|  |
| --- |
| Application state diagram |
| Datasheet Application |
| This document will show how the program works. |

Baptiste MORAND

6/28/2016

Contents

[I. Main Function 2](#_Toc454886860)

[A. Description 2](#_Toc454886861)

[B. Function 2](#_Toc454886862)

[II. Task create 3](#_Toc454886863)

[A. Description 3](#_Toc454886864)

[B. Function 3](#_Toc454886865)

[III. TimeProtocolTask 3](#_Toc454886866)

[A. Task 3](#_Toc454886867)

[1. Description 3](#_Toc454886868)

[2. Function 3](#_Toc454886869)

[B. Sender 4](#_Toc454886870)

[1. Description 4](#_Toc454886871)

[2. Function 4](#_Toc454886872)

[C. Receiver 5](#_Toc454886873)

[1. Description 5](#_Toc454886874)

[2. Function 5](#_Toc454886875)

[D. Correction 6](#_Toc454886876)

[1. Description 6](#_Toc454886877)

[2. Function 6](#_Toc454886878)

[E. Correct Offset 7](#_Toc454886879)

[1. Description 7](#_Toc454886880)

[2. Function 7](#_Toc454886881)

[IV. HMI Task 7](#_Toc454886882)

[A. Description 8](#_Toc454886883)

[B. Function 8](#_Toc454886884)

[V. Interrupt 8](#_Toc454886885)

[A. PPS 8](#_Toc454886886)

[1. Description 8](#_Toc454886887)

[B. Internal Timer 9](#_Toc454886888)

[1. Description 9](#_Toc454886889)

# Main Function

int main (void) (main.cpp)

## Description

Call at the beginning of the application, configure peripherals create the task and run the scheduler

## Function

Hmi : void configureHMI(void) (hmi.h)

InternalClock : void configureInternalClock(void)(internalClock.h)

Pps : void configurationPPS(void) (ppsGPS.h)

Network : void configurationNetwork(void) (network.h)

Create Task : void kernelConfig(void) (utask.h)

Start Scheduler : void vTaskStartScheduler(void) (task.h)

# Task create

void kernelConfig(void) (utask.h)

## Description

Create the task, the task will be manage by the scheduler :

* Task HMI is priority 1 (low)
* Task TimeProtocol is priority 3 (high)

## Function

**Hmi task :** void HMITask(void) (hmi.h)

**Time Protocol task :** void timeProtocolTask(void) (timeProtocol.h)

# TimeProtocolTask

## Task

### Description

This task will be running in a loop, and consist in 3 function to sending receiving and make the correction.

