A Software Prototype for Isolated Ramp-Metering

Taale, Henk

Publication date
1991

Document Version
Final published version

Citation (APA)
Taale, H. (1991). *A Software Prototype for Isolated Ramp-Metering*. Rijkswaterstaat Dienst Verkeerskunde.

DRIVE PROJECT (V 1035)

CHRISTIANE

(MOTORWAY TRAFFIC FLOW MONITORING AND CONTROL)

to be included in DELIVERABLE number 7b

A software prototype for isolated ramp-metering

CXR92014.rap

- 1 July 1991 -

DRIVE PROJECT (V 1035)

CHRISTIANE

(MOTORWAY TRAFFIC FLOW MONITORING AND CONTROL)

to be included in DELIVERABLE number 7b

A software prototype for isolated ramp-metering

CXR92014.rap

- 1 July 1991 -

Project Reference Number:	V 1035
Project Acronym:	**CHRISTIANE**
Project Title:	MOTORWAY TRAFFIC FLOW MONITORING AND CONTROL
Prime Contractor:	Institut National de Recherche sur les Transports et leur Sécurité (INRETS, France)
List of Partners:	Scetauroute (France) Technical University of München (Germany) Transport and Road Research Laboratory (United Kingdom) Wootton Jeffrey's Consultants (United Kingdom) University of Thessaloniki (Greece) Rijkswaterstaat (The Netherlands)
Subcontractors:	CERT/DERA/SERALP (France)
Document status:	public report
Prepared by:	Dutch Ministry of Transport and Public Works Transportation and Traffic Research Division P.O.box 200 3000 BA Rotterdam tel: +31-10-4026610 fax: +31-10-4148115
Authors:	ir. Henk Taale ir. Frans Middelham
Date:	1 July 1991

CONTENTS

1. INTRODUCTION 2
   1.1. DRIVE-context 1
   1.2. Project-context 1
   1.3. Summary 2

2. FLEXSYT 3
   2.1. Introduction 3
   2.2. Design requirements 3
   2.3. MANAGER concept 3
   2.4. USER concept 4
   2.5. Traffic control language (FLEXCOL-76-) 4
      2.5.1. Elements of FLEXCOL-76- 4
         2.5.1.1. Names of elements 4
         2.5.1.2. Indices of elements 5
      2.5.2. Events in FLEXCOL-76- 5
      2.5.3. Conditions in FLEXCOL-76- 5
      2.5.4. Functions in FLEXCOL-76- 6
      2.5.5. Collections in FLEXCOL-76- 6
      2.5.6. Behaviour of elements in FLEXCOL-76- 7
      2.5.7. Formula concept of FLEXCOL-76- 8
         2.5.7.1. Event assignment statement 8
         2.5.7.2. Boolean assignment statement 8
   2.6. Structure of the program FLEXSYT-I- 9
      2.6.1. Input data sets of FLEXSYT-I- 10
      2.6.2. Output data sets of FLEXSYT-I- 11
   2.7. Comparison of FLEXSYT-I- with real life 11
   2.8. Development of FLEXSYT-II- 11

3. PROTOTYPE 13
   3.1. Introduction 13
   3.2. MANDAT dataset 13
   3.3. CONDAT dataset 15
      3.3.1. Initiation part 15
      3.3.2. Counting and measuring part 17
      3.3.3. Signal handling part 20
   3.4. Conclusion 21

APPENDICES 23

A. Examples

   A.1. MANDAT dataset 24
   A.2. CONDAT dataset 25
   A.3. NETDAT dataset 26
   A.4. REGFIL dataset 28
   A.5. RESFIL dataset 29

B. Prototype input datasets 32
B.1. MANDAT dataset 32
B.2. CONDAT dataset 34

1. INTRODUCTION

1.1. DRIVE-context

One of the projects of the European programme DRIVE is concerned with the problem of "Motorway Traffic Flow Monitoring and Control" and is named *Christiane* (Ref.nr. V1035). Prime Contractor of the project is INRETS (F); the Partners are Scetauroute (F), Technical University Munich (FRG), TRRL (UK), Wootton Jeffrey's Consultants (UK), University of Thessaloniki (Greece) and Rijkswaterstaat (NL). Subcontractors are CERT, DERA and SERALP (F).

The general objective of DRIVE is threefold: improve road transport efficiency, improve road transport safety and reduce environmental pollution. Motorway traffic monitoring and control are related to all of these objectives. Motorway control aims at optimal utilisation of the capacity of motorway networks, for which motorway monitoring is a necessary prerequisite. With respect to control strategies the DRIVE objective is "to establish a common framework in which different system control methods can be accommodated without losing compatibility". (Annual Project Review Report of Christiane, 27-09-89).

1.2. Project-context

The type of motorway control considered in the Christiane project are control by Variable Message Signs (VMS) and by ramp-metering. The project is divided in two parallel developments: traffic flow modelling and traffic flow control. The models may be used for the monitoring purpose and also for designing or testing control strategies. In the control direction there are two approaches: aid-to-decision and automatic control.

The first step in the project has been a review of existing models and control strategies. The result of this review are reported in the project Deliverable no. 1 of Work Package 1 (April 1989). With respect to subject of the ramp-metering the review has recognised the following approaches:

- ALINEA, developed by the Technical University Munich, and tested by INRETS on the Boulevard Périphérique;
- a strategy developed by Wootton Jeffrey's, applied on several (6) ramps of the M6 in the UK;
- the strategies applied by Rijkswaterstaat at the Coentunnel and near Delft in The Netherlands;
- the theoretically developed strategy of the University of Thessaloniki.

A detailed comparison of the several strategies and a discussion on some main differences (like vehicle by vehicle vs. platoon metering, feedback vs. feedforward control etc.) may be found in the project Deliverable 5 of Work Package 3 called "Isolated Ramp-Metering" (April 1990).

1.3. Summary

It is the aim of the ramp-metering part of the project to use the existing expertise to identify the applicability of existing approaches. Apart from a desk-top comparison like the deliverable just cited, two **field trials** were planned:

- a field trial with ALINEA at the Coentunnel in Amsterdam, The Netherlands;
- a field trial with the Wootton Jeffrey's strategy on the BP in Paris, France.

These trials should lead to insight in the range of conditions over which the approaches are viable, the limits of the applicability of isolated control and possibly lead to control improvements. The results of the trial in The Netherlands are reported in Deliverable 7a: *Isolated Ramp-metering: Real life Study in the Netherlands*, march 1991.
The main conclusion of this report states that: *The ALINEA algorithm for ramp-metering produces results comparable to or better than those of the RWS algorithm. ALINEA increases the total service of the system, the motorway and the ramp at the Coentunnel. When the homogenisation of the traffic flow in the bottleneck is one of the aims of ramp-metering, the fact that ALINEA gives a 10 % greater deviation in the speeds at the bottleneck is of importance.*
For further research it is necessary to have a implementation of the RWS and ALINEA strategy in a software prototype. The theoretical background of both strategies is discussed in Deliverable 7a.

This document is a continuation of Deliverable 7a and discusses the implementation of the Rijkswaterstaat and ALINEA strategy in a software prototype. The prototype is developed in the computer program FLEXSYT. Therefore the program FLEXSYT is discussed first and then the software prototype of a ramp-metering installation is developed.

2. FLEXSYT

2.1. Introduction

FLEXible-traffic-network-**S**imulation-stud**Y**-**T**ool was developed in the seventies en eighties by Frans Middelham. It is a computer program for traffic management studies and simulates traffic on a microscopic scale. On a stochastic base vehicles move through the network and travel times are calculated. In this way it is possible to do research on the structure of the network, such as the lay-out of intersections, length and number of lanes, effects of bus lanes, etc., and on traffic-control alternatives, like the fixed-time control strategy, vehicle-actuated control strategies, traffic-depended control strategies, etc.
In this chapter the basic principles and the structure of FLEXSYT are discussed.

2.2. Design requirements

The design requirements for a traffic-control-language are of the same importance as those for a common computer language. When starting the work on FLEXSYT, the following design requirements had been formulated:

- well defined to be unambiguous;
- user friendly;
- no built-in control philosophy;
- independent of manufacturer's;
- independent of computer systems.

2.3. MANAGER concept

The MANAGER concept was chosen to avoid the use of a built-in control philosophy. It was found that, up to then, most of the traffic control simulation programs had a built-in control philosophy and therefore were problem dedicated. FLEXSYT, in contrary, is a general purpose traffic control simulation program.
The MANAGER of the program must specify some parts of the traffic control process at implementation time of the program. It gives the possibility to generalize signal handling and calculations. At the same time the MANAGER may specify a traffic control philosophy. It gives the MANAGER the opportunity to force its USERS to use a standard type of control
Examples of traffic control philosophy's that can be implemented are 'round-abouts', 'arterials', 'toll-plaza's', 'ramp-metering installations', etc.

2.4. USER concept

In FLEXSYT the USER is defined as the person to whom the MANAGER has allowed the use of the program and the use of his MANAGER dataset (MANDAT).
It's the task of the USER to solve a certain traffic control problem. Therefor the USER has to define his network and his problem dedicated control strategy.

2.5. Traffic control language (FLEXCOL-76-)

A traffic control language for program description of flexible controlled intersections was developed in 1976. Further extensions to meet the requirements of program descriptions of network control were added later. This flexible network traffic control language (known as FLEXCOL-76-) is based upon the rules of boolean algebra and the clear differentiation between the 'change of state of an element' and the 'state of an element'.
The 'change of state of an element' is called EVENT and has no logical meaning. The 'state of an element' has a logical meaning and is called CONDITION, that is: the description of the state of an element is logically true or false.

2.5.1. Elements of FLEXCOL-76-

Basically the traffic control process has three (and no more than three) types of elements. Using these elements gives the opportunity to control the traffic in every desired manner. These elements are: MEMORY elements, TIMER elements and DETECTOR elements.
In many circumstances, the only values for a MEMORY element are '0' and '1'. For this reason a LOGICAL element has been created. Thanks to the simple contents of this element, the syntax of a LOGICAL element is simpler. This increases readability of the traffic control program and gives the opportunity to decrease the use of computer-memory.

2.5.1.1. Names of elements

To meet the requirements of flexibility and independency, the name of each element is declared by the MANAGER. Each element name can have up to 20 significant characters. Legal characters are: capitals, letters and 'underscores'.
So legal element names are:

- MEASURE_PERIOD_	(memory-element),
- GREEN_TIMER_	(timer-element),
- OCCUPANCY_DETECTOR_	(detector-element),
- ALINEA_	(logical-element).

To be user friendly, to increase readability and to decrease the size of a traffic-control-specification, the MANAGER may select an abbreviation instead of the full element-name.

2.5.1.2. Indices of elements

Each element in the traffic control program can have an index. The index of an element can have three levels. Therefore one has to consider, that the traffic engineer has to deal with three levels of control. These levels are: NETWORK level, INTERSECTION level and SIGNAL level.
When dealing for instance with intersection number 001 and with signal number 05 and 45, examples of legal combinations of element names and indices are:

- METERING_TIME_001/00 (INTERSECTION-level),
- ONRAMP_VOLUME_001/00 (INTERSECTION-level),
- _GREEN_001/05 (SIGNAL-level),
- CLEARANCE_TIME_001/45 (SIGNAL-level).

Because of the fact that, at installation time of the traffic control language, the MANAGER doesn't know the intersection numbers nor the signal numbers to be used, we have to define implicit indices, symbolized by the character '$'. Thus legal combinations of element names and indices, to be used by the MANAGER, are:

- MAX_METERING_TIME_$$$/00 (INTERSECTION-level),
- _YELLOW_$$$/$$ (SIGNAL-level).

If it is obvious, the index of an element may be omitted. The use of the default index increases the readability of the control program.

2.5.2. Events in FLEXCOL-76-

The manipulation of all elements is effected by the description of the 'change of state' (EVENT) of these elements by the MANAGER and/or the USER. The syntax of an EVENT (of an element called M with index i) is:

S(Mi = x)

The meaning of 'x' is explained later.

The manipulation of logical elements is the description of the 'change of state' of such an element from 'false' to 'true' (SET EVENT) or from 'true' to 'false' (END EVENT or RESET EVENT). The syntax of these EVENTS (of an element called L with index i) is:

SLi and ELi

2.5.3. Conditions in FLEXCOL-76-

The syntax of a test on the 'state of an element' (CONDITION) appears in six ways. In words, this test can be read as: is the contents of an element (not) equal (or greater) (or less) than the contents of an arithmetical expression. As invert operator the letter 'N' was chosen. The syntax of the six possible CONDITIONS is:

(Mi = x) and (Mi = x)N
(Mi > x) and (Mi > x)N
(Mi < x) and (Mi < x)N

Here also the syntax of a test on the 'state of a logical element' is simpler:

Li and LiN

2.5.4. Functions in FLEXCOL-76-

The meaning of 'x' in the above definitions is more complicated. When changing the state of MEMORY elements or TIMER elements and when using the change of states of DETECTOR elements, the MANAGER and the USER can use functions in a way similar to computer languages. The syntax of the use of these functions is defined in the following way: x is a constant, x is the contents of an element or x is part of a FUNCTION(x,y,z,..). FUNCTIONS that can be used in FLEXCOL-76- are:

- ADD(x,y)	for readability: x+y,
- SUBTRACT(x,y)	for readability: x-y,
- MULTIPLY(x,y)	for readability: x*y,
- DIVIDE(x,y)	for readability: x/y,
- POWER(x,y)	for readability: x**y,
- SQRT(x)	(square-root),
- ALOG(x)	(natural-logarithm),
- LOG(x)	(common-logarithm),
- EXP(x)	(exponential),
- ABS(x)	(absolute-value),
- MAX(x,y,z,..)	(maximum),
- MIN(x,y,z,..)	(minimum).

