# RCS

## Railway Crossing System

SDL-2000 Design Contest

**Qing Li** **Robert Probert** **Alan Williams**
**qli@site.uottawa.ca** **bob@site.uottawa.ca** **awilliam@site.uottawa.ca**

School of Information Technology and Engineering (SITE)
University of Ottawa, Canada

May 11, 2002
ReadMe Documentation

## Executive Summary

The Railway Crossing System (RCS) is designed for the SDL-2000 Design Contest at SAM (SDL and MSC) 2002. RCS can control traffic in a variety of railway crossing contexts. Four strategies are provided to handle various situations and requirements. The user can **switch among the strategies dynamically** when the system is running. For the sake of safety, a manual over-ride strategy is included, which allows an emergency response to system malfunctions or unexpected and dangerous circumstances.

RCS is implemented in Telelogic Tau 4.3. A running system is attached in a set of SDL files. MSC files are also attached for documentation and validation purposes, for the:

i) High level requirement model
ii) Critical Scenarios model
iii) Manual Override and Emergency Handling Procedures
iv) A complete set of validation MSCs for detailed design testing.

RCS demonstrates how to use features of SDL and MSC. It also demonstrates how easy it is in SDL-2000 and MSC-2000 to design a real-time system and to validate the design.

This document introduces the four configuration (prioritization) strategies, identifies some errors, omissions and ambiguities in the functional requirements, summarizes the most important requirements by high level MSCs, and gives important critical race scenarios and safety ensuring scenarios for high-risk use cases. We summarize strengths and limitations of this approach, and include a navigation guide and glossary of signals for easy reference.

# RSC
## ReadMe Documentation
## Table of Contents

# RCS Outline

This ReadMe file should be read to understand how the Railway Crossing System (RCS) is designed and implemented. It explains how files are organized to be easily found. A glossary of signals from the environment is included to help users run the simulator easily. There are 8 sections, a Glossary of Signals and an Appendix in this file.

Section 1 introduces what strategies are used and how they work

Section 2 analyzes the requirements, points out ambiguities, errors, and omissions found in the system requirements, and gives assumptions necessary to complete a realistic, working RCS system.

Section 3 gives normal scenarios MSCs to model the high level design.

Section 4 gives high level MSCs to model critical scenarios, including a high-risk critical race for Strategies 2, 3 and 4.

Section 5 introduces how Strategy 1 (Manual Override) works and how emergencies are handled. MSCs are used to model several example scenarios, including a demonstration of how unsafe manual operations are not allowed in RCS.

Section 6 summarizes the strengths of this system.

Section 7 points out some limitations to this model, and gives possible modifications.

Section 8 explains how files are organized. This is a “must read” section before trying to retrieve any files from the “.zip” file.

The Glossary lists all necessary signals from the environment. If you have any problem simulating the system using an SDL tool, please refer to this section first.

The Appendix contains all the figures used in Section 3, 4 and 5. There are also “.msc” files attached. However they are only high-level models. The complete set of MSC files are attached with system files in a separate chapter. These MSCs are generated by running test cases with Simulator. Some important example test cases are also attached. To locate these files, please consult Section 8 of the Navigation Guide.

# 1. Introduction of Strategies

As specified in requirements 9 and 10, several strategies are needed. To make the system flexible and safe, we provide four strategies, which are designed to handle a variety of situations. The user can switch among the strategies **dynamically** when the system is running.

**Note: A guiding principle is to ensure at all times that the train signal is "Green" only when the gate is closed.**

**Strategy 1 ---- Manual Override**

When Strategy 1 is selected, the whole system is controlled manually. This provides a means of guaranteeing the system continue to run, even if a part of the system does not function properly. For example, suppose the gate is closed, but it cannot send a "Closed" signal properly. In this case, the "Green" track signal will not be sent automatically by the RCS controller. Therefore it should be able to be sent manually, to allow the trains to proceed.

To assure safety, any unsafe manual actions will be rejected. As an example, when an "Open" signal is sent to open the gate, the system will first check if there are any moving trains still in the crossing area. If there are none, it will first set track signal to "Red" on all tracks, and then send an "Open" signal to open the gate. If there are one or more trains, it will wait until all the moving trains have passed the crossing area or have stopped, and then open the gate.

Strategy 1 provides complete flexibility to handle any traffic situations. It allows the system not to be constrained by preset control options.

**Strategy 2 ---- Trains Take Precedence**

In this strategy, trains take precedence, including fast trains and slow trains. The gate is normally closed. It only opens when more than one car is waiting and no train is in the crossing area. It closes again either after all cars pass, or when a train is approaching, even if there are still many cars waiting.

This strategy is useful in areas far from cities, where there are normally not many cars at the crossing (long delay between cars). It allows trains to pass faster, and the gate does not need to open and close very often.

**Strategy 3 ---- Normal Strategy**
**Order of Priority: Fast train - many cars - slow train**

In this strategy, fast trains take precedence, then many cars, and then slow trains. The gate is normally open for cars.

When a fast train approaches, close the gate, no matter how many cars are waiting.

When a slow train approaches, if there is more than one car is waiting, stop the train. Close the gate after all cars have passed.

After all trains have passed the crossing area, even if no cars are waiting, open the gate.

This strategy is the major portion of the RCS system. It is used in most real railway crossing systems, and is applicable to a variety of locations and traffic densities.

**Strategy 4 ---- Many Cars Take Precedence**

In this strategy, "many cars" take precedence. The gate is normally opened for cars.

When a train approaches, the system first checks if there are cars waiting. If so, stop the train until all cars pass the crossing area, then close the gate.

As soon as all moving trains pass the crossing area, the system will check whether there are cars waiting. If so, it sets track signal to "Red" on all tracks, and then open the gate, even if many trains are stopped and waiting, or a train is approaching.

This strategy is useful in big cities or during rush hour in areas where there is heavy car traffic.

## 2. Ambiguity in Requirements and Assumptions

**Points of Ambiguities, Errors and Omissions in Functional Requirements:**

The requirement specification has some problematic ambiguities, errors, and omissions, including:

1. No capability is described to detect a car stalled or stopped on the track. This is an extremely dangerous omission from the requirements.

2. The car sensor does not report the exact number of cars waiting.

Point 2 in the specification mentions "number of cars waiting".
Point 4 in the specification says "There is also a sensor that indicates when there is more than one car waiting". This implies that the car sensor only indicates "more than one car waiting" or "not more than one car waiting". It doesn't count "number of cars waiting".

3. The system cannot detect if the train has seen the stopping signal or if the train has started braking.

By specification point 7, "A train will start the braking phase as soon it sees a stopping signal and will restart when the stopping signal goes off".
By specification point 4, "Each track has two sensors: one sensor when a train is approaching and one sensor when the train is leaving the gate." This implies that the controller has no direct input from the train except whether the train is approaching or leaving. This is not adequate for determining if the train has started braking. Thus, the train's behaviour cannot be monitored for compliance to specification point 7.

4. Specification is unrealistic – having only one gate.

In specification points 3, 6, 8, only one gate is referred to. This restricts control of traffic flow to one direction only, i.e., the motorway is a one-way road. This is unrealistic and unnecessarily restrictive.

5. One car sensor cannot detect cars on both sides of the road (in a more realistic situation)

Specification point 4 says "There is also a (ONE) sensor" for cars. This is not adequate to determine the presence of cars waiting on both sides of the crossing in a realistic instance of the contest problem.

6. No capability is described to sense **only one** car waiting.

This appears to present a fairness problem in Strategy 2 (trains take precedence). It is possible in the real world (based on the delay between car arrivals) that a car may have to wait a long time until the next car arrives to cross the intersection even if no trains come. However, in the model, this

problem does not arise because of the stated assumption (specification point 3) of a constant delay between car arrivals.

**Assumptions:**

In order to address these ambiguities and other requirements in a practical way, we make the following assumptions and clarifications:

-Car sensor

Even though it is somewhat unrealistic as noted above, we shall assume that there is only one car sensor, which indicates only "more than one car waiting" or "not more than one car waiting" based on the total number of cars waiting in both directions. It does not count the actual "number of cars waiting". We also assume this car sensor constantly reports to the controller what it has detected.

**Note: In later sections, when we say "car waiting", we mean "more than one car waiting", and "no car waiting", we mean "not more than one car waiting".**

- Trains

The controller does not communicate with the trains directly, but through the two sensors and the train signal.

All trains in the same track receive the same signal ("Red" or "Green"), so that they either all "go" or all "stop" base the signal they received.

- Gate:

There should be one gate at each side of the road. We assume both gates always act the same way like two doors in one elevator. Therefore they are considered as one gate in the system.

We model the gate arm, but not any traffic lights associated with the gate. We assume that the gate takes some interval of time to raise and lower the arm, and it gives signals (flashing lights, clanging bells, etc.) to the motorway before it starts to lower its arm. For safety's sake, we assume that the cars will not try to cross unless the gate is completely opened.

- InSensor:

This is the sensor used to detect and report an approaching train. There is one of these sensors on the arrive part of each track. They are installed at different positions on the fast and slow tracks. To allow for the increased stopping distance required by a fast train, the InSensor will be placed far enough away from the crossing to give the fast train enough time to stop after it receives the Red stopping signal.

- OutSensor:

This is the sensor used to detect and report when a train has passed out of the crossing area completely. There is one of these sensors for each track. For both fast and slow tracks, they can be otherwise identical.

- TrainSignal:

TrainSignal is a conceptual signal, which can be a set of traffic lights on the tracks, lights on the dashboards in trains, or even radio signals. It assures all trains at the same track receive the same signal when they are in the crossing area, which means after they have been detected by the InSensor, and before being detected by the OutSensor.

To make the model more natural and generic, we use a “Green” signal to indicate “Let the train go”, and a “Red” signal to stop a train.

- EmergencySignal:

This is a conceptual emergency-handling process, which can be a set of flashing red lights with alarm, a telephone connection or any other devices used to report emergency automatically to maintenance personnel. EmergencySignal is triggered and turned on when an emergency happens. For example, when the gate cannot close or open, or one of the TrainSignals cannot change to “Red” or “Green” etc.

