13th IFAC Symposium on Control in Transportation Systems
The International Federation of Automatic Control
September 12-14, 2012. Sofia, Bulgaria

Automatic Interlocking Table Generation for Railway Stations using Symbolic Algebra

Uğur Yıldırım*, Mustafa S. Durmuş, Mehmet T. Söylemez****

**Control Engineering Department, Istanbul Technical University, Istanbul, Turkey, (Tel: +90-544-5984158; e-mail: yildirimu@itu.edu.tr).*
***Control Engineering Department, Istanbul Technical University, Istanbul, Turkey, (e-mail: {durmusmu, soylemezm}@itu.edu.tr)*

Abstract: Interlocking is the most important component of a signalization system, which ensures safe travel and transportation. An interlocking system helps movement of trains on desired routes in fixed block signaling systems. The first step in the design of interlocking systems is to generate interlocking tables. Generation of interlocking tables for small scaled stations or railway yards can be realized easily. However, when the topology of the station is complicated, generation of the corresponding interlocking table becomes considerably difficult. In this study, a program that automatically generates interlocking tables for a given railway yard is explained. One of the main advantages of the program is that, switches and signal lights are placed automatically for a given station or railway yard depending on the topology.

Keywords: Railway Signalization Systems, Interlocking Table, Traffic control, Symbolic Algebra

1. INTRODUCTION

An interlocking program is a kind of safety related software which controls all components of a railway signalization system such as track circuits, switches and signal lights. As mentioned in (Haxthausen et. al., 2010), the interlocking also allows reservation of routes on which the trains can safely move. Before designing an interlocking system, first route specifications have to be determined correctly. All route specifications are shown in a table named interlocking table. Therefore, the first step in the design of an interlocking system is the construction of the interlocking table.

For small sized railway stations, constructions of the interlocking table is relatively easy, but when the size of the railway station gets bigger, construction of the interlocking table becomes very time-consuming and complex. This increases possibility of making fatal logical errors, which are very difficult to detect in the later stages of the design and test process. In the literature, some studies are made about automatic generation of the interlocking table (Kuzu et. al., 2011) and verification (Fokkink and Hollingshead, 1998), (Petersen, 1998), (Tombs et. al., 2002), (Winter, 2002), in order to eliminate this kind of problems. Formal methods like Abstract States Machines (Winter and Robinson, 2003), Finite State Machines (Miradabi and Yazdi, 2009), Coloured Petri Nets (Anunchai, 2010) or Boolean Logic (Roanes-Lozano et. al., 2010) is also considered in some studies. In all these studies, the generated interlocking table does not give detailed information and before the interlocking table generation, the components of the railway station have to be defined manually.

In this study, an automatic interlocking table generator programme, which was written using symbolic algebra in Mathematica™ programming environment, is introduced. The main advantage of the programme is its simple user interface that allows defining a railway station topology by only drawing lines. In section 2, brief information is given about all components of a railway signalization system. Basic principles of the automatic interlocking table generator programme are explained in section 3. An interlocking table is generated for a model railway station by using the programme. Finally, conclusions are given in section 5.

2. RAILWAY SIGNALIZATION COMPONENTS

A railway signalization system consists of a traffic control centre (TCC), an interlocking system and a railway station. Traffic control centre makes route reservation requests for trains to move on the railway station. Interlocking system performs required actions for the incoming route requests. Interlocking system also checks error conditions of the railway station components such as track circuits, switches and signals. A simple exemplary railway station is given in Fig. 1.

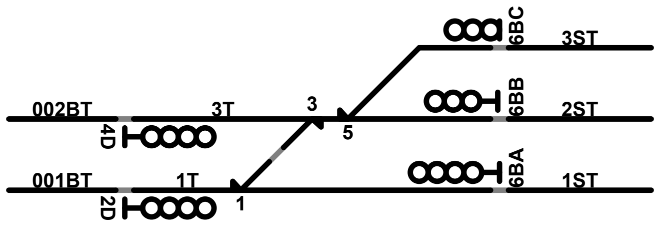


Fig. 1. A simple railway station.

2.1 Track Circuits

Track circuits give information about the occupation of the railway blocks. When a train enters to the railway block, the track circuit is short-circuited by the axles of the train and the track circuit gives that information to the interlocking system. Some railway blocks can contain more than one track circuit, especially if the railway block has a switch. Track circuits are denoted as 001BT, 002BT, 1T, 3T, 1ST, 2ST and 3ST in Fig. 1.