The definition of 'x' is recurrent, so nesting of the functions is allowed. Functions in FLEXSYT follow the common rules in computation. Apart form using indices, an example of the manipulation of elements is exponential smoothing, with FLOW_OLD_ the element containing the smoothed value, ALPHA_ the element containing the smoothing factor and FLOW_NEW_ the element containing the value of the measured period:

S(FLOW_OLD_016 = ALPHA_ * FLOW_NEW_016 + (1-ALPHA_) * FLOW_OLD_016)

2.5.5. Collections in FLEXCOL-76-

Many times, the traffic engineer implicitly uses the collection or set concept. Examples of a collection in traffic control are: the intersections of a network, the signals of an intersection, the signals of a stage an the conflicts of a signal.
To suit the use of collections, the MANAGER must define the names of the collections he wants his USER to operate with. Definition of these names follows the same rules as the definition of the names of elements. Legal collection names are:

- METERING_INSTALL_	(NETWORK-level),
- METERING_SIGNAL_	(INTERSECTION-level).

In collections the use of an index of an element is implicit. An example of the use of a collection is:

METERING_SIGNAL_001: 01 05 45

2.5.6. Behaviour of elements in FLEXCOL-76-

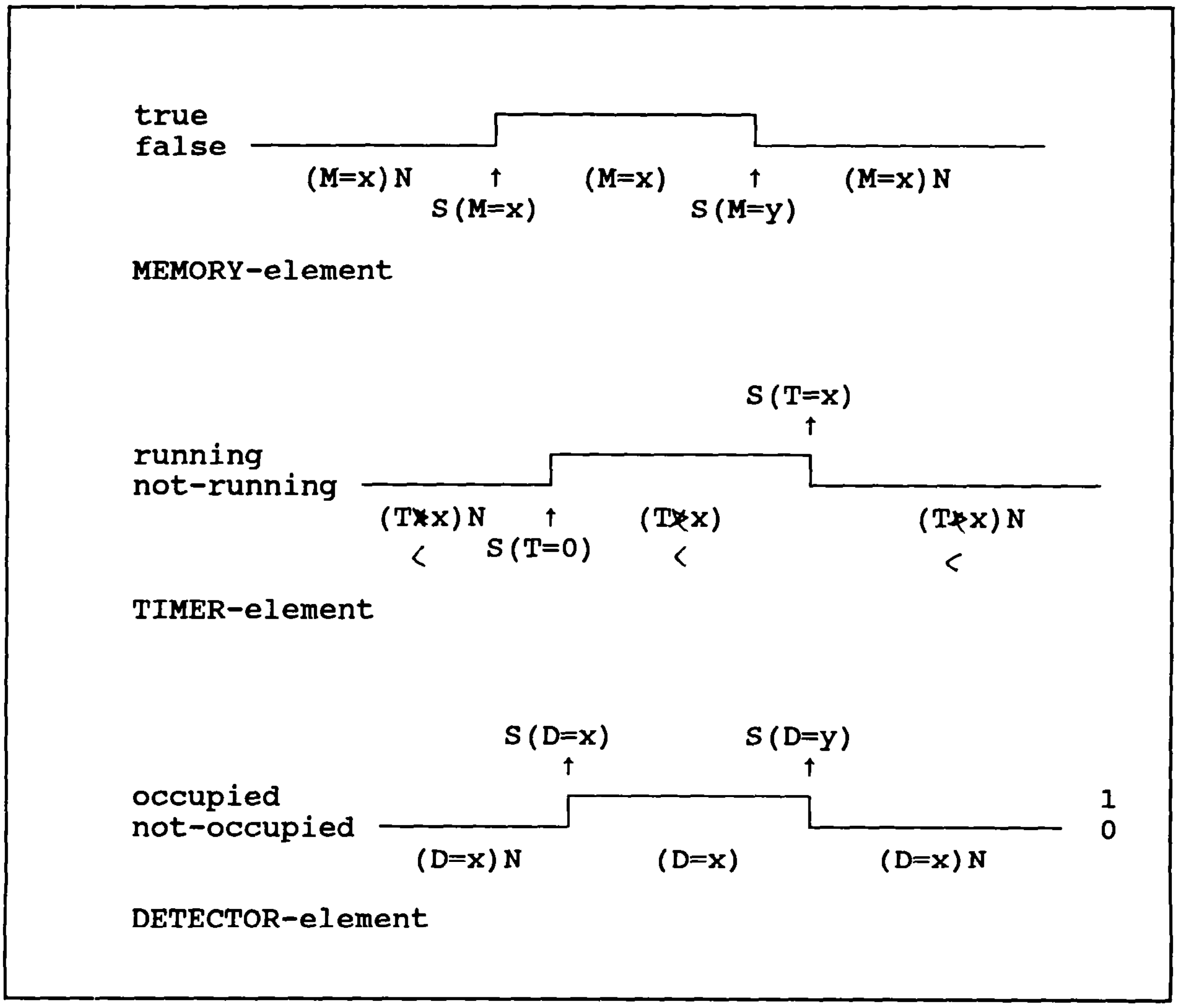

Figure 1. Logical behaviour of the MEMORY-, TIMER-, and DETECTOR-elements in time

The behaviour of the elements in time is given in figure 1. As can be seen from this figure, there is a difference in the use of the arrows:

- an arrow pointing *towards* the figure is a REACTION EVENT. The specification of the 'change of state' of an element is done by the MANAGER and/or the USER.
- an arrow pointing *outwards* the figure is an ACTION EVENT. The MANAGER and/or the USER only may use the 'change of state' but cannot cause the 'change of state'.

2.5.7. Formula concept of FLEXCOL-76-

2.5.7.1. Event assignment statement

The program of the traffic control application (being the specification of the signal plan) contains the descriptions of the 'change of states' of elements in EVENT ASSIGNMENT STATEMENTS. In general, the EVENT ASSIGNMENT STATEMENT is defined in the following way:

REACTION EVENT .= ACTION EVENT . 'condition'

So the manipulation of an element is caused by an event. The way in which this occurs is very similar to the way in which SUBROUTINE calls are effected in computer programs. In the above general description, the meaning of each part is:

- 'REACTION EVENT' contains the description of the element(s) to be manipulated,
- '.=' is the event assignment operator sign,
- 'ACTION EVENT' contains the description of element(s), the manipulation of which causes the assignment,
- '.' is an operator sign
- 'condition' contains the description of a combination of logical conditions that must be fulfilled to effect the assignment. The definition of 'condition' is recurrent, so:

  - 'condition' is 'empty',
  - 'condition' is CONDITION,
  - 'condition' is (CONDITION)N,
  - 'condition' is CONDITION + CONDITION,
  - 'condition' is CONDITION . CONDITION.

  This combination of logical conditions follows the common rules of the boolean-algebra, with the operators:

  - 'N' is the LOGICAL INVERT,
  - '+' is the LOGICAL OR,
  - '.' is the LOGICAL AND.

The definition of the EVENT ASSIGNMENT STATEMENT is also recurrent. So an ACTION EVENT may be substituted by a REACTION EVENT. This allows (but doesn't force) the MANAGER and USER to construct complex control algorithms.

2.5.7.2. Boolean assignment statement

There also exists an easier description of the EVENT ASSIGNMENTS on logical elements if the formulation of the SET EVENT is the inverse of the formulation of the RESET EVENT. For example (using the logical elements A, B and C with index i) when:

SAi .= SBi.Ci + SCi.Bi
EAi .= EBi + ECi

then the simplified form of these two EVENT ASSIGNMENT STATEMENTS is one BOOLEAN ASSIGNMENT STATEMENT:

Ai = Bi . Ci

The BOOLEAN ASSIGNMENT STATEMENT is defined as:

'state of logical element' = 'condition'

Then meaning of each part is:

- 'state of logical element' contains the description of logical elements to be manipulated,
- '=' is the boolean assignment operator sign,
- 'condition' is similar to the definition in the event assignment statement, except that 'condition' may not be 'empty'.

2.6. Structure of the program FLEXSYT-I-

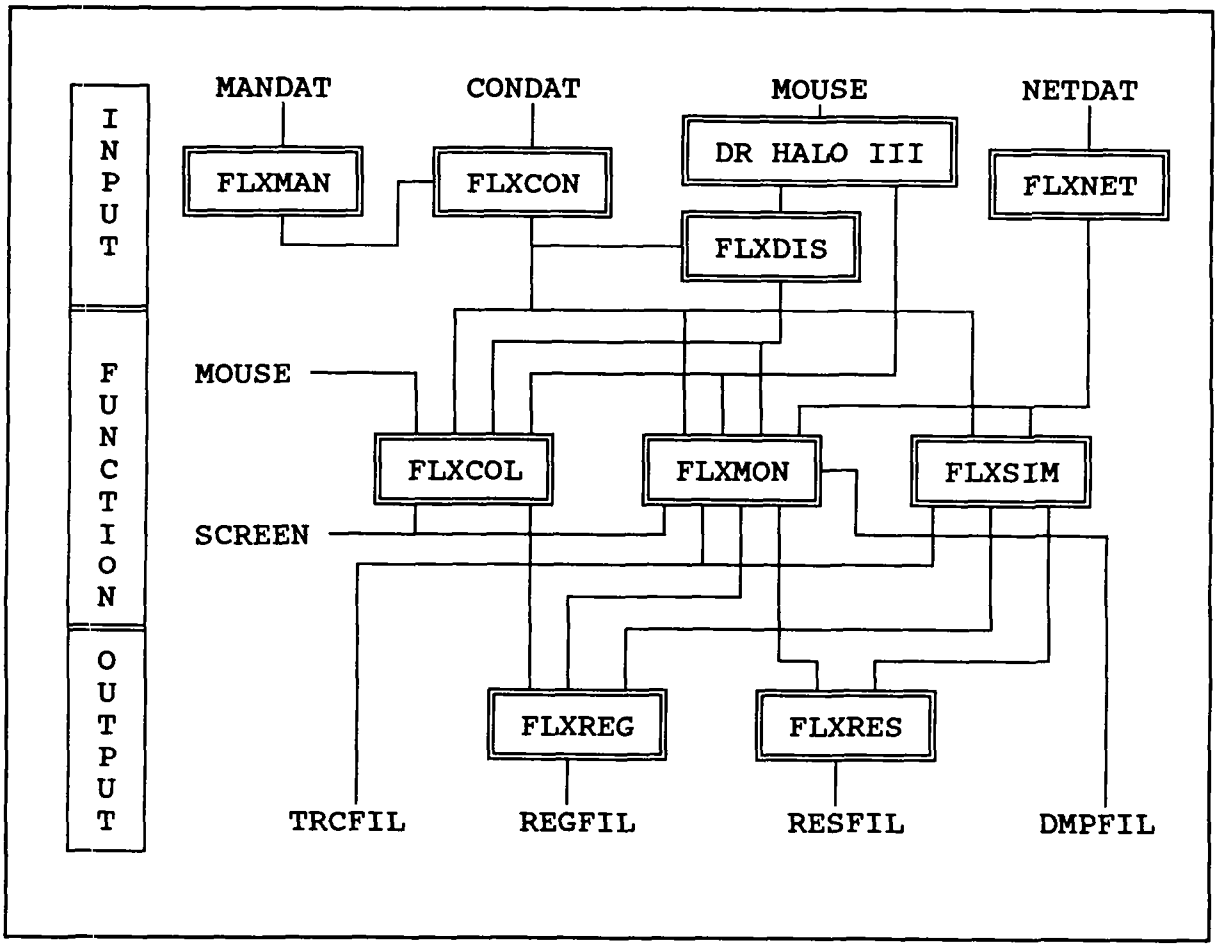

Figure 2. Structure of FLEXSYT

The structure of FLEXSYT is shown in figure 2. FLEXSYT consists of a number of programs which require three input data sets. In the MANDAT dataset, the names of

elements and collections are specified. Also the traffic control routines on network-, controller, and signal level are given here, being the default routines for the user.
In the CONDAT dataset, the problem defined input of the traffic control strategies are given.
In the NETDAT dataset, the problem defined input for the network lay-out is given, as well as the simulation parameters, the traffic flow values and the location of the stoplines and detectors.
Before the simulation starts, the input data sets are screened and checked for a correct syntax and use of parameters by the programs FLXMAN, FLXCON and FLXNET. With FLXDIS it is possible to match a picture of the network, drawn with Dr. Halo or Dr. Genius, with the control strategy specified in the CONDAT dataset. One can check this control strategy by hand, thus without running a simulation, with FLXCOL.
If the input has been compiled successfully, the simulation is effected with the program FLXSIM. With FLXMON the simulation can be followed on the screen. If 'congestion' occurs during simulation. a dataset DMPFIL is created with information about the state of the network. Errors in the traffic control strategy may be traced in the dataset TRCFIL, containing an 'event trace' and available upon request.
While testing the control strategy, a 'state trace' for every simulation second may be obtained with the help of the program FLXREG and available on the REGFIL dataset. This offers a very helpful facility during the development and testing of control strategies.
The results of the simulation can be obtained with FLXRES and are available in the RESFIL dataset.
All other files that are made, like the MANFIL, CONFIL, NETFIL, DISFIL, SIMFIL, CNTFIL, etc., are intermediate and have no meaning to the user.

