Proceedings of the 2011 IEEE
International Conference on Mechatronics
April 13-15, 2011, Istanbul, Turkey

Automatic Interlocking Table Generation from Railway Topology

Ahmet Kuzu[#1] , Ozgur Songuler[#2], Arcan Sonat[#3], Serhat Turk[#4], Berkin Birol[#5], Ersin H. Dogruguven[#6]

TUBITAK-BILTEN-BTE, Kocaeli, 41470, TURKEY
Istanbul Technical University, Istanbul, 34469, TURKEY

[1]ahmet.kuzu@bte.tubitak.gov.tr
[2]ozgur.songuler@bte.tubitak.gov.tr
[3]arcan.sonat@bte.tubitak.gov.tr
[4]serhat.turk@bte.tubitak.gov.tr
[5]berkin.birol@bte.tubitak.gov.tr
[6]ersin.dogruguven@bte.tubitak.gov.tr

Abstract— **In this paper, an algorithm for automatic interlocking table generation and its implementation is introduced. The specially developed software generates an interlocking table conforming to Turkish Railway regulations. Pseudo codes of the method and their descriptions are given. The method is demonstrated on a small station area, and sample outputs are presented.**

***Keywords*— railway, interlocking, signal, point, a**

I. Introduction

An Interlocking Table is a tabular representation of the rules and pre-conditions governing route allocation based on all vital interrelationships between different signaling equipments such as signals, points and track circuits. The Interlocking Table has an important role in the design of an interlocking system since it acts as an agreement between the railway administrators and the contractors and defines the prerequisites of safe interlocking [1, 2].

In a complicated railway signalization project, developing several interlocking tables manually for each station is a very labor intensive task and thus prone to errors. Automatic generation of interlocking tables can reduce both the time spent and the number of errors. In this paper, an easy and very effective approach is presented for this purpose. This method does not only provide a rapid way to develop a interlocking table but also a safer way.

There are a limited number of works in the literature pertaining to automatic generation of interlocking tables [1,3]. However, in these works while the points and routes to be locked are considered, the signal outputs (aspects) are not. Thus these generated interlocking tables do not include the signal aspect of start signals of the routes.

II. Interlocking Tables

The format and content of interlocking tables is not standardized, and may vary even within the same railway administration [1]. For this reason, brief information about route 1BT-AT in the first row of the interlocking table in

TABLE I
EXAMPLE INTERLOCKING TABLE

Route No	Route	Start Signal		Aspect		Lock		Track Control	Route Control
rt1	1BT-AT	2D	A	G	52D Y	1	2B 52BA 54BA	1T, AT	(1T)
				Y	52D R, RY				
				RY				1T, ~AT	
rt2	1BT-BT		B	GY	54D Y	~1, ~3	4B 52BB 54BB	1T, BT	(1T)
				YY	54D R, RY				
				RY				1T, ~BT	
rt3	1BT-(C)		3S	RY		~1, 3	52BC 54BC	1T	(1T)

978-1-61284-985-0/11/$26.00 ©2011 IEEE

Table 1 is given. This route is named 1BT-AT for the reason that it starts from signal 2D and terminates at the signal 52D. 2D indicates the number of the route signal. In the event that the permission to allocate route 1BT-AT is given, the colors that signal 2D must indicate based conditionally on the following signal 52D in the same direction is represented in the "Signal Aspect" column.

Point number 1 existing in the "Lock Column" infers point 1 is to be locked in the normal position. A tilde before the point numbers indicates that the related points should be locked in the inverse position. Signal 2B listed in the same column is required to be locked owing to the fact that it depicts the opposite direction. We define a locked signal as a signal in the red color indication state. 52BA denotes that the route starting from signal 52B and terminating at track circuit AT should be locked. Similarly 54BA that the route starting from signal 54B and ending at track circuit AT should be locked. Thus, collisions between railroad vehicles moving along the route 1BT-AT and another railroad vehicle coming from the opposite direction can be prevented. 1T and AT listed in the signal control and detector lock column represents the track circuits which must be controlled after the route allocation request. Completion of route allocation and green or yellow indication of signal 2D depends on the availability of track circuits 1T and AT. 1T and tilde AT below the same column implies that 1T is available and AT is occupied. In this situation, signal 2D indicates yellow over red. This case is only valid for station track circuits. Other track circuits except the station track circuits must definitely be available for route allocation.

