CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

One aspect of the project aims at developing a traffic control algorithm for future technology. The design of the traffic control system can be evaluated in two steps – synthesis and analysis. Several models and multiple control strategies exist, and engineers must decide between them using a priori knowledge of the real system. Previously collected information can help to choose the appropriate model, parameters, measurement and control methodologies to create the optimal solution.

1.2 OBJECTIVE OF PROJECT

A newly emerged area is demand estimation through microscopic traffic modelling. The dynamic aspect of traffic simulation requires estimation volumes of traffic continuously. For instance, the observation of constantly varying turning rates.

According to the development of our project it is a step by step analysis of traffic system. At first we consider two routes (i.e. East-West and North-South) have equal volumes of traffic at all time. So both the routes are assigned equal clearance time.

But for some special cases we require human interruption for clearing traffic at some time. Hence a manual mode is essential that is built into the system and can be switched by the traffic police as and when desired.

Emergency sensor is built upon, to continuously check the emergency situations like ambulance, police, and fire-brigade etc providing immediate clearance to it irrespective of present state. After clearance the state is resumed

1.3 APLICATIONS & LIMITATIONS

1. the system performs continuous monitoring of traffic flow parameters in the city road network with the help of multiple vehicle detectors of all types (inductive, radar, infrared and video detectors);
2. Adaptive traffic control is performed according to measured or calculated (on the basis of measurements) traffic flow parameters;

3. Drivers are informed about the recommended speed, according to the current control plans, with the help of electronic speed indicators;
4. Drivers and pedestrians are informed about the duration of the green and red light signals, existing or potential traffic jams and possible bypasses.

1.4 THESIS:

Chapter 1: includes overview, objective of the project, applications.

Chapter 2: includes implementation, tools and platform used.

Chapter 3: includes technology, IBM rational rhapsody code,report onautomated road traffic control,panal diagram,seqance diagram,animated diagram diagram,stae chart.

Chapter 4: includes lpc 2138 arm controller, ARM architecture.

Chapter 5: includes keil software, flash magic, and keil code.

Appendices : includes software coding implementation code

CHAPTER 2
IMPLEMENTATION

2.1 What is an Embedded System?

An embedded system is an application that contains at least one programmable computer (typically in the form of a microcontroller, a microprocessor or digital signal processor chip) and which is used by individuals who are, in the main, unaware that the system is computer-based.

Typical examples of embedded applications that are constructed using the techniques discussed in this book include:

- **Mobile phone systems** (including both customer handsets and base stations).
- **Automotive applications** (including braking systems, traction control, airbag release systems, engine-management units, steer-by-wire systems and cruise control applications).
- **Domestic appliances** (including dishwashers, televisions, washing machines, microwave ovens, video recorders, security systems, garage door controllers).
- **Aerospace applications** (including flight control systems, engine controllers, autopilots and passenger in-flight entertainment systems).
- **Medical equipment** (including anesthesia monitoring systems, ECG monitors, drug delivery systems and MRI scanners).
- **Defence systems** (including radar systems, fighter aircraft flight control systems, radio systems and missile guidance systems)

2.2 TOOLS AND PLATFORM USED

Tools : Rational Rhapsody 7.5.2

Platform: Rational Rhapsody 7.5.2

2.3.1 Hardware Requirement

1. Intel Pentium 1.5 GHZ

2. 1 GB main memory.
3. 3 GB of free hard disk space.
4. 14” or bigger monitor.
5. Mouse.
6. Standard Keyboard

2.3.2 Software Requirement

1. **Operating System:** Windows XP
2. **Compiler:** MinGW

CHAPTER 3
TECHNOLOGIES USED

3.1 UNIFIED MODELLING LANGUAGE

The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components.

Thus, UML is a 'language' for specifying and not a method or procedure. The UML is used to define a software system; to detail the artifacts in the system, to document and construct - it is the language that the blueprint is written in. The UML may be used in a variety of ways to support a software development methodology (such as the Rational Unified Process) - but in itself it does not specify that methodology or process.

There are 14 kinds of UML diagrams:

1. Class Diagram
2. Object Diagram
3. Deployment
4. Component
5. Package
6. Profile
7. Structure
8. Use Case
9. Activity
10. Sequence
11. State Machine
12. Communication
13. Interaction Overview
14. Timing

Class Diagram:

The class diagram is the main building block of object oriented modelling. It is used both for general conceptual modelling of the systematics of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects and or interactions in the application and the objects to be programmed. In the class diagram these classes are represented with boxes which contain three parts.
A class has 3 sections:

- The upper part holds the name of the class
- The middle part contains the attributes of the class
- The bottom part gives the methods or operations the class can take or undertake.

Object Diagram:

An Object diagram focuses on some particular set of object instances and attributes, and the links between the instances. A correlated set of object diagrams provides insight into how an arbitrary view of a system is expected to evolve over time. Object diagrams are more concrete than class diagrams, and are often used to provide examples, or act as test cases for the class diagrams. Only those aspects of a model that are of current interest need be shown on an object diagram.

Use Case Diagram:

A use case defines the interactions between external actors and the system under consideration to accomplish a goal. Actors must be able to make decisions, but need not be human: An actor might be a person, a company or organization, a computer program, or a computer system — hardware, software, or both. Actors are always stakeholders, but many stakeholders are not actors, since they never interact directly with the system, even though they have the right to care how the system behaves.

Sequence Diagram:

A sequence diagram in a Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the

scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams typically are associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

State-Machine Diagram:

A sequence diagram in a Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams typically are associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

SysML

The Systems Modeling Language (SysML) is a general-purpose modeling language for systems engineering applications. It supports the specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems.

3.2 IBM RATIONAL RHAPSODY

IBM® Rational® Rhapsody® family provides collaborative design and development for systems engineers and software developers creating real-time or embedded systems and software. Rational Rhapsody helps diverse teams collaborate to understand and elaborate requirements, abstract complexity visually using industry standard languages (UML, SysML, AUTOSAR, DoDAF, MODAF, UPDM), validate functionality early in development, and automate delivery of innovative, high quality products.

3.3 Rhapsody Report on Automated Road Traffic Control

Object Model Diagram Information

Object Model Diagram name: Model1

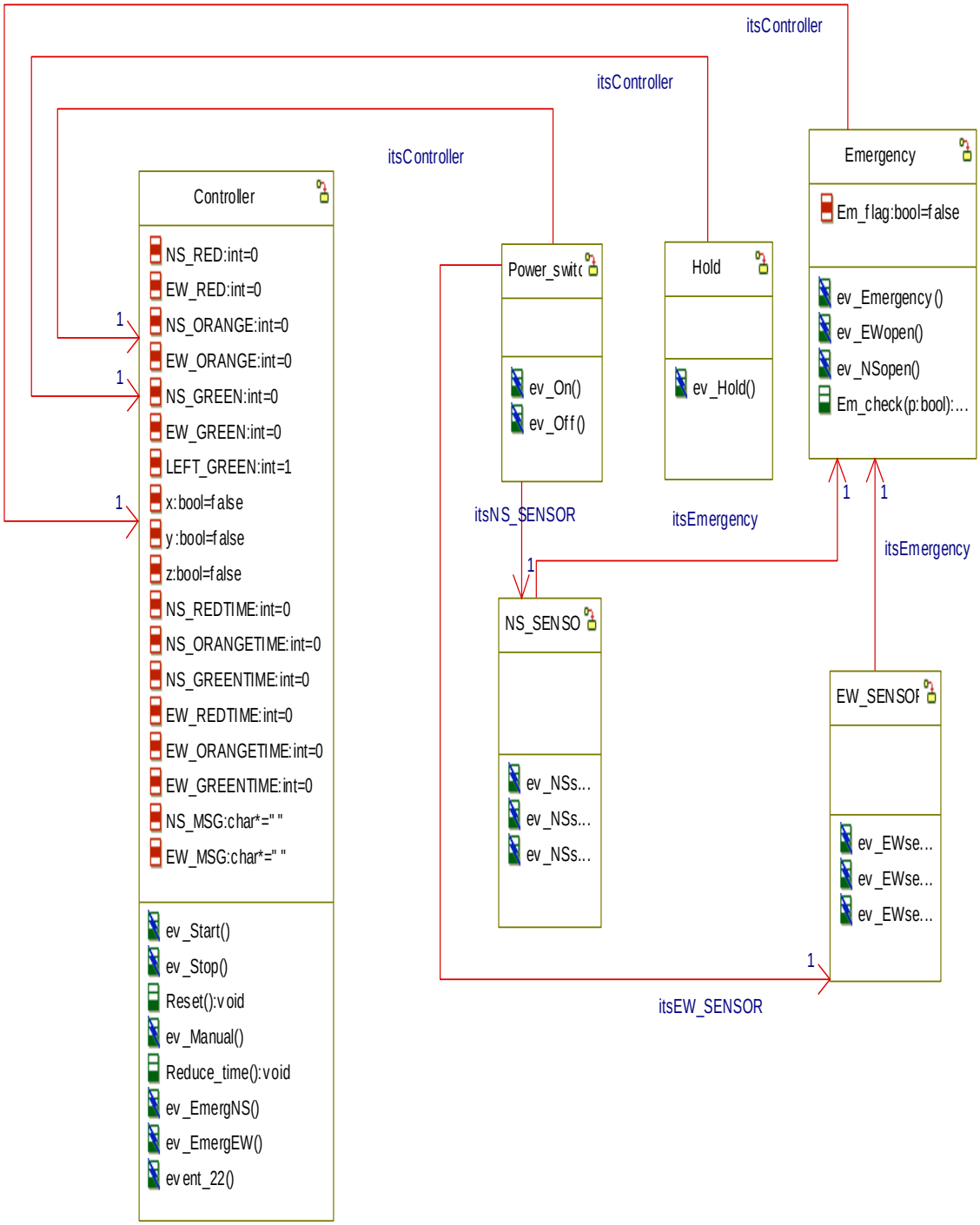


Fig 3.1 : Object Model Diagram Information

Object Model Diagram name: density

Fig 3.2: Object Model Diagram name: density

Object Model Diagram name: Normal_Indian

Description:

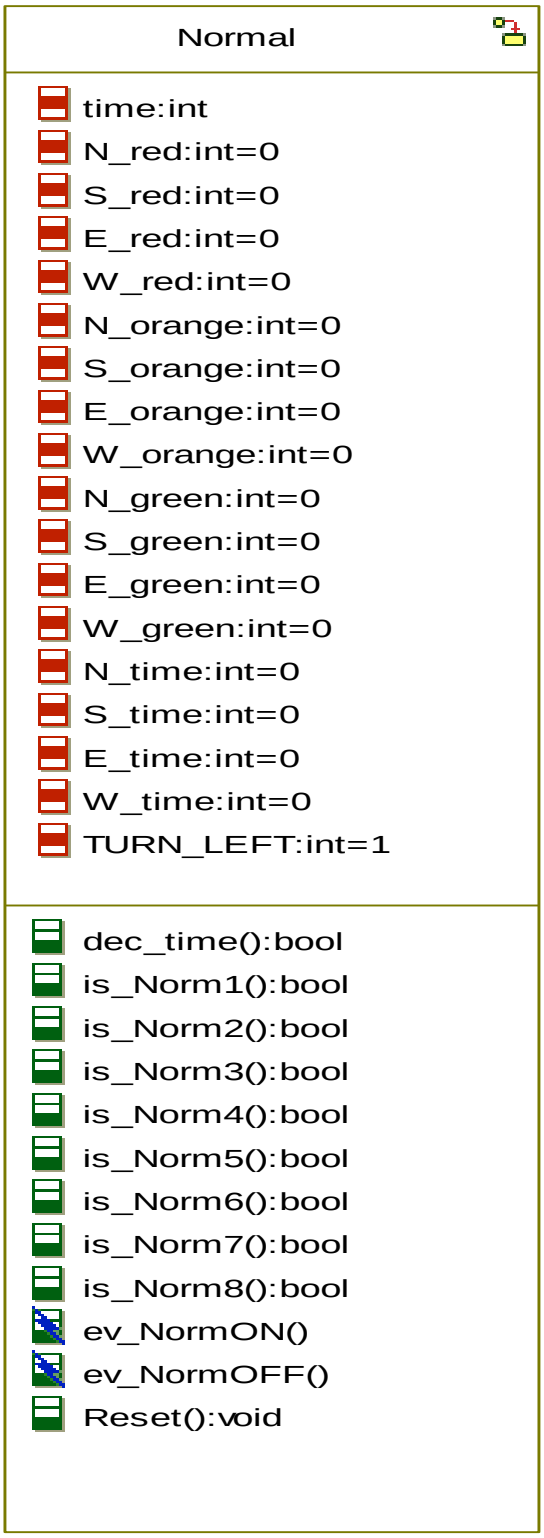


Fig 3.3 : Object Model Diagram name: Normal_Indian

3.4 Components Information

Component Name: Default Component

Type: executable

Directory :C:\Documents and

Settings\Administrator\Desktop\Traffic4\DefaultComponent\DefaultConfig

Libraries:

Additional Sources:

Standard Headers:

Include Path:

Description:

Configuration information for Component: DefaultComponent

DefaultConfig Configuration

Configuration Name: DefaultConfig

Description:

Initialization Scope:explicit

Initialization Code:

Directory:DefaultComponent\DefaultConfig

Libraries:

Additional Sources:

Standard Headers

Include Path:

Instrumentation:animate

Time Model:real

Statechart Implementation:flat

BuildSet: Debug

Compiler Switches: $IncludeDirectories $DefinedSymbols $(INST_FLAGS)

$(INCLUDE_PATH) $(INST_INCLUDES) $CompilerFlags

$OMCPPCompileCommandSet -c

Link Switches:$OMLinkCommandSet $LinkerFlags

Class name: Controller

Description:

Active: false

Behavior Overridden: false

Composite: false

Reactive: true

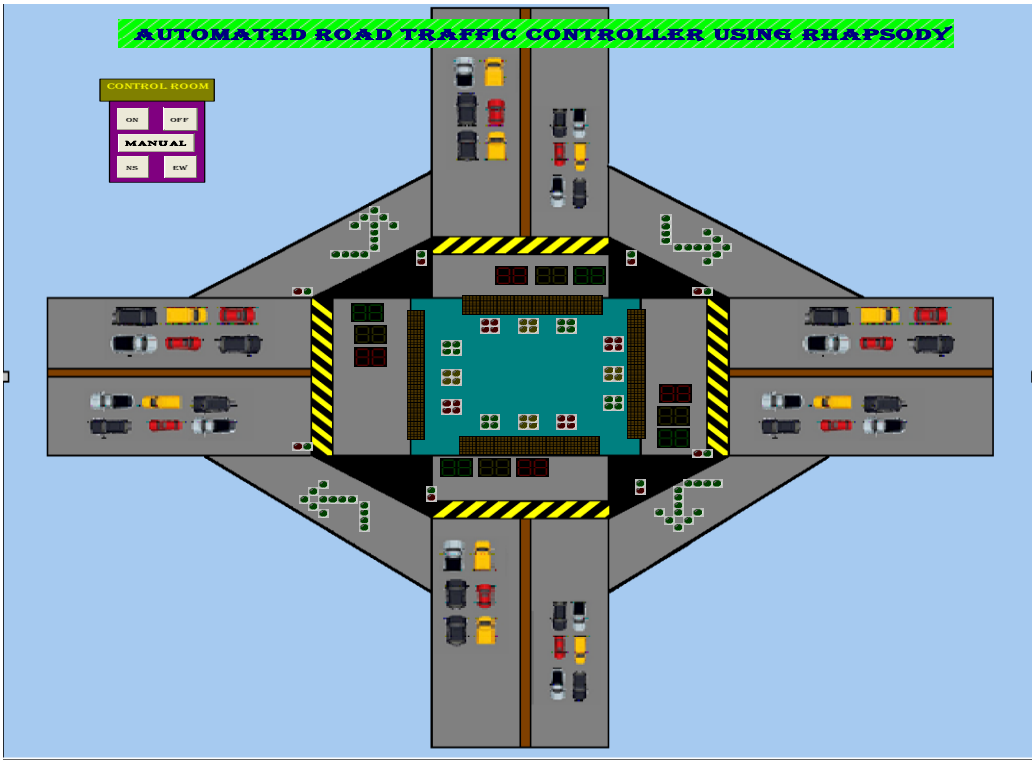


Fig3.4: Traffic density panel diagram

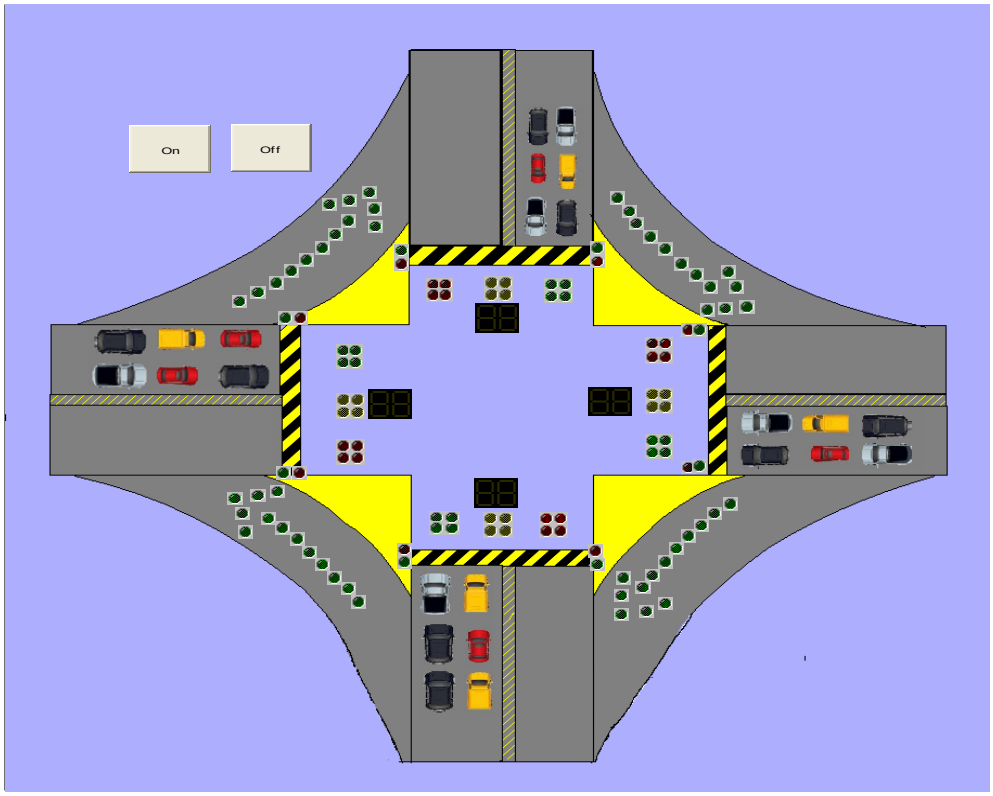


Fig3.5 : Panel Diagram name: Normal_Indianpaneldiagram

Panel Diagram

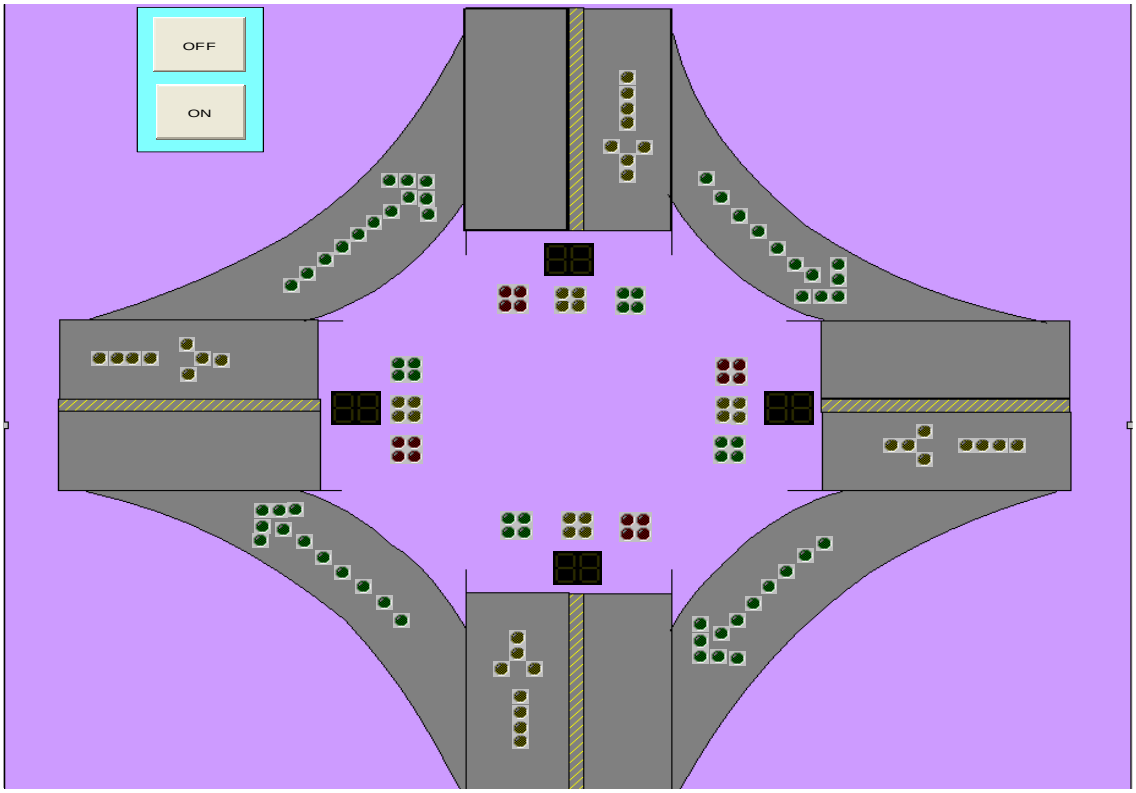


Fig3.6 : panel diagram

CHAPTER 4

PROJECT INFORMATION

4.1 Attribute Information For Class: controller

Attribute Name: NS_RED

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: EW_RED

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: NS_ORANGE

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: EW_ORANGE

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: NS_GREEN

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: EW_GREEN

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: LEFT_GREEN

Default Value: 1

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: x

Default Value: false

Static: false

Visibility: public

Type: bool

Stereotype:

Description:

Attribute Name: y

Default Value: false

Static: false

Visibility: public

Type: bool

Stereotype:

Description:

Attribute Name: z

Default Value: false

Static: false

Visibility: public

Type: bool

Stereotype:

Description:

Attribute Name: NS_REDTIME

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: NS_ORANGETIME

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: NS_GREENTIME

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: EW_REDTIME

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: EW_ORANGETIME

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: EW_GREENTIME

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: NS_MSG

Default Value: " "

Static: false

Visibility: public

Type: char*

Stereotype:

Description:

Attribute Name: EW_MSG

Default Value: " "

Static: false

Visibility: public

Type: char*

Stereotype:

Description:

4.2 Operation information for Class: Controller

Operation name: Reduce_time

Initializer:

Const: false

Trigger: false

Body: if(NS_REDTIME>0)

NS_REDTIME--;

if(NS_ORANGETIME>0)

NS_ORANGETIME--;

if(NS_GREENTIME>0)

NS_GREENTIME--;

if(EW_REDTIME>0)

EW_REDTIME--;

if(EW_ORANGETIME>0)

EW_ORANGETIME--;

if(EW_GREENTIME>0)

EW_GREENTIME--;

Abstract: false

Static: false

Virtual: false

Visibility: public

Signature: Reduce_time()

Return Type: void

Description:

Operation name: Reset

Initializer:

Const: false

Trigger: false

Body: NS_RED=0;

EW_RED=0;

NS_GREEN=0;

EW_GREEN=0;

NS_ORANGE=0;

EW_ORANGE=0;

Abstract: false

Static: false

Virtual: false

Visibility: public

Signature: Reset()

Return Type: void

Description:

4.3 Event Reception information for Class: Controller

Event Reception name: ev_Start

Signature: ev_Start()

Description:

Event Reception name: ev_Stop

Signature: ev_Stop()

Description:

Event Reception name: ev_Manual

Signature: ev_Manual()

Description:

Event Reception name: ev_EmergNS

Signature: ev_EmergNS()

Description:

Event Reception name: ev_EmergEW

Signature: ev_EmergEW()

Description:

Event Reception name: event_22

Signature: event_22()

Description:

4.4 Statechart information for Class: Controller

Description:

Overridden: false

Class name: Power_switch

Description:

Active: false

Behavior Overridden: false

Composite: false

Reactive: true

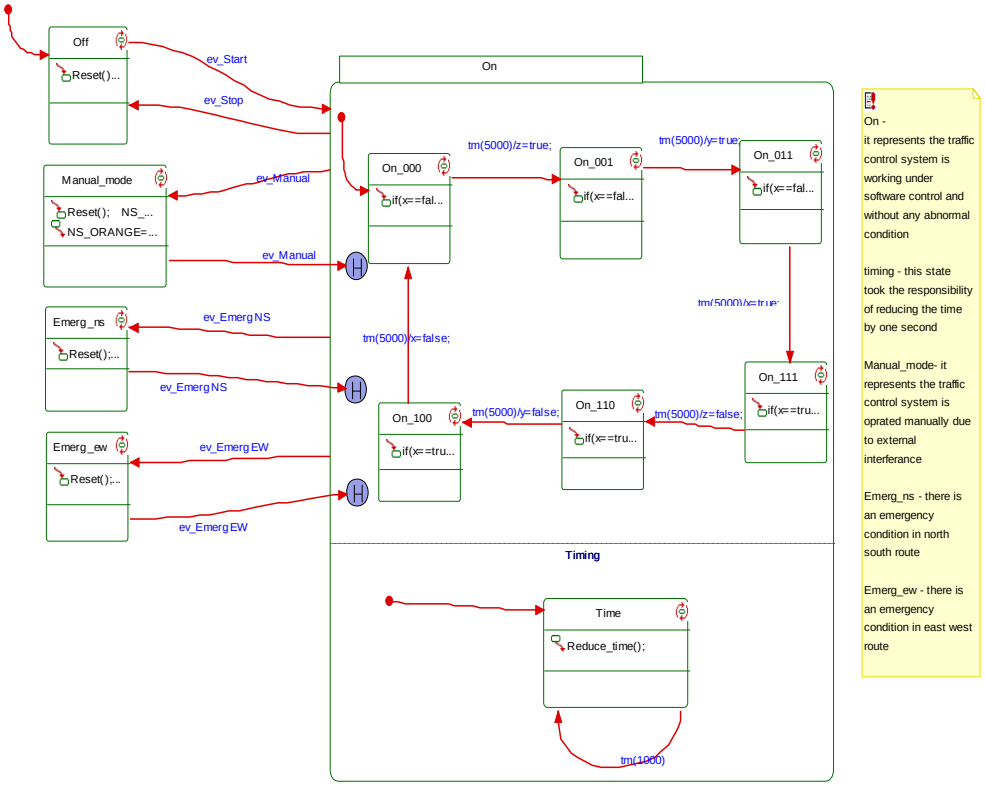


Fig 4.1 :Controller

4.2.2 Event Reception information for Class: Power_switch

Event Reception name: ev_On

Signature: ev_On()

Description:

Event Reception name: ev_Off

Signature: ev_Off()

Description:

Name	**Inverse**	**Source**	**Target**
itsController		Power_switch	Controller
itsNS_SENSOR		Power_switch	NS_SENSOR
itsEW_SENSOR		Power_switch	EW_SENSOR

Statechart information for Class: Power_switch

Description:

Overridden: false

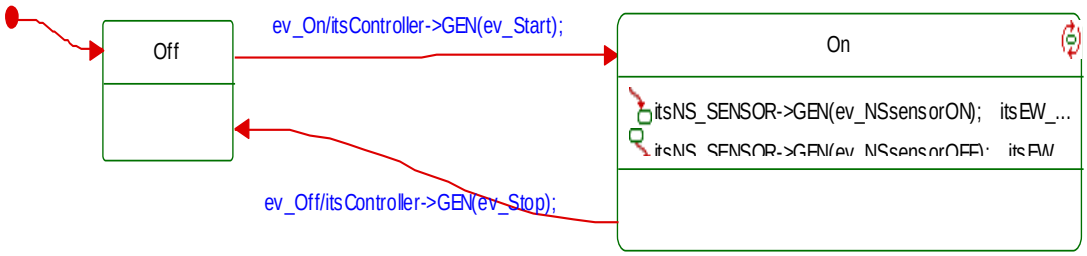


Fig 4.2 :power switch

Class name: Hold

Description:

Active: false

Behavior Overridden: false

Composite: false

Reactive: true

EventReception information for Class: Hold

Event Reception name: ev_Hold

Signature: ev_Hold()

Description:

Relation information for Class: Hold

Name	Inverse	Source	Target
itsController		Hold	Controller

Statechart information for Class: Hold

Description:

Overridden: false

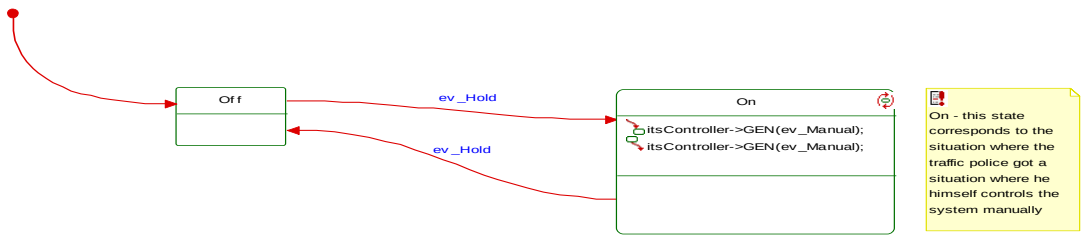


Fig4.3: hold

Class name: Emergency

Description:

Active: false

Behavior Overridden: false

Composite: false

Reactive: true

Attribute Information for Class: Emergency

Attribute Name: Em_flag

Default Value: false

Static: false

Visibility: public

Type: bool

Stereotype:

Description:

Operation information for Class: Emergency

Operation name: Em_check

Initializer:

Const: false

Trigger: false

Body: Em_flag=p;

Abstract: false

Static: false

Virtual: false

Visibility: public

Signature: Em_check(bool p)

Return Type: void

Description:

Argument information for Operation Em_check

Name	Type	Direction
P	bool	In

EventReception information for Class: Emergency

Event Reception name: ev_Emergency

Signature: ev_Emergency()

Description:

Event Reception name: ev_EWopen

Signature: ev_EWopen()

Description:

Event Reception name: ev_NSopen

Signature: ev_NSopen()

Description:

Relation information for Class: Emergency

Name	Inverse	Source	Target
itsController		Emergency	Controller

Statechart information for Class: Emergency

Description:

Overridden: false

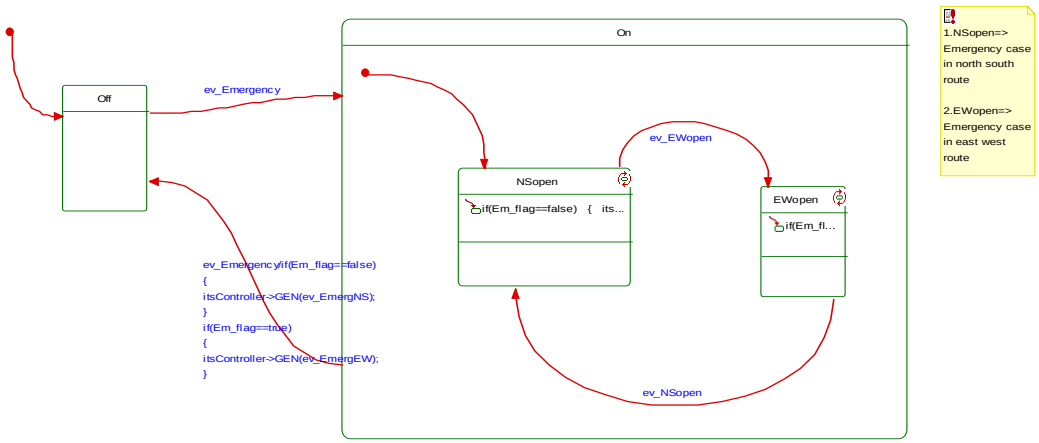


Fig 4.4 : Emergency

Class name: Traffic_Main

Description:

Active: false

Behavior Overridden: false

Composite: true

Reactive: true

Class name: NS_SENSOR

Description:

Active: false

Behavior Overridden: false

Composite: false

Reactive: true

EventReception information for Class: NS_SENSOR

Event Reception name: ev_NSsensorON

Signature: ev_NSsensorON()

Description:

Event Reception name: ev_NSsensorOFF

Signature: ev_NSsensorOFF()

Description:

Event Reception name: ev_NSsense

Signature: ev_NSsense()

Description:

Relation information for Class: NS_SENSOR

Name	Inverse	Source	Target
itsEmergency		NS_SENSOR	Emergency

Statechart information for Class: NS_SENSOR

Description:

Overridden: false

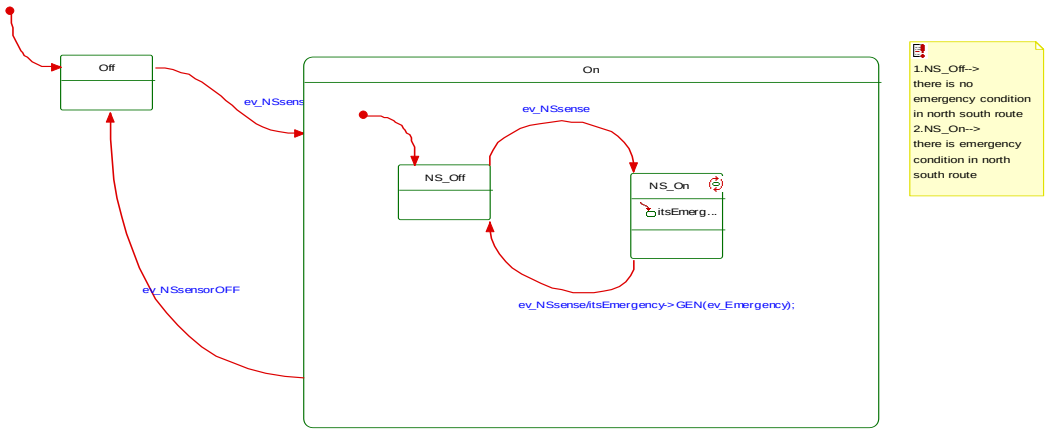


Fig 4.5: NS sensor

Class name: EW_SENSOR

Description:

Active: false

Behavior Overridden: false

Composite: false

Reactive: true

EventReception information for Class: EW_SENSOR

Event Reception name: ev_EWsensorON

Signature: ev_EWsensorON()

Description:

Event Reception name: ev_EWsensorOFF

Signature: ev_EWsensorOFF()

Description:

Event Reception name: ev_EWsense

Signature: ev_EWsense()

Description:

Relation information for Class: EW_SENSOR

Name	Inverse	Source	Target
itsEmergency		EW_SENSOR	Emergency

Statechart information for Class: EW_SENSOR

Description:

Overridden: false

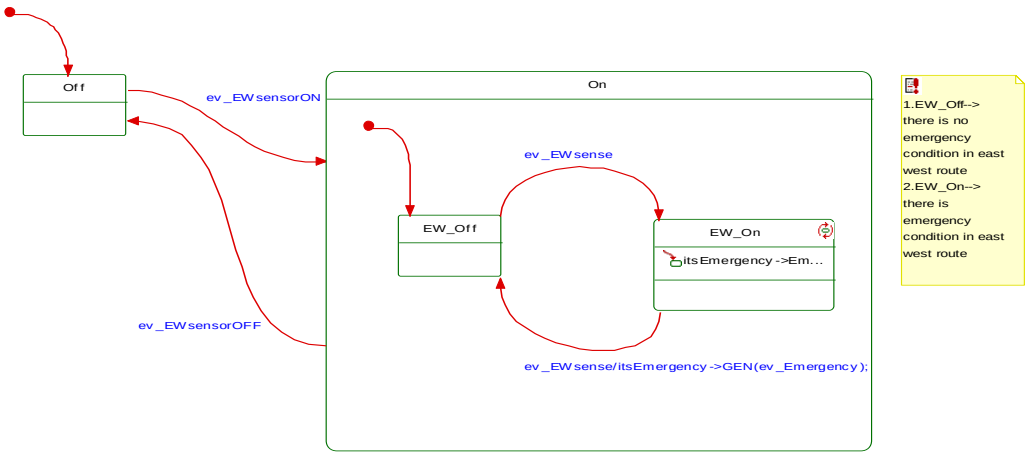


Fig 4.6 : ew sensor

Class name: controller_density

Description:

Active: false

Behavior Overridden: false

Composite: false

Reactive: true

Attribute Information for Class: controller_density

Attribute Name: time Static: false

Default Value: Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: NS_red

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name:

NS_green

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: NS_orange

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: EW_orange

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: EW_green

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: EW_red

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: d

Default Value: true

Static: false

Visibility: public

Type: bool

Stereotype:

Description:

Attribute Name: free_left

Default Value: 1

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Operation information for Class: controller_density

Operation name: Reset

Initializer:

Const: false

Trigger: false

Body: NS_red=0;

NS_orange=0;

NS_green=0;

EW_red=0;

EW_orange=0;

EW_green=0;

Abstract: false

Static: false

Virtual: false

Visibility: public

Signature: Reset()

Return Type: void

Description:

Operation name: is_Change1

Initializer:

Const: false

Trigger: false

Body: if(time==0)

return(true);

else

return(false);

Abstract: false

Static: false

Virtual: false

Visibility: public

Signature: is_Change1()

Return Type: bool

Description:

Operation name: is_Change2

Initializer:

Const: false

Trigger: false

Body: if(time==0)

return(true);

else

return(false);

Abstract: false

Static: false

Virtual: false

Visibility: public

Signature: is_Change2()

Return Type: bool

Description:

Operation name: set_status

Initializer:

Const: false

Trigger: false

Body: d=n;

Abstract: false

Static: false

Virtual: false

Visibility: public

Signature: set_status(bool n)

Return Type: void

Description:

Argument information for Operation set_status

Name	Type	Direction
N	bool	In

EventReception information for Class: controller_density

Event Reception name: ev_Den

Signature: ev_Den()

Description:

Event Reception name: ev_DenON

Signature: ev_DenON()

Description:

Event Reception name: ev_DenOFF

Signature: ev_DenOFF()

Description:

Statechart information for Class: controller_density

Description:

Overridden: false

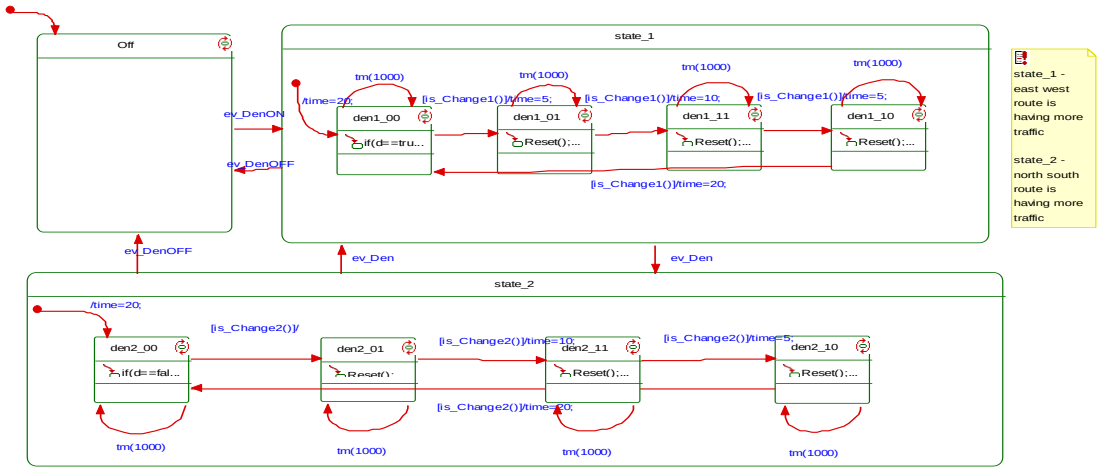


Fig 4.7 controller density

Class name: Density_sensor

Description:

Active: false

Behavior Overridden: false

Composite: false

Reactive: true

Attribute Information for Class: Density_sensor

Attribute Name: NS

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: EW

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Operation information for Class: Density_sensor

Operation name: compare

Initializer:

Const: false

Trigger: false

Body: if(NS>EW)

itsController_density->set_status(true);

else

itsController_density->set_status(false);

Abstract: false

Static: false

Virtual: false

Visibility: public

Signature: compare()

Return Type: void

Description:

Operation name: density

Initializer:

Const: false

Trigger: false

Body: NS=rand()%50;

EW=rand()%50;

Abstract: false

Static: false

Virtual: false

Visibility: public

Signature: density()

Return Type: void

Description:

Relation information for Class: Density_sensor

Name	Inverse	Source	Target
itsController_density		Density_sensor	controller_density

Statechart information for Class: Density_sensor

Description:

Overridden: false

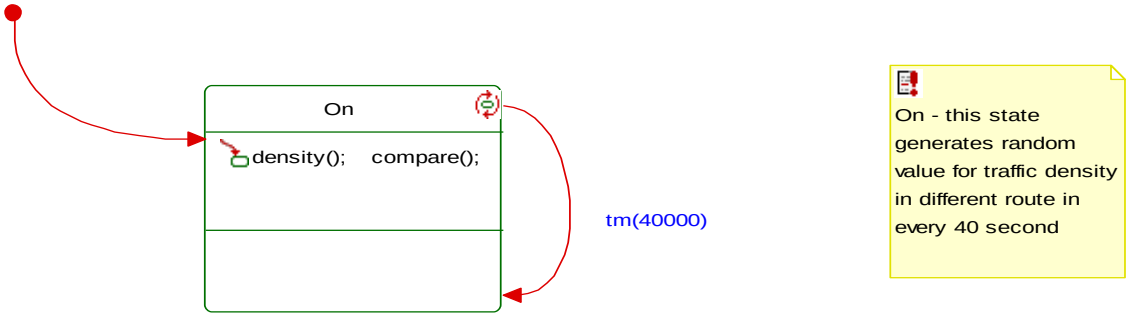


Fig 4.8: density sensor

Class name: Traffic_density

Description:

Active: false

Behavior Overridden: false

Composite: true

Reactive: true

Class name: Normal

Description:

Active: false

Behavior Overridden: false

Composite: false

Reactive: true

Attribute Information for Class: Normal

Attribute Name: time

Default Value:

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: N_red

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: S_red

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: E_red

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: W_red

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: N_orange

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: S_orange

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: E_orange

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: W_orange

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: N_green

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: S_green

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: E_green

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: W_green

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: N_time

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: S_time

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: E_time

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: W_time

Default Value: 0

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Attribute Name: TURN_LEFT

Default Value: 1

Static: false

Visibility: public

Type: int

Stereotype:

Description:

Operation information for Class: Normal

Operation name: Reset

Initializer:

Const: false

Trigger: false

Body: N_red=0;

N_green=0;

N_orange=0;

S_red=0;

S_green=0;

S_orange=0;

E_red=0;

E_green=0;

E_orange=0;

W_red=0;

W_green=0;

W_orange=0;

Abstract: false

Static: false

Virtual: false

Visibility: public

Signature: Reset()

Return Type: void

Description:

Operation name: dec_time

Initializer:

Const: false

Trigger: false

Body: N_time--;

S_time--;

E_time--;

W_time--;

Abstract: false

Static: false

Virtual: false

Visibility: public

Signature: dec_time()

Return Type: bool

Description:

Operation name: is_Norm1

Initializer:

Const: false

Trigger: false

Body: if(N_time==5 && E_time==0 && S_time==10 && W_time==0)

return true;

else

return false;

Abstract: false

Static: false

Virtual: false

Visibility: public

Signature: is_Norm1()

Return Type: bool

Description:

Operation name: is_Norm2

Initializer:

Const: false

Trigger: false

Body: if(N_time==0 && E_time==0 && S_time==5 && W_time==15)

return true;

else

return false;

Abstract: false

Static: false

Virtual: false

Visibility: public

Signature: is_Norm2()

Return Type: bool

Description:

Operation name: is_Norm3

Initializer:

Const: false

Trigger: false

Body: if(N_time==0 && E_time==5 && S_time==0 && W_time==10)

return true;

else

return false;

Abstract: false

Static: false

Virtual: false

Visibility: public

Signature: is_Norm3()

Return Type: bool

Description:

Operation name: is_Norm4

Initializer:

Const: false

Trigger: false

Body: if(N_time==15 && E_time==0 && S_time==0 && W_time==5)

return true;

else

return false;

Abstract: false

Static: false

Virtual: false

Visibility: public

Signature: is_Norm4()

Return Type: bool

Description:

Operation name: is_Norm5

Initializer:

Const: false

Trigger: false

Body: if(N_time==10 && E_time==0 && S_time==5 && W_time==0)

return true;

else

return false;

Abstract: false

Static: false

Virtual: false

Visibility: public

Signature: is_Norm5()

Return Type: bool

Description:

Operation name: is_Norm6

Initializer:

Const: false

Trigger: false

Body: if(N_time==5 && E_time==15 && S_time==0 && W_time==0)

return true;

else

return false;

Abstract: false

Static: false

Virtual: false

Visibility: public

Signature: is_Norm6()

Return Type: bool

Description:

Operation name: is_Norm7

Initializer:

Const: false

Trigger: false

Body: if(N_time==0 && E_time==10 && S_time==0 && W_time==5)

return true;

else

return false;

Abstract: false

Static: false

Virtual: false

Visibility: public

Signature: is_Norm7()

Return Type: bool

Description:

Operation name: is_Norm8

Initializer:

Const: false

Trigger: false

Body: if(N_time==0 && E_time==5 && S_time==15 && W_time==0)

return true;

else

return false;

Abstract: false

Static: false

Virtual: false

Visibility: public

Signature: is_Norm8()

Return Type: bool

Description:

EventReception information for Class: Normal

Event Reception name: ev_NormON

Signature: ev_NormON()

Description:

Event Reception name: ev_NormOFF

Signature: ev_NormOFF()

Description:

Statechart information for Class: Normal

Description:

Overridden: false

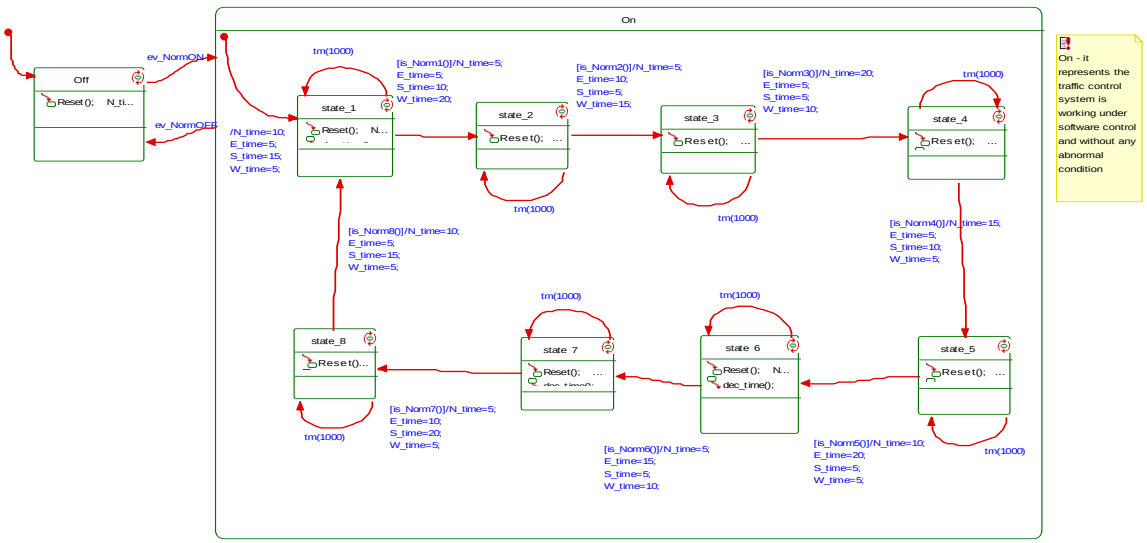


Fig 4.9: Normal

Actor Information for Package: Default

Actor name: TPI

Description:

Relation information for Actor TPI

Name	Inverse	Source	Target
itsPower Switch On	itsTPI	TPI	Power Switch On
itsPower Switch Off	itsTPI	TPI	Power Switch Off
itsPower_On	itsTPI	TPI	Power_On

itsManual	itsTPI	TPI	Manual
itsMode_Selection	itsTPI	TPI	Mode_Selection
itsPower_Off	itsTPI	TPI	Power_Off
itsEmergency_Mode	itsTPI	TPI	Emergency_Mode
itsDensity	itsTPI	TPI	Density

Actor name: Public

Description:

Relation information for Actor Public

Name	Inverse	Source	Target
itsFollowing rules of Traffic	itsPublic	Public	Following rules of Traffic

Use Case Information for Package: Default

Use Case name: Power Switch On

Description:

Extension Points:

Relation information for Use case Power Switch On

Name	Inverse	Source	Target
itsTPI	itsPower Switch On	Power Switch On	TPI

Use Case name: Power Switch Off

Description:

Extension Points:

Relation information for Use case Power Switch Off

Name	Inverse	Source	Target
itsTPI	itsPower Switch Off	Power Switch Off	TPI

Use Case name: Select Mode

Description:

Extension Points:

Generalization information for Use Case Select Mode

Name	Base	Derived
Emergency Open	Emergency Open	Select Mode
Emergency Open EW	Emergency Open EW	Select Mode

Use Case name: Manual

Description:

Extension Points:

Generalization information for Use Case Manual

Name	Base	Derived
Select Mode	Select Mode	Manual
Traffic	Traffic	Manual

Relation information for Use case Manual

Name	Inverse	Source	Target
itsTPI	itsManual	Manual	TPI

Use Case name: Emergency Open

Description:

Extension Points:

Use Case name: Emergency Open EW

Description:

Extension Points:

Use Case name: Power_On

Description:

Extension Points:

Relation information for Use case Power_On

Name	Inverse	Source	Target
itsTPI	itsPower_On	Power_On	TPI

Use Case name: Mode_Selection

Description:

Extension Points:

Relation information for Use case Mode_Selection

Name	Inverse	Source	Target
itsTPI	itsMode_Selection	Mode_Selection	TPI

Use Case name: Emergency_Mode

Description:

Extension Points:

Generalization information for Use Case Emergency_Mode

Name	Base	Derived
Traffic	Traffic	Emergency_Mode

Relation information for Use case Emergency_Mode

Name	Inverse	Source	Target
itsTPI	itsEmergency_Mode	Emergency_Mode	TPI

Use Case name: NS_Emergency

Description:

Extension Points:

Generalization information for Use Case NS_Emergency

Name	Base	Derived
Emergency_Mode	Emergency_Mode	NS_Emergency

Use Case name: EW_Emergency

Description:

Extension Points:

Generalization information for Use Case EW_Emergency

Name	Base	Derived
Emergency_Mode	Emergency_Mode	EW_Emergency

Use Case name: Power_Off

Description:

Extension Points:

Relation information for Use case Power_Off

Name	Inverse	Source	Target
itsTPI	itsPower_Off	Power_Off	TPI

Use Case name: Density

Description:

Extension Points:

Generalization information for Use Case Density

Name	Base	Derived
Mode_Selection	Mode_Selection	Density

Relation information for Use case Density

Name	Inverse	Source	Target
itsTPI	itsDensity	Density	TPI

Use Case name: NS_Density

Description:

Extension Points:

Generalization information for Use Case NS_Density

Name	Base	Derived
Density	Density	NS_Density

Use Case name: EW_Density

Description:

Extension Points:

Generalization information for Use Case EW_Density

Name	Base	Derived
Density	Density	EW_Density

Use Case name: Traffic

Description:

Extension Points:

Generalization information for Use Case Traffic

Name	Base	Derived
Mode_Selection	Mode_Selection	Traffic

Use Case name: Normal_Indian

Description:

Extension Points:

Generalization information for Use Case Normal_Indian

Name	Base	Derived
Mode_Selection	Mode_Selection	Normal_Indian

Use Case name: Following rules of Traffic

Description:

Extension Points:

Relation information for Use case Following rules of Traffic

Name	Inverse	Source	Target
itsPublic	itsFollowing rules of Traffic	Following rules of Traffic	Public

Event information for Package Default

Event name: ev_Start

Signature: ev_Start()

Description:

Event name: ev_Stop

Signature: ev_Stop()

Description:

Event name: ev_On

Signature: ev_On()

Description:

Event name: ev_Off

Signature: ev_Off()

Description:

Event name: ev_Emergency

Signature: ev_Emergency()

Description:

Event name: ev_Hold

Signature: ev_Hold()

Description:

Event name: ev_Manual

Signature: ev_Manual()

Description:

Event name: ev_NSsensorON

Signature: ev_NSsensorON()

Description:

Event name: ev_NSsensorOFF

Signature: ev_NSsensorOFF()

Description:

Event name: ev_EWsensorON

Signature: ev_EWsensorON()

Description:

Event name: ev_EWsensorOFF

Signature: ev_EWsensorOFF()

Description:

Event name: ev_EWopen

Signature: ev_EWopen()

Description:

Event name: ev_NSopen

Signature: ev_NSopen()

Description:

Event name: ev_EmergNS

Signature: ev_EmergNS()

Description:

Event name: ev_EmergEW

Signature: ev_EmergEW()

Description:

Event name: ev_EWsense

Signature: ev_EWsense()

Description:

Event name: ev_NSsense

Signature: ev_NSsense()

Description:

Event name: ev_Den

Signature: ev_Den()

Description:

Event name: ev_DenON

Signature: ev_DenON()

Description:

Event name: ev_NormON

Signature: ev_NormON()

Description:

Event name: ev_NormOFF

Signature: ev_NormOFF()

Description:

Event name: ev_DenOFF

Signature: ev_DenOFF()

Description:

Use Case Diagram Information

Use Case Diagram name: Traffic_usecasediagram

Description:

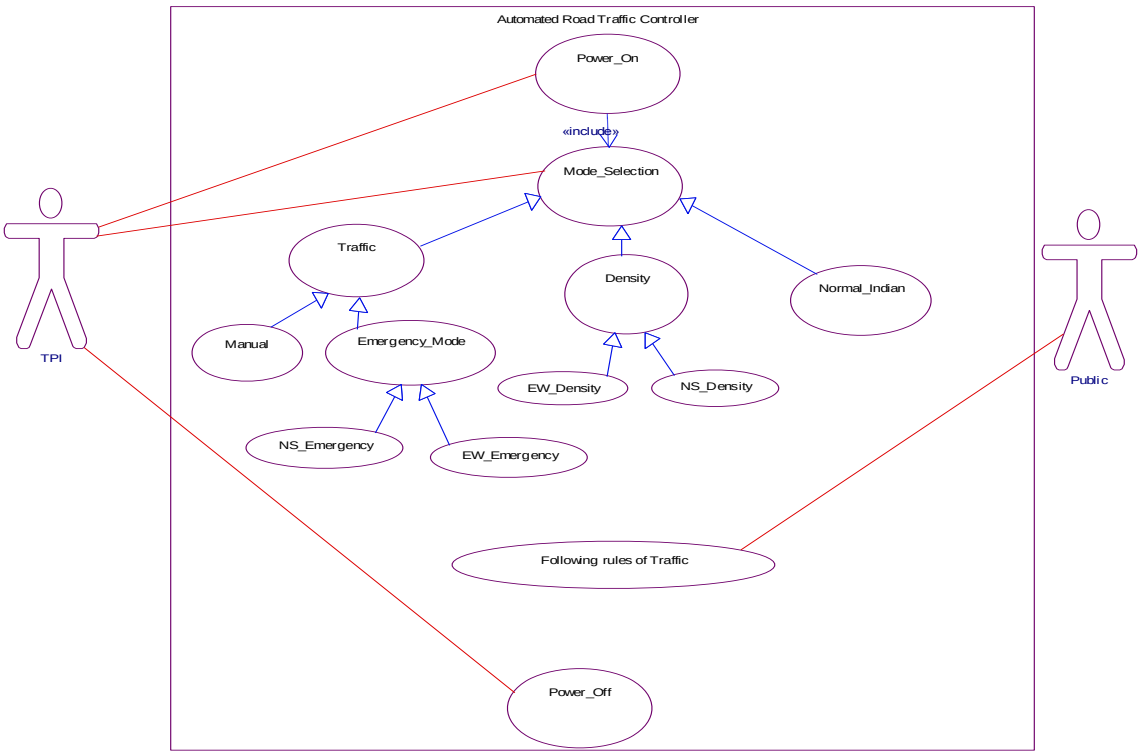


Fig 4.10: Traffic usecasediagram

Package: PredefinedTypes

Description:

Type information for Package PredefinedTypes

Type name: RhpInteger

Description: Predefined RhpInteger

Kind: Language

Type name: RhpCharacter

Description: Predefined RhpCharacter

Kind: Language

Type name: RhpString

Description: Predefined RhpString

Kind: Language

Type name: RhpReal

Description: Predefined RhpReal

Kind: Language

Sequence Diagram Information

Sequence Diagram name: Traffic_sequencediagram

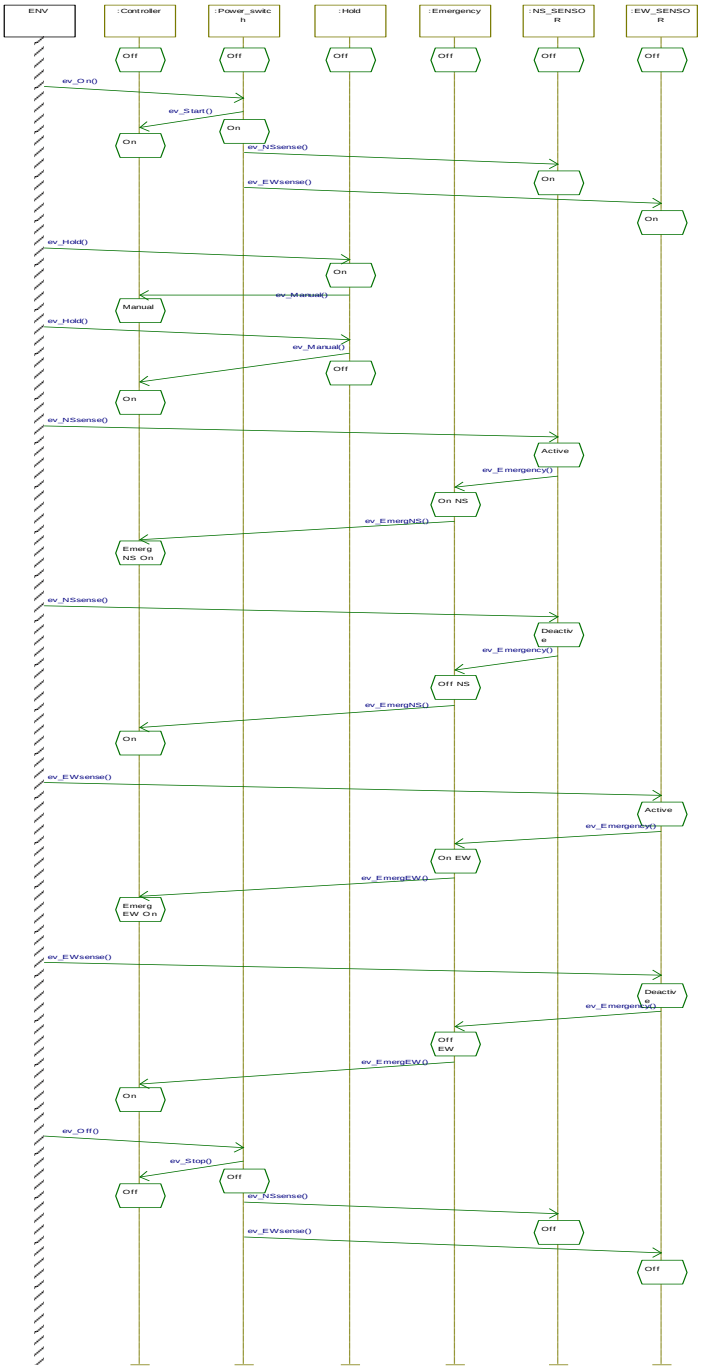


Fig 3.17 traffic seqance diagram

Sequence Diagram name: Density_sequencediagram

Description:

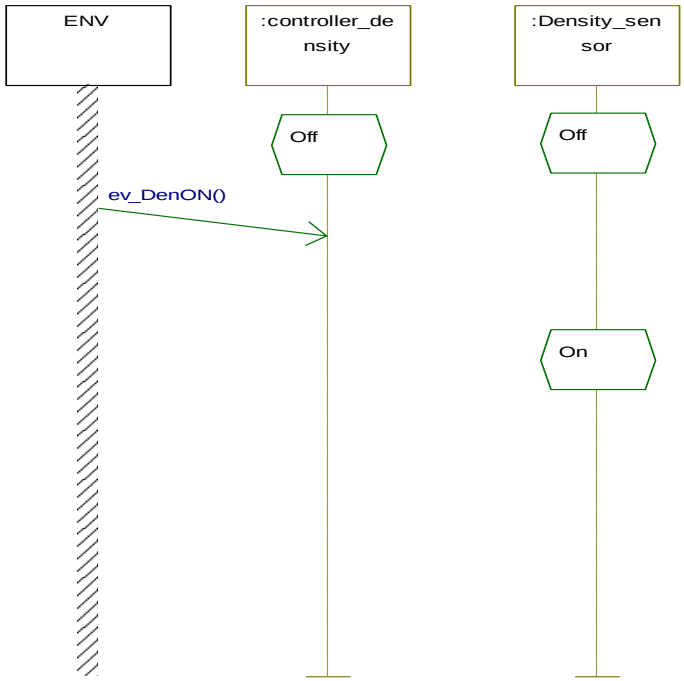


Fig 3.18 Density_sequencediagram

Sequence Diagram name: Normal_sequencediagram

Description:

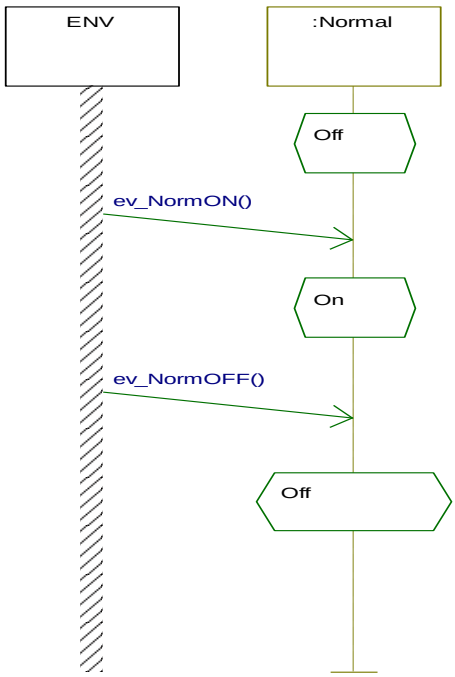


Fig 3.19 Normal_sequencediagram

Structure Diagram Information

Structure Diagram name: Traffic_structurediagram

Description:

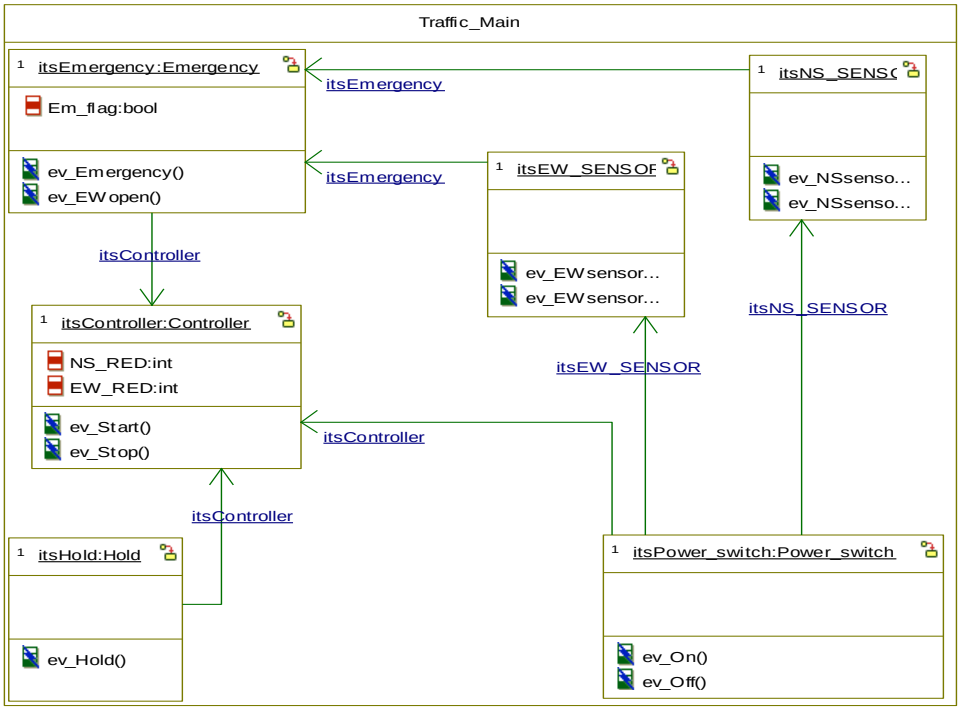


Fig 3.20 Traffic_structurediagram

Structure Diagram name: Trafficdensitystructurediagram

Description:

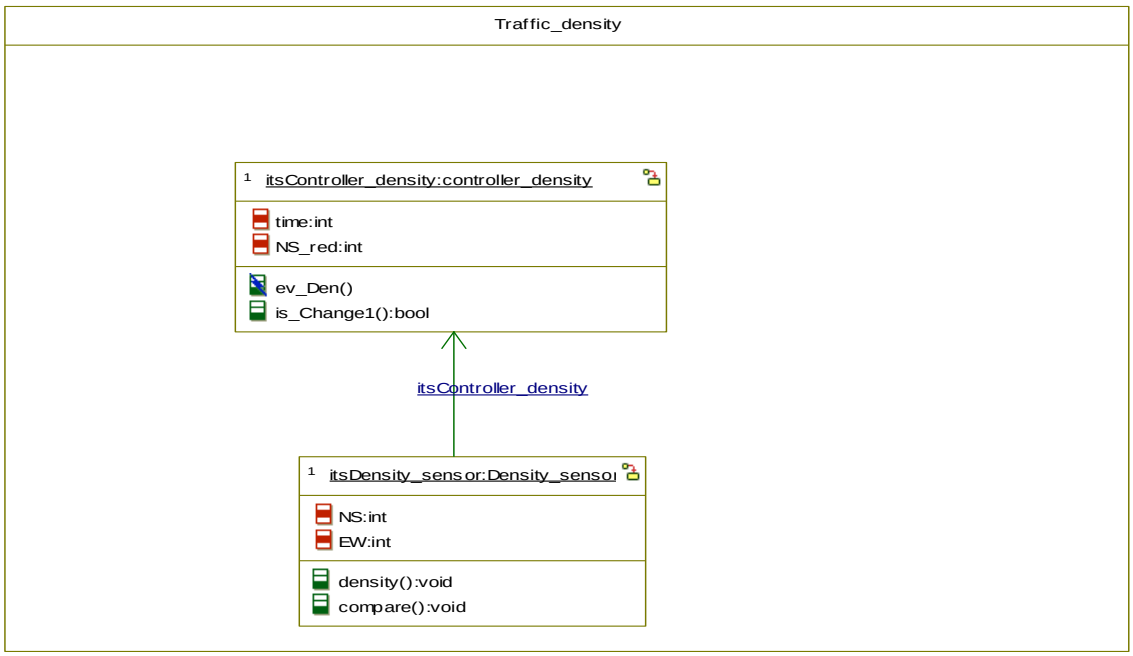


Fig 3.21 Trafficdensitystructurediagram

Type name: RhpVoid

Description: Predefined RhpVoid

Kind: Language

Type name: RhpPositive

Description: Predefined RhpPositive

Kind: Language

Type name: RhpAddress

Description: Predefined RhpAddress

Kind: Language

Type name: RhpBoolean

Description: Predefined RhpBoolean

Kind: Language

Type name: RhpUnlimitedNatural

Description: Predefined RhpUnlimitedNatural

Kind: Language

Stereotype information for Package: PredefinedTypes

Stereotype name: AppliedProfile

Description:

OfMetaClass: Dependency

Stereotype name: ImportedProfile

Description:

OfMetaClass: Dependency

Stereotype name: Settings

Description:

OfMetaClass: Profile

Stereotype name: Merge

Description:

OfMetaClass: Dependency

Stereotype name: Redefines

Description:

OfMetaClass: Dependency

Stereotype name: EclipseConfiguration

Description:

OfMetaClass: Configuration

Stereotype name: Interface

Description:

OfMetaClass: Class

Stereotype name: Usage

Description:

OfMetaClass: Dependency

Stereotype name: Framework

Description:

OfMetaClass: Package

Stereotype name: Metaclass

Description:

OfMetaClass: Class

Stereotype name: Specification

Description:

OfMetaClass:

Actor,Class,AssociationClass,Statechart,FlowItem,Event,Node,Type,UseCase

Stereotype name: Realization

Description:

OfMetaClass:

Actor,Class,AssociationClass,Statechart,FlowItem,Event,Node,Type,UseCase

Stereotype name: Send

Description:

OfMetaClass: Dependency

Stereotype name: Resource

Description:

OfMetaClass: Class

Stereotype name: Singleton

Description:

OfMetaClass: Object_type

Stereotype name: MessageQueue

Description:

OfMetaClass: Object_type

Stereotype name: Timer

Description:

OfMetaClass: Object_type

Stereotype name: Semaphore

Description:

OfMetaClass: Object_type

Stereotype name: Mutex

Description:

OfMetaClass: Object_type

Stereotype name: EventFlag

Description:

OfMetaClass: Object_type

Stereotype name: Task

Description:

OfMetaClass: Object_type

Stereotype name: Executable

Description:

OfMetaClass: Component

Stereotype name: Library

Description:

OfMetaClass: Component

Stereotype name: Table

Description:

OfMetaClass: Component

Stereotype name: Document

Description:

OfMetaClass: Component

Stereotype name: Realization

Description:

OfMetaClass: Generalization

Stereotype name: A2D

Description:

OfMetaClass: Node

Stereotype name: Board

Description:

OfMetaClass: Node

Stereotype name: Bus

Description:

OfMetaClass: Node

Stereotype name: Button

Description:

OfMetaClass: Node

Stereotype name: D2A

Description:

OfMetaClass: Node

Stereotype name: DigitalIO

Description:

OfMetaClass: Node

Stereotype name: Disk

Description:

OfMetaClass: Node

Stereotype name: Display

Description:

OfMetaClass: Node

Stereotype name: Keyboard

Description:

OfMetaClass: Node

Stereotype name: Motor

Description:

OfMetaClass: Node

Stereotype name: Mouse

Description:

OfMetaClass: Node

Stereotype name: Panel

Description:

OfMetaClass: Node

Stereotype name: Printer

Description:

OfMetaClass: Node

Stereotype name: Processor

Description:

OfMetaClass: Node

Stereotype name: Sensor

Description:

OfMetaClass: Node

Stereotype name: include

Description:

OfMetaClass: Dependency

Stereotype name: extend

Description:

OfMetaClass: Dependency

Stereotype name: trace

Description:

Specifies a trace relationship between model elements or sets of model elements that represent the same concept in different models. Traces are mainly used for tracking requirements and changes across models. Since model changes can occur in both directions, the directionality of the dependency can often be ignored. The mapping specifies the relationship between the two, but it is rarely computable and is usually informal.

OfMetaClass: Dependency

Stereotype name: refine

Description:

Specifies a refinement relationship between model elements at different semantic levels, such as analysis and design. The mapping specifies the relationship between the two elements or sets of elements. The mapping may or may not be computable, and it may be unidirectional or bidirectional. Refinement can be used to model transformations from analysis to design and other such changes.

OfMetaClass: Dependency

S**tereotype name: derive**

Description:

Specifies a derivation relationship among model elements that are usually, but not necessarily, of the same type. A derived dependency specifies that the client may be computed from the supplier. The mapping specifies the computation. The client may be implemented for design reasons, such as efficiency, even though it is logically redundant.

