Real-Time Software Development

Metro Train Controller

Selva Stefano
Gatto Marco

Project Specifications

The specifications involve the development of a real-time software that is able to control some basic actions concerning a Metro train.

The system is supposed to run on a STM32F10x microcontroller.

System Requirements

The main requirements that the system has to meet are:

1. Activate/Deactivate the regular brake commands issued by the driver
2. Increase/Decrease the speed according to the driver's commands
3. Enforce the stop commands issued by a stop signal
4. Respond to emergency brake requests
5. Manage the communications from the train traffic managers to the driver

Specifications: commands issued by the driver

The specifications concerning the requirements 1 and 2 can be treated as if they were a single one.
All these commands are issued by the driver through a sliding lever. In particular, each command is related to a specific position of the lever and they can be represented as follows:

- +3: maximum acceleration
- +2 : medium acceleration
- +1 : minimum acceleration
- 0: no acceleration and no braking
- -1 : minimum braking
- -2 : medium braking
- -3 : strong braking

Specifically, the position 0 of the lever is defined as the 'IDLE' position.
The position of the lever is communicated to the software controller through a set of pins of the General-Purpose Input Output port B. In particular, the inputs issued by the lever must be received according to the following table:

Input	Source	Type
Maximum acceleration	GPIOB Pin8	Bit
Medium acceleration	GPIOB Pin7	Bit
Minimum acceleration	GPIOB Pin6	Bit
IDLE position	GPIOB Pin5	Bit
Minimum braking	GPIOB Pin4	Bit
Medium braking	GPIOB Pin3	Bit
Strong braking	GPIOB Pin2	Bit

In order to ensure a correct performance of the regular operations, these commands must be read with a frequency of 100Hz (or equivalently a read operation every 10ms).

Once the controller receives a command issued through the lever, it is expected to react in a consistent way. Specifically, it has to communicate to both the engine and the braking system the commands issued by the driver. Furthermore, the controller must never issue a request to accelerate to the engine and a request to brake to the braking system at the same time.

In normal conditions, the position of the lever is simply converted in the corresponding acceleration and braking outputs. These commands are communicated to both the engine and the braking system through a

set of pins of the General-Purpose Input Output port C. In particular, the controller must issue the outputs according to the following table:

Output	Exit	Type
Maximum acceleration	GPIOC Pin2	Bit
Medium acceleration	GPIOC Pin1	Bit
Minimum acceleration	GPIOC Pin0	Bit
Minimum braking	GPIOC Pin8	Bit
Medium braking	GPIOC Pin9	Bit
Strong braking	GPIOC Pin10	Bit

When the lever is in the 'IDLE' position, the controller must ask the engine not to accelerate and the braking system not to brake. Specifically, it is converted with all the pins of the GPIOC port switched off.

For safety reasons, the controller also has to check whether two conditions are verified:

1. If the maximum acceleration pin has been enabled for more than 4s, the controller must ask the engine to set the power to medium acceleration, even though the lever is still in position +3.
2. When the lever switches its position from position A to position B, the input provided to the controller is temporarily empty. In this case, the controller must assume that the provided input is the last one, and so it is not supposed to change the outputs. However, the controller must check whether this condition holds for more than 3s. consecutively. In this case, it very likely means that something wrong is happening with the lever, and so the controller must enable the braking system with medium braking power, wait the train to be stopped and then wait indefinitely for a manual reset of the system.

Specifications: stop signal management

The requirement 3 assumes that there is a button allowing someone to issue a stop signal to the metro train. This request must be caught and managed by the controller. Furthermore, the signal generated by the pressed button is asynchronous and unpredictable, and so the system must be able to react accordingly.

Specifically, this button communicates with the controller through the Pin 1 of the GPIOB port.

When the stop signal is detected, the train must be stopped "gently". In particular, the system must ask the engine to disable the acceleration power and the braking system to brake with medium braking power. After the train has been stopped, the controller must wait both the stop signal to be cleared and the lever to be put in idle position. All the inputs issued by the lever must be ignored until the two conditions previously defined are not satisfied. Only after these conditions are met, the normal situation can be restored and the train can restart the journey.

The output expected when a stop signal is enabled is:

- GPIOC Pins 0, 1, 2 switched off: no acceleration
- GPIOC Pin 9 switched on: medium braking power
- All other GPIOC Pins must be switched off

Specifications: emergency braking management

The requirement 4 assumes that there is a button allowing someone to issue an emergency braking signal to the metro train. This request must be caught and managed by the controller as soon as possible. Furthermore, signal generated by the emergency braking button is asynchronous and unpredictable, and so the system must be able to react accordingly.

Specifically, the emergency braking button communicates with the controller through the Pin 0 of the GPIOB port.

When the emergency braking signal is detected, the train must be stopped as soon as possible. In particular, the system must ask the engine to disable the acceleration power and the braking system to brake with maximum braking power. For this reason, the Pin 11 of the GPIOC port must be configured and enabled to communicate the braking system that an emergency braking is occurring and the train must be stopped with the maximum braking power. This situation must be held indefinitely until the system is manually reset.

The emergency braking request has the maximum priority and the system must satisfy it immediately, whatever the current state of the train and the system.

When an emergency braking signal is enabled, the expected output is:

- GPIOC Pins 0, 1, 2 switched off: no acceleration
- GPIOC Pin 11 switched on: (maximum) emergency braking power
- All other GPIOC Pins must be switched off

Specifications: communication messages management

The requirement 5 assumes that there is a traffic management center that is able to send communication messages to the driver. In particular, the controller must be able to receive some messages from the traffic management center via a serial line and then send the received messages to a driver display via another serial line.

Specifically, the traffic management center communicates with the controller through the USART1 (GPIOA) serial line, while the controller communicates with the driver display through the USART2 (GPIOA) serial line.

The communication messages management has the lowest priority, and so it should be performed only when there are no other actions to perform.

In particular, whenever a communication message is received, the system must check whether there are some other tasks to manage, and only if the previous condition is not satisfied it can forward the message to the driver display.

Other Specifications: input simulation

It can be useful to add a task is able to generate events with the same meaning as the expected inputs.

Summary table of inputs and outputs

Input	Source	Exit
Maximum acceleration	GPIOB Pin8	GPIOC Pin2
Medium acceleration	GPIOB Pin7	GPIOC Pin1
Minimum acceleration	GPIOB Pin6	GPIOC Pin0
IDLE position	GPIOB Pin5	-
Minimum braking	GPIOB Pin4	GPIOC Pin8
Medium braking	GPIOB Pin3	GPIOC Pin9
Strong braking	GPIOB Pin2	GPIOC Pin10
Stop signal	GPIOB Pin1	GPIOC Pin9
Emergency braking	GPIOB Pin0	GPIOC Pin11
Communication messages	USART1 (GPIOA)	USART2 (GPIOA)

Project Design

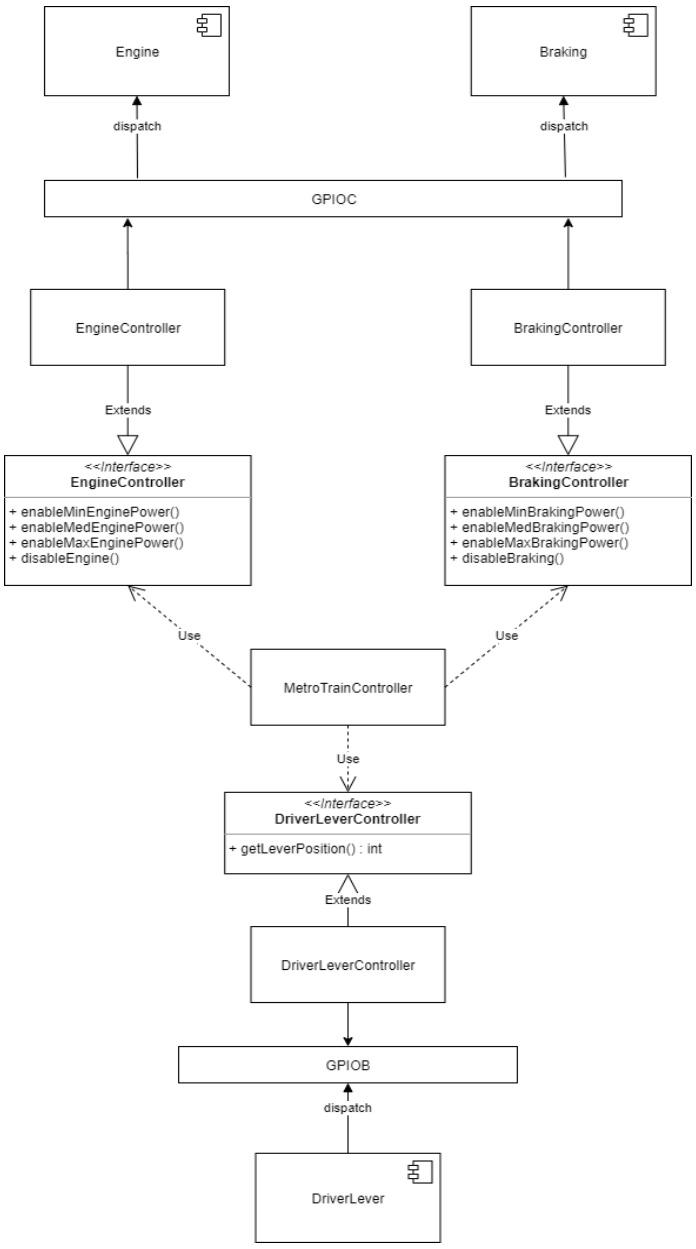
In order to design and implement the system in a straightforward way, an incremental approach is adopted. In particular, according to the specifications, the system design is split into 5 incremental steps:

1. Driver lever management
2. Stop signal management
3. Emergency braking management
4. Communication messages management
5. Input simulation and test data

Design: driver lever management

The step 1 involves the design of a system that only has to manage the inputs issued by the driver lever.

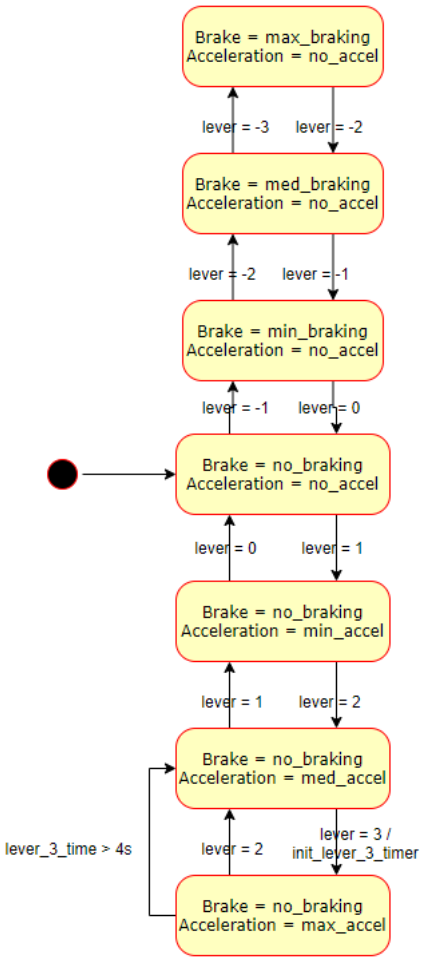
The physical system can be described with the following component diagram:



According to the specifications, this diagram shows how the components should interact each other. In particular, the messages dispatched by the DriverLever through the GPIOB port must be caught by the

software controller and then consistently forwarded to both the engine and the braking system through the GPIOC port.

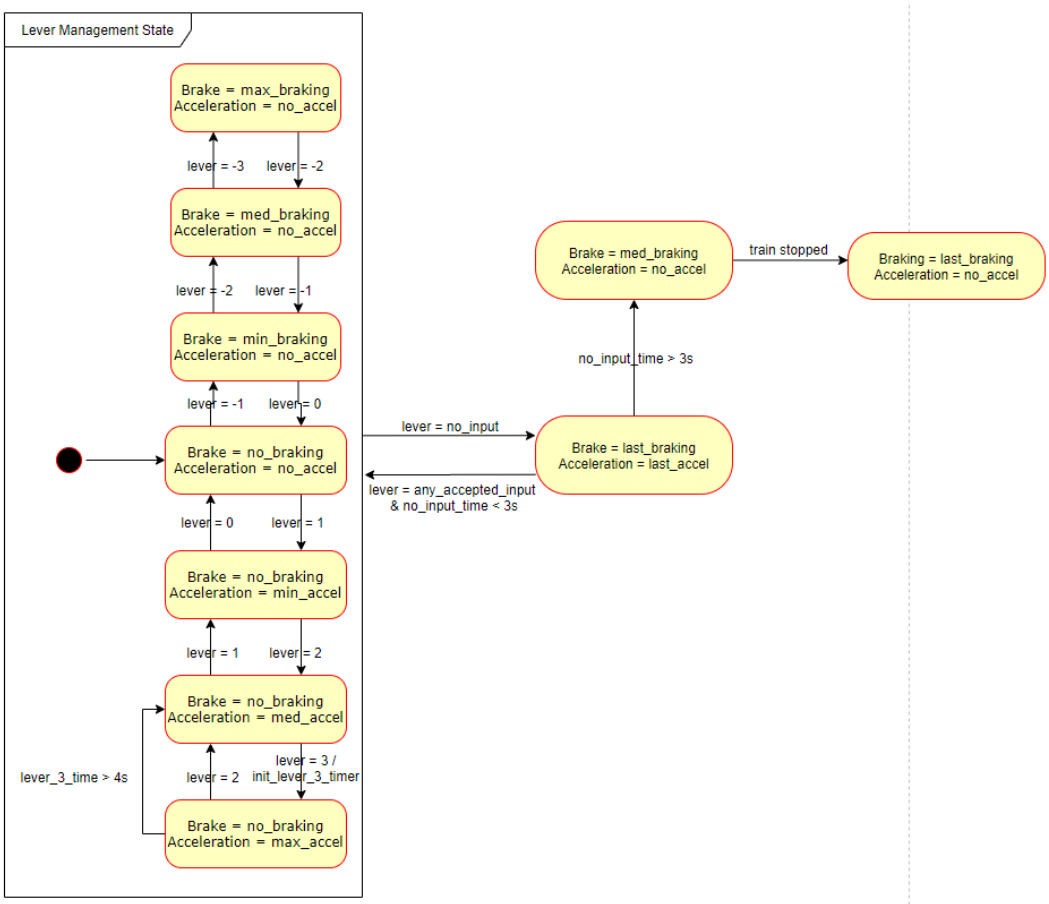
The dynamic evolution of the system can be described with the following state machine diagram:



In this diagram, each state is characterized by the output commands issued from the controller to both the engine and the braking system and the transitions between the states are represented by the inputs issued by the driver lever.

The diagram also specifies the variable lever_3_time, that is used to check whether the maximum acceleration power has been enabled for more than 4s.