Route lock 1T is used for rejecting route allocation requests for routes which include the same points as route 1BT-AT, such as route AT-1BT. When the train completely enters the last track circuit of the route, route locks and the allocation of the route is released. For route 1BT-AT, after the train abandons 1T and enters AT, the allocation of 1BT-AT and 1T route lock is released.

III. Railway Modeling

In the proposed system a class named railway element is used to model a railway system. The whole system is a memory tree consisting of railway elements. The railway element class includes two pointers called *West Port* and *East Port*, pointing to other railway elements on its west and east sides respectively. This class has a property named direction that can take on the values of *WestToEast*, *EastToWest* or *Both*. The other properties of the class are field name, instance name and type. Field name takes on the same value as the name of the physical element in the railway signalization layout and the instance name is the pointer of the class. Lastly, the type property of the railway element class takes on the values of *track circuit*, *signal*, *point* or *field port*. In this paper, this class is represented with an icon similar to a circuit element shown in Figure 1. The port on the left side represents the west port, the port on the right side represents the east port, and the arrow inside the icon represents direction. The text in quotations located above the figure is the name given in the railway signalization layout and the other text located below the figure is the instance name. All of the classes introduced below, are inherited from this class.

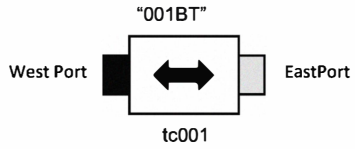

Fig 1. Icon of Railway Element

The second class used is Field Port Class. This class represents the end point of the field. When Field Port is located at the West end, the west port property of the class is a null pointer and the east port points to the following railway element. The value of the direction property should be west to east. A Field Port located at the East end may be similarly realized. In a field there can be more than one east and one west field port. This class will be represented with the icon shown in Figure 2 in the following parts.

Fig 2. Icons of Field Ports

The third class used is the Track Class. This class has an additional property called *isIndicationExist* representing the indication of track occupancy. This class will be shown by the same icon used for railway elements shown in Figure 1 in the following parts, but the value of the direction property is always set to *Both*.

The fourth class used is the Point Class. The point class has two additional properties named *Normal Port* and *Reverse Port*. Normal Port points to the address of the railway element connected to the normal position exit of the point. Reverse Port is similarly defined for the reverse position. When the direction of the point is west to east, east port is ignored and either normal port or reverse port are used as east port. For the east to west direction, west port is ignored and either normal or reverse port is used as west port. In order to express this situation better in the figures, the icon of the point is different from the ordinary railway element, and shown in Figure 3.

Fig 3. Icons of Points

The last set of classes used in the model is signal classes. These classes are *Dwarf Signal Class*, *3 Light High Signal Class*, *4 Light High Signal*, and *Virtual Signal Class*. Except for the last one, these classes are identical to the railway element class. The direction value of these classes cannot assume a "Both" value; it can be either *WestToEast* or *EastToWest*. Thus the signals and track circuits in the figures can be identified by their direction. The last class, Virtual Signal, is a special class among the signal classes because it is used where there should be a signal but in the physical field area is not because of cost reduction.

Added to this situation in some cases there can be another sign informing the locomotive driver to take care the near signal indications. Virtual Signal has the property called *isPointedAnySignal* to realize this case. When *isPointedAnySignal* is true the virtual signal points the near signal in its field name. When it is false the virtual signal behaves just like an ordinary signal. All signals classes except Virtual Signal class has the same icon with railway element while the virtual class has similar icon but with a border drawn dashed. Both icons are shown in Figure 4.

Fig 4. Icons of Signals

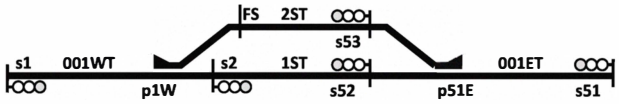

Fig 5. Simple Field for Describing Railway Memory Mapping

In order to express the usage of these classes, a simple field is shown in Figure 5. This field includes four track circuits, 001WT, 1ST, 2ST, 001ET, five signals s1, s2, s51, s52, s53, two points 1W, 51E, and a sign board pointing s2. First, we should create the classes of field ports. Field has two ends, one in west and one in east, so we can create them with the names wp01, ep01. Then we should create five ordinary signs and one virtual sign whose field names should be same with s2. Both track circuits 001WT and 001ET are divided into three parts by points p1W and p51E. We should create all the divided parts separately but give them the same field name with the parent part. In this approach we will have six tracks created from two parent tracks. After creating all track circuits named as tc001WT_1, tc001WT_2, tc001WT_3, tc1ST, tc2ST, tc001ET_1, tc001ET_2, tc001ET_3, we should point their pointers west port, east port, normal port, reverse port assigned properly. Memory view of the field is shown in Figure 5. Lines mean pointer assignments. For instance s1's east port is connected to tc1's west port. This means s1'east port's value is assigned to tc001WT_1 and tc001WT_1's west port's value is assigned to s1.