# 3. High Level Requirements Model

This section models normal scenarios using different strategies. Strategy 1 (Manual Override) is not included here. It will be described in detail in Section 5.

**1). Simplest Generic Scenario (no car) -- Applicable to all strategies**

Start State: Gate is Open and TrainSignal is Red
End State: Gate is Open and TrainSignal is Red

Actions:
1. When a train (fast or slow) approaches, close the gate, and set TrainSignal to Green.
2. After the train passed the crossing area, set TrainSignal to Red, and open the gate.

See Figure HL01.msc

**2). All Trains Have Precedence Scenario -- Applicable to Strategy 2**
(In Strategy 2, normally the gate is closed)

Start State: Gate is Closed and TrainSignal is Green
End State: Gate is Closed and TrainSignal is Green

Actions:
1. Car waiting and no train; open the gate
2. A train (fast or slow) approaches; close the gate
3. The train has passed the crossing area, and cars are waiting; open the gate
4. No cars are waiting; close the gate.

See Figure HL02.msc

**3). Normal Slow Train Urban Scenario -- Applicable to Strategy 3**
(In Strategy 3, normally the gate is open to expedite car traffic)

Start State: Gate is Open and TrainSignal is Red
End State: Gate is Open and TrainSignal is Red

Actions:
1. Cars are waiting; keep the gate open.
2. A slow train approaches while cars are waiting; stop the train and keep the gate open.
3. No cars are waiting, and a slow train is waiting; close the gate, and set TrainSignal to Green.
4. The train has passed the crossing area; set TrainSignal to Red, and open the gate

See Figure HL03.msc

**4). Cars Have Precedence Scenario -- Applicable to Strategy 4**
(In Strategy 4, normally the gate is open to expedite car traffic)

Start State: Gate is Open and TrainSignal is Red
End State: Gate is Open and TrainSignal is Red

Actions:
1. Cars are waiting; keep the gate open.
2. A fast train approaches and cars are waiting; stop the train and keep the gate open
3. No car is waiting; close the gate, and set TrainSignal to Green
4. The train has passed the crossing area; set TrainSignal to Red and open the gate

See Figure HL04.msc

# 4. Critical Scenarios model

This section use different strategies to model critical scenarios, which have the highest risk of collision.

**1). High Risk Critical Race Scenario – Trains have precedence, applicable to Strategy 2**
(In Strategy 2, normally the gate is closed)

Start State: Gate is Closed and TrainSignal is Green
End State: Gate is Open and TrainSignal is Red

Actions:
1. Cars are waiting and no trains; set TrainSignal to Red and open the gate
2. While the gate is being raised, a train (fast or slow) approaches;
Stop raising the gate, and close it instead. Then set TrainSignal to Green
3. After the train passes the crossing area, set TrainSignal to Red and open the gate

See Figure CS01.msc

**2). Critical Race Scenario – Slow and Fast Train Arriving -- Applicable to Strategy 3**
(In Strategy 3, normally the gate is open to expedite car traffic)

Start State: Gate is Open and TrainSignal is Red
End State: Gate is Closed and TrainSignal is Green

Actions:
1. Cars are waiting; keep the gate open.
2. A slow train approaches and cars are waiting; stop the train and keep the gate open
3. A fast train approaches; close the gate, set TrainSignal to Green and let all trains go

See Figure CS02.msc

**3). Critical Race – Multiple Tracks and Trains -- Cars Have Precedence**
-- Applicable to Strategy 4

(In Strategy 4, normally the gate is open to expedite car traffic)

Start State: Gate is Open and TrainSignal is Red
End State: Gate is Closed and TrainSignal is Green

Actions:
1. A train approaches Track 1; close the gate, and set TrainSignal to Green
2. Cars are waiting and the train is still in the crossing area; keep cars waiting, until the train leaves

3. Another train approaches Track 2. Since cars are waiting and cars have precedence, stop the train by setting TrainSignal on Track 2 to Red, and keep the gate closed
4. When the train on Track 1 has passed the crossing area, set TrainSignal on Track 1 to Red, and open the gate for cars
5. When no cars are waiting, close the gate for the waiting train, and set TrainSignal on all tracks to Green.

See Figure CS03.msc

# 5. Manual Override and Emergency Handling

## 5.1 Manual Override – Requires extra checking for safety

This section models the manual override strategy (Strategy 1), which provides the flexibility required to handle various traffic situations. It also shows how unsafe actions are rejected to assure safety.

### 1). Manually Open Gate – Safe Scenario

Start State: Gate is Closed and TrainSignal is Green
End State: Gate is Open and TrainSignal is Red

Actions:
Send Open signal manually
-- If no train in crossing area, open the gate

See Figure MC01.msc

### 2). Unsafe Manually Stopping Gate Closing – Train Approaching

Start State: Gate is Closing and TrainSignal is Red
End State: Gate is Open and TrainSignal is Red

Actions:

Send Open signal manually when the gate is being lowered
-- Check if there are trains approaching.
If there is one, continue closing the gate, set TrainSignal to Green and let the train pass.
After the train passes, open the gate

See Figure MC02.msc

**Note: An unsafe action is not allowed by RCS**

### 5.2 Emergency Handling

This section models handling of emergencies. There are three kinds of emergency: The gate cannot close, the gate cannot open, and TrainSignal in one or more of the tracks cannot be set to Red.

When an emergency occurs, RCS will be locked, an emergency signal will be triggered and turned on automatically to inform trains and cars, and it will also be sent to maintenance personnel.

Unlocking RCS is easy. After fixing the problematic parts, simply use Manual Override to send “Open” or “Close” signal to RCS. It will be unlocked and begin again to operate normally.

#### 1). Gate Cannot Close Scenario – Applicable to all strategies

Start State: Gate is Open, TrainSignal is Red, EmergencySignal is Off
End State: Gate is Open or half way down, TrainSignal is Red, EmergencySignal is On

Actions:

1. A train approaches; try to close the gate
2. Gate cannot close -- either half way down or cannot move at all
3. Try to close it again until timeout -- Train cannot stop safely after this moment
4. Turn on Emergency Signal "GateCannotClose" to inform Trains and Cars and maintenance personnel. RCS is locked

See Figure EM01.msc

#### 2). Gate Cannot Open Scenario – Applicable to all strategies

Start State: Gate is Closed, TrainSignal is Green, EmergencySignal is Off
End State: Gate is Closed or half way up, TrainSignal is Green, EmergencySignal is On

Actions:

1. If no train in the crossing area and cars are waiting; set TrainSignal to Red and try to open the gate
2. Gate cannot open -- either half way up or cannot move at all
3. Try to open it again for 3 times
4. The gate still does not open; turn on Emergency Signal "GateCannotOpen" to inform maintenance personnel, and set TrainSignal to Green to let all upcoming trains pass

See Figure EM02.msc

**3). TrainSignal Cannot Set To Red Scenario – Applicable to all strategies**

Start State: Gate is Closed, TrainSignal is Green, EmergencySignal is Off
End State: Gate is Closed, TrainSignal is Green, EmergencySignal is On

Actions:
1. If no train in crossing area and cars are waiting; set TrainSignal to Red
2. One of the TrainSignals cannot be set to Red
3. Try it again for 3 times
4. Still cannot set to Red; turn on Emergency Signal "TrainSignal Cannot Set to Red" to inform maintenance personnel and set TrainSignal to Green to let all upcoming train pass

See Figure EM03.msc

**4). Unlock RCS Scenario after emergency – Applicable to all strategies**

The Manual Override mode can be used to send “Open” or “Close” signal to the RCS. The gate will be open or closed, respectively, and the RCS will be unlocked and operate normally.

See Figure EM04.msc and Figure EM05.msc

# 6. Summary of Strengths

1. The number of tracks and the number of fast tracks are modelled as constants, which can be changed before system start up. This design makes the system **generic**. It can be used at a variety of crossing intersections with different number of tracks.

2. Strategies are defined as signals from the environment. This makes it possible to **change strategies dynamically when the system is running**.
   As an alternative, different strategies can be modelled as subclasses of a super class, which conform to object-oriented principles. However, when a different strategy is needed, the system has to be restarted. Switching to a different strategy dynamically becomes impossible.

3. Timers are set up at all critical transitions to monitor proper signal exchange. If a signal gets lost, or a part of the system does not work properly, it will be reported right away, and a corresponding action will be taken automatically. This **assures safety** at all times, which is extremely important for this system.

4. Emergency handling provides a mechanism to report all high-risk errors, which provides another means of assuring the system's **safety.**

5. Manual Override provides **flexibility** to handle various traffic situations. However, Manual Override has a potential risk of allowing unsafe actions. In RCS, any unsafe action will be rejected.
   Manual Override also makes it easy to **restart the system** after an emergency.

**Thus, this RCS provides great flexibility and ease of use, while simultaneously ensuring a high degree of safety.**

## 7. Limitations and Possible Corresponding Modifications

Specification point 3 says "The pattern of cars approaching is simply given by a constant delay between the cars". To make the system more generic, we didn't follow this requirement.

