James DeBacker and Nicholas Fuselier

April 6, 2017

# Lab 2
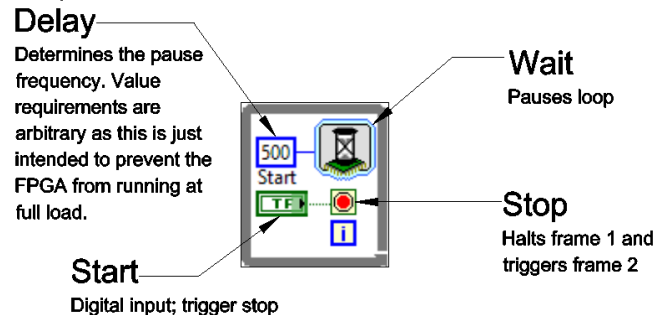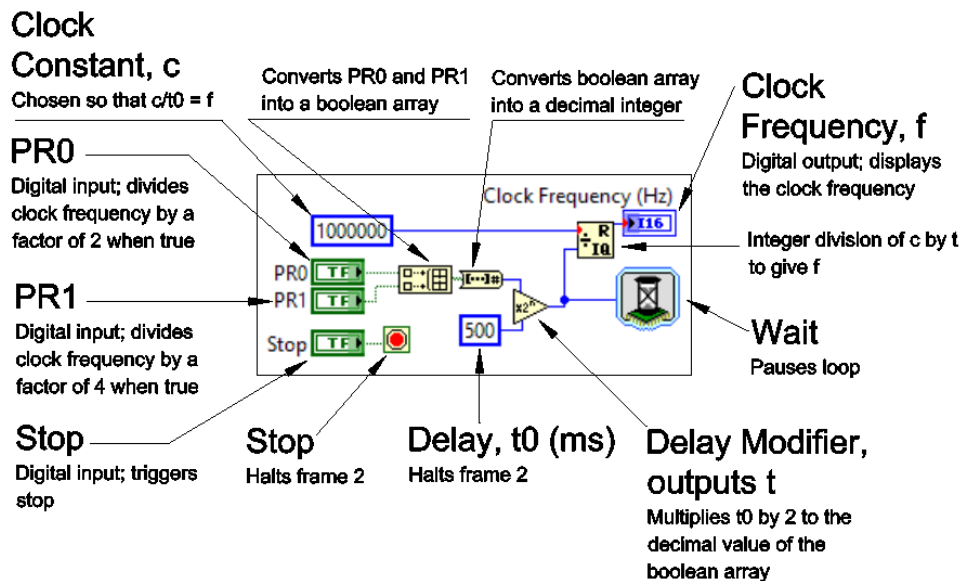
## main_FPGA.vi (Basic Light)

This VI runs on the FPGA and is responsible for controlling stoplight operation and interfacing physical and digital I/O necessary to alter stoplight functionality.

## Block Diagram

## Functional Subunits
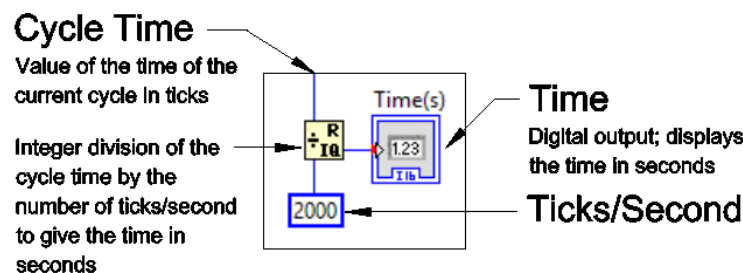
1: Frame 1 – Prevents execution of frame 2 until stoplight operation is desired
2: Frame 2 – Contains the stoplight loop
3: Hungry Loop – The mechanism by which frame 1 is maintained and frame 2 is held off



**Delay**
Determines the pause frequency. Value requirements are arbitrary as this is just intended to prevent the FPGA from running at full load.