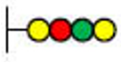
2.2 Switches

Switches are used to make the trains to change tracks in a railway station. Most of the switches have an entrance track and two exit tracks named *normal* and *reverse*. Switches are turned to required positions by the interlocking system when a route is requested. Switches can also be controlled manually by TCC if there is no route reservation on them. Switches are denoted as 1, 3 and 5 in Fig. 1.

2.3 Signal Lights

Signal lights give information about the next railway block to the train driver. In Turkish State Railways, there are mainly three kinds of signal lights: four aspect tall, three aspect tall and three aspect dwarf signal. Explanations of color combinations for these signal lights are given in Table 1. Signal lights are denoted as 2D, 4D, 6BA, 6BB and 6BC in Fig. 1.

Table 1. Definitions of the signal lights

Type	Colours and Definitions
Four aspect tall signal light 	*Red:* Stop, next block is occupied. *Yellow:* Next block is free, but the second block is occupied. Proceed carefully. *Green:* Next two blocks are free, train can proceed. *Yellow-Yellow:* There is a turning and next block is free, but the second block is occupied. *Yellow-Green:* There is a turning and next two blocks are free. Proceed with a predefined speed.
Three aspect tall signal light	*Red:* Stop, next block is occupied. *Yellow:* Next block is free, but the second block is occupied. Proceed carefully. *Green:* Next two blocks are free, train can proceed.
Three aspect dwarf signal light	*Red:* Stop, next block is occupied. *Yellow:* There is a turning and next block is free, but the second block is occupied. *Green:* There is a turning and next two blocks are free. Proceed with a predefined speed.

2.4 Routes

In a railway station, trains can proceed if there is a reserved route for them by TCC. Routes can be reserved between two track circuits with no switches. When a route request is send to the interlocking system, conflicting routes are checked first. If there is no reservation for a conflicting route, related switches are turned to the required positions. After that if there is no error for related components of the route, the route is reserved and the starting signal light of the route opens the related colours. There are 10 different routes such as 001BT – 1ST, 001BT – 2ST, 1ST – 001BT and 1ST – 002BT, in the railway station given in Fig. 1.

2.5 Interlocking System

Interlocking system is a vital component of a signalization system. It checks the incoming request from TCC and compares these requests with the actual situation of the components of the railway station. If all safety criteria are met the request is accepted, otherwise it is rejected. The interlocking system also compares incoming signals from the railway station and can detect component errors. For example if normal and reverse indication signals of a switch are coming to the interlocking system, an error in the switch is detected and all requests about the related switch is rejected.

2.6 Interlocking Table

Interlocking table shows all possible route reservations in a railway station with details. In an interlocking table, each route is shown with related track circuits, related switches and required switch positions and starting signal light with its colour combinations with respect to the end signal.

For a safe interlocking system design, first the interlocking table must be obtained correctly. If a track circuit or a switch is added to the station, the interlocking table can change completely. Therefore as the railway station is getting bigger, the generation of the interlocking table is getting harder.

3. AUTOMATIC INTERLOCKING TABLE GENERATION

Obtaining an interlocking table for a complex railway station takes too much time and needs extreme attention. If there is an error in the interlocking table, there will be errors in the design of the interlocking system that can result in fatal accidents. Therefore, in order to eliminate possible errors in the interlocking table, a computer algorithm can be used. In this study, an automatic interlocking table generator programme is realized in a symbolic algebra environment.

The programme has a simple user interface built on Microsoft .NET® framework as shown in Fig. 2. User can easily define a railway yard topology by drawing the track parts in the railway yard. Track circuits, switches, signal lights and routes are identified by the programme and for a feasible topology the interlocking table is automatically generated. In this section, basic principles of the automatic interlocking table generation are explained.