2.6.1. Input data sets of FLEXSYT-I-

Preceding the start of a simulation run, the program FLEXSYT needs to be loaded with the specification of the subject to be studied or the problem to be solved. This specification is split up in three input data sets:

- MANager DATaset (appendix A.1), with items: title, definition of element names, standard network control statements, standard intersection control statements and standard signal control statements,
- CONtrol DATaset (appendix A.2), with items: title, specific network control statements and specific intersection/signal control statements.
- NETwork DATaset (appendix A.2), with items: title, simulation parameters, generator element values (traffic load), network element description, with: length, saturation flow, signal relation, speed or travel time (per car type) and route parameters (per car type), signal/stopline positions and detector element positions,

These datasets are necessary to run a simulation with FLXSIM. If FLXMON or FLXCOL is used some other files are necessary. The picture of the area to be studied, drawn with Dr. Halo of Dr. Genius, is contained in a .PIC file and the colour pallet in a .PAL file. With FLXDIS the signal movements and detectors defined in the CONDAT dataset can be matched with the picture. This information is stored in the DISplay DATaset and the DISFIL. The DISDAT dataset can be used later on for a new session with FLXDIS. The DISFIL is, together with the CONFIL, the .PIC file and the .PAL file, input for FLXMON and FLXCOL.

2.6.2. Output data sets of FLEXSYT-I-

Before actual simulation, the program checks the input data very thoroughly. It issues DISASTER, FATAL, ERROR or WARNING messages if needed. With correct input data sets the program starts producing its results. In an interactive environment the SIMULATION DATA may appear immediate on a terminal, with items: successful end of simulation or end of simulation, because of congestion. The last message occurs when: the control plan is erroneous, the control plan isn't yet optimized or the network cannot handle the traffic.
After a successful end of simulation with the program FLXREG one can produce a REGFIL (appendix A.4), with every (model) second, the state of signals and detectors. The REGFIL is very helpful during the development of the signal plan.
The results of a simulation are produced with the program FLXRES. The RESFIL (appendix A.5) contains the following traffic data:

- for the signals (totals at stoplines): car flow in vehicles/hour, delay in seconds, green time in seconds, non-green time in seconds and cycle time in seconds.

- for the network (per car type)(totals for each element): flow in vehicles/hour, distance travelled in vehicles*kilometres/hour, delay in vehicles*hour/hour, time spent in vehicles*hour/hour, stops in vehicles/hour and maximum queue in vehicles.

- for the network (per car type)(totals for the network): total distance travelled in vehicles*kilometres/hour, total time spent in vehicles*hour/hour, total delay in vehicles*hour/hour, total stops in vehicles/hour and speed in kilometres/hour,

- traffic streams from generators: delay in seconds/vehicle and distribution of delay (classwidth 10 seconds).

- traffic streams from stoplines: delay in seconds/vehicle and distribution of delay (classwidth 10 seconds).

2.7. Comparison of FLEXSYT-I- with real life

A scientific comparison of FLEXSYT-I- with real life never had been carried out. This was due to the fact that the development of the program wasn't a part of a survey project nor was sponsored by any company and because a new version of FLEXSYT is being developed. After some studies with FLEXSYT, which gave good results, the program was adopted by the Dutch Ministry of Transport.

2.8. Development of FLEXSYT-II-

The good results obtained with FLEXSYT-I- were the reason for the Dutch Ministry of Transport to start a project for the development of FLEXSYT-II-. The main purpose of the project was to integrate some models of the computer program SIGSIM. The main improvements of FLEXSYT-II-, in comparison with FLEXSYT-I-, are going to be:

- from 2 to 7 types of traffic participants: private-cars, lorries, trucks, buses, tramcars, pedestrians, bicycles,
- route-vehicles,
- secondary-conflicts
- unsignalized intersections.

The main objective of the project however is the scientific validation of the network-aspects of the models. For SIGSIM the scientific validation of the intersection-aspects was already performed.
The development of FLEXSYT-II- is in its final stage.

3. PROTOTYPE

3.1. Introduction

In this chapter the input data sets of FLEXSYT for a prototype program for an isolated ramp-metering installation are discussed. First a piece of the input dataset is given and then some comments. The complete text, in one piece, of the MANDAT and CONDAT dataset is contained in appendices B1 and B2. This prototype can be seen in operation on the demo floppy diskette, available with this deliverable.

3.2. MANDAT dataset

```
//MANDAT-dataset for a PROTOTYPE of an isolated RAMP-METERING INSTALLATION

//DEtector-element-names
  _ SPEED_DETECTOR_  LONG_LOOP_  OCCUPANCY_DETECTOR_  SHORT_LOOP_

//TIMer-element-names
  INIT_TIMER_
  GREEN_TIMER_          YELLOW_TIMER_          RED_TIMER_
  METERING_TIMER_       MEASURE_TIMER_

//MEMory-element-names
  PRINT_
  MIN_GREEN_TIME_       MAX_GREEN_TIME_
  MIN_YELLOW_TIME_      MAX_YELLOW_TIME_
  MIN_RED_TIME_
  MIN_METERING_TIME_    MAX_METERING_TIME_
  CUR_METERING_TIME_    METERING_TIME_
  MEASURE_PERIOD_       PERIOD_
  MEASURE_MET_TIME_     CLEARANCE_TIME_

  FLOW_NEW_             FLOW_OLD_              FLOW_CUR_
  FLOW_PER_HOUR_
  SET_ON_FLOW_          SET_OFF_FLOW_

  SPEED_NEW_            SPEED_OLD_             SPEED_CUR_
  VEH_SPEED_OLD_        VEH_SPEED_CUR_
  SPEED_UPSTREAM_       SPEED_DOWNSTREAM_
  SET_MAX_ON_SPEED_     SET_MAX_OFF_SPEED_
  SET_ON_SPEED_         SET_OFF_SPEED_

  ALPHA_INCREASE_FLOW_  ALPHA_DECREASE_FLOW_
  BETA_INCREASE_SPEED_  BETA_DECREASE_SPEED_
  ALPHA_                BETA_

  CAPACITY_
  OCCUPANCY_CUR_      OCCUPANCY_SETPOINT_
  ALINEA_CONSTANT_
  ONRAMP_VOLUME_

//LOGical-element-names
  _GREEN_        _YELLOW_       _RED_
  _DV_           _DP_           _DA_          _DK_
  _MIN_GREEN_    _MAX_GREEN_
  _MIN_YELLOW_   _MAX_YELLOW_
  _MIN_RED_      _VEHICLE_DEPARTED_
  _METERING_     _METER_        _CLEARED_
```

```
_PROC_          _DUMMY_
_ALTERNATE_
_SPEED_LEVEL_
_RWS_           ALINEA_

//COLLECTION-names
 METERING_INSTALL_  METERING_SIGNAL_
```

In the MANDAT dataset the names of the elements are chosen in a more or less self-explanatory way. The meaning of the elements is explained in the next section. e.g. the CONDAT dataset.

```
//$$$/00 statements on intersection level

S(PERIOD_=3600/MEASURE_PERIOD_)                  .= S(INIT_TIMER_=0)
S(MEASURE_TIMER_=0)                              .= S(INIT_TIMER_=0)
S(CUR_METERING_TIME_=MIN_METERING_TIME_)         .= S(INIT_TIMER_=0)
```

In this initiation procedure a help element PERIOD_ is calculated. If for instance the measure period is 60 seconds then PERIOD_ becomes 60. Further the timer for the measure period is started and the metering time is set to its minimum.

```
S_METERING_,S(METERING_TIMER_=0) .= S_GREEN_$$
E_METERING_                      .= S(METERING_TIMER_=CUR_METERING_TIME_)
```

In these two statements an element _METERING_ is introduced, which makes sure that the next green period of a movement can only start when the timer for the metering time has reached a certain value. The current metering time is calculated in the CONDAT dataset.

```
//$$$/$$ statements on signal level

S_CLEARED_,S_RED_ .= S(INIT_TIMER_=0)
```

This part of the MANDAT dataset concerns the initiation statements on signal level. It is stated that the intersection is cleared and that all signals are set on red.

```
S_MIN_GREEN_,S_MAX_GREEN_,S(GREEN_TIMER_=0)    .= S_GREEN_
E_MIN_GREEN_                                   .= S(GREEN_TIMER_=MIN_GREEN_TIME_)
E_MAX_GREEN_                                   .= S(GREEN_TIMER_=MAX_GREEN_TIME_)

S_MIN_YELLOW_,S_MAX_YELLOW_,S(YELLOW_TIMER_=0) .= S_YELLOW_
E_MIN_YELLOW_                                  .= S(YELLOW_TIMER_=MIN_YELLOW_TIME_)
E_MAX_YELLOW_                                  .= S(YELLOW_TIMER_=MAX_YELLOW_TIME_)

S_MIN_RED_,S(RED_TIMER_=0)                   .= S_RED_
E_MIN_RED_                                   .= S(RED_TIMER_=MIN_RED_TIME_)

//END
```

In the last part of the MANDAT dataset the logical elements _MIN_GREEN_ and _MAX_GREEN_ are set to TRUE and the GREEN_TIMER_ is activated when the logical element _GREEN_ of a signal movement is set to TRUE. This means that if _MIN_GREEN_ is TRUE the timer is running. The elements _MIN_GREEN_ and _MAX_GREEN_ are set to

FALSE when the GREEN_TIMER_ reaches a preset time, the minimum green time and maximum green time respectively. The same is applied to the yellow and red periods of the signal movement.

3.3. CONDAT dataset

3.3.1. Initiation part

```
//CONDAT-dataset for a PROTOTYPE of an isolated RAMP-METERING INSTALLATION,
  with the RWS and ALINEA control strategy.

//000
/METERING_INSTALL_:   001
//001
/METERING_SIGNAL_00:     01  05  45
```

The first record is on network level and states that there is one location with a metering installation. The second record states that there are three signals on this site (or intersection) 001, namely 01, 05 and 45, where 01 is a dummy signal, 05 is the signal for the metering light and 45 for the bus light.

```
/RWS_                  = T
/ALINEA_               = F
```

Here a choice can be made between the Rijkswaterstaat and ALINEA algorithm. If RWS_ is TRUE the Rijkswaterstaat strategy and if ALINEA is TRUE the ALINEA strategy is in use.

```
/CAPACITY_             = 4400
/ALINEA_CONSTANT_      = 70
/OCCUPANCY_SETPOINT_   = 0.18
/ONRAMP_VOLUME_        = 800
/MEASURE_MET_TIME_     = 1.0
/MIN_METERING_TIME_    = 4.5
/MAX_METERING_TIME_    = 12.0
/MEASURE_PERIOD_       = 30.0

/ALPHA_INCREASE_FLOW_ = 0.25
/ALPHA_DECREASE_FLOW_ = 0.25
/BETA_INCREASE_SPEED_ = 0.2
/BETA_DECREASE_SPEED_ = 0.1
```

In this part of the CONDAT dataset the intersection memory elements are initiated. The meaning of these elements is:

CAPACITY_ : (fixed) capacity in veh/hour of the motorway used in the Rijkswaterstaat control law (see also Deliverable 7a, 3.2.)

ALINEA_CONSTANT_ : positive factor for the integral part of the ALINEA control law (see also Deliverable 7a, 3.1.)

OCCUPANCY_SETPOINT_ : set-value for the downstream occupancy (see also Deliverable 7a, 3.1.)

