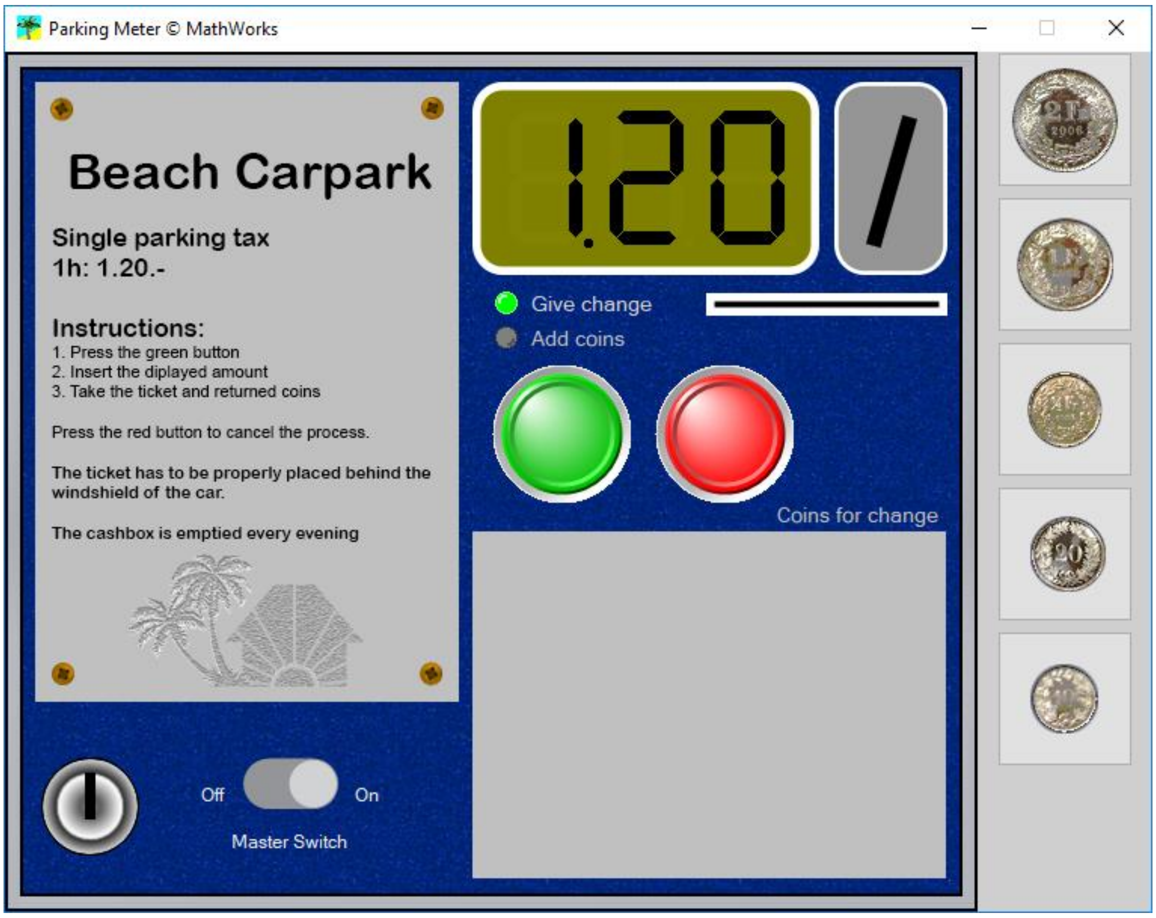
Parking Meter Specifications

Version 1.2



Beach Carpark

Single parking tax
1h: 1.20.-

Instructions:
1. Press the green button
2. Insert the diplayed amount
3. Take the ticket and returned coins

Press the red button to cancel the process.

The ticket has to be properly placed behind the windshield of the car.

The cashbox is emptied every evening

Contents

Tasks to accomplish

This document specifies the needed work to do in order to develop and implement the program to control the new parking meter that will be produced this year.