OfMetaClass: Dependency

Stereotype name: flowPort

Description:

OfMetaClass: SysMLPort

Stereotype name: FlowChart

Description:

OfMetaClass: ActivityDiagram

Stereotype name: CallBehavior

Description:

OfMetaClass: ReferenceActivity

Stereotype name: ModelLibrary

Description:

OfMetaClass: Package

Stereotype name: VariationPoint

Description:

OfMetaClass: Class

Stereotype name: Varies

Description:

OfMetaClass: Dependency

Stereotype name: Variant

Description:

OfMetaClass: Class

Stereotype name: Static

Description:

OfMetaClass: Generalization

Stereotype name: VisualStudioConfiguration

Description:

OfMetaClass: Configuration

Stereotype name: ActivityFinal

Description:

OfMetaClass: State

Stereotype name: DecisionNode

Description:

OfMetaClass: Connector

Stereotype name: MergeNode

Description:

OfMetaClass: Connector

Stereotype name: ControlFlow

Description: Control Flow edges model the movement of control from one node to another.

OfMetaClass: Transition

Stereotype name: ObjectFlow

Description: Object flow edges model the flow of objects or data from one node to another.

OfMetaClass: Transition

Stereotype name: interruptibleRegion

Description:

OfMetaClass: State

Stereotype name: Class Diagram

Description:

OfMetaClass: ObjectModelDiagram

Package: PredefinedTypesCpp

Description:

Type information for Package PredefinedTypesCpp

Type name: int

Description: Predefined int

Kind: Language

Type name: char

Description: Predefined char

Kind: Language

Type name: char*

Description: Predefined char*

Kind: Language

Type name: double

Description: Predefined double

Kind: Language

Type name: void

Description: Predefined void

Kind: Language

Type name: long

Description: Predefined long

Kind: Language

Type name: void *

Description: Predefined void *

Kind: Language

Type name: OMBoolean

Description: Predefined boolean

Kind: Language

Type name: OMString

Description: Predefined String

Kind: Language

Type name: short

Description: Predefined short

Kind: Language

Type name: unsigned int

Description: Predefined unsigned int

Kind: Language

Type name: unsigned short

Description: Predefined unsigned short

Kind: Language

Type name: unsigned char

Description: Predefined unsigned char

Kind: Language

Type name: unsigned long

Description: Predefined unsigned long

Kind: Language

Type name: long double

Description: Predefined long double

Kind: Language

Type name: bool

Description:

Kind: Language

Type name: float

Description: Predefined float

Kind: Language

Stereotype information for Package: PredefinedTypesCpp

Stereotype name: Friend

Description:

OfMetaClass: Dependency

Stereotype name: CORBAInterface

Description:

OfMetaClass: Class

Stereotype name: CORBAException

Description:

OfMetaClass: Class

Stereotype name: CORBAModule

Description:

OfMetaClass: Package

Stereotype name: Subsystem

Description:

OfMetaClass: Class

Stereotype name: ConnectionPoint

Description:

OfMetaClass: Dependency

Stereotype name: Web Managed

Description:

OfMetaClass: Class,Attribute,PrimitiveOperation,TriggeredOperation,Event

Stereotype name: Reactive Interface

Description:

OfMetaClass: Class

.ANIMATED SEQUENCE DIAGRAM FOR TRAFFIC_MAIN

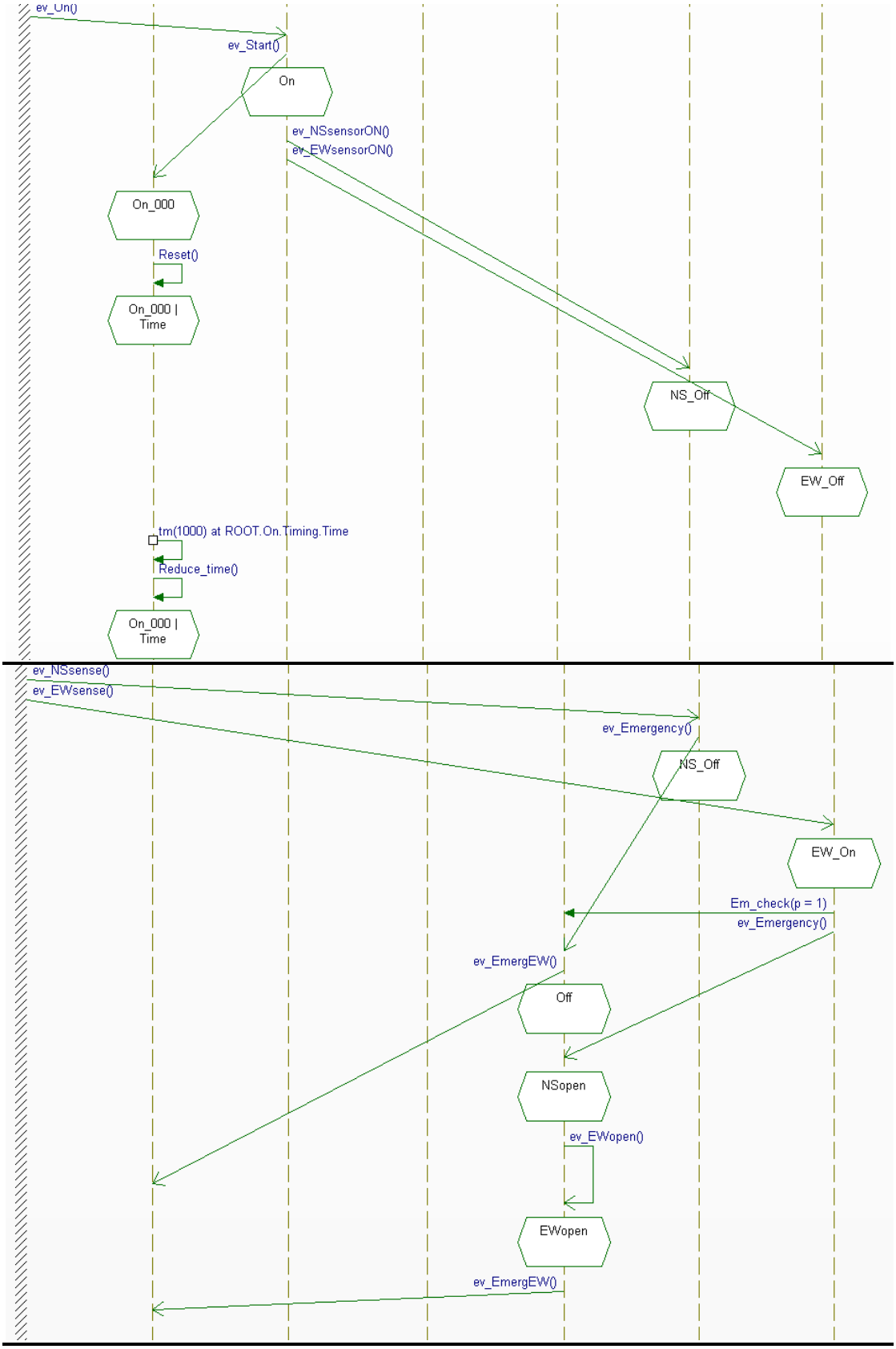


fig 3.22 animated sequence diagram for traffic main

ANIMATED SEQUENCE DIAGRAM FOR TRAFFIC_NORMAL_INDIAN

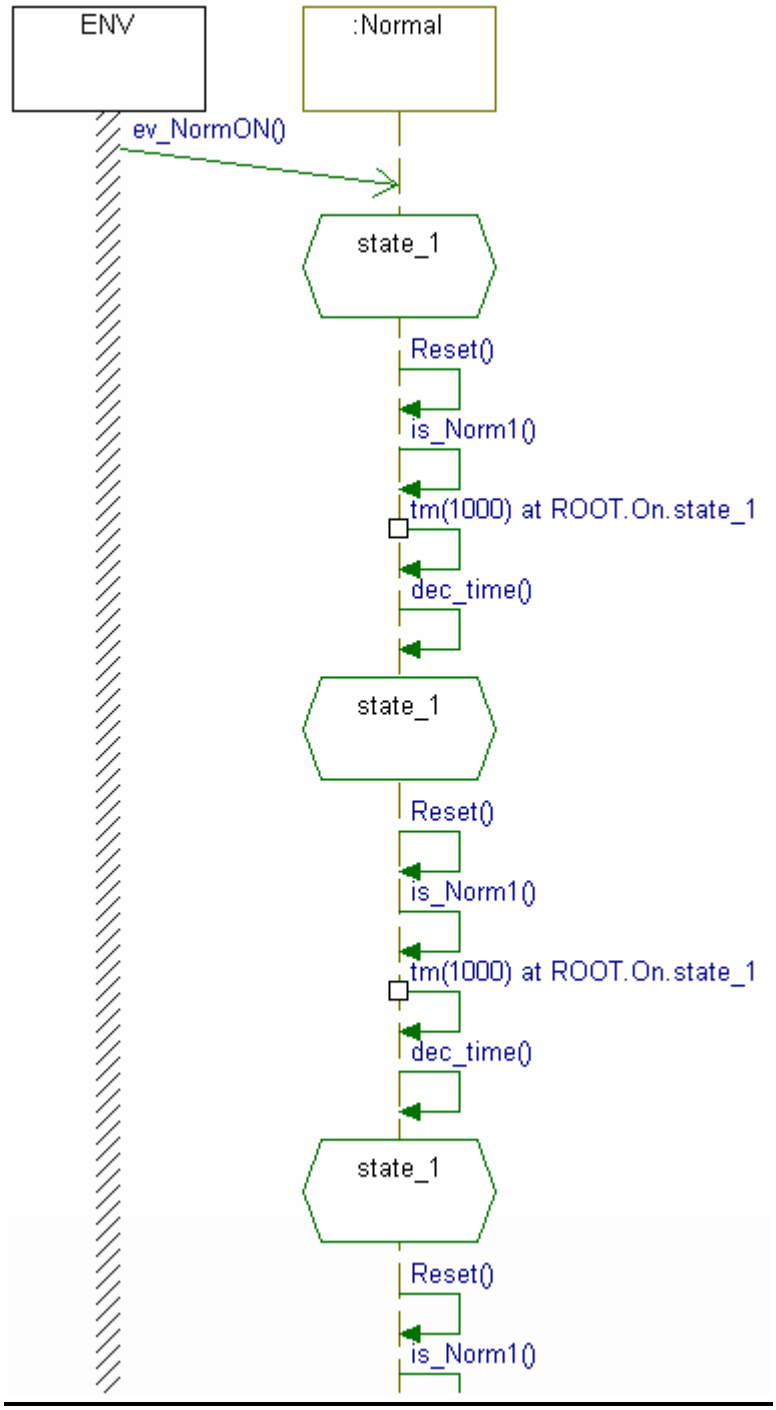
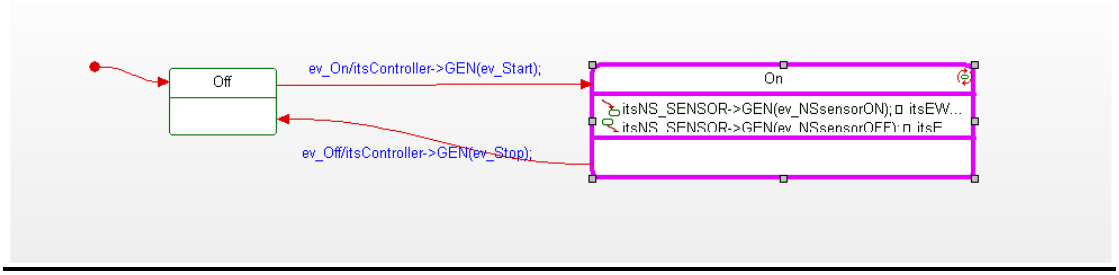


fig 3.23 animated sequence diagram for traffic normal indian

ANIMATED STATECHART FOR POWER SWITCH



Power switch is used to on the controller and off it as and when required.

ANIMATED STATECHART FOR CONTROLLER

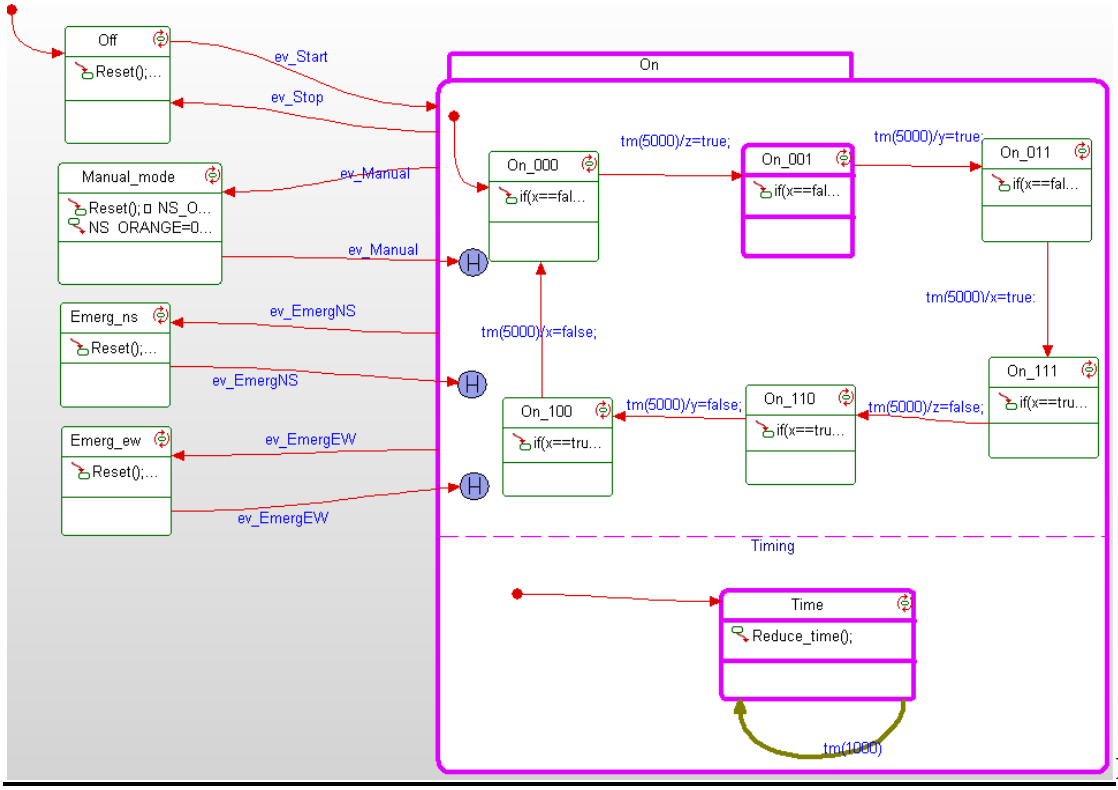
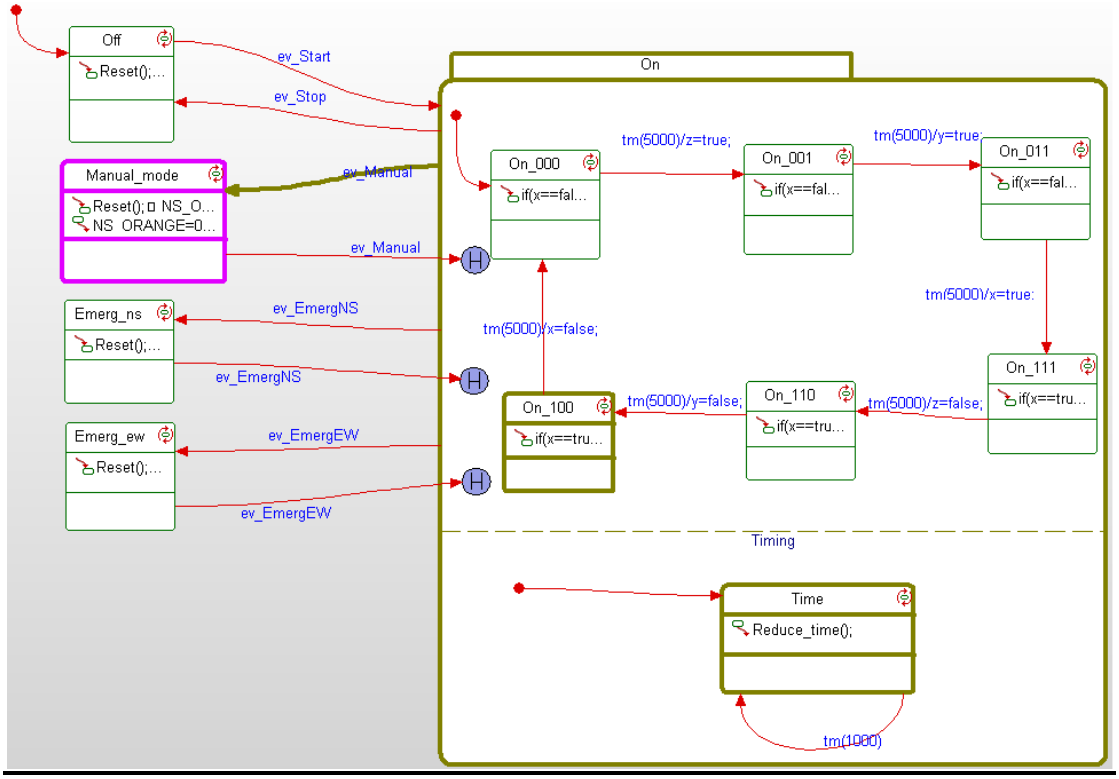
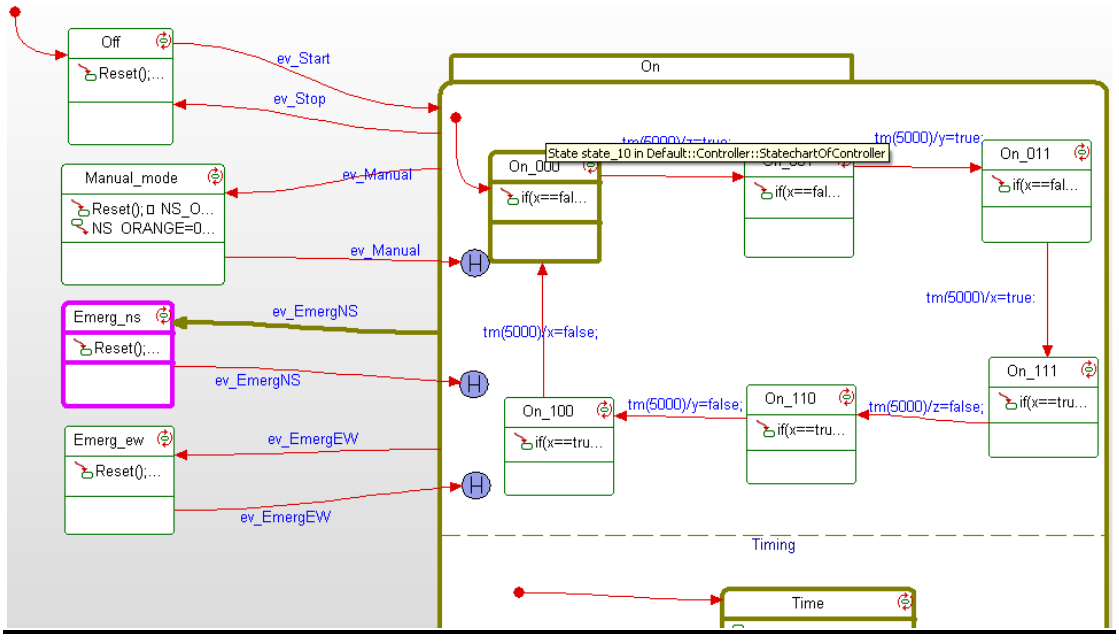
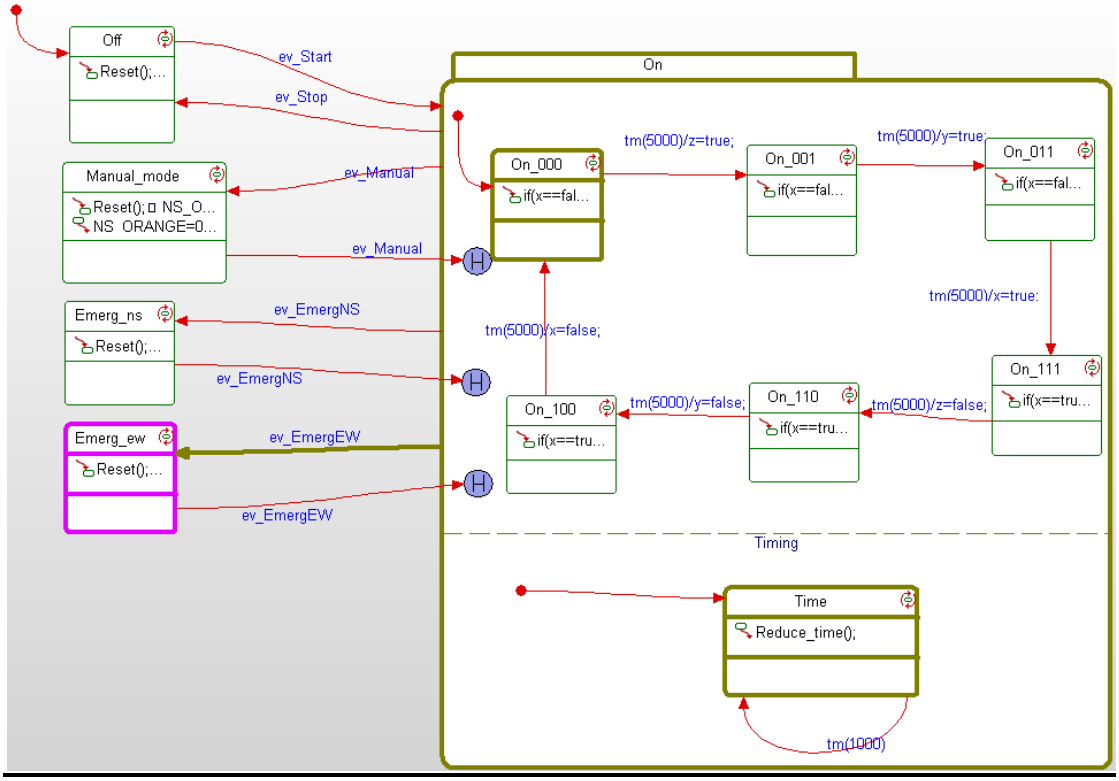