**Wait**
Pauses loop

**Stop**
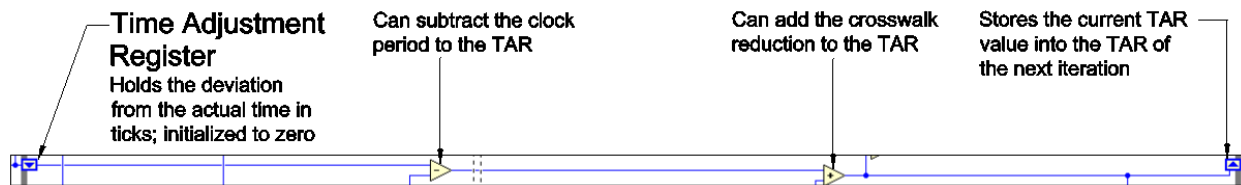Halts frame 1 and triggers frame 2

**Start**
Digital input; trigger stop

4: Stoplight Loop – Contains the logic for stoplight operation and handles I/O in a manner corresponding to that specified in the lab instructions.
5: Clock – Determines the frequency of the loop clock in response to PR0 and PR1, with a default frequency of 2kHz



**Clock Constant, c**
Chosen so that c/t0 = f

**Converts PR0 and PR1 into a boolean array**

**Converts boolean array into a decimal integer**

**Clock Frequency, f**
Digital output; displays the clock frequency

**PR0**
Digital input; divides clock frequency by a factor of 2 when true

Integer division of c by t to give f

**PR1**
Digital input; divides clock frequency by a factor of 4 when true

**Wait**
Pauses loop

**Stop**
Digital input; triggers stop

**Stop**
Halts frame 2

**Delay, t0 (ms)**
Halts frame 2

**Delay Modifier, outputs t**
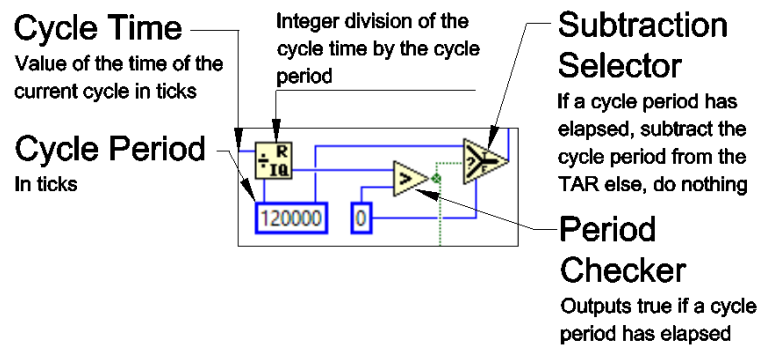Multiplies t0 by 2 to the decimal value of the boolean array

6: Time – Converts the number of iterations (here being the equivalent to clock ticks) into seconds. 2000 ticks/second.



**Cycle Time**
Value of the time of the current cycle in ticks

Integer division of the cycle time by the number of ticks/second to give the time in seconds

**Time**
Digital output; displays the time in seconds
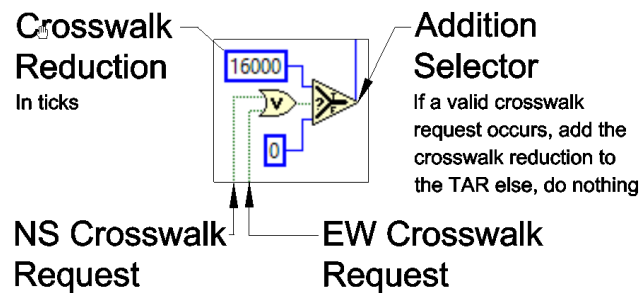
**Ticks/Second**

7: Time Adjustment Register (TAR) – Modifies the transmitted value of the clock register in response to the completion of a cycle of a crosswalk request.

**Time Adjustment Register**
Holds the deviation from the actual time in ticks; initialized to zero

Can subtract the clock period to the TAR

Can add the crosswalk reduction to the TAR

Stores the current TAR value into the TAR of the next iteration

7.1: Cycle Adjust – Subtracts the cycle period (120s) from the TAR on the completion of a cycle

**Cycle Time**
Value of the time of the current cycle in ticks

Integer division of the cycle time by the cycle period

**Subtraction Selector**
If a cycle period has elapsed, subtract the cycle period from the TAR else, do nothing

**Cycle Period**
In ticks

120000    0

**Period Checker**
Outputs true if a cycle period has elapsed

7.2: Crosswalk Adjust – Adds the crosswalk reduction (8s) to the TAR on a valid crosswalk request.

**Crosswalk Reduction**
In ticks

16000

0

**Addition Selector**
If a valid crosswalk request occurs, add the crosswalk reduction to the TAR else, do nothing

**NS Crosswalk Request**

**EW Crosswalk Request**

8: Time Register – Adds the value in the TAR to the number of iterations (right) to get a cycle specific virtual time (left).