In order to design the other safety specification, it is defined another state machine diagram:

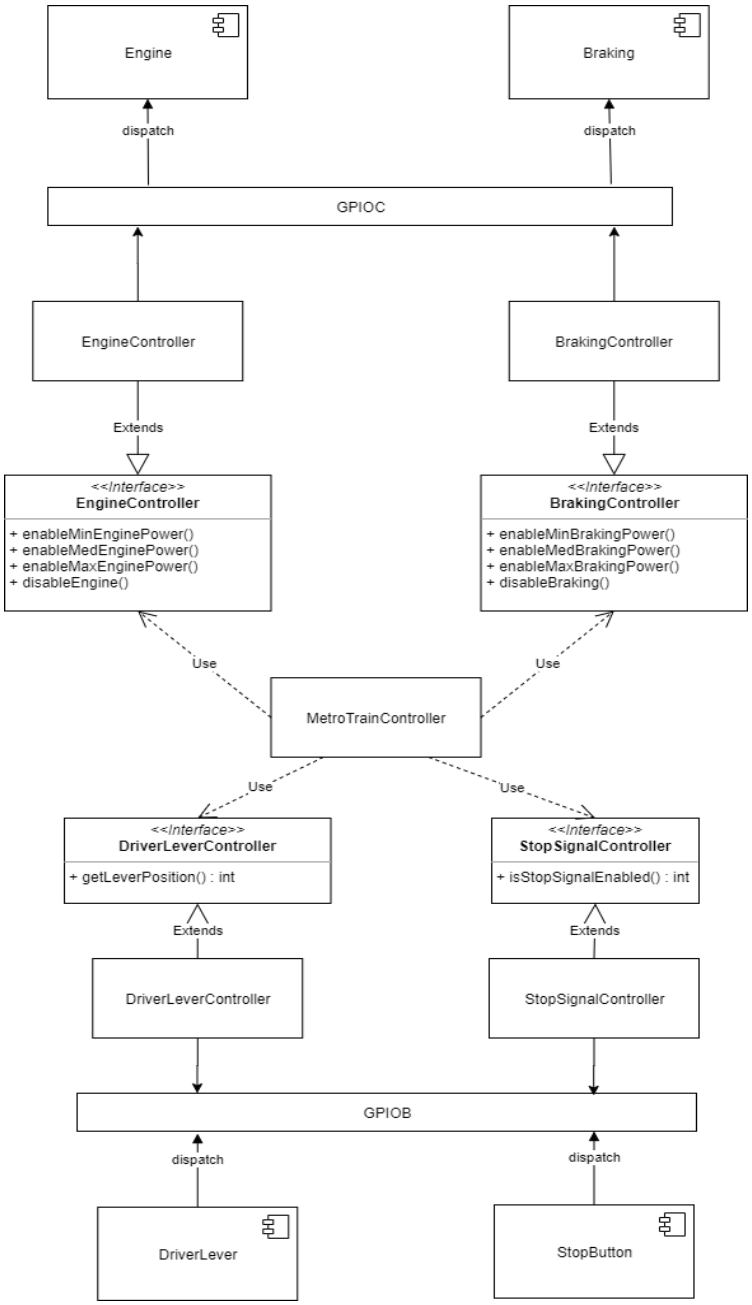


In particular, the most important detail in this diagram is the existence of a state with no exit transitions. This state is used to represent the condition of ‘waiting indefinitely until the system is manually reset’.

Design: stop signal management

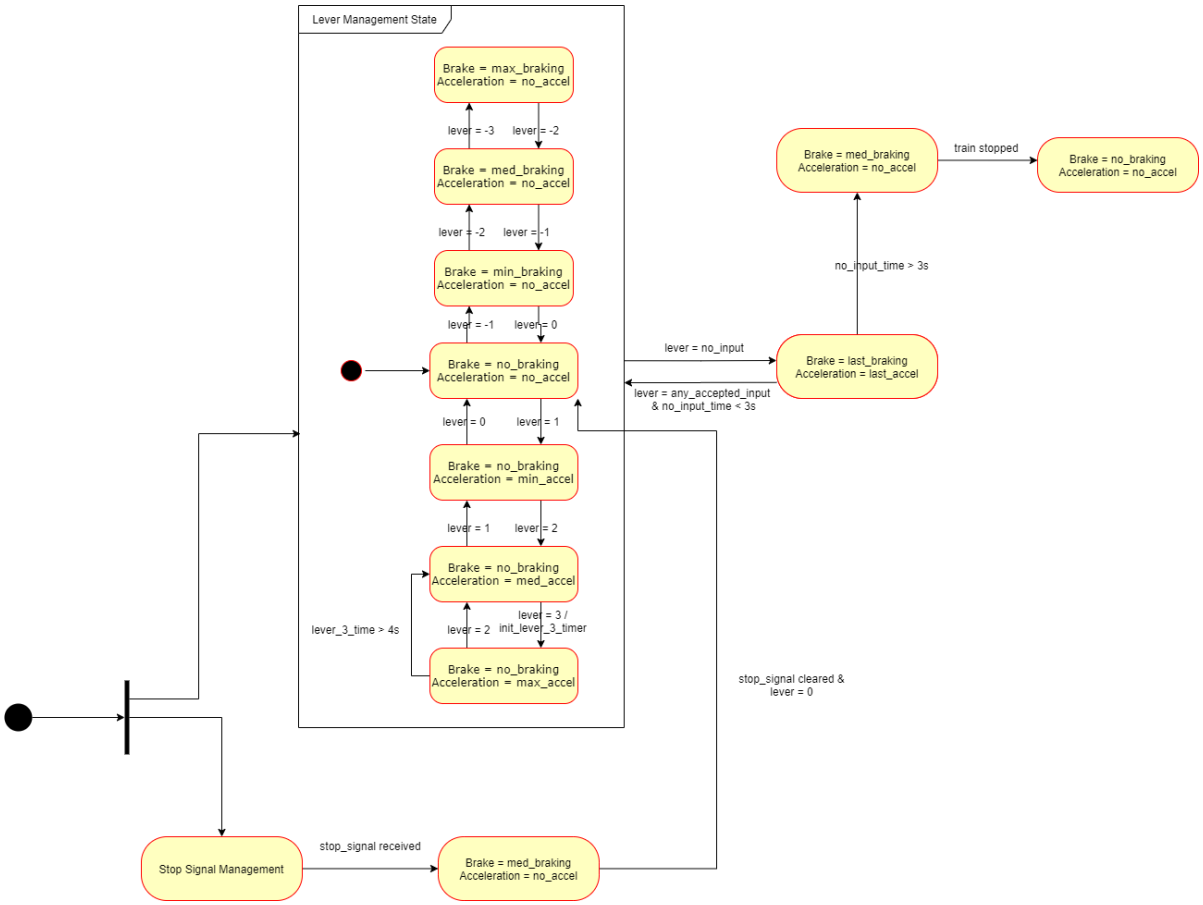
The step 2 involves the design of a system that must be able to manage both the inputs issued from the lever and the stop signals.

The physical system can be described with the following component diagram:



With respect to the previous step, the StopButton component is taken into account. Consequently, it is added a controller that allows the management of the signals issued from the button.

The dynamic evolution of the system can be described with the following state machine diagram:

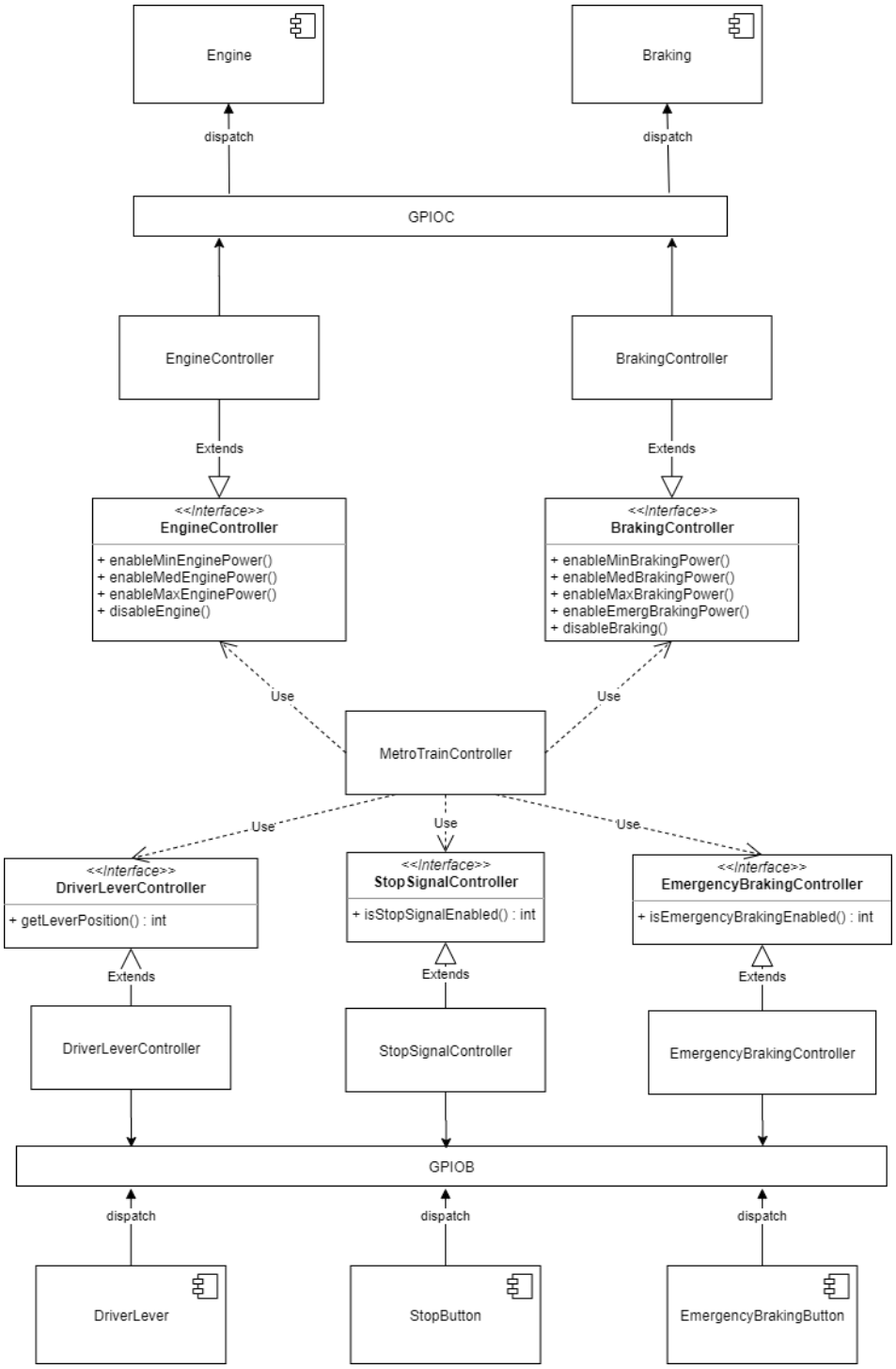


With respect to the previous step, the controller responsibilities are split into 2 parallel tasks. The first one is responsible for the safe management of the lever, and specifically for all the specifications concerning the step 1. The second task is responsible for the stop signal management, and it must have a higher priority than the other one.

Design: emergency braking management

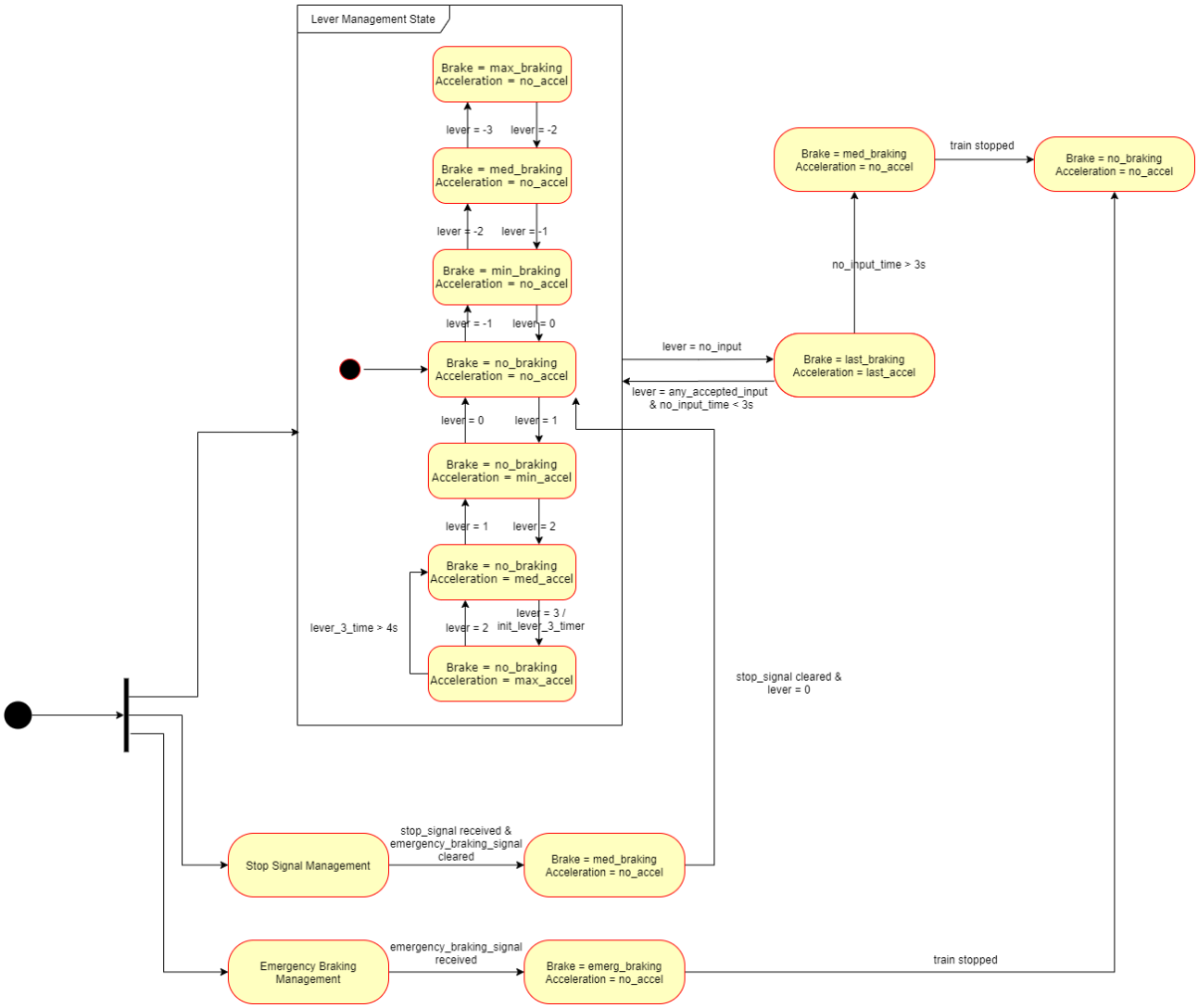
The step 3 involves the design of a system that must be also able to manage the signals issued by the emergency braking button.

