**Dual Waste**

**Software Design Review**



Written by: Slava Chuhovich

June 2012

Table of Contents

[1. Introduction 3](#_Toc327971830)

[1.1 Background 3](#_Toc327971831)

[1.2 Design Goals 3](#_Toc327971832)

[1.3 Abbreviations and Acronyms 3](#_Toc327971833)

[2. Architecture and Implementation 4](#_Toc327971834)

[2.1 Overview 4](#_Toc327971835)

[2.2 Will not be implemented at this stage 4](#_Toc327971836)

[2.3 User Interface 5](#_Toc327971837)

[2.4 Embedded Application 6](#_Toc327971838)

[2.5 Container Module Explanation 7](#_Toc327971839)

[2.6 Logging 8](#_Toc327971840)

[2.7 Parameters 8](#_Toc327971841)

[2.8 OCB 9](#_Toc327971842)

[3. Development 9](#_Toc327971843)

[3.1 Limitations & Risks 9](#_Toc327971844)

[3.2 Development Stages 9](#_Toc327971845)

[4. Operation 10](#_Toc327971846)

[4.1 Limitations 10](#_Toc327971847)

[4.2 Installation 10](#_Toc327971848)

[5. Effort Estimation 10](#_Toc327971849)

# Introduction

## Background

The purpose is to add additional Waste cartridge to Objet1000 machines, to increase continuous waste capacity and cartridge redundancy. In addition, to introduce “hot swap” mechanism between two waste cartridges.

## Design Goals

1. Add additional load cell sensor and connect both sensors to Cordillia card (MSC).
2. Implement “in place” sensors for both waste cartridges. (MSC card)
3. Implement actuator for additional waste pump. (MSC card)
4. Implement “hot swap” mechanism between 2 waste cartridges.
5. Implement user notifications and printing interaction.
6. Implement Maintenance Counter for additional pump.
7. Implement accuracy calibration for additional load cell. (Calibration dialog and wizard)
8. Implement user interface changes.
9. Implement all needed functionality in the OCB firmware.
10. Implement / expand needed software parameters.

## Abbreviations and Acronyms

|  |  |
| --- | --- |
| **Abbreviation** | **Description** |
| GUI | Graphic User Interface |
| UI | User Interface |
| OCB | Objet Control Board (Main board) |
|  |  |
|  |  |

# Architecture and Implementation

## Overview

1. Both waste cartridges will be handled as any other (model or support) cartridge and as part of a general cartridge handling mechanism.
2. Waste cartridge “hot swap” mechanism will be as Model / Support “hot swap” mechanism, but cartridge with a higher (larger) weight will be active.
3. When one cartridge reaches 17.5Kg (full), it will automatically switch to another.
4. Waste alarm reference will be 17Kg for each load cell. (500 gr below full)
5. Application will alert user and will disable starting of new job when both load cells reach 34Kg.
6. Printing will stop and/or not start when both load cells have reached 35Kg.
   1. A message will be issued to user. (already present)
   2. When waste was emptied below 35Kg, printing can start or resume.
7. All cartridges weight indication on the main UI, will be shown in Kg units when above 10 Kg.
8. Combined and remaining weight of both waste cartridges (if present) will be sent to Host side.

## Will not be implemented at this stage

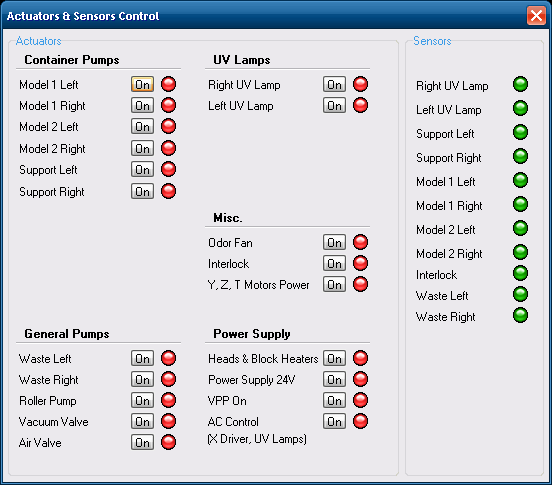
* When printing is in progress and waste weight does not increase in more than TBD after TBD time, the application shall switch to the other cartridge and alert the user “Left/Right Waste cartridge does not fill. Switching to Right/Left cartridge”. (SRS 4.7.2)