IV. Interlocking Table Generation

There is no general standard defining the format and the contents of an interlocking table. Different railway administrations have different formats and the tables may vary even for two stations within the same railway administration because of the specific road conditions. Nevertheless, general principles of control table design are alike. Some of the main control methods used by the railway companies for designing the interlocking tables are listed below:

-Route locking: Route setting involves a collection of adjacent track circuits, points and signals. To assure the safety, firstly, the interlocking system verifies that the route does not conflict with other routes previously set. The column "Requires Route Normal" shows conflict routes. A route cannot be set if any conflict routes have been set and not yet released. [4,5]

-Approach locking: After a route is set; the points are locked; and the entry signal is cleared, if the track circuit in front of the entry signal is occupied, then the signalman cannot cancel the route and the entry signal by the normal procedure. Approach locking prevents the train driver from the sudden change of signal aspect from green or yellow to red. [4,5]

-Flank protection: The equipments within the surrounding area of the reserved route that may cause an accident shall be protected even if no train is expected to pass such a signal or such points. For example points

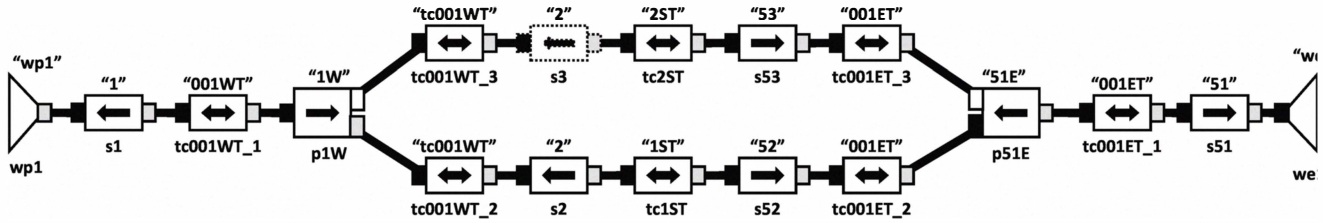

Fig 6. Memory Mapping of the Field in Figure 5

be in such positions that they do not give immediate access to the route. [4,5] In the proposed paper we used all three methods in order to increase the safety of the system.

After getting field topology from memory, algorithm runs five different procedures shown in Figure 5 to build up the interlocking table. These are *Find Route Procedure*, *Get Primary IT Procedure*, *Reduce IT Procedure*, *Add Conflicted Routes Procedure*, and *Add Signal Controls Procedure*.

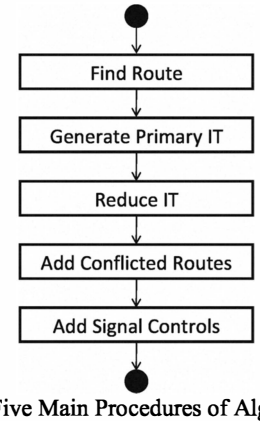

Fig 7. Five Main Procedures of Algorithm.

Find Route Procedure is the one that finds all possible routes in the field. This procedure saves the route in a list called *routeList*, and then generates another list called *hugeList* consists of *routeLists*. Every element of *hugeList* represents a separated route list, and every route list's element is a pointer to the railway element class. A route is defined as the distance between two same direction signals. [3] Thus, the procedure runs *scan* sub procedure taking the signal, new *routeList* and direction of the signal as parameters. Procedure's pseudo code is given below.

```
findRoute()
for each signal
   scan(signal, new routeList, signal->direction)
end for
```

The *scan* sub-procedure is a recursive function which calls itself with next item in the topology regarding direction. There are just two exceptions. One is stop condition for scanning. When the parameter element is another signal that has the same direction with the first signal, sub-procedure doesn't call itself and returns. The other exception happens when the element is a point whose direction equals to scanning direction. In this condition, the algorithm duplicates the parameter *routeList*, adds new one to the *hugeList*, and calls two scan sub-procedures, branching to point's both normal and reverse positions. After all recursive scan sub-procedures return, all routes starting with the first signal called in *findRoute* procedure, are identified.

```
Scan(element, roueList, direction)
if roueList empty
   hugeList-> Insert(list);
end if
roueList ->insert(element);
if roueList ->count()>1
  if element=signal and direction=element->direction
     return;
  end if
end if
if element->type=point and direction=element-> direction
   listTemp=new list->clone();
   hugeList-> Insert(listTemp);
   scan(element->normalPort, roueList,direction);
   scan(element->reversePort, listTemp,direction);
else
   if direction=WestToEast
      scan(element->eastPort,list,direction);
   else
     scan(element->westPort,list,direction);
   end if
end if
```

