# TÓM TĂT

ĐẠI HỌC ĐÀ NẴNG

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

KHOA ĐIỆN TỬ - VIỄN THÔNG

**CỘNG HÒA XÃ HÔI CHỦ NGHĨA VIỆT NAM**

Độc lập - Tự do - Hạnh phúc

# NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên: HUỲNH VŨ TIẾN Số thẻ sinh viên: 106120206

Lớp: 12DT4 Khoa: ĐIỆN TỬ VIỄN THÔNG Ngành: KỸ THUẬT MÁY TÍNH

1. Tên đề tài đồ án:

………………………………………………..……………………………………

2. Đề tài thuộc diện: ☐ Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện

3. Các số liệu và dữ liệu ban đầu:

4. Nội dung các phần thuyết minh và tính toán:

5. Các bản vẽ, đồ thị ( ghi rõ các loại và kích thước bản vẽ ):

6. Họ tên người hướng dẫn: …………………………………..……………………

7. Ngày giao nhiệm vụ đồ án: ……../……./201…..

8. Ngày hoàn thành đồ án: ……../……./201…..

Đà Nẵng, ngày tháng năm 201

**Trưởng Bộ môn Người hướng dẫn**

# LỜI NÓI ĐẦU

# LỜI CAM ĐOAN

Sinh viên thực tập

# MỤC LỤC

# DANH SÁCH BẢN BIỂU VÀ HÌNH VẼ

# DANH SÁCH CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT

# MỞ ĐẦU

# CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

* 1. **Đặt vấn đề**

Cùng với sự phát triển không ngừng của khoa học công nghệ, công nghiệp ô tô cũng đã và đang được hoàn thiện nhằm mục đích đem đến cho người dùng những trải nghiệm ngày càng thoải mái và tiện nghi hơn. Sự ra đời của các công nghệ, thiết bị hiện đại như cảnh báo chệch làn đường, cảnh báo tiền va chạm, hệ thống hỗ trợ phanh và điều khiển động cơ bánh lái…đã mang lại cho người dùng những chuyến hành trình ngày càng thú vị.

Tuy nhiên, để áp dụng những công nghệ mới này lên trên một sản phẩm xe ô tô thực tế để đưa ra ngoài thị trường thì từ khâu kiểm thử, vận hành cho đến phát triển các ứng dụng về sau thì đây cũng là một khó khăn đối với các nhà nghiên cứu, cũng như sản xuất ô tô công nghiệp. Việc sử dụng cả hệ thống thực tế chỉ để kiểm thử với một ECU (Electronic Control Unit) mới nhằm hướng tới sự tương thích giữa các ECU với nhau có thể gây ra sự phức tạp cho toàn bộ hệ thống CAN (Controller Area Network ).

Chính vì vậy, việc xây dựng nên một phần mềm có thể mô phỏng hoạt động như một hệ thống hoàn chỉnh nhằm thay thế các ECU thực tế là thật sự cần thiết. Nó sẽ giúp cho việc phát triển thêm ứng dụng đơn giản hơn, với độ tin cậy và hiệu quả cao. Đó là lý do hiện tại, công ty FPT Software đang phát triển và xây dựng “Hệ thống mô phỏng CAN Bus” để có thể giải quyết được vấn đề trên

* 1. **Tổng quan về hệ thống**

Đây là sơ đồ khối mô hình kết nối khi sử dụng, sơ đồ khối mô tả chung của đề tài.



*Hình 1.1 : Sơ đồ khối mô hình kết nối khi sử dụng*



*Hình 1.2 : Sơ đồ khối mô tả chung hệ thống CAN SIMULATOR*.

Đề tài được thực hiện chủ yếu dựa trên việc lập trình ngôn ngữ C/C++, sử dụng kit TM4C123G của hãng Texas Instruments và một số phần mềm hỗ trợ khác như CCS, Qt Creator,....

Hệ thống gồm có 3 phần chính:

* CAN Software
* Driver
* CAN Simulartor Device
  + 1. **CAN Software**

Là một phần mềm độc lập, có nhiệm vụ xử lí việc truyền nhận dữ liệu cũng như phân tích và hiển thị nội dung của các thông điệp, mà định dạng của các thông điệp đó được lấy từ D atabase edit software.

Chức năng:

* Cấu hình hệ thống, cấu hình được cổng CAN nào đang hoạt động, tốc độ Baund
* Load được database vào test model
* Add/remove được sơ đồ khối kết nối đồng thời cập nhật cấu hình và lưu lại.
* Add/remove được các message vào trong IG.
* Trong IG (interactive generator) có thể sửa được dạng sóng của các signal: hình sin, răng cưa, hình chữ nhật,…
* Trong IG có thể setting cycle của từng message.
* Trong lúc đang simulate có thể edit nội dung message ở dạng raw data hoặc physical value.
* Trace window hiển thị các thuộc tính của message: ID, length, direction, data, signal value.
  + 1. **Driver**

PC Driver có nhiệm vụ giao tiếp giữa hardware và software .Khi một device được kết nối với PC, driver nhận nhiệm vụ tạo ra vùng nhớ đệm thông qua các file descriptor. Khi ta muốn truyền một thông điệp từ PC xuống Hardware thì Driver sẽ nhận biết được thông điệp, sau đó ghi vào file descriptor tương ứng. Hardware sẽ đọc dữ liệu từ file descriptor và thực hiện nội dung thông điệp. Thông điệp phản hồi sẽ được truyền theo chiều ngược lại đến Software.

* Nhận biết được thiết bị connect, disconnect thông qua cổng USB, đưa ra thông tin, trạng thái của thiết bị.
* CAN software có thể đóng/ mở thiết bị thông qua device node.
* CAN software có thể cấu hình baurd rate thông qua file descriptor.

CAN software có thể gửi/nhận message thông qua file descriptor

* + 1. **CAN Simulator Device**

Cấu hình 2 giao tiếp CAN và USB để có thể nhận và gửi các message giữa PC và board cũng như từ board ra Bus CAN .

