signal

Tín hiệu wakeup sẽ được phát ra trong thời gian 250us-5ms, sau đó sẽ node đó sẽ chờ 150ms-250ms mới truyền một tín hiệu wake up mới. Trong trường hợp node slave truyền một tín hiệu wake up trong thời gian như trên thì slave sẽ nhận và công nhận trường field này. Sau 3 yêu cầu failing thì node sẽ chờ ít nhất 1.5s trước khi phát 4 tín hiệu wake up.

### 3.5.3 Go to sleep

Master node thiết lập cluster sang trạng thái sleep bằng lệnh. Chỉ Master node có quyền truyền các lệnh go to sleep và các slave node chỉ cần quan tâm đến field thứ nhất là 0 , nếu ngừng lại thì thiết lập là 0xFF.



Figure 5.18 Go to sleep command

Slave node sẽ sleep sớm nhất sau 4s và muộn nhất là sau 10s

## 3.6 Status management

Mục đích: Xác định lỗi xảy ra trong quá trình hoạt động:

* Thay thế các đơn vị mắc lỗi.
* Cung cấp cho các node đến chế độ mềm khi xảy ra lỗi.

### 3.6.1 Concept

Master node sẽ quản lý trạng thái cluster, nó báo cáo trạng thái hiển thị từ mỗi node và lọc/tổng hợp các node lỗi.

### 3.6.2 Event triggered frames

### 3.6.3 Reporting to the cluster

Mỗi slave node sẽ được gắn với một tín hiệu vô hướng

## 3.7 Appendices

- Valid Frame Identifiers: page 50-51

- Tính toán checksum:

Tại các **node phát đi**: tính tổng các frame data và PID. Sau đó, đảo các bit lại => checksum.

Các **node nhận frame** sẽ tính tổng các frame data và PID sau đó cộng với check sum. Nếu kết quả là 0xFF => true

## 3.8 API

### 3.8.1 Cấu trúc file code

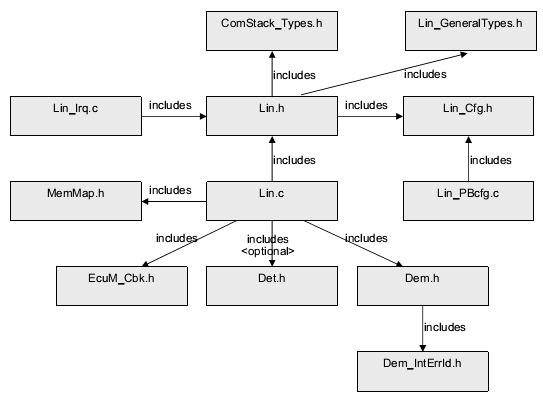
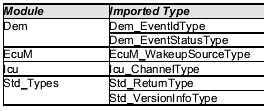


Figure 5.19 Header file structure for the LIN driver

### 3.8.2 Imported types

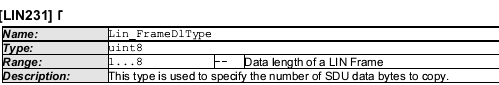


Tất cả các kiểu include từ các module khác sẽ được liệt kê ra.

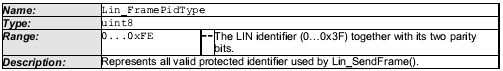
### 3.8.3 Type definitions

* **Lin\_Configtype**:

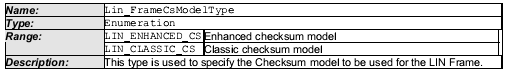
Hàm này cung cấp bởi file **Lin.h**. Đây là kiểu cấu trúc mở rộng bao gồm khởi tạo cho LIN driver và thiết lập SFR ảnh hưởng đến các kênh LIN.



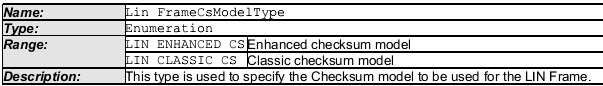
* **Lin\_FramePidType**:



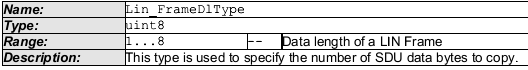
* **Lin\_FrameCsModelType**:



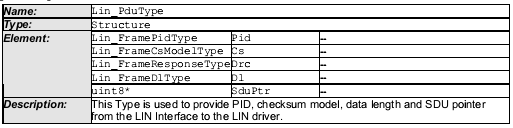
* **Lin\_FrameResponseType**:



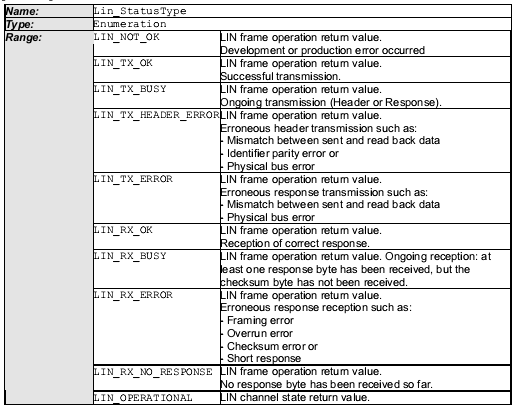
* **Lin\_FrameDIType**: Độ dài của frame LIN

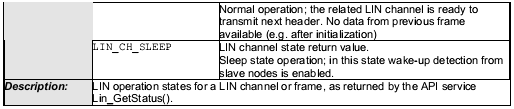


* **Lin\_PduType**:

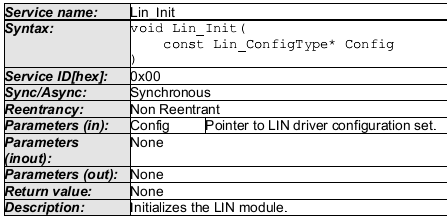


* **Lin\_StatusType**:



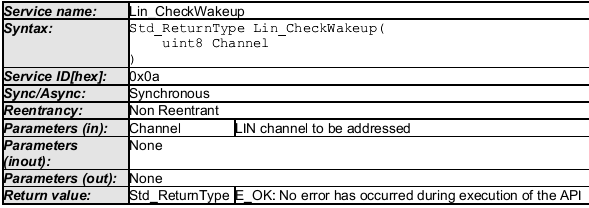


### 3.8.4 Lin\_Init



Khởi tạo module Lin tùy theo điểm thiết lập cấu hình bởi cấu hình tham số.

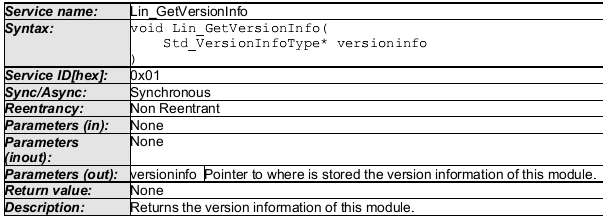
### 3.8.5 Lin\_CheckWakeup



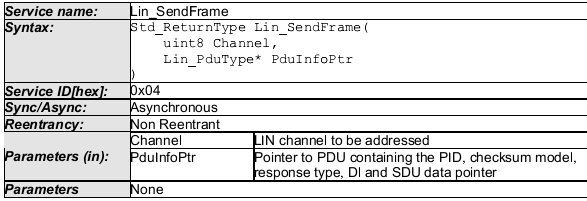


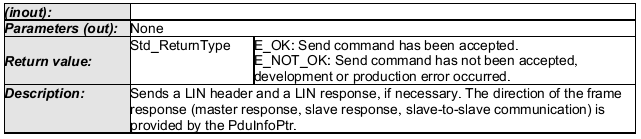
Hàm này kiểm tra nếu một wakeup đã xảy ra trên địa chỉ kênh LIN.

### 3.8.6 Lin\_GetVersionInfo



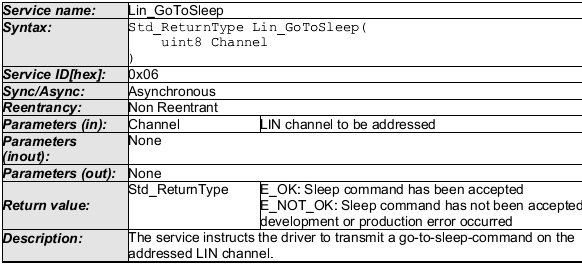
### 3.8.7 Lin\_SendFrame





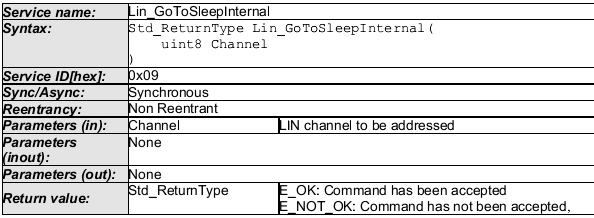
Gửi một header LIN và response LIN nếu cần thiết. Định hướng frame response (master, slave và slave-slave) được cung cấp bởi PduInfoPtr.