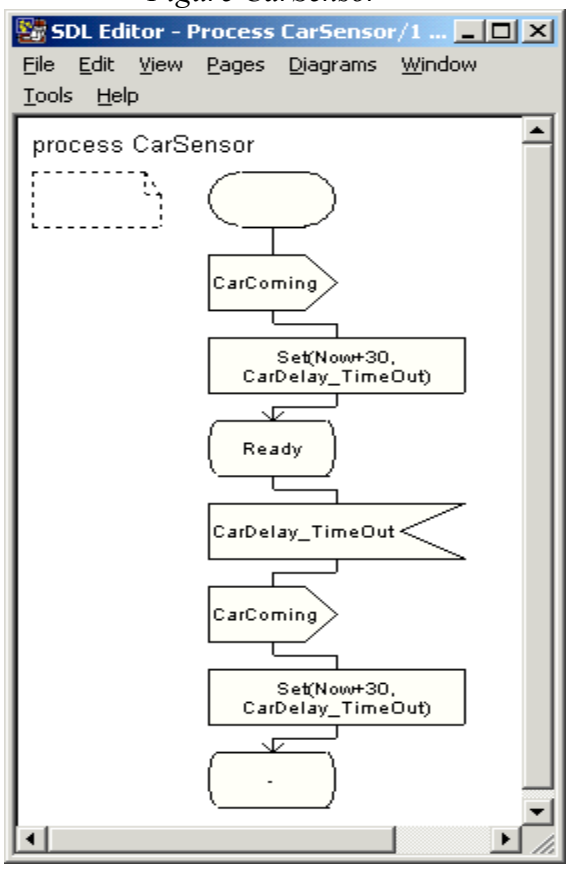
In reality, cars can arrive within a very short or no delay in rush hour in urban areas, or they can arrive infrequently in a rural area. To make it more realistic, we model cars as signals from environment, so that they can arrive in any density.

The requirement above in specification point 3 is a good way to simplify the RCS system, making it easy to simulate. To implement it, we only need to make a small change in the existing model. The change is as follows:

Instead of using the existing CarSensor process, we change it as shown in Figure CarSensor below. In this process, CarSensor does not receive signals from the environment, but only sends signals out after every "CarDelay_TimeOut", which is the "constant delay" mentioned in the requirement.

In the Controller process, we need to add a variable "CarCounter" to count how many cars are waiting. It increments by 1 every time it receives a CarComing signal from CarSensor, and is reset to zero as soon as the gate is open.

*Figure CarSensor*



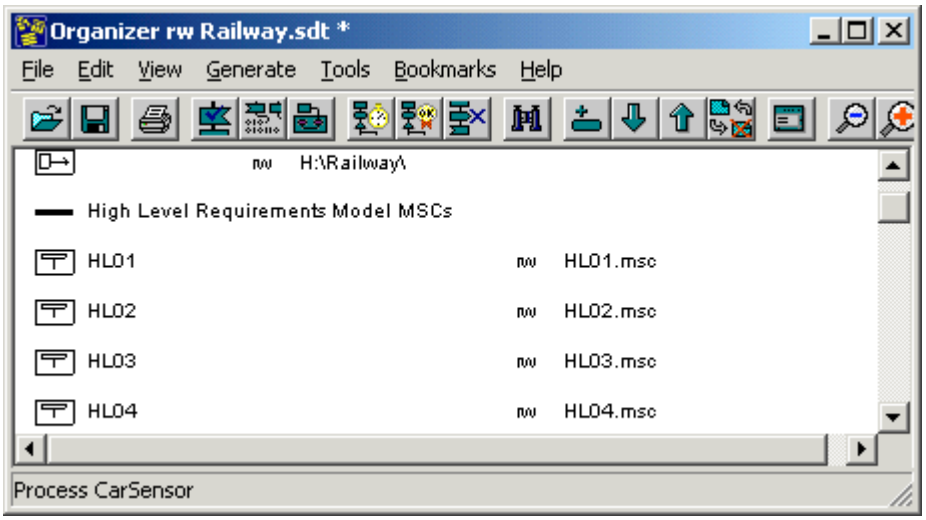
# 8. Navigation Guide

This section explains how files are organized to help users to find them easily.

**System file:**
Railway.sdt – This is the only file you need to find, to run the system using Telelogic Tau. After unzipping the .zip file, double click the Railway.sdt file. The Organizer will open, and all files will be listed in the Organizer window.

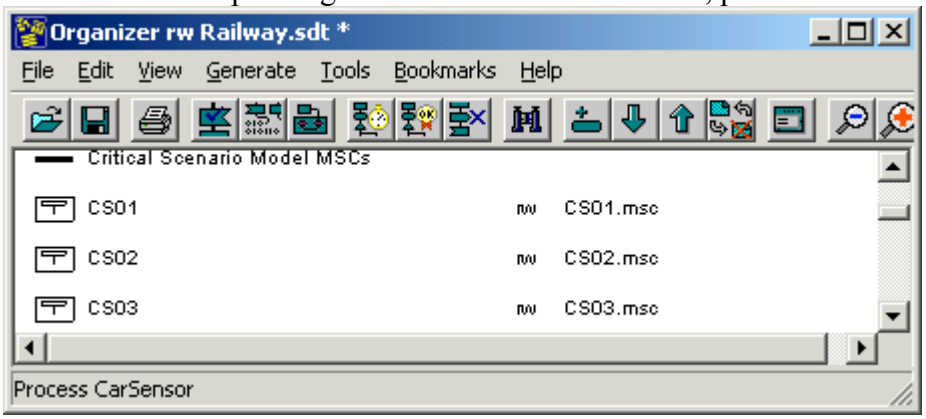
**High Level Requirement Model MSCs:**

These MSCs are organized in "High Level Requirement Model MSCs" chapter in the Organizer.
To find the corresponding models that use these MSCs, please refer to Section 3.



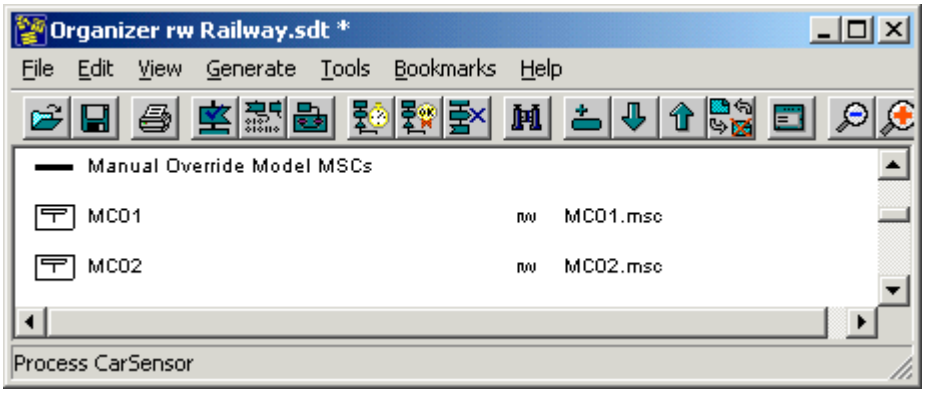

**Critical Scenarios Model MSCs:**

These MSCs are organized in "Critical Scenarios Model MSCs" chapter in the Organizer.
To find the corresponding models that use these MSCs, please refer to Section 4.



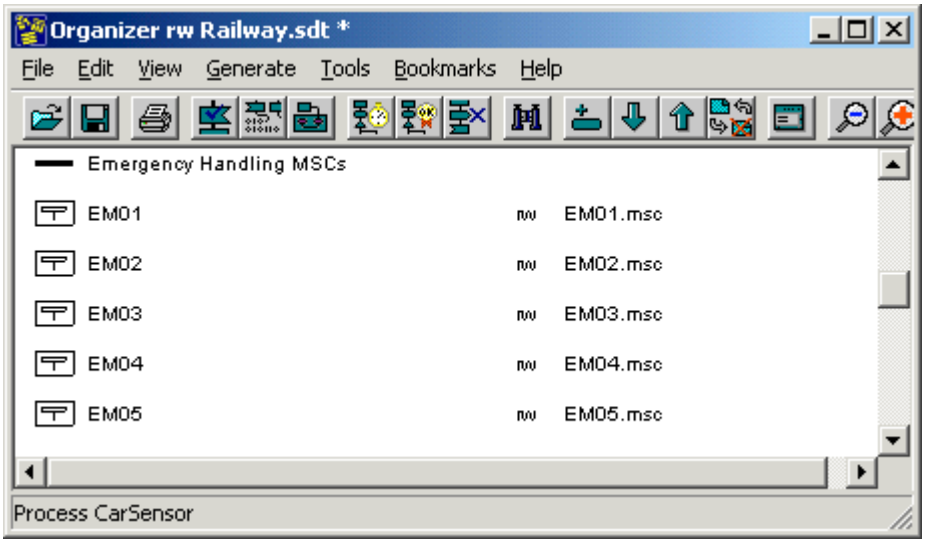
**Manual Override Model MSCs:**

These MSCs are organized in “Manual Override Model MSCs” chapter in the Organizer.
To find the corresponding models that use these MSCs, please refer to Section 5.




**Emergency Handling MSCs:**

These MSCs are organized in “Emergency Handling MSCs” chapter in the Organizer.
To find the corresponding models that use these MSCs, please refer to Section 5.

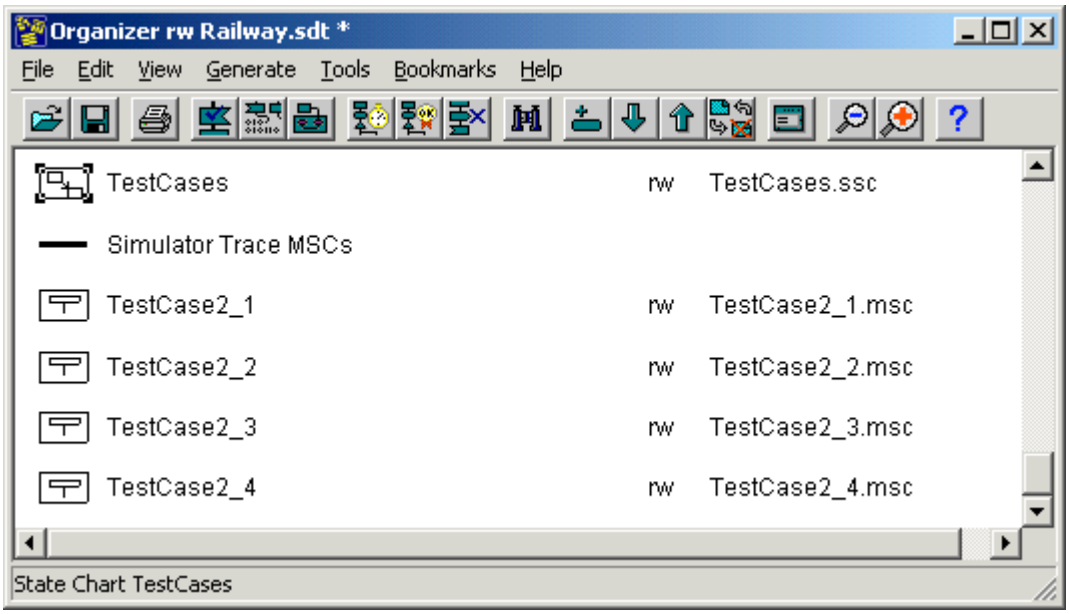



**Test Cases:**

Test cases are designed using state charts, which are saved in TestCases.ssc file. It is also listed in the Organizer.