* Nhiệm vụ chính:
* Giao tiếp USB nhận và gởi message để giao tiếp giữa board và PC
* Giao Tiếp CAN
* Nhận CAN Frame từ CAN Bus và gởi dữ liệu thu được từ CAN Frame lên PC.
* Kiểm tra thiết bị đã được kết nối với CAN Bus hay chưa, trả kết quả lại cho PC.
* Cấu hình được tốc độ của thiết bị để có thể kết nối được với Bus CAN bất kỳ.
  1. **Các giải pháp và công nghệ dự kiến sử dụng.**
  2. **Phạm vi và bảng phân công nhiệm vụ**
  3. **Kết luận chương**

Trong phần này chúng ta đã tìm hiểu sơ bộ về đề tài và các thành phần chính trong đề tài. Tuy nhiên vì đề tài rất lớn với nhiều nhóm sinh viên cùng thực hiện các phần nhỏ liên quan, nên sẽ có nhiều nhóm nhỏ thực hiện các phần khác nhau. Chính lý do chia nhỏ đề tài như trên, trong cuốn đồ án này chúng ta chỉ tập trung tìm hiểu về CAN Simulator Device.

Vì vậy, trong các chương tiếp theo chúng ta sẽ tìm hiểu cụ thể hơn về cơ sở lý thuyết của CAN và USB dùng để thực hiện CAN Simulator Device.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

* 1. **Giới thiệu chương**

Như đã giới thiệu ở chương 1, ở phần này chúng ta sẽ đi sâu vào cơ sở lý thuyết của phần CAN và USB, cũng như tìm hiểu sơ bộ về Board DK-TM4C123G (phần cứng được sử dụng để thực hiện thiết bị).

* 1. **Lý thuyết về giao thức CAN và USB**

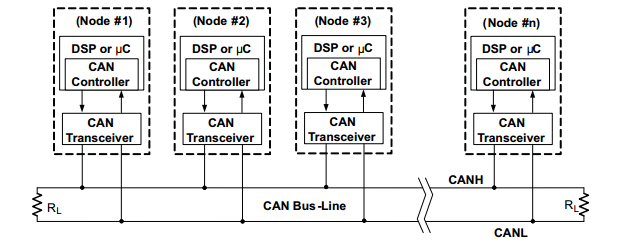
Chức năng chính của CAN Simulator Device là tạo một node CAN có thể tùy chỉnh được tốc độ của node có thể kết nối được với một Bus CAN bất kỳ để lấy các CAN\_Frame trong CAN Bus gởi lên PC cũng như đẩy CAN\_Frame xuống Bus.

Để thực hiện các chức năng đó chúng ta dung giao thức USB và CAN. Board DK-TM4C123G hỗ trợ đầy đủ các yêu cầu trên và chúng ta sẽ tìm hiểu kỹ về giao thức CAN, USB trong phần này.

* + 1. **CAN** 
       1. **Giới thiệu tổng quan về giao thức CAN**

Controller Area Network (CAN) là một giao thức truyền thông bus nối tiếp phát triển bởi Bosch (một nhà sản xuất thiết bị điện tại Đức) vào đầu những năm 1980. Sau đó, CAN đã được chuẩn hóa theo tiêu chuẩn ISO-11898 ,ISO-11519 và nó được sử dụng như là giao thức chuẩn để giao tiếp kết nối mạng trong ngành công nghiệp ôtô.

CAN được tổ chức International Standization Organization (ISO) định nghĩa là một chuẩn truyền thông nối tiếp, chuyên cho ngành công nghiệp Ô tô, nhằm thay thế cho việc đi dây phức tạp bằng Bus hai dây.



*Hình 2.1 Sơ đồ kết nối của một Bus CAN*

CAN được thiết kế là một mạng đa chủ, mỗi thiết bị đều có thể gởi và nhận tín hiệu. Không giống như các loại mạng truyền thống, CAN không gởi một gói tin lớn từ điểm này đến điểm kia mà nó chia mà nó gởi nhiều gói tin nhỏ theo dạng broadcast đến tất cả các Node trong Bus. Các Node trên Bus bắt các gói tin và dựa và trường ID trong gói tin nhận được để xử lý hay loại bỏ gói tin.

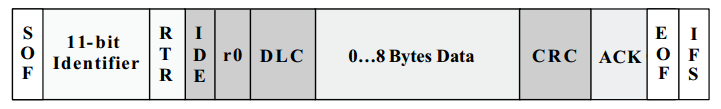
CAN được dựa trên giao thức CSMA / CD+AMP (Carrier-Sense Multiple Access: Đa truy cập có tránh xung đột / Collision Detection Arbitration on Message Priority: Giám sát phát hiện xung đột dựa trên thông điệp ưu tiên).

Khi mạng Ethernet phát hiện một vụ xung đột, các nút đang gửi dữ liệu chỉ đơn giản là dừng truyền và đợi một khoảng thời gian ngẫu nhiên trước khi cố gắng để gửi một lần nữa. Tuy nhiên, đối với giao thức CAN có một chúc khác biệt đó là các vấn đề xung đột được giải quyết bằng cách sử dụng các nguyên tắc của trọng tài, nơi chỉ có các nút ưu tiên cao nhất được trao quyền để gửi dữ liệu của nó.

* + - 1. **Standard CAN – Extended CAN**

Hiện này chúng ta có hai chuẩn CAN là Standard CAN và Extend CAN, với Extern CAN là chuẩn CAN phiên bản cải tiến của CAN.

