

PARAMICS Plugin Document – Actuated signal Control

Lianyu Chu, Henry X. Liu, Brian Smolke, Will Recker

PATH ATMS Center
University of California, Irvine

Plugin Compatibility: V3/V4
Release date: 12/23/2003

522 Social Science Tower
Irvine, CA 92697-3600
URL: <http://www.its.uci.edu/>



Table of Contents

Table of Contents	2
1. Introduction	3
2 Plugin implementation	4
2.1 Control logic	4
2.2 Modeling vehicle detection	5
2.3 Pseudo code	7
3. Step-by-step user manual	9
3.1 Data preparation	9
3.2 Adding detectors and checking network coding	9
3.3 Preparation of worksheet	9
3.4 Preparation of “signal_control” file	11
3.5 Preparation of “priorities” information	11
3.6 Loading plugin	14
3.7 Error checking	14
3.8 Exercises	14
4. Working with different phasing sequences	16
4.1 Lead-leg left-turns	16
4.2 Through movement first	18
4.3 How to split phases using the Actuated Signal Plugin	20
5. PROGRAMMER capabilities	25
5.1 Interface functions	25
5.2 How to use interface functions in other plugins	26
6 Technical Supports	27
6.1 Limitations of this plugin	27
6.2 FAQ:	27
6.3 Tools	28
6.4 Release notes	28
6.5 Future development	28
6.6 Contact information	28
6.7 References	28
APPENDIX 1 Worksheet	30
APPENDIX 2 Signal Timing Chart	31
APPENDIX 3 Geometric layout of the intersection	33
APPENDIX 4 Completed worksheet for the example intersection	34
APPENDIX 5 The priorities information for the example intersection	35
APPENDIX 6 Loading plugins in PARAMICS	38

1. Introduction

Generally, modes of traffic signal operation can be divided into three primary categories (USDOT, 1996): pre-timed, actuated and traffic responsive. PARAMICS can basically model the fixed-time signal control. Besides, PARAMICS also provides a plan/phase language (i.e. a kind of script language) to simulate some simple actuated signal control logic. However, in the field the widely used actuated signal controller uses the complex NEMA logic or type-170 logic. Our experiences found this script language is difficult to be used to model these types of complex control schemes and to replicate these schemes to multiple signalized intersections.

A plugin was created to more easily model actuated signal control within PARAMICS. This report discusses the logic of this plugin as well as its implementation.

2 Plugin implementation

2.1 Control logic

The layout of a typical actuated signal intersection is shown in Figure 1.

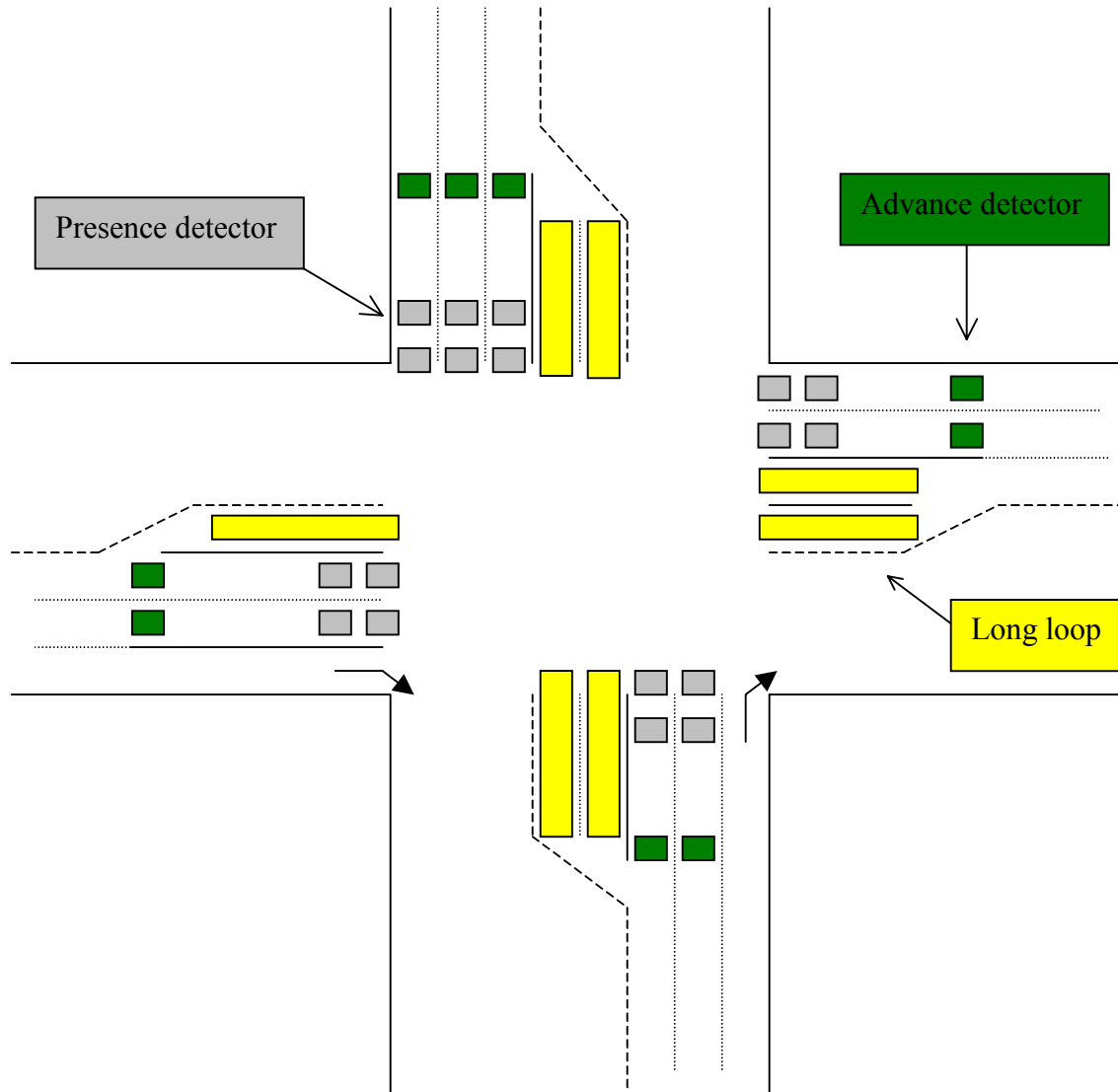


Figure 1 Typical Intersection Layout

The control logic that is implemented in the plugin is for an eight-phase, dual-ring, concurrent controller actuated signal. The dual-ring, concurrent concept is illustrated briefly in Figure 2. Note that eight phases are shown, each of which accommodates one of the through or left turning movements. A “barrier” separates the north-south phases from the east-west phases. Any phase in the top group (Ring 1) may be displayed with any phase in the bottom group (Ring 2) on the same side of the barriers without

introducing any traffic conflicts. For simplicity, the right turns are omitted and assumed to proceed with the through movements.

In fully-actuated signal control, all phases at an intersection are actuated. Therefore the length of each phase, and consequently the cycle length, will vary with each cycle. Some phases may be skipped if there is no vehicle actuation. To simulate the real controller better, the order and sequence of phases can also be altered. The detailed description on how actuated signal works can be found in the textbook by McShane et al (1998).

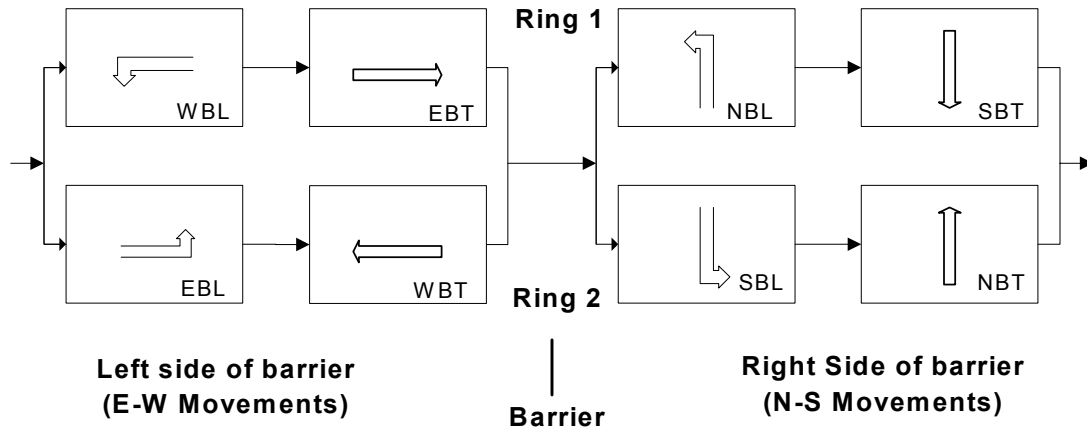


Figure 2 Dual-ring concurrent phasing scheme with assigned movements

2.2 Modeling vehicle detection

The vehicle detection is an important part of the actuated signal system. There are three groups of detectors in each approach for the typical intersection in the real world:

- (1) Stopline detectors, located in the through lanes and very close to the stop line, for the presence detection of through vehicles. There may be 2-3 presence detectors for a lane that are typically about six feet by six feet in size;
- (2) Advance loop detector, located at almost 150—300 feet from the stop bar, used to detect vehicles for the extension of the through movement phase; and
- (3) Long loop detector for left turns, with the length of about 50-70 feet, for the presence detection of left turn vehicles. In some cases a set of individual detectors are used instead of a single long one.