**Cycle Time**
Gives the time of the current cycle in ticks; initialized to 0

**Iterations**
Acts as the real time reference

Adds the value of the TAR to the number of iterations to give the cycle time
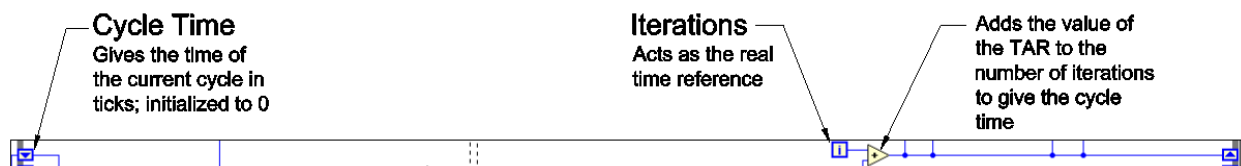
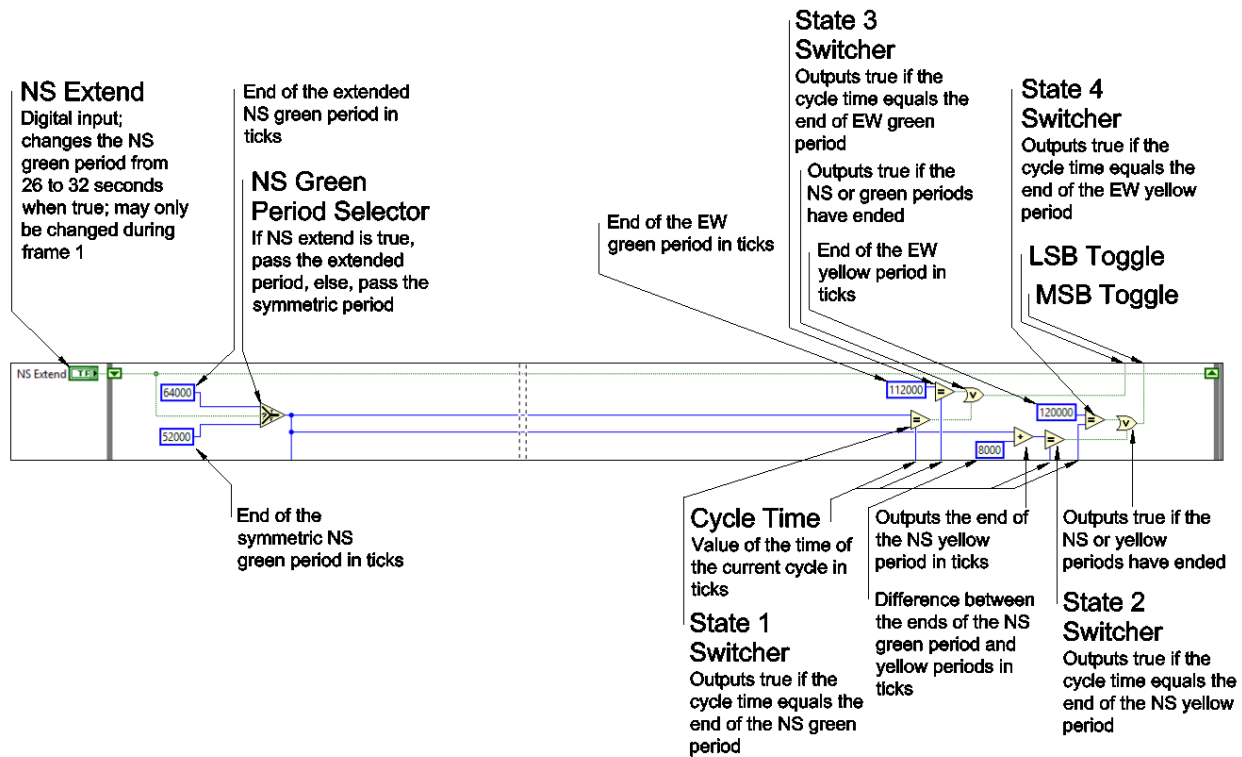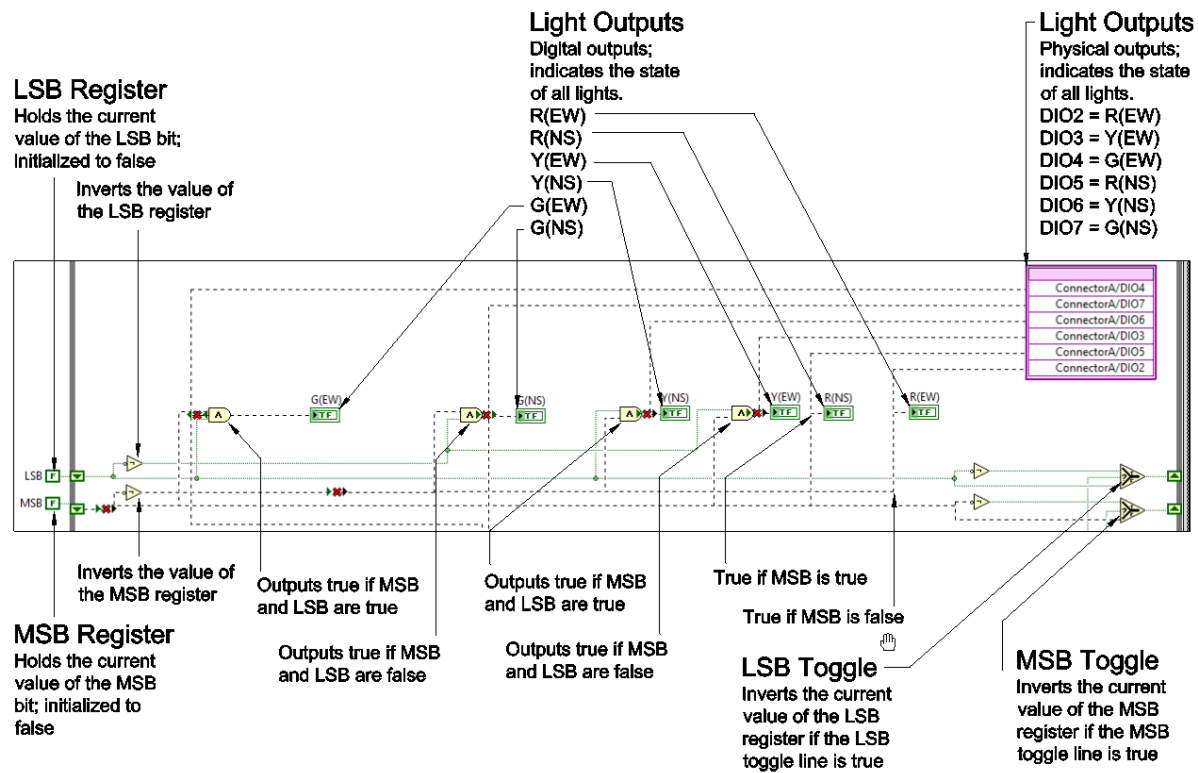9: Cross Request Validator – Approve the transmission of a crosswalk request in accord with the constraints specified in the lab instructions. Accepts physical and digital crossing requests. Most functionality is mirrored between the EW and NS validators.