The second procedure is named as *Generate Primary IT (Interlocking Table) Procedure*. Algorithm starts to build up the interlocking table with the data acquired from parsing a route list in the *hugeList*. First and last elements of a *routeList* are the start and the exit signals of that particular route respectively. The second element of the *routeList* gives the first track and the second to last element gives the last track. Parsing process continues with generating some small lists called *trackList*, *signalList*, *normalPointlList* and *reversePointList*. These lists include the other necessary information for the interlocking table such as tracks, signals with opposite direction, points in normal position, and points in reverse position located on the route.

```
generatePrimaryIT()
for each list in hugeList
   row= new interlockingTableRow;
   row->firstSignal=list->firstElement
   row-> lastSignal=list->lastElement
   row-> firstTrackCircuit=list->firstElement-> Next()
   row->lastTrackCircuit=list->lastElemen-> Previous()
   for each element in list
     case element->type
       trackCircuit:
         row->trackList->insert(element)
       signal:
         row->signalList->insert(element)
       point:
          if (element->normalPort=list->Previou()
              or if element->normalPort=list-> Next())
                 row->normalPointList->insert(element)
```

```
        else
              row->reversePointList->insert(element)
        end if;
      end case
   end for
   interlockingTable->insert(row)
end for
```

In the identification of the routes phase it is possible to have more than one route with the same start and exit signals but different point combinations. In this case, the shortest route should be selected so it is suitable to use a cost function computing the number of the points in the route. *reduceIT* function scans all routes and if that condition happens it deletes the route including the more points. The pseudo code of this procedure is written below.

reduceIT()

```
for row1 from interlockingTable firstRow to lastRow
  for row2 from interlockingTable row1->Next() to
lastRow
    if (row1->firstSignal= row2->firstSignal
        and row1->lastSignal= row2->lastSignal)
      if(row1->normalPointList->count()
         +row1-> reversePointList->count())
        >(row2->normalPointList->count()
          +row2->reversePointList->count())
        interlockingTable->delete(row1)
      else
        interlockingTable->delete(row2)
      end if
    end if
   end for
end for
```

Add Conflicted Routes is the procedure that finds the routes that cannot be set altogether at the same time. This information uses not just the owner row data but also the other rows' data. In this procedure we find the rows having the same last signal and determine these rows as conflicted routes. These rows are written in the conflicted routes column in the interlocking table. Its pseudo code is shown below.

addConflictedRoutes()

```
for row1 from interlockingTable firstRow to lastRow
  for row2 from interlockingTable row1->Next() to
lastRow
    if row1->lastSignal= row2->lastSignal
      row1->conflictedRouteList->insert(row2)
      row2->conflictedRouteList->insert(row1)
    end
   end for
end for
```

The last procedure is *Add Signal Controls* procedure. This procedure is the action column of the IT, where others are preventing columns. This function describes what the start signal light will be according to the exit signal light. For instance, in which condition green light should be driven. These conditions are differed with government regulations. In this paper, because of paper length problems we are able to write only one example yellow light conditions. Other conditions are similar.

addSignalControls()

```
for each row  in interlockingTable
  greenCondition(row)
  yellowCondition(row)
  yellowOverYellowCondition(row)
  greenOverYellowCondition(row)
  redOverYellowCondition(row)
  flashGreenCondition(row)
  flashYellowCondition(row)
  flashRedCondition(row)
end for
```

All start signal conditions depends on the exit signal indication. For example in Turkish regulations, it says in an item that "If all track circuits in the route have indication, and train does not turn among the route then the start signal should show yellow". Moreover in another item it is said that "If all track circuits of the route have indication and the start signal is dwarf signal, and train turns out at least once then the start signal should show yellow". Combining these two items the yellow condition is written as follows. Other conditions are defined in a similar manner

yellowCondition (row)

```
if every row->TrackCircuits-> IsIndicationExist=true
   if row->ReversePointList->Count()>0
     if row1-> firstSignal->type=dwarfSignal
      row->condition.Y="YY, GY, RY, R"
     end if
   else
    row->condition.Y="YY, GY, RY, R"
   end if
end if
```

V. Case Study

In this section a model of a field which is published in a peer reviewed thesis [7] is used. 10 signals, 9 track circuits and 5 points comprise the section of line in this study. The signalization layout of this field is shown in Figure 8.