1.Introduction

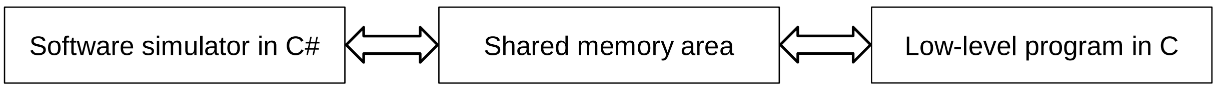
For a new parking meter being developed it is needed to produce the program that will operate it and that will be downloaded onto the embedded target. As the first Hardware prototype of this parking meter is not yet ready, and so that it is possible to test the program during its development, a Software simulator has been created. Moreover, the following C Software library must be used to access the Hardware registers from the electronics board:

```
unsigned char read_register(unsigned short address);
void write_register(unsigned short address, unsigned char value);
```

-1- Low-level interface C functions

2.Interface to the Simulator

The way data are exchanged between the Software simulator in C# and the low-level program in C is done via a shared memory area that is created in the RAM of a Windows 10 64bit machine.



-2- Software simulator interface diagram

As soon as the program can be deployed onto the real HW, the interface layer between the low-level C program and the shared memory area will have to be replaced/updated so that it can communicate properly with the real HW registers of the board.

3.Usage of the Simulator

The following actions with their corresponding effects are available through the Software simulator:

Action	Effect
Click on the Master Switch	If it is 'ON', the Parking Meter GUI can be used normally. If it is 'OFF', the low-level program stops properly
Click on the Keyhole	Reload the cashbox that can return coins
Click on the area that gives coins back	Take the money and emptied the area
Click on the parking ticket	Take the ticket
Click on the coins on the right-hand side	Simulate the insertion of the selected coin

-3- List of possible interactions with the GUI

4.Hardware Architecture of the Parking Meter

The parking meter is made of several Hardware registers to communicate with the electronical and mechanical parts from the application's program.

Register Address in hexadecimal	Functionality	
0xA000	Seven Segments Display Number 1*	
0xA001	Seven Segments Display Number 2*	
0xA002	Seven Segments Display Number 3*	
	* The description of the segments encoding for the displays is given at the §7.1.	
0xA003	Give Coins Change Back	
	Bit 0	Give a coin of 10 cents back**
	Bit 1	Give a coin of 20 cents back**
	Bit 2	Give a coin of 50 cents back**
	Bit 3	Give a coin of 1 franc back**
	Bit 4	Trigger the printing of the parking ticket**
	Bit 5	Green LED "Give change"
	Bit 6	Red LED "Add coins"
	Bit 7	Open and close the coin's flap
	** To give a coin back, or to trigger the printing of the ticket, it is needed to generate a pulse of a duration equal or bigger than 100ms on the corresponding bit's entry	

-4- Output registers that can have data written by the program

Register Address in hexadecimal	Functionality	
0xA010	Get Buttons and Switch Status	
	Bit 0	Status of the green button
	Bit 1	Status of the red button
	Bit 2	Status of the master switch
0xA011	Get the Type of Coin Inserted	
	Bit 0	10 cents coin*
	Bit 1	20 cents coin*
	Bit 2	50 cents coin*
	Bit 3	1 franc coin*
	Bit 4	2 francs coin*
	Bit 5	(N/A)
	Bit 6	(N/A)
	Bit 7	(N/A)
	* The bit corresponding to the inserted coin is set to '1'	
0xA012	Get the Type of Coin that can be Returned	
	Bit 0	10 cents coin**
	Bit 1	20 cents coin**
	Bit 2	50 cents coin**
	Bit 3	1 franc coin**
	Bit 4	(N/A)
	Bit 5	(N/A)
	Bit 6	(N/A)
	Bit 7	(N/A)
	** The bit corresponding to the type of coin indicates if there is at least one coin of this type remaining in the return coins box. If one coin's type is emptied, the change cannot always be fully given back	

-5- Input registers that can have data read by the program

5. Test Functions

At first, the test functions of the program must be developed to test that the electronics of the parking meter is working correctly. In this test mode, a choice menu will be displayed in the console view. The menu must provide the following functions allowing to test the input and output registers of the parking meter:

1) Write into a register by providing its address and the new value to write
2) Display an amount of money (input value to get) on the three 7-segments displays
3) Switch on and off the green LED “Give change”
4) Switch on and off the red LED “Add coins”
5) Give coins of 10, 20, 50 cents and 1 franc back
6) Print a parking ticket
7) Open and close the coin’s flap
8) Display the status of the red and green boutons as well as the introduced coins on a new line of the console every 0.1 seconds for 10 seconds

The menu must also have an entry allowing to call the standard mode of operation of the parking meter once it will be available.