**Simulator Trace MSCs:**

Simulator trace MSCs are generated by using the simulator to run test cases. They are organized in "Simulator Trace MSCs" chapter in the Organizer. To find corresponding test cases, please refer to the test case state charts in TestCases.ssc file.



## 9. Glossary of Signals from Environment

To assist in running RCS simulations, here we list the signals from the environment to the system, and give brief explanations.

**Strategies:** Four Strategies are used in this system. They are:
1 -- Strategy 1 (Manual Override)
2 – Strategy 2 (Trains Take Precedence)
3 – Strategy 3 (Normal Strategy --Order of Priority: Fast train - many cars - slow train)
4 – Strategy 4 (Many Cars Take Precedence)

**YesE:** More than one car is waiting

**NoE:** Not more than one car is waiting

**TrainC:** A train approaches
The number of fast tracks is defined as a constant, "NoOfFastTracks".
If you want to make it a fast train, send it to a track whose "TrackNumber" is less than or equal to NoOfFastTracks;
Otherwise, send it to a track where the "TrackNumber" is greater than NoOfFastTracks

**TrainL:** A train has just passed the crossing area completely

**EOpen:** A signal from Manual Override, which enforces the gate open, whichever state the gate is in. This signal has a potential risk of unsafe operations. To assure safety, it will not be accepted except in Strategy 1.

**EClose:** A signal from Manual Override, which enforces the gate close, whichever state the gate is in.

**EGreenRQ:** A signal from Manual Override, which sets TrainSignal in a specific track to Green, whichever state the TrainSignal is in.

**ERedRQ:** A signal from Manual Override, which sets TrainSignal in a specific track to Red, whichever state the TrainSignal is in.

**EStopS:** A signal from Manual Override, which turns emergency signal on, closes the gate and sets TrainSignal on all tracks to Red.