### 3.8.8 LinGoToSleep



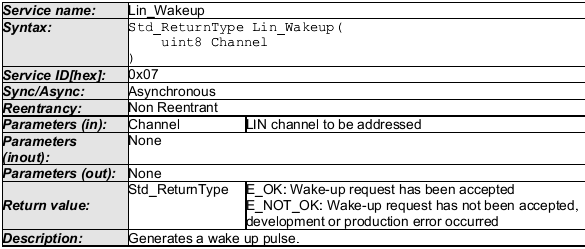
Chỉ dẫn dịch vụ driver truyền một lệnh go to sleep trên địa chỉ kênh LIN.

### 3.8.9 Lin\_GoToSleepInternal

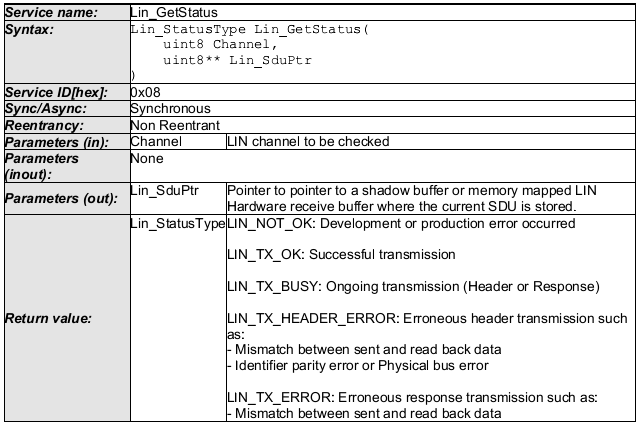


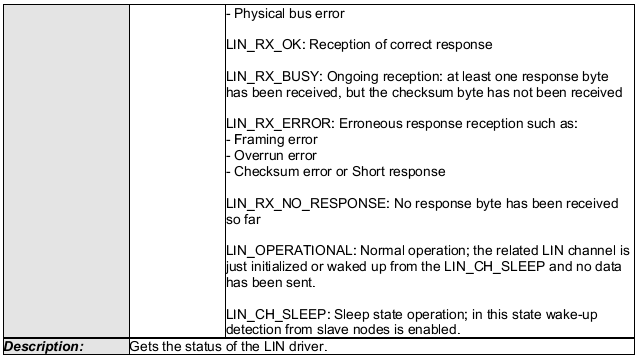
Thiết lập trạng thái kênh tới LIN\_CH\_SLEEP, cho phép wake up xác định và tùy chọn thiết lập đơn vị phần cứng LIN.

### 3.8.10 Lin\_Wakeup



### 3.8.11 Lin\_GetStatus





Trả lại trạng thái của LIN driver.

### 

# 4. CC, CQ

## 4.1 ClearCase

Quản lý file, thư mục code (Thể hiện về code)

* Element => component => VOB
* View: Workspace
* Config spec: Load các thành phần của VOB
* Branch: Phát triển các nhánh code
* Merge:
* Sử dụng label: nhanh hơn khi config spec

## 4.2 CQ?

Thể hiện về mặt record

* Lý do, mục đích thay đổi code
* Dạng text
* Tạo ra các CR (tương đương branch)
* Có 3 loại CR: defect, newwork, code

# 5. OVERVIEW OF GROUP OPERATION

* MCAL (IP++): code driver (hiểu là library) các dòng VĐK Race, …
* ITG: Validation, test code driver của nhóm Dev IP++ trong MCAL, theo 3 tiêu chí (Yêu cầu):
* AUTOSAR specific
* Customer request
* Improve

Công việc: MAF (Code test tín hiệu đầu ra sau quá trình BEART) & **BEART**

* Các server:
  + MCAL server VOB: **SASW**
  + Server chung VOB: **vnv\_dist**
  + Server ITG VOB: **vnv\_test**

# 6. FOLDER TREE

**B40707\_view\_Matterhorn\_Test/vnv\_dist/make/local\_cofig.mak**

**B40707\_view\_Matterhorn\_Test/vnv\_test/auto/asr/mcal/test/test\_port/generic/[include/src] (TC.h/TC.c)**

**B40707\_view\_Matterhorn\_Test/vnv\_test/auto/asr/mcal/test/test\_port/make(TS.mk)**

**B40707\_view\_Matterhorn\_Test/vnv\_test/auto/asr/mcal/test/test\_port/specific/[include/src] (TS.h/TS.c)**

**B40707\_view\_Matterhorn\_Test/vnv\_test/OUTPUT**