Smart Traffic Junction (4■Way) — Siemens TIA (LAD + SCL/ST)

Formal Technical Documentation

1. Overview

This document describes a four■way smart traffic signal controller implemented for Siemens SIMATIC S7■1200/1500 PLCs using a minimal Ladder (LAD) main organization block and Structured Text (SCL) for the core logic. The solution supports adaptive green extensions based on vehicle detection, pedestrian crossing phases with all■red interlocks, and a parameter database for on■line tuning. The design is suitable for demonstration, training, and prototyping purposes. For deployment in public roadways, applicable traffic control standards and safety integrity requirements must be followed.

2. System Architecture and Data Flow

The program comprises: (i) an SCL function (FC) that computes adaptive green extensions, (ii) an SCL function block (FB) that implements the complete finite■state machine and calls the function, (iii) a global parameter data block (DB) containing tunable timing parameters, and (iv) a minimal LAD OB1 that instantiates the FB, connects physical I/O, and passes parameters.

```
OB1 (LAD)
  ■■ FB "JunctionCtrl" (Instance DB)
       • Wires detectors, pedestrian requests, and lamps
       • Receives timing parameters from DB "DB_Params_Junction"
       • Owns IEC TON timers and state machine; calls FC "Calc_Extension"

FC "Calc_Extension" (SCL)
  • Returns TIME = per■scan extension (bounded by remaining allowance and step)

DB "DB_Params_Junction" (Global)
  • Min/Max green, step, yellow, all■red, pedestrian walk/clear intervals
```

3. Blocks and Roles

3.1 Function (FC) — “Calc_Extension” (SCL)

Purpose: Compute the additional green time to be granted during the current scan/cycle when a vehicle queue is detected. The function returns a TIME value via the function name (as per TIA SCL). Variable references use the “#” prefix.

```
FUNCTION "Calc_Extension" : TIME
{ S7_Optimized_Access := 'TRUE' }
VAR_INPUT
    HasQueue     : BOOL;   // detector for the CURRENT green approach
    ElapsedGreen : TIME;   // #tGreen.ET from the FB
    MaxExt       : TIME;   // total extra time allowed beyond MinGreen
    Step         : TIME;   // per-scan (or per-cycle) extension step
END_VAR
VAR_TEMP
    remaining : TIME;
    effStep   : TIME;
END_VAR

#effStep := #Step;
IF #effStep < T#0S THEN
```

```
    #effStep := T#0S;
END_IF;

#remaining := #MaxExt - #ElapsedGreen;
IF #remaining < T#0S THEN
    #remaining := T#0S;
END_IF;

IF #HasQueue THEN
    IF #remaining > #effStep THEN
        #Calc_Extension := #effStep;
    ELSE
        #Calc_Extension := #remaining;
    END_IF;
ELSE
    #Calc_Extension := T#0S;
END_IF;

END_FUNCTION
```