# Appendix

*Figure HL01.msc*

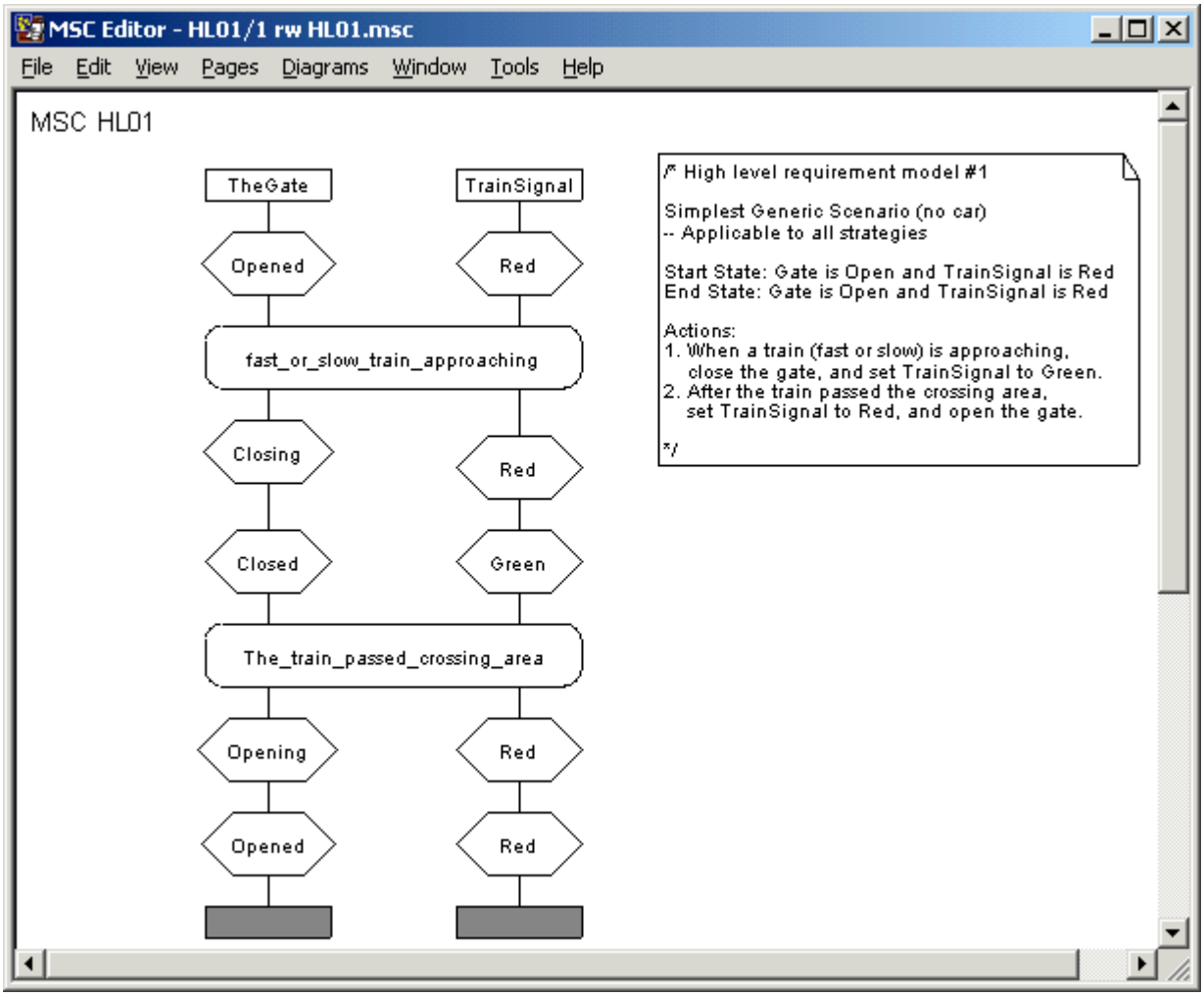


*Figure HL02.msc*

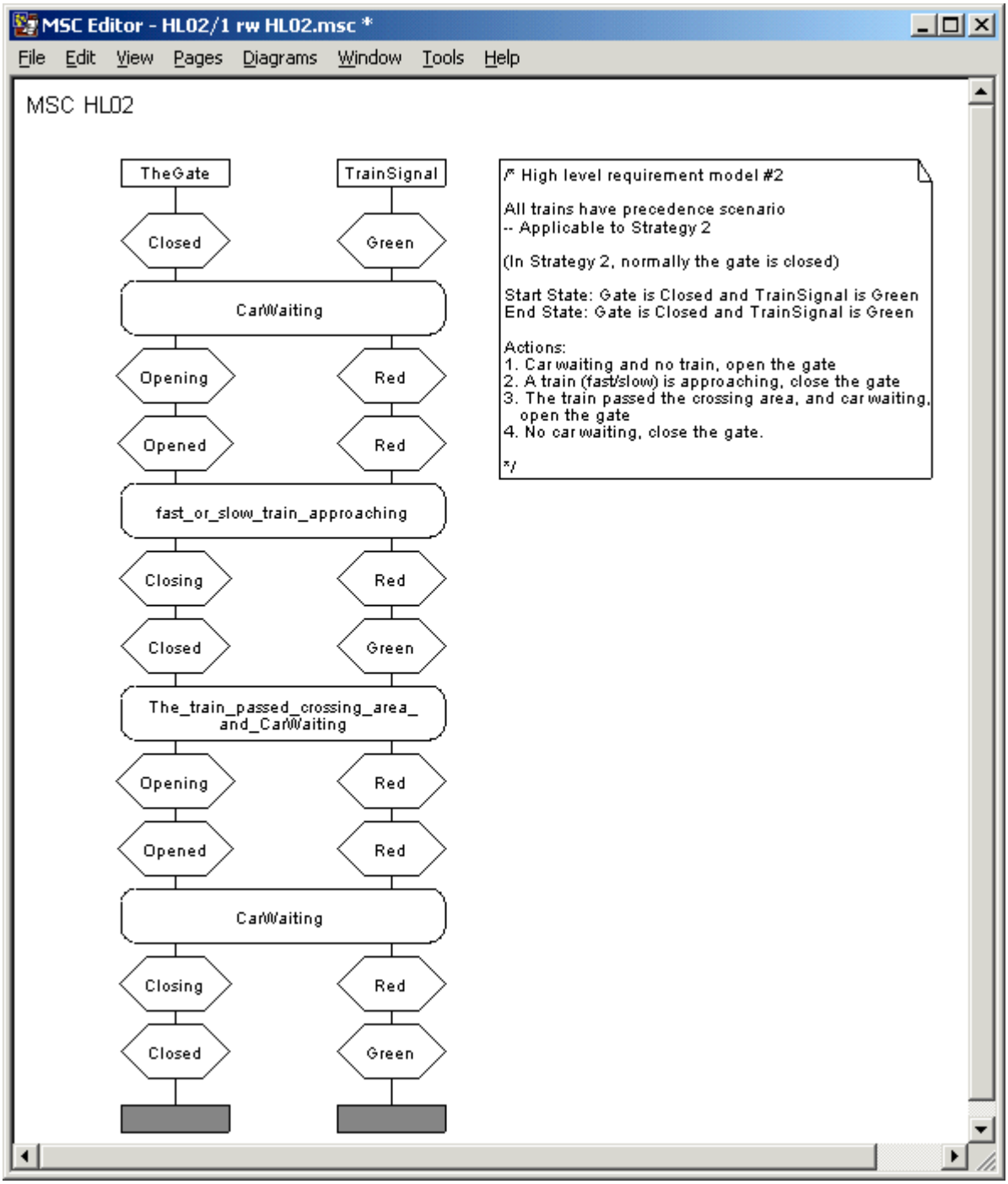


*Figure HL03.msc*

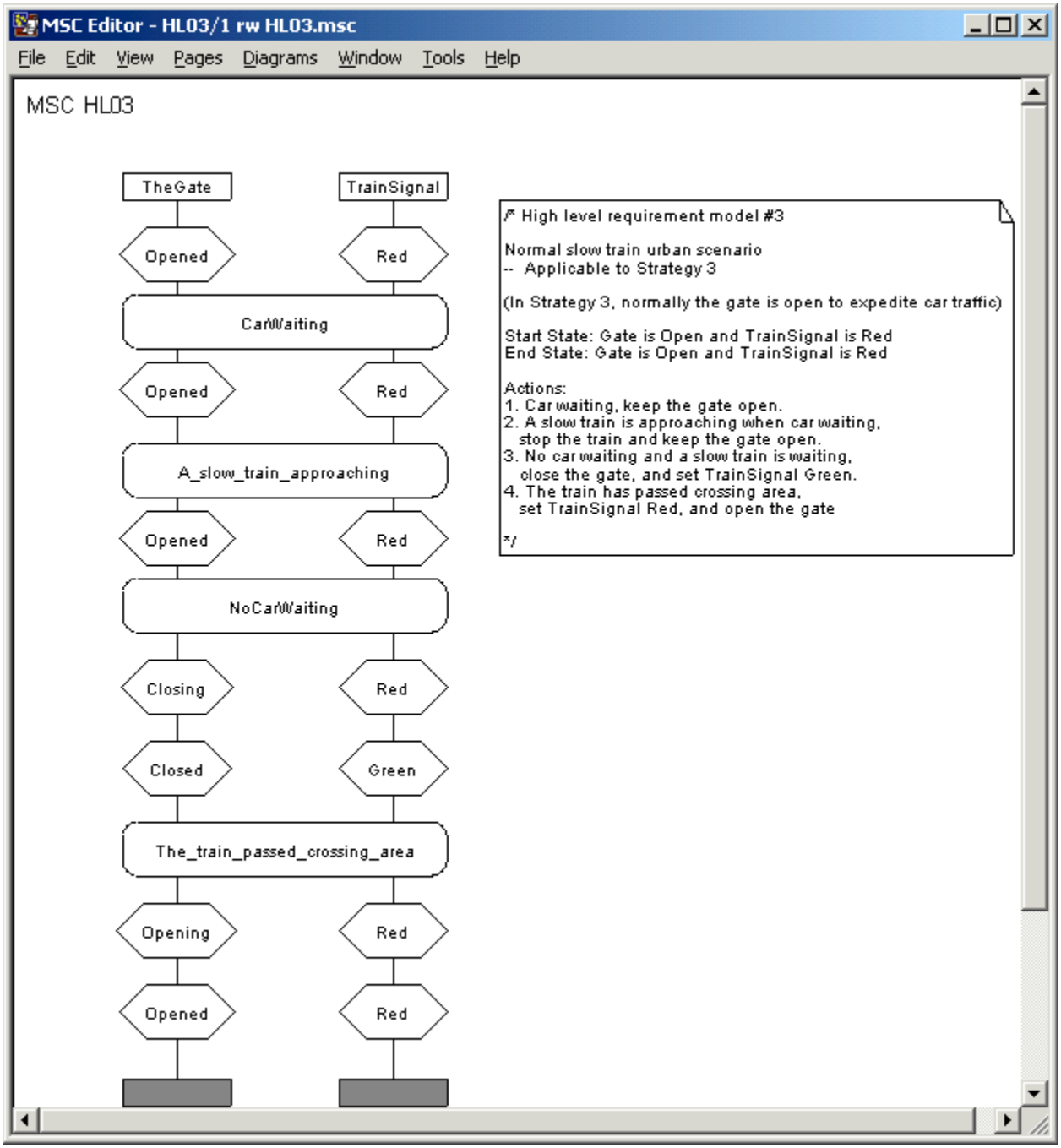


*Figure HL04.msc*