6. Normal Mode of Operation

In the normal mode of operation, there is no interactive console user interface. The program handles the parking customers interactions via the buttons, displays and peripheral components. The program must provide the following capabilities:

1) When the user is pressing on the green button, the amount of the parking tax must be displayed
2) After each inserted coin, the remaining amount to pay must be displayed
3) Once a big enough amount of money has been inserted, the printing of a parking ticket must be triggered et change must be given back to the user
4) If the user is pressing the red button, the ticket purchase must be aborted. The display must come back to its initial empty state. The inserted coins must be given back to the user
5) After each purchase or cancellation process, the LEDs status must be updated to indicate if the parking meter can still give change back
6) If some type of coins is missing, the change should still be given back wherever possible

7. Techniques of Implementation

The implementation consists of using graphical programming techniques to implement concurrent code execution in Simulink through a time scheduler in Stateflow and it has to be fully tested. The final step is to generate C code automatically and to build it has a standalone executable that can run under Windows 10 64bit.

For the Model-Based Design implementation, the architecture of the model should be as following:

- The program has to execute at a rate of 1ms that represents the base system clock rate
- The model must run for an infinite amount of time and it must be stopped only if the “Main Switch” is set to the OFF position
- As the model runs in concurrent execution mode and modules of the model will access to shared resources (read and write functions), there is a need to use a main time scheduler to schedule the access to them. Software interrupts will be generated to trigger the execution of the needed algorithms at the right time. The figure -6- shows how interrupts are mapped to modules
- All steps of the parking ticket purchase must be organized in a state machine and the algorithms executed from them must be implemented into separated modules/blocks
- A low-level interface layer must be implemented to setup the communication with the HW registers. Ideally a library should be used for this. This layer must contain the access to the following functions: *read_register()* and *write_register()*. The goal of this layer is to be flexible regarding the implementation of the algorithm onto different platforms. The algorithm must be target independent and only those functions must be adapted

- Constants used in the model must be defined in a Simulink Data Dictionary so that it provides enough flexibility when it comes to C code generation
- The Normal Mode of Operation as well as the Test Mode must be implemented and tested in Simulink. The testing of the various modules should be done using test harnesses in Simulink Test to do proper unit testing
- C code generation must be fully usable out of the Simulink model for the "Accelerator Mode" and standalone executable deployment onto a Windows 10 64bit Desktop machine

Interrupt Service Routines	Modules to execute
ISR_0	Stop the execution of the Model/Executable
ISR_1	Write values to the seven segments displays
ISR_2	Switch the state of the coin's flap
ISR_3	Write data to the available coins box
ISR_4	Write data to the peripherals and generate pulse
ISR_5	Read buttons status
ISR_6	Read status of the inserted coins

-6- Interrupts mapping table

7.1. System Composer

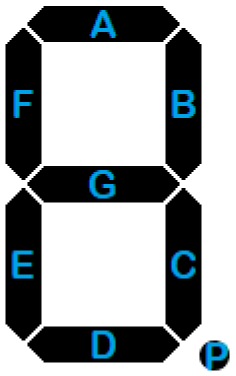
For this Model-Based Design implementation, a system layer can be put on top of the Simulink implementation. It is made of components representing the Human Machine Interface (HMI), the Scheduler and the Termination of the application. These three basic components are connected together via buses for: user's inputs, application's outputs, system events.

8. Annexes

This section contains detailed descriptions of specific HW components/systems to better design the algorithm on the SW side.

8.1. Seven Segments Display Encoding

The figure -5- shows how the segments are labeled so that they can be encode correctly.

