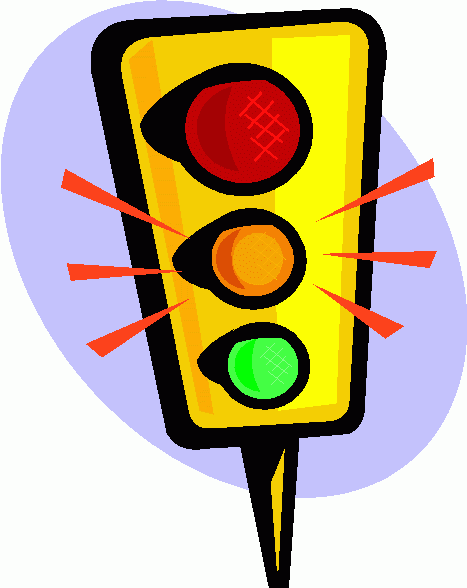
**Signal Tower**

**Phase I**

**Software Design Review**



Written by: Slava Chuhovich

September 2012

Table of Contents

[1. Introduction 3](#_Toc334968113)

[1.1 Background 3](#_Toc334968114)

[1.2 Design Goals 3](#_Toc334968115)

[1.3 Abbreviations and Acronyms 3](#_Toc334968116)

[2. Architecture and Implementation 4](#_Toc334968117)

[2.1 Hardware 4](#_Toc334968118)

[2.2 OCB Firmware 4](#_Toc334968119)

[2.3 User interface 4](#_Toc334968120)

[2.4 Embedded Application 5](#_Toc334968121)

[2.5 Parameters 6](#_Toc334968122)

[3. Development 6](#_Toc334968123)

[3.1 Limitations & Risks 6](#_Toc334968124)

[3.2 Development Stages 6](#_Toc334968125)

[4. Operation 6](#_Toc334968126)

[4.1 Limitations 6](#_Toc334968127)

[4.2 Installation 7](#_Toc334968128)

[5. Effort Estimation 7](#_Toc334968129)

# Introduction

## Background

The purpose is to add a Signal Tower (traffic light – like light source) device to a system, which will consist Red, Green and Yellow lights and will notify the operator of various system states and conditions.

Phase I of the implementation will include hardware infrastructure and minimal operational functionality.

## Design Goals

OCB2 firmware

1. Add Signal Tower unit to handle functionality.
2. Implement communication task.
3. Implement blinking task.
4. Add new Embedded – OCB communication message.
5. Add new actuator definitions.

Embedded software

1. Add Signal Tower unit to handle functionality.
2. Add new Embedded – OCB communication message.
3. Add new actuator definitions.
4. Add new actuators to UI at Actuators dialog.
5. Add required parameters to Parameters Manager.
6. Create runtime Signal Tower object in Machine Sequencer.
7. Expose required activation APIs to Back-End and Roster (python console).
8. Implement very basic functionality (Phase I).

## Abbreviations and Acronyms

|  |  |
| --- | --- |
| **Abbreviation** | **Description** |
| GUI | Graphic User Interface |
| UI | User Interface |
| OCB / OCB2 | Objet Control Board (Main board) |
| ST | Signal Tower |
| PM | Parameter’s Manager |

# Architecture and Implementation

## Hardware

1. ST device will be connected to J24 and J41, 24V power outputs on OCB2 card.
2. The hardware mapping is as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Light color | MCU port | Connector | OCB led | Actuator ID |
| Red | 4.0 | J41 - 2 | D30 | 0 |
| Yellow | 5.6 | J24 - 5 | D70 | 14 |
| Green | 4.1 | J41 - 1 | D29 | 1 |

## OCB Firmware

1. Define new actuator IDs for ST lights (table above).
2. Implement new Signal Tower unit to handle functionality.
3. Implement new OCB – Embedded communication message:  
     
    typedef struct

{

BYTE RedLightState;

BYTE GreenLightState;

BYTE YellowLightState;

WORD DutyOnTime;

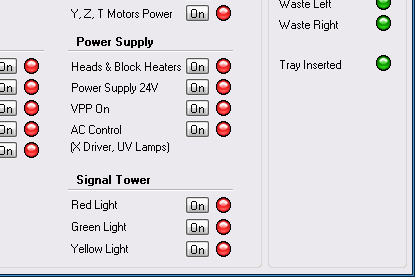
WORD DutyOffTime;

}TActivateSignalTowerMsg;

1. Implement SignalTowerAckTask to handle communication.
2. Implement SignalTowerBlinkingTask to handle blinking.
3. Implement SetSignalTowerLights function to handle direct activation.

## User interface

Implement following controls in “Actuators & Sensors” dialog (enabled by SignalTowerEnabled param):



## Embedded Application

1. Add SignalTower unit to the project.
2. Implement CSignalTower and CSignalTowerDummy classes to handle “connected” and emulation functionality.
3. Implement new OCB – Embedded communication message:  
     
    struct TOCBActivateSignalTowerMessage

{

BYTE MessageID;

BYTE RedLightState;

BYTE GreenLightState;

BYTE YellowLightState;

WORD DutyOnTime;

WORD DutyOffTime;

}STRUCT\_ATTRIBUTE;

1. Implement actuators for each light. (indexes in table above)
2. Implement activation types enumeration:  
     
    typedef enum

{

ST\_LIGHT\_ON = 1 << 0,

ST\_LIGHT\_OFF = 1 << 1,

ST\_LIGHT\_BLINK = 1 << 2,

ST\_LIGHT\_NC = 1 <<3

}TSignalTowerLightState;

1. Implement CSignalTower ::ActivateSignalTower method, which receives the states for each light and duty ON and duty OFF times (msec) as parameters.
2. Expose ST activation to Roster, to be used in python console or wizards if required.
3. Expose ST activation to Back-End interface. (Only 3 parameters for the lights states. Duty times taken from PM).
4. Create ST object in MachineSequencer.
5. Implement basic (Phase I) functionality, which means:
   1. Green light is ON during any printing process.
   2. Green light blinks (PM’s duty ON/OFF) when printing is finished, but the doors are yet closed.
6. Writing to application log ST activation / deactivation with a corresponding light color.

## Parameters

1. Add new Signal Tower section to PM.
2. Add SignalTowerEnabled check box type parameter. (default value is “checked” on Objet1000 and “unchecked” on other machines)
3. Add LightDutyOnTime\_ms parameter, which signifies blinking ON time in msec. (default value is 1000).
4. Add LightDutyOffTime\_ms parameter, which signifies blinking OFF time in msec. (default value is 1000).

# Development

## Limitations & Risks

None

## Development Stages

1. All development will be done in Trunk.
2. ST feature will be activated by SignalTowerEnabled parameter in Parameter’s Manager.
3. All hardware related functionality will be implemented in an OCB2 firmware code.
4. At initial stage, the development and testing will be done using “offline” OCB2 card and ST device.
5. At more advances stages of development the testing will be done on an Objet1000 Alpha machine.

# Operation

* SignalTowerEnabled parameter must be checked in Parameters Manger in order for this feature to work.

## Limitations

* Must use new OCB2 card with Silabs C8051F12x controller.

## Installation

1. Install Signal Tower electric device by connecting it’s connectors to J24 and J41 OCB2 ports.
2. Burn new OCB firmware (HEX file).
3. Install new embedded executable.

# Effort Estimation

Effort estimation is 10 working days, including implementation and testing.