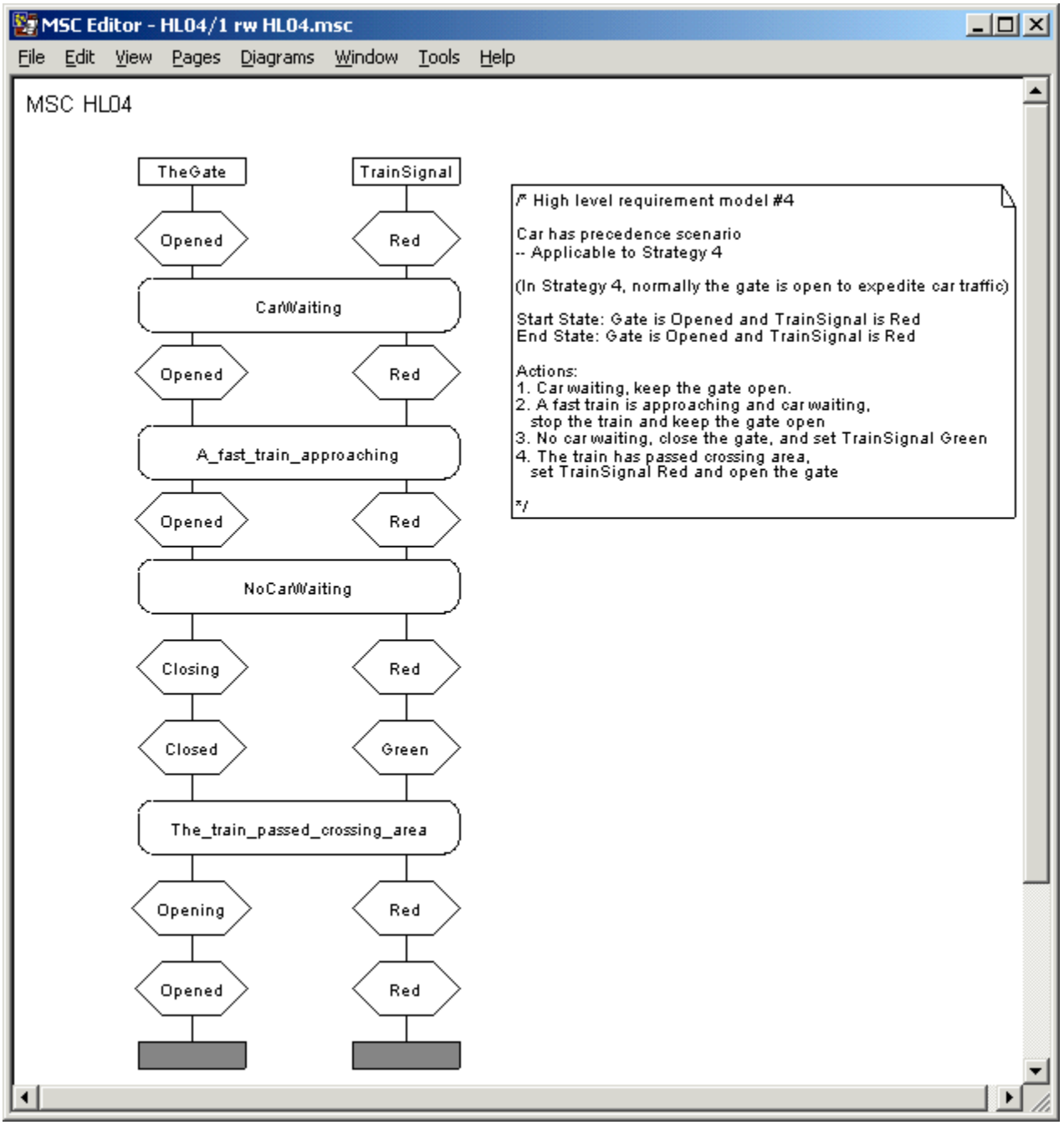


*Figure CS01.msc*

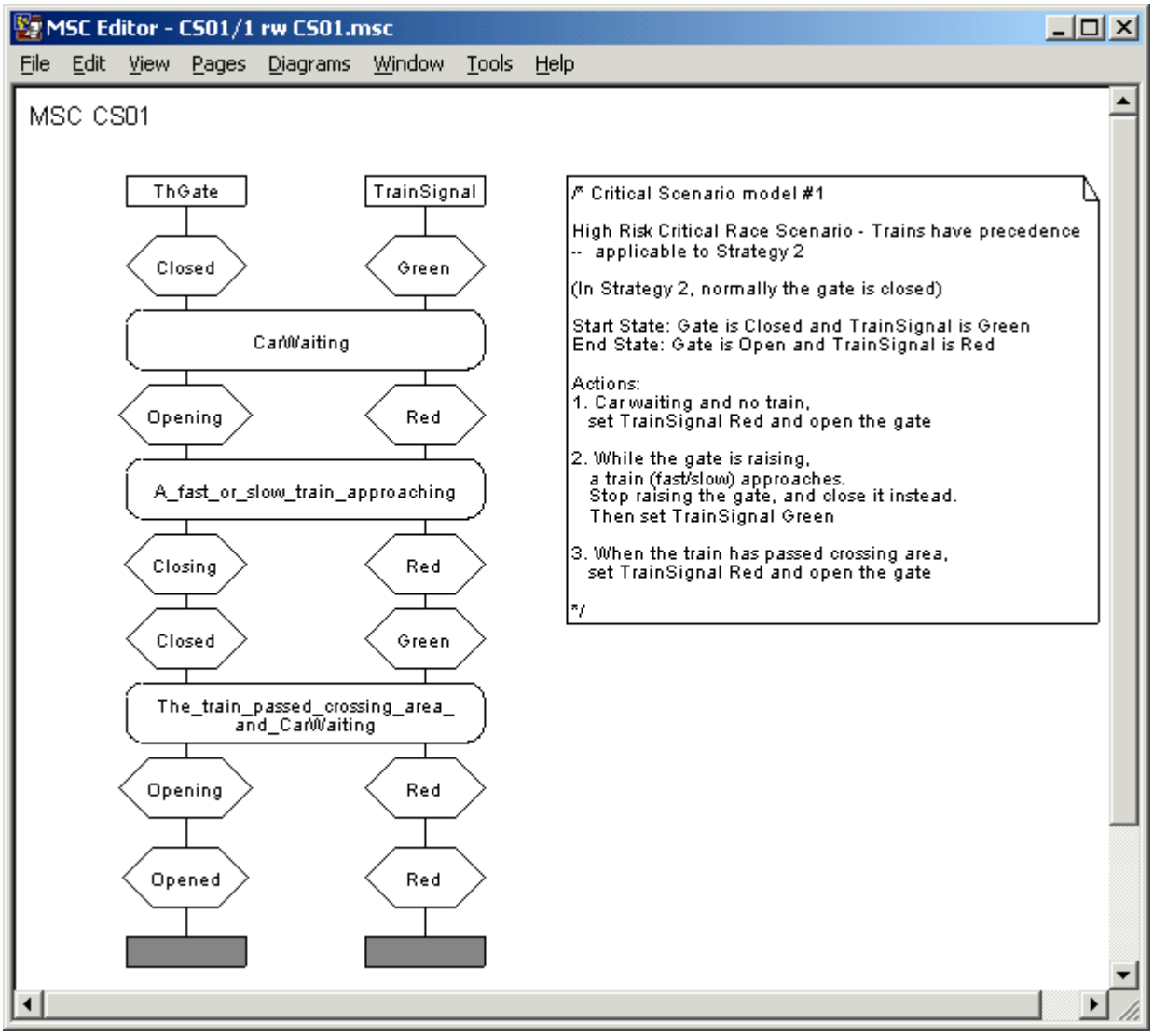


*Figure CS02.msc*

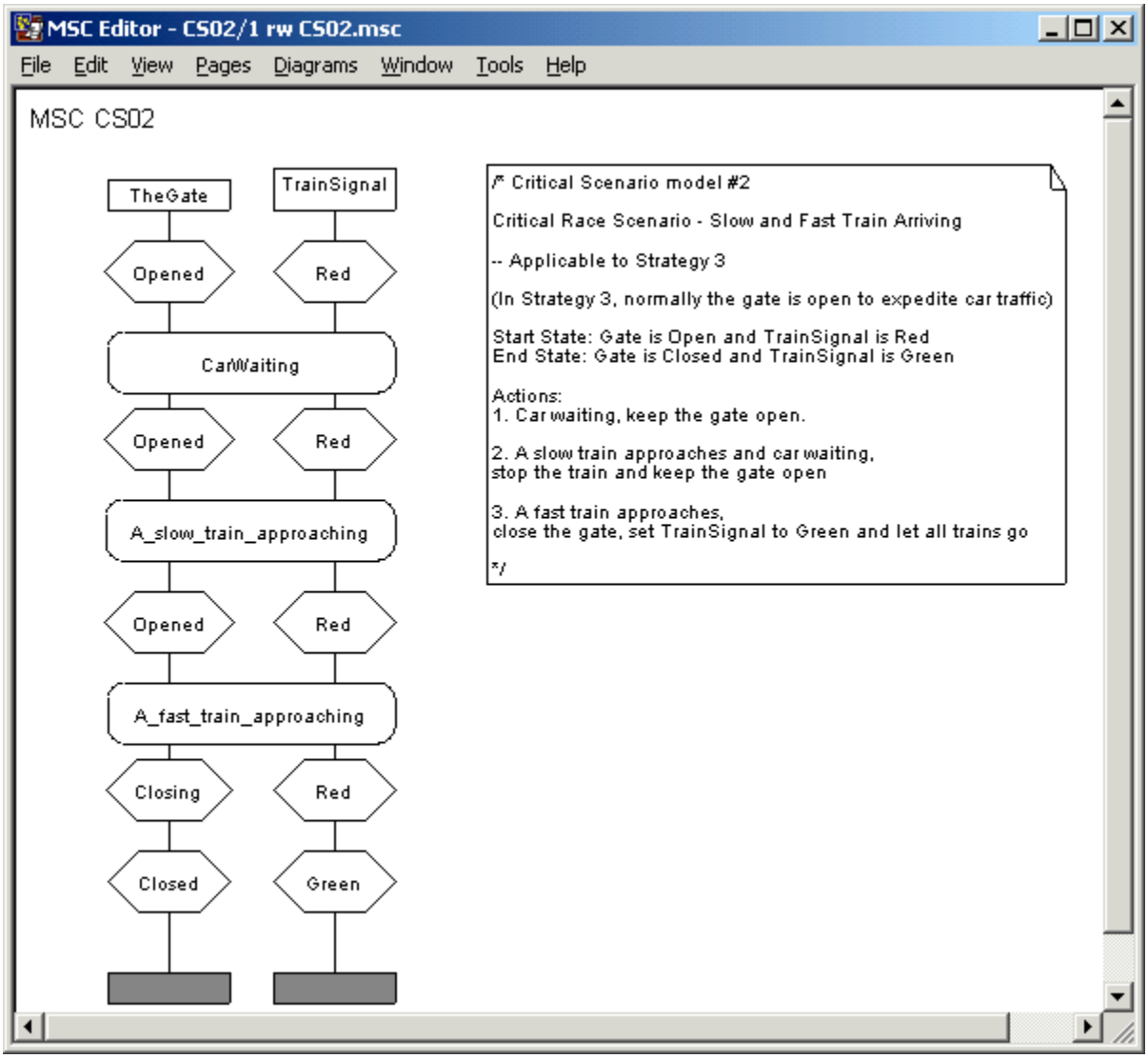


*Figure CS03.msc*