### Function

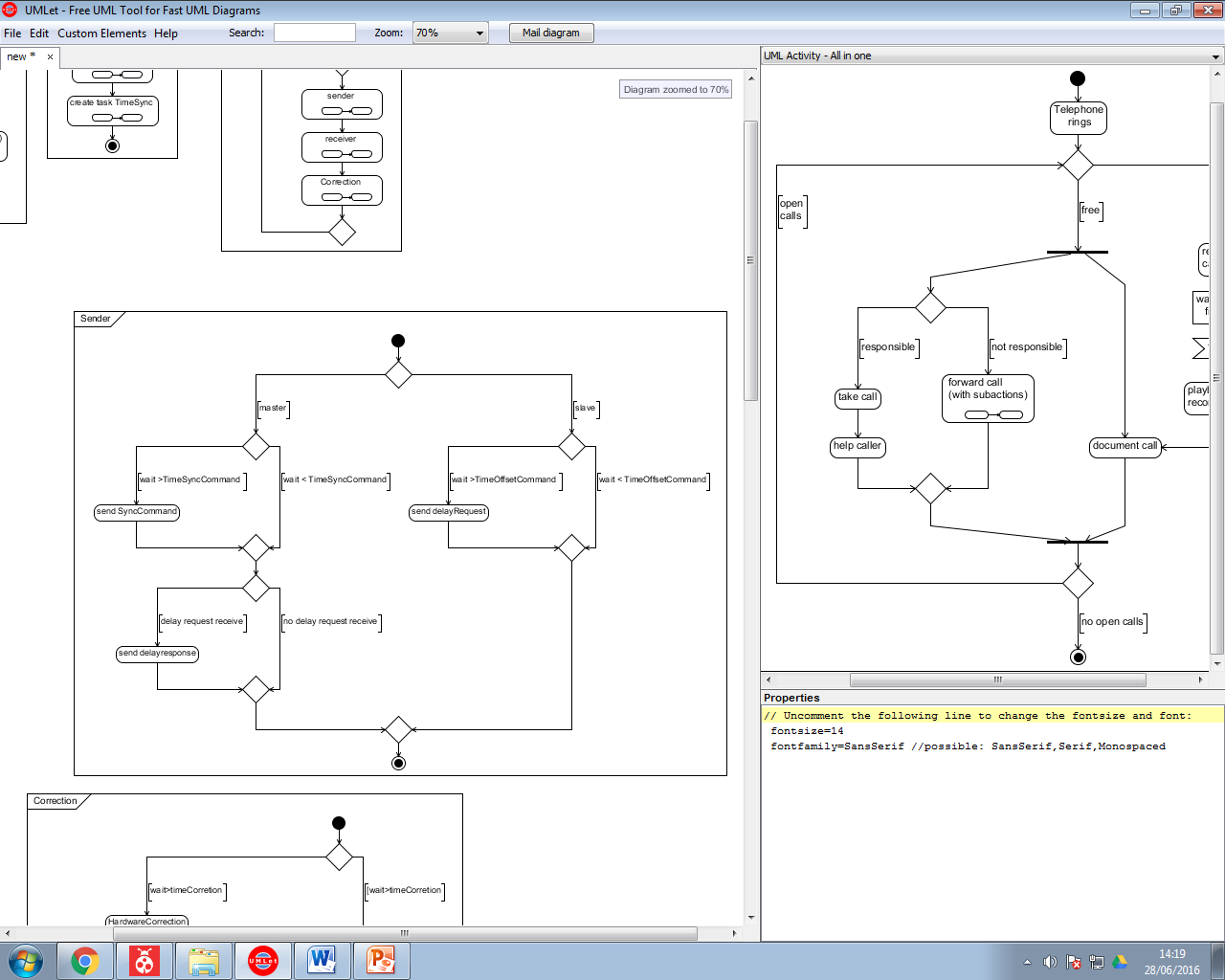
**Function receiving :** void receiver(void) (timeProtocol.h)

**Function sending :** void sender(void) (timeProtocol.h)

**Function correction:** void correction(void) (timeProtocol.h)

## Sender

void sender(void) (timeProtocol.h)



### Description

In function of the node it will send the correct command, all command are trigger by a time or an event.

Trigger event :

* **Sync Request :** time define in conf\_time\_protocol.h manage by the time of the RTOS
* **DelayRequest :** time define in conf\_time\_protocol.h manage by the time of the RTOS
* **Delay Response :** event declare if timeRequest is receive

