CHƯƠNG 3: KẾT QUẢ THỰC HIỆN VÀ ĐÁNH GIÁ

3.1. Giới thiệu chương

Chương này trình bày tiến trình thực hiện các chức năng của phần mềm, nêu rõ từng bước thực hiện các nhiệm vụ, cùng với kết quả đạt được ở mỗi phần .Cối cùng là phần tự đánh giá kết quả đã đạt được trong thời gian thực hiện đồ án của nhóm.

3.2. Thiết kế và thi công

3.3.1. Phần giao diện

3.3.1.1. Giao diện chính:

Giao diện CAN database gồm có giao diện chính và các cửa sổ chỉnh sửa các thông điệp hoặc signal.

* **Giao diện của sổ chính:**
* **Tiến trình thực hiện:**

Giao diện màn hình chính là một cửa sổ chính sử dụng QMainWindow, trong đó có Menu các hành động như: tạo mới database, mở database có sẵn, lưu database đã mở, lưu lại database dưới dạng tên khác, thoát khỏi chương trình, chức năng hổ trợ Help,…những hành động này sẽ được mô tả bởi các QAction trong Qt và toàn bộ những QAction sẽ được gói trong QMenu tại thanh MenuBar.

* **Kết quả:**

Giao diện chính khi chạy chương trình

* Các chức năng tại thanh Menu bao gồm:

Tạo database

Mở database

Lưu database dưới dạng file .txt

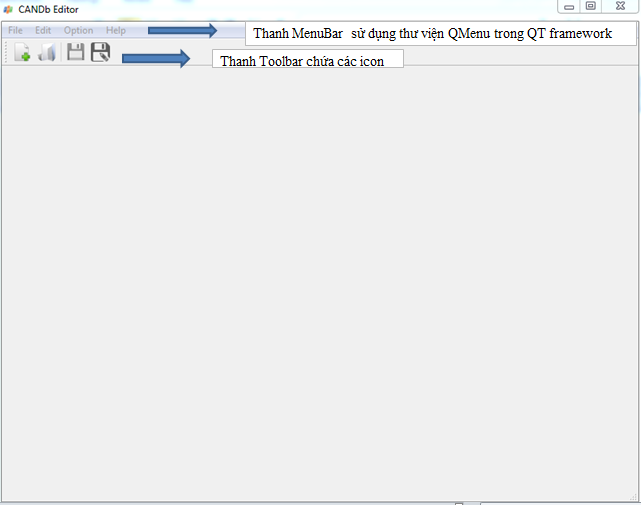
Thoát khỏi chương chình đang chạy

* F:\FPT\12-5\CANDbEditor\icons\new.pngCác chức năng được đưa ra tại thanh Toolbar dưới dạng icon bao gồm:

Tạo database :

F:\FPT\12-5\CANDbEditor\icons\saveas.pngMở database:

Lưu database



Hình 3.2. Giao diện chính CAN database

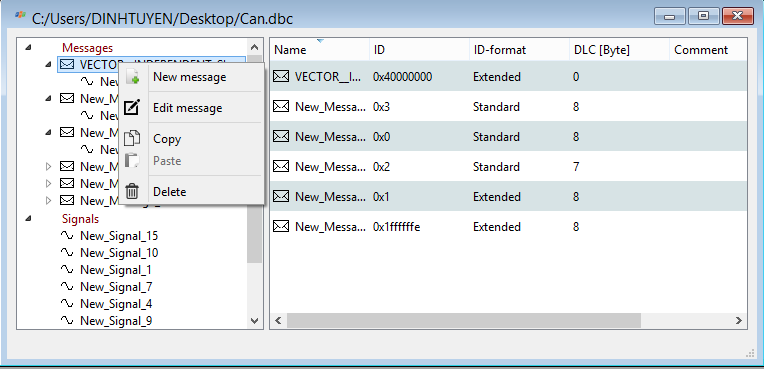
* **Giao diện chính sẽ chứa thông tin message và signal:**
* **Tiến trình thực hiện:**