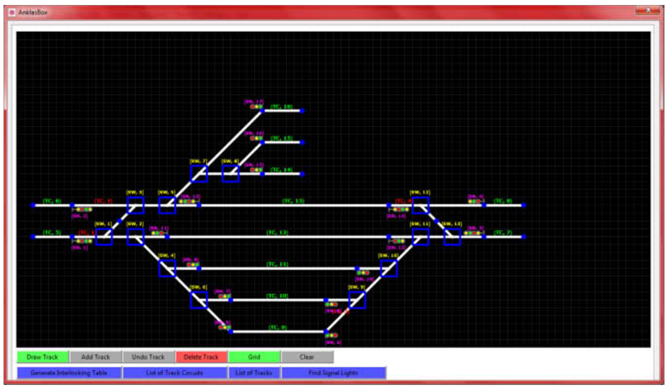


Fig. 2. User interface of the automatic interlocking table generator.

3.1 *Track Identification*

Each line drawn by the user is considered as a *railway track* by the programme. Starting and finishing points of the *track* are saved in a list named *PointList*. Every *track* (except the first track) must start with a point which already exists in the *PointList*. The *tracks* are saved in a separate list named *TrackList*. Track identification process is illustrated in Fig. 3.

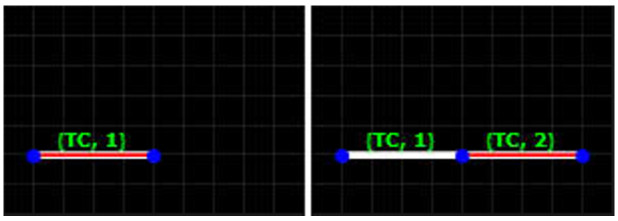


Fig. 3. Track identification.

A pseudo code for track identification is given as follows:

```
> Define point1 when the mouse is clicked.
> Define point2 when the mouse is released.
> Draw a line from point1 to point2.
> Define a "Track" between point1 and point2.
> Add track "to" the "Track List".
> Add points to the "Point List".
```

3.2 *Switch Identification*

A point can be a component of one, two or three tracks. If a point is a component of three tracks, then the point is defined as a switch point by the programme and it is not allowed to draw a new track starting from this point. The switch points are saved in a seperate list named *SwitchPointList*.

In order to define entrance and exit tracks of a switch, the tracks, which contain the switch point, are considered. By considering the other points of the tracks, *the entrance track* is determined as the track which is on the other side of the switch point with respect to the other two tracks on the x axis. The other two tracks are separated with respect to their slopes. The track, which has the same slope with the entrance track, is defined as *the normal exit track*. Then, the last track is defined as *the reverse exit track*. The process of switch identification is illustrated in Fig. 4.

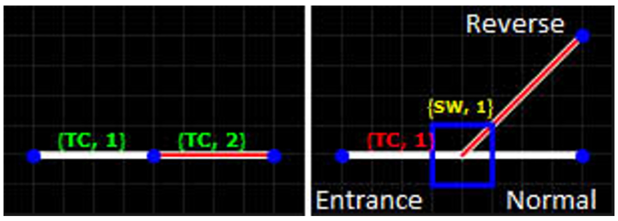


Fig. 4. Switch identification.

A pseudo code for switch identification is given as follows:

```
> Count the number of common points in the
"Point List".
> If there is a point repeated for 3 times,
determine this point as a "Switch Point".
> Draw a blue rectangle around this "Switch
Point".
> Add "Switch Point" to the "Switch Point
List".
> Find tracks of "Switch Point" from the
"Track List".
> Determine x axis coordinates of these
tracks to obtain the entrance track.
> Compare slopes of the other tracks with the
entrance track to obtain the normal and the
reverse exit tracks of this "Switch Point".
> Add switch to the "Switch List".
```

3.3 *Track Circuit Identification*

In a railway station, there can be two kinds of track circuits. These are *stationary track circuits* (STC) which contain only one track and *switch area track circuits* (SATC) which contain more than one track. STCs do not contain a switch point. Firstly, a track is chosen from the *TrackList* to identify track circuits in the railway station. Then, the points of the chosen track are examined to test if they are switch points. If none of points is a switch point, the track is defined as an STC. If the chosen track has at least one switch point, the other tracks, which have the same switch point, are found in the *TrackList* and all these tracks are defined as an SATC. Lastly, the track circuits are saved in a list named *TrackCircuitList*. The process of track circuit identification is illustrated in Fig. 5.