## User Interface



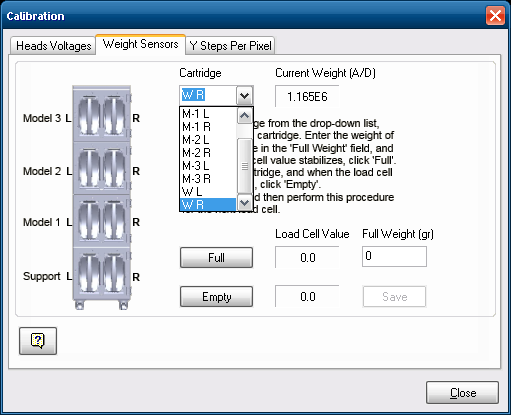
Waste containers will be added to

Main user interface



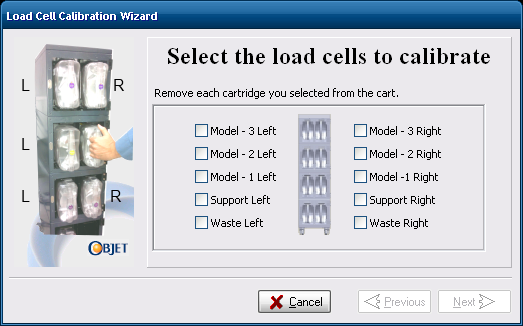
Actuators and sensors for both Right and Left waste

Containers will be added to “Actuators & Sensors Control” dialog



Both Waste containers could be calibrated from

The “Calibration” dialog



Both load cells could be calibrated from the

Load Cell Calibration Wizard

## Embedded Application

1. Concentrate all waste activation mechanism in Container.
2. Implement following methods in Container classes (and additional if needed):
   1. CContainerBase::ActivateWaste(bool Activate)
   2. CContainerBase::ActivateRollerAndPurgeWaste(bool Activate)
   3. CContainerBase::IsWasteFull()
   4. CContainer::GetWasteWeightStatusDuringPrinting()
   5. CContainerBase::GetRemainingWasteWeight()
3. Expose waste activation and notification through Backend interface.
4. Remove all direct waste pump activation by actuators throughout the code and replace with Waste Mechanism activation.
5. Update all cartridges iterations (loops) throughout the code with waste cartridges. Change and adapt logic if needed.
6. Update all required messages to OCB with additional fields for waste tanks.
7. Implement Waste “hot swap” mechanism.
8. Add indexes to required CAppParams parameters and relations.
9. Add additional actuator for Right Waste Pump.
10. Add load cell sensor for Right Waste Tank.
11. Add 2 additional “in place” sensors (micro-switches).
12. Implement “in place” functionality for both waste tanks.
13. Implement Right Waste Tank gain and offset calibration (load cell accuracy).
14. Implement required changes in Load Cell Calibration wizard.
15. Remove user visible references to M3 cartridge (non-existent).
16. Implement required UI changes.
17. Update global tanks and pumps enumerations with additional indexes.
18. Implement weight display in Kg units.
19. Implement and verify all printing related functionality.
20. Implement sending combined cartridges weight to Host.
21. Throw exception when activating waste with full / removed cartridges (during wizards).

## Container Module Explanation

Container is a module in embedded application that is responsible for material (model and support) and waste cartridges operation and logic. The main tasks of the module are:

1. “In place” (micro-switches) functionality for all tanks.
2. Load cell weights for all tanks.
3. RFID state machine and decision making for each tank.
4. “Hot swap” mechanism.
5. Communication with OCB regarding all tanks data (weights, in-places, RF …)
6. Tanks operation modes and activation for each print-block chamber.
7. Parameters updates regarding tanks operation and relations (PumpTankRelation, ChamberTankRelation …)
8. User notification (pop-ups) RFID messages.