* + - * 1. **Standard CAN**

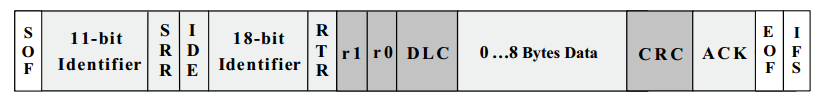


*Hình 2.2: Khung truyền của Standard CAN*

*Bảng :*

|  |  |  |
| --- | --- | --- |
| Tên | Độ dài | Mục đích |
| SOF | 1 bit | Bit bắt đầu của khung truyền |
| Identifier(ID) | 11 bit | Chứa ID của message |
| RTR | 1 bit | Phân biệt data frame hay remote frame |
| IDE | 1 | Phân biệt loại extended hay standard |
| R0 | 1 | Bit dành riêng.(chỉ dành để sủa đổi tiêu chuẩn trong tương lai) |
| DLC | 4 | Độ dài của trường data đơn vị (byte) |
| Data | 0-64 bit | Dữ liệu của message |
| CRC | 16 bit | Mã kiểu tra lỗi CRC |
| ACK | 2 bit |  |
| EOF | 1 bit | Kết thúc của Frame |
| IFS | 7 bit | Interframe Space (chứa thời gian yêu cầu từ controller để di chuyển khung truyền nhận được vào vị trí thích hợp trong bộ đệm ) |

* + - * 1. **Extend CAN**



*Hinh 2.3 Extended CAN Frame*

Khác biệt cơ bản nhất giữa Standard CAN và Extended CAN là trường ID 29 bit.

* + - * 1. **Cơ chế trọng tài (Arbitration)**

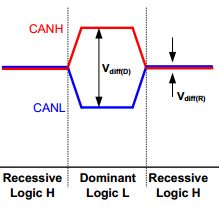
**Bài toán xung đột trong Bus**

Vì CAN là một mạng đa chủ, mỗi thiết bị trong mạng đều có thể gởi và truyền tín hiệu, nên sẽ có khả năng cùng một thời điểm sẽ có nhiều hơn một thiết bị có nhu cầu sử dụng đường truyền. Khi điều đó xảy ra thì xung đột sẽ xảy ra trên đường truyền. Vì thế chúng ta cần một cơ chế để có thể giải quyết được xung đột trên.

Để giải quyết xung đột trên, CAN sử dụng cơ chế trọng tài (Arbitration) để giải quyết vấn đề này.

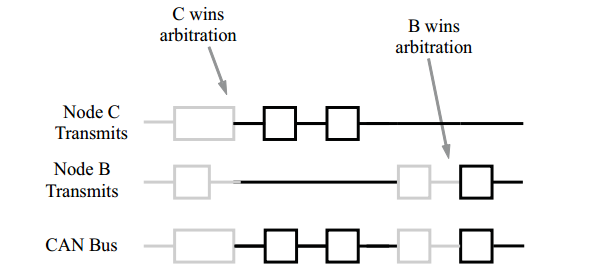
**Cách giải quyết xung đột trong Bus**

Với một các mạng bình thường thì mức logic cao sẽ tương ứng với mức 1, và múc logic thấp là mức 0 – tuy nhiên điều này không chính xác trong CAN. Trong CAN, mức logic cao sẽ là Recessive và múc thấp là Dominant.



Hình 2.4: phân định trạng thái High và Low trong Bus

Như đã nêu ở trên, khi có nhiều hơn một thiết bị trên Bus yêu cầu đường truyền thì trên Bus sẽ xảy ra xung đột. Vì vậy trong CAN sẽ phân cấp trường ID trong các message, việc phân cấp này sẽ giúp chúng có các ID đươc ưu tiên trương Bus. Nếu hai thiết bị cùng truyền đi hai message thì message có trường ID với mức ưu tiên cao hơn sẽ được gởi truyền đi trong Bus.

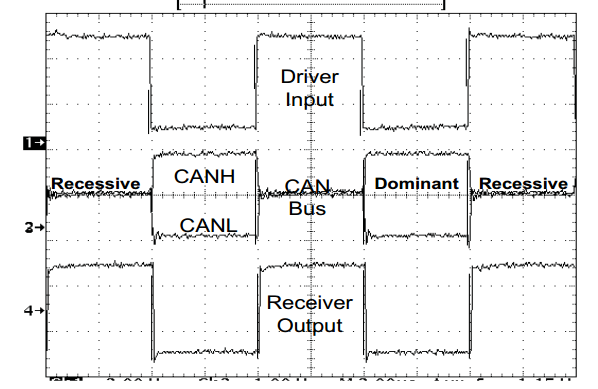


*Hình 2.5: cơ chế trọng tài giải quyết xung đột trong CAN Bus*

Chính nhờ sự ưu tiên khác nhau của các message mà chúng ta có thể gởi nhiều message trong cùng một lúc tuy nhiên chỉ có message có múc ưu tiên cao nhất là được truyền trong Bus. Vì vậy trong một Bus, các ID có mức ưu tiên cao sẽ được để dành để đánh số ID cho các message có yêu cầu quan trọng trọng hệ thống, ví dụ như hệ phanh, ga, túi khí, …. Còn các hệ thống có độ ưu tiên thấp như hệ thống đa phương tiện, điều hòa, … sẽ được nhập mưc ưu tiên cho ID thấp hơn các hệ thống quan trọng khác.

* + - * 1. **Nhận dạng dữ liệu trong Bus**

Trong hoạt động của CAN Bus thì một lúc chỉ có một node được gởi dữ liệu và tất cả các node còn lại đều nhận tin nhắn đó. Chúng dựa vào trường ID để để nhận CAN\_Frame.



*Hình 2.6: kênh nhận và truyền nhận dạng trạng thái trên Bus*

* + 1. **Giao thức USB** 
       1. **Tổng quan về giao thức USB**

USB (Universal Serial Bus) là một chuẩn kết nối tuần tự đa dụng trong máy tính. USB sử dụng để kết nối các thiết bị ngoại vi với máy tính, chúng thường được thiết kế dưới dạng các đầu cắm cho các thiết bị tuân theo chuẩn cắm-và-chạy (plug and play) mà với tính năng cắm nóng thiết bị (nối và ngắt các thiết bị không cần phải khởi động lại hệ thống).