For some intersections, there may be no advance detector at some approaches of an intersection. If presence detectors are only placed on the minor cross street, the signal has semi-actuated control.

To better simulate the functionality of detectors, ideally detectors should be modeled in PARAMICS according to the real-world configuration. However, in Build 3 of PARAMICS, detectors are not lane specific. A detector covers all lanes of a link and thus a PARAMICS detector represents a detector station. Therefore, we cannot model a

separate long loop (for left turn use) in the actuated signal system. As a result, we use three small detectors instead of a long loop, as shown in Figure 3.

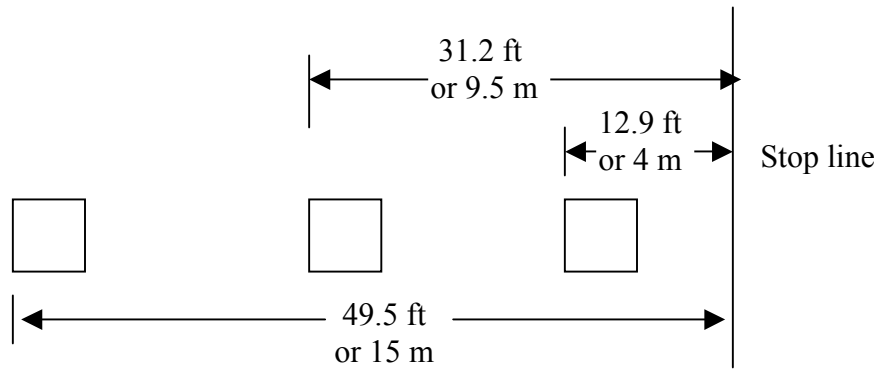


Figure3 Modeling the left turn long loop detector

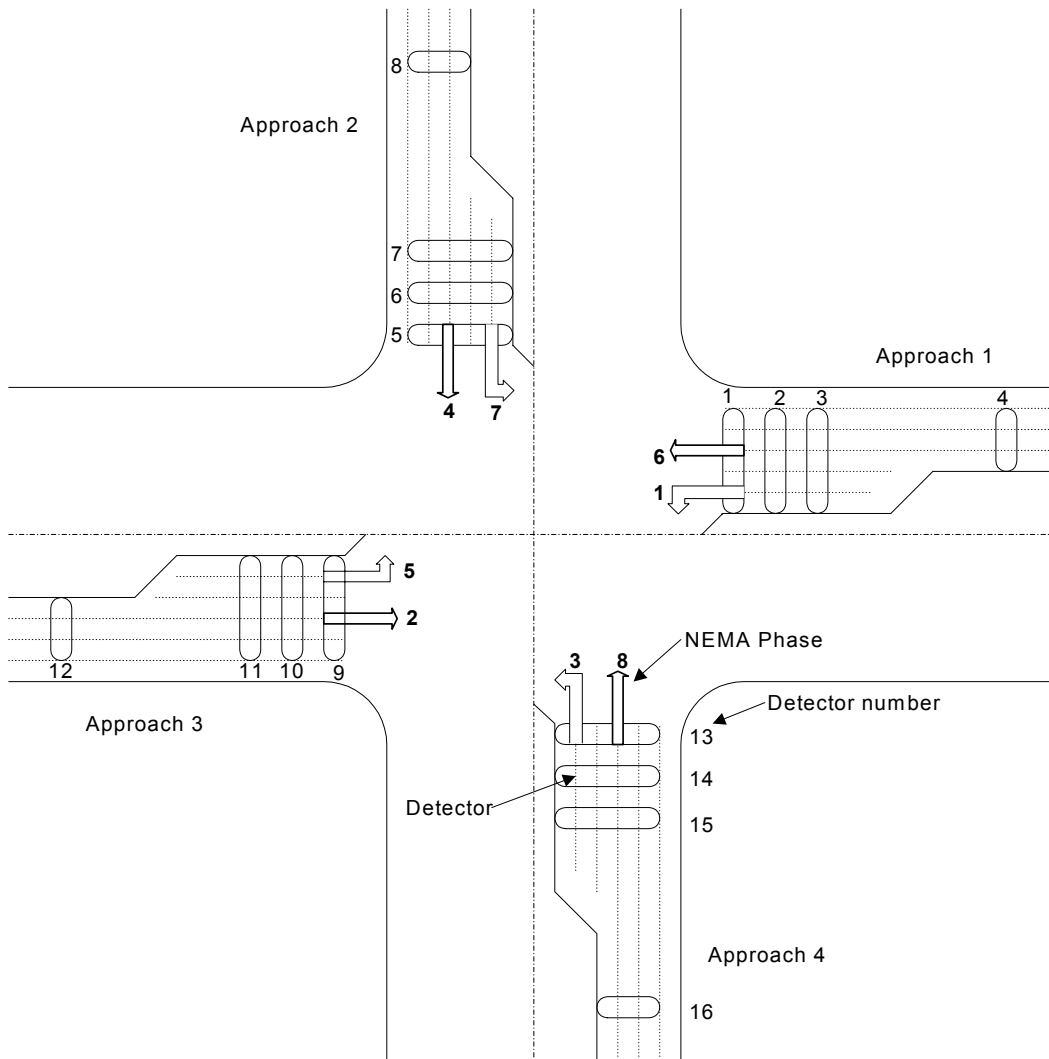


Figure 4. Typical Intersection Layout in PARAMICS with NEMA phases

Three 2 m or 6.6 ft detectors are used to mimic one 50ft long loop detector. These detectors model the stopline presence detectors as well as the left-turn detectors. The default length of detectors in PARAMICS is 2 meters, or 6.6 feet. The lengths of these detectors in PARAMICS do not match the common real-world length of six feet, but for the purposes of simulation this works fine.

As illustrated in Figure 2, we modeled 16 detectors for a typical intersection in PARAMICS and each detector covers all lanes of a link. For each approach, there are three detectors close to the stop line for through and left-turn vehicle presence detection, and one advance detector located at about 150-300 feet to the stop line for detecting vehicles for the extension of the through movement phase. For stopline detectors, all three of them employ the vehicle presence of left turn lanes; the two detectors close to the stop line are used for detecting the presence of through vehicles.

Due to improvements in the long loop detection in a later versions of PARAMICS (later than Build V.3.0.7), we can use one long loop instead of three stopline loop detectors for vehicle presence. As a result, we only need to model 8 detectors for an intersection. That is to say, detectors 1, 5, 9 and 13 are long loops (with a typical length of 50 feet), and there is no need to code detectors 2, 3, 6, 7, 10, 11, 14, and 15. This is our recommended method to model detectors of an actuated signal intersection.

In version 4 of PARAMICS, detectors can be lane specific. This plugin does not support the use of this type of detector.

2.3 Pseudo code

The pseudo code for the main control logic of this plugin is given as follows:

1. Actuated Signal plugin set up using `api_setup()`, includes signal data input, memory allocation, and initial signal phase set up.
2. At every time step, `net_action` is called:
 - For controller intersection = 1 : n {
 - a. Inquiry the current signal information using `signal_inquiry()`.
 - b. If (left green time == 0) {
 - Amber and red time are counted.
 - If (amber and red time are reached)
 - Set the next signal phase parameters through `signal_action()`.
 - }
 - else {
 - vehicle presence detection (`pp_presence_detection()`).
 - execute the current signal plan (`pp_execute_plan()`) {
 - If (left green time < extension &&
 - vehicle presence for extension &&
 - expired green < (maximal green – extension)) {
 - green time increased by (extension – left green).

```
    }  
    If ( left green time <= time step )  
        Find the next phase by vehicle presence  
    }  
}
```


3. Step-by-step user manual

3.1 Data preparation

The data input to this plugin is the signal timing plan, the geometry and detector information of actuated signal intersections.