MEASURE_MET_TIME_	: step for smoothing the metering time if the calculated metering time is different from the current metering time
MIN_METERING_TIME_	: minimum metering time
MAX_METERING_TIME_	: maximum metering time
MEASURE_PERIOD_	: period in which the flow, speed and occupancy are measured and the speed is smoothed; after this period the flow is smoothed and the metering time is calculated
ALPHA_INCREASE_FLOW_	: factor to smooth an increasing flow
ALPHA_DECREASE_FLOW_	: factor to smooth an decreasing flow
BETA_INCREASE_SPEED_	: factor to smooth an increasing speed
BETA_DECREASE_SPEED_	: factor to smooth an decreasing speed

```
/SET_ON_FLOW_          = 3500
/SET_OFF_FLOW_         = 3000
/SET_ON_SPEED_         = 50.0
/SET_OFF_SPEED_        = 70.0
/SET_MAX_ON_SPEED_     = 35
/SET_MAX_OFF_SPEED_    = 50
```

These values are threshold values for putting the program of the ramp-metering installation in operation and out of operation and for setting the metering time to its maximum value and back to its calculated value.

```
/CLEARANCE_TIME_05     = 2.0
/CLEARANCE_TIME_45     = 2.0

/MIN_GREEN_TIME_05     = 1.0
/MIN_GREEN_TIME_45     = 1.0
/MAX_GREEN_TIME_05     = 5.0
/MAX_GREEN_TIME_45     = 5.0
/MIN_YELLOW_TIME_05    = 0.5
/MIN_YELLOW_TIME_45    = 0.5
/MAX_YELLOW_TIME_05    = 2.0
/MAX_YELLOW_TIME_45    = 2.0
/MIN_RED_TIME_05       = 2.0
/MIN_RED_TIME_45       = 2.0
```

In the above part several memory elements on signal level are initiated. The meaning of these elements is:

CLEARANCE_TIME_	: time needed to clear the intersection
MIN_GREEN_TIME_	: minimum green time
MAX_GREEN_TIME_	: maximum green time
MIN_YELLOW_TIME_	: minimum yellow time
MAX_YELLOW_TIME_	: maximum yellow time
MIN_RED_TIME_	: minimum red time

```
_DV_011 = (SPEED_DETECTOR_011>0)
_DV_012 = (SPEED_DETECTOR_012>0)
_DV_013 = (SPEED_DETECTOR_013>0)
_DV_014 = (SPEED_DETECTOR_014>0)
_DV_015 = (SPEED_DETECTOR_015>0)
_DV_016 = (SPEED_DETECTOR_016>0)

_DK_051 = (SHORT_LOOP_051>0)
_DK_052 = (SHORT_LOOP_052>0)
_DK_053 = (SHORT_LOOP_053>0)
```

```
_DP_054 = (LONG_LOOP_054>0)
_DK_055 = (SHORT_LOOP_055>0)
_DK_451 = (SHORT_LOOP_451>0)
_DK_452 = (SHORT_LOOP_452>0)
_DK_453 = (SHORT_LOOP_453>0)
```

In the above statements the detector elements are joined with logical elements to improve readability and to avoid ambiguous warning messages.
The speed detectors are situated on both lanes of the main road 1400 meters and 400 meters upstream the on-ramp and 400 meters downstream the on-ramp.
In reality the occupancy detectors are placed 400 meters downstream the on-ramp and they measure the **time** that they are occupied. Occupancy detectors in FLEXSYT measure the **number of vehicles** that occupy a detector and divide that number through the maximum number of vehicles that can occupy the detector. So in the program the occupancy detectors are placed right downstream the on-ramp and are 600 meters long. This length guarantees an accuracy of one percent, because the vehicles in FLEXSYT are six meters long.
The short loop detectors and the long loop detector on the on-ramp are used for counting the arrivals, for an early detection, for a demand, for starting yellow, for starting red and for counting the departures.
The situation of all the detectors is shown in figure 3

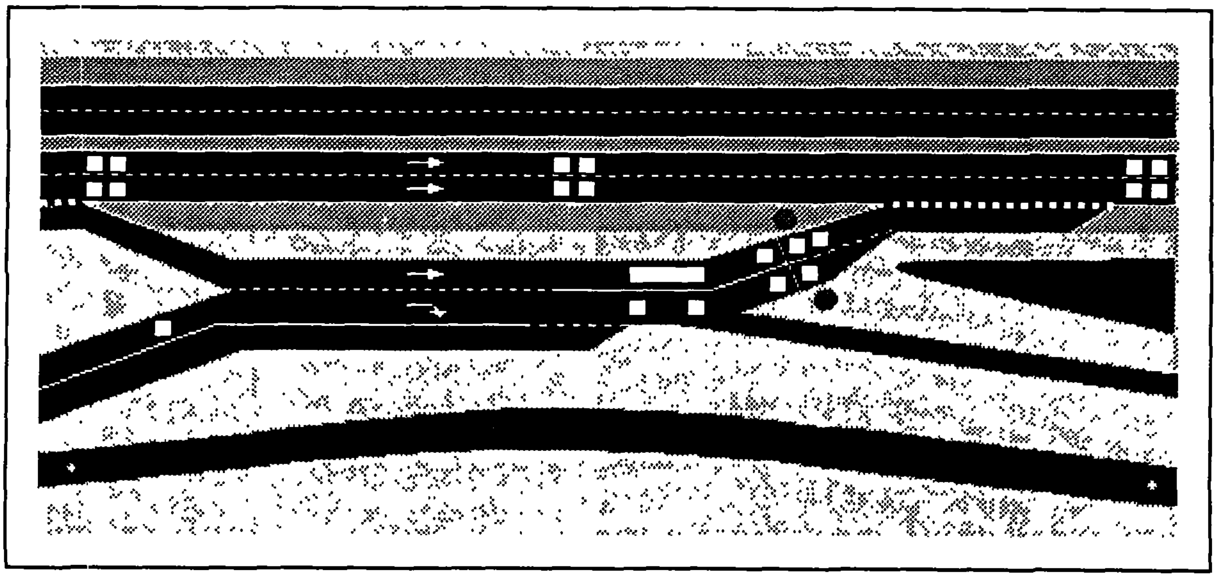

Figure 3. Situation of detectors in the simulation

3.3.2. Counting and measuring part

```
S(FLOW_NEW_016=FLOW_NEW_016+1) .= S_DV_011+S_DV_012
S(FLOW_NEW_014=FLOW_NEW_014+1) .= S_DV_013+S_DV_014
S(FLOW_NEW_052=FLOW_NEW_052+1) .= S_DK_055 + S_DK_453
S(FLOW_NEW_051=FLOW_NEW_051+1) .= S_DK_052 + S_DK_451
```

These statements count the vehicles 1400 and 400 meters upstream the on-ramp and the arriving and departing vehicles on the on-ramp.

```
S(VEH_SPEED_CUR_013=SPEED_DETECTOR_013) .= S_DV_013

S(BETA_=BETA_INCREASE_SPEED_) .= S_DV_013.(VEH_SPEED_CUR_013>VEH_SPEED_OLD_013)
S(BETA_=BETA_DECREASE_SPEED_) .= S_DV_013.(VEH_SPEED_CUR_013<VEH_SPEED_OLD_013)

S(VEH_SPEED_OLD_013=BETA_*VEH_SPEED_CUR_013+(1-BETA_)*VEH_SPEED_OLD_013) .= S_DV_013


S(VEH_SPEED_CUR_014=SPEED_DETECTOR_014) .= S_DV_014

S(BETA_=BETA_INCREASE_SPEED_) .= S_DV_014.(VEH_SPEED_CUR_014>VEH_SPEED_OLD_014)
S(BETA_=BETA_DECREASE_SPEED_) .= S_DV_014.(VEH_SPEED_CUR_014<VEH_SPEED_OLD_014)

S(VEH_SPEED_OLD_014=BETA_*VEH_SPEED_CUR_014+(1-BETA_)*VEH_SPEED_OLD_014) .= S_DV_014


S(VEH_SPEED_CUR_015=SPEED_DETECTOR_015) .= S_DV_015

S(BETA_=BETA_INCREASE_SPEED_) .= S_DV_015.(VEH_SPEED_CUR_015>VEH_SPEED_OLD_015)
S(BETA_=BETA_DECREASE_SPEED_) .= S_DV_015.(VEH_SPEED_CUR_015<VEH_SPEED_OLD_015)

S(VEH_SPEED_OLD_015=BETA_*VEH_SPEED_CUR_015+(1-BETA_)*VEH_SPEED_OLD_015) .= S_DV_015


S(VEH_SPEED_CUR_016=SPEED_DETECTOR_016) .= S_DV_016

S(BETA_=BETA_INCREASE_SPEED_) .= S_DV_016.(VEH_SPEED_CUR_016>VEH_SPEED_OLD_016)
S(BETA_=BETA_DECREASE_SPEED_) .= S_DV_016.(VEH_SPEED_CUR_016<VEH_SPEED_OLD_016)

S(VEH_SPEED_OLD_016=BETA_*VEH_SPEED_CUR_016+(1-BETA_)*VEH_SPEED_OLD_016) .= S_DV_016
```

In this part of the CONDAT dataset the individual speed of the vehicles 400 meters upstream and 400 meters downstream the on-ramp is smoothed.

```
S_PROC_,E_PROC_,S(MEASURE_TIMER_=0) .= S(MEASURE_TIMER_=MEASURE_PERIOD_)
```

This statement starts a procedure for smoothing the flow and calculating the metering time after a measure period has passed. After this is done, the measure timer is again started.

```
S(ALPHA_=ALPHA_INCREASE_FLOW_) .= S_PROC_.(FLOW_NEW_016>FLOW_OLD_016)
S(ALPHA_=ALPHA_DECREASE_FLOW_) .= S_PROC_.(FLOW_NEW_016<FLOW_OLD_016)

S(FLOW_OLD_016=ALPHA_*FLOW_NEW_016+(1-ALPHA_)*FLOW_OLD_016) .= S_PROC_
S(FLOW_CUR_016=FLOW_NEW_016),S(FLOW_NEW_016=0)              .= S_PROC_

S(ALPHA_=ALPHA_INCREASE_FLOW_) .= S_PROC_.(FLOW_NEW_014>FLOW_OLD_014)
S(ALPHA_=ALPHA_DECREASE_FLOW_) .= S_PROC_.(FLOW_NEW_014<FLOW_OLD_014)

S(FLOW_OLD_014=ALPHA_*FLOW_NEW_014+(1-ALPHA_)*FLOW_OLD_014) .= S_PROC_
S(FLOW_CUR_014=FLOW_NEW_014),S(FLOW_NEW_014=0)              .= S_PROC_

S(FLOW_CUR_052=FLOW_NEW_052),S(FLOW_NEW_052=0) .= S_PROC_
S(FLOW_CUR_051=FLOW_NEW_051),S(FLOW_NEW_051=0) .= S_PROC_
```

These statements smooth the flow 1400 meters and 400 meters upstream the on-ramp. Before the smoothing is carried out, a smoothing factor is determined. Further the flow on the on-ramp is measured.

```
S(SPEED_UPSTREAM_=(VEH_SPEED_OLD_013+VEH_SPEED_OLD_014)/2)   .= S_PROC_
S(SPEED_DOWNSTREAM_=(VEH_SPEED_OLD_015+VEH_SPEED_OLD_016)/2) .= S_PROC_
```

The average speed upstream and downstream the on-ramp is calculated.

```
S(FLOW_PER_HOUR_ = FLOW_OLD_014 * PERIOD_) .= S_PROC_

S_METER_ .= S_PROC_.((SPEED_UPSTREAM_ < SET_ON_SPEED_) + (SPEED_DOWNSTREAM_ < SET_ON_SPEED_) +
               + (FLOW_PER_HOUR_ > SET_ON_FLOW_))
E_METER_ .= S_PROC_.(SPEED_UPSTREAM_ > SET_OFF_SPEED_) . (SPEED_DOWNSTREAM_ > SET_OFF_SPEED_) .
               . (FLOW_PER_HOUR_ < SET_OFF_FLOW_) . _DP_054N
```

In this procedure it is determined whether or not ramp-metering is necessary. Therefor the flow per hour is calculated with the help of the element PERIOD_. The installation program is activated when a measure period is over and if the (smoothed) upstream speed is below a preset value or if the (smoothed downstream speed is below the same value or if the (smoothed) upstream flow is above a preset value. The ramp-metering installation program is deactivated when a measure period is over and if the upstream speed is above a certain value and if the downstream speed is above the same value and if the upstream flow goes is a preset value. But this is only done if there is no queue before the stopline on the on-ramp.

```
S_SPEED_LEVEL_ .= S_PROC_.((SPEED_UPSTREAM_<SET_MAX_ON_SPEED_)+(SPEED_DOWNSTREAM_<SET_MAX_ON_SPEED_))
E_SPEED_LEVEL_ .= S_PROC_.(SPEED_UPSTREAM_>SET_MAX_OFF_SPEED_).(SPEED_DOWNSTREAM_>SET_MAX_OFF_SPEED_)
```

After every measure period it is determined whether or not the metering time must be set to its maximum value. This is the case if the upstream or the downstream speed is below a certain value. The metering time is reset to its calculated value if both the upstream and downstream speed are above a preset value.

```
S(ONRAMP_VOLUME_=CAPACITY_-FLOW_PER_HOUR_)                    .= S_PROC_.RWS_._SPEED_LEVEL_N
```

In the Rijkswaterstaat algorithm the allowed on-ramp volume is calculated directly from the capacity and the smoothed upstream flow (see Deliverable 7a, 3.2.).

```
S(OCCUPANCY_CUR_=(OCCUPANCY_DETECTOR_011+OCCUPANCY_DETECTOR_012)/2) .= S_PROC_.ALINEA_
S(ONRAMP_VOLUME_=ONRAMP_VOLUME_+ALINEA_CONSTANT_*(OCCUPANCY_SETPOINT_-OCCUPANCY_CUR_))
                                                               .= S_PROC_.ALINEA_
```

In the ALINEA algorithm the allowed on-ramp volume is calculated from the occupancy (see Deliverable 7a, 3.1.).

```
S(METERING_TIME_=3600/ONRAMP_VOLUME_)                         .= S_PROC_
S(METERING_TIME_=MIN(METERING_TIME_,MAX_METERING_TIME_))      .= S_PROC_
S(METERING_TIME_=MAX(METERING_TIME_,MIN_METERING_TIME_))      .= S_PROC_
S(METERING_TIME_=MAX_METERING_TIME_)                          .= S_PROC_.RWS_._SPEED_LEVEL_
```

Then the metering time is calculated and compared with a minimum and maximum value. The metering time is set to its maximum value when _SPEED_LEVEL_ is TRUE, but only in the Rijkswaterstaat algorithm (see Deliverable 7a, 3.3.).

