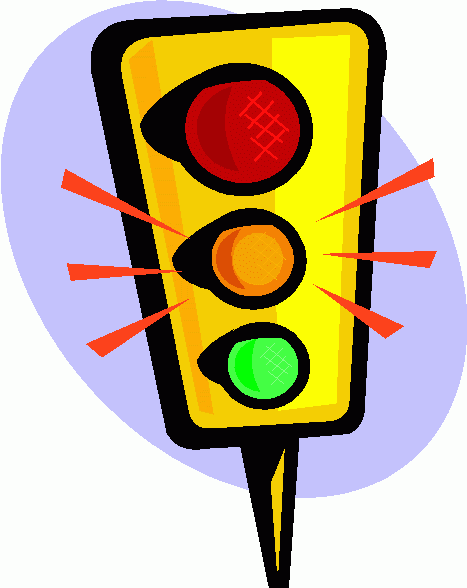
**Signal Tower Advanced**

**Phase 1**

**Software Design Review**



Written by: Doron Or

November 2012

Table of Contents

[1. Introduction 3](#_Toc341377385)

[1.1 Background 3](#_Toc341377386)

[1.2 Design Goals 3](#_Toc341377387)

[1.3 Abbreviations and Acronyms 3](#_Toc341377391)

[2. Architecture and Implementation 4](#_Toc341377392)

[2.1 Overview 4](#_Toc341377393)

[2.2 Embedded Application 5](#_Toc341377394)

[3. Development 5](#_Toc341377395)

[3.1 Limitations & Risks](#_Toc341377397)  5

[3.2 Development Stages 5](#_Toc341377399)

[4. Testing 6](#_Toc341377400)

[5. Operation 6](#_Toc341377401)

[5.1 Limitations](#_Toc341377403) 6

[5.2 Installation 6](#_Toc341377404)

[6. Effort Estimation 6](#_Toc341377408)

# Introduction

## Background

The purpose of this project is to add operational functionality to the existing Signal Tower implementation \* (traffic light – like light source).

The device consists of Red, Green and Yellow lights and will notify the operator of various system states and conditions while he is away from the machine.

This DR applies to SRS from: svn://octopus/ControlSW/Trunk/ObjetFamily/Documents/SRS\_SoftwareRequirementsSpecifications/Signal Tower Advanced/SRS- SignalTower- XL\_ver9.doc

\*DR for “Signal Tower – Basic” is located at svn://octopus/ControlSW/Trunk/ObjetFamily/Documents/DR Documents/Signal Tower DR.docx

## Design Goals

Embedded software

1. Add as few new logics as possible – use existing logics in sequences to turn on/off the signals
2. Modify the activation / deactivation of signals in fewer code points as possible. Find the most common locations for each signal change.

## 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 |
| EM | Embedded Machine |
| SIM | Simulator |

# Architecture and Implementation

## Overview

Green light signal – signifies normal printing status:

Off

**Green Light Sequence Machine**

Preprint

Print Completed successfully

Interlock opened

Printing Stopped\*

\*Printing Stopped for any reason

Yellow light signal – signifies warnings demanding user intervention.

Off

**Yellow Light Sequence Machine**

Waste Almost Full or   
 Cartridge replacement is needed

Or Head cleaning wizard is needed

All Issues Solved\*\*\*\*

* When model or support cartridges total reaches 100gr and a warning is posted to the user (while printing)
* printing material inserted
* 500gr from waste carriage limit
* Waste cartridge emptied
* When printing using ABS resin, the system demand to run Head & Wiper wizard every 33 hours, and activate flashing Yellow- ( Only when “Recommend\_HC\_WC\_DuringPrint” is true)

**When parameter “Recommend\_HC\_WC\_DuringPrint” is true:**

Typedef enum

{

WASTE\_IS\_FULL\_OR\_NO\_RESIGN, //0

RECOMMAND\_HC\_WC //1

}TReasonToBlinkYellow;

**Adding new array:**

**bool ReasonToBlinkYellow** [REASON\_TO\_BLINK\_YELLOW]; // does not represent actual HW state

Every time, flashing yellow is needed, updating the list “ReasonToBlinkYellow” list.