The physical system can be described with the following component diagram:



With respect to the previous steps, the EmergencyBrakingButton is taken into account. Consequently, the system is improved with the design of a controller that allows the management of the signals issued from the emergency button and a further braking power option.

The dynamic evolution of the system can be described with the following state machine diagram:



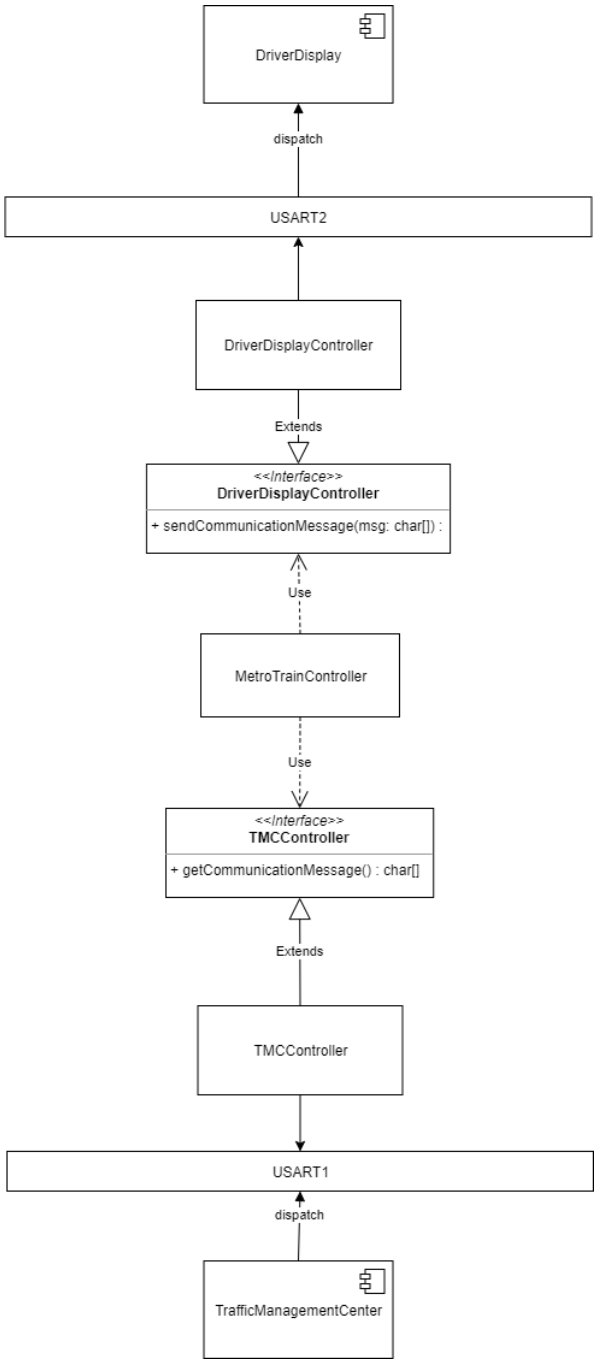
With respect to the previous steps, another parallel task must be added to the system. This task is responsible for the emergency braking signal management and its priority must be the highest between all the tasks. Even in this case, whenever an emergency braking signal is issued, the system will enter in a state with no exit after the train has been stopped.

Furthermore, the task that is responsible for the stop signal management also has to check whether an emergency braking signal has been issued before the processing of the stop signal request.

Design: communication messages management

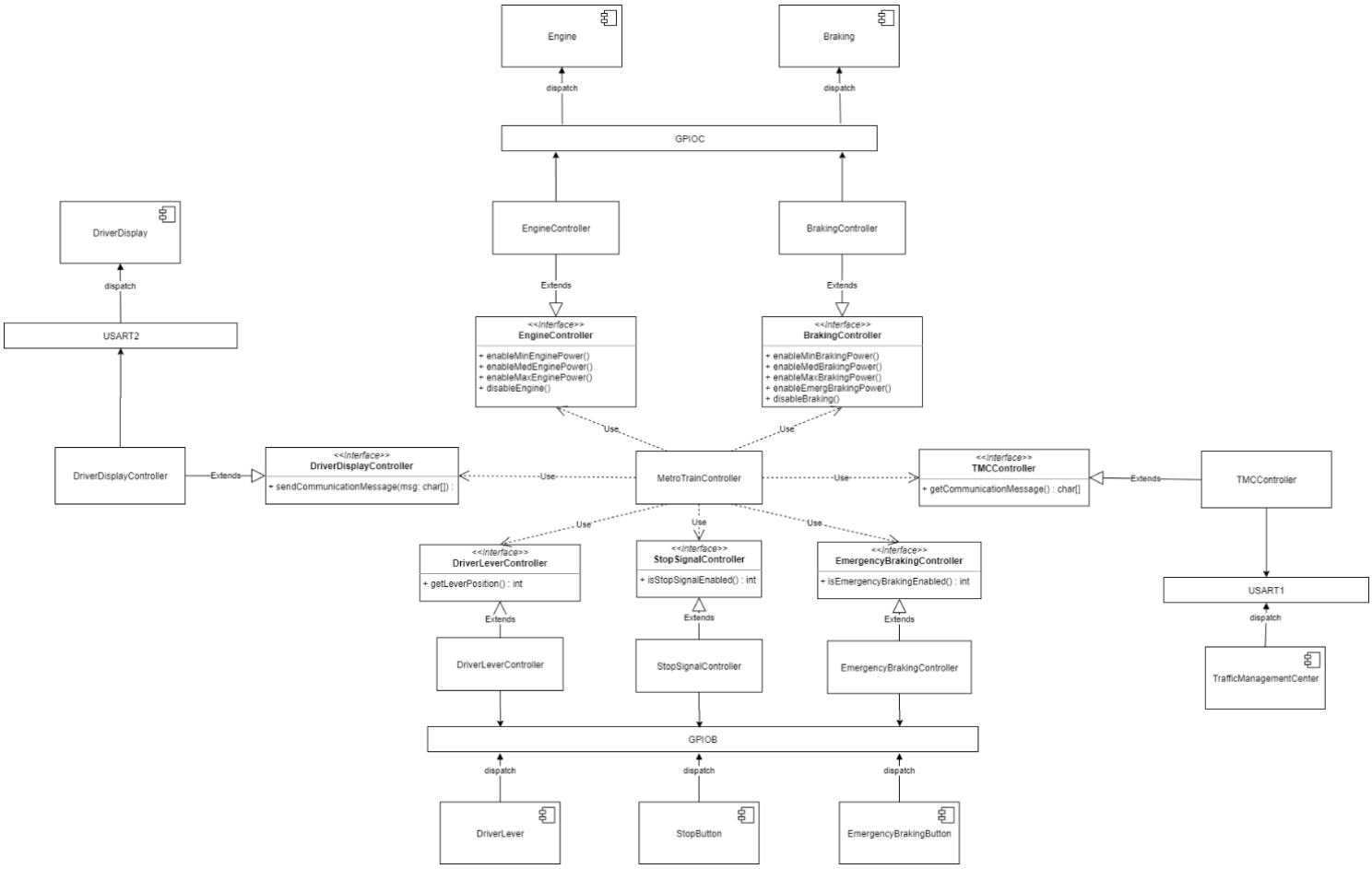
The step 4 involves the design of a system that must be also able to manage the messages received from the traffic management center.

The physical system concerning only the messages management part can be described with the following component diagram:

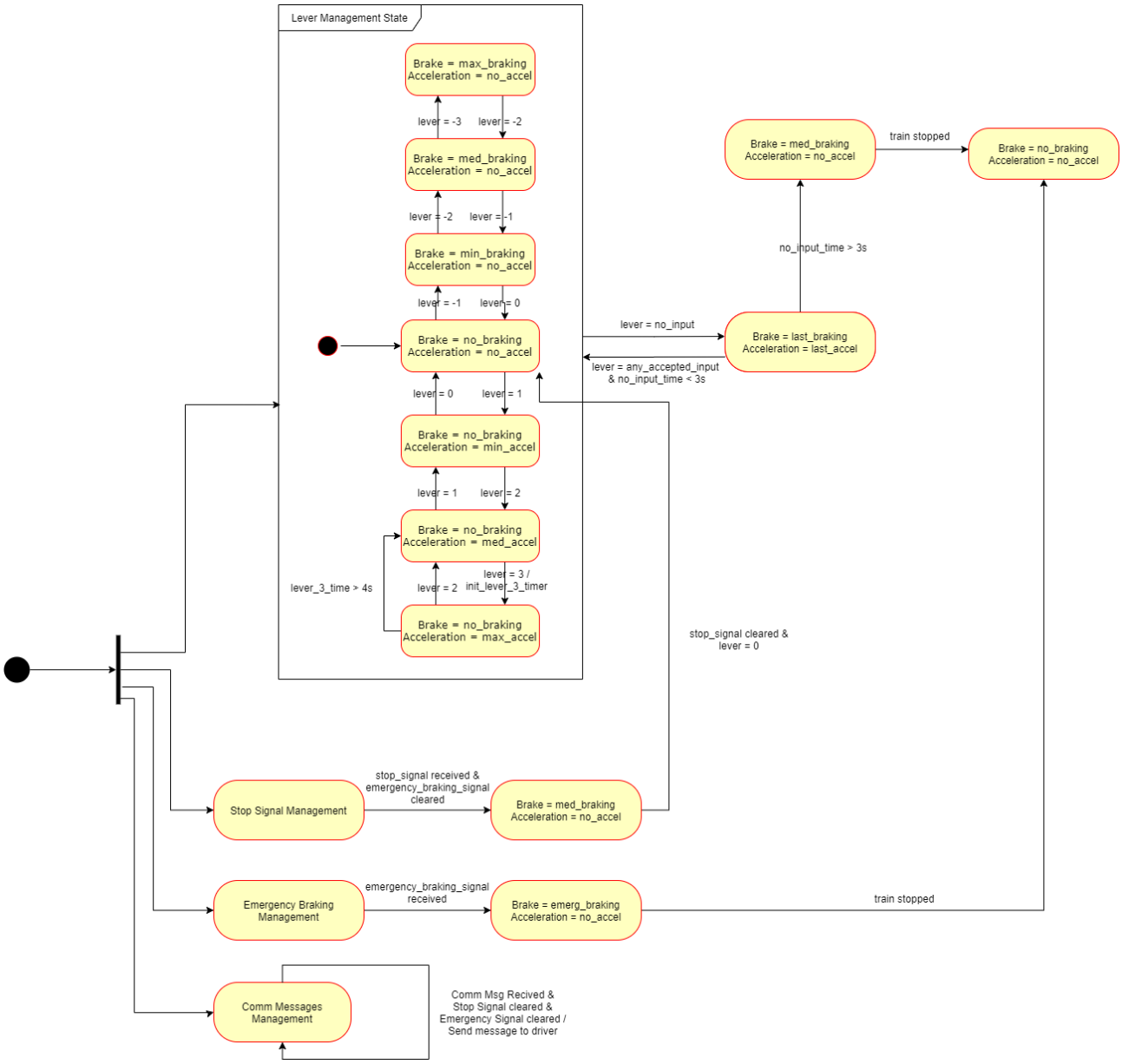


According to the specifications, this diagram shows how the components should interact each other. In particular, the messages are sent by the TrafficManagementCenter through the USART1 serial line. Then they must be caught by the software controller and forwarded to DriverDisplay through the USART2 serial line.

Considering the whole physical system, it can be described with the following component diagram:



The dynamic evolution of the system can be described with the following state machine diagram:



With respect to the previous steps, another parallel task must be added to the system. This task is responsible for the communication messages management and its priority must be the lowest between all the tasks. Whenever the system receives a communication message from the traffic management center, it has to check whether there are some other actions to be performed, and only if the previous condition is not satisfied, it can forward the message to the driver. The controller always re-enters the same state each time a message is sent through the USART2.

Design: input simulation and test data

The step 5 involves the design of a task that should be able to simulate the inputs provided from all the components with which the controller interacts.

Specifically, the task should be able to:

- Read a simulated input from an array. In particular, a simulated input can be represented as a triple composed by the input provided to the controller, the task that must manage the input and a delay time.
- Provide the input to the specified task.
- Wait for an amount of time equal to the delay.

In this way, it is possible to create test data, represented by the sequences of inputs, in order to exercise some specific behaviors of the system and check whether the expected results are generated.

In order to exercise the main features of the system, the following test data have been designed:

1. Test data that checks the correctness of the normal lever management
2. Test data that checks the correctness of the safe lever management
3. Test data that checks the correctness of the stop signal management
4. Test data that checks the correctness of the emergency braking signal management
5. Test data that checks the correctness of the communication messages management

Since this task should be able to simulate each kind of input, it should have the maximum priority.

Test Data and Results

The test data are used to exercise the different features of the system.

In particular, 5 test data and their results will be presented:

1. Check the correctness of the driver lever management and the first safety specification (the maximum acceleration power must not be enabled for more than 4s consecutively)
2. Check the correctness of the second safety specification (if no input is provided from the lever for more than 3s, the train must be stopped)
3. Check the correctness of the stop signal management
4. Check the correctness of the emergency braking management
5. Check the correctness of the communication messages management

Each test data is represented as an array of triples, and each triple is composed by

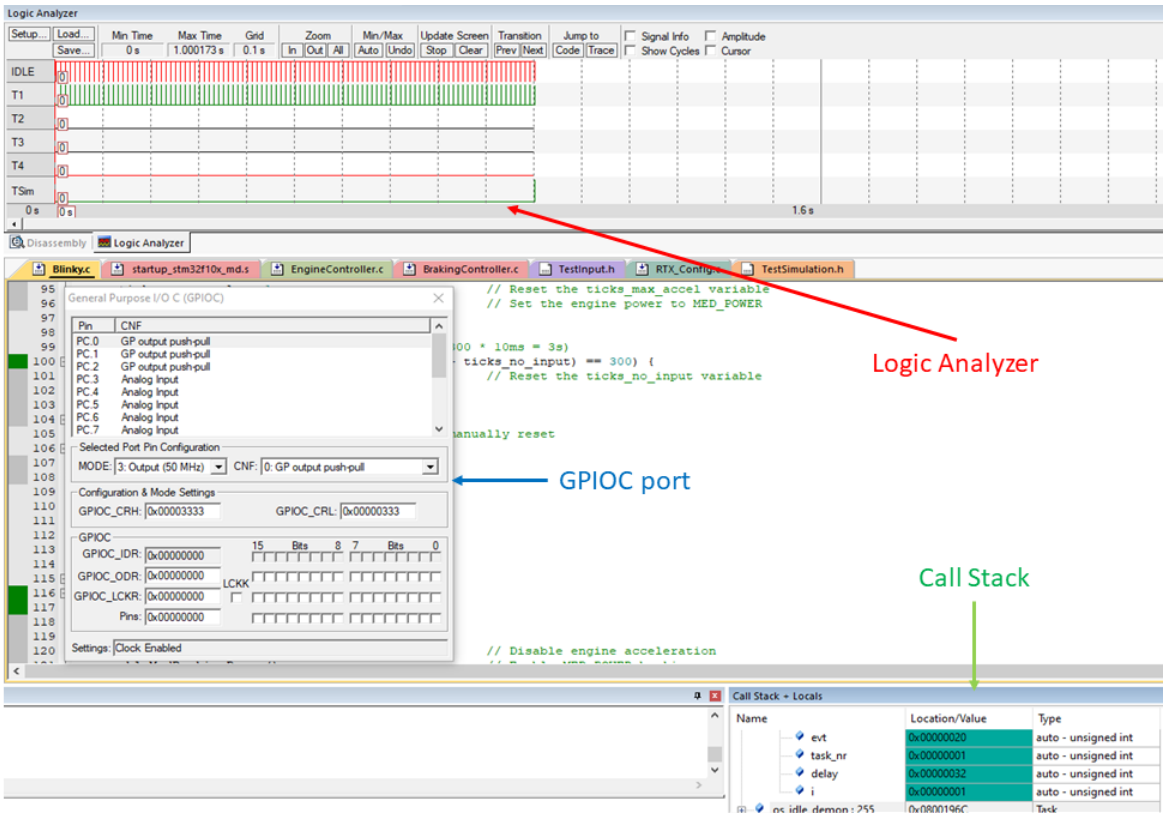
- the input that must be simulated
- the number of the task that must manage the input
- the delay (in system ticks) that the task must wait until the next input.