**Cross Register**
Holds true if a valid cross request has been submitted; initialized to false

**Cycle Time**
Value of the time of the current cycle in ticks

The time in ticks that a NS cross request must be submitted before the NS green period ends

Gives the time in ticks by which a NS cross request must be submitted

The time in ticks that the NS green period ends

**NS Cross Period Checker**
Outputs true if the current time is within the NS cross period

True if a cycle period has elapsed

True if the NS light is green

True if the EW light is green

**EW Cross Request**

**NS Cross Request**

**Cross Request Validator**
Outputs true no previous cross requests were valid, the current time is within the cross period, and a cross request has been recieved

**Cross Reset**
Sets the value of the cross register to false after a cycle period has elapsed

True if a cycle period has not elapsed

If true, passes the current value of the cross register, else makes the value of the cross register true on a cross request

Cross(NS)

16000

Crosswalk(NS) new

NS phy button
Crosswalk(NS) new
NS RT button
NS Accepted

Accept Physical Inputs?

ConnectorA/DIO0
ConnectorA/DIO1

False

Invert Physical Inputs?

EW phy button
Crosswalk(EW) new
EW Accepted
EW RT button

96000

Cross(EW)

The time in ticks that the NS green period ends

**NS Cross Period Checker**
Outputs true if the current time is within the NS cross period

Digital input; toggles whether or not physical inputs are inverted for debugging purposes

**Crosswalk Inputs**
Physical inputs; outputs true if a crosswalk request occurs; DIO0 corresponds to NS and DIO1 corresponds to EW crossing requests

Digital input; toggles whether or not physical inputs are accepted for debugging purposes

**Physical Crosswalk Indicator**
Digital output; indicates whether or not the physical crosswalk button is being pressed for debugging purposes

**Crosswalk Input**
Digital inputs; outputs true if a crosswalk request occurs

Outputs true if a physical inputs are being acccepted and a crosswalk request occurs

Outputs true if an accepted crosswalk request occurs

**Digital Crosswalk Indicator**
Digital output; indicates whether or not the digital crosswalk button is being pressed for debugging purposes

**Crosswalk Indicator**
Digital output; indicates whether or not an accepted crosswalk button is being pressed for debugging purposes

Passes a crosswalk request if the EW light is green; not neccesary for the NS light because it is the starting state of each cycle period

10: Light Switcher – Determines the periods of light operation in response to the virtual time and the value of the NS Extend Register. Operated via the toggling of the LSB and MSB registers.