As mentioned before; the format and content of interlocking tables are not standardized, and may vary even within the same railway administration [1]. For this reason, brief information about the route 1BT-2ST in the first row

of the interlocking table is given. This route is named 1BT-2ST for the reason that it starts from signal 2D located in front of the track 1BT and terminates at the signal 52DA located in front of the track 2ST. In the event that the permission to allocate route 1BT-2ST is given, the colors that start signal 2D must indicate based conditionally on the following signal 52DA in the same direction is represented in the "Signal Aspect" column. Point number 1 existing in the "Lock" column indicates point 1 is to be locked in the normal position. ~ Characters left to the point numbers indicate that the related points should be locked in the reverse position. Signal 2BA listed in the same column is required to be locked owing to the fact that it depicts the opposite direction. We define a locked signal as a signal in the red color indication state. 52B2 denotes that the route starting from signal 52B and terminating at track circuit 1ST should be locked. Similarly 54B2 states that the route starting from signal 54B and ending at track circuit 2ST should be locked. Thus, collisions between railroad vehicles moving along the route 1BT-2ST and another railroad vehicle coming from the opposite direction can be prevented.

3T and 2ST listed in the signal control and detector lock column represents the track circuits which must be controlled after the route allocation request. Completion of route allocation and green or yellow indication of signal 2D depends on the availability of track circuits 3T and 2ST. 3T and circled 2ST below the same column implies that 3T is available and 2ST is occupied. In this situation, signal 2D indicates yellow over red. This case is only valid for station track circuits. Other track circuits except the station track circuits must definitely be available for route allocation. Route lock 3T is used for rejecting route allocation requests for routes which include the same points as route 1BT-2ST, such as route 2ST-1BT. When the train completely enters the last track circuit of the route, route locks and the allocation of the route is released. For route 1BT-2ST, after the train abandons 3T and enters 2ST, the allocation of 1BT-2ST and 3T route lock is released.

Then developed software is run with this field topology. Generated interlocking table is shown in Table 2. It is completely same within the thesis [6]. More significantly, it is also run with the topology of a real railway station included in the National Railway Signalization Project of Turkey. With no surprise, the automatic interlocking table generation software gives the same results with the interlocking table generated manually and approved by the railway administration authorities. However due to the limited paper size, it is not possible to give a detailed representation of the station.

VI. Conclusion

In this paper, a software and algorithm capable of translating a railway topology into an interlocking table was described. A sample track layout input was used to demonstrate the method.

While there are examples of works [1],[3] which present methods of generating interlocking tables, we were not able to find examples in the literature which implement the signal light control output conditions of the interlocking table. In the course of this work a novel railway topology to interlocking table generator was designed and implemented.

Acknowledgment

This work has been carried out as part of the National Railway Signalization Project of Turkey project nos. TUBITAK 108G025 and ITU 108G186. The authors wish to thank TUBITAK UEKAE and Istanbul Technical University for their support.

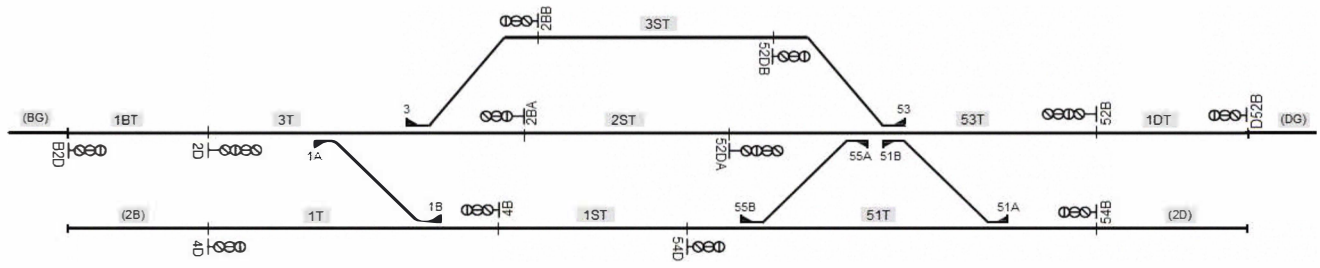

Fig 8. Signalization Layout

REFERENCES

[1] D. Tombs, N. Robinson, and G. Nikandros, "Signalling Control Table Generation and Verification", *Conference on Railway Engineering Wollongong*, pp: 10-13, 2002.