### Function

**Send Sync command :** void sync(void) (timeProtocol.h)

**Send Delay Request :** void delayRequest(void) (timeProtocol.h)

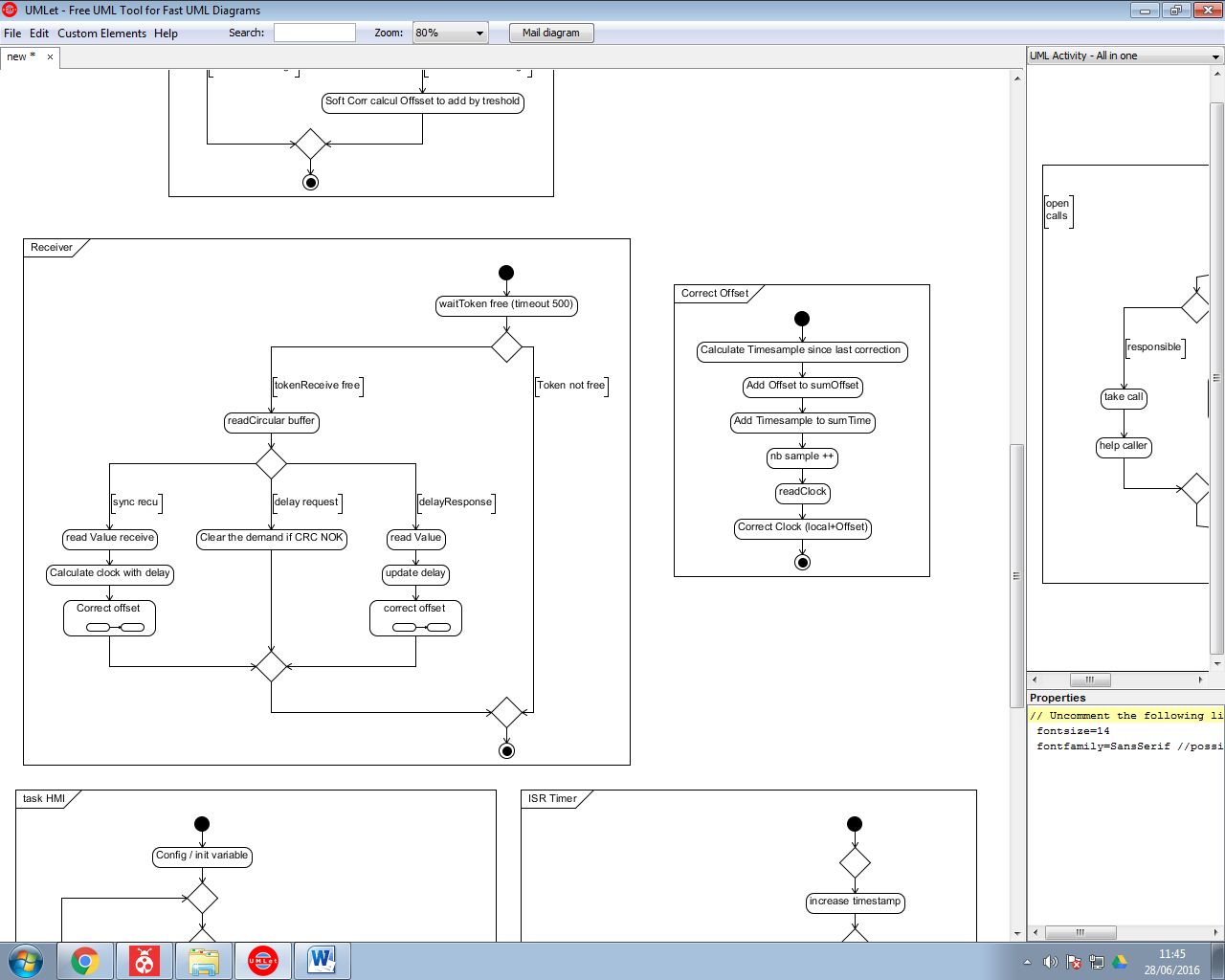
**Send Delay Response :** void delayResponse(uint8\_t id) (timeProtocol.h) with id the id of the slave

**Time sync :** #define TIMESYNC (conf\_timeProtocol.h)

**Time Delay Request :** #define TIMEDELAYREQUEST (conf\_timeProtocol.h)

## Receiver

void receiver(void) (timeProtocol.h)



### Description

The token will be free in an interrupt to synchronise the task with the reception (<http://www.freertos.org/binary-semaphore.gif>).

After he will parse the command and decide what he have to do.