If the purpose of simulation is to model a real-world network, the following information is required in order to make actuated signals:

- (1) Signal Timing Chart obtained from the proper government agency
- (2) Geometric layout of the intersection; the best source of this information is usually from as-built plans.

If the purpose of simulation is to evaluate an intersection design (or, test signal timing plans), you can obtain the signal timing from traffic signal software, such as SYNCHRO, based on historical traffic patterns.

3.2 Adding detectors and checking network coding

Based on the discussion in Section 2.2, we can either code 16 detectors or 8 detectors to a four-legged actuated signal intersection. The exact set-back distance of the advance detector can found in the “Geometric layout of the intersection”.

The following geometric information needs to be checked:

- (1) Number of lanes for each approach;
- (2) Lane use information at intersections (for example, at an approach of an intersection, which lanes are assigned to the left turn, through, or right turn movements). If the default lane configuration is not the same as that shown in the “Geometric layout of the intersection”, the corresponding intersection needs to be re-coded via the PARMICS Modeller GUI (Node->Modify junction) or by editing the “junctions” file manually.

3.3 Preparation of worksheet

Running MODELLER, zoom in to the intersection. Fill out a worksheet that includes geometry and signal timing information of the intersection. The worksheet has been attached in APPENDIX 1 of this document.

The following is a list of necessary information in the worksheet:

- (1) Write down the name of the intersection, i.e. Alton & ICD, and the signal ID that is shown in the first page of signal timing chart.

- (2) Write down the two street names, the direction, and the PARAMICS designation of the junction node and the four adjacent nodes on four approaches.
- (3) Find NEMA movement number 1, generally a left turn, from the Signal Timing Chart. Write down the turn arrow and the movement number 1. As a result, all NEMA movements / phases can be determined based on the definition of the standard NEMA phases / movements shown in figure 1. Write down all NEMA movements on the worksheet.
- (4) Write down the approach number on the worksheet. The approach that the 1st NEMA movement locates is defined as approach 1 here. The counter-clockwise approaches around the junction are defined as approach 2, 3, 4.;
- (5) Fill out the 3-5 rows (Initial green, Extension, Max green) of the table on the bottom of the worksheet. The ini_green corresponds to the “Initial”(green time) and the max_green corresponds to the “Max Green” in the Signal Timing Chart.
- (6) Find out the recall movement from “Signal Timing Chart”. Enter the two recall movement numbers into the first two columns of the “recall” row;
- (7) Find out how many lanes correspond to each NEMA movement from “layout of the intersection” or PARAMICS environment. Fill them in the row of “lanes” in the worksheet. The first value in the row corresponds to the number of lanes for NEMA movement 1 and the second value corresponds to NEMA movement 2, etc. In many situations there are lanes that are shared by different movements. For example, one lane may allow both left turning and through vehicles to pass. In this case, the lane will count both as one through lane and as one half (0.5) of a left-turning lane.
- (8) From the layout, find out how many right turn lanes for each approach (1 -> 4). Please refer to the definition at step 4 for the definition of approaches 1 to 4. Write down these numbers in the row of “Right-turn lanes”. As in the case of lanes that allow both left and through movements, lanes that allow through and right-turn movements will count as one through lane and one half of a right-turning lane.
- (9) The row of “detector 1” to “detector 4” should be filled with the name of detectors (the sequence is from stopline detectors to the advance detector, seen in figure 1) on “approach 1” to “approach 4”. Please refer to the definition at step 4 for the definition of approaches 1 to 4. In some cases, one or more of the detectors for an approach does not need to be modeled. Each missing detector needs to be specified as “N/A” in the worksheet. In Paramics v3.0 build 6, it was necessary to place three separate detectors at the stopline to ensure proper detection. However, build 7 of Paramics 3.0 and all later versions only need one long detector. To allow reverse compatibility, it still might be desirable to place three separate detectors.

3.4 Preparation of “signal_control” file

The plugin requires a file titled “signal_control” to be in the PARAMICS network directory. An example of the “signal_control” file is shown in Figure 5.

The first line of this file specifies the number of actuated signals modeled in the network. The remainder of the file contains the signal timing information. The information in this file has a very similar format to that of the worksheet. There are two signals modeled in Figure 5. The first one uses 16 detectors and the second used 8 detectors.

```
total number of actuated signals is: 2

node 1167 ICD & BARRANCA
movements 1 2 3 4 5 6 7 8
ini_green 5 5 5 8 5 5 5 8
extension 3 4 3 5 3 4 3 5
max_green 24 32 24 32 24 32 24 32
recall 4 8
lanes 2.0 2.0 2.0 3.0 2.0 2.0 2.0 3.0
rightturn 1.0 1.0 1.0 1.0
detector1 icbsw icb2w icb3w icbuw
detector2 icbss icb2s icb3s icbus
detector3 icbse icb2e icb3e icbue
detector4 icbsn icb2n icb3n icbun

node 147 BARRANCA & SAND CNYN
movements 1 2 3 4 5 6 7 8
ini_green 5 8 5 8 5 8 5 8
extension 2 5 2 5 2 5 2 5
max_green 20 24 20 32 20 24 20 32
recall 2 6
lanes 2.0 2.0 2.0 2.0 2.0 2.0 2.0 3.0
rightturn 1.0 0.5 1.0 1.0
detector1 bscsw N/A N/A bscuw
detector2 bscss N/A N/A bscus
detector3 bscse N/A N/A bscue
detector4 bscsn N/A N/A bscun
...
```

3.5 Preparation of “priorities” information

The “priorities” file defines what movement can be allowed under each phase of an intersection. For pre-timed signal control, the priorities information can be edited through

the PARAMICS GUI. However, for the actuated signal, the file “priorities” must be edited directly with a text editor.

We need to generate the “priorities” information of an actuated signalized intersection based on the worksheet we made on step 2, in which the node names of adjacent nodes of an intersection have been written down. Figure 6 is an example of the node designations for a four-legged intersection. “approach 1” is considered to be in the direction starting at node 7511 and heading towards the junction node 528z.

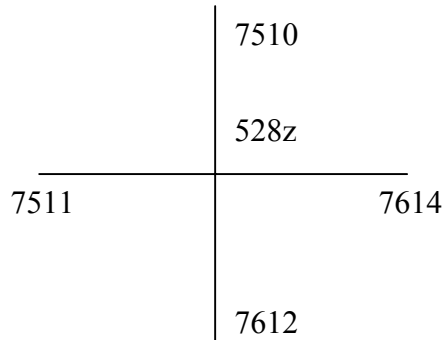


Figure 6. Intersection Layout

The “priorities” for a four-legged full-actuated intersection will have eight phases. As illustrated in Figure 7, “Phase 1” will correspond to the situation where the left-turning NEMA movements 1 and 5 will be given the green. “Phase 2” will account for the situation where movements 5 and 2 will be given the green, and “phase 3” will be for movements 1 and 6. “Phase 4” will be for the through movements 2 and 6. The last four phases will follow the pattern of the first four phases, starting with the left-turn movements 3 and 7.