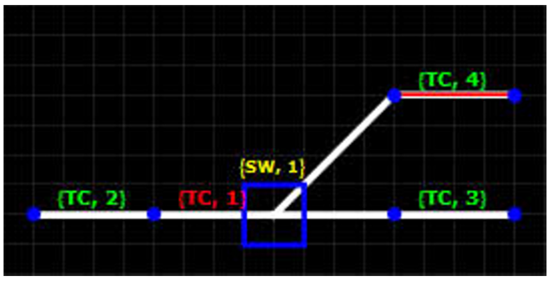


Fig. 5. Track circuit identification.

A pseudo code for track circuit identification is given as follows:

```
> Define a new empty list.
> Select tracks in order from the "Track
List" and add to new list.
> Examine the points of each selected track
in the new list if they are a "Switch Point"
or not.
> If track has a "Switch Point", check the
related tracks which contain common point to
the new list and go to previous step.
> Set this new list as "Track Circuit".
> If "Track Circuit" has only one track, set
"Track Circuit" as "Stationary Track
Circuit". Otherwise, set "Track Circuit" as
"Switch Area Track Circuit".
> Add "Track Circuit" to the "Track Circuit
List".
```

3.4 Route Identification

Routes are defined between two STCs. To identify routes, firstly an STC is chosen from the *TrackCircuitList*. Then, the points of the chosen track circuit are examined and the other track circuits, which are connected to the chosen track circuit, are found. If the connected track circuits contain a switch(ie one of the connected track circuits is an SATC), then other track circuits connected to this SATC are found. This procedure continues until all possible track circuit chains are found. Each track circuit chain (with required switch positions) is saved in a list named *RouteList*. Route identification is illustrated in Fig. 6.

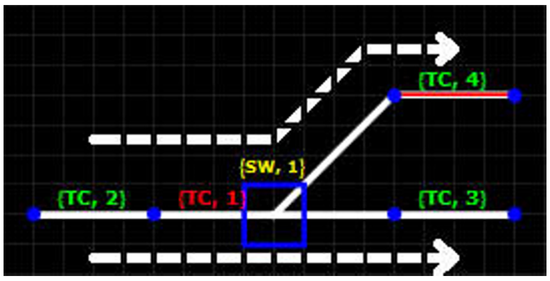


Fig. 6. Route identification.

Pseudo code about route identification is as follows:

```
> Choose a "Stationary Track Circuit" from
the "Track Circuit List".
> Find connected track circuits with the
chosen track circuit.
> If there is a "Switch Area Track Circuit"
in the connected track circuits, find track
circuits connected to this "Switch Area Track
Circuit".
> Generate all possible track circuit chains
which start with the chosen track circuit and
ends with another "Stationary Track Circuit"
and contains zero or more "Switch Area Track
Circuits".
> Define each possible track circuit chain as
a route.
> Find the required switch positions for the
route and add this information into the
route.
> Add the route to the "Route List".
```

3.5 Signal Identification

Signal lights are placed on the connection of station and switch track circuits. There are three types of signal lights as mentioned in section 2.3 generally used in Turkish Railways. Three aspect dwarf signals inform the driver that there is at least one reverse switch for all possible routes. Three aspect tall signals mean that there is no reverse switch for all possible routes. Four aspect signals, which are used generally on the entrance of railway stations, indicate reverse switches exist for some routes. They have one colour declarations (yellow or green) for straight routes and two colour declarations (yellow-yellow or yellow-green) for routes that have a turning (reverse switch). These criteria are used to determine the type of the signal light of a chosen STC. Defined signals are saved in a list named *SignalList*. Signal identification is illustrated in Fig. 7.

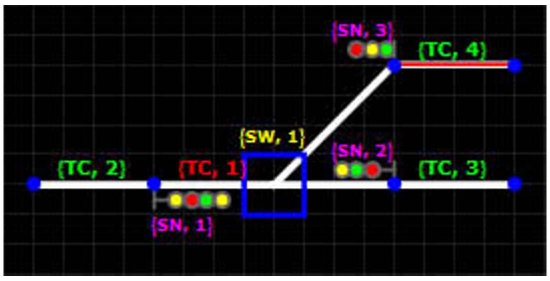


Fig. 7. Signal identification.