Each system tick lasts 10ms.

For each test data, it will be displayed an image containing

- The logic analyzer, showing the time flow and the task execution times
- The GPIOC port, showing the outputs generated by the system
- The call stack, showing the simulated input values

For example



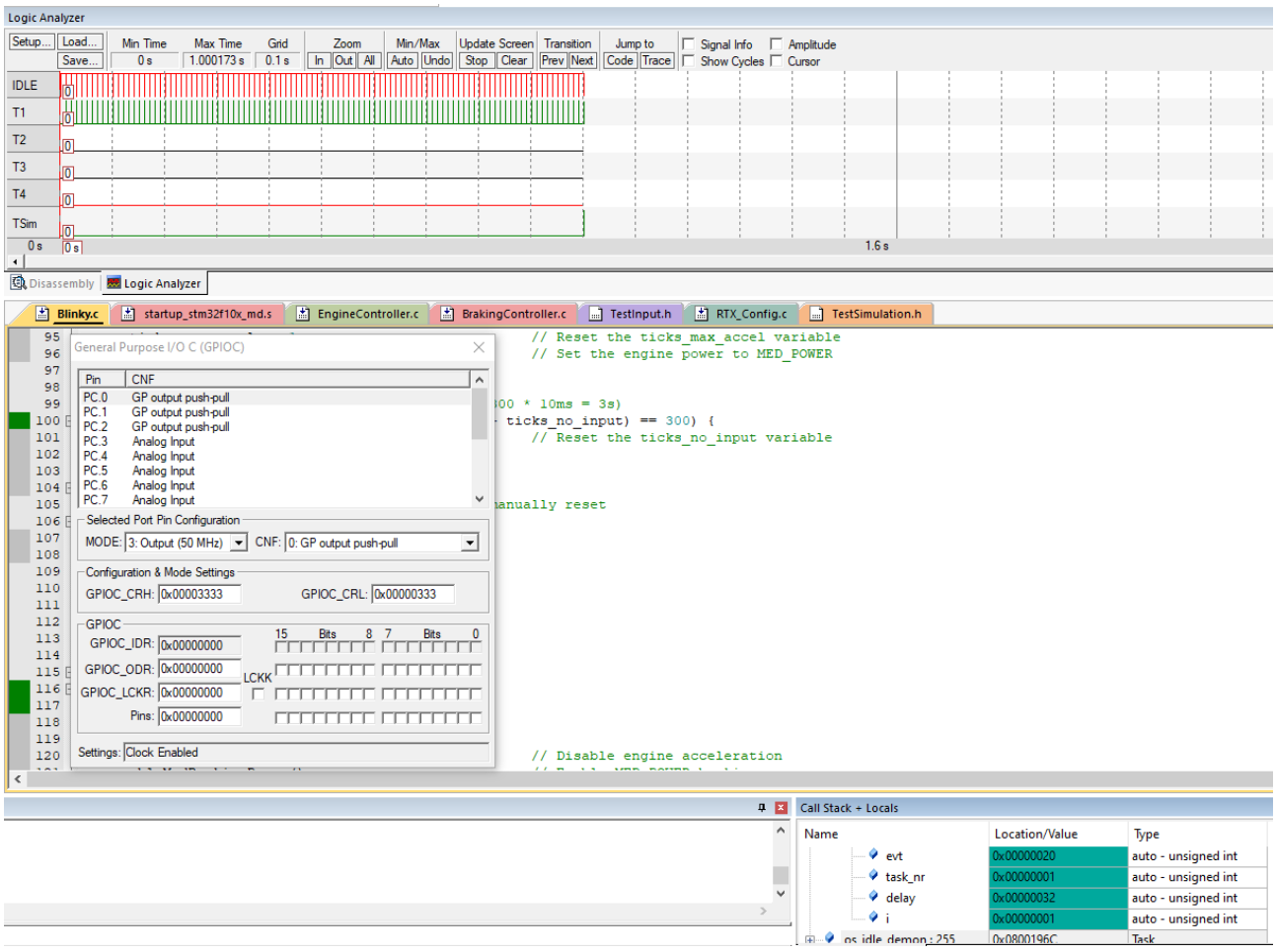
For conciseness reasons, not all the test data executions will be displayed. However, the test data can be entirely executed by running the code.

Test data 1

The test data is composed as follows:

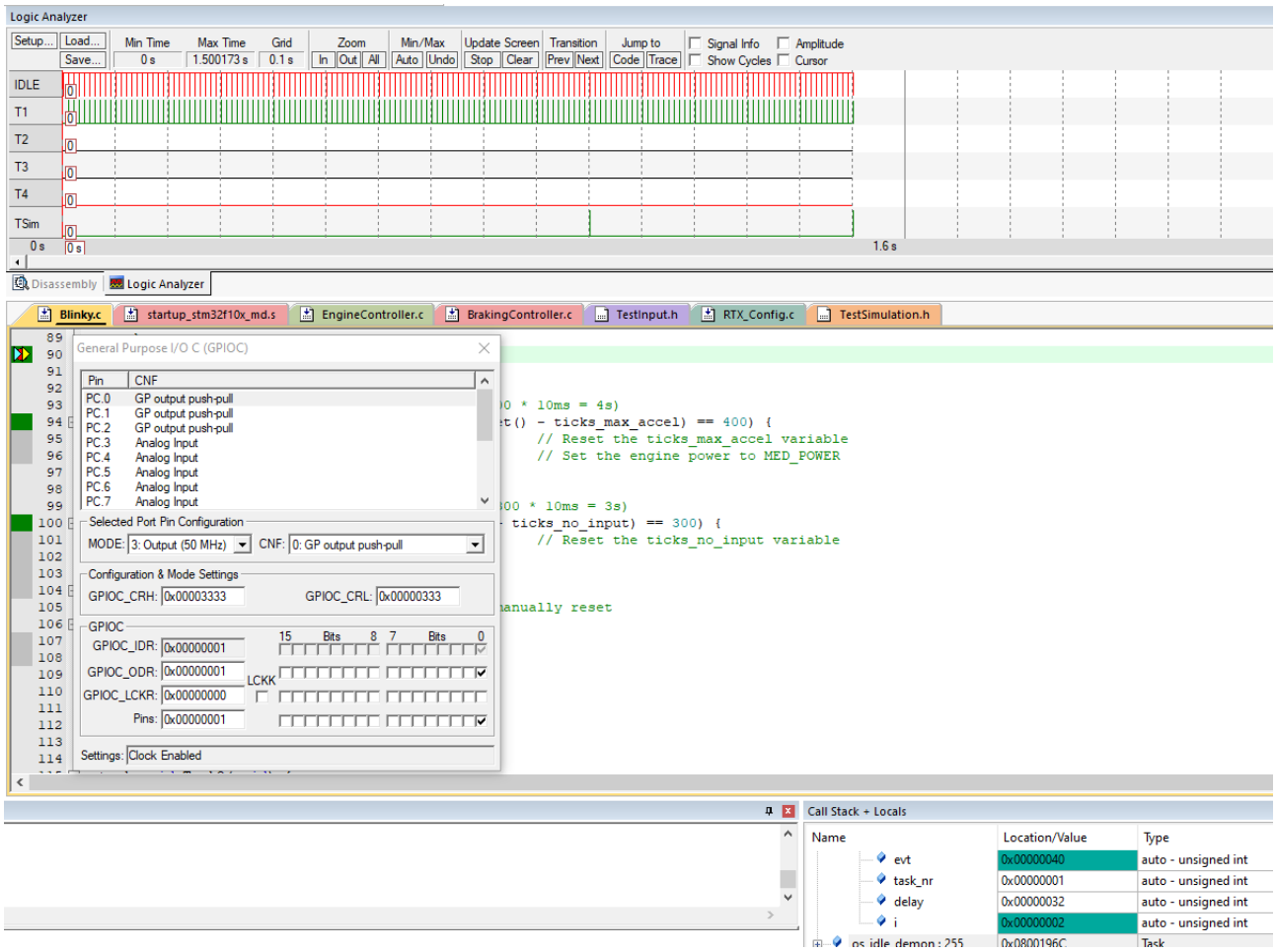
1. {PIN_IDLE_LEVER, 1, 50}
2. {PIN_MIN_ACCEL, 1, 50}
3. {PIN_MED_ACCEL, 1, 50}
4. {PIN_MAX_ACCEL, 1, 500}
5. {PIN_MED_ACCEL, 1, 50}
6. {PIN_MIN_ACCEL, 1, 50}
7. {PIN_IDLE_LEVER, 1, 50}
8. {PIN_MIN_BRAKING, 1, 50}
9. {PIN_MED_BRAKING, 1, 50}
10. {PIN_MAX_BRAKING, 1, 50}

Providing to the system the input 1, the results are



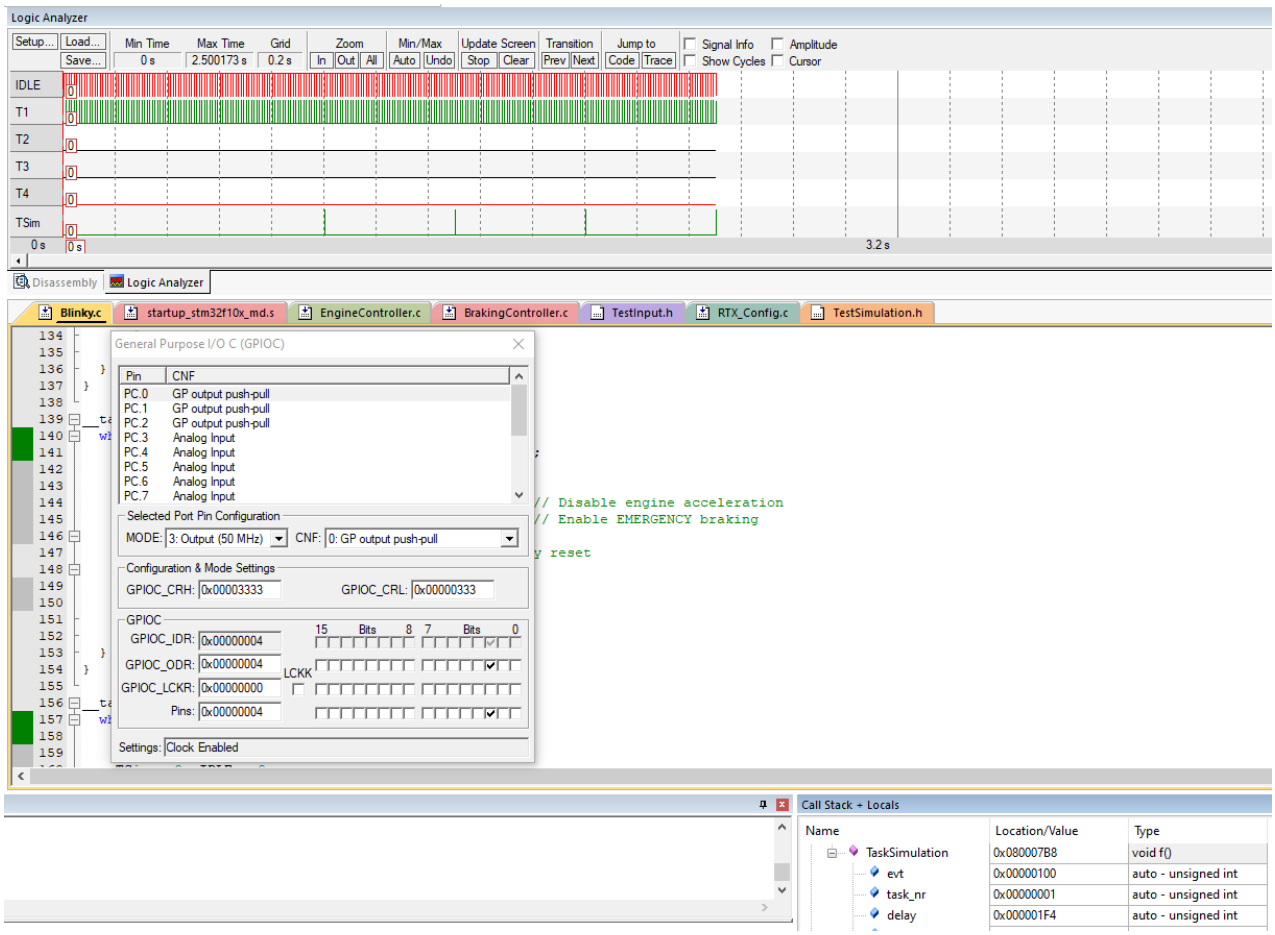
Specifically, the system does not issue any output through the GPIOC port. This is correct because the received input was the lever set in 'IDLE' position (0x00000020 = 1<<5 = GPIOB Pin5).