3.2 Function Block (FB) — “JunctionCtrl” (SCL)

Purpose: Owns the finite■state machine (NS/EW Green and Yellow, All■Red, Pedestrian Walk/Clear), the IEC TON instances, and all output driving logic. Pedestrian requests are latched and served in all■red windows. The block regularly invokes the function “Calc_Extension” during active green to grant incremental extensions up to configured caps.

```
FUNCTION_BLOCK "JunctionCtrl"
{ S7_Optimized_Access := 'TRUE' }

VAR_INPUT
    Enable           : BOOL;

    // Detectors
    NS_Det           : BOOL;
    EW_Det           : BOOL;

    // Pedestrian requests (to cross the named roadway)
    PedReq_NS        : BOOL;
    PedReq_EW        : BOOL;

    // Timings (can be sourced from a parameter DB)
    T_MinGreen_NS    : TIME := T#10S;
    T_MinGreen_EW    : TIME := T#10S;
    T_MaxExt_NS      : TIME := T#15S;
    T_MaxExt_EW      : TIME := T#15S;
    T_ExtStep        : TIME := T#3S;
    T_Yellow         : TIME := T#3S;
    T_AllRed         : TIME := T#1S;
    T_PedWalk        : TIME := T#6S;
    T_PedClear       : TIME := T#4S;
END_VAR

VAR_OUTPUT
    NS_R, NS_Y, NS_G : BOOL;
    EW_R, EW_Y, EW_G : BOOL;
    PedWalk_NS       : BOOL;
    PedWalk_EW       : BOOL;
    CurrState        : INT;
END_VAR

VAR
    // State control
    State            : INT := 0;
    PrevState        : INT := -1;
    NextIsEW         : BOOL := TRUE;

    // Latches for ped calls
```

```
    PedReq_NS_L      : BOOL;
    PedReq_EW_L      : BOOL;

    // IEC Timers (instances)
    tGreen           : TON;
    tYellow          : TON;
    tAllRed          : TON;
    tPed             : TON;

    // scratch
    ext              : TIME;
END_VAR

VAR CONSTANT
    ST_NS_GREEN      : INT := 0;
    ST_NS_YELLOW     : INT := 1;
    ST_PED_NS_WALK   : INT := 2;
    ST_PED_NS_CLEAR  : INT := 3;
    ST_ALL_RED       : INT := 4;
    ST_EW_GREEN      : INT := 5;
    ST_EW_YELLOW     : INT := 6;
    ST_PED_EW_WALK   : INT := 7;
    ST_PED_EW_CLEAR  : INT := 8;
END_VAR

// Reset timers on state change
IF #State <> #PrevState THEN
    #tGreen(IN := FALSE, PT := T#0S);
    #tYellow(IN := FALSE, PT := T#0S);
    #tAllRed(IN := FALSE, PT := T#0S);
    #tPed(IN := FALSE, PT := T#0S);
END_IF;

// Fail-safe: all red if disabled
IF NOT #Enable THEN
    #NS_R := TRUE;  #NS_Y := FALSE; #NS_G := FALSE;
    #EW_R := TRUE;  #EW_Y := FALSE; #EW_G := FALSE;
    #PedWalk_NS := FALSE; #PedWalk_EW := FALSE;
    #CurrState := ST_ALL_RED;
    #PrevState := ST_ALL_RED;
    RETURN;
END_IF;

// Latch ped requests
IF #PedReq_NS THEN #PedReq_NS_L := TRUE; END_IF;
IF #PedReq_EW THEN #PedReq_EW_L := TRUE; END_IF;

// Defaults (all red, no walk)
#NS_R := TRUE; #NS_Y := FALSE; #NS_G := FALSE;
#EW_R := TRUE; #EW_Y := FALSE; #EW_G := FALSE;
#PedWalk_NS := FALSE; #PedWalk_EW := FALSE;

// State machine
CASE #State OF

    // N-S GREEN
    ST_NS_GREEN:
        #NS_R := FALSE; #NS_G := TRUE;
        #ext := Calc_Extension(
                    HasQueue     := #NS_Det,
                    ElapsedGreen := #tGreen.ET,
                    MaxExt       := #T_MaxExt_NS,
                    Step         := #T_ExtStep);
        #tGreen(IN := TRUE, PT := #T_MinGreen_NS + #ext);
        IF #tGreen.Q THEN
            #State    := ST_NS_YELLOW;
            #NextIsEW := TRUE;
        END_IF;

    // N-S YELLOW
```

```
ST_NS_YELLOW:
    #NS_R := FALSE; #NS_Y := TRUE;
    #tYellow(IN := TRUE, PT := #T_Yellow);
    IF #tYellow.Q THEN
        IF #PedReq_NS_L THEN
            #State := ST_PED_NS_WALK;
        ELSE
            #State := ST_ALL_RED;
        END_IF;
    END_IF;

// Pedestrians cross N-S roadway
ST_PED_NS_WALK:
    #PedWalk_NS := TRUE;
    #tPed(IN := TRUE, PT := #T_PedWalk);
    IF #tPed.Q THEN
        #State := ST_PED_NS_CLEAR;
    END_IF;

ST_PED_NS_CLEAR:
    #tPed(IN := TRUE, PT := #T_PedClear);
    IF #tPed.Q THEN
        #PedReq_NS_L := FALSE;
        #State := ST_ALL_RED;
    END_IF;

// ALL RED (buffer)
ST_ALL_RED:
    #tAllRed(IN := TRUE, PT := #T_AllRed);
    IF #tAllRed.Q THEN
        IF #NextIsEW THEN
            #State := ST_EW_GREEN;
        ELSE
            #State := ST_NS_GREEN;
        END_IF;
    END_IF;

// E-W GREEN
ST_EW_GREEN:
    #EW_R := FALSE; #EW_G := TRUE;
    #ext := Calc_Extension(
                HasQueue     := #EW_Det,
                ElapsedGreen := #tGreen.ET,
                MaxExt       := #T_MaxExt_EW,
                Step         := #T_ExtStep);
    #tGreen(IN := TRUE, PT := #T_MinGreen_EW + #ext);
    IF #tGreen.Q THEN
        #State    := ST_EW_YELLOW;
        #NextIsEW := FALSE;
    END_IF;

// E-W YELLOW
ST_EW_YELLOW:
    #EW_R := FALSE; #EW_Y := TRUE;
    #tYellow(IN := TRUE, PT := #T_Yellow);
    IF #tYellow.Q THEN
        IF #PedReq_EW_L THEN
            #State := ST_PED_EW_WALK;
        ELSE
            #State := ST_ALL_RED;
        END_IF;
    END_IF;

// Pedestrians cross E-W roadway
ST_PED_EW_WALK:
    #PedWalk_EW := TRUE;
    #tPed(IN := TRUE, PT := #T_PedWalk);
    IF #tPed.Q THEN
        #State := ST_PED_EW_CLEAR;
```

```
        END_IF;

    ST_PED_EW_CLEAR:
        #tPed(IN := TRUE, PT := #T_PedClear);
        IF #tPed.Q THEN
            #PedReq_EW_L := FALSE;
            #State := ST_ALL_RED;
        END_IF;

    // default boot state
    ELSE
        #State := ST_NS_GREEN;
END_CASE;

#CurrState := #State;
#PrevState := #State;

END_FUNCTION_BLOCK
```

3.3 Global Parameter DB — "DB_Params_Junction"

The following parameters are intended for online tuning. All are of type TIME.