3.3.3. Signal handling part

```
S_DUMMY_012,S(CUR_METERING_TIME_=CUR_METERING_TIME_+MEASURE_MET_TIME_) .= (E_RED_05+E_RED_45).
  .(METERING_TIME_>(CUR_METERING_TIME_+MEASURE_MET_TIME_)).(_DK_053+_DK_452)
S_DUMMY_012,S(CUR_METERING_TIME_=CUR_METERING_TIME_-MEASURE_MET_TIME_) .= (E_RED_05+E_RED_45).
  .(METERING_TIME_<(CUR_METERING_TIME_-MEASURE_MET_TIME_)).(_DK_053+_DK_452)
S(CUR_METERING_TIME_=METERING_TIME_)                                  .= (E_RED_05+E_RED_45).
  ._DUMMY_012N
E_DUMMY_012                                                           .= E_RED_05 + E_RED_45
```

Every green period of the metering light or the bus light it is determined whether or not the metering time must be adjusted. Therefor a dummy element is used. The metering time is stepwise adjusted to the calculated metering time, but only when a detector just before a stopline is occupied, otherwise it is done directly.

```
S_GREEN_45                                         .= E_RED_45
E_CLEARED_45                                       .= S_GREEN_45
E_GREEN_45 .= (E_MIN_GREEN_45._VEHICLE_DEPARTED_45 + S_DK_451._MAX_GREEN_45 + E_MAX_GREEN_45)._METER_
             + S_METER_
S_YELLOW_45,E_ALTERNATE_05,E_VEHICLE_DEPARTED_45 .= E_GREEN_45
S_ALTERNATE_45                                     .= E_GREEN_45 . _DK_053
E_YELLOW_45                                        .= E_MIN_YELLOW_45
S_RED_45                                           .= E_YELLOW_45
E_RED_45    .= ((S_DK_452+E_METERING_+E_MIN_RED_45+S_RED_05+S_CLEARED_05)._DK_452._METERING_N.
              ._MIN_RED_45N._RED_05._CLEARED_05._ALTERNATE_45N)._METER_ + E_METER_

S_VEHICLE_DEPARTED_45                              .= S_DK_451 . _MIN_GREEN_45
S_CLEARED_45                                       .= S(RED_TIMER_45 = CLEARANCE_TIME_45)


S_GREEN_05                                         .= E_RED_05
E_CLEARED_05                                       .= S_GREEN_05
E_GREEN_05 .= (E_MIN_GREEN_05._VEHICLE_DEPARTED_05 + S_DK_052._MAX_GREEN_05 + E_MAX_GREEN_05)._METER_+
             + S_METER_
S_YELLOW_05,E_ALTERNATE_45,E_VEHICLE_DEPARTED_05 .= E_GREEN_05
S_ALTERNATE_05                                     .= E_GREEN_05 . _DK_452
E_YELLOW_05                                        .= S_DK_051._MIN_YELLOW_05N + E_MAX_YELLOW_05
S_RED_05                                           .= E_YELLOW_05
E_RED_05    .= ((S_DK_053+E_METERING_+E_MIN_RED_05+S_RED_45+S_CLEARED_45)._DK_053._METERING_N.
              ._MIN_RED_05N._RED_45._CLEARED_45._ALTERNATE_05N)._METER_ + E_METER_

S_VEHICLE_DEPARTED_05                              .= S_DK_052 . _MIN_GREEN_05
S_CLEARED_05                                       .= S(RED_TIMER_05 = CLEARANCE_TIME_05)

//END
```

In the last part of the CONDAT dataset the handling of the metering and bus signal is described.

A green period is started when a red period is terminated. The starting of a green period makes the logical element for the clearing of the intersection FALSE. A green period is terminated under three events: when the minimum green time is reached, if the vehicle is departed, when the detector after the stopline becomes occupied, if the green period is in extended green and when the maximum green time is reached.

The termination of a green period starts the yellow period, gives the turn to the other signal, if there is a demand for that signal, and makes the logical element for the departure of a vehicle FALSE. An yellow period is terminated for the bus signal when the minimum yellow time is reached and for the metering signal when the detector six meters after the stopline is occupied, if the minimum yellow time is reached and when the maximum yellow time is reached.

The termination of an yellow period starts a red period. A red period is terminated when

the detector for the stopline is occupied, when the metering time is reached, when the minimum red time is reached, when the red phase of the other signal is started or when the intersection is cleared. But also several conditions must be satisfied: the detector before the stopline must be occupied, the metering time and the minimum red time must have been reached, the other signal must be red, the intersection must have been cleared and the signal must have the turn. The red period is also ended when ramp-metering is no longer necessary.
A vehicle is said to be departed when the detector after the stopline becomes occupied within the minimum green time.
The intersection is said to be cleared when the red timer of the signal has reached a certain preset value.

3.4. Conclusion

This completes the software prototype for an isolated ramp-metering installation. With this deliverable a floppy diskette is included. On this diskette a demonstration of the prototype can be seen on the Coentunnel sight and further information on FLEXSYT, input and output data sets is available. Just put the diskette in a station and type **go<RETURN>** after the DOS-prompt. A screen will be shown which gives information about how the demonstration can be started.
If the demonstration is running several things can be seen. When a detector changes colour (from grey to white) a vehicle is detected. When there is no ramp-metering the lights are both green. Whether or not there is ramp-metering can be seen in the right lower corner of the screen. When there is ramp-metering the light before the strategy that is used is green and the others are red. When there is no ramp-metering the light before **no ramp-metering** is green.
The state of demonstration can be changed with function key **<F4>** from **SPEEDY** to **REAL TIME** and back. With function key **<F3>** a situation can be held and with **<F1>** help is available. With **<F2>** the demonstration can be stopped.
The screen which appears in the demonstration on the floppy is shown in figure 7.

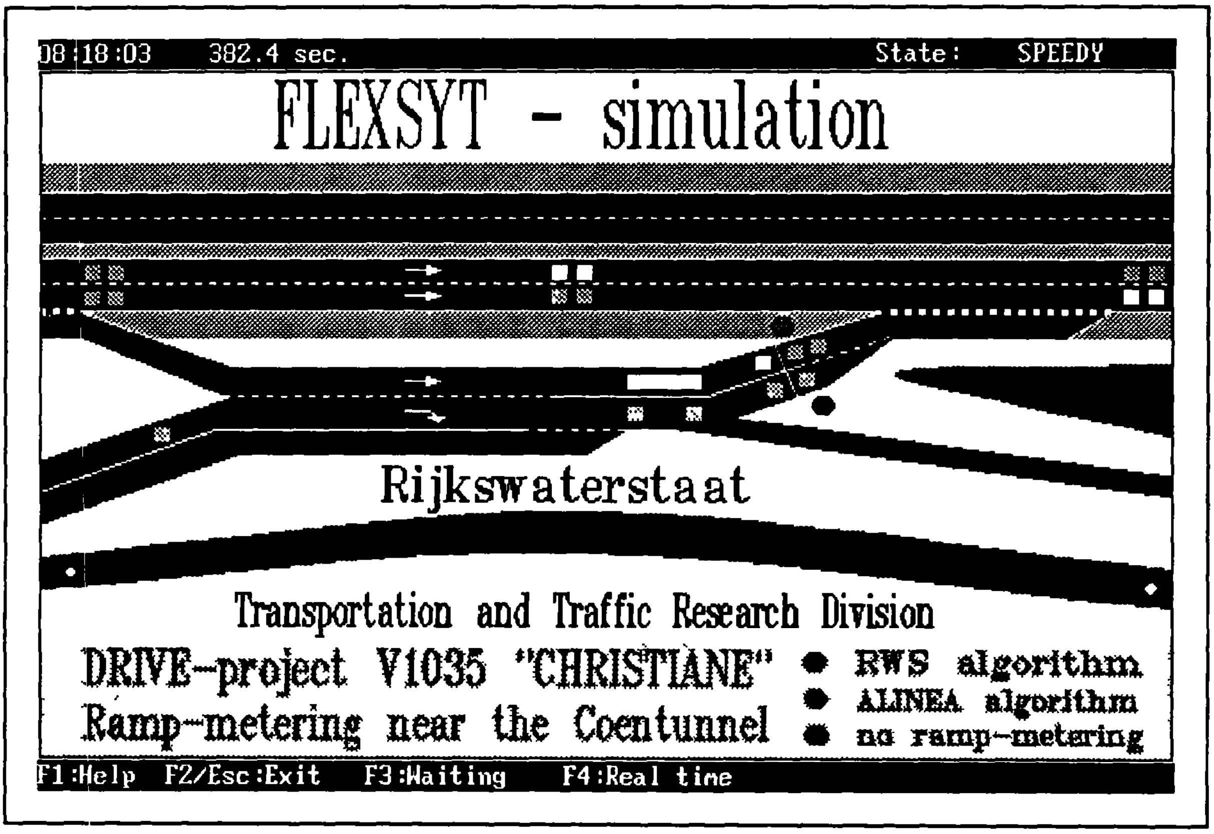

Figure 4. Demonstration of the software prototype

APPENDICES

A. Examples

A.1. MANDAT dataset

```
//MANDAT dataset for FIXED-TIME control

//DET
   DS   DV   DP   DA   DK
//TIM
   CYCLE_timer   YELLOW_timer
//MEM
   PRINT   CYCLE_time   GO_time   NOGO_time
//LOG
   GO_signal
   DS_  DV_  DP_  DA_  DK_
//COL
   CONTROLLERS   SIGNALS
//$$$/$$
S(PRINT1='*')   .= SDS_2
S(PRINT1='=')   .= SDP_2+S(PRINT1=' ').DP_2.DS_2N
S(PRINT1='>')   .= SDK_2+S(PRINT1=' ').DK_2.DP_2N.DS_2N
S(PRINT1=' ')   .= EDS_2+EDP_2+EDK_2

S(PRINT2='#')   .= SGO_signal
S(PRINT2='x')   .= EGO_signal
S(PRINT2=' ')   .= S(YELLOW_timer=3)

S(PRINT3='*')   .= SDS_1
S(PRINT3='=')   .= SDP_1+S(PRINT3=' ').DP_1.DS_1N
S(PRINT3='<')   .= SDK_1+S(PRINT3=' ').DK_1.DP_1N.DS_1N
S(PRINT3=' ')   .= EDS_1+EDP_1+EDK_1


SGO_signal      .= S(CYCLE_timer=GO_time)
EGO_signal      .= S(CYCLE_timer=NOGO_time)

S(YELLOW_timer=0).= EGO_signal
S(CYCLE_timer=0) .= S(CYCLE_timer=CYCLE_time$$$/00)

//END
```

A.2. CONDAT dataset

```
//CONDAT dataset for FIXED-TIME control of the TRANSYT network

//000
/CONTROLLERS:    001    002    003
/CYCLE_time$$$/:  72     72     36

//001
/SIGNALS:        02   05   08   11
/PRINT$$=         1    2    3    4
/GO_time$$=      22   47   22   47
/NOGO_time$$=    41   16   41   19

;detector interface
DS_111 = (DS111>0)
DS_112 = (DS112>0)

DP_021 = (DP021>0)
DP_051 = (DP051>0)
DP_081 = (DP081>0)
DP_111 = (DP111>0)

DK_021 = (DK021>0)
DK_051 = (DK051>0)
DK_081 = (DK081>0)
DK_111 = (DK111>0)

//002
/SIGNALS:        05   08   11
/PRINT$$=         6    7    8
/GO_time$$=      11   66   11
/NOGO_time$$=    61    5   61

;detector interface
DS_111 = (DS111>0)
DS_112 = (DS112>0)

DP_051 = (DP051>0)
DP_081 = (DP081>0)
DP_111 = (DP111>0)

DK_051 = (DK051>0)
DK_081 = (DK081>0)
DK_111 = (DK111>0)

//003
/SIGNALS:        02   08   10   12
/PRINT$$=        10   11   12   13
/GO_time$$=      15   15    0    0
/NOGO_time$$=    31   32   12   11

;detector interface
DP_021 = (DP021>0)
DP_081 = (DP081>0)
DP_101 = (DP101>0)
DP_121 = (DP121>0)

DK_021 = (DK021>0)
DK_081 = (DK081>0)
DK_101 = (DK101>0)
DK_121 = (DK121>0)

//END
```