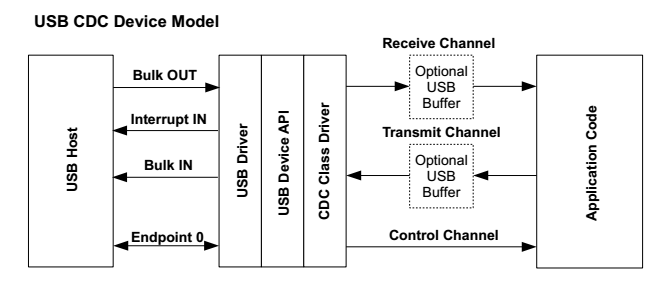
USB là một giao thức nối tiếp tốc độ cao, nó có thể cung cấp điện cho các thiết bị kết nối với nó. Một bus USB sử dụng trường địa chỉ 7-bit để đánh địa chỉ cho các thiết bị trong Bus, lưu ý rằng địa chỉ 0 không được sử.

Giao thức USB có hai mode hoạt động chính là host mode và device mode:

* USB Host : thiết bị đóng vai trò điều khiển toàn bộ mạng USB (có thể lên tới tối đa 126 thiết bị). Ví dụ như trên máy tính, USB Host được gắn trên mainboard. Để giao tiếp và điều khiển các USB device, USB Host controller cần được thiết kế tích hợp với USB RootHub (Hub mức cao nhất). Vai trò của thiết bị USB Host:
* Trao đổi dữ liệu với các USB device.
* Quản lý USB Bus
* USB Device: là các thiết bị đóng vai trò như các slave giao tiếp với USB Host. Xin lưu ý một điều hết sức quan trọng đó là các thiết bị này hoàn toàn đóng vai trò bị động, không bao giờ được tự ý gửi gói tin lên USB Host hay gửi gói tin giữa các USB Device với nhau, tất cả đều phải thông qua quá trình điều phối của USB Host. Các bạn sẽ hiểu cơ chế này rõ hơn trong phần truyền thông của chuẩn USB. Chức năng của thiết bị USB Device:
* Trao đổi dữ liệu với USB Host
* Phát hiện gói tin hay yêu cầu từ USB Host
  + - 1. **Các khái niệm cơ bản**
* Endpoint: giống như là môt phần địa chỉ duy nhất của thiết bị, nó là nguồn thông tin trong dòng liên lạc giữa Host và thiết bị. Về cơ bản các thiết bị được thiết kế với tối ta 16 Endpoint, được phần loại thành hai loại:
* Các Endpoint truyền dữ liệu từ USB Device tới USB Host là endpoint IN.
* Các Endoint truyền dữ liệu từ USB Host tới USB Device là endpoint OUT.
* Pipes: là sự kết hợp giữa endpoint ở thiết bị với controller trên Host. Host có thể tạo các pipes mới hay hủy các pipe không cần thiết luân phiên theo yêu cầu của giao tiếp với thiết bị.
* Các chế độ truyền: chuẩn USB cung cấp cho chúng ta bốn chế độ truyền, tùy và mục đích và yêu cầu sử dụng mà người sử dụng chế độ phù hợp
* Truyền điều khiển (Control transfer): là chế độ truyền được tất cả các thiết bị USB hỗ trợ để truyền các thông tin điều khiển với tốc độ tương đối chậm.
* Truyền ngắt (Interrupt transfer): sử dụng cho các thiết bị cần truyền một lượng dữ liệu nhỏ, tuần hoàn theo thời gian ví dụ như chuột, bàn phím. Khi đó, ví dụ cứ 10s một lần USB Host sẽ gửi request xuống và USB Device sẽ trả dữ liệu về cho USB Host (với trường hợp Interrupt In Endpoint).
* Truyền theo khối (Bulk transfer): sử dụng cho các thiết bị cần truyền một lượng dữ liệu lớn, yêu cầu độ chính xác tuyệt đối, không có ràng buộc quá chặt chẽ về thời gian thực ví dụ như thẻ nhớ USB, máy in. Cái này tương tự như giao thức TCP trong mạng Ethernet
* Truyền đẳng thời (Isochronos transfer): sử dụng cho các thiết bị cần truyền một lượng dữ liệu lớn với tốc độ rất nhanh, đảm bảo ràng buộc về thời gian thực tuy nhiên chấp nhận hy sinh độ chính xác ở một mức nhất định như các thiết bị nghe nhạc, xem phim kết nối theo chuẩn USB
  + - 1. **USB Communications Device Class**

USB Communications Device Class là một lớp thiết bị trong USB Device, nó cho phép các ứng dụng của thiết bị được truyền nhận dữ liệu thông qua cổng serial ảo trong hệ thống. Driver cung cấp hai kênh truyền, một truyền và một nhận. Các kênh truyền kết hợp với bộ đệm cung cấp cho chúng ta các giao diện để đọc và ghi dữ liệu với host USB.

Dữ liệu trong USB Communications Device Class sử dụng bốn Endpoint để truyền và nhận dữ liệu. Hai Bulk Endpoint để truyền và nhận dữ liệu từ host và Interrupt IN Endpoint để dùng để báo hiệu các lỗi (lỗi khung truyền, lỗi Parity bit được thiết bị phát hiện) cho Host. Endpoint 0 mang các yêu cầu của chuẩn USB và các yêu cầu cụ thể của CDC thông qua control chanel callback.