Cửa sổ này được tạo từ QMdiArea và có 2 phần QTreeView và một QStackedWidget chứa các QTableView:

* QTreeView chứa danh sách tên các message và signal cũng như là các signal ở trong message có trong mạng CAN
* QStackedWidget chứa 3 QTableView tương ứng với 1 bảng chứa danh sách thoong tin chi tiết của message/signal và signal trong từng message

Giao diện này được thiết kế mô hình model/view Qt nên tất cả dữ liệu trên view đều được lấy từ model. Chúng ta sẽ tạo ra một model kế thừa từ QAbstractItemModel và tái định nghĩa các phương thức ảo trong lớp cơ sở để làm model cho phần này.

* **Kết quả:**



Hình 3.2. Giao diện CAN database sau khi load dữ liệu lên

Giao diện chính sau khi thiết kế sẽ có hình dạng như trên. Ngoài phần hiển thị dữ liệu, còn có thêm các context menu hiển thị các chức năng thêm/sửa/xóa khi click chuột phải vào một vị trí thuộc QTreeView hoặc QTableView nào đó.

3.3.1.2. Giao diện của sổ chỉnh sửa:

* **Tiến trình thực hiện:**

Phần này gồm 3 cửa sổ:

- Cửa sổ chỉnh sửa dùng cho message

- Cửa sổ chỉnh sửa dùng cho signal

- Cửa sổ chỉnh sửa dùng cho signal trong message

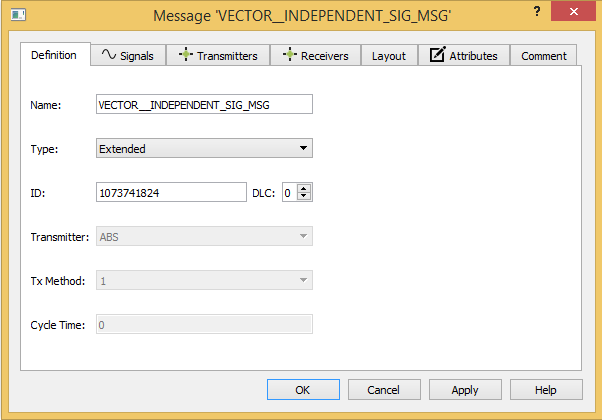
Mỗi cửa sổ sẽ là một QDialog, trong từng QDialog sẽ chứa các QTabWidget với nhiều tab chứa thông tin chi tiết về đối tượng được trỏ tới.

Để có được giao diện cân đối trong từng tab, ta sẽ sửa dụng QLayout và các lớp dẫn xuất QVBoxLayout,QHBoxLayout hay QGridLayout.

Tiêu đề của cửa sổ sẽ là tên của message/signal hoặc signal trong message tương ứng.

Mỗi một cửa sổ sẽ có các hành động: OK, Cancel, Apply, Help được tích hợp vào các QPushButton (nhờ vào signal click). Các hành động này sẽ được mô tả chi tiết trong phần thiết kế xử lí dữ liệu.

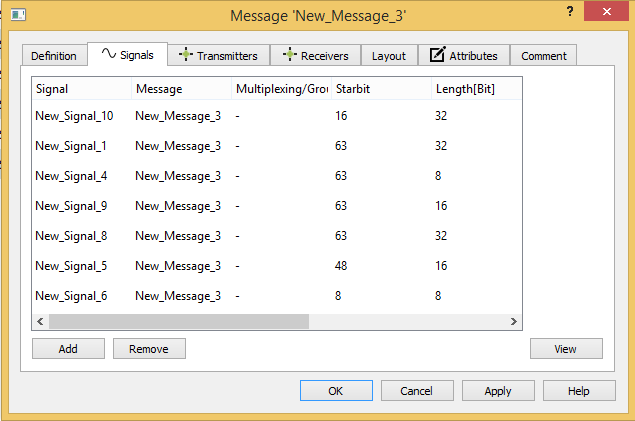
* **Kết quả:**



Hình 3.3. Giao diện cửa sổ message editor: tab definition

Để có được hộp thoại như trên và sử dụng cho tất cả các trường hợp cần chỉnh sửa thông tin message(kích phải vào context menu và click đôi vào message trên cả treeview và tableview ), ta sẽ định nghĩa một lớp MsgEditor keesthuwaf từ QDialog.