A.3. NETDAT dataset

```
//NETDAT dataset of the TRANSYT network (100% traffic load)

;VOR TIM NRN BL1 BL2 REG RES   RND
 240 240   1 100 100   1   0 12345

//GEN
;GNUM  GTOL GTYP >--------------------GINT---------------------<
  102 -1025    1   50   0   0   0   0   0   0   0   0   0   0   0
  105 -1054    1  350   0   0   0   0   0   0   0   0   0   0   0
  108 -1084    1  300   0   0   0   0   0   0   0   0   0   0   0
  208 -2084    1  200   0   0   0   0   0   0   0   0   0   0   0
  211 -2115    1  840   0   0   0   0   0   0   0   0   0   0   0
  251 -2115    2   60   0   0   0   0   0   0   0   0   0   0   0
  302 -3024    1  200   0   0   0   0   0   0   0   0   0   0   0

//NET
;LNUM LTH LSAT CON SGL >-LYTM-< >-LEQL-< >----LTOL----< >-------LTFR-------<
-1054 999 1800   0   0  40    0    0   0    0 1053    0    0 350   0
 1053 300 1800   0   0  40    0    0   0    0 1052    0    0 350   0
 1052  30 1800 001  05  40    0    0   0    0 1051    0    0 350   0
 1051  20 1200 001  05  40    0    0   0    0 2053 3083   50 150 105

 2053 350 1800   0   0  42    0    0   0    0 2052    0    0 250   0
 2052  30 1800 002  05  42    0    0   0    0 2051    0    0 250   0
 2051  20 1500 002  05  42    0    0   0    0    0 3114    0 150 100

-1084 999 1800   0   0  34    0    0   0    0 1083    0    0 300   0
 1083 300 1800   0   0  34    0    0   0    0 1082    0    0 300   0
 1082  30 1800 001  08  34    0    0   0    0 1081    0    0 300   0
 1081  20 1500 001  08  34    0    0   0 2053 3083    0   50 200  50

 3083 220 1800   0   0  37    0    0   0    0 3082    0    0 350   0
 3082  30 1800 003  08  37    0    0   0    0 3081    0    0 350   0
 3081  20 1800 003  08  37    0

-2084 999 1800   0   0  39    0    0   0    0 2083    0    0 200   0
 2083 300 1800   0   0  39    0    0   0    0 2082    0    0 200   0
 2082  30 1800 002  08  39    0    0   0    0 2081    0    0 200   0
 2081  20 1500 002  08  39    0    0   0    0 3114 1115   50 100  50

 3114 250 1800   0   0  42    0    0   0 3103 3123    0  100 250   0
 3103  10 1800   0   0  42    0    0   0    0 3102    0    0 350   0
 3102  30 1800 003  10  42    0    0   0    0 3101    0    0 250   0
 3101  20 1200 003  10  42    0    0   0    0    0 1024    0   0 100
 3123  10 1800   0   0  42    0    0   0 3122    0    0  100   0   0
 3122  30 1800 003  12  42    0    0   0 3121    0    0  100   0   0
 3121  20 1500 003  12  42    0

-2115 999 1800   0   0  39   36    0   0    0 2114    0    0 840   0   0 60   0
 2114 150 1800 002  11  39   36    0   0    0 2113    0    0 840   0   0 60   0
 2113 150 1800   0   0  39   36    0   0    0 2112    0    0 840   0   0 60   0
 2112  30 1800 002  11  39   36    0   0    0 2111    0    0 840   0   0 60   0
 2111  20 1500 002  11  39   36    0   0 3114 1115    0  150 690   0   0 60   0

 1115 188 1800   0   0  43   36    0   0    0 1114 1514    0 740   0   0  0  60
 1114  12 1800 001  11  43    0    0   0    0 1113    0    0 740   0   0  0   0
 1514  12  720 001  11   0  -20    0   0    0 1113    0    0   0   0   0 60   0
 1113 150 1800 001  11  43   36    0   0    0 1112    0    0 740   0   0 60   0
 1112  30 1800 001  11  43   36    0   0    0 1111    0    0 740   0   0 60   0
 1111  20 1500 001  11  43   36

-3024 999 1800   0   0  36    0    0   0    0 3023    0    0 200   0
 3023 300 1800   0   0  36    0    0   0    0 3022    0    0 200   0
 3022  30 1800 003  02  36    0    0   0    0 3021    0    0 200   0
 3021  20 1800 003  02  36    0    0   0    0 1023    0    0 200   0

-1025 999  720   0   0  25    0    0   0    0 1024    0    0  50   0
 1024 150 1800   0   0  25    0    0   0    0 1022    0    0  50   0
 1023 220 1800   0   0  41    0    0   0    0 1022    0    0 300   0
 1022  30 1800 001  02  41    0    0   0    0 1021    0    0 350   0
```

```
1021  20 1500 001  02  41    0    0   0    0    0 2053   0 300  50

//SIG
1021 1051 1081 1111 2051 2081 2111 3021 3081 3101 3121

//DET
/DS1
 2111 1111

/DS2
 2114 1114 1514

/DP1
 1022 1052 1082 1112 2052 2082 2112 3022 3082 3102 3122

/DK1
 1021 1051 1081 1111 2051 2081 2111 3021 3081 3101 3121

//END
```

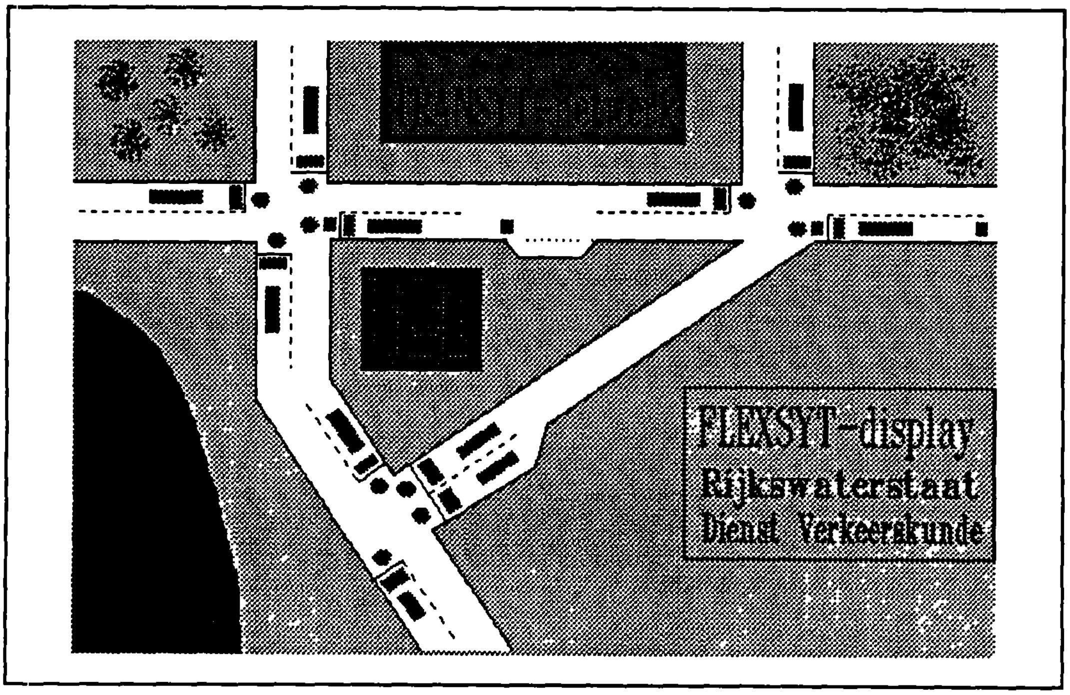

Figure 5. The network of the NETDAT data set

A.4. REGFIL dataset

```
FLEXSYTI-9.0.1.    * TRAFFIC-CONTROL-SIMULATION-PACKAGE * PART FLXREG * RWS/DVK_CX * ROTTERDAM  *
07-10-90  *  15:09:37  * TABLE 1
ONETDAT dataset of the TRANSYT network (100% traffic load)

 CONDAT dataset for FIXED-TIME control of the TRANSYT network

 MANDAT dataset for FIXED-TIME control


          >-------------INDICES---------------------<
   TIME   001   001         002         003   003

     IN    02    08          08          08    12

SECONDS      001   001   002   002   003   003

              05    11    05    11    02    10

    241    #=  < #=  =    #   = #=    #  #
    242    #=  < #=  <    #     #<    #  #=  <
    243    #=  < #=  <    #   < #=    #  #=  <
    244    #=  < #=  <    #   < #<    #  #=  <
    245    #<  < #<  <    #   < #=    #= #=  <
    246    #=  < #=  <    #   < #<    #= #   <
    247    #=  < #=  <    #   < #=    #= #<  <
    248    #<  = #<  <    #   < #<    x= #   <
    249    #=  = #=  =    #   < #=    x  x   <
    250    #   = #<  =    #   < #=    x< x   <
    251    #<  = #=  =    #   < #=     = x   <
    252    #   < #=  =    #   < #<     =     <
    253    #   = #<  =    #   < #=     =    #< #
    254    #<  = #=  =    #   < #<     <    #< #=
    255    #<  = #<  =    #   < #=     <    #< #=
    256    #   = #   =    #   < #=     <    #< #=
    257    #   = #<  =    #   < #<     <    #  #
    258    x   = x=  =    #   < #=     <    #  #<
    259    x   = x=  =    #   < #<     <    #  #
    260    x   = x=  =    #   < #=     <  = #  #=
    261        =  =  =    #   < #<     <  = #  #=
    262        =     =    #   < #=     <  = #  #=
    263        =  <  =    #   < #<     <    #  #=
    264       #=  < #=    #   < #=     <  < #  x=
    265       #=  < #=    #   < #=     <  < x  x<
    266       #=  < #=    #   < #<     <  < x  x
    267       #=  < #=    #   < #=     <  < x   <
    268       #=  < #=    #   < #<    #< #<     <
    269       #<  = #<    #   < #=    #< #=     <
    270       #<  = #=    #   < #<    #< #=     <
    271       #=  = #=    #   < #=    #  #=     <
    272       #=  = #=    #   < #=    #< #=     <
    273       #=  < #<    #   < #<    #  #=     <
    274       #<  = #=    #   < #=    #  #<     <
    275       #=  = #<    #   < #<    #  #=     <
    276       #=  = #=    #   < #=    #  #=     <
    277       #=  = #=    #   < #<    #  #=     <
    278       #=  = #<    x   < x     #  #<     <
    279       #<  = #=    x=  < x<    #  #=     =
    280       #=  = #<    x=  < x     #  #<     =
    281       #=  = #=     =  <       #  #=     =
    282       #=  = #<     =  <       #  #      <
    283     = #<  = #=     < #<       #  #      <
    284     = #=  = #<     < #<       x  #<     <
    285     = #   = #=     < #<       x  x      <
    286     = #   = #<     < # *      x  x      <
    287       #=  = #=     < #           x      <
    288     < #<  = #<     < #   =          =     <
    289     < #=  = #=     < #   =          = #  #<
```

A.5. RESFIL dataset

FLEXSYTI-9.0.1. * TRAFFIC-CONTROL-SIMULATION-PACKAGE * PART FLXRES * RWS/DVK_CX * ROTTERDAM * 30-09-90 * 14:37:01 * page 1
NETDAT dataset of the TRANSYT network (100% traffic load)
CONDAT dataset for FIXED-TIME control of the TRANSYT network
MANDAT dataset for FIXED-TIME control

>>>> ATTENTION >>>>>>>>> traffic-models of FLXSYT-I- not scientific validated, so be careful with results <<<<<<<<< ATTENTION <<<<
Results obtained after 10800. seconds of simulation

NON-GREENTIMES / PERCENTAGES-OF-NON-GREENTIMES-IN-CLASSES-OF-10-SECONDS-WIDTH

SIGNAL NUMBER	NUM OF CYCL	AV	SD	MIN	MAX	0/10	10/20	20/30	30/40	40/50	50/60	60/70	70/80	80/90	90/100	100/110	110/120	120/130	130/140	140/150	150/160	160/170	170/180	180/190	190/200	200/210	210/220	220/230	230/240	240/INF
001/02	150	53	0	53	53	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
001/05	150	31	0	31	31	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GREENTIMES / PERCENTAGES-OF-----GREENTIMES-IN-CLASSES-OF-10-SECONDS-WIDTH

SIGNAL NUMBER	NUM OF CYCL	AV	SD	MIN	MAX	0/10	10/20	20/30	30/40	40/50	50/60	60/70	70/80	80/90	90/100	100/110	110/120	120/130	130/140	140/150	150/160	160/170	170/180	180/190	190/200	200/210	210/220	220/230	230/240	240/INF
001/02	149	19	0	19	19	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
001/05	150	41	0	41	41	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CYCLE TIMES / PERCENTAGES-OF----CYCLE TIMES-IN-CLASSES-OF-10-SECONDS-WIDTH