Fig 3.24 animated state chart for controller


Off
Reset();...
ev_Start
ev_Stop
On
Manual_mode
Reset(); NS_O...
NS_ORANGE=0...
ev_Manual
ev_Manual
On_000
if(x==fal...
tm(5000)/z=true;
On_001
if(x==fal...
tm(5000)/y=true;
On_011
if(x==fal...
tm(5000)/x=true;
Emerg_ns
Reset();...
ev_EmergNS
ev_EmergNS
tm(5000)/x=false;
On_100
if(x==tru...
tm(5000)/y=false;
On_110
if(x==tru...
tm(5000)/z=false;
On_111
if(x==tru...
Emerg_ew
Reset();...
ev_EmergEW
ev_EmergEW
Timing
Time
Reduce_time();
tm(1000)


Off
Reset();...
ev_Start
ev_Stop
On
Manual_mode
Reset(); NS_O...
NS_ORANGE=0...
ev_Manual
ev_Manual
State state_10 in Default::Controller::StatechartOfController
On_000
if(x==fal...
tm(5000)/y=true;
On_011
if(x==fal...
tm(5000)/x=true;
Emerg_ns
Reset();...
ev_EmergNS
ev_EmergNS
tm(5000)/x=false;
On_100
if(x==tru...
tm(5000)/y=false;
On_110
if(x==tru...
tm(5000)/z=false;
On_111
if(x==tru...
Emerg_ew
Reset();...
ev_EmergEW
ev_EmergEW
Timing
Time

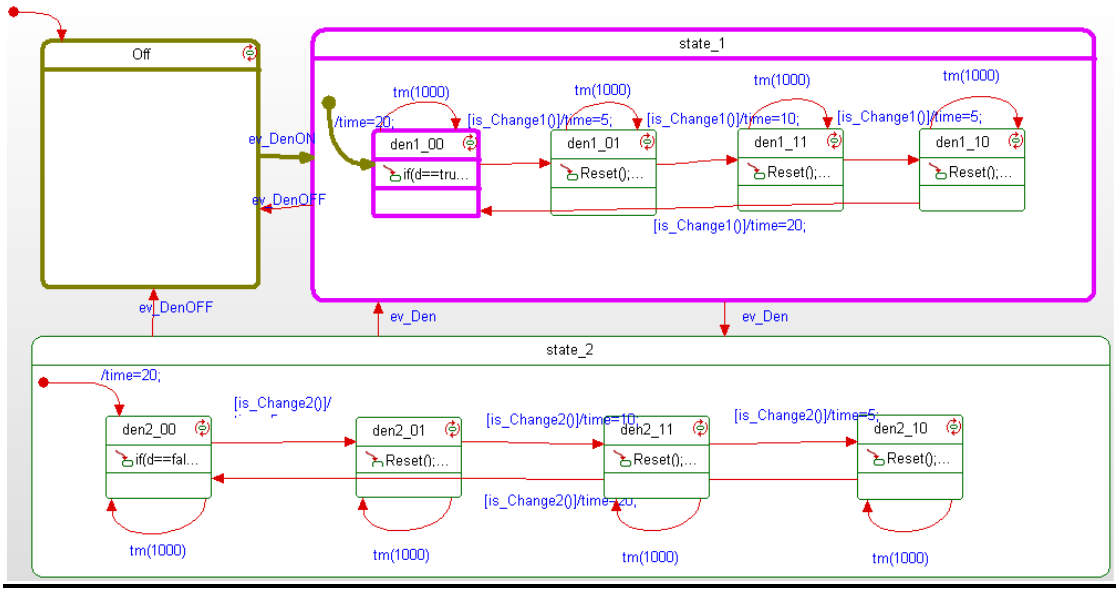


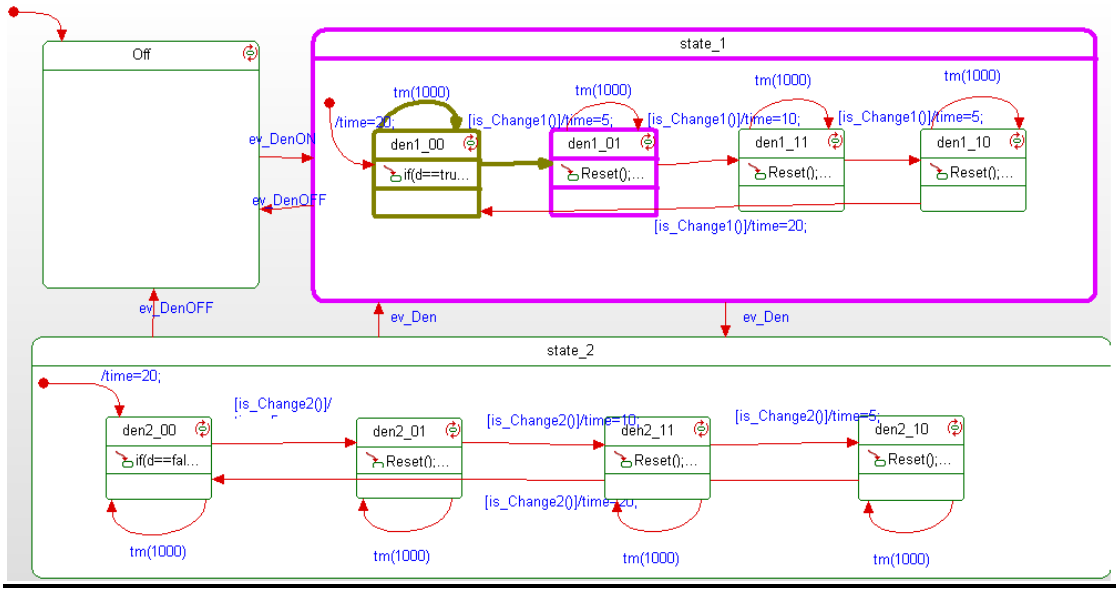
According to the development of our project it is a step by step analysis of traffic system. At first we consider two routes (i.e. East-West and North-South) have equal volumes of traffic at all time. So both the routes are assigned equal clearance time.

But for some special cases we require human interruption for clearing traffic at some time. Hence a manual mode is essential that is built into the system and can be switched by the traffic police as and when desired.

Emergency sensor is built upon, to continuously check the emergency situations like ambulance, police, and fire-brigade etc providing immediate clearance to it irrespective of present state. After clearance the state is resumed.

DENSITY STATE CHART





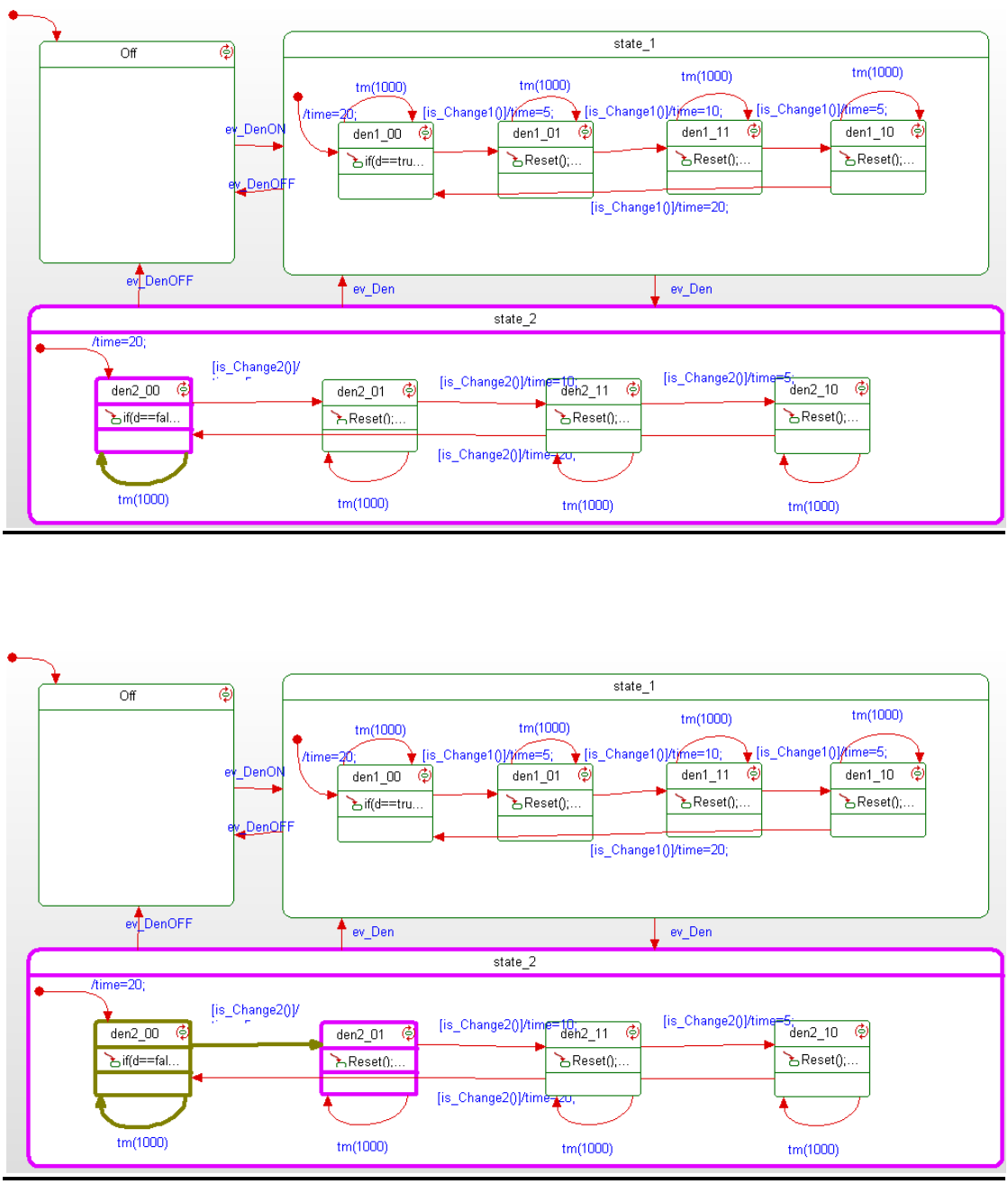
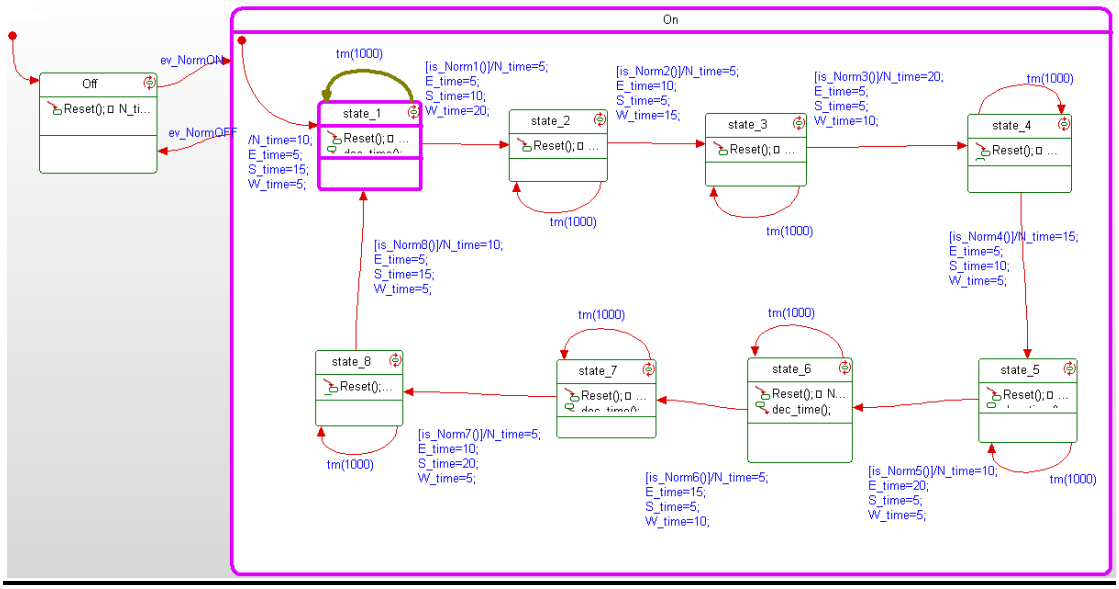


Fig 3.25 density state chart

Providing same clearance time for all routes may waste time. Considering this a density sensor is attached which senses traffic density continuously. This enables intelligent switching time allotment for both routes. The route having higher traffic density is provided with higher clearance time.

ANIMATED STATECHART FOR NORMAL INDIAN



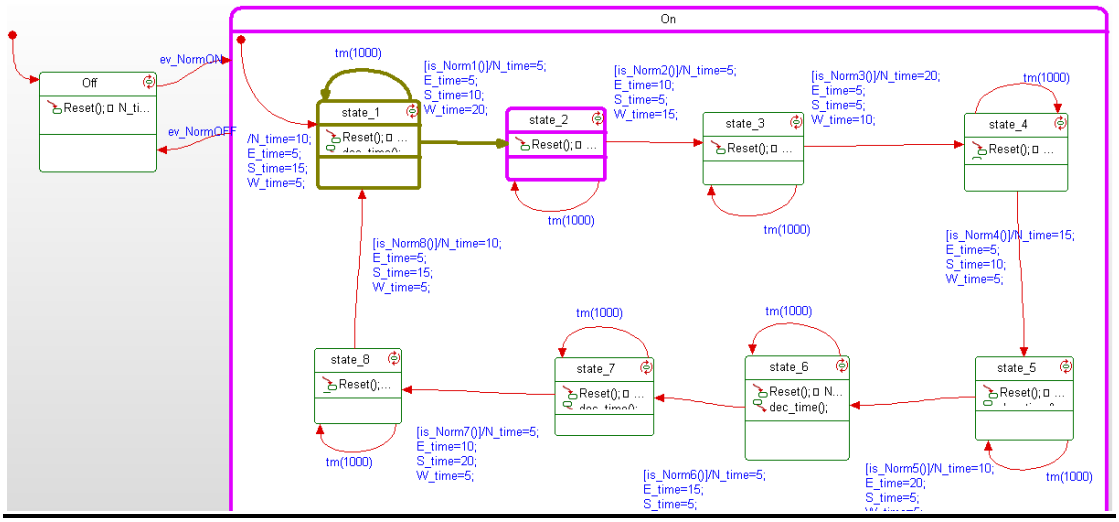


fig 3.26 animated state chart for normal Indian

We then deal with all four routes (i.e. East, West, North and South) separately giving same clearance time like the conventional Indian method.

