# Can Machine

**Thomas Thelin**

## 1. Introduction

This document presents the top level design for the Can Machine. The main purpose of this document is to divide the system into modules which can be individually developed by different development teams.

This document is structured as follows: Section 2 gives an overview of the system and its design. A detailed description of the interfaces in the system is presented in 3. A description of the interaction design is found in section 4, in a MSC notation.

## 2. Design Overview

Figure 1 depicts the three modules in the system. The design of the system constitutes of two software components, `CanHandler` and `MoneyHandler.` Furthermore, the system consists of an `Environment` which represents the user interface.

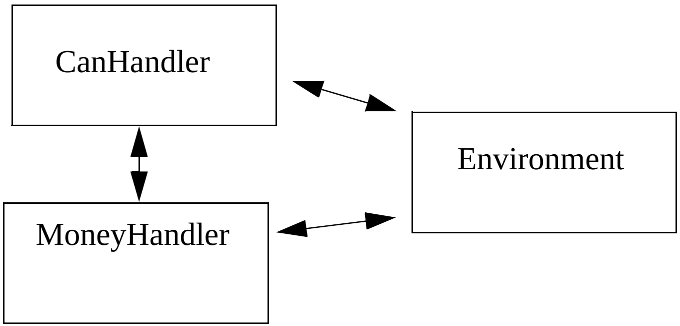


FIGURE 1. System modules

## 3. Interface Description

This section describes the interface to the system. The application interface (API) is described as SDL signals. That is, the design assumes that SDL signals are sent to and from the system.

### 3.1 Environment-MoneyHandler API

#### 3.1.1 Signals from Environment to MoneyHandler

- `Abort` - received when the abort-button is pressed.
- `Coin(x)` - received when a coin with the value `x` has been inserted.

#### 3.1.2 Signals from MoneyHandler to Environment

- `GiveCoins(x)` - coins amounting to `x` is returned.

- `KeepCoins` - the coins inserted in the input slot is dropped in the money reserve.
- `ReleaseCoins` - the coins inserted in the input slot are returned back to the user.

## 3.2 Environment-CanHandler API

### 3.2.1 Signals from Environment to CanHandler

- `Cola` - received when the cola-button is pressed.
- `Water` - received when the water-button is pressed.
- `Beer` - received when the beer-button is pressed.

### 3.2.2 Signals from the CanHandler to the Environment

- `ReleaseCan(aCan)` - a can of type `CanType` is given to the user. `CanType` can be one of the following literals: `colaCan`, `waterCan`, `beerCan`.

## 3.3 MoneyHandler-CanHandler API

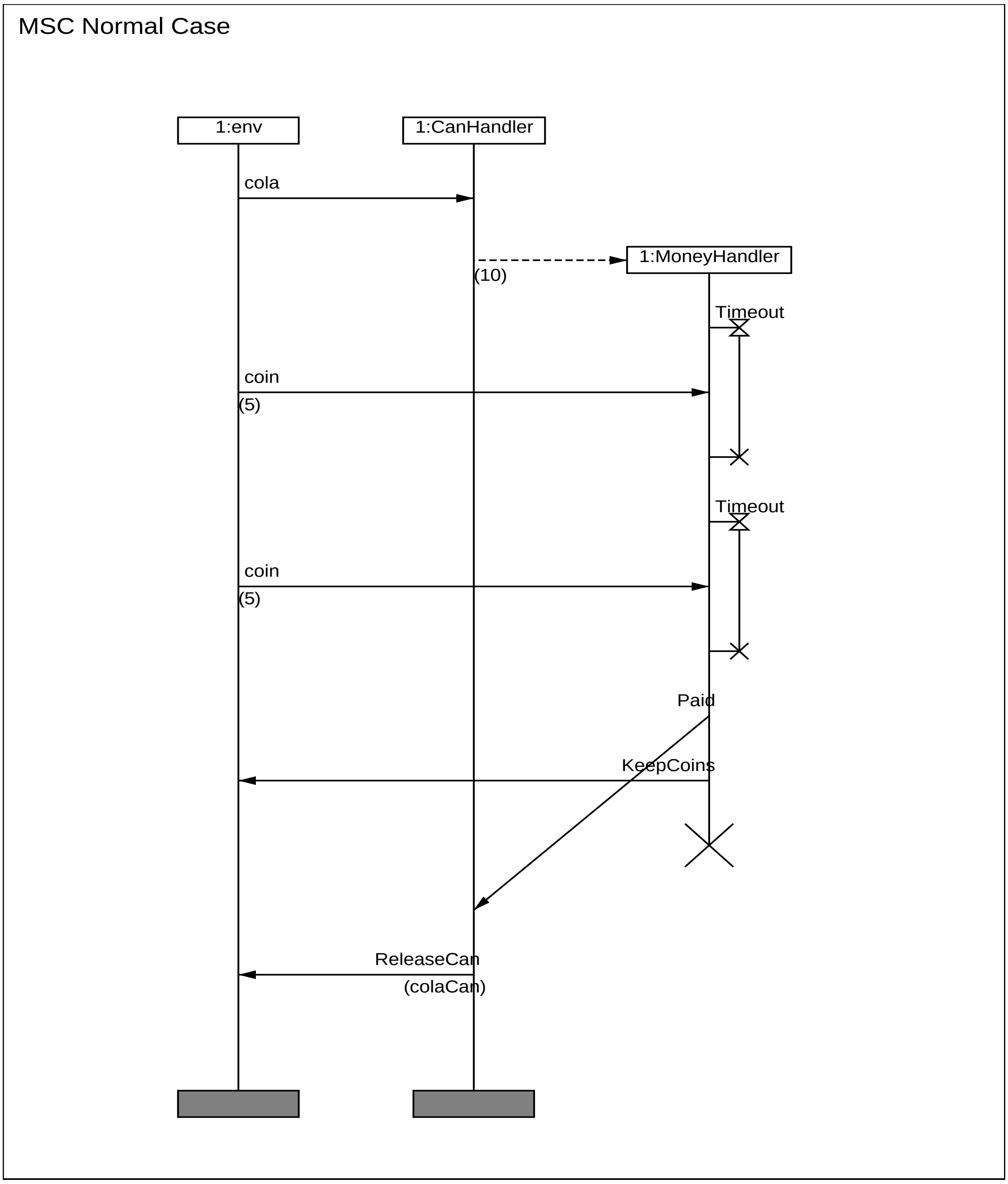
### 3.3.1 Signals from MoneyHandler to CanHandler