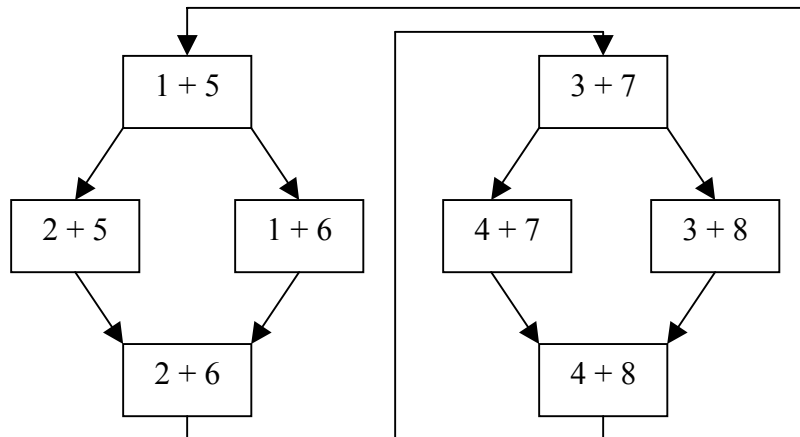


Figure 7. Eight phases of the four-legged full-actuated signal intersection

For the intersection in the previous figure, the definition of phases and actions (movements) in “priorities” file would be:

```
actions 528z
phase offset 0.00 sec
phase 1
    0.00
    max 100.00
red phase 0.00
fill
all barred except
from 7510 to 7511 minor
from 7511 to 7612 minor
from 7511 to 7510 major
from 7612 to 7614 minor
from 7614 to 7612 major
from 7614 to 7510 minor
phase 2
    0.00
    max 100.00
red phase 0.00
fill
all barred except
from 7510 to 7511 minor
from 7511 to 7612 minor
from 7511 to 7614 major
from 7511 to 7510 major
from 7612 to 7614 minor
from 7614 to 7510 minor
phase 3
...

phase 8
    0.00
    max 100.00
red phase 0.00
fill
all barred except
from 7510 to 7511 minor
from 7510 to 7612 major
from 7511 to 7612 minor
from 7612 to 7614 minor
from 7612 to 7510 major
from 7614 to 7510 minor
```

In this example, the movements of each phase are “major” while all right turns are “minor”. We set the default signal time of each phase as 0 sec (This is the reason that we

cannot edit these “actions” information through GUI). The plugin will assign a certain length of time to each phase based on the presence of vehicles.

Then, update the above priorities information of the corresponding signalized node in the “priorities” file of the network.

Please note that the network with modified “priorities” file must use together with this actuated signal plugin. Without this plugin, all movements of those actuated signal intersections are in red light.

3.6 Loading plugin

After the completion of the “signal_control” file and the update the “priorities” file, you can load the simulation network together with this plugin. The names of this plugin files are:

actuated_signal.dll: Modeller Plugin
actuated_signal -p.dll: Processor Plugin

Run simulation and then you will see that this plugin is used to emulate the actuated signal control at specified intersections via GUI.

3.7 Error checking

If there is any mistake happened in the “signal_control” file, the plugin will be disabled. The report window of PARAMICS will show whether this plugin is working. This plugin generates a file named “Log-signal.txt” under the network directory, which can be used to check if the “signal_control” file has been understood by this plugin correctly.

The detector information in the “signal_control” file is connected with the “priorities” information of the signal intersection. The mismatch of them may cause the signal work abnormally. Two methods can be used to judge if the actuated signal control has the correct logic:

- (1) Based on the observation from GUI (Node->Modify junction->Signal display), or
- (2) Making a long time simulation run and then check if there are any serious congestion happened at actuated signal intersections. If an actuated signal control is not working correctly, all input files need to be double checked for any mistakes.

The correct use of this plugin depends on your knowledge of signal control. If necessary, please have a look at related chapters in the textbooks listed in Section 5.7.

3.8 Exercises

APPENDIX 2, 3 shows the “Signal Timing Chart” and “Geometric layout of the intersection ICD & BARRANCA”. Based on Section 2.2, we filled in the worksheet, shown in APPENDIX 4. Based on this worksheet, the “signal_control” information is shown in Section 3.4. Its “priorities” information is shown in APPENDIX 5.

This plugin can be used to model more complex actuated signal control through proper configurations of the “priorities” and “signal_control” information. Users can learn more from one of our example Irvine networks, which includes 37 actuated signals.

4. Working with different phasing sequences

In dual-ring operation, full-actuated signal controllers are capable of a number of phase sequences between barriers. For each of the two major phase groups, there are three basic phase sequences:

1. Left-turn first
2. Lead-leg left-turns, and
3. Through movement first

The developed full-actuated signal plugin can work with all three sequences. We have described how to work with the “left-turn first” case in the previous section. This section will discuss how to make the plugin to work under the second and third phase sequences. Please refer to the example networks for further understanding this section.

4.1 Lead-leg left-turns

The layout of a typical intersection is as shown in Figure 8.

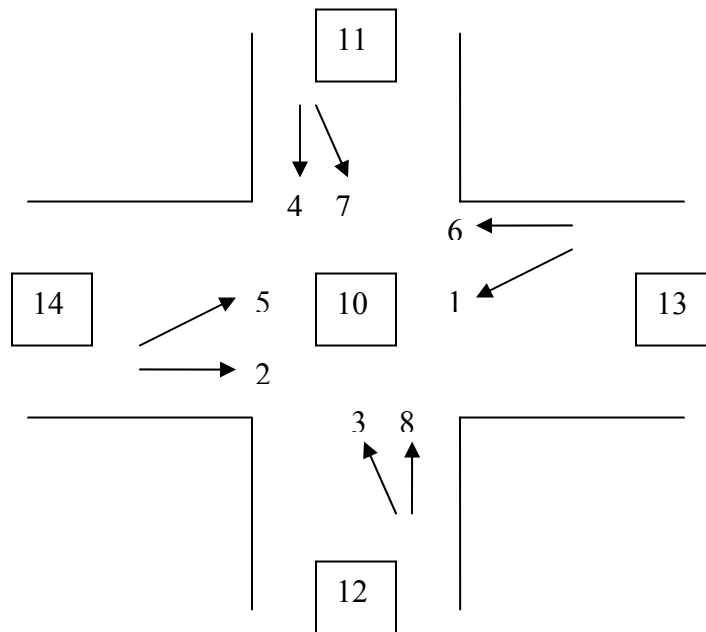


Figure 8 Phase layout of a signalized intersection

In the `signal_control` file, the two phases on the lead leg need to be put to the columns of movement 1 and movement 5. If we want to make link 14:10 as the lead leg, the phase sequence will be $2\&5 \rightarrow 1\&5 \rightarrow 2\&6 \rightarrow 1\&6^i$, the corresponding `signal_control` file needs to be configured as follows.

ⁱ The real-world controller may not have the phase combination of 1 & 5. Our plugin cannot avoid having it. But its existence does not have any negative (but positive) influence on the operation of the control logic.

Movements	2	1	3	4	5	6	7	8
ini_green	10	5	5	8	5	10	5	8
extension	4	3	3	5	3	4	3	5
max_green	32	24	24	32	24	32	24	32
recall	4	8						
lanes	2	2	2	3	2	2	2	3
rightturn	1	1	1	1				
detector1	icbsw	N/A	N/A	icbuw				
detector2	icbss	N/A	N/A	icbus				
detector3	icbse	N/A	N/A	icbue				
detector4	icbsn	N/A	N/A	icbun				

Based on phase sequences, 2&5 -> 1&5 -> 2&6 ->1&6, the priorities file needs to put 2&5 to phase 1, 1&5 to phase 2, 2&6 to phase 3 and 1& 6 to phase 4.

```

actions 10
phase offset 0.00 sec
phase 1
    0.00
    max 100.00
red phase 0.00
fill
all barred except
from 14 to 11 major
from 14 to 12 major
from 14 to 13 major
from 11 to 14 minor
from 13 to 11 minor
from 12 to 13 minor
phase 2
    0.00
    max 100.00
red phase 0.00
fill
all barred except
from 13 to 12 major
from 14 to 11 major
from 11 to 14 minor
from 13 to 11 minor
from 14 to 12 minor
from 12 to 13 minor
phase 3
    0.00
    max 100.00
red phase 0.00
fill
all barred except

```

```

from 13 to 14 major
from 13 to 11 major
from 14 to 13 major
from 14 to 12 major
from 11 to 14 minor
from 12 to 13 minor
phase 4
    0.00
    max 100.00
red phase 0.00
fill
all barred except
from 13 to 14 major
from 13 to 12 major
from 13 to 11 major
from 11 to 14 minor
from 14 to 12 minor
from 12 to 13 minor
phase 5
    0.00
    max 100.00
red phase 4.00
fill
all barred except
...

```

If we want link 13:10 as the lead leg, “signal_control” will be:

movements	1	2	3	4	6	5	7	8
ini_green	5	10	5	8	10	5	5	8
extension	3	4	3	5	4	3	3	5
max_green	24	32	24	32	32	24	24	32
recall	4	8						
lanes	2	2	2	3	2	2	2	3
rightturn	1	1	1	1				
detector1	icbsw	N/A	N/A	icbuw				
detector2	icbss	N/A	N/A	icbus				
detector3	icbse	N/A	N/A	icbue				
detector4	icbsn	N/A	N/A	icbun				

The corresponding priorities file will not be listed here. Users can easily figure out.