Tab definition có chứa các thông tin về đối tượng. Tab này là một widget trong đó chứa các tiêu đề là các QLabel và phần nội dung có thể là QLineEdit, QComboBox, QSpinBox được layout theo kiểu QGridLayout.



Hình 3.3. Giao diện cửa sổ message editor: tab signal

Tab signal là một widget dung QVBoxLayout để layout 2 phần: 1 QTableWidget chứa danh sách các signal nằm trong message đó, và các QPushButton Add/Remove/View để thực hiện các hành động thêm/view/xóa các signal trong message.

Ngoài 2 tab chính , còn có các tab phụ khác như Layout, Attribute, Comment…do thời gian hạn chế nên trong đồ án này chúng tôi chỉ tập trung vào các tab chứa nội dung chính.

Ở đây, chúng tôi chỉ đưa vào giao diện cửa sổ chỉnh sửa message , 2 cửa sổ còn lại được thiết kế tương tự.

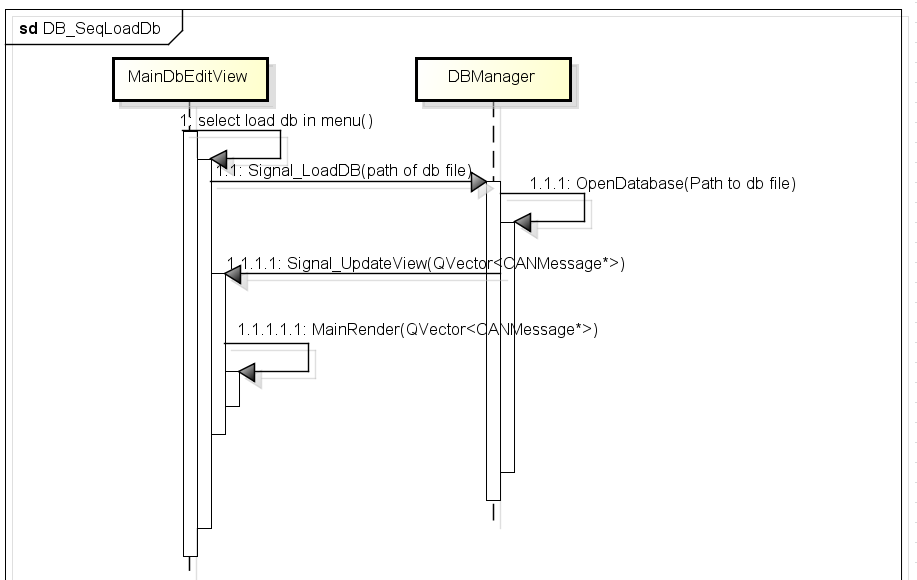
3.3.1.3 Đánh giá:

- Giao diện đã thể hiện được đầy đủ các thông tin message/signal

- Giao diện vẫn chưa hoàn thiện và đang chỉnh sửa để có giao diện đầy đủ như các khối Network, ECUs, Environment variables và có Icon dễ nhìn hơn.

3.3.2. Phần xử lí back-end:

3.3.2.1. Thực hiện các chức năng trên thanh menu màn hình chính:

* Load database

Hình 3.4. Sequence diagram LoadDB

* Trình tự đọc database lên từ file.dbc:

Bước 1: Khi click OpenDatabase 1 hộp thoại yêu cầu chọn database cần chỉnh sửa được hiển thị .