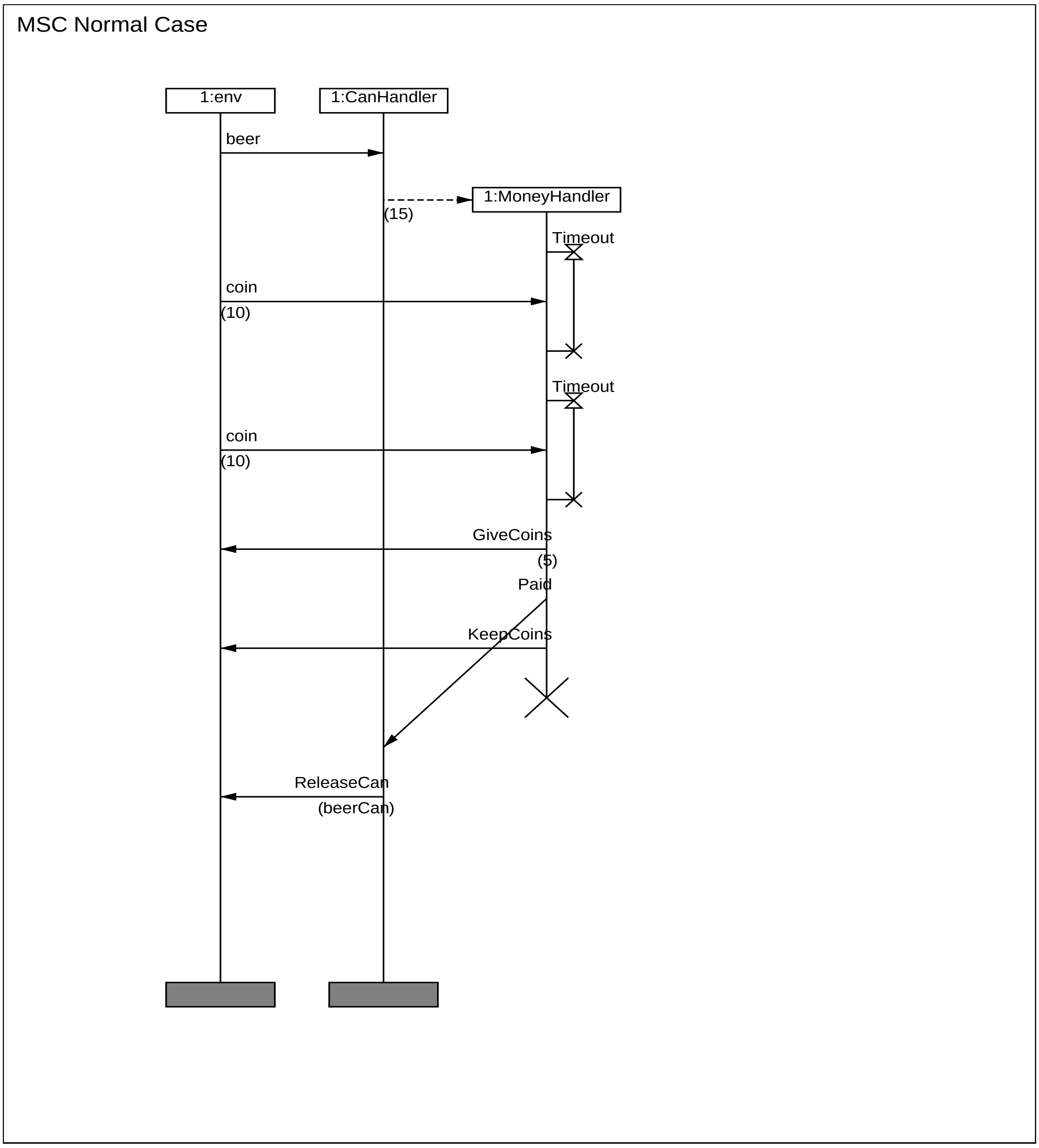
- `Paid` - if enough money is paid with not too much delay.
- `Aborted` - if the user pressed the abort-button or waited too long.