4.2 Through movement first

Based on the description of the last section, we can deduce that the phase 2 and 6 should be put to the location of the columns of movement 1 and movement 5.

As shown in the below figure, if we want phases 2 and 6 go first, the following phases will be 2&6 -> 1&6 -> 2&5 -> 1&5. The signal_control file should be:

movements	2	1	3	4	6	5	7	8
ini_green	10	5	5	8	10	5	5	8
extension	4	3	3	5	4	3	3	5
max_green	32	24	24	32	32	24	24	32
recall	4	8						
lanes	2	2	2	3	2	2	2	3
rightturn	1	1	1	1				
detector1	icbsw	N/A	N/A	icbuw				
detector2	icbss	N/A	N/A	icbus				
detector3	icbse	N/A	N/A	icbue				
detector4	icbsn	N/A	N/A	icbun				

For the “priorities” file, we can just put 2&6, 1&6, 2&5, and 1&5 to the phase 1, 2, 3 and 4, as shown below.

```

actions 10
phase offset 0.00 sec
phase 1
    0.00
    max 100.00
red phase 0.00
fill
all barred except
from 13 to 11 major
from 13 to 14 major
from 14 to 12 major
from 14 to 13 major
from 11 to 14 minor
from 12 to 13 minor
phase 2
    0.00
    max 100.00
red phase 0.00
fill
all barred except
from 13 to 11 major
from 13 to 14 major
from 13 to 12 major
from 11 to 14 minor
from 14 to 12 minor
from 12 to 13 minor
phase 3
    0.00
    max 100.00

```

```

red phase 0.00
fill
all barred except
from 14 to 11 major
from 14 to 12 major
from 14 to 13 major
from 11 to 14 minor
from 13 to 11 minor
from 12 to 13 minor
phase 4
    0.00
    max 100.00
red phase 0.00
fill
all barred except
from 13 to 12 major
from 14 to 11 major
from 11 to 14 minor
from 13 to 11 minor
from 14 to 12 minor
from 12 to 13 minor
phase 5
    0.00
    max 100.00
red phase 4.00
fill
all barred except
...

```

4.3 Split phases

Except the above-mentioned cases, users may need to make signals work under split phases. For example, there are only two possible phase combinations, 1&6 and 2&5, in the first phase group. Under this situation, the signal timing chart from the local transportation agency may provide phase information, which may not make this plugin work as expected.

4.3.1 2&5 first and 1&6 second

The “signal_control” file can be configured as:

movements	2	1	3	4	9	9	7	8
ini_green	10	5	5	8	0	0	5	8
extension	4	3	3	5	0	0	3	5
max_green	32	24	24	32	0	0	24	32
recall	4	8						

lanes	5	5	2	3	0	0	2	3
rightturn	0	1	0	1				
detector1	icbsw	N/A	N/A	icbuw				
detector2	icbss	N/A	N/A	icbus				
detector3	icbse	N/A	N/A	icbue				
detector4	icbsn	N/A	N/A	icbun				

There is no phase 5 and 6. Only phases 1 and 2 exist. Note that movement (i.e. phase) 1 includes all lanes of link 13:10 and phase 2 includes all lanes of link 14:10 no matter the lane is reserved for left turns or through movements. We can also configure the “signal_control” file in another way, i.e. without phases 1 and 2 but with phases 5 and 6, as shown below. The previous phase 1 goes to phase 6 and the previous phase 2 goes to phase 5.

movements	9	9	3	4	5	6	7	8
ini_green	0	0	5	8	10	5	5	8
extension	0	0	3	5	4	3	3	5
max_green	0	0	24	32	32	24	24	32
recall	4	8						
lanes	0	0	2	3	5	5	2	3
rightturn	0	1	0	1				
detector1	icbsw	N/A	N/A	icbuw				
detector2	icbss	N/A	N/A	icbus				
detector3	icbse	N/A	N/A	icbue				
detector4	icbsn	N/A	N/A	icbun				

For the “priorities” file, phase 1, 2 and 3 have the same allowed movements.

```

actions 10
phase offset 0.00 sec
phase 1
    0.00
    max 100.00
red phase 0.00
fill
all barred except
from 14 to 11 major
from 14 to 12 major
from 14 to 13 major
from 11 to 14 minor
from 13 to 11 minor
from 12 to 13 minor
phase 2
    0.00
    max 100.00
red phase 0.00
fill

```

```

all barred except
from 14 to 11 major
from 14 to 12 major
from 14 to 13 major
from 11 to 14 minor
from 13 to 11 minor
from 12 to 13 minor
phase 3
    0.00
    max 100.00
red phase 0.00
fill
all barred except
from 14 to 11 major
from 14 to 12 major
from 14 to 13 major
from 11 to 14 minor
from 13 to 11 minor
from 12 to 13 minor
phase 4
    0.00
    max 100.00
red phase 0.00
fill
all barred except
from 13 to 14 major
from 13 to 12 major
from 13 to 11 major
from 11 to 14 minor
from 14 to 12 minor
from 12 to 13 minor
phase 5
    0.00
    max 100.00
red phase 0.00
fill
all barred except
...

```

4.3.2 1&6 first and 2&5 second

The “signal_control” file needs to be one of the following two:

movements	1	2	3	4	9	9	7	8
ini_green	5	10	5	8	0	0	5	8
extension	3	4	3	5	0	0	3	5
max_green	24	32	24	32	0	0	24	32

recall	4	8						
lanes	5	5	2	3	0	0	2	3
rightturn	0	1	0	1				
detector1	icbsw	N/A	N/A	icbuw				
detector2	icbss	N/A	N/A	icbus				
detector3	icbse	N/A	N/A	icbue				
detector4	icbsn	N/A	N/A	icbun				
movements	9	9	3	4	6	5	7	8
ini_green	0	0	5	8	5	10	5	8
extension	0	0	3	5	3	4	3	5
max_green	0	0	24	32	24	32	24	32
recall	4	8						
lanes	0	0	2	3	4	5	2	3
rightturn	0	1	0	1				
detector1	icbsw	N/A	N/A	icbuw				
detector2	icbss	N/A	N/A	icbus				
detector3	icbse	N/A	N/A	icbue				
detector4	icbsn	N/A	N/A	icbun				

The corresponding “priorities” file is:

```

actions 10
phase offset 0.00 sec
phase 1
    0.00
    max 100.00
red phase 0.00
fill
all barred except
from 13 to 14 major
from 13 to 12 major
from 13 to 11 major
from 11 to 14 minor
from 14 to 12 minor
from 12 to 13 minor
phase 2
    0.00
    max 100.00
red phase 0.00
fill
all barred except
from 13 to 14 major
from 13 to 12 major
from 13 to 11 major
from 11 to 14 minor
from 14 to 12 minor

```

from 12 to 13 minor
phase 3

0.00

max 100.00

red phase 0.00

fill

all barred except

from 13 to 14 major

from 13 to 12 major

from 13 to 11 major

from 11 to 14 minor

from 14 to 12 minor

from 12 to 13 minor

phase 4

0.00

max 100.00

red phase 0.00

fill

all barred except

from 14 to 11 major

from 14 to 12 major

from 14 to 13 major

from 11 to 14 minor

from 13 to 11 minor

from 12 to 13 minor

phase 5

0.00

max 100.00

red phase 0.00

fill

all barred except

...

5. PROGRAMMER capabilities

5.1 Interface functions

Interface functions have been provided by this plugin for external modules to acquire and change the default timing plan. This plugin provided a couple of interface functions for external plugin modules to acquire the current signal timing plan and set a new timing plan to a specific signal. An advanced signal control algorithm plugin can be further developed based on them. The prototypes of these interface functions are shown below.

Signal* uci_signal_get_parameters(char *nodeName);

Function: Querying the current signal timing plan of a specific actuated signal

Return Value: The current timing plan of an actuated signal.

Parameters: **nodeName** is the name of the signal node.