A pseudo code for signal identification is given as follows:

```
> Choose a "Stationary Track Circuit" from
the "Track Circuit List".
> Find all routes which start with the chosen
track circuit.
> Compare switch positions of all routes.
> If there is a switch which is in reverse
position for all routes, determine the signal
light as a three aspect dwarf signal.
> If there does not exist any switch in
reverse position for all routes, determine
the signal light as a three aspect tall
signal.
> Otherwise determine the signal light as a
four aspect tall signal.
> Add the signal to the "Signal List".
```

3.6 Generation of Interlocking Table

In the interlocking table all possible routes are shown in a detailed table. Each route is shown with its name, track circuits, switch positions, signal lights and starting signal light color combinations. After drawing the railway stations topology, the interlocking table can be automatically generated by pressing a button. All identifications are done automatically.

4. INTERLOCKING CODE DESIGN AND APPLICATION

The model railway station given on Fig. 8. is used to illustrate the working principles of the programme.

Fig. 8. Model railway station (Hardware simulator).

Firstly, the model railway station topology is drawn in the programme and the proper track circuits, switches, routes and signal lights are found automatically as shown in Fig. 9. After that, interlocking table of the model railway station is generated automatically. 34 different routes are found in the station topology. A part of the generated interlocking table is shown in Fig. 10. The first column of the table shows the route number, the second column shows track circuits, the third column shows normal position switches, the fourth column shows reverse position switches and the fifth column shows starting and finishing signals with colour combinations.

After obtaining the interlocking table, the interlocking code can easily be converted to Programmable Logic Controller (PLC) codes by using general function blocks as discussed in (Durmuş et. al., 2012). Each function block takes TCC requests as an input and has an output which contains interlocking system decisions. Some function blocks have also specific inputs for the connection between the other blocks.

For example, the route function block has inputs for related track circuits, switches and signals. In the generated interlocking table, Route 7 (RT7) has three switches (SW1, SW3 and SW5), four track circuits (TC8, TC1, TC2 and TC10), a starting signal (SN2) and a finishing signal (SN6). Usage of the route function block for RT7 is shown in Fig. 11. The signal function block has inputs for color combinations. In the interlocking table, it is seen that Signal 1 (SN1) must be yellow when RT2 is reserved and SN6 is red. Usage of the signal function block for Signal 1 (SN1) is illustrated in Fig. 12.

Lastly, the interlocking code is tested using an interlocking test programme (ITP). After all tests are passed with ITP, software and hardware simulators which are mentioned in (Mutlu et. al., 2012) are used to complete the validation of the system in the laboratory.

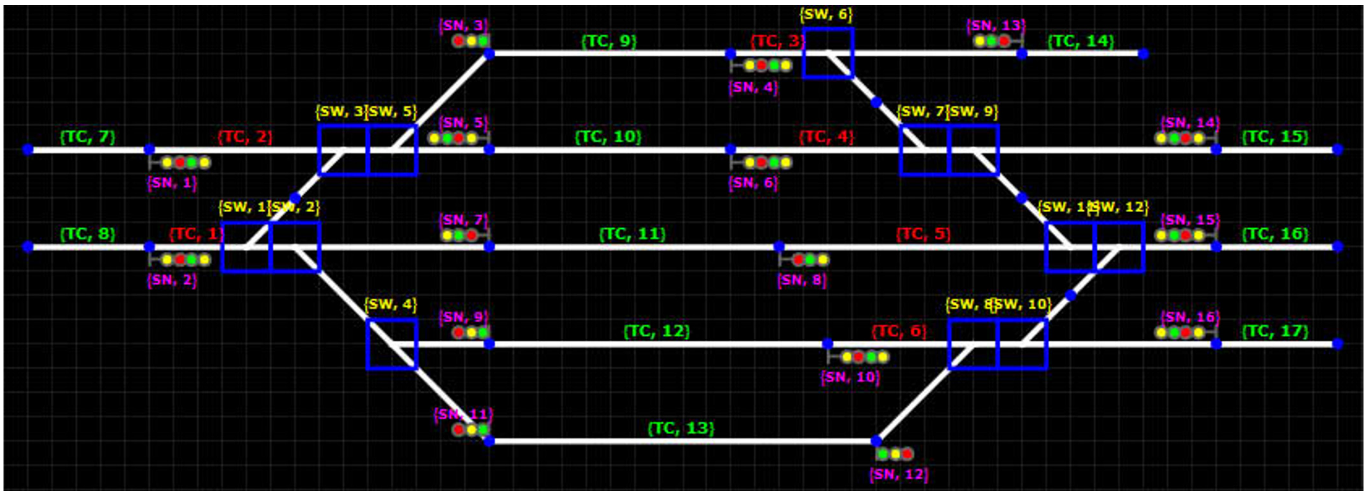
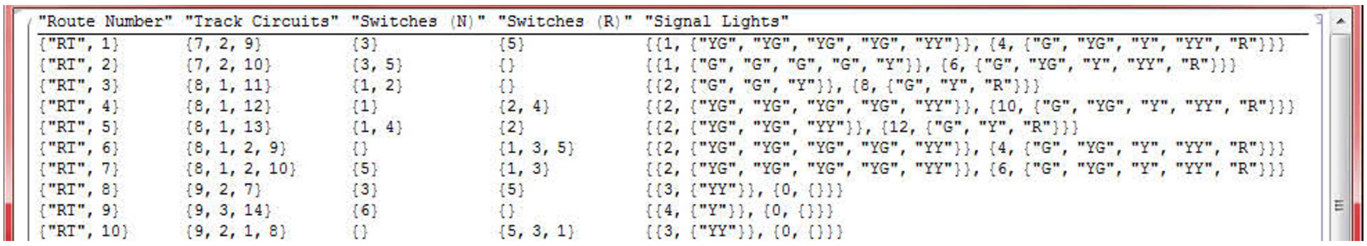