Tag	Type	Default	Description
MinGreen_NS	TIME	T#10S	Base minimum green for N■S
MinGreen_EW	TIME	T#10S	Base minimum green for E■W
MaxExt_NS	TIME	T#15S	Maximum extension beyond MinGreen on N■S
MaxExt_EW	TIME	T#15S	Maximum extension beyond MinGreen on E■W
ExtStep	TIME	T#3S	Extension step per evaluation
Yellow	TIME	T#3S	Amber interval
AllRed	TIME	T#1S	All■red interlock duration
PedWalk	TIME	T#6S	Pedestrian steady walk interval
PedClear	TIME	T#4S	Pedestrian clearance interval

4. Example I/O Mapping

Signal	Address	Meaning
NS_Det	I0.0	N■S vehicle detector
EW_Det	I0.1	E■W vehicle detector
PedReq_NS	I0.2	Pedestrian request to cross N■S roadway
PedReq_EW	I0.3	Pedestrian request to cross E■W roadway
NS_R / NS_Y / NS_G	Q0.0 / Q0.1 / Q0.2	N■S lamp heads (Red/Yellow/Green)
EW_R / EW_Y / EW_G	Q0.3 / Q0.4 / Q0.5	E■W lamp heads (Red/Yellow/Green)
PedWalk_NS	Q0.6	Pedestrian walk indicator (N■S crossing)
PedWalk_EW	Q0.7	Pedestrian walk indicator (E■W crossing)

5. State Machine and Sequence

The controller cycles between N■S and E■W vehicle phases. Each vehicle phase comprises a Green interval (extendable) followed by a fixed Yellow interval. Transitions between phases and before/after pedestrian phases are protected by an All■Red buffer. Pedestrian requests are latched and inserted after the active vehicle Yellow. During pedestrian phases all vehicle signals are red; pedestrian Walk is followed by a clearance interval.

```
States:
  ST_NS_GREEN      – N■S green, adaptive extension applied.
  ST_NS_YELLOW     – N■S yellow.
  ST_PED_NS_WALK   – Pedestrians cross N■S roadway (all vehicles red).
  ST_PED_NS_CLEAR  – Pedestrian clearance (all vehicles red).
  ST_ALL_RED       – Interlock buffer between phases.
  ST_EW_GREEN      – E■W green, adaptive extension applied.
  ST_EW_YELLOW     – E■W yellow.
  ST_PED_EW_WALK   – Pedestrians cross E■W roadway (all vehicles red).
  ST_PED_EW_CLEAR  – Pedestrian clearance (all vehicles red).
```

6. Adaptive Extension Algorithm

During active green, the FB calls the FC to request an increment of extra time. The FC computes the remaining allowance (MaxExt – ElapsedGreen) clamped to zero, clamps negative step values to zero, and returns

min(remaining, step) when a queue is present; otherwise it returns zero. The returned extension is added to the green timer's preset (#PT) together with the minimum green to realize incremental extensions without breaching the cap.

7. Project Setup (TIA Portal) and Build Order

1) Create FC “Calc_Extension” (SCL) and paste the code above. 2) Create FB “JunctionCtrl” (SCL) and paste the finite■state machine code. 3) Create global DB “DB_Params_Junction” and define the TIME parameters. 4) In OB1 (LAD), instantiate FB “JunctionCtrl” (TIA will create an instance DB), wire I/O, and pass DB parameters. 5) Compile in this order: FC → FB → OB1. Download to PLC or PLCSIM.

8. Simulation with PLCSIM

• Toggle I0.0 (NS_Det) and I0.1 (EW_Det) to simulate vehicle presence; observe Q0.2 (NS_G) and Q0.5 (EW_G). • Assert I0.2 or I0.3 to request pedestrian crossings; observe Q0.6/Q0.7 for Walk signals and the all■red windows. • Modify parameters in DB_Params_Junction on■line to evaluate behavior changes.

9. HMI and Diagnostics

Expose CurrState for textual mapping on an HMI (e.g., via a text list). Display lamp outputs and pedestrian indicators on a mimic, and optionally provide controls for Enable and parameter editing (with appropriate access levels).

10. Safety and Failsafe Behavior

If Enable is FALSE, the controller drives all vehicle signals to red and disables pedestrian signals. All transitions are separated by an All■Red interval. For real■world deployment, implement external interlocks (e.g., lamp monitoring, hardwired E■Stops) and conform to local traffic signal standards and safety integrity requirements.

11. Troubleshooting

• Compilation errors: confirm block interfaces match exactly and that SCL variables use the “#” prefix. • RET_VAL usage: in SCL, assign the function return via #Calc_Extension; in LAD/FBD the output appears as RET_VAL. • TIME type mismatches: ensure all timing parameters and ext are TIME; tGreen.ET is TIME on S7■1200/1500. • No extension: verify detectors are TRUE during green, MaxExt_* > 0, and ExtStep > 0.

Note: This document and associated code are provided for educational and prototyping use only.