Providing to the system the input 2, the results are



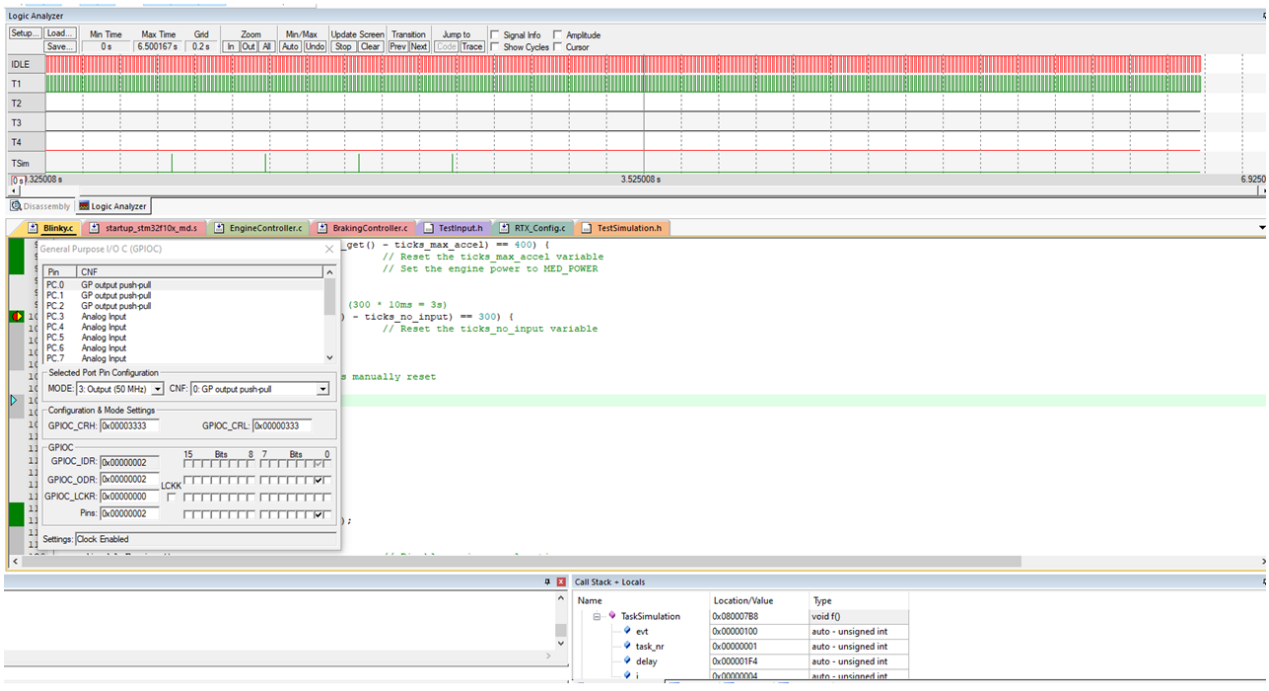
Specifically, the system switches on the GPIOC Pin0, corresponding to the minimum acceleration power. This is correct because the received input was the lever set in position +1 (0x00000040 = 1<<6 = GPIOB Pin6).

Providing to the system the input 4, the results are



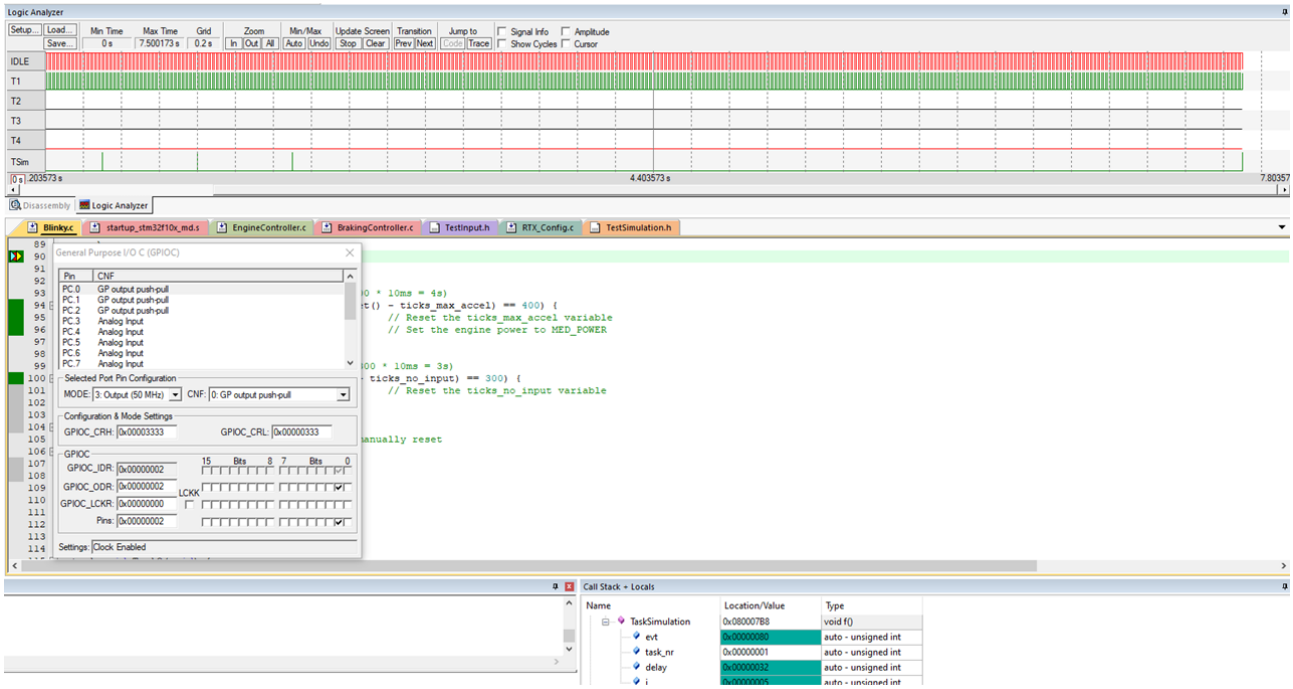
Specifically, the system switches on the GPIOC Pin2, corresponding to the maximum acceleration power. This is correct because the received input was the lever set in position +3 (0x00000100 = 1<<8 = GPIOB Pin8).

The simulation task is supposed to sleep for 5s after the input 4. However, after 4s, the system switches the output on the GPIOC port from the Pin2 to the Pin1.



This situation is consistent with the first safety specification.

Providing to the system the input 5, the results are



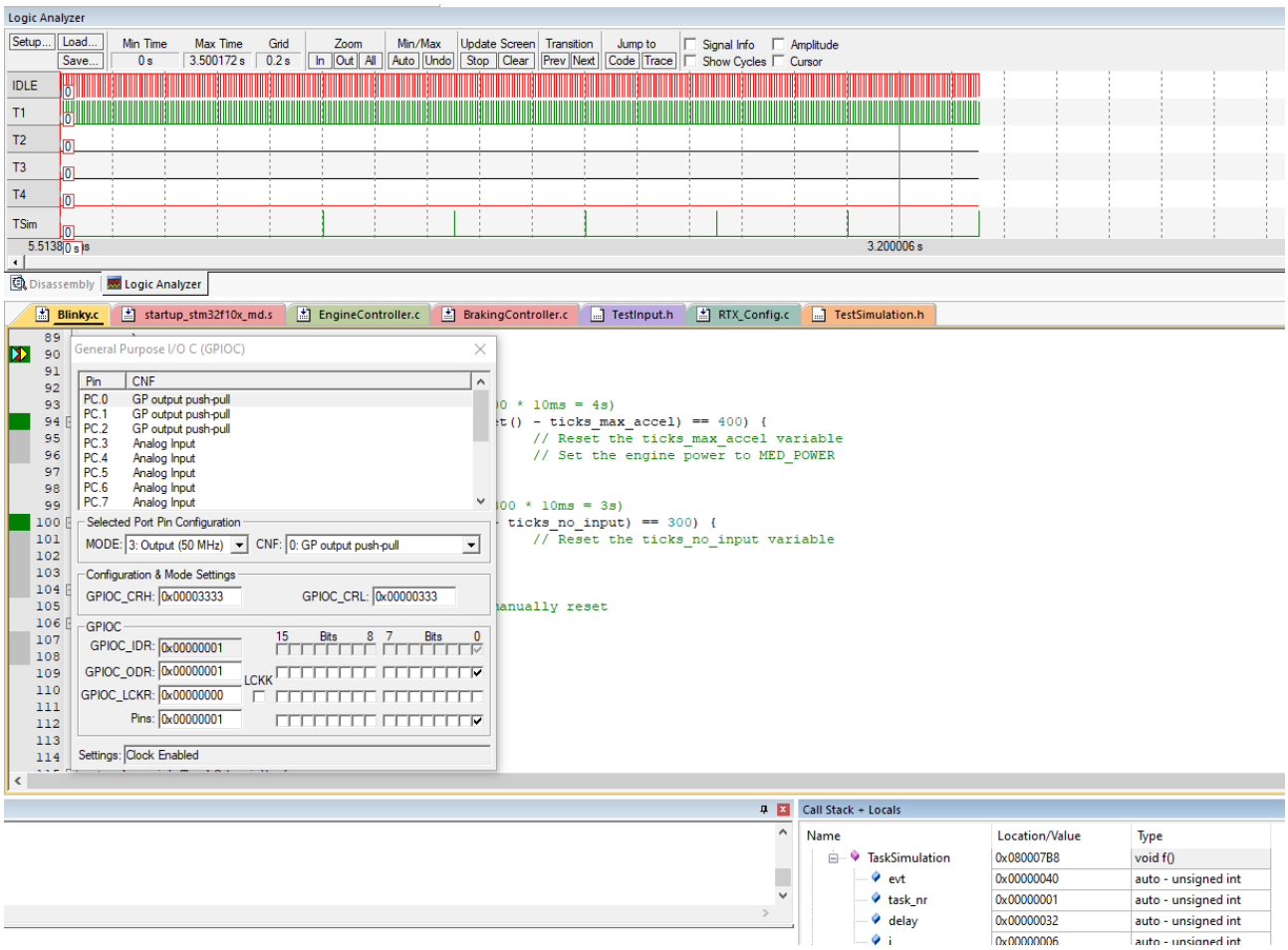
Specifically, the system does not change the output on the GPIOC port. This is correct because the received input was the lever set in position +2 and the output was still asking the engine to accelerate with medium power (0x00000080 = 1<<7 = GPIOB Pin7).

Test data 2

The test data is composed as follows:

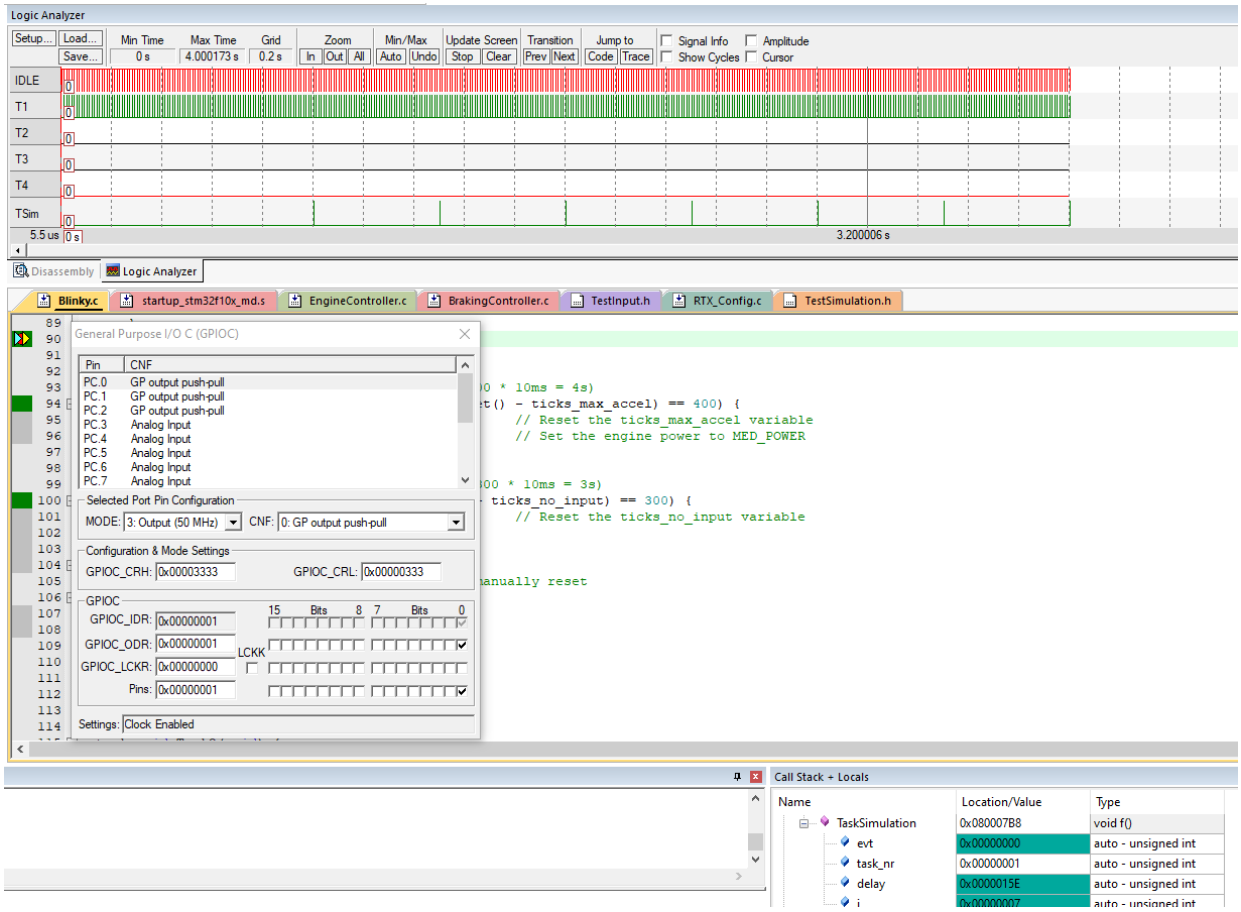
1. {PIN_IDLE_LEVER, 1, 50},
2. {PIN_MIN_ACCEL, 1, 50},
3. {PIN_MED_ACCEL, 1, 50},
4. {PIN_MAX_ACCEL, 1, 50},
5. {PIN_MED_ACCEL, 1, 50},
6. {PIN_MIN_ACCEL, 1, 50},
7. {NO_INPUT, 1, 350}
8. {PIN_IDLE_LEVER, 1, 50},
9. {PIN_MAX_ACCEL, 1, 50},
10. {PIN_MIN_BRAKING, 1, 50}