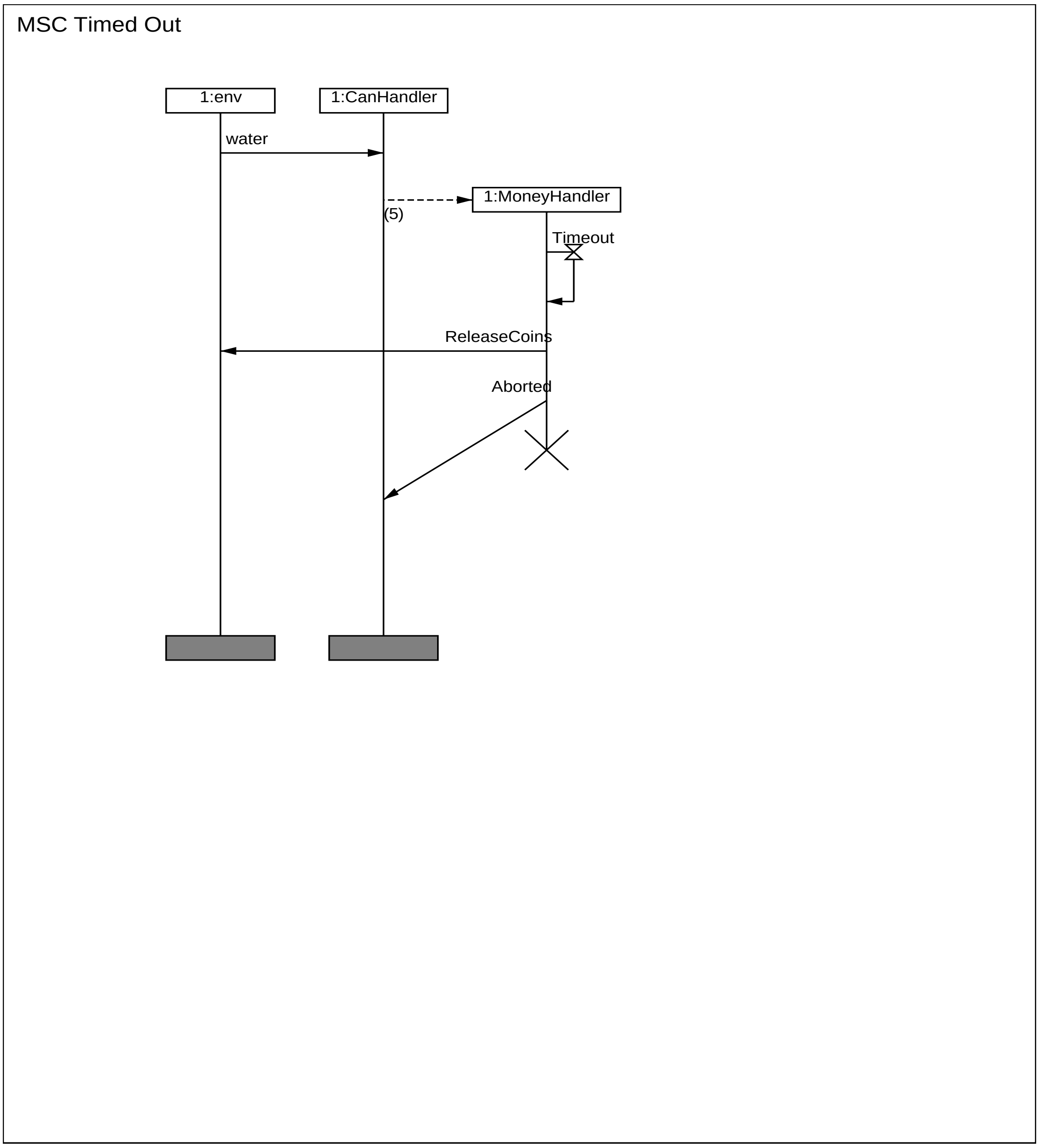
### 3.3.2 Signals from CanHandler to MoneyHandler

No signals are sent from CanHandler to MoneyHandler.

# 4. Message Sequence Charts




MSC Normal Case
1:env
1:CanHandler
beer
1:MoneyHandler
(15)
Timeout
coin
(10)
Timeout
coin
(10)
GiveCoins
(5)
Paid
KeepCoins
ReleaseCan
(beerCan)


MSC Timed Out
1:env
1:CanHandler
water
(5)
1:MoneyHandler
Timeout
ReleaseCoins
Aborted