Bước 2: Sau khi đã chọn database cần chỉnh sửa thì signal Signal\_LoadDB(path) trên lớp MainDbEditView sẽ được phát ra

Bước 3:. Tại lớp DbManager sẽ có slot OpenDatabase(path) được kết nối với signal Signal\_LoadDB(path) , slot này sẽ gọi các hàm loadDB(path) để lấy thông tin từ database đưa vào các cấu trúc dữ liệu như Qvector chứa danh sách các message/signal

Bước 4 : Sau khi có được thông tin message/signal thì Signal\_updateview(Qvector<CAnMessage\*>) sẽ được gửi đi, view sẽ cập nhật thông qua slot tương ứng.

* Phân tích dữ liệu CAN database để đưa dữ liệu từ file lên:
* Ta cần phải phân tích được dữ liệu để có thể load dữ liệu 1 cách chính xác:

Phân tích để nhận diện được Message, Signal và tất cả các thông số của chúng để đưa vào các QVector<CANMessage\*> và QVector<CANSignal\*> dùng làm dữ liệu nguồn cho toàn bộ hệ thống.

* Format của 1 file .dbc
* Format của 1 Message

BO\_ 2 New\_Message\_3: 8 Vector\_\_XXX

Dòng text trên là dạng Message như sau:

BO\_ : Nhận dạng Message

ID của message với 2 loại tương ứng

BO\_ 2 New\_Message\_3: 8 Vector\_\_XXX : CAN Standard

BO\_ 2147483651 New\_Message\_4: 8 Vector\_\_XXX : CAN Extended

New\_Message\_3 : Tên message

8: Chiều dài message, tính theo Byte

Vector\_XXX: Comment

* Format của 1 Signal

SG\_ New\_Signal\_1 : 24|8@1- (1,0) [0|0] "" Vector\_\_XXX

Dòng text trên là dạng Signal như sau:

SG\_ : Nhận dạng Signal

New\_Signal\_1 : Tên của Signal

24 : StarBit

8: Chiều dài của signal, tính theo Bit

Byte Order của Signal với 2 loại tương ứng

SG\_ New\_Signal\_1 : 24|8@1- (1,0) [0|0] "" Vector\_\_XXX : Intel

SG\_ New\_Signal\_10 : 23|8@0- (1,0) [0|0] "" Vector\_\_XXX : Motorola

Value Type của Signal với 2 loại tương ứng

SG\_ New\_Signal\_11 : 32|8@1+ (1,0) [0|0] "" Vector\_\_XXX : UnSigned

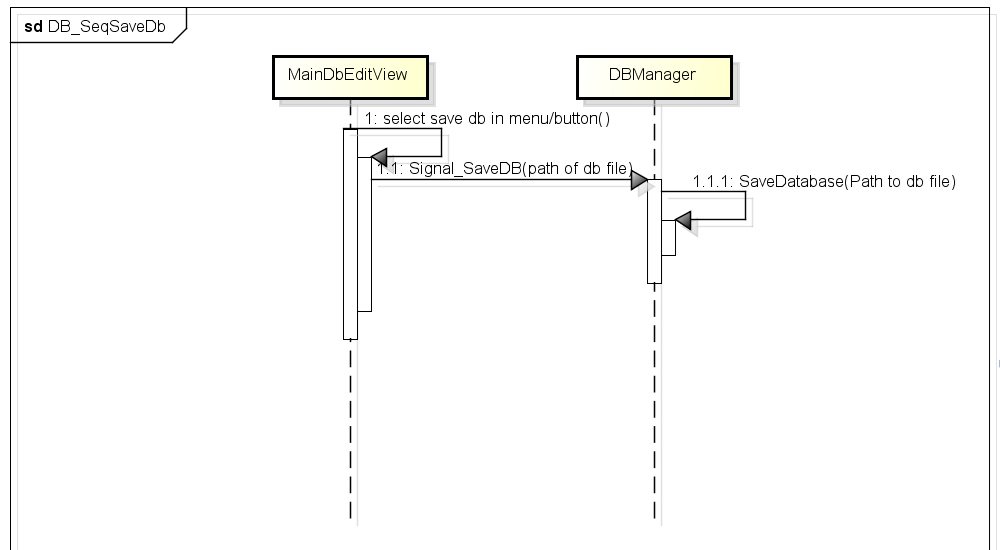
SG\_ New\_Signal\_1 : 24|8@1- (1,0) [0|0] "" Vector\_\_XXX : Signed