Fig. 9. Drawing of the model railway station.



"Route Number"	"Track Circuits"	"Switches (N)"	"Switches (R)"	"Signal Lights"
{"RT", 1}	{7, 2, 9}	{3}	{5}	{{1, {"YG", "YG", "YG", "YG", "YY"}}, {4, {"G", "YG", "Y", "YY", "R"}}}
{"RT", 2}	{7, 2, 10}	{3, 5}	{}	{{1, {"G", "G", "G", "G", "Y"}}, {6, {"G", "YG", "Y", "YY", "R"}}}
{"RT", 3}	{8, 1, 11}	{1, 2}	{}	{{2, {"G", "G", "Y"}}, {8, {"G", "Y", "R"}}}
{"RT", 4}	{8, 1, 12}	{1}	{2, 4}	{{2, {"YG", "YG", "YG", "YG", "YY"}}, {10, {"G", "YG", "Y", "YY", "R"}}}
{"RT", 5}	{8, 1, 13}	{1, 4}	{2}	{{2, {"YG", "YG", "YY"}}, {12, {"G", "Y", "R"}}}
{"RT", 6}	{8, 1, 2, 9}	{}	{1, 3, 5}	{{2, {"YG", "YG", "YG", "YG", "YY"}}, {4, {"G", "YG", "Y", "YY", "R"}}}
{"RT", 7}	{8, 1, 2, 10}	{5}	{1, 3}	{{2, {"YG", "YG", "YG", "YG", "YY"}}, {6, {"G", "YG", "Y", "YY", "R"}}}
{"RT", 8}	{9, 2, 7}	{3}	{5}	{{3, {"YY"}}, {0, {}}}
{"RT", 9}	{9, 3, 14}	{6}	{}	{{4, {"Y"}}, {0, {}}}
{"RT", 10}	{9, 2, 1, 8}	{}	{5, 3, 1}	{{3, {"YY"}}, {0, {}}}

Fig. 10. A part of generated interlocking table of the model railway station.

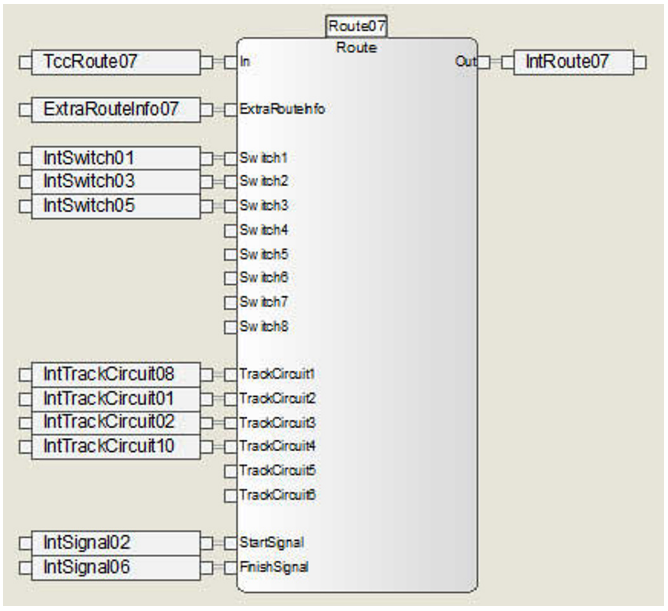


Fig. 11. Route function block.