Every time need to turn off flashing yellow, need to remove the irrelevant reason and check:

* If list is empty – turn off the yellow flashing.
* If list is not empty – Keep flashing yellow.

Every time pop up a message window for Running HC\_WC – we update the list and turn off the signal tower (Flashing yellow)

Every time the pop up message is closed by application or user interface – updating the list:

**ReasonToBlinkYellow[**RECOMMAND\_HC\_WC**] = false;**

Checking if all items in the list are “false” – turning off yellow flashing.

Red light signal – signifies a stop-printing-fault has occurred during **Pre-Print** and **Printing** states.

Off

**Red Light Sequence Machine**

Preprint -or- Resetting EM app

Printing Error Stoppage

## Embedded Application

1. Controlling ST using ActivateSignalTower() under “CBackEndInterface”.   
   This function takes into account several parameters in CAppParams:
   1. SignalTowerEnabled check box type parameter. (default value is “checked” on Objet1000 and “unchecked” on other machines)
   2. LightDutyOnTime\_ms parameter, which signifies blinking ON time in msec. (default value is 1000).
   3. LightDutyOffTime\_ms parameter, which signifies blinking OFF time in msec. (default value is 1000).
2. Using existing sections of code in EM to control ST state. (ie: Machine Sequencer, Machine Manager) according to state machines shown in section 2.1
3. During power-up sequence all lights will be turned on to validate ST functionality and then turned off.
4. CSignalTower will now remember its state. Since it’s not possible to track the state of each light (HW wise) we can only track the state in which the light should be.  
   We can get ST current state using function: GetLightState(… ), preferably through CBackEndInterface::GetSignalTowerLightState(…)

# Development

## Limitations & Risks

None

## Development Stages

1. All development will be done in Trunk.
2. At first stage the development and testing will be done in Emulation mode.
3. Later, additional testing will be done using simulator
4. As more advances stages of development the testing will be done on an Objet1000 Alpha machine with ST installed.

# Testing

1. Normal tests
   1. During tests in Emulation mode, verify that the application modifies the signals in all states and sequences (specified in 2.1)
   2. Testing with SIM
   3. Testing on actual machine – Objet1000
2. Abnormal tests
   1. Test behavior of application upon changing the value of ‘SignalTowerEnabled’ without restarting / during printing.

# Operation

* Must use new OCB2 card with Silabs C8051F12x controller, and OCB HEX version 30.10 or greater
* SignalTowerEnabled parameter must be checked in Parameters Manger in order for this feature to work.

## Limitations

1. The OCB is not required to give a special signal once losing communication with EM application. This means that upon closing the EM application, the current light signals will remain the same until running the EM again.
2. If ‘SignalTowerEnabled’ is modified to be ‘0’ (/un-checked) at the time when one or more of the signals are ON, the signal(s) will remain “as is” until this parameter is changed again.
3. Activating / deactivating the actuators of the signal tower manually from the ‘Actoators & Sensors’ can cause the ST to not represent the current status correctly.
4. Additional requests for signal activation / deactivation will be added in future phases of ST development.

## Installation

1. Preparations of SW Installation package
   * New EM EXE is compiled
2. This feature will use the following new parameters:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Parameter*** | ***Category*** | ***Type*** | ***Default value*** | ***Meaning*** | ***Attributes*** | ***Exists in files*** |
| ***SignalTowerEnabled*** | Signal Tower | checkbox | true | Define if signal tower is active(existing) | RestartRequired | Q2rt.cfg |
| ***LightDutyOnTime\_ms*** | Signal Tower | integer | 1000 | Signifies blinking ON time in msec | None | Q2rt.cfg |
| ***LightDutyOffTime\_ms*** | Signal Tower | integer | 1000 | Signifies blinking OFF time in msec | None | Q2rt.cfg |

# Effort Estimation

Effort estimation is 8 working days, including implementation and testing (depending on machine availability), according to the following break down:

1. Implementation + unit testing in Emulation mode: EE ~= 6d.
2. Unit testing with SIM: EE ~= 1d.
3. Testing on actual machine
   1. Testing: EE ~= 2d.

Sanity – let machine run with new code…