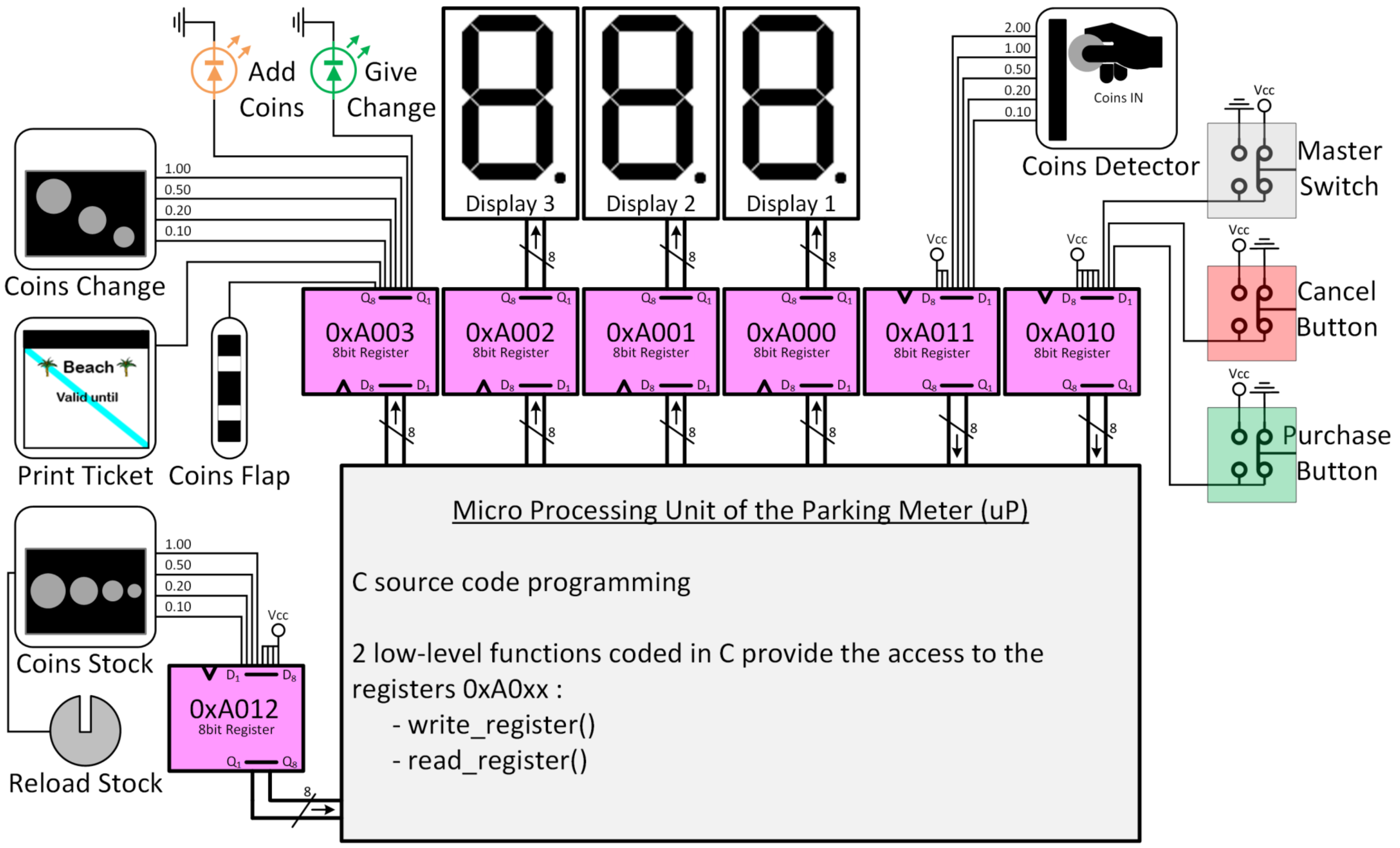


-7- Seven Segments Display Encoding

The encoding of the 8bit value to write in the register for the corresponding 7 segments display is given has followed in binary: **0bABCD'EFGP**.

Therefore, if one of this bit is set to '1', it switches the corresponding segments in the 7 segments display on and if one of this bit is set to '0', it switches the corresponding segments off.

8.2. Block Diagram



-8- Block Diagram of the Parking Meter system