(1,0) : (Factor,Offset)

[0|0] : [Min|Max]

Việc đọc dữ liệu từ file được thực hiện nhờ QFile. Dữ liệu được tách thành các chuỗi và chia thành từng phần dùng hàm phương thức split với các biểu thức chính quy trong Qt.

* Kết quả: Đã đọc được database đúng định dạng message/signal và các thông số của chúng vào đúng vị trí.
* Save database
* Tiến trình thực hiện:



Hình 2.4. Sequence diagram SaveDB

Bước 1: Tương tác của người dùng sẽ được thực hiện trên lớp view, khi chọn save database thì signal Signal\_SaveDB(path) sẽ được gửi đi

Bước 2: Tại lớp DBManager, một slot tương ứng với signal Signal\_SaveDB(path) là SaveDatabase(path) , tại đây dữ liệu sẽ được lưu lại tại file có đường dẫn path

Dữ liệu được lưu lại chính là dữu liệu trong vector QVector<CANMessage\*> đã được chỉnh sửa trong quá trình tương tác giữa người dùng với giao diện. Như vậy, database sẽ được thay đổi đúng theo mục đích của người dùng.

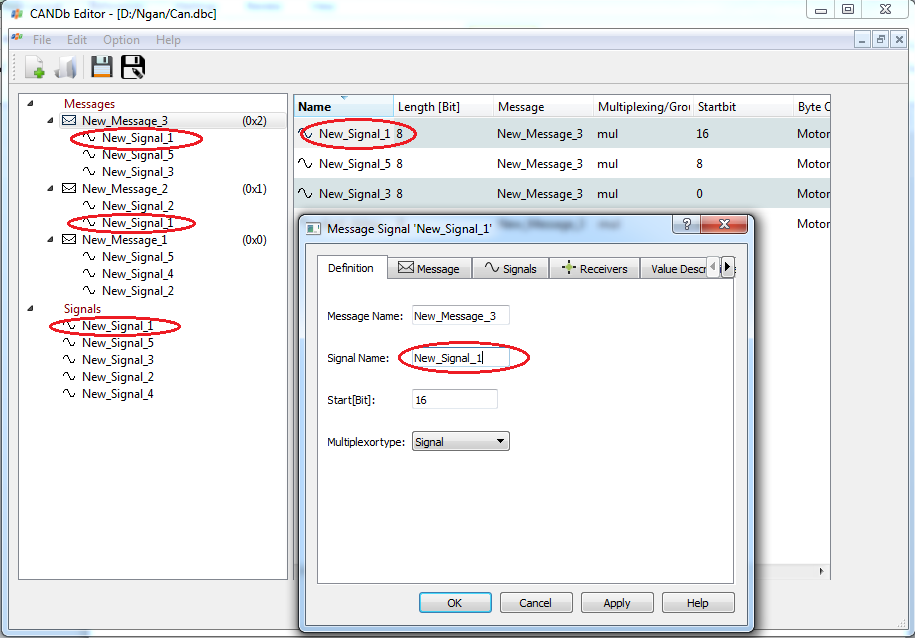
* Kết quả:
* Đã save được database gần giống với định dạng của file .bdc, tuy chưa thể thực hiện giống hoàn toàn do còn một số chức năng chưa được hoàn thiện nhưng database vẫn thể hiện được các nội dung chính gồm danh sách các message/signal
* Đánh giá
* Thực hiện được các chức năng như mở file, tạo file và lưu file
* Khi load database, giữ liệu đã đọc được chính xác với file .dbc có sẵn, các thông tin về message, signal đã được lưu trữ đúng vị trí