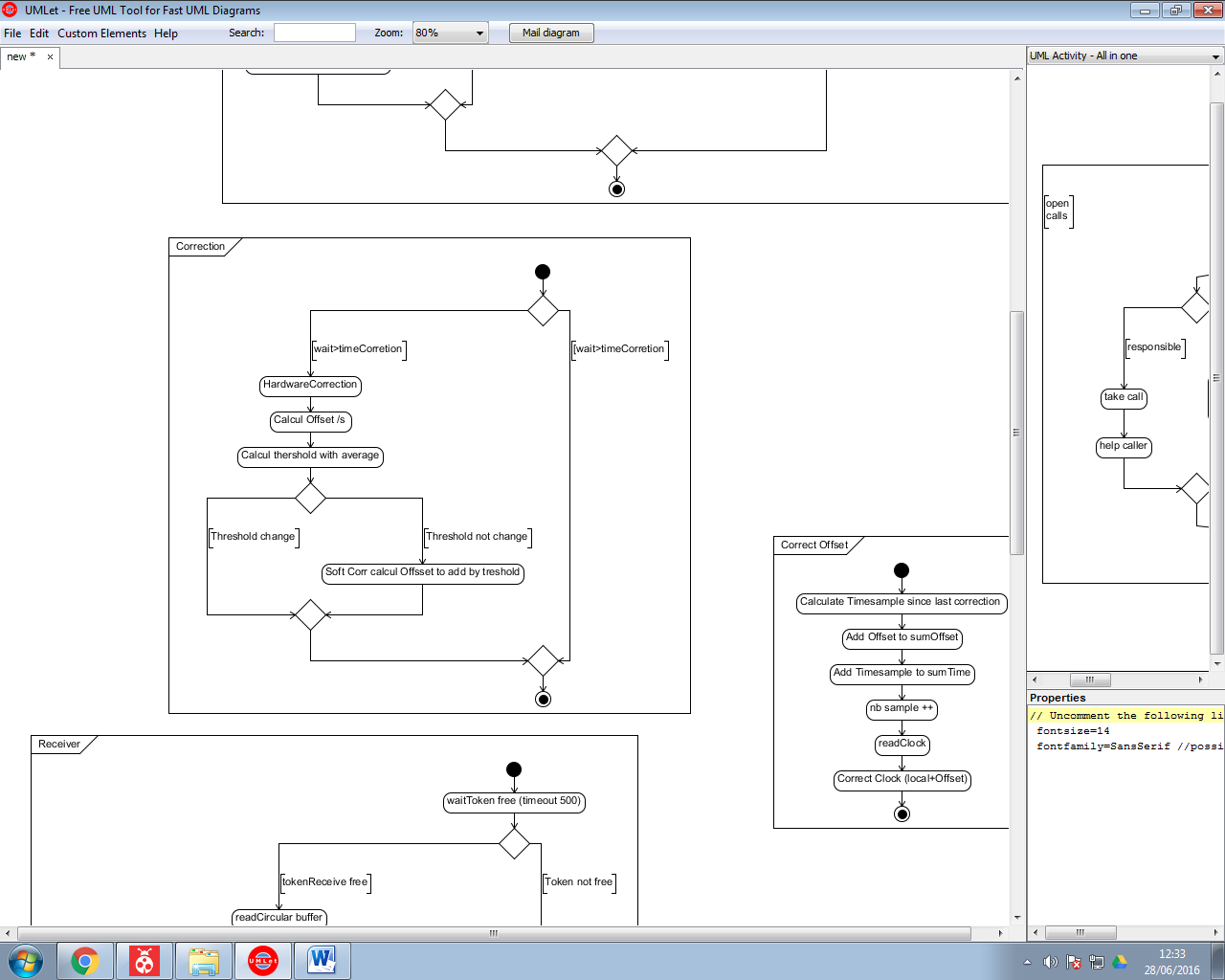
### Function

All the code are in void receiver(void) (timeProtocol.h)

**Function correctOffset :** void updateClock(void) (timeProtocol.h) the current offset is a global value. (timeprot.offset)

## Correction

void correction(void) (timeProtocol.h)



### Description

Correction is trigger by a time or an event.

The programme have 2 things to correct :

* Hardware Correction : Hardware correction will change the threshold see part internalClock.
* Software Correction will correct the clock every n threshold .