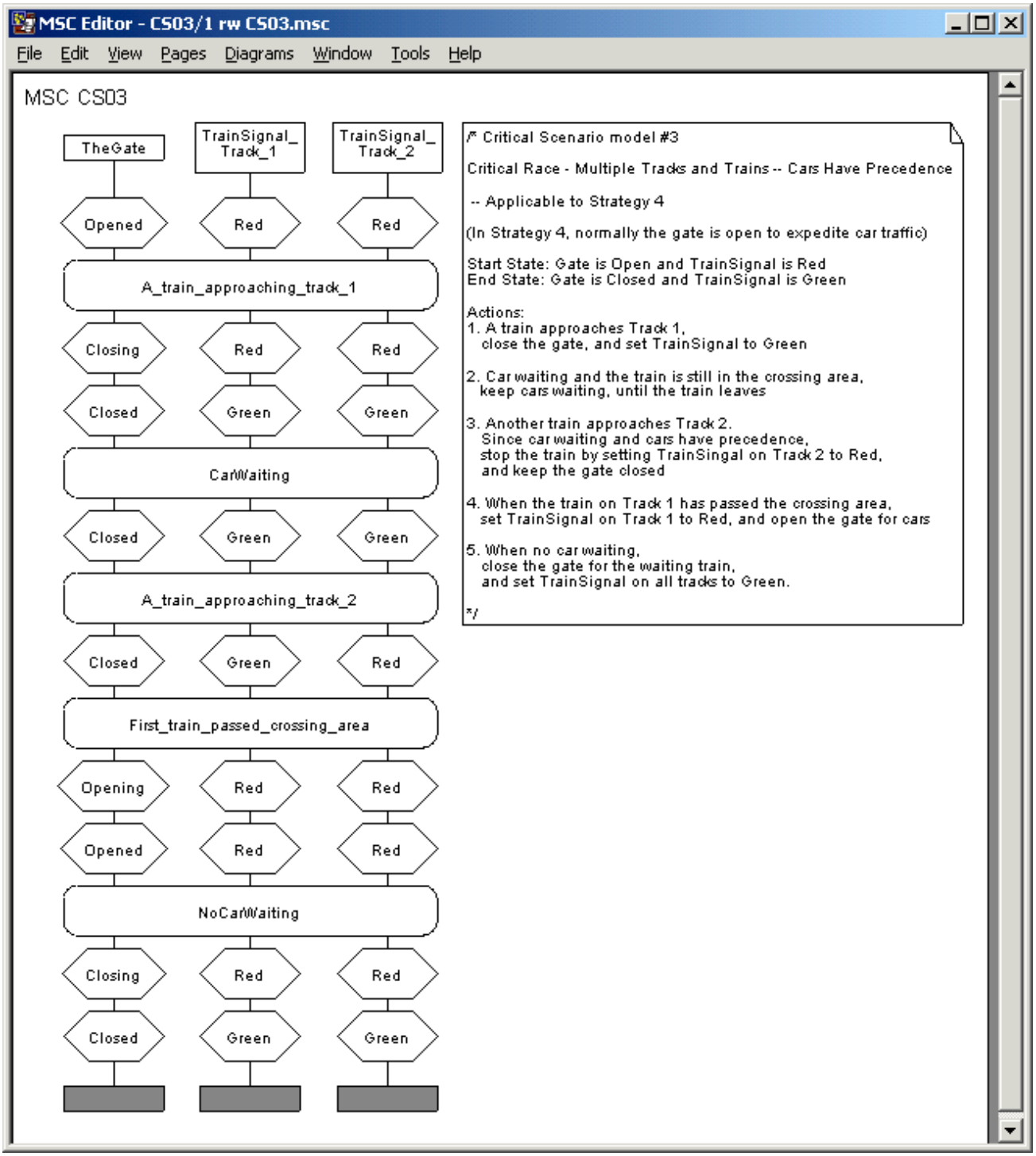


*Figure MC01.msc*

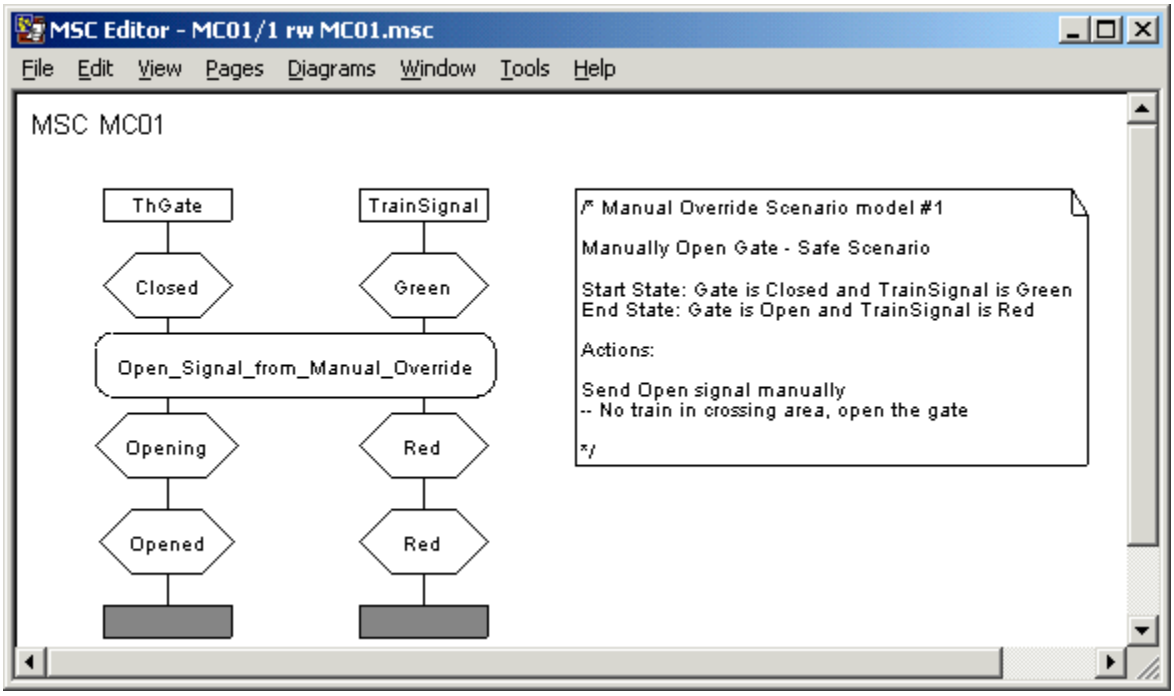


*Figure MC02.msc*

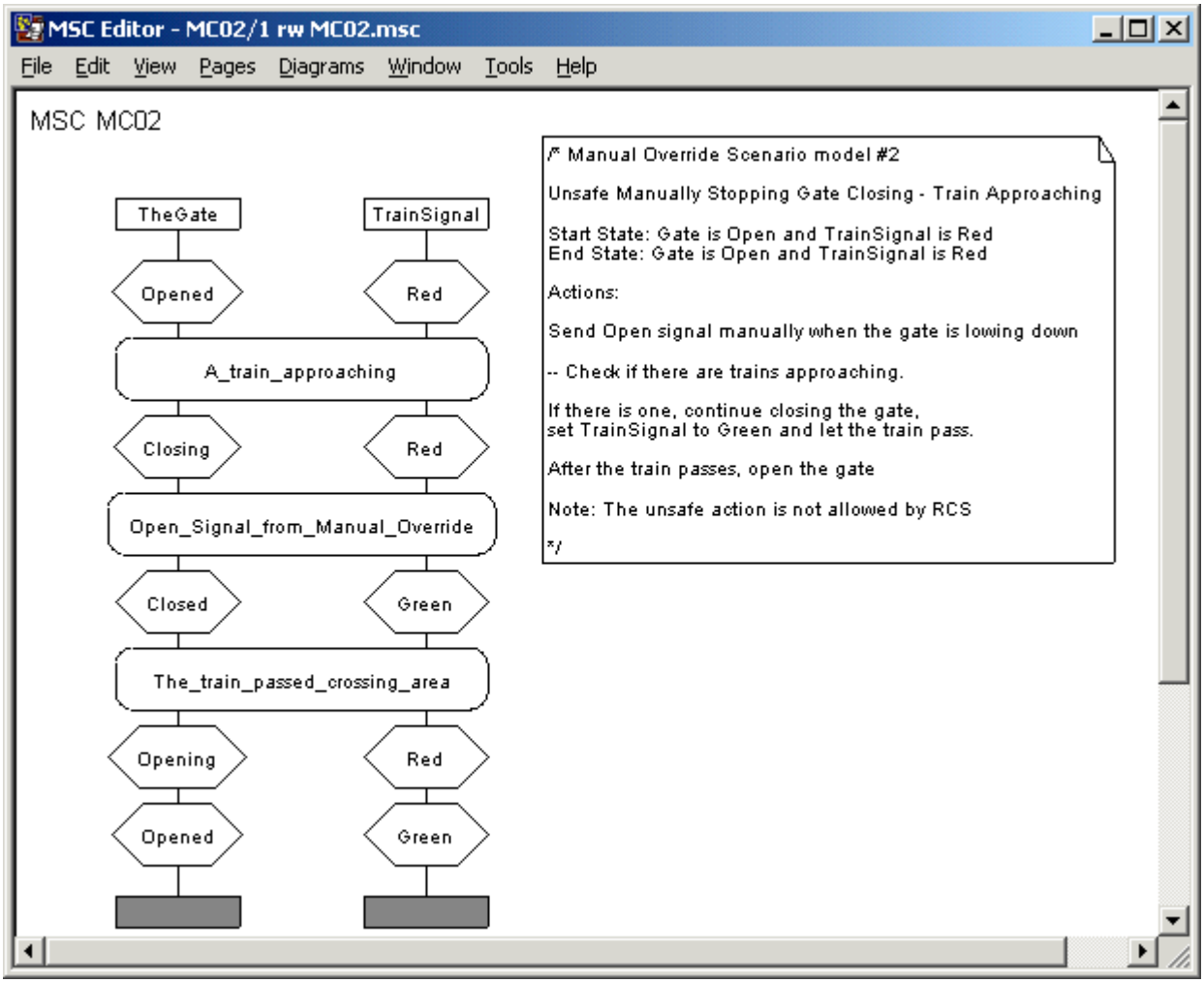


*Figure EM01.msc*

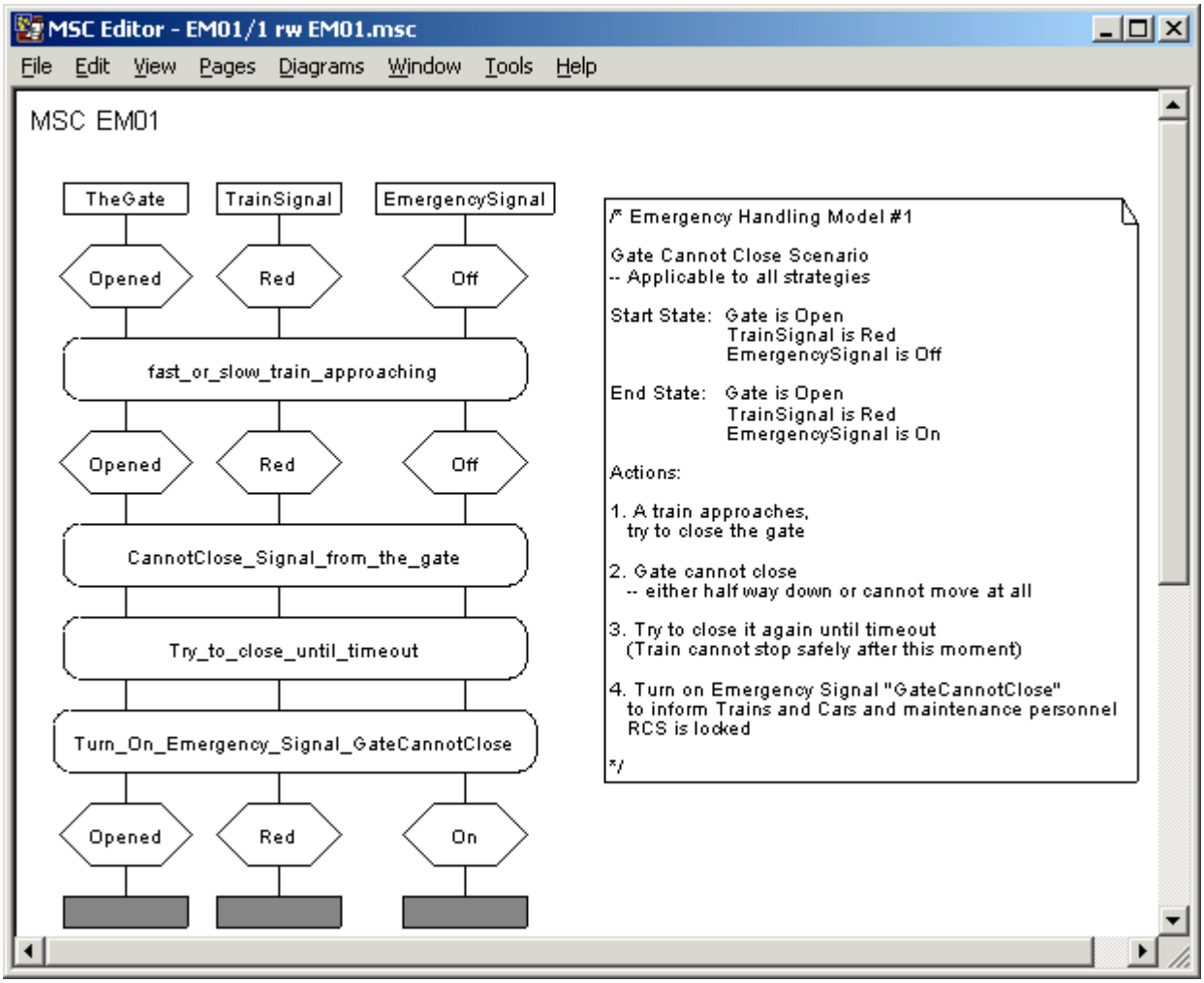