CHAPTER 5
LPC 2138 MICROCONTROLLER

5.1 Overview On ARM Architecture:

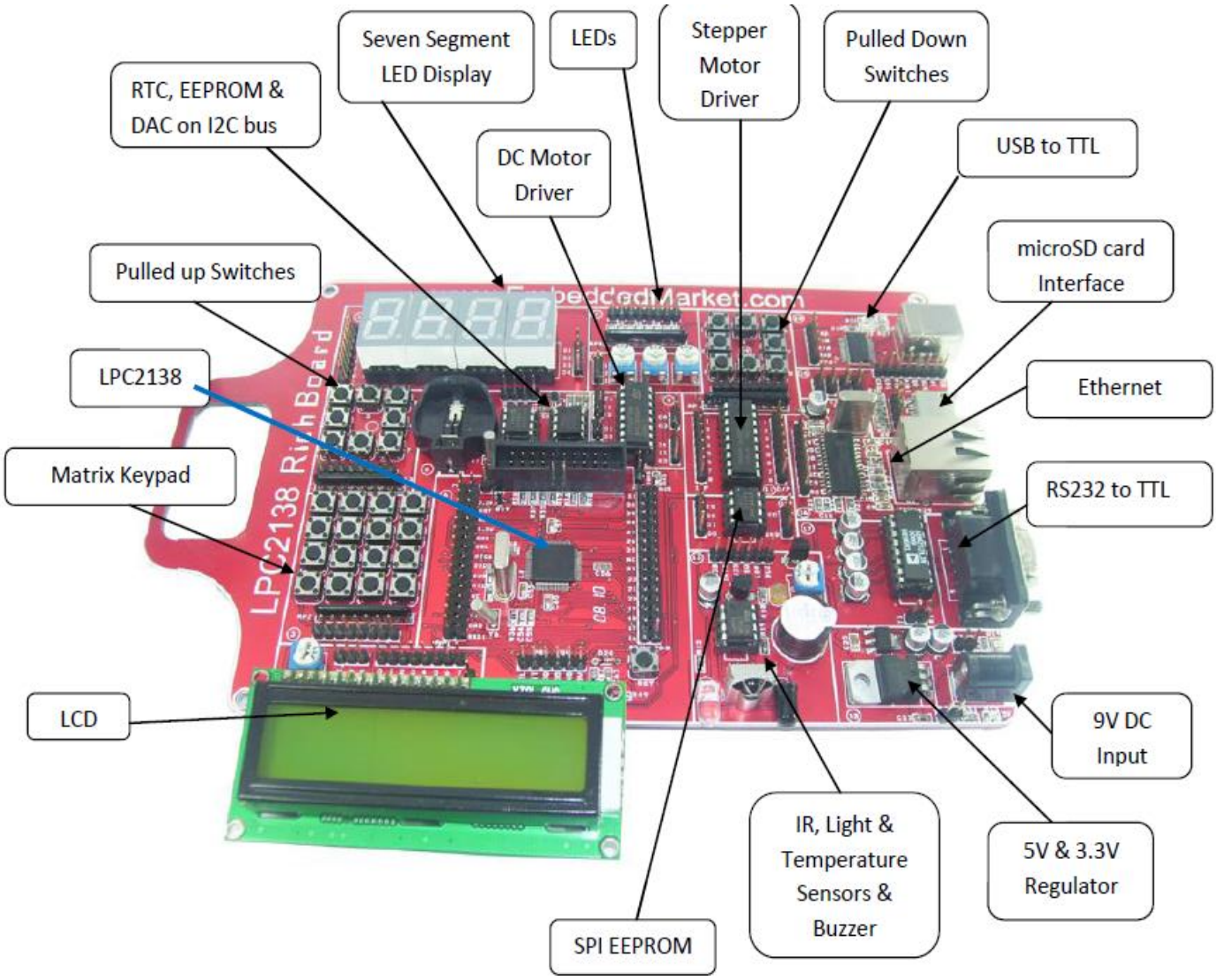
The ARM architecture describes a family of RISC- based computer processors designed and licensed by British company ARM Holdings. It was first developed in the 1980s. ARM Holdings itself does not manufacture its own electronic chips, but licenses its designs to other semiconductor manufacturers. Using a RISC based approach to computer design, ARM processors require significantly fewer transistors than processors that would typically be found in a traditional computer.

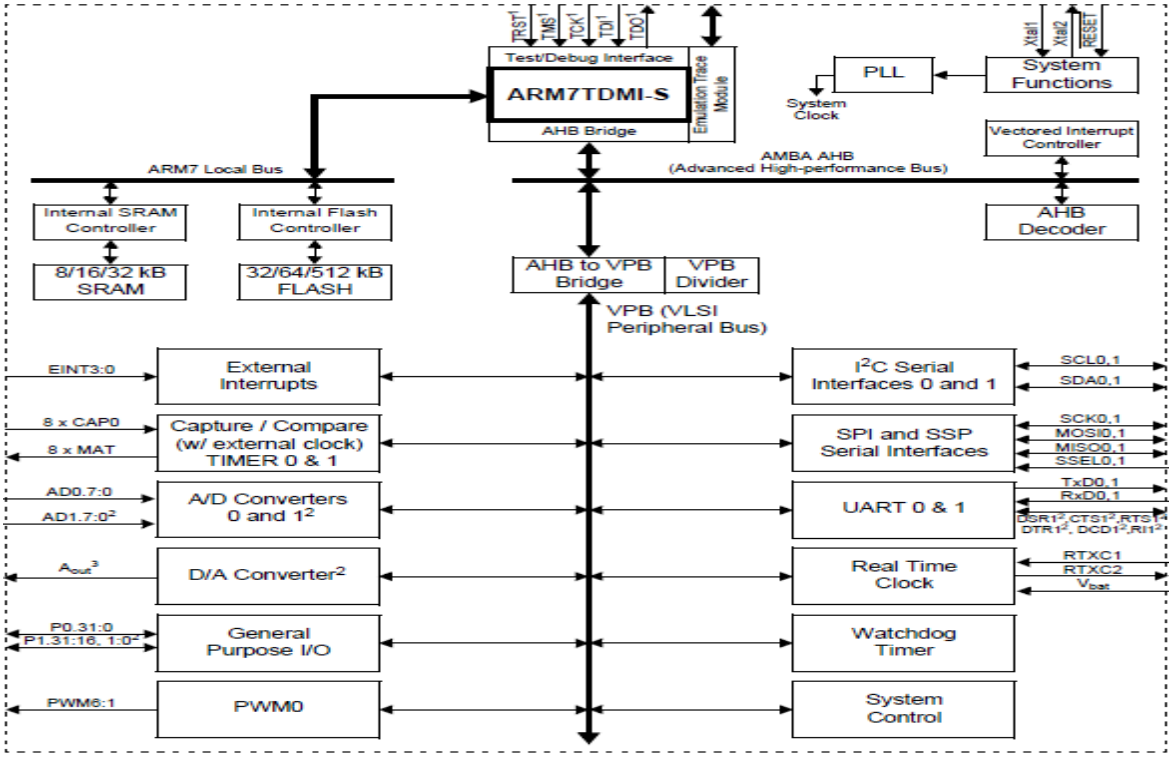
The benefits of this approach are lower costs, less heat, and less power usage, traits that are desirable for use in light, portable, battery-powered devices such as smart phones and tablet computers. The reduced complexity and simpler design allows companies to build a low-energy system on a chip for an embedded system incorporating memory, interfaces, radios, etc.

5.2 FEATURES LPC 2138 MICROCONTROLLER

- 32-bit ARM7TDMI-S microcontroller.
- 32 KB of on-chip static RAM and 512 KB of on-chip Flash program memory. 128 bit wide interface/accelerator enables high speed 60 MHz operation.
- Single Flash sector or full chip erase in 400ms and programming of 256 bytes in 1 ms.
- Two (LPC2138) 8 channel 10-bit A/D converters provide a total of up to 16 analog inputs, with conversion times as low as 2.44 s per channel.

- Single 10-bit D/A converter provides variable analog output. Two 32-bit timers/counters (with four capture and four compare channels each), PWM unit (six outputs) and watchdog.
- Real-time clock equipped with independent power and clock supply permitting extremely low power consumption in power save mode Multiple serial interfaces including two UARTs (16C550).
- Vectored interrupt controller with configurable priorities and vector addresses. Up to 47 of 5 V tolerant general purpose I/O pins. Up to nine edge or level sensitive external interrupt pins available.
- 60 MHz maximum CPU clock available from programmable on-chip Phase-Locked Loop (PLL) with settling time of 100microseconds. On-chip crystal oscillator with an operating range of 1 MHz to 30 MHz's Power saving modes includes idle and Power-down. Individual enable/disable of peripheral functions as well as peripheral clock scaling down for additional power optimization. Processor wake-up from Power-down mode via external interrupt. Single power supply chip with Power-On Reset (POR) and Brown-Out Detection (BOD) circuits:- CPU operating voltage range of 3.0 V to 3.6 V (3.3 V 10 %) with 5 V tolerant I/O pads.


Seven Segment LED Display
LEDs
Stepper Motor Driver
Pulled Down Switches
RTC, EEPROM & DAC on I2C bus
DC Motor Driver
USB to TTL
Pulled up Switches
microSD card Interface
LPC2138
Ethernet
Matrix Keypad
RS232 to TTL
LCD
9V DC Input
IR, Light & Temperature Sensors & Buzzer
5V & 3.3V Regulator
SPI EEPROM


TRST
TMS
TCK
TDI
TDO
Test/Debug Interface
ARM7TDMI-S
AHB Bridge
Emulation Trace Module
Xtal1
Xtal2
RESET
System Functions
PLL
System Clock
Vectored Interrupt Controller
ARM7 Local Bus
AMBA AHB (Advanced High-performance Bus)
Internal SRAM Controller
Internal Flash Controller
AHB Decoder
8/16/32 kB SRAM
32/64/512 kB FLASH
AHB to VPB Bridge
VPB Divider
VPB (VLSI Peripheral Bus)
EINT3:0
External Interrupts
I2C Serial Interfaces 0 and 1
SCL0,1
SDA0,1
8 x CAP0
8 x MAT
Capture / Compare (w/ external clock) TIMER 0 & 1
SPI and SSP Serial Interfaces
SCK0,1
MOSI0,1
MISO0,1
SSEL0,1
AD0.7:0
AD1.7:02
A/D Converters 0 and 12
UART 0 & 1
TxD0,1
RxD0,1
Aout3
D/A Converter2
Real Time Clock
RTXC1
RTXC2
Vbat
P0.31:0
P1.31:16, 1:0
General Purpose I/O
Watchdog Timer
PWM6:1
PWM0
System Control

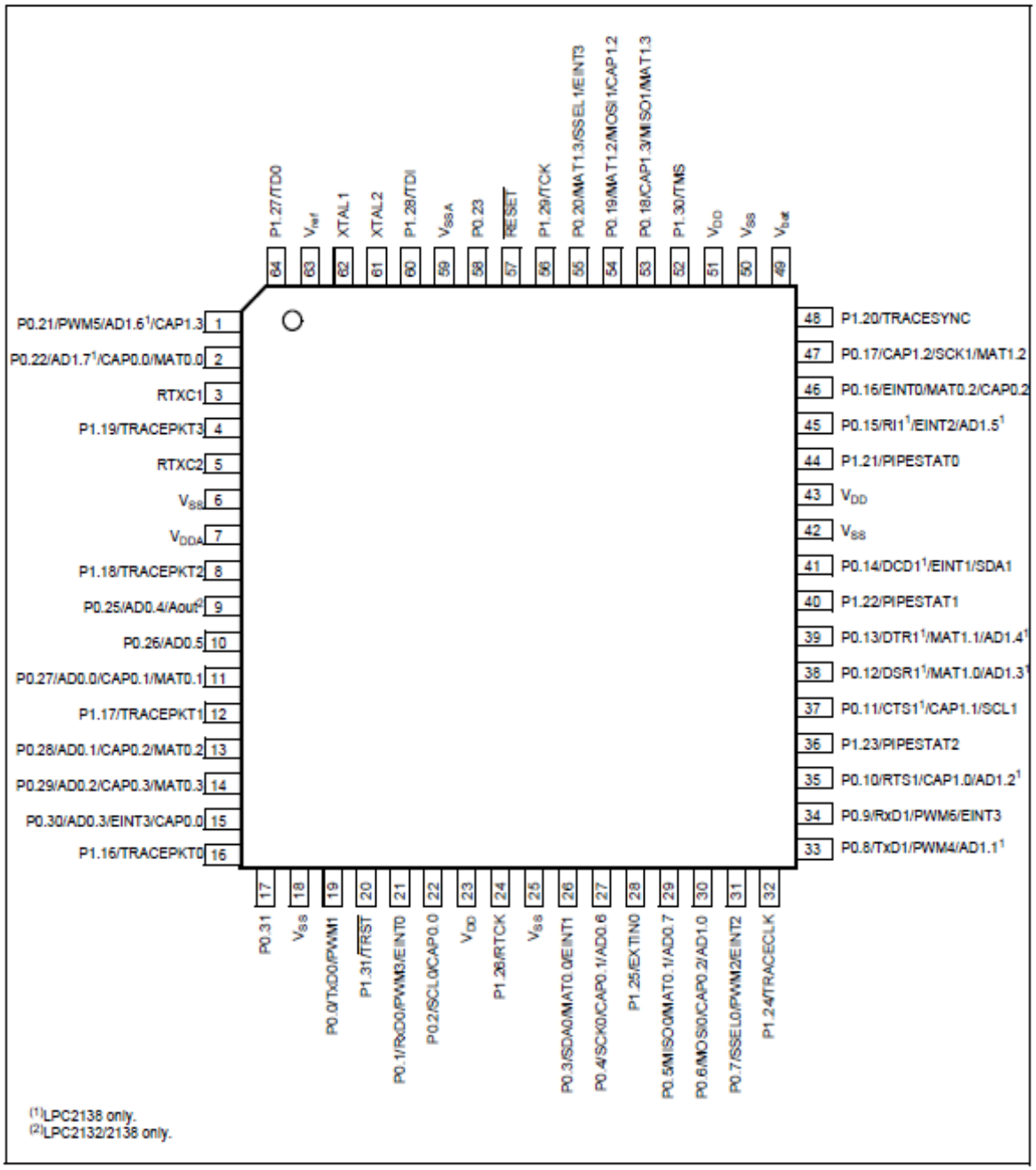


Figure 15: LPC2131/2132/2138 64-pin package

Fig 5.2: lpc2138 pin package

5.3 PIN CONNECTIONS

Port 1

p1->pin 16-->7-segment LED's a

p1->pin 17-->7-segment LED's b

p1->pin 18-->7-segment LED's c

p1->pin 19-->7-segment LED's d

p1->pin 20-->7-segment LED's e

p1->pin 21-->7-segment LED's f

p1->pin 22-->7-segment LED's g

p1->pin 23-->7-segment LED's .

p1->pin 28-->key 4

p1->pin 29-->key 3

p1->pin 30-->key 2

p1->pin 31-->key 1

Port 0

p0->pin 16-->7-segment LED's base->D1 // To make 7-segment LED On/Off

p0->pin 17-->7-segment LED's base->D2

p0->pin 18-->7-segment LED's base->D3

p0->pin 19-->7-segment LED's base->D4

p0->pin 28-->Potentiometer-3 // To get the analog input

p0->pin 29-->Potentiometer-2 // To get the analog input

p0->pin 30-->Potentiometer-1 //To get the analog input

CHAPTER 6

KEIL SOFTWARE

6.1 Introduction

The Keil MDK-ARM Microcontroller Development Kit, (Keil MDK-ARM), tools generate embedded applications for many popular ARM-powered devices. The tools allow engineers to write software programs in assembly language or in a high-level language (like C or C++), and do create a complete application that can be programmed into a microcontroller or other memory device. The Keil MDK-ARM development tools are designed for the professional software developer, but any level of programmer can use them to get the most out of the ARM7/9 and Cortex-M microcontroller.

➢ Features

1. Complete support for Cortex-M, Cortex-R4, ARM7, and ARM9 devices
2. Industry-leading ARM C/C++ Compilation Toolchain
3. µVision4 IDE, debugger, and simulation environment
4. Keil RTX deterministic, small footprint real-time operating system (with source code)
5. TCP/IP Networking Suite offers multiple protocols and various applications
6. USB Device and USB Host stacks are provided with standard driver classes
7. Complete GUI Library for embedded systems with graphical user interfaces
8. ULINK*pro* enables on-the-fly analysis of running applications and records every executed Cortex-M instruction
9. Complete Code Coverage information about your program's execution
10. Execution Profiler and Performance Analyzer enable program optimization
11. Numerous example projects help you quickly become familiar with MDK-ARM's powerful, built-in features
12. CMSIS Cortex Microcontoller Software Interface Standard compliant.
13. MDK MDK-ARM is available in four editions: MDK-Lite, MDK-Basic, MDK-Standard, and MDK-Professional. All editions provide a complete C/C++

development environment and MDK-Professional includes extensive middleware libraries.

6.2 Software Development Cycle

When you use the the µVision4, the development cycle is roughly the same as it is for any other software development project:

1. Create a new project, select the target chip from the Device Database, and configure the tool settings.
2. Create source files in C, C++, or Assembly.
3. Build your application with the project manager.
4. Correct errors in source files.
5. Test the linked application.
6. The following block diagram illustrates the build process and the involved components, which are described below.