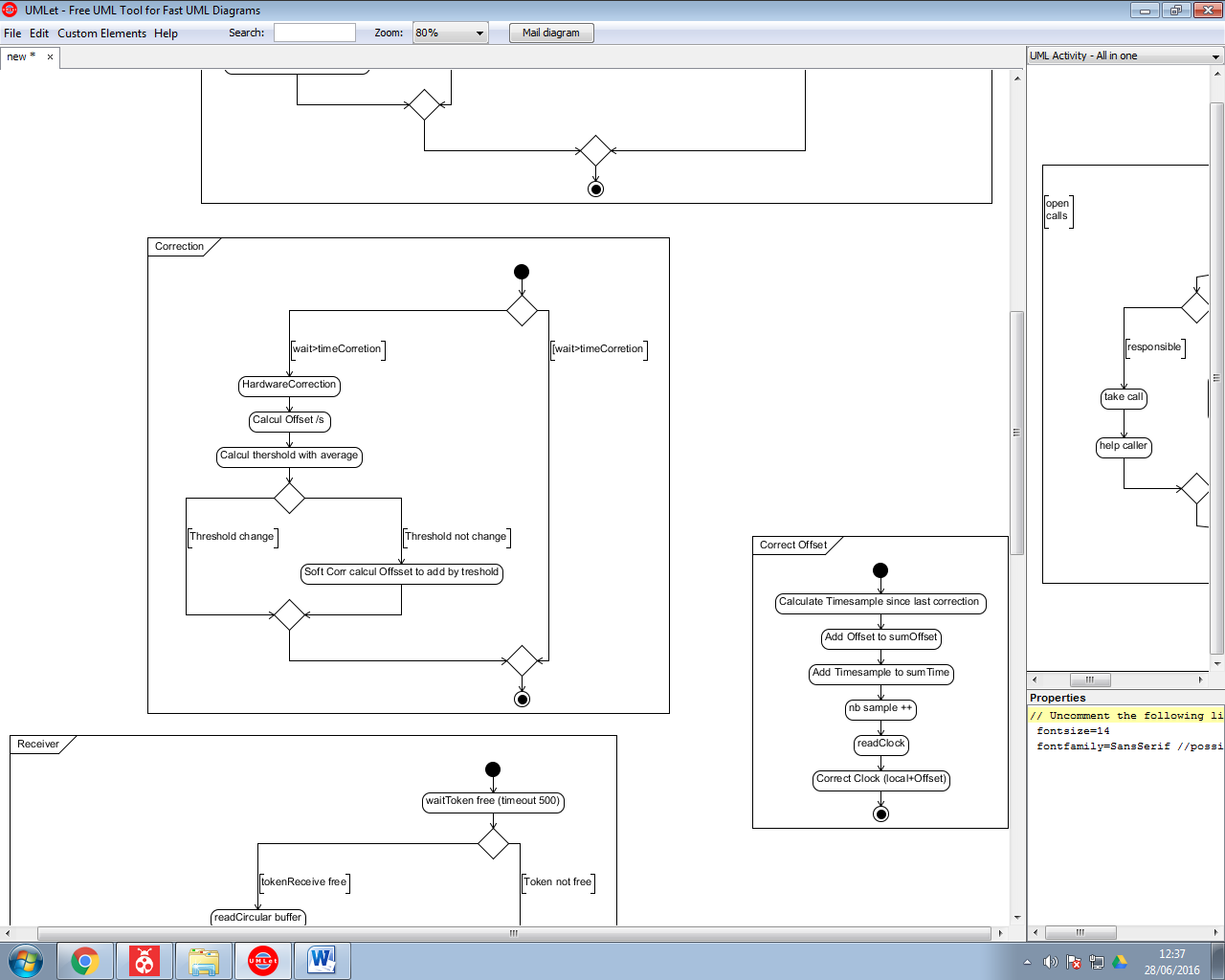
If Hardware Correction is made you have to disable software correction up to the next correction.

### Function

**Time Correction :** #define TIME\_CORRECTION (conf\_timeProtocol.h)

## Correct Offset

void updateClock(void)



### Description

For the correction he will save the offset and the timeSample. And after he will correct his clock.