**State 3 Switcher**
Outputs true if the cycle time equals the end of EW green period

**NS Extend**
Digital input; changes the NS green period from 26 to 32 seconds when true; may only be changed during frame 1

End of the extended NS green period in ticks

**NS Green Period Selector**
If NS extend is true, pass the extended period, else, pass the symmetric period

Outputs true if the NS or green periods have ended

End of the EW green period in ticks

End of the EW yellow period in ticks

**State 4 Switcher**
Outputs true if the cycle time equals the end of the EW yellow period

**LSB Toggle**

**MSB Toggle**

End of the symmetric NS green period in ticks

**Cycle Time**
Value of the time of the current cycle in ticks

Outputs the end of the NS yellow period in ticks

Outputs true if the NS or yellow periods have ended

**State 1 Switcher**
Outputs true if the cycle time equals the end of the NS green period

Difference between the ends of the NS green period and yellow periods in ticks

**State 2 Switcher**
Outputs true if the cycle time equals the end of the NS yellow period

11: Light Logic – Determines the output configuration of the lights in response to the values of the LSB and MSB register in accord with the truth table specified in the lab instructions. Outputs digitally and physically.

**Light Outputs**
Digital outputs;
indicates the state
of all lights.
R(EW)
R(NS)
Y(EW)
Y(NS)
G(EW)
G(NS)

**Light Outputs**
Physical outputs;
indicates the state
of all lights.
DIO2 = R(EW)
DIO3 = Y(EW)
DIO4 = G(EW)
DIO5 = R(NS)
DIO6 = Y(NS)
DIO7 = G(NS)

**LSB Register**
Holds the current
value of the LSB bit;
initialized to false

Inverts the value of
the LSB register

ConnectorA/DIO4
ConnectorA/DIO7
ConnectorA/DIO6
ConnectorA/DIO3
ConnectorA/DIO5
ConnectorA/DIO2

G(EW)   G(NS)   Y(NS)   Y(EW)   R(NS)   R(EW)

LSB

MSB

Inverts the value of
the MSB register

**MSB Register**
Holds the current
value of the MSB
bit; initialized to
false

Outputs true if MSB
and LSB are true

Outputs true if MSB
and LSB are false

Outputs true if MSB
and LSB are true

Outputs true if MSB
and LSB are false

True if MSB is true

True if MSB is false

**LSB Toggle**
Inverts the current
value of the LSB
register if the LSB
toggle line is true

**MSB Toggle**
Inverts the current
value of the MSB
register if the MSB
toggle line is true

# 📊 main_RT.vi

This VI runs on the RT module and is responsible for routing physical and digital I/O to the targeted FPGA bitfile.

---

## Front Panel



Stoplight Control Panel                    Debug Control Panel

## Block Diagram



## ▶ Remote Panel Methods Client.vi

This VI illustrates how to programmatically connect to a remote front panel. A connection is established to the specified VI and control can also be requested. Once control is requested, the VI can only be run once since the server will close the connection automatically after one run.

## Front Panel

## Block Diagram

The **Open Connection To Server** method opens a remote panel connection to the specified VI at the given address and port. When the user clicks the Request Control button, the **Client Request Control** method gives control of the target VI to this instance of LabVIEW.

VI Name `abc`

**App**
RP.Open Connection To Server
VI Name
Server Address
Server Port
Request Control
Secure Connection

**VI**
FP.Open
Activate
State

The target VI opens as a top-most window. This **FP.Open** call with the **Activate** parameter set to TRUE will bring this VI back to the front.