// Ngân viết thêm hi

3.3.2 Phần edit cơ sở dữ liệu

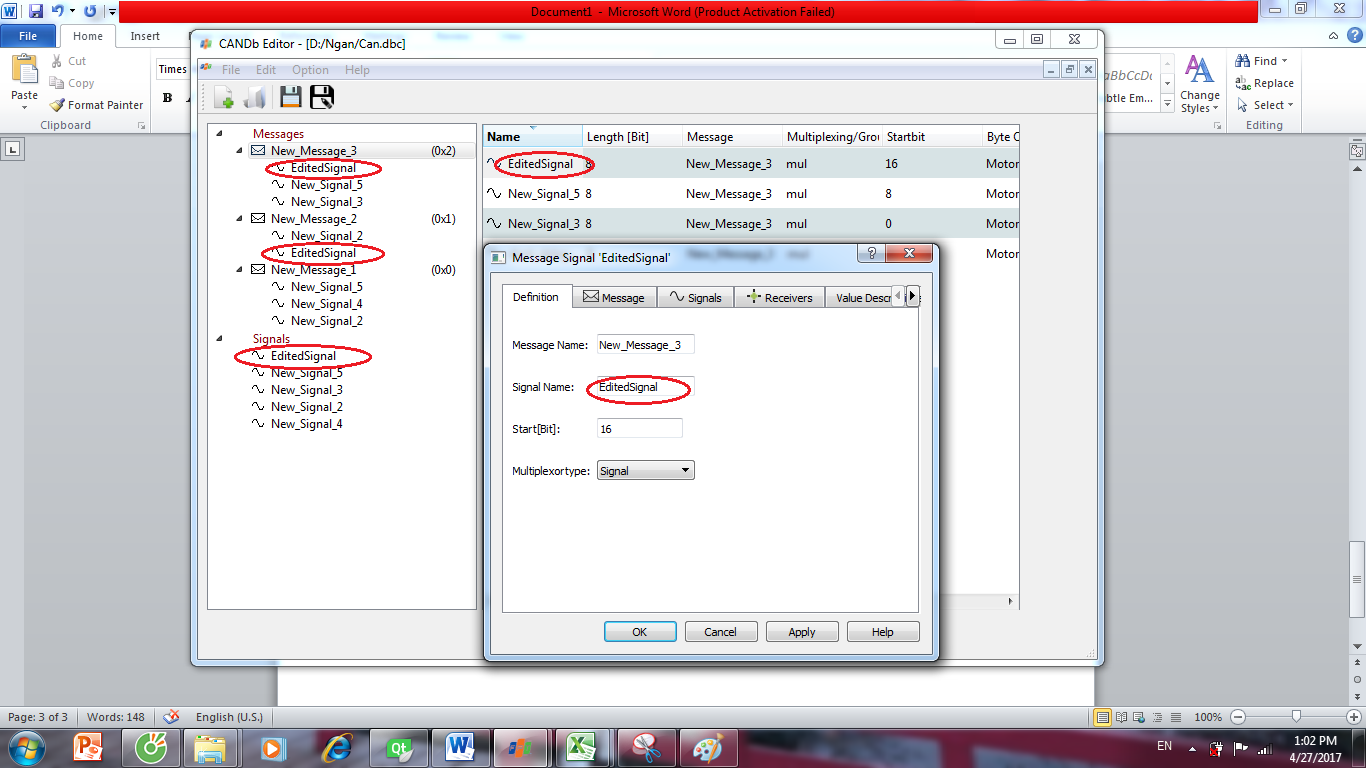
3.3.2.1. Kết quả

* Chỉnh sửa
* Khi ta click đôi chuột pải vào table view thông tin cụ thể của các signal sẽ được hiển thị tại 1 dialog khác để tiện theo dõi vào chỉnh sửa