SIGNAL NUMBER	NUM OF CYCL	AV	SD	MIN	MAX	0/10	10/20	20/30	30/40	40/50	50/60	60/70	70/80	80/90	90/100	100/110	110/120	120/130	130/140	140/150	150/160	160/170	170/180	180/190	190/200	200/210	210/220	220/230	230/240	240/INF
001/02	149	72	0	72	72	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
001/05	150	72	0	72	72	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXSYTI-9.0.1. * TRAFFIC-CONTROL-SIMULATION-PACKAGE * PART FLXRES * RWS/DVK_CX * ROTTERDAM * 30-09-90 * 14:37:01 * page 2

NETDAT dataset of the TRANSYT network (100% traffic load)
CONDAT dataset for FIXED-TIME control of the TRANSYT network
MANDAT dataset for FIXED-TIME control
>>>> ATTENTION >>>>>>>>> traffic-models of FLXSYT-I- not scientific validated, so be careful with results <<<<<<<<< ATTENTION <<<<

Results obtained after 10800. seconds of simulation

DELAY-AT-STOPLINE / PERCENTAGES-OF-DELAY-IN-CLASSES-OF-5-OR-10-SECONDS-WIDTH

FROM GENER.	TO SIGNAL	TYP	NUM OF CARS	AV	SD	MAX	0/5	5/10	10/15	15/20	20/30	30/40	40/50	50/60	60/70	70/80	80/90	90/100	100/110	110/120	120/130	130/140	140/150	150/160	160/170	170/180	180/190	190/INF
102	001/02	1	132	60.3	39.9	188.5	3	2	4	5	6	16	11	13	8	5	5	6	3	3	1	4	2	1	0	0	2	0
105	001/05	1	1034	11.5	11.3	46.7	42	10	11	10	20	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

>-----------DELAY-AT-STOPLINE------------< >--------------PERCENTAGES-OF-DELAY-IN-CLASSES-OF-5-OR-10-SECONDS-WIDTH--------------<

FROM SIGNAL	TO SIGNAL	TYP	NUM OF CARS	AV (IN SECONDS)	SD (IN SECONDS)	MAX (IN SECONDS)	0/ 5/	5/ 10/	10/ 15/	15/ 20/	20/ 30/	30/ 40/	40/ 50/	50/ 60/	60/ 70/	70/ 80/	80/ 90/	90/100/	100/110/	110/120/	120/130/	130/140/	140/150/	150/160/	160/170/	170/180/	180/190/	190/INF
001/02	002/05	1	139	14.5	8.1	28.8	18	6	16	33	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
001/05	002/05	1	516	2.1	3.2	15.5	80	17	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXSYTI-9.0.1. * TRAFFIC-CONTROL-SIMULATION-PACKAGE * PART FLXRES * RWS/DVK_CX * ROTTERDAM * 30-09-90 * 14:37:01 * page 3

NETDAT dataset of the TRANSYT network (100% traffic load)
CONDAT dataset for FIXED-TIME control of the TRANSYT network
MANDAT dataset for FIXED-TIME control

>>>> ATTENTION >>>>>>>>> traffic-models of FLXSYT-I- not scientific validated, so be careful with results <<<<<<<<< ATTENTION <<<<
Results obtained after 12 subrun(s) of 900. seconds each

SIGNAL NUMBER	LANE AT LINK NUM	SAT FLOW VH/H	TYPE-ONE FLOW VH/H	TYPE-ONE NUM SUB	TYPE-ONE DELAY AV	TYPE-ONE DELAY SD	TYPE-ONE DELAY MIN	TYPE-ONE DELAY MAX	TYPE-TWO FLOW VH/H	TYPE-TWO NUM SUB	TYPE-TWO DELAY AV	TYPE-TWO DELAY SD	TYPE-TWO DELAY MIN	TYPE-TWO DELAY MAX	DEG OF SAT %
001/02	1021	1500	336	12	65.0	30.2	37.7	129.1	0	0	*****	*****	*****	*****	85
001/05	1051	1200	345	12	11.4	2.1	8.6	14.5	0	0	*****	*****	*****	*****	50

FLEXSYTI-9.0.1. * TRAFFIC-CONTROL-SIMULATION-PACKAGE * PART FLXRES * RWS/DVK_CX * ROTTERDAM * 30-09-90 * 14:37:01 * page 3

NETDAT dataset of the TRANSYT network (100% traffic load)
CONDAT dataset for FIXED-TIME control of the TRANSYT network
MANDAT dataset for FIXED-TIME control

>>>> ATTENTION >>>>>>>>> traffic-models of FLXSYT-I- not scientific validated, so be careful with results <<<<<<<<< ATTENTION <<<<
Results obtained after 10800. seconds of simulation

>------------------<>--------------DISTRIBUTION-OF-MAXIMUM-QUEUE-DURING-CYCLES-OF-SIGNAL-05-OF-INTERSECTION-001---------------<

LINK NUMBER	SIGNAL NUMBER	MAX QUEUE	0/ 0/	1/ 1/	2/ 2/	3/ 3/	4/ 4/	5/ 5/	6/ 7/	8/ 9/	10/ 11/	12/ 13/	14/ 15/	16/ 17/	18/ 19/	20/ 21/	22/ 23/	24/ 25/	26/ 30/	31/ 35/	36/ 40/	41/ 45/	46/ 50/	51/ 55/	56/ 60/	61/ 65/	66/ 70/	71/ 75/	76/INF/
1053		50	146	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1052	001/05	5	109	12	11	10	2	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1051	001/05	3	16	37	30	66	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2052	002/05	5	140	6	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXSYTI-9.0.1. * TRAFFIC-CONTROL-SIMULATION-PACKAGE * PART FLXRES * RWS/DVK_CX * ROTTERDAM * 30-09-90 * 14:37:01 * page 4
NETDAT dataset of the TRANSYT network (100% traffic load)
CONDAT dataset for FIXED-TIME control of the TRANSYT network
MANDAT dataset for FIXED-TIME control
>>>> ATTENTION >>>>>>>>> traffic-models of FLXSYT-I- not scientific validated, so be careful with results <<<<<<<<< ATTENTION <<<<
Results obtained after 12 subrun(s) of 900. seconds each
This table concerns car-type=1

NUM OF SUB	LINK NUMBER	FLOW VEH/H AV	SD	MIN	MAX	SAT FLOW VH/H	LINK lgth MTRS	DIST. TRVLLD VH.KM/H	DELAY VEH.H/H AV	SD	MIN	MAX	YNEY TIME SECS	TIME SPENT VEH.H/H	STOPS VEH/H AV	SD	MIN	MAX	MAX QUEUE ALL TYPES VEH %	CAP	AV	SD	MIN	MAX
12	1053	346	33	284	408	1800	300	103.80	.03	.01	.02	.06	27.0	2.63	1	3	0	12	0	50	0	0	0	2
12	1052	345	32	284	408	1800	30	10.35	.25	.12	.04	.44	2.7	.51	38	21	0	72	11	5	3	1	0	5
12	1051	344	34	280	408	1200	20	6.89	.79	.14	.60	.99	1.8	.96	104	14	84	132	30	3	3	0	3	3
12	2053	269	32	208	316	1800	350	94.38	.03	.01	.01	.04	30.0	2.28	0	0	0	0	0	58	0	0	0	0

NUM OF SUB	TOTAL DISTANCE TRAVELLED VEH.KM/H	TOTAL TIME SPENT VEH.H/H	TOTAL DELAY VEH.H/H	TOTAL STOPS VEH/H	SPEED KM/H	
12	1337.52	62.69	28.80	2388.	21.51	AVERAGE
	65.03	7.12	5.99	363.	1.86	STANDARD DEVIATION
	1250.29	51.02	18.96	1884.	18.93	MINIMUM
	1469.91	77.64	40.35	3232.	24.74	MAXIMUM

FLEXSYTI-9.0.1. * TRAFFIC-CONTROL-SIMULATION-PACKAGE * PART FLXRES * RWS/DVK_CX * ROTTERDAM * 30-09-90 * 14:37:01 * page 6

NETDAT dataset of the TRANSYT network (100% traffic load)
CONDAT dataset for FIXED-TIME control of the TRANSYT network
MANDAT dataset for FIXED-TIME control

>>>> ATTENTION >>>>>>>>> traffic-models of FLXSYT-I- not scientific validated, so be careful with results <<<<<<<<< ATTENTION <<<<
Results obtained after 12 subrun(s) of 900. seconds each
This table concerns car-type=2

NUM OF SUB	LINK NUMBER	FLOW VEH/H AV	SD	MIN	MAX	SAT FLOW VH/H	LINK LGTH MTRS	DIST. TRVLLD VH.KM/H	DELAY VEH.H/H AV	SD	MIN	MAX	YNEY TIME SECS	TIME SPENT VEH.H/H	STOPS VEH/H AV	SD	MIN	MAX	MAX QUEUE ALL TYPES VEH %	CAP	AV	SD	MIN	MAX
12	2114	60	1	56	64	1800	150	9.00	.00	.00	.00	.00	15.0	.25	0	0	0	0	0	25	0	0	0	0
12	2113	60	2	56	64	1800	150	9.05	.03	.03	.00	.09	15.0	.28	4	4	0	12	6	25	10	6	2	23

NUM OF SUB	TOTAL DISTANCE TRAVELLED VEH.KM/H	TOTAL TIME SPENT VEH.H/H	TOTAL DELAY VEH.H/H	TOTAL STOPS VEH/H	SPEED KM/H	
12	45.08	1.95	.39	49.	23.16	AVERAGE
	.79	.11	.10	15.	1.19	STANDARD DEVIATION
	44.05	1.81	.24	24.	21.54	MINIMUM
	46.75	2.11	.53	72.	24.91	MAXIMUM

B. Prototype input datasets

B.1. MANDAT dataset

```
//MANDAT-dataset for a PROTOTYPE for an isolated RAMP-METERING INSTALLATION

//DETector-element-names
  _ SPEED_DETECTOR_  LONG_LOOP_  OCCUPANCY_DETECTOR_  SHORT_LOOP_

//TIMer-element-names
  INIT_TIMER_
  GREEN_TIMER_        YELLOW_TIMER_        RED_TIMER_
  METERING_TIMER_     MEASURE_TIMER_

//MEMory-element-names
  PRINT_
  MIN_GREEN_TIME_        MAX_GREEN_TIME_
  MIN_YELLOW_TIME_       MAX_YELLOW_TIME_
  MIN_RED_TIME_
  MIN_METERING_TIME_     MAX_METERING_TIME_
  CUR_METERING_TIME_     METERING_TIME_
  MEASURE_PERIOD_        PERIOD_
  MEASURE_MET_TIME_      CLEARANCE_TIME_

  FLOW_NEW_              FLOW_OLD_           FLOW_CUR_
  FLOW_PER_HOUR_
  SET_ON_FLOW_           SET_OFF_FLOW_

  SPEED_NEW_             SPEED_OLD_          SPEED_CUR_
  VEH_SPEED_OLD_         VEH_SPEED_CUR_
  SPEED_UPSTREAM_        SPEED_DOWNSTREAM_
  SET_MAX_ON_SPEED_      SET_MAX_OFF_SPEED_
  SET_ON_SPEED_          SET_OFF_SPEED_

  ALPHA_INCREASE_FLOW_   ALPHA_DECREASE_FLOW_
  BETA_INCREASE_SPEED_   BETA_DECREASE_SPEED_
  ALPHA_                 BETA_

  CAPACITY_
  OCCUPANCY_CUR_         OCCUPANCY_SETPOINT_
  ALINEA_CONSTANT_
  ONRAMP_VOLUME_

//LOGical-element-names
  _GREEN_        _YELLOW_      _RED_
  _DV_           _DP_          _DA_          _DK_
  _MIN_GREEN_    _MAX_GREEN_
  _MIN_YELLOW_    _MAX_YELLOW_
  _MIN_RED_      _VEHICLE_DEPARTED_
  _METERING_     _METER_       _CLEARED_
  _PROC_         _DUMMY_
  _ALTERNATE_
  _SPEED_LEVEL_
  _RWS_          ALINEA_

//COLLECTION-names
  METERING_INSTALL_  METERING_SIGNAL_


//$$$/00 statements on intersection level

S(PERIOD_=3600/MEASURE_PERIOD_)               .= S(INIT_TIMER_=0)
S(MEASURE_TIMER_=0)                           .= S(INIT_TIMER_=0)
S(CUR_METERING_TIME_=MIN_METERING_TIME_)      .= S(INIT_TIMER_=0)

S_METERING_,S(METERING_TIMER_=0) .= S_GREEN_$$
E_METERING_                      .= S(METERING_TIMER_=CUR_METERING_TIME_)
```

```
//$$$/$$ statements on signal level

S_CLEARED_ .= S(INIT_TIMER_=0)
S_RED_     .= S(INIT_TIMER_=0)

S_MIN_GREEN_,S_MAX_GREEN_,S(GREEN_TIMER_=0)    .= S_GREEN_
E_MIN_GREEN_                                   .= S(GREEN_TIMER_=MIN_GREEN_TIME_)
E_MAX_GREEN_                                   .= S(GREEN_TIMER_=MAX_GREEN_TIME_)

S_MIN_YELLOW_,S_MAX_YELLOW_,S(YELLOW_TIMER_=0) .= S_YELLOW_
E_MIN_YELLOW_                                  .= S(YELLOW_TIMER_=MIN_YELLOW_TIME_)
E_MAX_YELLOW_                                  .= S(YELLOW_TIMER_=MAX_YELLOW_TIME_)

S_MIN_RED_,S(RED_TIMER_=0)                     .= S_RED_
E_MIN_RED_                                     .= S(RED_TIMER_=MIN_RED_TIME_)

//END
```