**App**
RP.Client Request Control
VI Name
Server Address
Server Port

Remote Panel Server Address
`abc`

Remote Panel Server Port
`I32`

# 🖼 main_FPGA.vi (With Sensors and Smart Timing)

The changes to this version over the above version are the addition of the NSEW sensors and the NS Priority and NS Smart modes. Their functional spread makes commenting impractical.
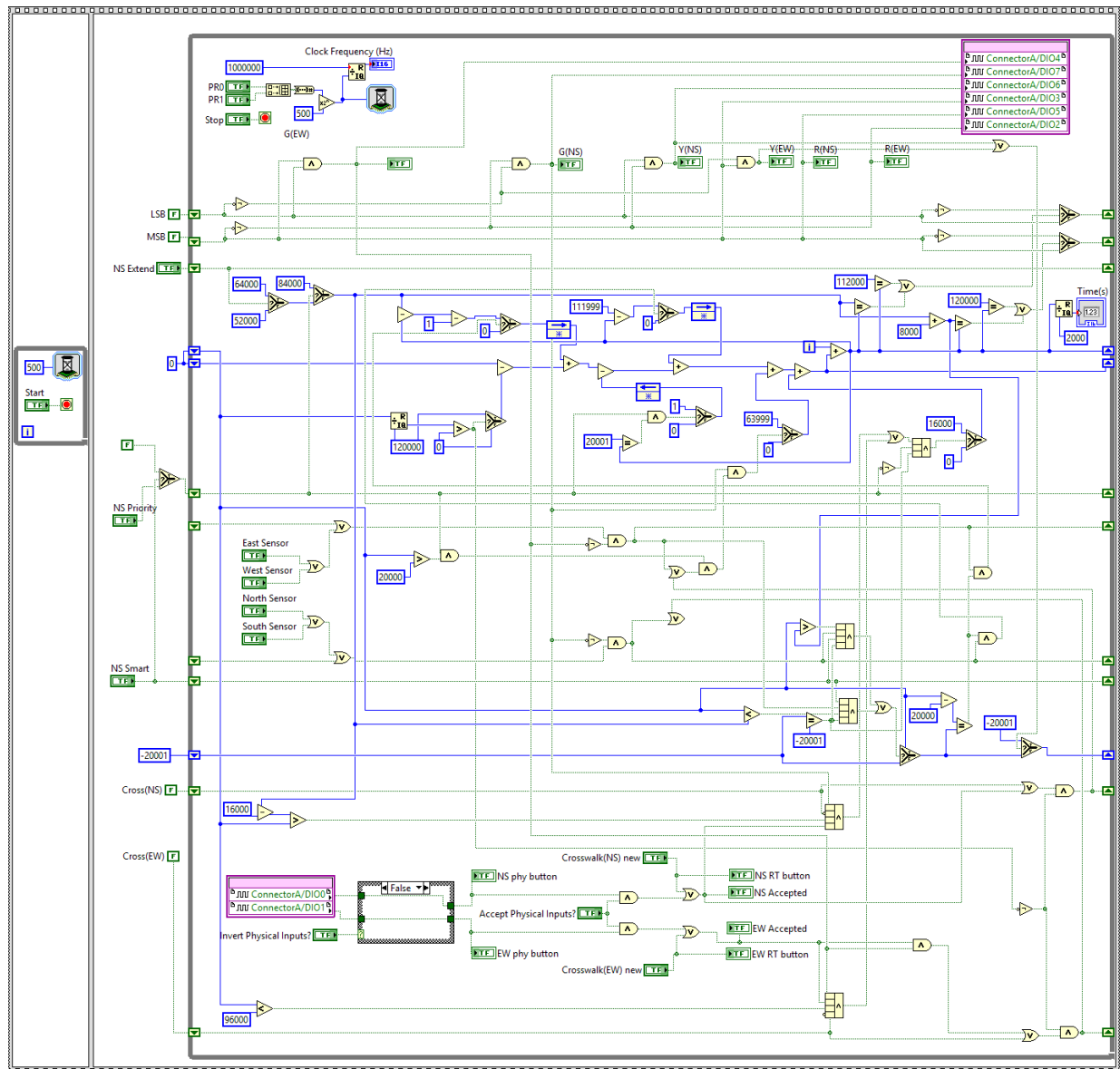


1. In NS Priority mode, the light is by default NS green. If greater than 10 seconds have elapsed since the light has been NS green, then either an EW crosswalk request or the EW sensors can trigger the continuation of light changes. The NS green light will be on for 10 seconds in this partial cycle. The 8 second reduction triggered by the crosswalks is disabled in this mode.

2. In NS Smart mode the lights will operate normally or with NS Extend enabled; if a sensor cross to the current green light is triggered, then the light will cycle to the next green period within 10 seconds. Crosswalks function normally in this mode.