Providing to the system the input 6, the results are



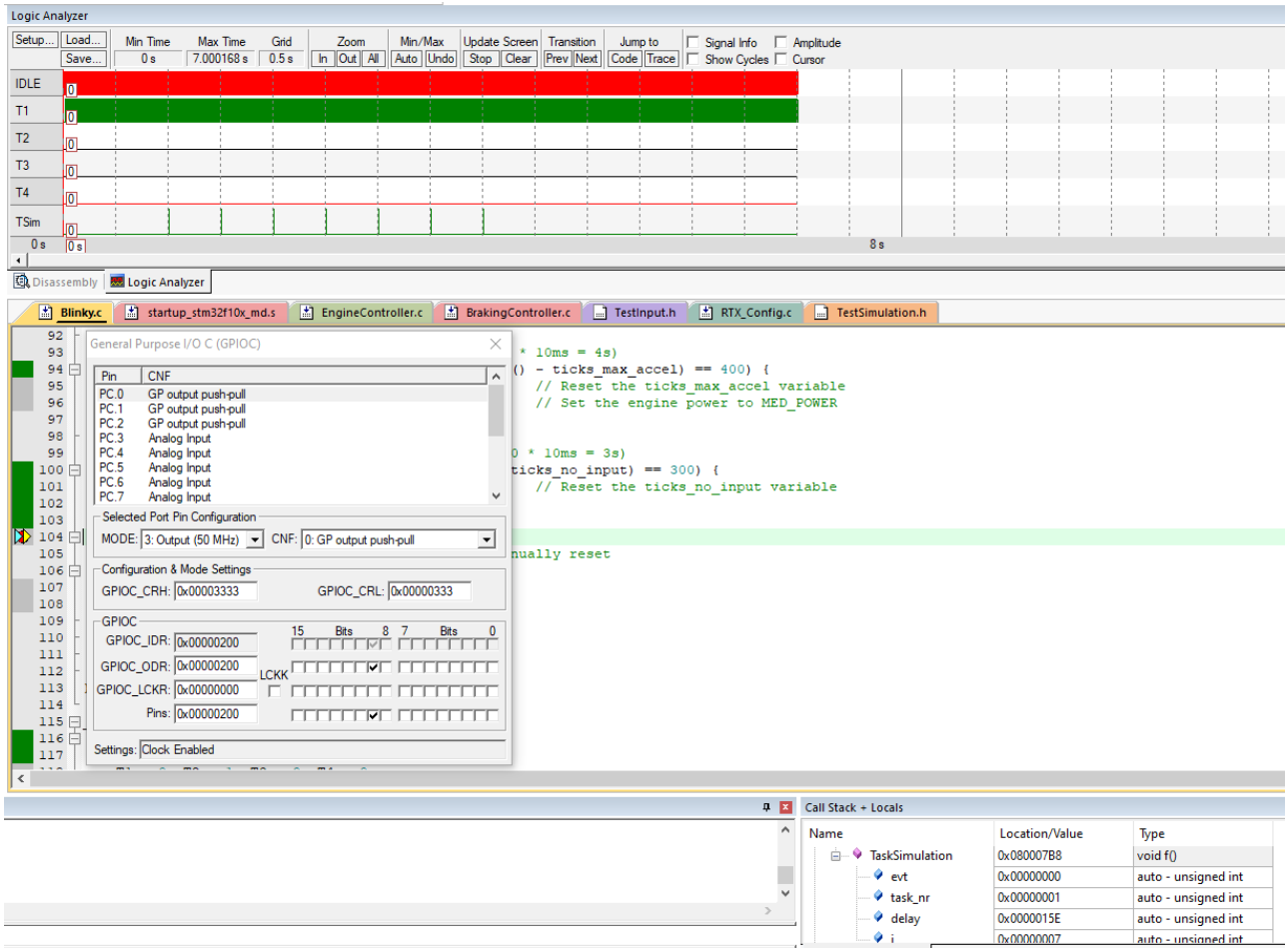
Specifically, the system switches on the GPIOC Pin0, corresponding to the minimum acceleration power. This is correct because the received input was the lever set in position +1 (0x00000040 = 1<<6 = GPIOB Pin6).

Providing to the system the input 7, the results are



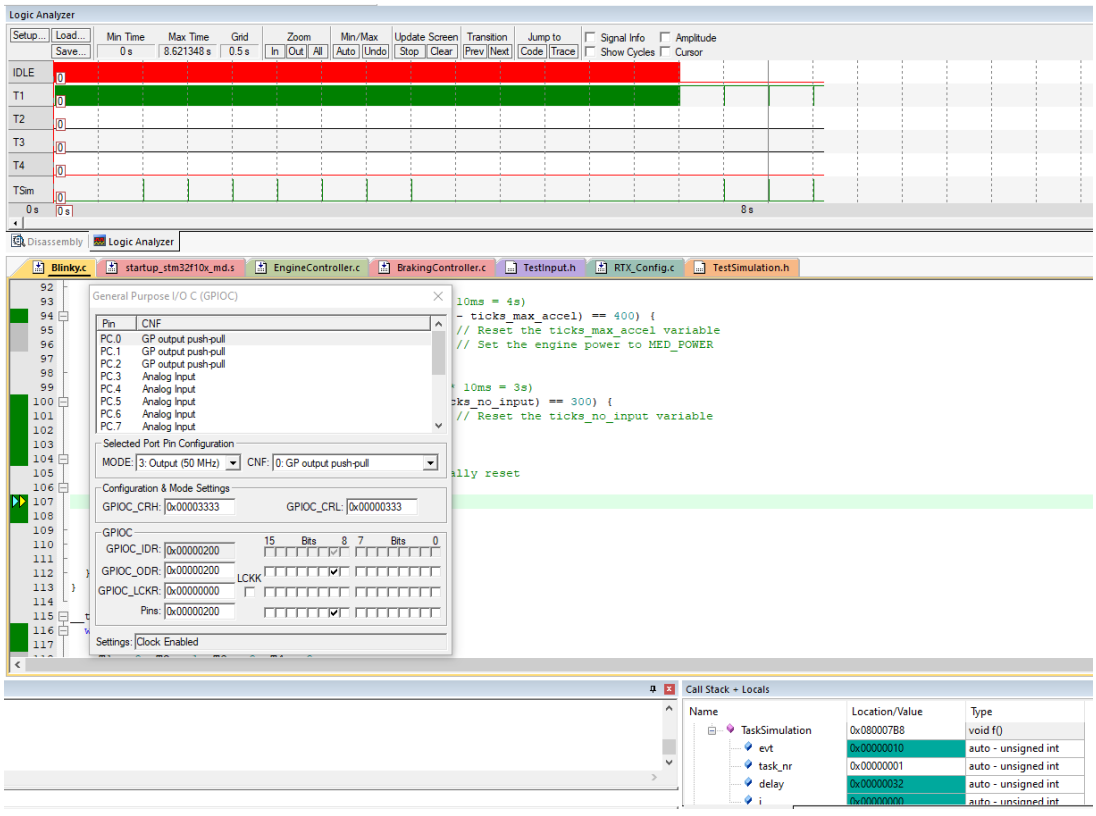
Specifically, the system does not change the output on the GPIOC port. This is correct because the received input was empty, and so the system must leave the output unchanged.

The simulation task is supposed to sleep for 3,5s after the input 7. However, after 3s, the system switches the output on the GPIOC port from the Pin0 to the Pin9.



This situation is consistent with the second safety specification.

Providing to the system all the other inputs, the results are



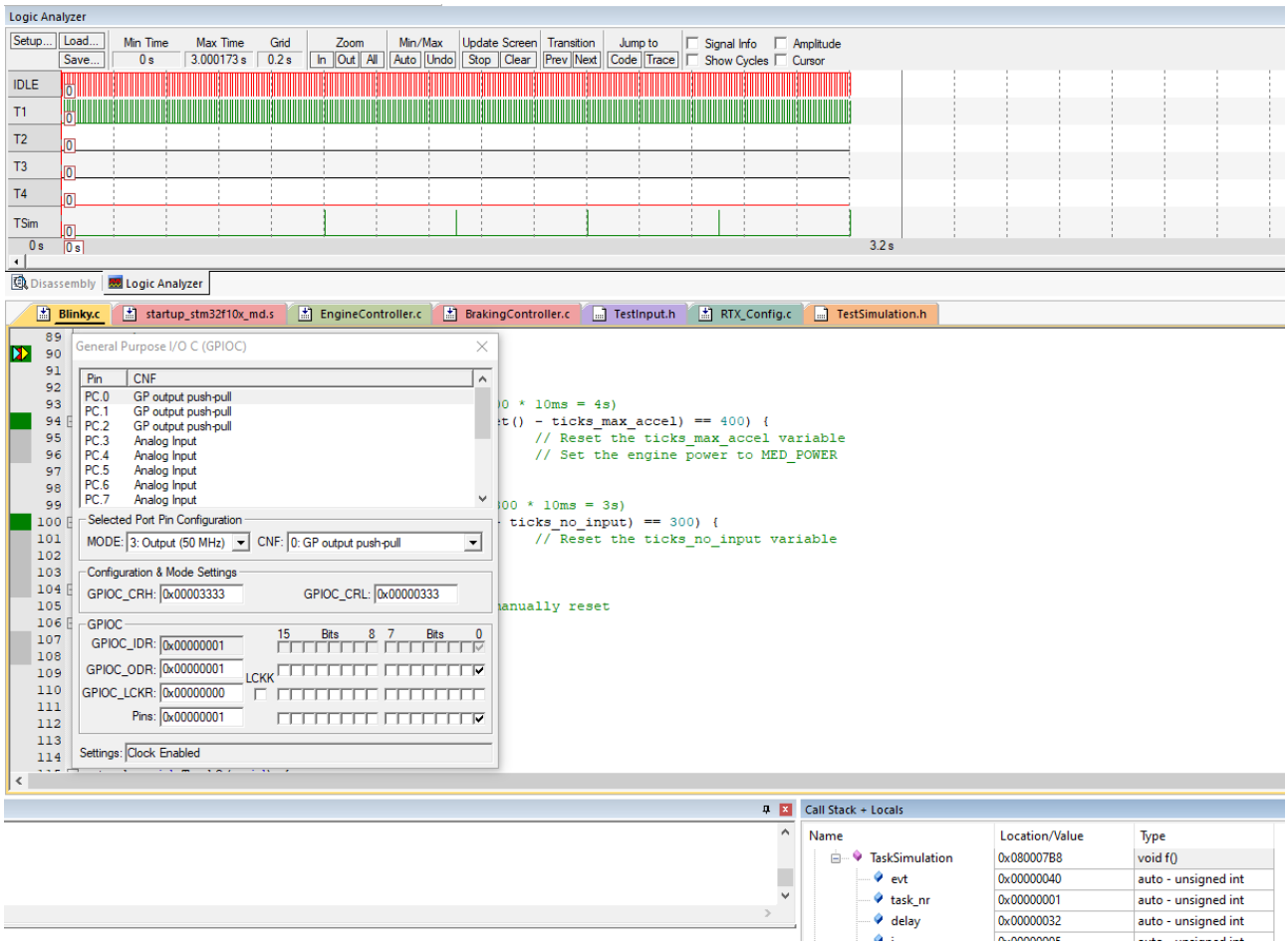
Specifically, the system does not change the output anymore on the GPIOC port. This is correct because the system must wait indefinitely until a manual reset is performed and all the received inputs must be ignored.

Test data 3

The test data is composed as follows:

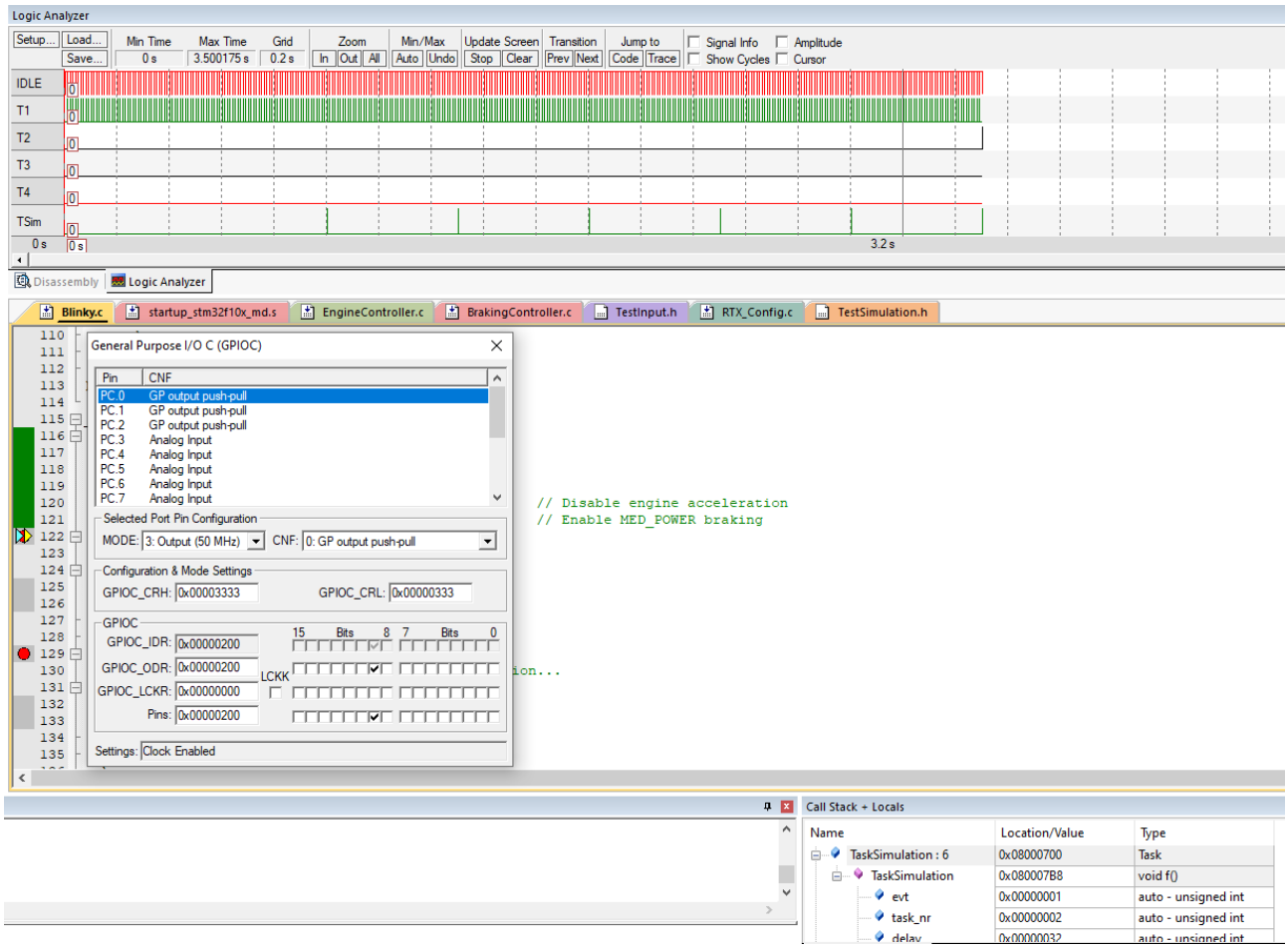
1. {PIN_MIN_BRAKING, 1, 50},
2. {PIN_MED_BRAKING, 1, 50},
3. {PIN_MIN_BRAKING, 1, 50},
4. {PIN_IDLE_LEVER, 1, 50},
5. {PIN_MIN_ACCEL, 1, 50},
6. {STOP_SIGNAL_ENABLED, 2, 50},
7. {PIN_MIN_BRAKING, 1, 50},
8. {STOP_SIGNAL_DISABLED, 2, 50},
9. {PIN_IDLE_LEVER, 1, 50},
10. {PIN_MIN_BRAKING, 1, 50}