*Figure EM02.msc*

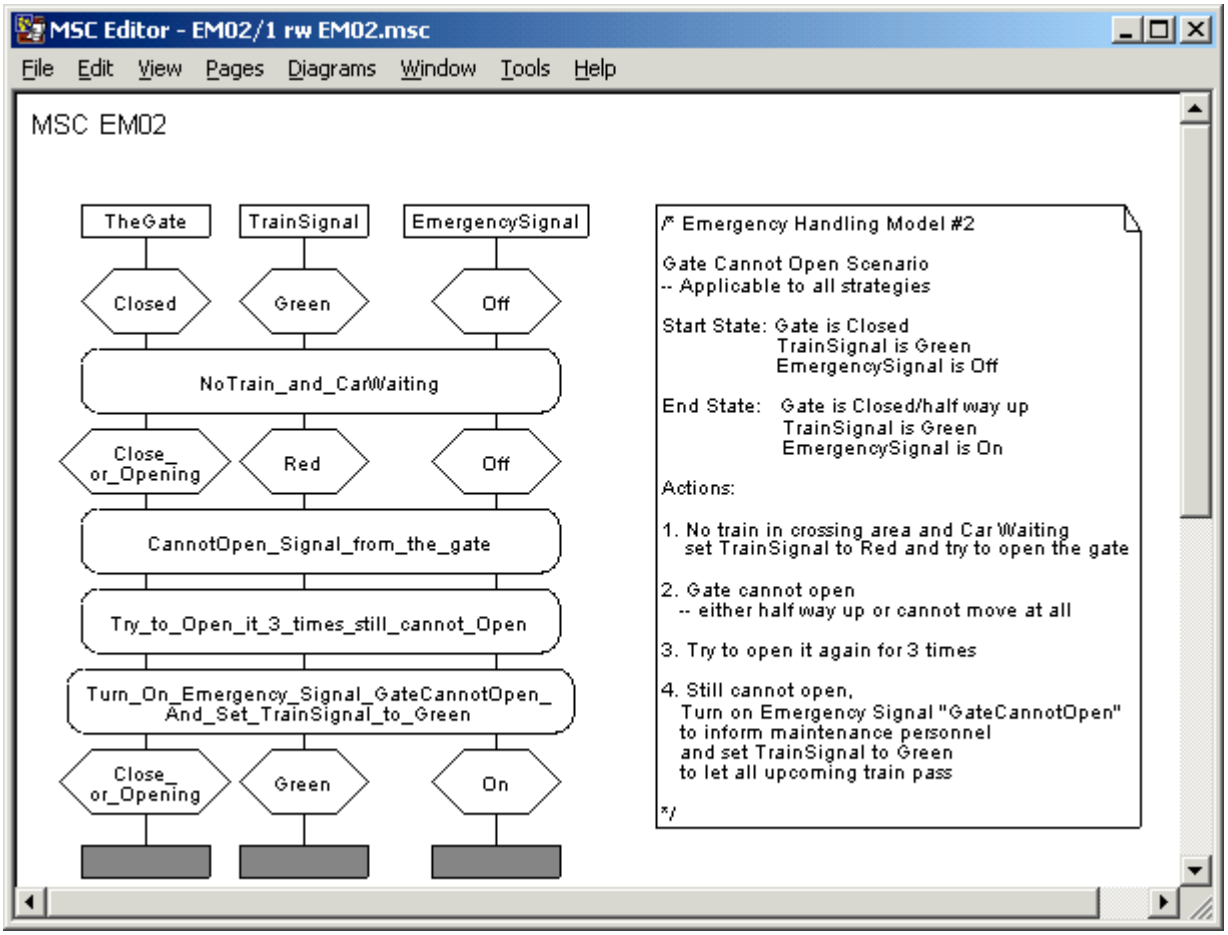


*Figure EM03.msc*

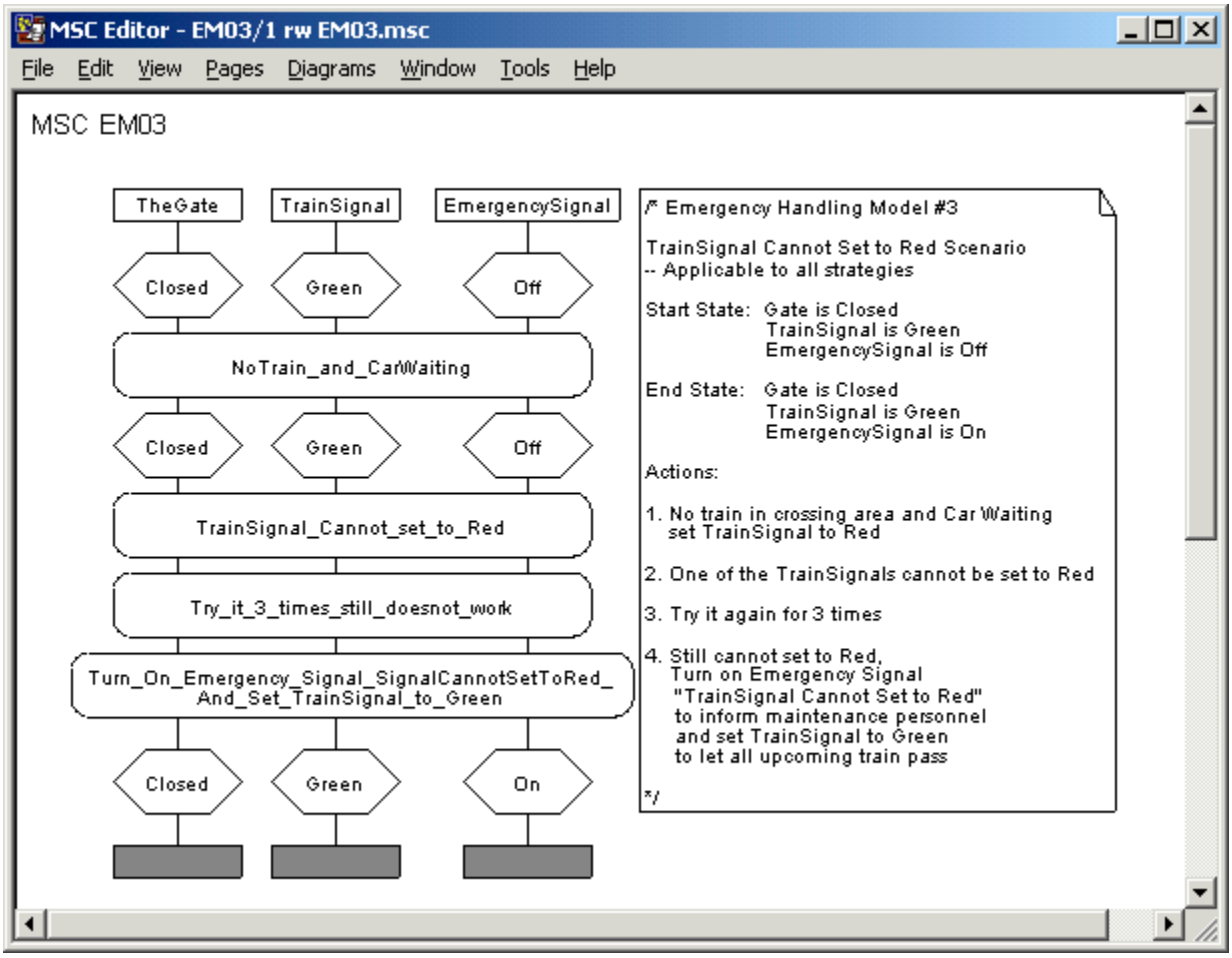


*Figure EM04.msc*

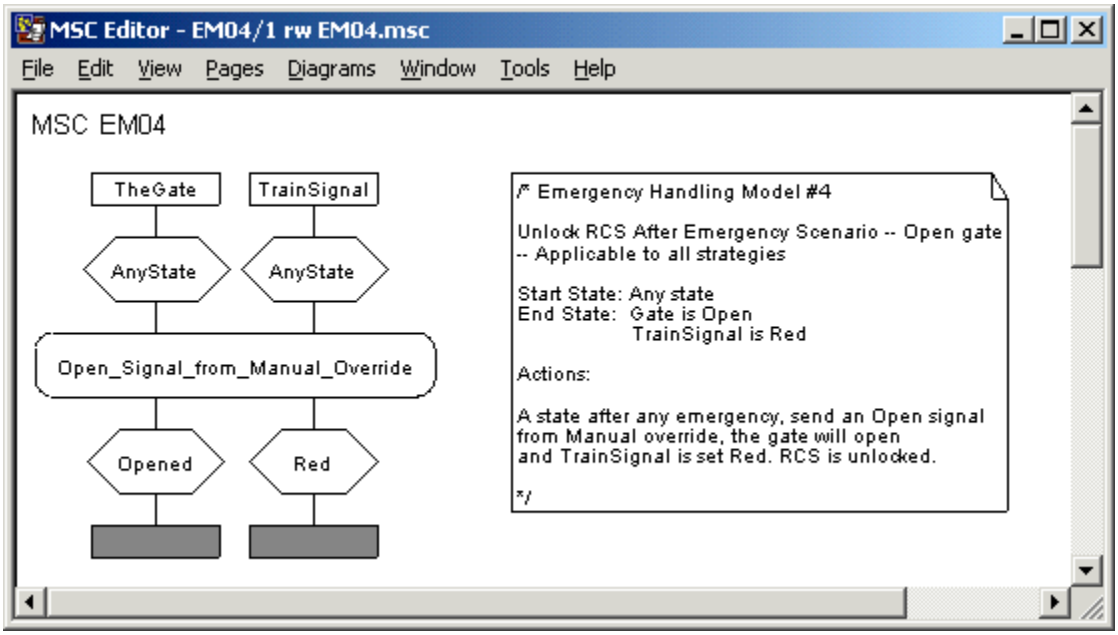



*Figure EM05.msc*