[2] C. Chevillat, D. Carrington, P. Strooper, J. G. Süß, and L. Wildman, "Model-Based Generation of Interlocking Controller Software from Control Tables. Berlin: Springer-Verlag, 2008, pp. 349-360

[3] A. Mirabadi., M.B.Yazdi, "Automatic generation and verification of railway interlocking control tables using FSM and NUSMV," *Transport Problems : an International Scientific Journal*, vol. 4, pp: 103-110, 2009

[4] S. Vanit-Anunchai, "Verification of Railway Interlocking Tables Using Coloured Petri Nets", Tenth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, Oct. 2009

[5] W. Fokkink, P. Hoolingshead, "Verification of Interlockings: from Control Tables to Ladder Logic Diagrams", *Proc. 3rd Workshop on Formal Methods for Industrial Critical Systems (FMICS'98)*, pp: 171-185, 1998

[6] Arcan Sonat, "Raylı Ulasim Sistemlerinde Anklasman Algoritmasi Tasarimi ve Otomat Yaklasimi ile Otomatik Kod Uretme", MSc Thesis, Istanbul Technical University, Istanbul, 2010

TABLE II

OUTPUT INTERLOCKING TABLE

Route No	Route	Start Signal	Aspect		Lock		Track Control	Route Lock
rt1	1BT – 1ST	2D	GY YY RY	54D Y 54D R, RY	~1	4B 52B1 54B1	3T 1T 1ST 3T 1T ~1ST	(3T) (1T)
rt2	1BT – 2ST		G Y RY	52DA Y 52DA R, RY	1 3	2BA 52B2 54B2	3T 2ST 3T ~2ST	(3T)
rt3	1BT – 3ST		GY YY RY	52DB Y 52DB R, RY	1 ~3	2BB 52B3 54B3	3T 3ST 3T ~3ST	(3T)
rt4	(2B) – 1ST	4D	G Y RY	54D Y 54D R, RY	1	4B 52B1 54B1	1T 1ST 1T ~1ST	(1T)
rt5	1ST – 1BT	4B	Y		~1	2D B2D	1T 3T 1BT	(1T) (3T)
rt6	1ST – (2B)		RY		1	4D	1T	(1T)
rt7	2ST – 1BT	2BA	Y		3 1	2D B2D	3T 1BT	(3T)
rt8	3ST – 1BT	2BB	Y		~3 1	2D B2D	3T 1BT	(3T)
rt9	1ST – 1DT	54D	Y		~55 51 53	52B D52B	51T 53T 1DT	(51T) (53T)
rt10	1ST – (2D)		RY		55 51	54B	51T	(51T)
rt11	2ST – 1DT	52DA	Y		55 51 53	52B D52B	53T 1DT	(53T)
rt12	2ST – (2D)		RY		55 51 53	54B	53T 51T	(53T) (51T)
rt13	3ST – 1DT	52DB	Y		~53 51 55	52B D52B	53T 1DT	(53T)
rt14	3ST – (2D)		RY		53 51	54B	53T 51T	(53T) (51T)
rt15	1DT – 1ST	52B	GY YY RY	4B Y 4B R, RY	53 51 ~55	54D 2D1 4D1	53T 51T 1ST ~1ST	(53T) (51T)
rt16	1DT – 2ST		G Y RY	2BA Y 2BA R	53 51 55	52DA 2D2	53T 2ST 53T ~2ST	(53T)
rt17	1DT – 3ST		GY YY RY	2BB Y 2BB R	~53 51 55	52DB 2D3	53T 3ST 53T ~3ST	(53T)
rt18	(2D) – 1ST	54B	G Y RY	4B Y 4B R, RY	51 55	54D 2D1 4D1	51T 1ST 51T ~1ST	(51T)
rt19	(2D)– 2ST		G Y RY	2BA Y 2BA R	~51 53 55	52DA 2D2	51T 53T 2ST ~2ST	(51T) (53T)
rt20	(2D)– 3ST		G Y RY	2BB Y 2BB R	~51 ~53	52DB 2D3	51T 53T 3ST ~3ST	(51T) (53T)
rt21	(BG)-1BT	B2D	G Y	2D G YY, RY, R		2BB 2BA 4B1B	1BT	
rt22	(DG)-1DT	D52B	G Y	52B G YY, RY, R		52DB1D 52DA1D 54D1D	1DT	