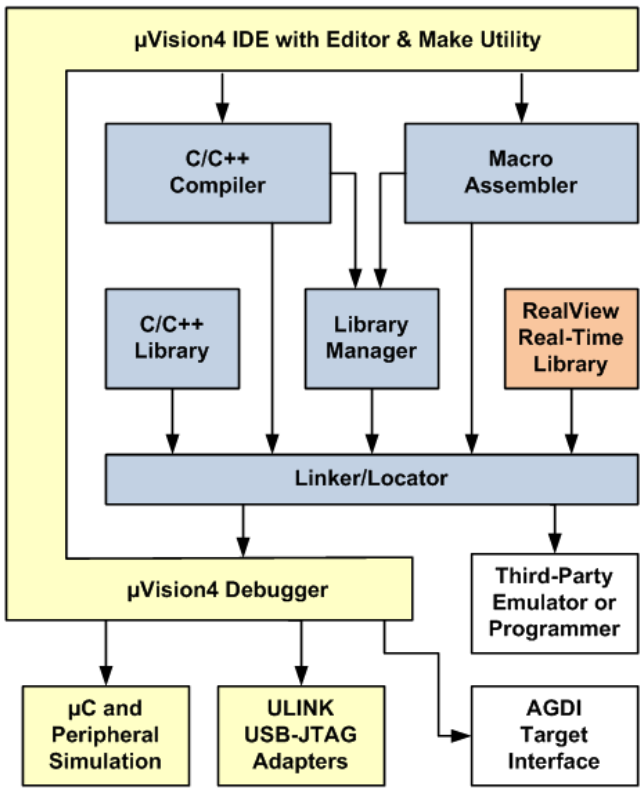


Fig 6.1 Software Development Cycle

µVision Integrated Development Environment (IDE)

The µVision IDE is a window-based software development platform combining a robust editor, project manager, and make facility. µVision supports all the Keil MDK-ARM tools including C/C++ compiler, macro assembler, linker, library manager, and object-HEX converter. Use µVision to create your source files and organize them into a project that defines your target application. µVision automatically compiles, assembles, and links your embedded application. It provides a single focal point for your development efforts.

C/C++ Compiler and Macro Assembler

Source files are created by the µVision IDE and are passed to the C/C++ compiler or macro assember. The compiler and assembler process source files and create relocatable object files.

Library Manager

The library manager allows you to create an object library from the object files created by the compiler and assembler. Libraries are specially formatted, ordered program collections of object modules that may be used by the linker at a later time. When the linker processes a library, only those object modules in the library necessary to create the program are used.

Linker/Locator

The linker/locator creates an absolute ELF/DWARF file using the object modules extracted from libraries and those created by the compiler and assembler. An absolute object file or module contains no relocatable code or data. All code and data reside at fixed memory locations. The absolute ELF/DWARF file may be used:

- To program an Flash ROM or other memory devices.
- With the µVision Debugger for target debugging and simulation.
- With an in-circuit emulator for the program testing.

µVision Debugger

The µVision source-level debugger is ideally suited for fast, reliable program debugging. The debugger includes a high-speed simulator capable of simulating most on-chip peripherals and external hardware. The attributes of the chip you use are automatically configured when you select the device from the Device Database.

6.3 Create KEIL Project File

µVision4 maintains the files that belong to a project in one project file. It takes only a few steps to create a new project file with µVision4:

1. Select Project- New Project from the µVision4 menu. This opens a standard Windows dialog, which prompts you for the new project file name.

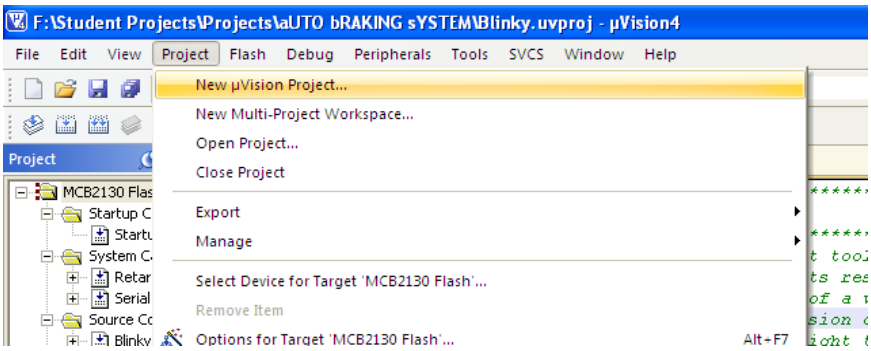


Fig 6.2: Selecting A New project

2. Create a new folder Test.
3. Switch to the new folder and type the project name MyApp. µVision4 automatically adds the extension .uvproj. Click Save.

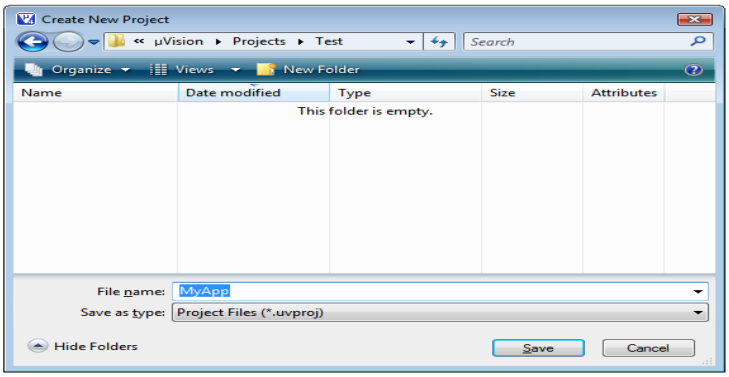


Fig 6.3 : Project location

4. It is good practice to use a separate folder for each project.

6.3.1 Select Device

When you create a new project, µVision4 asks you to select a microcontroller. This step customizes the µVision4 environment with pre-configured options and sets the tool options, peripherals, and dialogs for that particular device. The Select Device for Target dialog shows the µVision Device Database.

1. Select the microcontroller you use. For this example, choose NXP LPC2138. Click OK

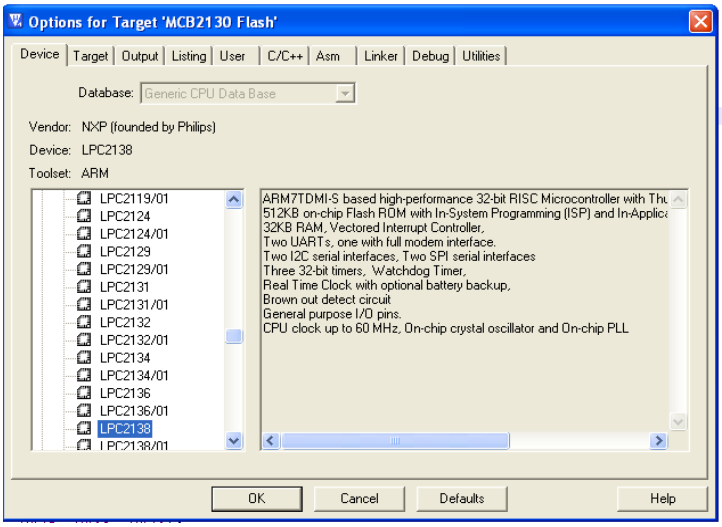


Fig 6.4: Selcting the Microcontroller

2..Click Yes to add the startup code, which is provided for most devices. The startup code provides configuration settings for the selected device.

6.3.2 Set Options for Target

In the Target page, you can specify all relevant parameters of your target and the components of the device you have selected.

Open the Options for Target dialog. Under Xtal(MHz) write 25.0. Click on OK

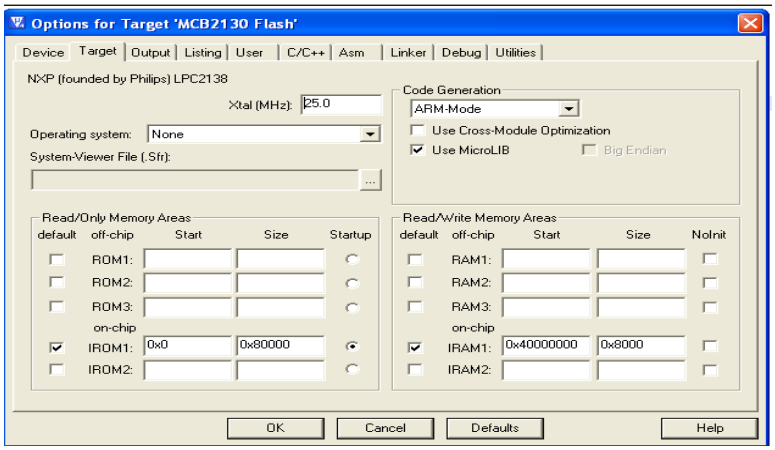


Fig 6.5: Properties of Target

6.3.3 Create Source Files

As in any project, you will structure your code and split it over different files.

1. Create your file from the toolbar or from the File - New menu. This opens an empty editor window where you can enter your source code. µVision4 enables color syntax highlighting, when the file is saved with the extension *.C or *.CPP.
2. Save the file using File - Save As..., and name it, for example, main.C.
3. Write your code.
4. Add the file to your project. Invoke the Context Menu of the Source Group 1 in the Project Window.

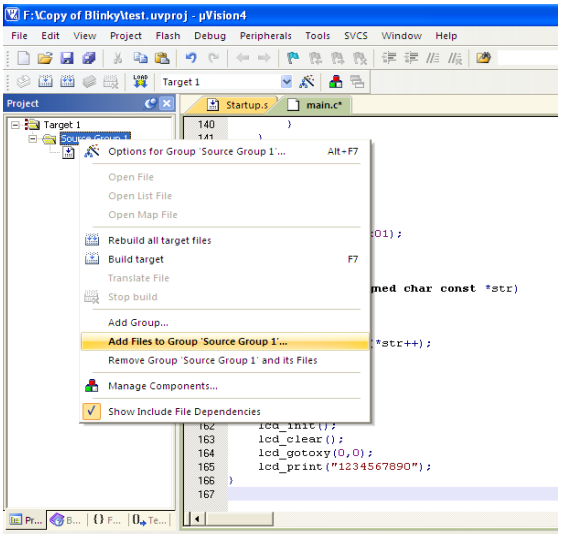


Fig 6.6: Adding Files To source Group

5. The option Add Files to Group opens a standard dialog. Select the file MAIN.C you have just created and click Add. The Project Window should look like the screenshot below.

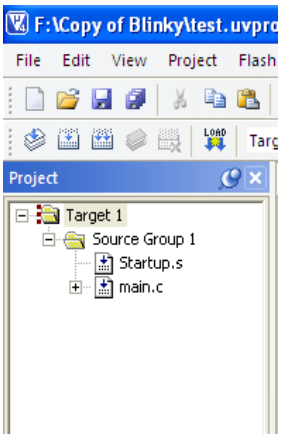


Fig 6.7 : Add Files To Project

6.3.4 Code Compilation

To compile your code goto project menu-> Build Target.

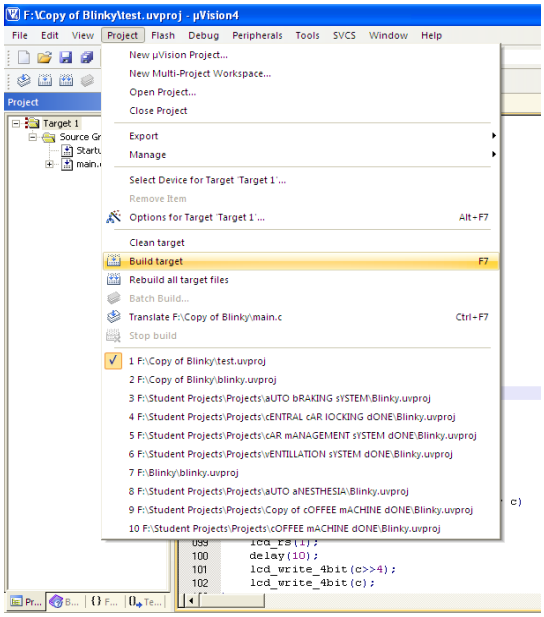


Fig 6.8: Compiling Project

Once the code is compiled successfully you can open Flash Magic software to flash this .hex file into the microcontroller LPC2138.

6.4 FLASH MAGIC SOFTWARE

Flash Magic is a PC tool for programming flash based microcontrollers from NXP using a serial or Ethernet protocol while in the target hardware.

FEATURES

- Straightforward and intuitive user interface
- Five simple steps to erasing and programming a device and setting key options
- Programs Intel Hex Files
- Automatic verifying after programming
- Fills unused Flash to increase firmware security
- Automatically program checksums. Using the supplied checksum calculation routine your firmware can easily verify the integrity of a Flash block, ensuring no unauthorized or corrupted code can ever be executed
- Program security bits

- Check which Flash blocks are blank or in use with the ability to easily erase all blocks in use
- Read any section of Flash and save as an Intel Hex File
- Reprogram the Boot Vector and Status Byte with the help of confirmation features that prevent accidentally programming incorrect values
- Display the contents of Flash in ASCII and Hexadecimal formats
- Single-click access to the manual, Flash Magic home page and NXP Microcontrollers home page
- Use high-speed serial communications on devices that support it.
- Command Line interface allowing use in IDEs and Batch Files
- Manual in PDF format
- Verify Hex Files previously programmed
- Save and open settings
- Control the DTR and RTS RS232 signals to place the device into BootROM and Execute modes automatically (requires hardware support)
- Send commands to place the device in Bootloader mode
- Play any Wave file when finished programming
- Powerful, flexible Just In Time Code feature. Write your own JIT Modules to generate last minute code for programming, for example serial number generation.
- Displays information about the selected Hex File, including the creation and modification dates, flash memory used, percentage of the current device used
- Ethernet bootloader for LPC1xxx/LPC2xxx devices
- Support programming certain LPC1xxx/LPC2xxx devices via Ethernet
- Read the device signature
- Can Be Used On A Production Line
- Python based scripting interface for production line programming and test
- Build your own Flash Magic based applications using the DLLs for C, C++, Python
- Build your own Flash Magic based applications using .NET languages (Windows only)

6.5 Steps to Run Flash Magic

Open Flash Magic.

1. Click on Select Device.
2. Select the microcontroller, LPC2138 in this case. Click OK.

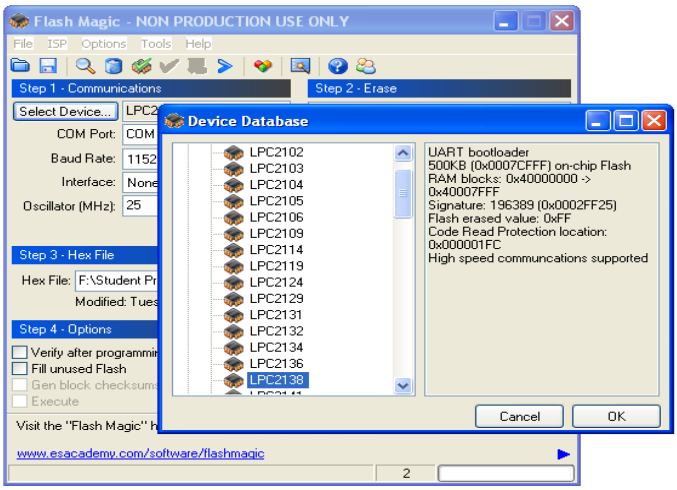


Fig 6.9: Select the microcontroller LPC2138

3. Connect the ARM7 Board with the USB port and power it ON.
4. Install the USB driver and check the COM Port No.
5. Select COM port and set Baud Rate as 115200. Select Oscillator (MHz) as 25.
6. Select Erase all Flash + Code Rd Prot.

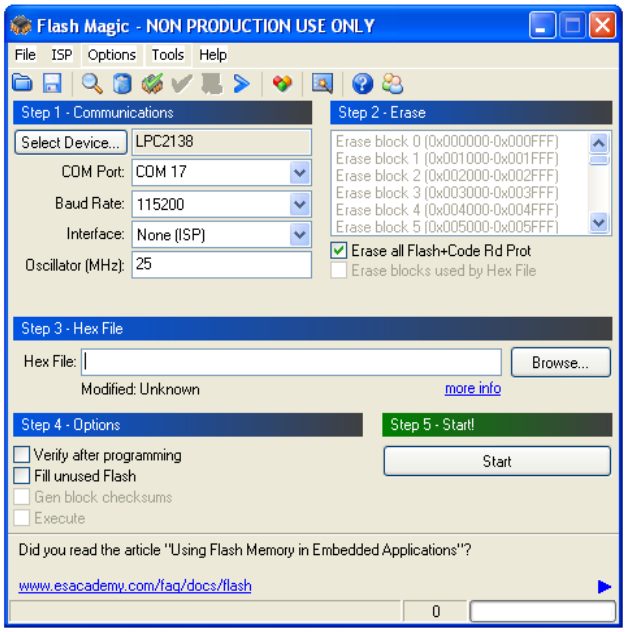


Fig 6.10 : Hex file dumping window

- Browse the Keil generated Hex File.
- Click on Start to flash the Hex Code into the microcontroller.

CONCLUSION

Traffic is the most essential part of modern world as “Time is money”. Hence we must have advanced traffic control system to handle traffic effectively so that we loose least time waiting in traffic. This will enable us to speed up progress of society.

The Quality Assurance Professional can be a part of each of these review process. And most importantly – we all know that it isn’t documented, it never happened. Software Developers must be persuaded to save the results of their hard work for future use.

In modern world because of increasing transport service, there is a huge increase in traffic at junctions. This results in great time loss. Certain incident shows that traffic jam continues in days. This is very detrimental to growth of nation. Hence we require efficient traffic systems that can manage traffic efficiently.

It is basically based on four way traffic. But with slight modifications it can be made for any way systems effectively as it is based on boolean algebra.

REFERENCES

1. Michael J. Vandeman, "Is Traffic Signal Synchronization Justifiable?", April 15, 1994
2. Meyer, Robinson. "Sorry, Los Angeles: Synchronizing Traffic Lights May not reduce emissions". Theatlantic.com. Retrieved 28 January 2013.
3. "Traffic Control Platform beneath Umbrella Installed at Intersections of Pyongyang". KCNA. 2009-08-13. Retrieved 2013-01-29.
4. "Meet The Ladies Of The Pyongyang Traffic Bureau". Jalopnik.com. 2010-03-05. Retrieved 2013-01-29.
5. Remaly, Jake (2012-10-29). "Fallen Trees, Wires, Poles Wreak Havoc on I-287, Local Roads in Morris County - Jefferson, NJ Patch". Jefferson.patch.com. Retrieved 2013-01-29.
6. "Houghton Mifflin Textbook - Chapter Outline". College.cengage.com. Retrieved 2013-01-29.
7. "U. Houston : PSYCH 1300 : Chptr_07". Coursehero.com. Retrieved 2013-01-29
8. www.google.com
9. www.wikipedia.com
10. www.transport-research.info