Hình 3.6. Thông tin Signal trước khi edit

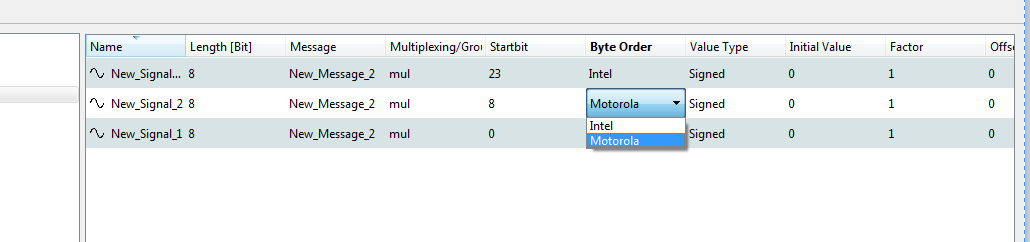
* Sau khi chỉnh sửa phần Signal Name tại Dialog và nhấn OK,
* Thông tin của Signal đó lập tức thay đổi ở các vị trí:
* Tại Table View
* Tại Tree View
* Tại Source



Hình 3.7. Thông tin Signal sau khi edit

* Tương tự đối với các thông tin khác của Message hoặc signal ta đều có thể thay eidt được tại phần mềm

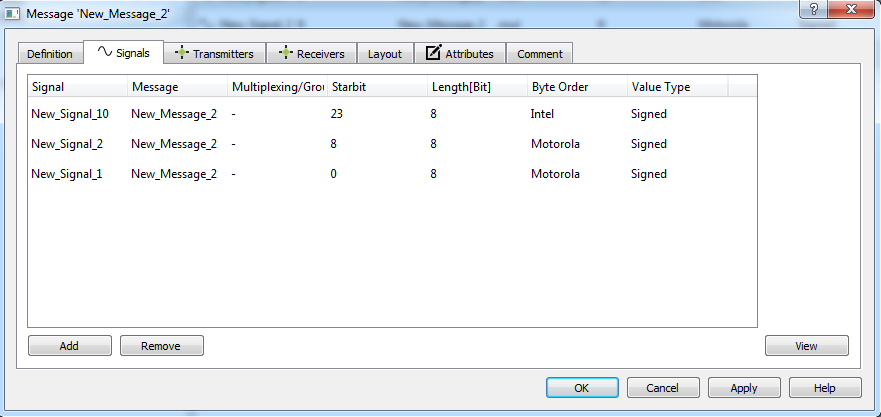
Ngoài việc chỉnh sửa thông qua các Tab, các thông tin có thể được chỉnh sửa bằng cách Delegate trực tiếp tại các Table bởi các Combobox trong QCombobox của Qt



Hình 3.8. Delegate dữ liệu

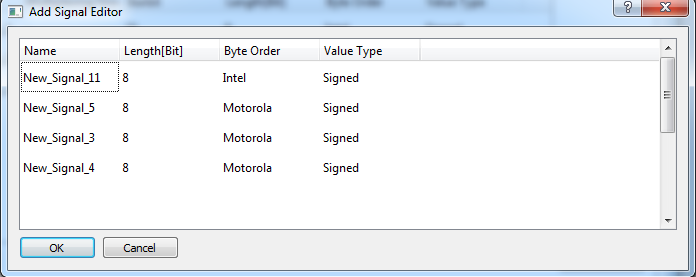
* Thêm

Tại mỗi dialog message tương ứng, ta chuyển tab sang tab Signal, thông tin của các Signal thuộc Message đó sẽ hiện ra dưới dạng Table



