FSM programmer guide

**Authors: Unai E.**

**Date:** November 6, 2018

**Version:** 1

**Revision:** 1

**Created:** November 6, 2019

**Last Revised on:** February 25, 2020

**Last Revised by:**

**Contents**

[1. Revision History 3](#_Toc24552538)

[2. List of Acronyms 4](#_Toc24552539)

[3. AIM 5](#_Toc24552540)

[4. General Description 6](#_Toc24552541)

[4.1 Description of “fsm\_api.h” file 6](#_Toc24552542)

[4.2 Description of generated \*.h file 6](#_Toc24552543)

[4.3 Description of generated \*.fsm file 7](#_Toc24552544)

[4.4 Application example 8](#_Toc24552545)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Version | Change Description |
| 2019-11-06 | Unai E. | 1.0 | Initial Revision |
| 2019-11-13 | Unai E. | 1.1 | Changes for platform abstraction |

# List of Acronyms

|  |  |
| --- | --- |
| Acronym | Definition |
| FSM | Finite State Machine |
| API | Application Programming Interface |

# AIM

The aim of this document is to explain the usage of the code generated by FSM export tool.

# General Description

The FSM exporting tool generates a \*.fsm (and, optionally, \*.h) file containing all the APIs and data types necessary for coding a finite state machine. This tool also provides the programmer with a “fsm\_api.h” file. This header file can be found in the tool repository (it is not generated by the tool because it is common to all the FSM). Have a look at “fsm\_exporting\_tool\_manual.docx” for further information about FSM exporting tool.

## Description of “fsm\_api.h” file

The FSM exporting tool allows the user the option to add debug logs to the state machine and/or to be multithread state machines. Therefore, the tool provides an API for registering callback functions which implement these functionalities (i.e mutex lock and unlock functions to protect the variables in the multithread state machines). Thus, the user is abstracted from the exported code.

This header file will be common to all different finite state machines. It contains the next APIs:

* **fsm\_log\_state\_change\_fn**: this function is called when the finite state machines changes states. It can be used for logging state changes.
* **fsm\_mutex\_lock\_fn**: this function is used for locking the protection of shared data at multithreading finite state machines.
* **fsm\_mutex\_unlock\_fn**: this function is used for unlocking the protection of shared data at multithreading finite state machines.

If any of the user option (debugging or multithreading) is not selected, there is no need to register the callback function.

## Description of generated \*.h file

This file contains all the data types related to the previously designed finite state machine:

* *file\_name***\_fsm\_t**:

Struct type data that defines the FSM. It contains the previous and current **states** of the FSM, **events** that have been triggered, pointer to the information of the **FSM** **API** and pointer to the **user data**. “User data” can be used as data exchange between application file and FSM API callback functions.

* *file\_name***\_fsm\_st\_t**:

Enumerate type data that defines all the states of the FSM.

## Description of generated \*.fsm file

This file contains the function declarations derived from the actions defined in the Dia design:

* void *file\_name*\_*name\_of\_action***\_action**(*file\_name***\_fsm\_t**\* fsm):

This function will contain the code of each action. The action can be an “entry action”, “do action”, “exit action” or “transition action”. It must be defined at the FSM main code (\*.c file).

* bool *file\_name*\_*name\_of\_action***\_action\_able**(*file\_name***\_fsm\_t**\* fsm) {Just for “do actions”}:

This function will return *true* if the “do action” is enabled to work. It must be defined at the FSM main code (\*.c file).

Apart from these function declarations, the \*.fsm file contains all the APIs necessary for the usage of the previously designed finite state machine:

* void *file\_name***\_tick**(*file\_name***\_fsm\_t**\* fsm):

This is the main function of the FSM. It checks why the FSM has been triggered (either through an event or through a “do action” enablement) and execute the actions that had been previously designed.

* void *file\_name***\_fire\_***name\_of\_event***\_event**(*file\_name***\_fsm\_t**\* fsm):

It triggers the event.

* void *file\_name***\_fsm\_init**(*file\_name***\_fsm\_t**\* fsm, **fsm\_api\_t**\* fsm\_api, **void**\* fsm\_user\_data):

This function initializes the FSM and saves pointers to API data and user data.

* *file\_name***\_fsm\_st\_t** *file\_name***\_get\_state**(*file\_name***\_fsm\_t**\* fsm):

It returns the current state of the FSM.

## Application example

*HarryPotter\_main\_file.c* is an application example of the \*.fsm and \*.h generated by FSM exporting tool. Double click on the icon to open them.