Providing to the system the input 5, the results are



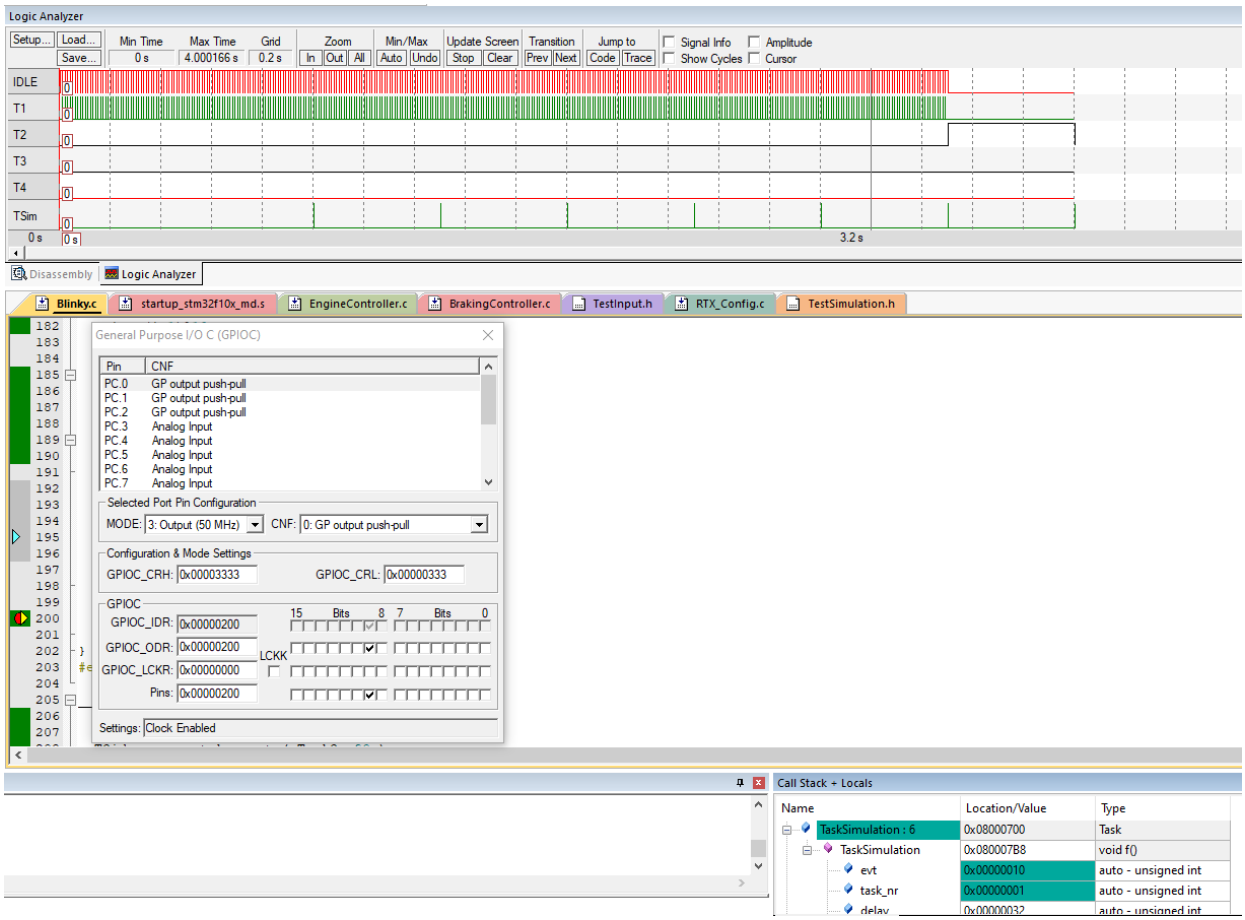
Specifically, the system switches on the GPIOC Pin0, corresponding to the minimum acceleration power. This is correct because the received input was the lever set in position +1 (0x00000040 = 1<<6 = GPIOB Pin6).

Providing to the system the input 6, the results are



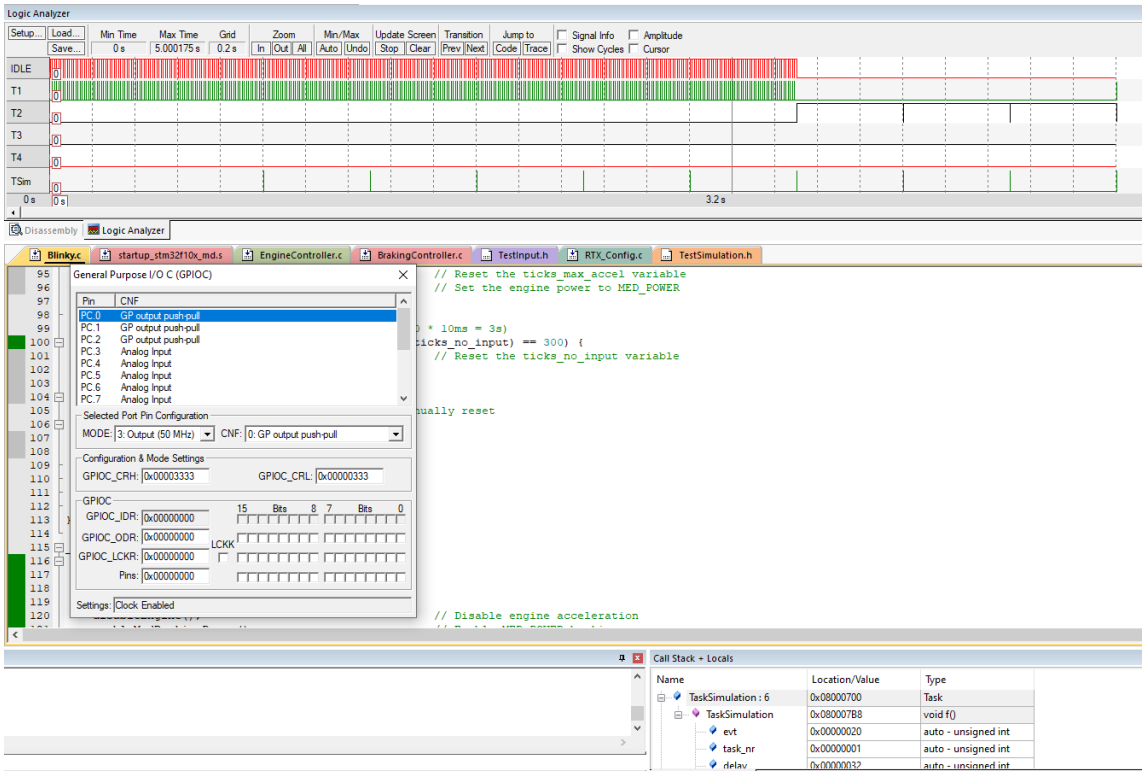
Specifically, the system switches on the GPIOC Pin9, corresponding to the medium braking power. This is correct because the received input was the stop signal.

Providing to the system the input 7, the results are



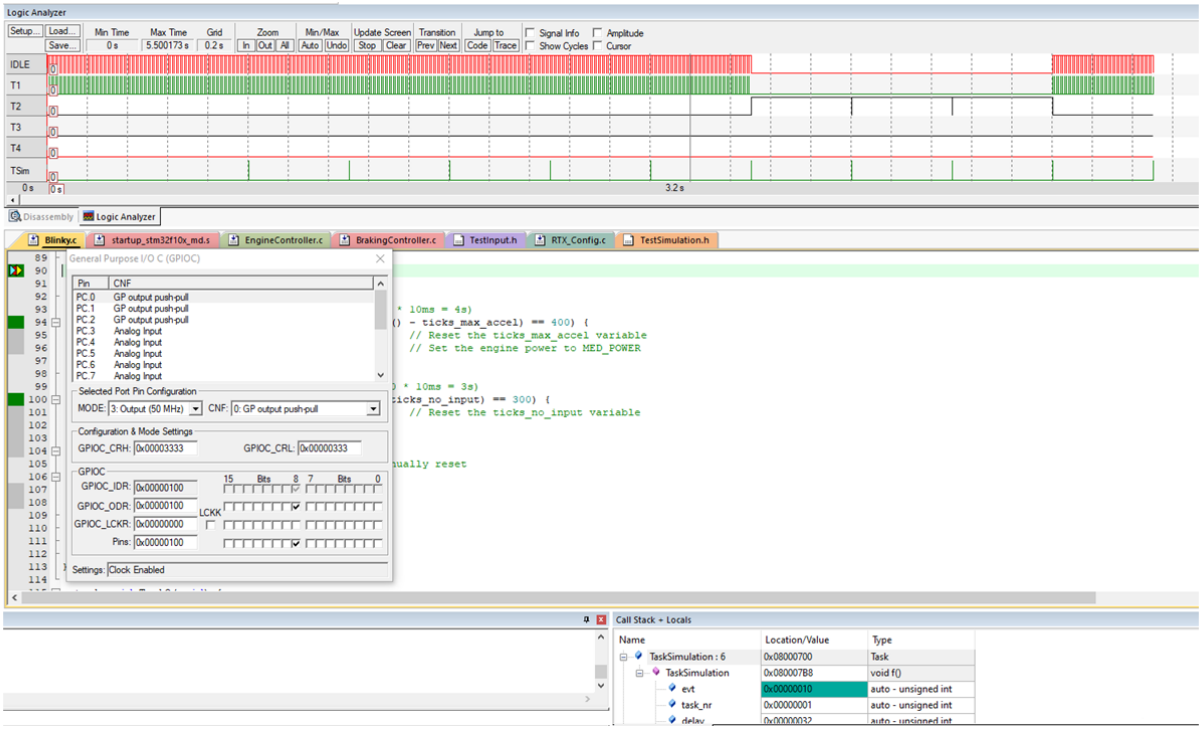
Specifically, the system does not change the output anymore on the GPIOC port. This is correct because the system must wait until both the stop signal is cleared and the lever is set in 'IDLE' position. All the received input must be ignored.

Providing to the system the inputs 8,9, the results are



Specifically, the system switches off all the GPIOC port Pins. This is correct because the stop signal was cleared (input 8) and the lever was set in 'IDLE' position (input 9). Furthermore, the controller restarts to check the inputs issued by the driver lever.

Providing to the system the input 10, the results are



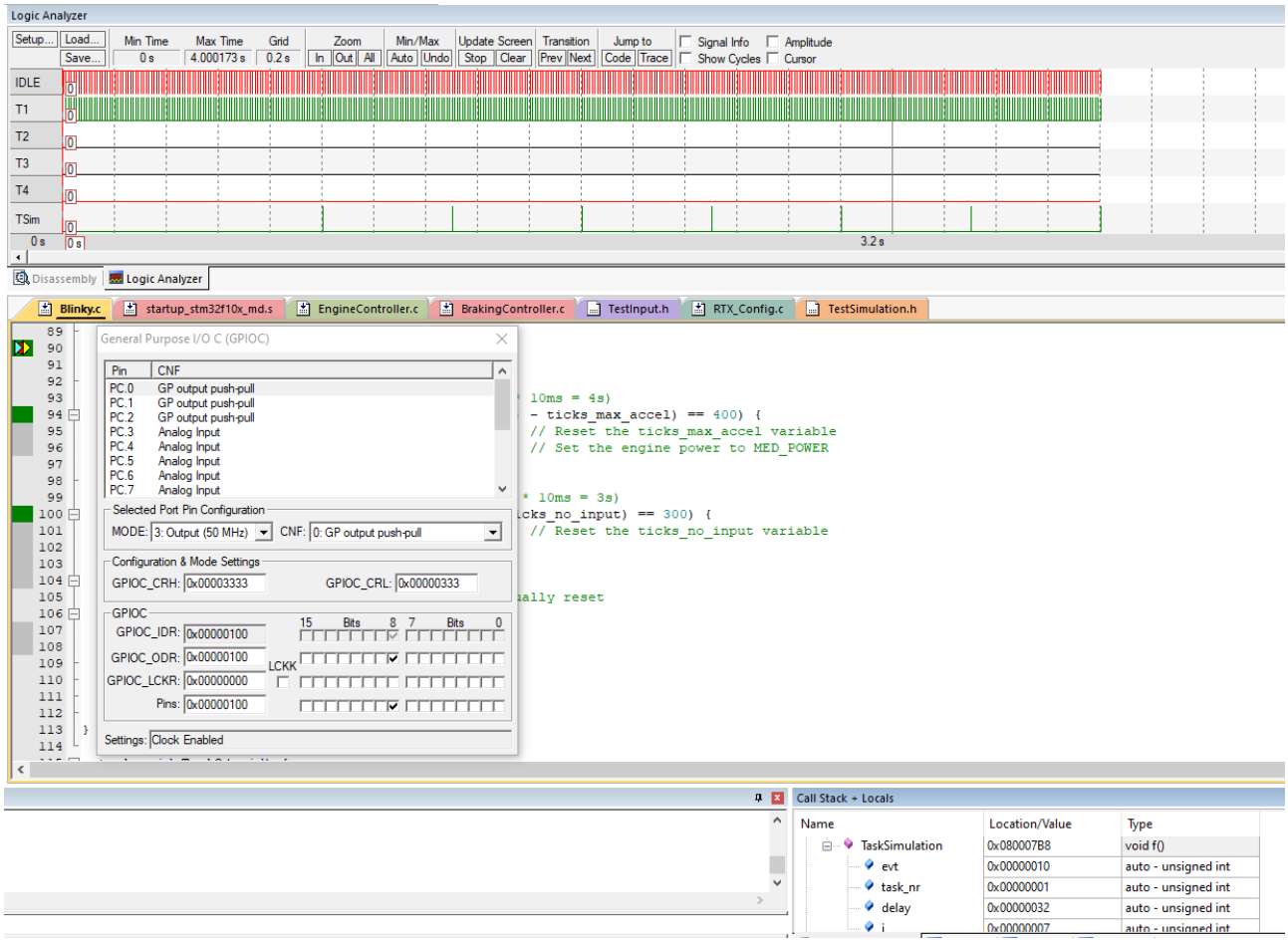
Specifically, the system switches on the GPIOC Pin8, corresponding to the minimum braking power. This is correct because the received input was the lever set in position -1 (0x00000010 = 1<<4 = GPIOB Pin4).