### Function

**Read clock :** uint32\_t readClock(Clock\* timeClock) (internalClock.h)

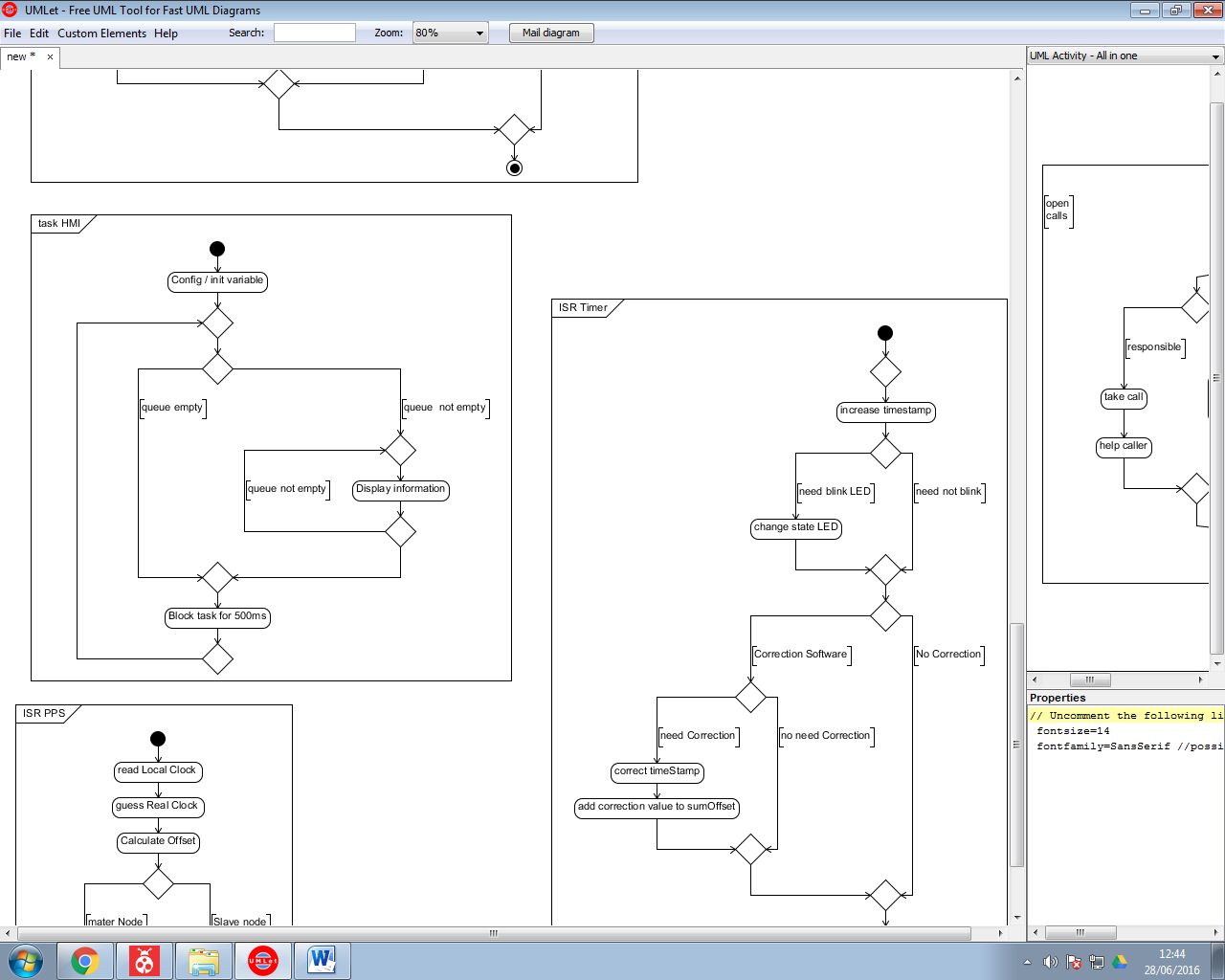
**Sum Offset :** sumOffset (timeProtocol.h)

**Sum Time :** timeProt.correction.sumTime(timeProtocol.h)

**Nb sample :** timeProt.correction.nbCorrection (timeProtocol.h)

# HMI Task

void HMITask(void) (hmi.h)



## Description

The HMI send to a UART communication the data in the queue.