Signal is the structure of actuated signal data, whose definition is:

```
type Signal
{
    // intersection name and location
    char *node;
    char *controllerLocation;

    // signal parameters
    int movements[8];
    float maximumGreen[8];
    float minimumGreen[8];
    float extension[8];
    float storedRed[8];
    float phaseGreenTime[8];
    float movementGreenTime[8];

    // current phase information
    int currentPhase;
    int expiredTime;
    float redTimeLeft;
    Bool cycleEndFlag;
}
```

Void uci_signal_set_parameters(Signal *sig);

Function: Setting a new timing plan to a specific signal.

Return Value: None

Parameters: **sig** stores the new timing plan.

5.2 How to use interface functions in other plugins

These two interface functions can be called in other plugins. The following setting is required:

- (1) In the workspace of your plugin that wants to use these interface functions, specify the library file “actuated_signal.lib” of the actuated signal plugin as an input object/library module. The path of “actuated_signal.lib” should be specified as well.
- (2) Specify the prototype of the interface function at the beginning of your plugin as follows:

```
_declspec(dllexport) void uci_signal_set_parameters(Signal *sig);  
_declspec(dllexport) Signal* uci_signal_get_parameters(char *nodeName);
```

6 Technical Supports

6.1 Limitations of this plugin

1) During our development on this full-actuated signal control plugin, we found that PARAMICS did not provide a plugin function for users to control the amber time (yellow light). Although yellow time can be set in the configuration file, it is a universal parameter for all the intersections and all the time. It is not convenient in the actuated signal case since some phases may be skipped (the amber time has to be skipped at the same time). In order to simulate the real world better, our developed plugins have to have a handle on the control of the amber time associated with each phase.

2) In PARAMICS, phase and movement are different. For the current actuated signal plugin implementation, each phase usually includes two major movements, and some minor movements. For instance, phase 1 may include dual left turn movements, and some right turn minor movements. PARAMICS runs through phase 1 to phase 8, some phases may be skipped depending on the vehicle presence. However, each movement has its own initial green and extension in the signal-timing sheet. Only one set of parameters could be used in each phase. Although a reasonable set of parameters is calculated and used during the simulation, and doing this does not hurt the simulation performance, the actual signal control cannot be fully simulated in this plugin. Ideally, we want each phase to include only one major movement, and two phases can be executed at the same time. Version 4 of PARAMICS provides users with this capability but we do not have time to implement this at the current time.

3) Only one timing plan for each intersection is supported by the current plugin. In order to support multiple signal plans, please use another plugin “multiple actuated signal plan” together with this plugin.

4) In version 3 of PARAMICS, vehicles may stop at stop lines because of routing problem (such as a through vehicle stopping on a left turn lane). Version 4 has bot this problem because it introduces the re-routing feature.

6.2 FAQ:

1. Grammar of input files

Unlike the parser system of PARAMICS, which allow flexible grammars and comments (i.e. ##), the format of the input file of this plugin is rigid and thus any problem in the file may cause the plugin not work well. Our recommendation for users is that the input file of the example network of this plugin is a good starting point to make your own input file in order to avoid editing problems.

2. Can a phase in priorities file have no movement information?

It is not good for a phase to have no movement information. Every phase corresponds to a combination of NEMA phases, if that phase is regarded to have vehicles and then a green signal will be given to that phase, which has no movement allowed. Then, the plugin may be locked to that phase. The solution is that you can repeat the movement information of a related phase. Please refer to Section 4.3.

6.3 Tools

In order to speed up the process of coding actuated signals, we also make two computer programs for the making of “signal_control” file and the “priorities” information. You can request these tools from PATH ATMS center.

6.4 Release notes

Compare to the plugin used in PARAMICS version 3, this plugin has the following modification:

1. “Log-signal.txt” is generated under the network directory after this plugin is loaded by PARAMICS. This file is used to check if the “signal_control” file includes no invalid input and has been understood by this plugin correctly.

6.5 Future development

We plan to integrate the signal optimization software, Synchro, into PARAMICS. Several plugins will be developed to convert a PARAMICS network into a Synchro network, exchange data between PARAMICS and Synchro, and interactively optimize signal timing plans as PARAMICS simulation is running in the front.

6.6 Contact information

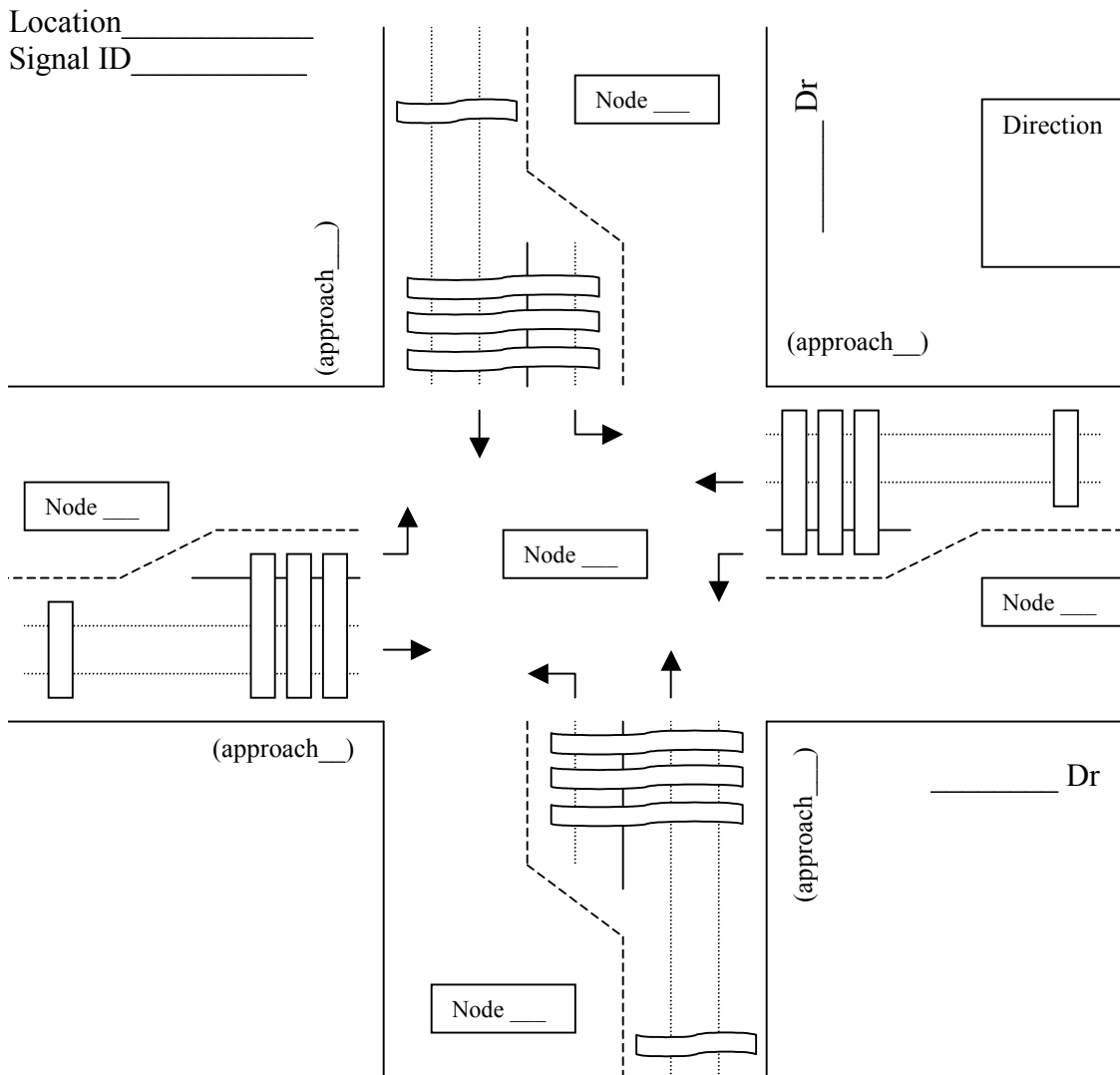
Any comments and suggestions are welcome. Please contact us at the email address: lchu@translab.its.uci.edu.

6.7 References

- (1) W.R. McShane, R.P. Roess and E.E. Prassas (1998). Traffic Engineering (Second Edition). Prentice-Hall.
- (2) Liu, X., Chu, L., and Recker, W. (2001) “*Paramics API Design Document for Actuated Signal, Signal Coordination and Ramp Control*”, California PATH Working Paper, UCB-ITS-PWP-2001-11, University of California at Berkeley.

- (3) USDOT, Federal Highway Administration (1996) Traffic Control Systems Handbook.