B.2. CONDAT dataset

```
//CONDAT-dataset for a PROTOTYPE of an isolated RAMP-METERING INSTALLATION, with the RWS and ALINEA
  control strategy.

//000
/METERING_INSTALL_:   001
//001
/METERING_SIGNAL_00:   01  05  45


/RWS_                    = T
/ALINEA_                 = F
/CAPACITY_               = 4400
/ALINEA_CONSTANT_        = 70
/OCCUPANCY_SETPOINT_     = 0.18
/ONRAMP_VOLUME_          = 800
/MEASURE_MET_TIME_       = 1.0
/SET_MAX_ON_SPEED_       = 25
/SET_MAX_OFF_SPEED_      = 45
/MIN_METERING_TIME_      = 4.5
/MAX_METERING_TIME_      = 12.0
/MEASURE_PERIOD_         = 30.0

/ALPHA_INCREASE_FLOW_    = 0.25
/ALPHA_DECREASE_FLOW_    = 0.25
/BETA_INCREASE_SPEED_    = 0.2
/BETA_DECREASE_SPEED_    = 0.1

/SET_ON_FLOW_            = 3500
/SET_OFF_FLOW_           = 3000
/SET_ON_SPEED_           = 50.0
/SET_OFF_SPEED_          = 70.0

/CLEARANCE_TIME_05       = 2.0
/CLEARANCE_TIME_45       = 2.0
/MIN_GREEN_TIME_05       = 1.0
/MIN_GREEN_TIME_45       = 1.0
/MAX_GREEN_TIME_05       = 5.0
/MAX_GREEN_TIME_45       = 5.0
/MIN_YELLOW_TIME_05      = 0.5
/MIN_YELLOW_TIME_45      = 0.5
/MAX_YELLOW_TIME_05      = 2.0
/MAX_YELLOW_TIME_45      = 2.0
/MIN_RED_TIME_05         = 2.0
/MIN_RED_TIME_45         = 2.0


_DV_011 = (SPEED_DETECTOR_011>0)
_DV_012 = (SPEED_DETECTOR_012>0)
_DV_013 = (SPEED_DETECTOR_013>0)
_DV_014 = (SPEED_DETECTOR_014>0)
_DV_015 = (SPEED_DETECTOR_015>0)
_DV_016 = (SPEED_DETECTOR_016>0)

_DK_051 = (SHORT_LOOP_051>0)
_DK_052 = (SHORT_LOOP_052>0)
_DK_053 = (SHORT_LOOP_053>0)
_DP_054 = (LONG_LOOP_054>0)
_DK_055 = (SHORT_LOOP_055>0)

_DK_451 = (SHORT_LOOP_451>0)
_DK_452 = (SHORT_LOOP_452>0)
_DK_453 = (SHORT_LOOP_453>0)


S(FLOW_NEW_016=FLOW_NEW_016+1) .= S_DV_011+S_DV_012
S(FLOW_NEW_014=FLOW_NEW_014+1) .= S_DV_013+S_DV_014
```

```
S(FLOW_NEW_052=FLOW_NEW_052+1) .= S_DK_055 + S_DK_453
S(FLOW_NEW_051=FLOW_NEW_051+1) .= S_DK_052 + S_DK_451


S(VEH_SPEED_CUR_013=SPEED_DETECTOR_013) .= S_DV_013

S(BETA_=BETA_INCREASE_SPEED_) .= S_DV_013.(VEH_SPEED_CUR_013>VEH_SPEED_OLD_013)
S(BETA_=BETA_DECREASE_SPEED_) .= S_DV_013.(VEH_SPEED_CUR_013<VEH_SPEED_OLD_013)

S(VEH_SPEED_OLD_013=BETA_*VEH_SPEED_CUR_013+(1-BETA_)*VEH_SPEED_OLD_013) .= S_DV_013


S(VEH_SPEED_CUR_014=SPEED_DETECTOR_014) .= S_DV_014

S(BETA_=BETA_INCREASE_SPEED_) .= S_DV_014.(VEH_SPEED_CUR_014>VEH_SPEED_OLD_014)
S(BETA_=BETA_DECREASE_SPEED_) .= S_DV_014.(VEH_SPEED_CUR_014<VEH_SPEED_OLD_014)

S(VEH_SPEED_OLD_014=BETA_*VEH_SPEED_CUR_014+(1-BETA_)*VEH_SPEED_OLD_014) .= S_DV_014


S(VEH_SPEED_CUR_015=SPEED_DETECTOR_015) .= S_DV_015

S(BETA_=BETA_INCREASE_SPEED_) .= S_DV_015.(VEH_SPEED_CUR_015>VEH_SPEED_OLD_015)
S(BETA_=BETA_DECREASE_SPEED_) .= S_DV_015.(VEH_SPEED_CUR_015<VEH_SPEED_OLD_015)

S(VEH_SPEED_OLD_015=BETA_*VEH_SPEED_CUR_015+(1-BETA_)*VEH_SPEED_OLD_015) .= S_DV_015


S(VEH_SPEED_CUR_016=SPEED_DETECTOR_016) .= S_DV_016

S(BETA_=BETA_INCREASE_SPEED_) .= S_DV_016.(VEH_SPEED_CUR_016>VEH_SPEED_OLD_016)
S(BETA_=BETA_DECREASE_SPEED_) .= S_DV_016.(VEH_SPEED_CUR_016<VEH_SPEED_OLD_016)

S(VEH_SPEED_OLD_016=BETA_*VEH_SPEED_CUR_016+(1-BETA_)*VEH_SPEED_OLD_016) .= S_DV_016


S_PROC_,E_PROC_,S(MEASURE_TIMER_=0) .= S(MEASURE_TIMER_=MEASURE_PERIOD_)


S(ALPHA_=ALPHA_INCREASE_FLOW_) .= S_PROC_.(FLOW_NEW_016>FLOW_OLD_016)
S(ALPHA_=ALPHA_DECREASE_FLOW_) .= S_PROC_.(FLOW_NEW_016<FLOW_OLD_016)

S(FLOW_OLD_016=ALPHA_*FLOW_NEW_016+(1-ALPHA_)*FLOW_OLD_016) .= S_PROC_
S(FLOW_CUR_016=FLOW_NEW_016),S(FLOW_NEW_016=0)              .= S_PROC_


S(ALPHA_=ALPHA_INCREASE_FLOW_) .= S_PROC_.(FLOW_NEW_014>FLOW_OLD_014)
S(ALPHA_=ALPHA_DECREASE_FLOW_) .= S_PROC_.(FLOW_NEW_014<FLOW_OLD_014)

S(FLOW_OLD_014=ALPHA_*FLOW_NEW_014+(1-ALPHA_)*FLOW_OLD_014) .= S_PROC_
S(FLOW_CUR_014=FLOW_NEW_014),S(FLOW_NEW_014=0)              .= S_PROC_


S(FLOW_CUR_052=FLOW_NEW_052),S(FLOW_NEW_052=0) .= S_PROC_
S(FLOW_CUR_051=FLOW_NEW_051),S(FLOW_NEW_051=0) .= S_PROC_


S(SPEED_UPSTREAM_=(VEH_SPEED_OLD_013+VEH_SPEED_OLD_014)/2)   .= S_PROC_
S(SPEED_DOWNSTREAM_=(VEH_SPEED_OLD_015+VEH_SPEED_OLD_016)/2) .= S_PROC_


S(FLOW_PER_HOUR_ = FLOW_OLD_014 * PERIOD_) .= S_PROC_

S_METER_ .= S_PROC_.((SPEED_UPSTREAM_ < SET_ON_SPEED_)+(SPEED_DOWNSTREAM_ < SET_ON_SPEED_)+
                     +(FLOW_PER_HOUR_ > SET_ON_FLOW_))
E_METER_ .= S_PROC_.(SPEED_UPSTREAM_ > SET_OFF_SPEED_).(SPEED_DOWNSTREAM_ > SET_OFF_SPEED_).
                    .(FLOW_PER_HOUR_ < SET_OFF_FLOW_)._DP_054N


S_SPEED_LEVEL_ .= S_PROC_.((SPEED_UPSTREAM_<SET_MAX_ON_SPEED_)+(SPEED_DOWNSTREAM_<SET_MAX_ON_SPEED_))
E_SPEED_LEVEL_ .= S_PROC_.(SPEED_UPSTREAM_>SET_MAX_OFF_SPEED_).(SPEED_DOWNSTREAM_>SET_MAX_OFF_SPEED_)

S(ONRAMP_VOLUME_=CAPACITY_-FLOW_PER_HOUR_)                   .= S_PROC_.RWS_._SPEED_LEVEL_N
```

```
S(OCCUPANCY_CUR_=(OCCUPANCY_DETECTOR_011 + OCCUPANCY_DETECTOR_012)/2) .= S_PROC_.ALINEA_
S(ONRAMP_VOLUME_=ONRAMP_VOLUME_+ALINEA_CONSTANT_*(OCCUPANCY_SETPOINT_-OCCUPANCY_CUR_))
                                                                        .= S_PROC_.ALINEA_


S(METERING_TIME_=3600/ONRAMP_VOLUME_)                           .= S_PROC_
S(METERING_TIME_=MIN(METERING_TIME_,MAX_METERING_TIME_))        .= S_PROC_
S(METERING_TIME_=MAX(METERING_TIME_,MIN_METERING_TIME_))        .= S_PROC_
S(METERING_TIME_=MAX_METERING_TIME_)                            .= S_PROC_.RWS_._SPEED_LEVEL_


S_DUMMY_012,S(CUR_METERING_TIME_=CUR_METERING_TIME_+MEASURE_MET_TIME_)        .= (E_RED_05 + E_RED_45).
   .(METERING_TIME_>(CUR_METERING_TIME_+MEASURE_MET_TIME_)).(_DK_053+_DK_452)
S_DUMMY_012,S(CUR_METERING_TIME_=CUR_METERING_TIME_-MEASURE_MET_TIME_)        .= (E_RED_05 + E_RED_45).
   .(METERING_TIME_<(CUR_METERING_TIME_-MEASURE_MET_TIME_)).(_DK_053+_DK_452)
S(CUR_METERING_TIME_=METERING_TIME_)                                                  .= (E_RED_05 +
E_RED_45)._DUMMY_012N
E_DUMMY_012                                                                   .= E_RED_05 + E_RED_45


S_GREEN_45                                          .= E_RED_45
E_CLEARED_45                                        .= S_GREEN_45
E_GREEN_45 .= (E_MIN_GREEN_45._VEHICLE_DEPARTED_45+S_DK_451._MAX_GREEN_45+E_MAX_GREEN_45) . _METER_ +
             + S_METER_
S_YELLOW_45,E_ALTERNATE_05,E_VEHICLE_DEPARTED_45 .= E_GREEN_45
S_ALTERNATE_45                                      .= E_GREEN_45 . _DK_053
E_YELLOW_45                                         .= E_MIN_YELLOW_45
S_RED_45                                            .= E_YELLOW_45
E_RED_45    .= ((S_DK_452+S_DK_453+E_METERING_+E_MIN_RED_45+S_RED_05+S_CLEARED_05).(_DK_452+_DK_453)
             ._METERING_N . _MIN_RED_45N . _RED_05 . _CLEARED_05 . _ALTERNATE_45N) . _METER_ + E_METER_

S_VEHICLE_DEPARTED_45                               .= S_DK_451._MIN_GREEN_45
S_CLEARED_45                                        .= S(RED_TIMER_45=CLEARANCE_TIME_45)


S_GREEN_05                                          .= E_RED_05
E_CLEARED_05                                        .= S_GREEN_05
E_GREEN_05 .= (E_MIN_GREEN_05._VEHICLE_DEPARTED_05+S_DK_052._MAX_GREEN_05+E_MAX_GREEN_05)._METER_+
             + S_METER_
S_YELLOW_05,E_ALTERNATE_45,E_VEHICLE_DEPARTED_05 .= E_GREEN_05
S_ALTERNATE_05                                      .= E_GREEN_05 . _DK_452
E_YELLOW_05                                         .= S_DK_051 . _MIN_YELLOW_05N + E_MAX_YELLOW_05
S_RED_05                                            .= E_YELLOW_05
E_RED_05    .= ((S_DK_053 + E_METERING_ + E_MIN_RED_05 + S_RED_45 + S_CLEARED_45) . _DK_053 . _METERING_N
             . _MIN_RED_05N . _RED_45 . _CLEARED_45 . _ALTERNATE_05N) . _METER_ + E_METER_

S_VEHICLE_DEPARTED_05                               .= S_DK_052._MIN_GREEN_05
S_CLEARED_05                                        .= S(RED_TIMER_05=CLEARANCE_TIME_05)

//END
```