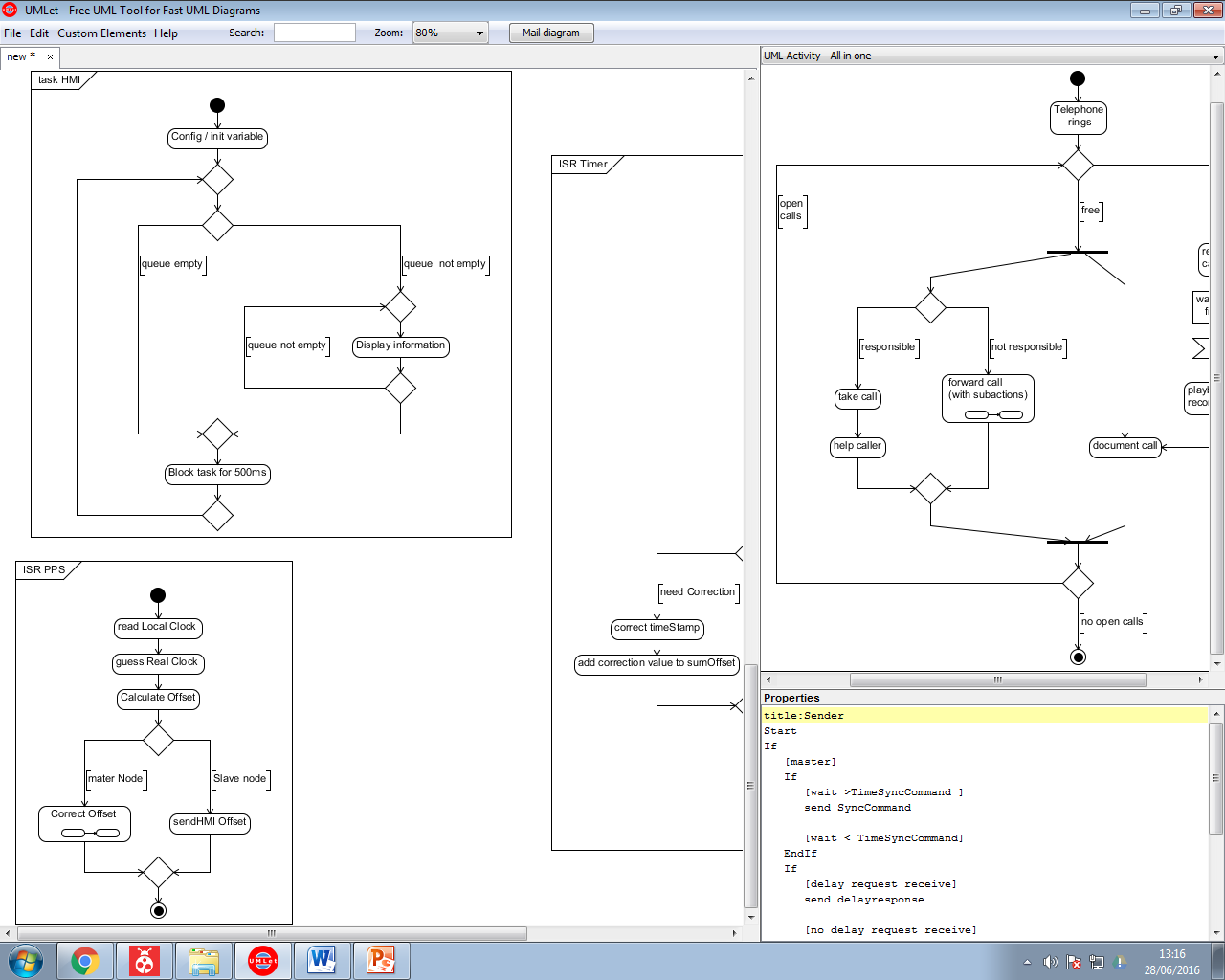
## Function

**HMI queue :** xQueueHandle uartQueue (hmi.h)

# Interrupt

## PPS

void ppsISR(void)