APPENDIX 1 Worksheet



Node								
Movement	1	2	3	4	5	6	7	8
Initial Green								
Extension								
Max Green								
Recall Phase								
Lanes								
Right-Turn lanes								
Detector 1								
Detector 2								
Detector 3								
Detector 4								

APPENDIX 2 Signal Timing Chart

Signal Timing Chart For Controller Type: 820 A

City Of Irvine - Traffic Operations

Signal ID: 193
ICU # 234

Address # 2
Interconnect: MUX # 59

Supersedes: 12/26/90

Designed By: C. N.
Approved By: B. G.
Installed By: STAR

Command Key: VMS
Master Location: ITRAC
Prom. Revision

Page 1 of 4

City Of Irvine

Signal Turn On: 12/5/90
Turn On By: A.T.

SEQUENCES:

West LT	East & Ped	North LT	South & Ped
←	→	↶	⇒
East LT	West & Ped	South LT	North & Ped
↷	←	↵	⇐

Phase	Vehicular Movement	Flash
1.	W/B Left Turn - Bairanca	R
2.	E/B Thru - Barranta	R
3.	N/B Left Turn - Irvine Center Drive	R
4.	S/B Thru - Irvine Center Drive	R
5.	E/B Left Turn - Barranta	R
6.	W/B Thru - Barranta	R
7.	S/B Left Turn - Irvine Center Drive	R
8.	N/B Thru - Irvine Center Drive	R

Overlap	Vehicular Movement	Flash
1.		
2.		
3.		
4.		

Notes and Comments:

- Set time and cab flash monitor @ 120v, press MISC key. See BIKE TIMING on page 2, (Free Parameters)
-
-
-
-

Detection Amplifier Setup:

NB-ADV: 1.5 sec	EXT. +	DEL.	NB-THRU: DEL.	NB-LT: DEL.	NB-RT: DEL.
SB-ADV: 1.5 sec	EXT. +	DEL.	SB-THRU: DEL.	SB-LT: DEL.	SB-RT: DEL.
EB-ADV: 1.5 sec	EXT. +	DEL.	EB-THRU: DEL.	EB-LT: DEL.	EB-RT: DEL.
WB-ADV: 1.5 sec	EXT. +	DEL.	WB-THRU: DEL.	WB-LT: DEL.	WB-RT: DEL.

LOCATION: IRVINE CENTER DR @ BARRANCA

SIGNAL ID: 193 TIMING CHANGE : 1/19/00 SUPERSEDES: 5/24/99

SET UP

INTERSECTION PHASING

Allowable Phases: 1 2 3 4 5 6 7 8

Flashing Walks: _____

Exclusive Phases: _____

Ped Phases: - 2 - 4 - 6 - 8

Density Phases: _____

START UP

Start In: RED

Start Up Time: 4.0

Start Up Phases: - 2 - 4 - 6 - 8

Start Timing In: Yellow

Start Veh Calls: 1 2 3 4 5 6 7 8

Start Ped Calls: - 2 - 4 - 6 - 8

PHASE FEATURES

Lag Phases: _____

Non Actuated 1: _____

Non Actuated 2: _____

Cond Serv Phases: _____

Red Rest Phases: _____

Flash Phases: _____

Dual Entry Phases: _____

Simultaneous Gap: No

Min Yellow Time: 3.0

Red Revert Time: 5.0

LCD SETUP / SHAPE

Xsect Shape: _____

South Thru: 4

South Turn: 7

West Thru: 6

West Turn: 1

North Thru: 8

North Turn: 3

East Thru: 2

East Turn: 5

South Ped: 4

West Ped: 6

North Ped: 8

East Ped: 2

FREE PARAMS

PHASE	1	2	3	4	5	6	7	8
Ped Walk:		5						
Ped Protect:		25						23
Add/Actuation:								
Initial:	5	5	5	8	5	5	5	8
Max Initial:								
Extension:	3	4	3	5	3	4	3	5
Min Extension:								
Before Reduce:								
Time to Reduce:								
Max Green:	24	32	24	32	24	32	24	32
Max II Green:	24	32	24	32	24	32	24	32
Yellow Change:	3	4	3	4	3	4	3	4
Red Clearance:	1	1	1	1	1	1	1	1
Bike Timing:		16		16		16		16