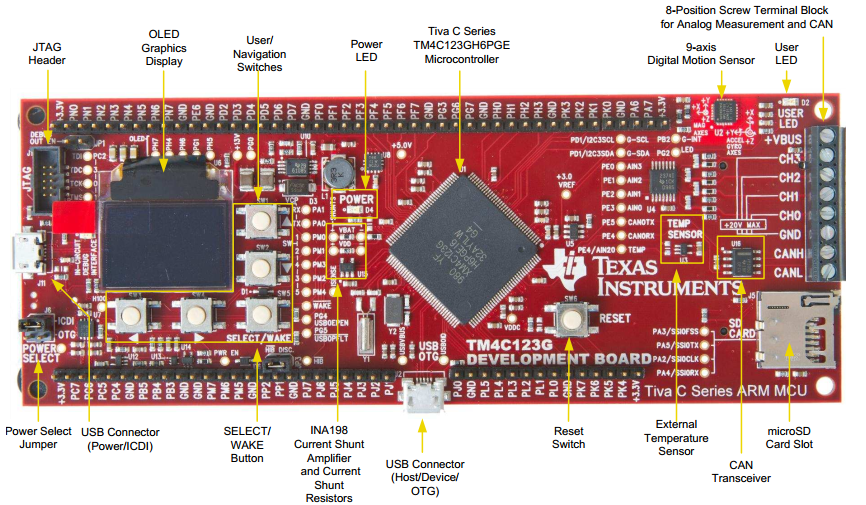
*Hình 2.1 mô hình của USB CDC hỗ trợ trong KIT DK-TM4C123G*

* 1. **Giới thiệu về phần cứng – Board TM4C123G**
     1. **Giới thiệu về Board DK-TM4C123G**

Kit DK-TM4C123G là dòng Kit phát triển do TI sản xuất, chuyên hỗ trợ cho các ứng dựng sử dụng giao thức USB và CAN.

DK-TM4C123G là dòng Kit mạnh mẽ và được phát triển sử dụng dong chip Tiva C Series TM4C123G ARM® Cortex™-M4-based. Kit được thiết kế đặc biệt với chip TM4C123G và được tích hợp các giao thức như:

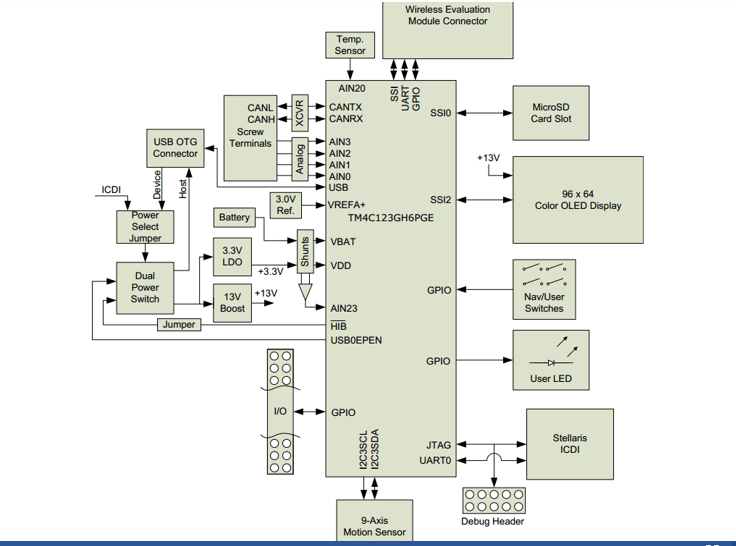
* Giao thức USB 2.0 (OTG, Host, Device).
* Giao thức Controller Area Network (CAN)
* 12-bit Analog-to-Digital Converter (ADC)
* Thời gian thực (RTC) cùng pin cấp nguồn cho module RTC riêng biệt
* …



*Hình 2.4: DK-TM4C123G do TI sản xuất*

Kit DK-TM4C123G có các thông số cơ bản sau:

* Vi xử lý Tiva TM4C123GH6PGE.
* Cảm biến gia tốc 9- trục (accelerometer + gyro + compass).
* Hai cảm biến nhiệt độ
  + Internal microcontroller temperature sensor
  + External TMP20 temperature sensor
* Controller Area Network (CAN) transceiver.
* Bốn chân đầu vào Analog (0-20V).
* Microcontroller current shunt amplifier
* Màn hinh màu OLED 96 x 64.
* USB Micro-AB connector for Host/Device/OTG
* microSD card slot
* Năm nút bấm
* LED hỗ trợ cho người dung tại chân PG2
* Debug
  + In-Circuit Debug Interface (ICDI)
  + Standard 10-pin JTAG header (debug-out capable)
* Pin hỗ trợ chế độ Hibernate
* Nút Reset



*Hình 2.5: sơ đồ khối cho Kit DK-TM4C123G*

* + 1. **Giới thiệu về phần USB trên Board DK-TM4c123G**

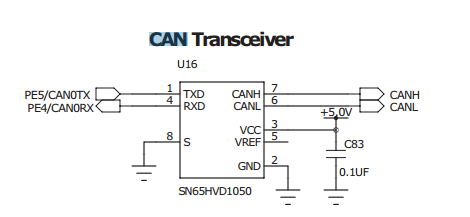
Kit DK-TM4C123G hỗ trợ giao tiếp USB Micro-AB

Sơ đồ chân:

|  |  |  |
| --- | --- | --- |
| GPIO Pin | Pin function | USB |
| PL6 | USB0DP | D+ |
| PL7 | USB0DM | D- |
| PB0 | USB0ID | ID |
| PB1 | USB0VBUS | USB VBUS |

* + 1. **Giới thiệu về phần CAN trên Board DK-TM4C123G**

Sử dụng chân PE4 và PE5 của TM4C123GH6PGE (module UART 5) để truyền nhận dữ liệu trong CAN Bus thông qua IC High-Speed CAN Transceceiver SN65HVD1050



*Hình 2.6: CAN Transceceiver*

Để sử dụng module CAN của Kit chúng kết nối hai chân CANH và CANL và Bus CAN cần kết nối.

* 1. **Kết luận chương**

Trong chương này chúng ta đã tìm hiểu được tương đối đầy đủ về giao thức CAN, USB (CDC) và tìm hiểu về thông số cấu hình chân của VXL Tiva TM4C123GH6PGE nhằm phục vụ cho mục đích mô phỏng node CAN. Trong chương tiếp theo chúng ta sẽ thực hiện quá trình lập trình và kiểm tra khả năng hoạt động của hệ thống thực hiện được.