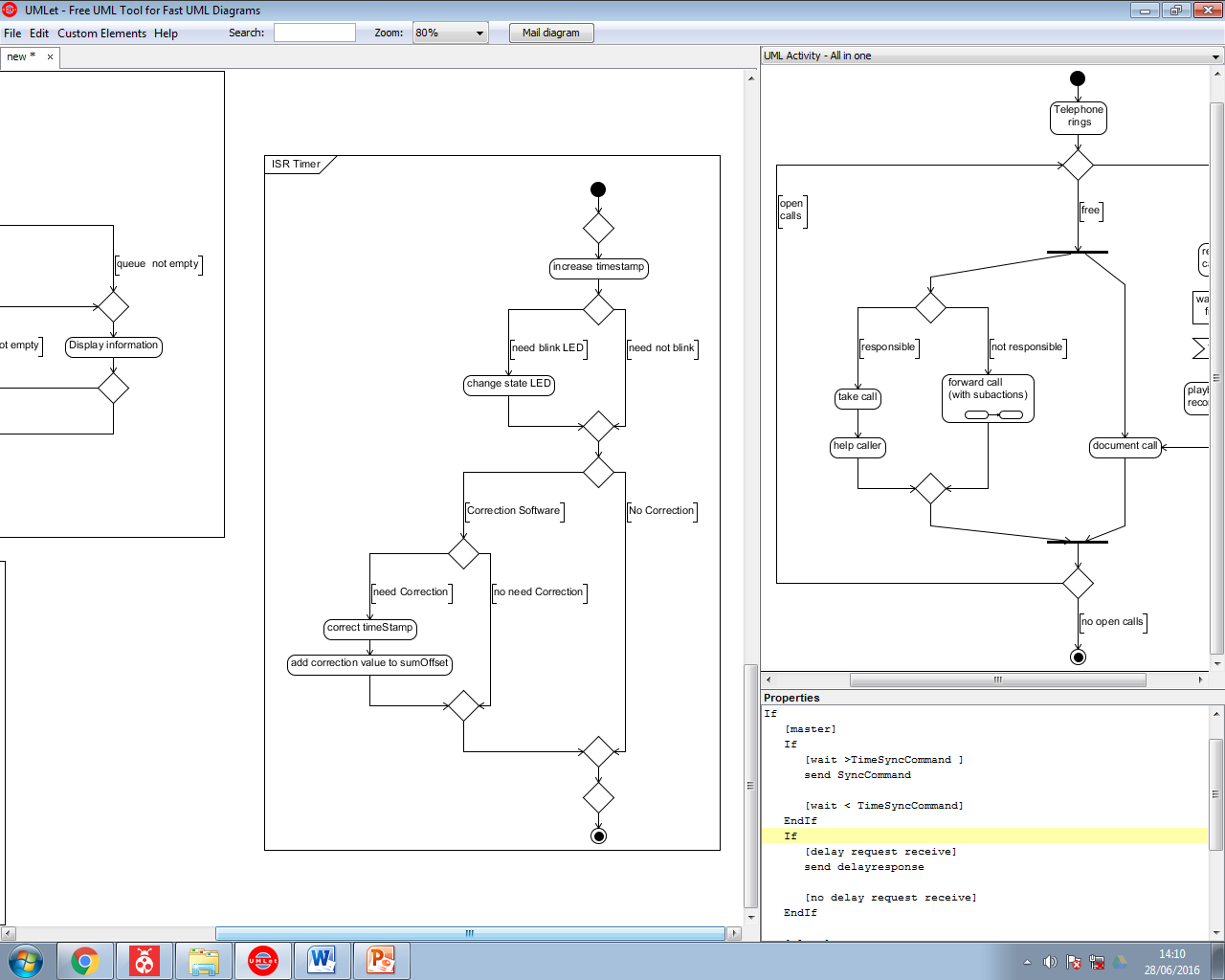


### Description

PPS will calculate the offset and print it for a Slave node or correct it if it is a master node.

## Internal Timer

void isrInternalClok(void )



### Description

This interrupt will manage the led, and the timestamp. If there is a Software correction he will add the correction to the timestamp.