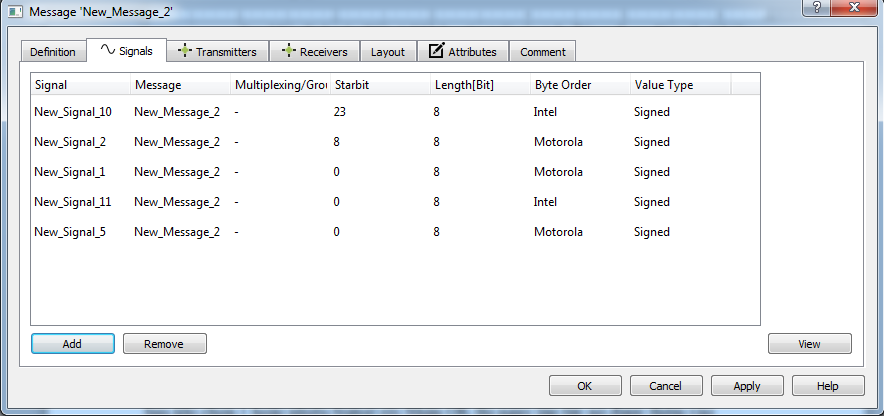
Hình 3.9. List Signal trước khi Add

Sau khi click vào Nút Add bên góc trái của tab, Bảng them Signal sẽ hiện ra với các Signal ngoài Signal đã thuộc Message(Vì 1 signal không thể thuộc 1 message 2 lần)



Hình 3.10. Bảng Thêm Signal

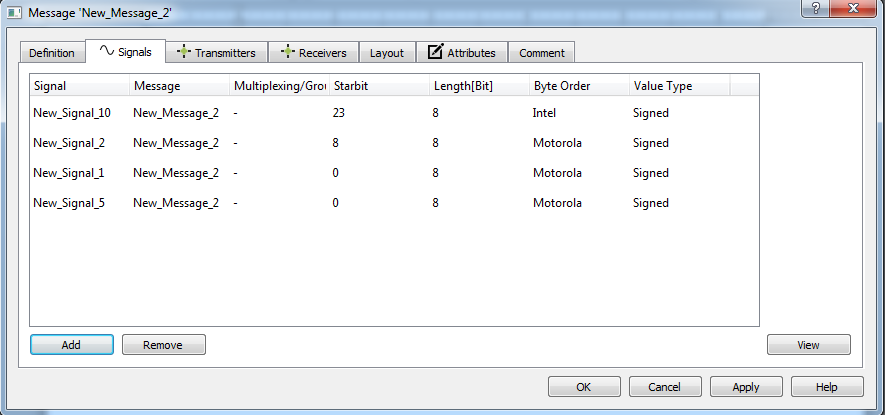
Sau khi chọn 1 hoặc nhiều Signal rồi Nhấn OK thì ngay lập tức nó được thêm vào Message



Hình 3.11. List Signal sau khi được thêm

* Xóa

Khi chọn 1 Signal và Nhấn OK, Signal đó sẽ được xóa khỏi danh sách Signal trong bảng



Hình 3.12. List Signal sau khi xóa Signal

3.2.2. Đánh giá

* Đã phân tích được đúng format của các Message và Signal để load đúng dữ liệu vào các Tab
* Dữ liệu của các Message và Signal đã được edit và up date
* Đã Delegate được dữ liệu tại các Table
* Hoạt động Thêm và Xóa đã hoạt động đúng mục đích

3.4. Quá trình chạy thử và sửa lỗi

Tuyến Viết khúc này hộ Ngân

3.5. Hướng phát triển của đề tài

* Đề tài này được thực hiện trong vòng 3 tháng và hiện tại vẫn còn rất nhiều chức năng chưa hoàn thiện, trong tương lai đề tài sẽ được củng cố các chức năng hiện tại, cũng như phát huy thêm nhiều chức năng mới như Setting và Tại TreeView sẽ cung cấp thêm các Header như Network, ECU hoặc Biến môi trường v.v
* Phần mềm sẽ được kết hợp với các phần mềm Simulator và Hardware để nhận các Message trực tiếp từ các ECU và view lên theo mong muốn

3.6. Kết luận chương

Các kết quả thực hiện đã được trình bày và hệ thống được xem là hoàn thiện các chức năng cơ bản có thể hoạt động tốt. Tuy rằng ban đầu có xuất hiện rất nhiều lỗi nhưng sau quá trình chỉnh sửa chúng tôi đã hoàn thành được đề tài, thực hiện được đúng các chức năng Thêm, Sửa, Xóa