# CHƯƠNG 3 : THỰC HIỆN VÀ ĐÁNH GIÁ, KẾT QUẢ THU ĐƯỢC

* 1. **Giới thiệu chương**

Mục đính chính của chúng ta là tìm hiểu cụ thể về giao thức CAN, quá trình truyền nhận dữ liệu trong CAN, cùng các xử lý các yêu cầu từ PC gởi xuống. Trong chương này, chúng ta sẽ tìm hiểu cụ thể và chi tiết về quá trình lập trình và thiết kế hoạt động của thiết bị hỗ trợ cho mô phỏng CAN Bus.

* 1. **Thiết kế hoạt động của CAN Device**
     1. **Các khung truyền cơ bản**
        1. **Cơ bản về khung truyền trong hệ thống**

Như đã nêu ở trên, chúng ta cần phải thống nhất khung truyền giữa PC và Board để hai bên có thể hoạt động với nhau được. Với mỗi hoạt động của hệ thống và yêu cầu xử lý riêng nên sẽ có nhiều định dạng khung truyền khác nhau, tuy nhiên chúng ta cần có một định dạng cố định như dưới.



*Hình 3.1: định dạng của khung truyền chung*

**SourceType/ SourceID/ DestID/ DestType :**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Definition** | **Value** | **Mô tả** |
| 1 | PROTO\_DEV\_TYPE\_SOFT | 0x01 | Software PC |
| 2 | PROTO\_DEV\_TYPE\_CAN | 0x02 | CAN Device |
| 3 | PROTO\_DEV\_TYPE\_SOFT | 0x03 | Chưa sử dụng |
| 4 | PROTO\_DEV\_TYPE\_SOFT | 0x04 | Chưa sử dụng |

**CommandID(2 byte):** xác định các trường hợp sẽ xảy ra khi giao tiếp giữa PC và Board, với kịch bản đã được thiết kế trước.

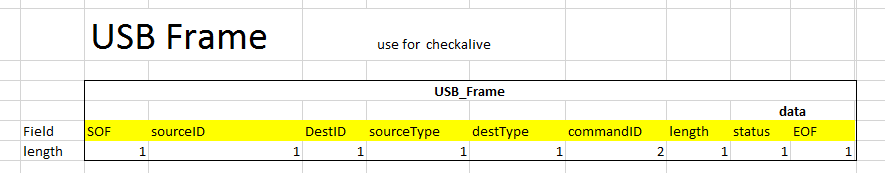
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CASE** | **STT** | **Definiton** | **Giá trị** | **Direction** | **Mô tả** |
| CheckAlive | 1 | CHECK\_ALIVE\_REQ | 0x0001 | PC->Board | PC yêu cầu Board kiểm tra kết nối giữa Board với Bus |
| 2 | CHECK\_ALIVE\_RES | 0x0002 | Board->PC | Board trả lời theo yêu cầu của PC |
| Send Message | 3 | SEND\_MSG\_REQ | 0x0003 | PC->Board | Gởi message yêu cầu gởi dữ liệu |
| 4 | SEND\_MSG\_RES | 0x0004 |  | Board trả lời yêu cầu gởi dữ liệu cho PC |
| 5 | SEND\_MSG | 0x0005 |  |  |
| 6 | SEND\_MSG\_FINISH | 0x0006 |  |  |
| Send Data | 7 | SEND\_DATA\_REQ | 0x0007 |  |  |
| 8 | SEND\_DATA\_RES | 0x0008 |  |  |
| 9 | SEND\_DATA | 0x0009 |  |  |
| 10 | SEND\_DATA\_FINISH | 0x000A |  |  |
| Error | 11 | ERROR\_MSG | 0x000B |  |  |
| Event | 12 | EVENT\_MSG | 0x000C |  |  |
| Config | 13 | BOARD\_RESGISTER\_REQ | 0x000D |  |  |
| 14 | BOARD\_RESGISTER\_RES | 0x000E |  |  |
| 15 | BOARD\_RESGISTER\_CONFIG\_REQ | 0x000F |  |  |
| 16 | BOARD\_RESGISTER\_CONFIG\_RES | 0x0010 |  |  |
| 17 | BOARD\_RESGISTER\_CONFIG\_INFOR | 0x0011 |  |  |
| 18 | BOARD\_RESGISTER\_FINISH | 0x0012 |  |  |
| Simulator | 19 | SIMULATION\_REQ | 0x0013 |  |  |
| 20 | SIMULATION\_RES | 0x0014 |  |  |
| 21 | SIMULATION\_CONFIG | 0x0015 |  |  |
| 22 | STOP\_SIMULATION\_REQ | 0x0016 |  |  |
| 23 | STOP\_SIMULATION\_RES | 0x0017 |  |  |
|  | 24 | CLOSE\_CONNECTION | 0x0018 |  |  |

**Length:**  chứa giá trị độ dài của cả khung truyền (đơn vị: byte)

**Data:** tùy và kịch bản mà chúng ta có các loại data khác nhau trong khung truyền USB **.**

* + - 1. **Khung truyền kiểm tra kết nối (CheckAlive Frame)**

Khung truyền này sử dụng cho kịch bản CheckAlive. Khi mới bắt đầu mô phỏng thì PC sẽ gởi khung truyền nầy xuống, yêu cầu board kiểm tra kết nối giữa board và Bus CAN và trả lại giá trị sau khi kiểm tra cho PC thông qua khung truyền CheckAlive.

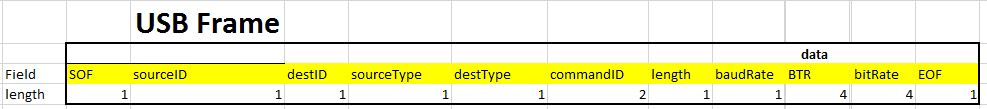


*Hình 3.2 định dạng của khung truyền Checkalive*

Khung truyền CheckAlive có cấu trúc giống với khung truyền chung, tuy nhiên trường data dùng để chứa thái của thiết bị trên Bus CAN.