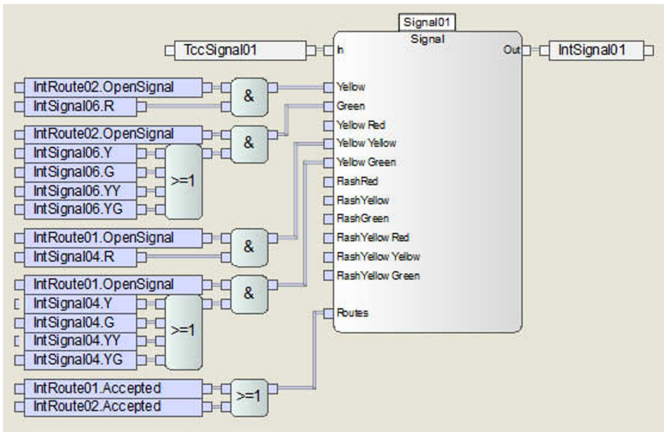


Fig. 12. Signal function block.

5. CONCLUSION

In this study, basic principles of an automatic interlocking table generator programme were explained. An interlocking table generated for a model railway station using the programme is given as an example. Laboratory tests show that the interlocking code that is written by the help of the generated interlocking table works correctly. The generation of the interlocking code can also be automatized in the future. Therefore, a safe interlocking code for large and complex stations can be obtained in a very short time.

ACKNOWLEDGMENT

This work is supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) project number 108G186 – The National Railway Signalization Project.

REFERENCES

Anunchai, S.V. (2010). Modelling Railway Interlocking Tables Using Coloured Petri Nets, *Lecture Notes in Computer Science*, pp. 137-151.

Durmuş, M. S., Yildirim, U., Eriş, O. and Söylemez, M.T. (2012). Safety-Critical Interlocking Software Development Process for Fixed-Block Signalization Systems, submitted to *The 13th IFAC Symposium on Control in Transportation Systems, CTS 2012*.

Fokkink, W.J. and Hollingshead, P.R. (1998). Verification of interlockings: From control tables to ladder logic diagrams, *Proceedings of 3rd Workshop on Formal Methods for Industrial Critical Systems*, pp. 171-185.

Haxthausen, A.E., Bliguet, M.L. and Kjaer, A. (2010). Modelling and Verification of Relay Interlocking Systems, *Lecture Notes on Computer Science*, pp. 141-153.

Kuzu, A., Songuler, Ö., Sonat, A., Turk, S., Birol, B., Doğrugüven, E.H. (2011). Automatic interlocking table generation from railway topology, *IEEE International Conference on Mechatronics, ICM'11*, Istanbul, Turkey, pp. 64-70.

Mirabadi, A. and Yazdi, M. B. (2009). Automatic Generation and Verification of Railway Interlocking Control Tables Using FSM and NUSMV, *Transportation Problems*. Vol 4, pp. 103-110.

Mutlu, İ., Ergenç, A.F., Ovatman, T. and Söylemez, M.T. (2012). Design of a Hardware and Software based Test Bed for Railway Signalization System, submitted to *The 13th IFAC Symposium on Control in Transportation Systems, CTS 2012*.

Petersen, J.L. (1998). Automatic verification of railway interlocking systems: a case study, *Proceedings of 2nd Workshop on Formal Methods in Software Practice*, pp. 1-6.

Roanes-Lozano, E., Hernando, A., Antonio Alonso, J., Laita, L.M. (2010). A logic approach to decision taking in a railway interlocking system using Maple, *Mathematics and Computers in Simulation*.

Tombs, D., Robinson, N., Nikandros, G. (2002). Signalling Control Table Generations and Verification, *Proceedings of Conference on Railway Engineering*.

Winter, K. (2002). Model Checking Railway Interlocking Systems, *Proceedings of the Australian Computer Science Communications*, 24.

Winter, K., Robinson, N. (2003). Modelling large railway interlockings and model checking small ones, *Proceedings of the 26th Australian Computer Science Conference*, pp. 309–316.