Container module consists of a several main classes:

1. CQSingleContainer – Functionality and data for a single cartridge of any type.
2. CContainerBase – Base class for the module.
3. CContainer – Inherits from CContainerBase. Functionality when working with hardware.
4. CContainerDummy – Inherits from CContainerBase. Functionality for Emulation.

CContainer object holds a list of CQSingleContainer objects, which represents each cartridge in the system. All functionality and data collection, such as: weights and in-places receiving from the OCB, “hot-swap” between the cartridges in each pair, RFID state machine for each cartridge…. Is performed by iterating (looping) through the list of Single Containers and performing required logic.

CContainer object runs as a separate thread and sampled each 200 msec.

CContainer

CQSingleContainer

CQSingleContainer

CQSingleContainer

…

CQSingleContainer

CQSingleContainer

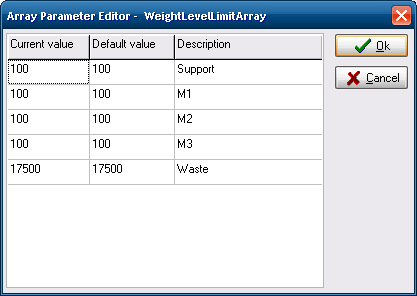
## Logging

Following scenarios will be written to application log file:

1. Activation of each waste pump (ON / OFF).
2. Activation / deactivation of a “waste mechanism”.
3. Insertion / Removal of each waste cartridge.
4. Swapping between waste cartridges in a “hot swap” mechanism.
5. “Waste full” events.
6. Any error or notification messages concerning waste.
7. Activation attempts when both waste cartridges are full / removed.

## Parameters

1. DualWasteEnabled parameter added to Weight Sensors section.
2. Additional indexes were added to following array type parameters to support 2 waste tanks:
   1. ChamberTankRelation
   2. SegmentTankRelation
   3. PumpTankRelation
   4. WeightSensorGainArray
   5. WeightSensorOffsetArray
   6. WeightOfflineArray
   7. WeightLevelLimitArray
   8. ActiveTanks
3. Values of these parameters must be changed if they present in a mode.
4. Default values of existing, required parameters will be changed accordingly.



## OCB

1. Define additional actuator for Right Waste Pump.
2. Define additional Load Cell sensor enumeration.
3. Define 2 additional micro-switch sensors.
4. Map all new definitions to correct ports on a MSC (Cordillia) board.
5. Add additional fields to container related messages, which are sent to Embedded.
6. Populate container related messages with Waste related data.
7. Same OCB firmware project as for Objet500 differentiated with DUAL\_WASTE define. We may have different solution when OCB2 project will be ready.
8. Any additional small code adaptations as required.

# Development

## Limitations & Risks

1. Current OCB card uses Silabs C8051F022 controller, that our OCB firmware code fully utilizes its 4K memory potential.
2. In order to develop Dual Waste mechanism, we’ll need to add additional messages and functionality to OCB firmware that will require more than 4K.
3. To develop Dual Waste mechanism we must use Silabs C8051F122 controller that has 8K of memory, which is a part of OCB2 project.

## Development Stages

1. All development will be done in Trunk.
2. Dual Waste feature will be activated by DualWasteEnabled parameter in Parameter’s Manager. Changing the value of this parameter will require physically reconnecting the waste concerned connectors and burning another OCB firmware (TBD).
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 OCB Simulator application.
5. At more advances stages of development the testing will be done on an Objet1000 Alpha machine.

# Operation

## Limitations

1. Since we use only 2 MSC (Cordillia) cards on the machine, we can connect only 4 pairs of containers (of any kind), meaning that we connect the waste containers instead of M3 model containers.
2. If we’ll want to add 2 additional Model containers to the machine (M1 x2, M2 x2, M3 x2, Support x2, Waste x2), additional MSC card will be needed. In addition, this will require changes in the OCB code.

## Installation

1. Install new OCB2 card with C8051F122 controller.
2. Burn new OCB firmware (HEX file)
3. Install new embedded executable.
4. Change all required parameters in mode files.
5. Connect both waste tanks required connectors to MSC (Cordillia) board.

# Effort Estimation

Effort estimation is 40 working days, including implementation and testing.