* Status: trạng thái của thiết bị với Bus CAN
* Nhận giá trị 0x01 board ở trạng thái BUS\_OFF
* Nhận giá trị 0x02 board ở trạng thái Alive
  + - 1. **Khung truyền cấu hình (Configuration Frame)**

Khung truyền cấu hình dùng để lấy thông tin cấu hình giữa của Board và điều chỉnh các thống số của cấu hình trên Board, để Board có thể hoạt động được với các thông số thây đổi của nhiều Bus CAN khác nhau.



*Hình 3.3: định dạng của khung truyền cấu hình*

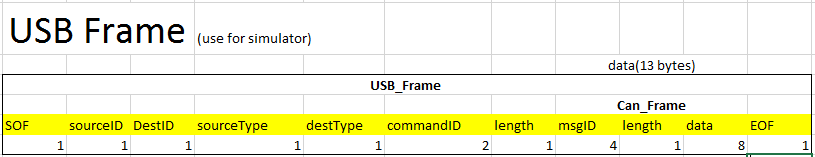
Khung truyền này chứa giá trị các thống số cấu hình của Board

* baudRate: tốc độ Baud của kênh truyền USB
* BTR: giá trị của thanh ghi Bit Timing của phần CAN trên Board

- bitRate: tốc độ của thiết bị trên Bus CAN (giá trị này chúng ta có thể thay đổi được cho phù hợp với Bus)

* + - 1. **Khung truyền mô phỏng (Simulation Frame)**

Loại khung truyền này sử dụng cho quá trình bắt đầu quá trình mô phỏng. Nó có nhiệm vụ mang gói dữ liệu mà Board lấy được từ cổng CAN và nhận các CAN\_Mesage từ PC xuống để truyền ra Bus CAN.

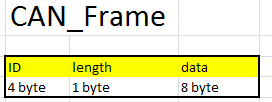


*Hình 3.4: định dạng của khung truyền mô phỏng*

Khung truyền này chủ yếu phục vụ cho việc mang giá trị của CAN Frame, sử dụng cho hoạt động mô phỏng nên chúng ta có các trường sau:

* msgID: trường ID của CAN Message
* length: độ dài của trường dataCAN Message (có giá trị từ 0 đến 8)
* data: dự liệu của CAN Message
  + - 1. **Khung truyền CAN (CAN Frame)**

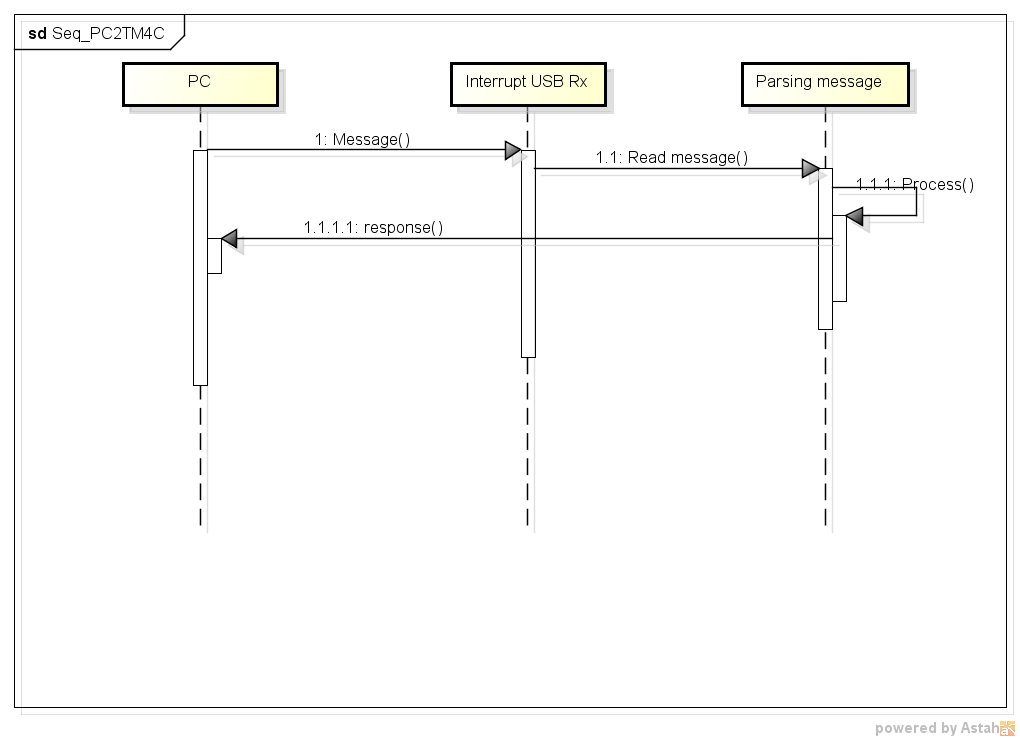
Khung truyền CAN (CAN Frame) chứa các giá trị của CAN Frame theo chuẩn CAN Standard. Tuy nhiên với mục đích tìm hiểu và mô phỏng ở mức độ cơ bản nên chúng ta chỉ thực hiện ba trường chính trong CAN Frame là: ID, length và data.