Test data 4

The test data is composed as follows:

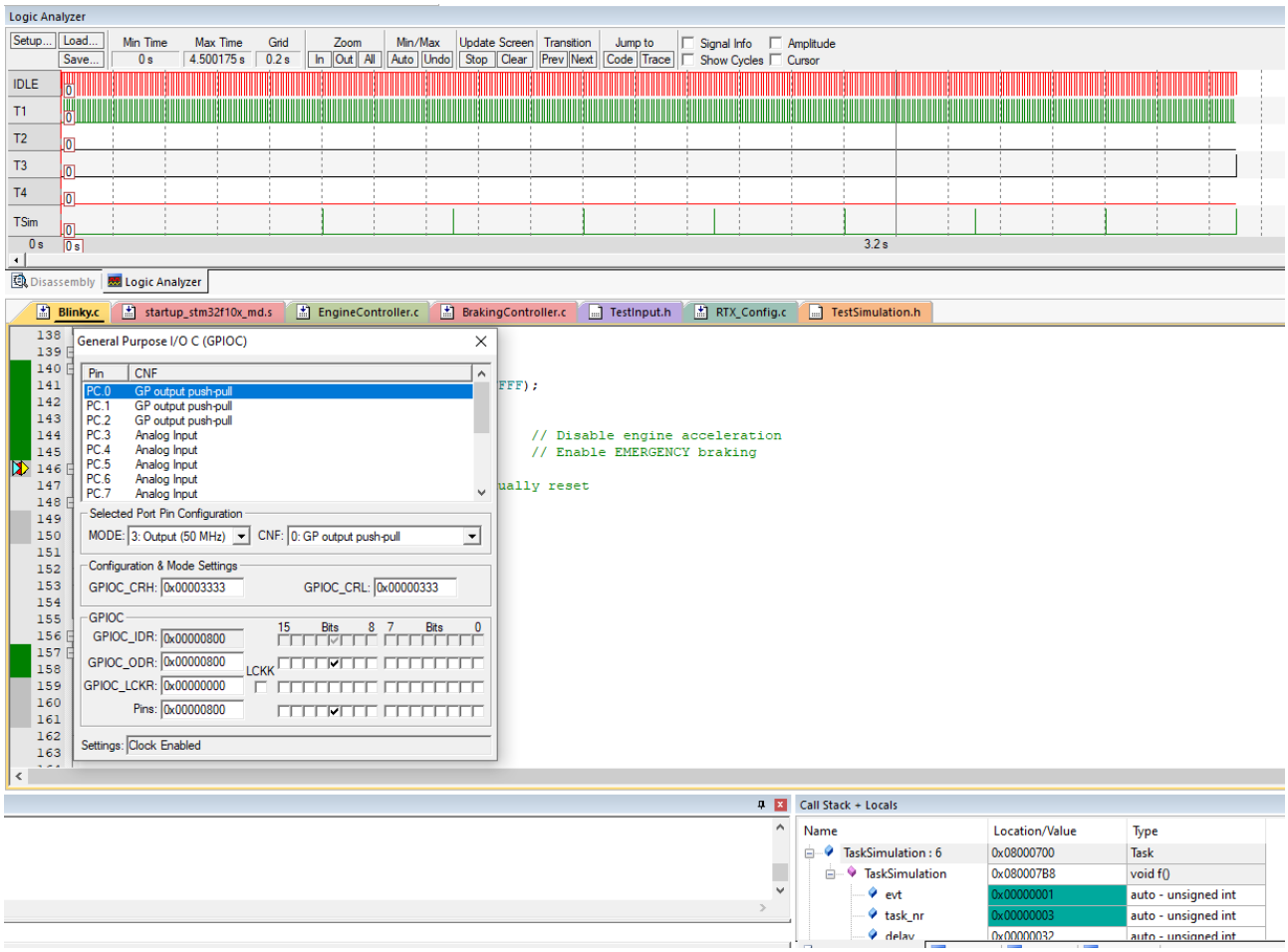
1. {PIN_MIN_BRAKING, 1, 50}
2. {PIN_MED_BRAKING, 1, 50}
3. {PIN_MAX_BRAKING, 1, 50}
4. {PIN_MED_BRAKING, 1, 50}
5. {PIN_MIN_BRAKING, 1, 50}
6. {PIN_IDLE_LEVER, 1, 50}
7. {PIN_MIN_BRAKING, 1, 50)
8. {EMERGENCY_SIGNAL_ENABLED, 3, 50}
9. {PIN_MIN_BRAKING, 1, 50},
10. {PIN_MIN_ACCEL, 1, 50}

Providing to the system the input 7, the results are



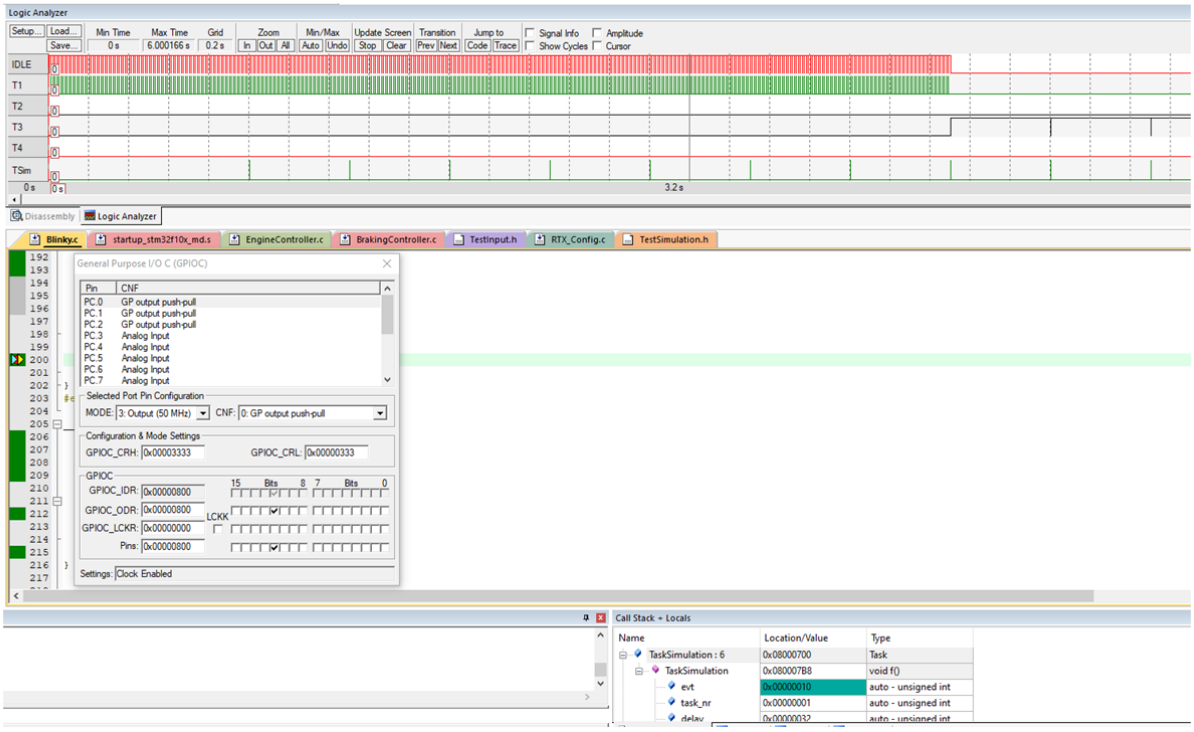
Specifically, the system switches on the GPIOC Pin8, corresponding to the minimum braking power. This is correct because the received input was the lever set in position -1 (0x00000010 = 1<<4 = GPIOB Pin4).

Providing to the system the input 8, the results are



Specifically, the system switches on the GPIOC Pin11, corresponding to the emergency (maximum) braking power. This is correct because the received input was the emergency braking signal.

Providing to the system the inputs 9,10, the results are



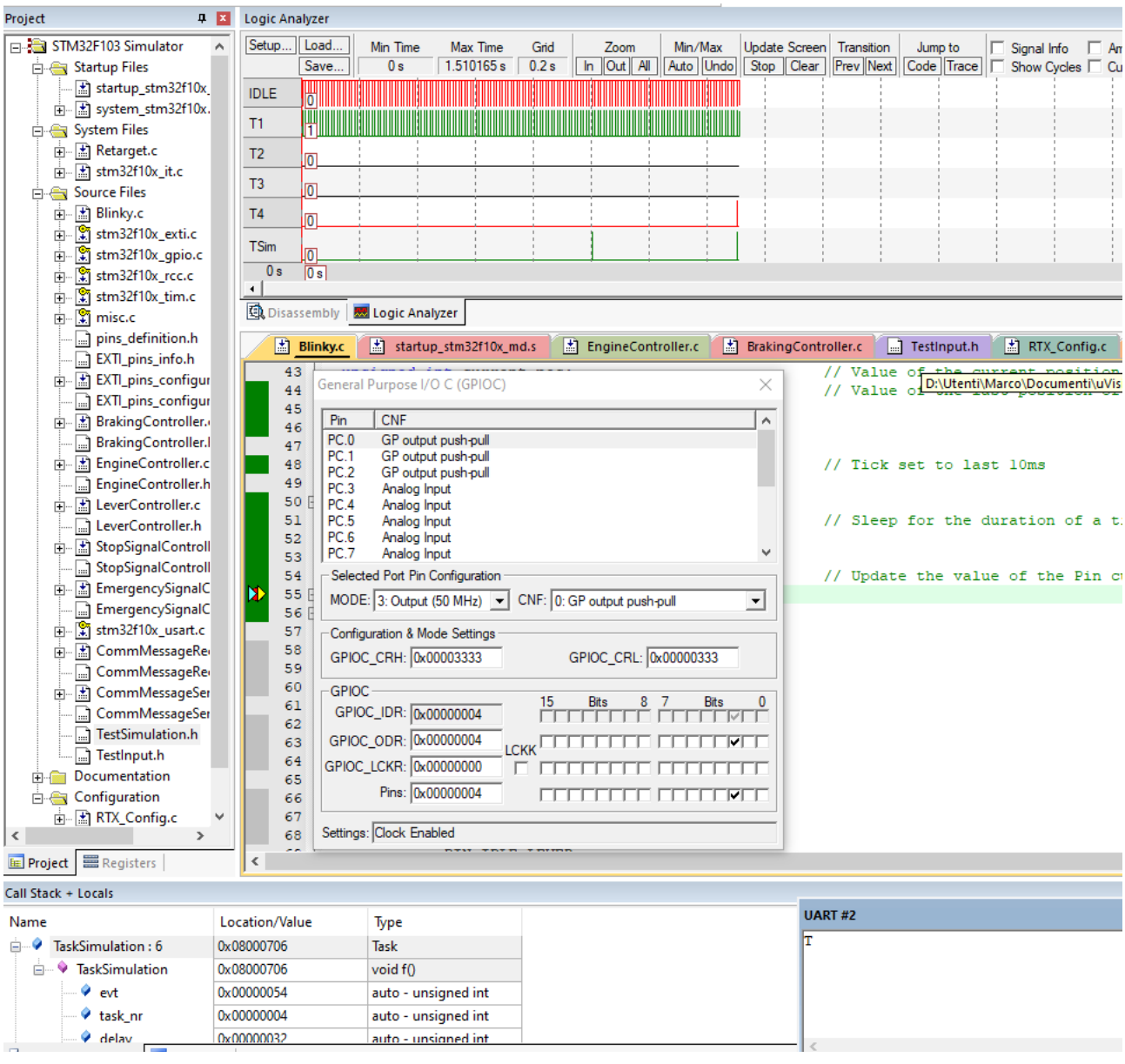
Specifically, the system does not change the output anymore on the GPIOC port. This is correct because the system must wait indefinitely until a manual reset is performed. All the received input must be ignored.

Test data 5

The test data is composed as follows:

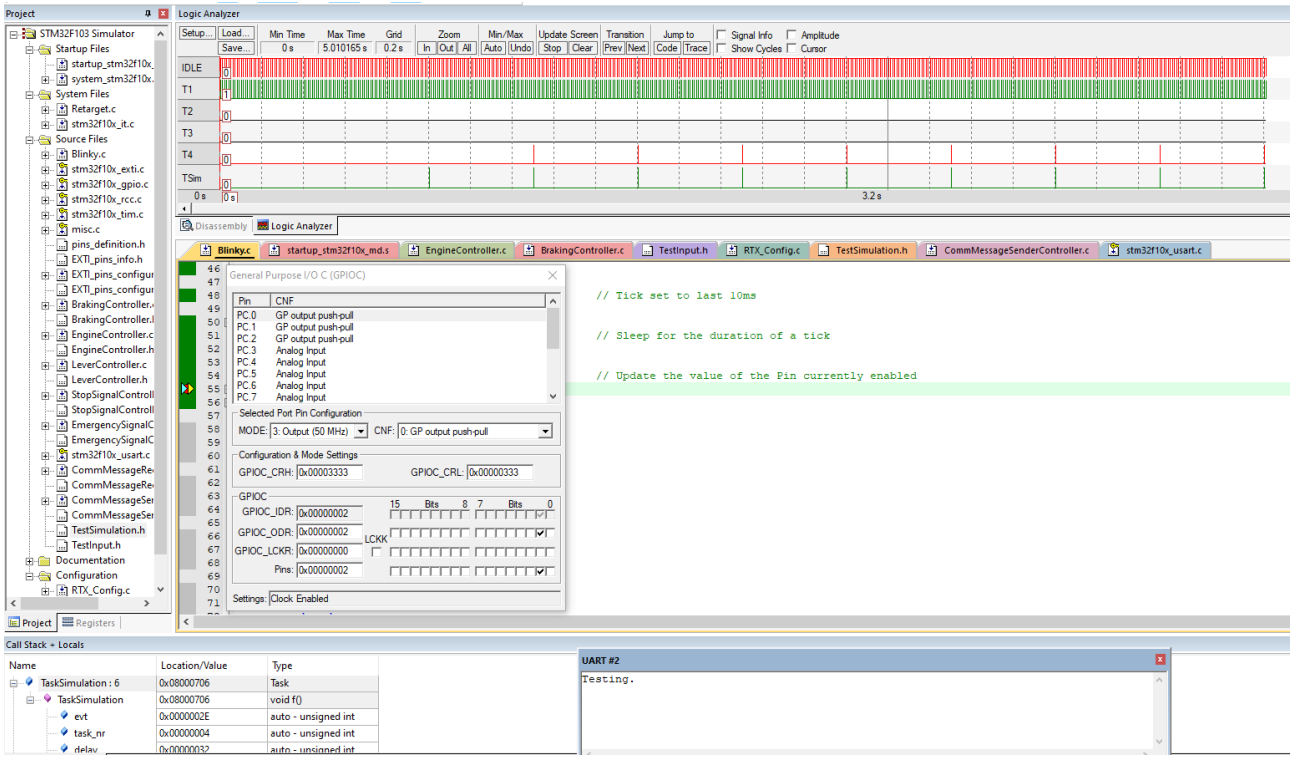
1. {PIN_MAX_ACCEL, 1, 50},
2. {'T', 4, 50},
3. {'e', 4, 50},
4. {'s', 4, 50},
5. {'t', 4, 50},
6. {'i', 4, 50},
7. {'n', 4, 50},
8. {'g', 4, 50},
9. {'.', 4, 50},
10. {PIN_IDLE_LEVER, 1, 50}

Providing to the system the input 2, the results are



Specifically, the system prints the char received in input into the USART2 serial line. The GPIOC port output was set because of the input 1 and it is not changed after the input 2 is received.

Providing to the system the inputs from 3 to 9, the results are



Specifically, the system prints all the chars received in input into the USART2 serial line. Furthermore, since the input 1 simulated the lever position +3 and more than 4s are spent from that input, the controller also changes the GPIOC output value from Pin2 to Pin1 (according to the first safety specification).