DETECTOR RECALLS

Min Recall: - - - 4 - - - 8 /

Max Recall: _____

Ped Recall: _____

Override Recall: _____

Soft Recall: - - - 4 - - - 8

OVERLAPS

1. Parent Phases: _____

Timing Method: _____

Overlap Type: _____

Green Extension: _____

Yellow Change: _____

Red Clearance: _____

2. Parent Phases: _____

Timing Method: _____

Overlap Type: _____

Green Extension: _____

Yellow Change: _____

Red Clearance: _____

OVERLAPS

3. Parent Phases: _____

Timing Method: _____

Overlap Type: _____

Green Extension: _____

Yellow Change: _____

Red Clearance: _____

4. Parent Phases: _____

Timing Method: _____

Overlap Type: _____

Green Extension: _____

Yellow Change: _____

Red Clearance: _____

EMERGENCY VEHICLE PREEMPTS

Delay Time: _____

Hold Time: _____

EVP Phases: _____

EVP Min Time: _____

Abort Min: _____

Abort Walk: _____

Abort FDW: _____

Ret Veh Calls: _____

Ret Ped Calls: _____

Delay Output: _____

RAILROAD PREEMPT

Delay Time: _____

Hold Time: _____

Track Phases: _____

Preempt Phases: _____

Track 0 Grm Time: _____

Intvl 5 In Flash: _____

Intvl 5 Min Grm: _____

Intvl 6 Yellow: _____

Intvl 7 Red Time: _____

Return Phases: _____

Ret Veh Calls: _____

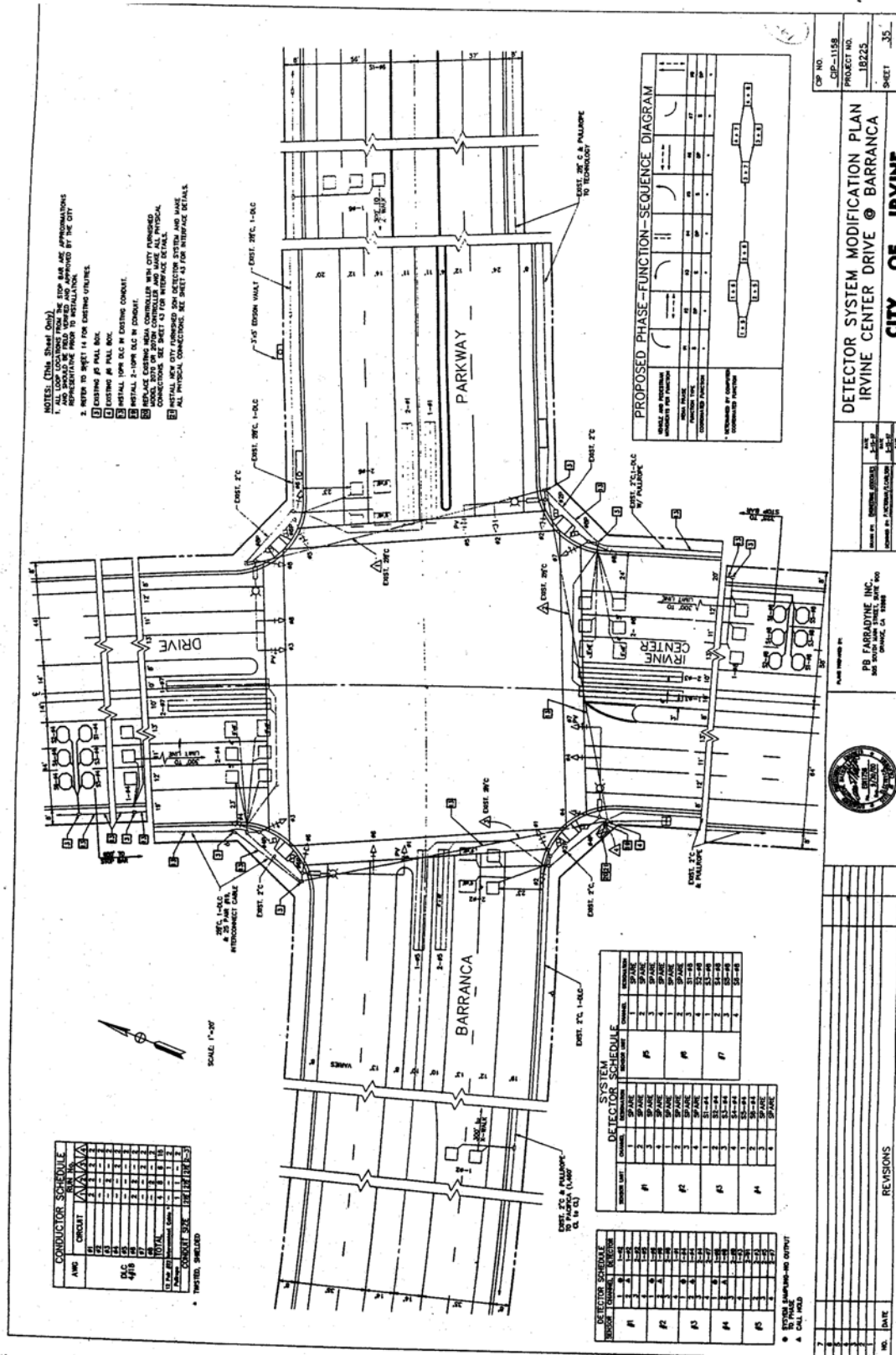
Ret Ped Calls: _____

Abort FDW: _____

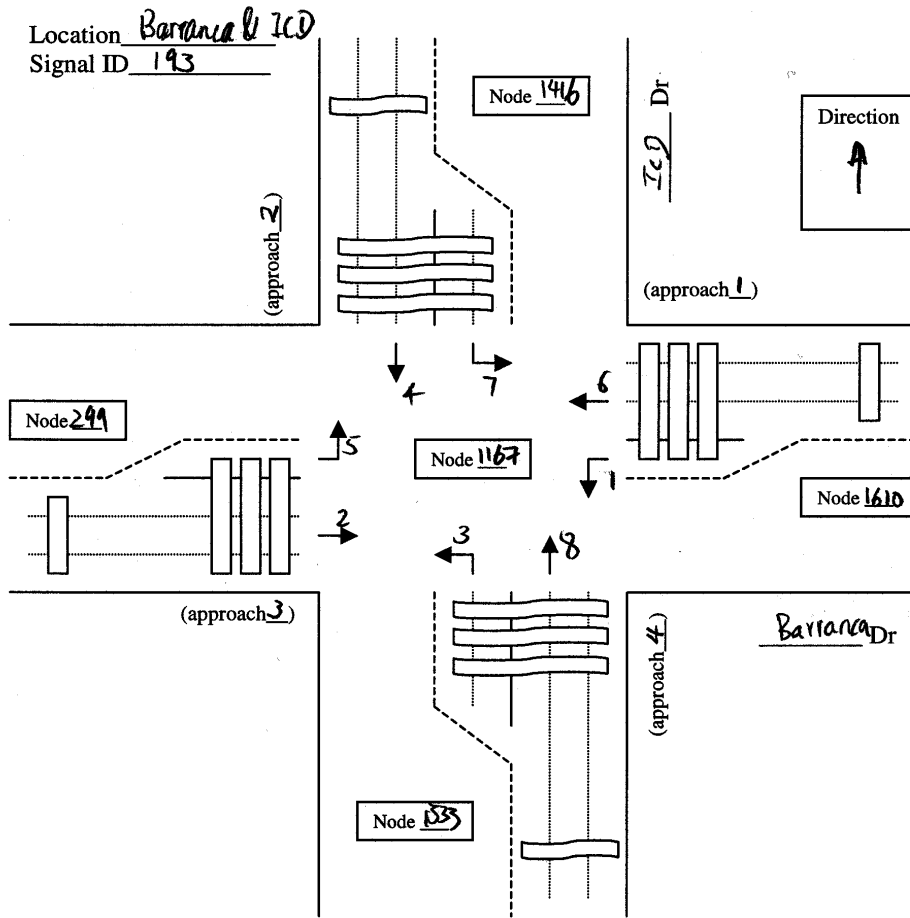
Preempt Enabled: _____

PREEMPT KEY

APPENDIX 3 Geometric layout of the intersection



APPENDIX 4 Completed worksheet for the example intersection



Node	1167							
Movement	1	2	3	4	5	6	7	8
Initial Green	5	5	5	8	5	5	5	8
Extension	3	4	3	5	3	4	3	5
Max Green	24	32	24	32	24	32	24	32
Recall Phase	4	8						
Lanes	2	2	2	3	2	2	2	3
Right-Turn lanes	1		1		1		1	
Detector 1	icb3w	icb2w	icb3w	icb1w				
Detector 2	icb3s	icb2s	icb3s	icb1s				
Detector 3	icb3e	icb2e	icb3e	icb1e				
Detector 4	icb3n	icb2n	icb3n	icb1n				

APPENDIX 5 The priorities information for the example intersection

```
actions 1167
phase offset 0.00 sec
phase 1
    0.00
    max 100.00
red phase 0.00
fill
all barred except
from 1416 to 299 minor
from 1610 to 1416 minor
from 1610 to 1533 major
from 299 to 1416 major
from 299 to 1533 minor
from 1533 to 1610 minor
phase 2
    0.00
    max 100.00
red phase 0.00
fill
all barred except
from 1416 to 299 minor
from 1610 to 1416 minor
from 299 to 1416 major
from 299 to 1533 minor
from 299 to 1610 major
from 1533 to 1610 minor
phase 3
    0.00
    max 100.00
red phase 0.00
fill
all barred except
from 1416 to 299 minor
from 1610 to 1416 minor
from 1610 to 299 major
from 1610 to 1533 major
from 299 to 1533 minor
from 1533 to 1610 minor
phase 4
    0.00
    max 100.00
red phase 0.00
fill
```

all barred except
from 1416 to 299 minor
from 1610 to 1416 minor
from 1610 to 299 major
from 299 to 1533 minor
from 299 to 1610 major
from 1533 to 1610 minor
phase 5

0.00

max 100.00

red phase 0.00

fill

all barred except
from 1416 to 299 minor
from 1416 to 1610 major
from 1610 to 1416 minor
from 299 to 1533 minor
from 1533 to 299 major
from 1533 to 1610 minor
phase 6

0.00

max 100.00

red phase 0.00

fill

all barred except
from 1416 to 299 minor
from 1416 to 1533 major
from 1416 to 1610 major
from 1610 to 1416 minor
from 299 to 1533 minor
from 1533 to 1610 minor
phase 7

0.00

max 100.00

red phase 0.00

fill

all barred except
from 1416 to 299 minor
from 1610 to 1416 minor
from 299 to 1533 minor
from 1533 to 1416 major
from 1533 to 299 major
from 1533 to 1610 minor
phase 8

0.00

max 100.00

red phase 0.00
fill
all barred except
from 1416 to 299 minor
from 1416 to 1533 major
from 1610 to 1416 minor
from 299 to 1533 minor
from 1533 to 1416 major
from 1533 to 1610 minor

APPENDIX 6 Loading plugins in PARAMICS

In version 3 of PARAMICS, the method to load a plugin is to specify the path and the name of the plugin in the “plugins” file located at
“\Program Files\Paramics\plugins\windows\”.

The grammar of this file is:

```
D:\Program Files\ParamicsV4\uci_plugins\actuated_signal.dll
```

Note there should be a “ENTER” at the end of the file. Otherwise the last plugin will not be loaded.

Version 4 of PARAMICS introduces a network specified method to load plugins. Each network has a “programming” file, which contains the plugins used together with the network. If you put this plugin in the PARAMICS root directory (where you can find other Quadstone’s plugins, including HOV, Loop aggregator, and Monitor), you do not need to specify the path of this plugin in the “programming” file:

```
actuated_signal.dll
```

If this plugin is stored to a directory other than the root directory of PARAMICS, the path of the loaded plugin need to be specified:

```
\Program Files\ParamicsV4\uci_plugins\actuated_signal.dll
```

Note PARAMICS thinks this plugin is in Drive C only. If you put this plugin to Driver D or others, PARAMICS will not find the plugin. In addition, note that no drive can be specified in this “programming” file. The following format will not be identified by PARAMICS as a valid plugin input:

```
D:\Program Files\ParamicsV4\uci_plugins\actuated_signal.dll
```

If you prefer to put this plugin to Driver D or others, you have to use the way of version 3 of PARAMICS to load plugins.