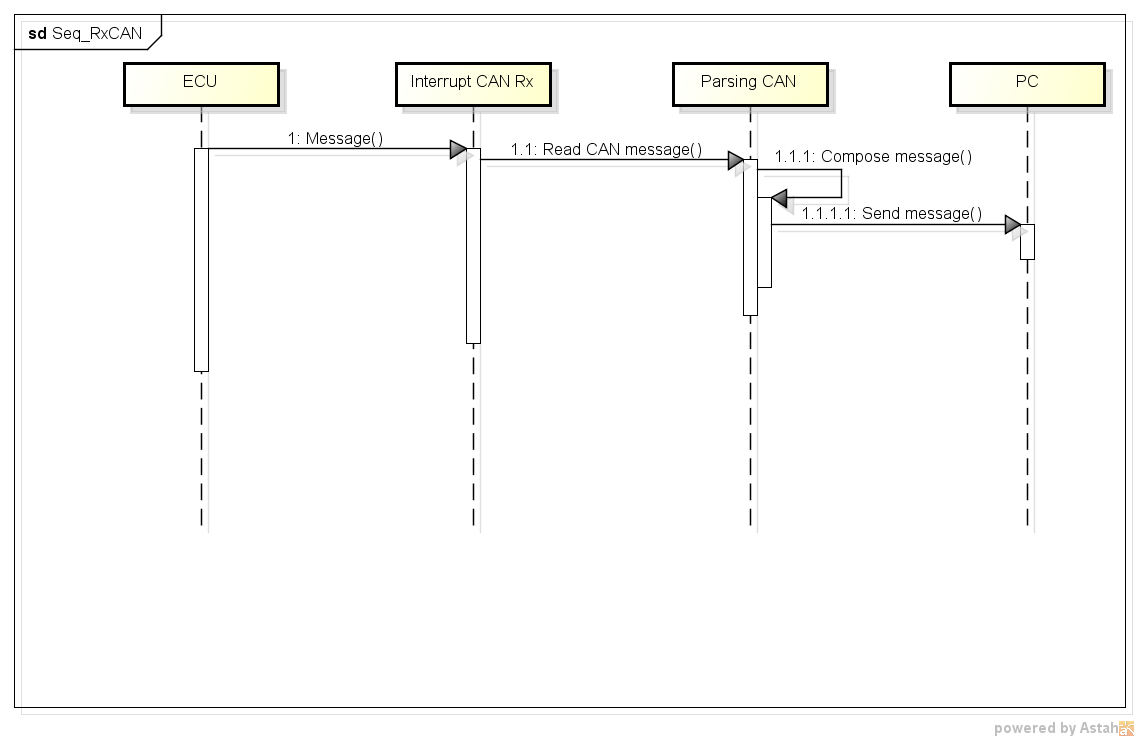
*Hình: 3.5 định dạng khung truyền của CAN*

Khung truyền này là khung dữ liệu sẽ được chúng ta sử dụng trong dự án để truyền và nhận dữ liệu trên Bus CAN

* ID : trường ID của message
* Length: độ dài của trường data (đơn vị: Byte)
* Data: dữ liệu của một CAN\_Frame.
  + 1. **Hoạt động của CAN – USB trong hệ thống**
       1. **Nhận dữ liệu của USB**



* + - 1. **Nhận dữ liệu của CAN**



* 1. **Lập trình**
     1. **Các struct sử dụng cho các khung truyền**
        1. **Khung truyền USB**
           1. **Khung truyền kiểm tra kết nối**

|  |
| --- |
| typedef struct CheckALive\_t  {  uint8\_t status; //status = 1: alive, status = 0: die  }CheckALive\_t;  typedef struct USB\_Frame\_forCheckAlive  {  uint8\_t sourceID;  uint8\_t destID;  uint8\_t sourceType;  uint8\_t destType;  uint16\_t commandID;  uint8\_t lenght;  CheckALive\_t CheckALive\_Frame;  } USB\_Frame\_forCheckAlive; |

* + - * 1. **Khung truyền cấu hình**

|  |
| --- |
| typedef struct Configuration\_t  {  uint8\_t baudRates; //BaudRates  uint8\_t BTR; //Bit Timing Register  uint32\_t bitRate; //Bit rate of CAN bus  }Configuration\_t;  typedef struct USB\_Frame\_forConfig  {  uint8\_t sourceID;  uint8\_t destID;  uint8\_t sourceType;  uint8\_t destType;  uint16\_t commandID;  uint8\_t lenght;  Configuration\_t Configuration\_Frame;  } USB\_Frame\_forConfig; |

* + - * 1. **Khung truyền mô phỏng**

|  |
| --- |
| typedef struct CAN\_Frame  {  uint32\_t ID;  uint8\_t lenght;  uint8\_t data[8];  } CAN\_Frame;  typedef struct USB\_Frame\_forSimulator  {  uint8\_t sourceID;  uint8\_t destID;  uint8\_t sourceType;  uint8\_t destType;  uint16\_t commandID;  uint8\_t lenght;  CAN\_Frame CAN\_Oject;  } USB\_Frame\_forSimulator; |

* + - 1. **Khung truyền CAN**

|  |
| --- |
| typedef struct CAN\_Frame  {  uint32\_t ID;  uint8\_t lenght;  uint8\_t data[8];  } CAN\_Frame; |

* + 1. **Hoạt động của hệ thống**
  1. **Kiểm tra và đánh giá hệ thống**
  2. **Kết luận chương**

# KẾT LUẬN

# TÀI LIỆU THAM KHẢO

[1] Tài liệu giới thiệu về giao tiếp CAN của hãng Texas Instruments - Application Report SLOA101A–August 2002–Revised July 2008 (http://www.ti.com/lit/an/sloa101a/sloa101a.pdf).

[2] Tài liểu mô tả về các chuẩn CAN của BOSCH - CAN Specification-1991, Robert Bosch GmbH, Postfach 50, D7000 Stuttgard 1. (<http://esd.cs.ucr.edu/webres/can20.pdf>).

[3] Sách “USB Complete The Developer's Guide” 3rd Edition of Jan Axelson.

[4] Datasheet của dòng chip Tiva™ TM4C123GH6PGE.

[5] Tài liệu hướng dẫn sử dụng thư viện TivaWare™ Peripheral Driver Library

[6] Tài liệu hướng dẫn sử dụng thư viện TivaWare™ USB Library

# PHỤ LỤC