**TEXASS: Software description**

EEROS is a fully open-source software solution for the development of robots used in research and as well for commercial used robot systems. It is supposed to be a flexible and universal usable platform which is safe and easy to use. It contains three sub systems: The Control System, the Safety System and the Sequencer Framework.

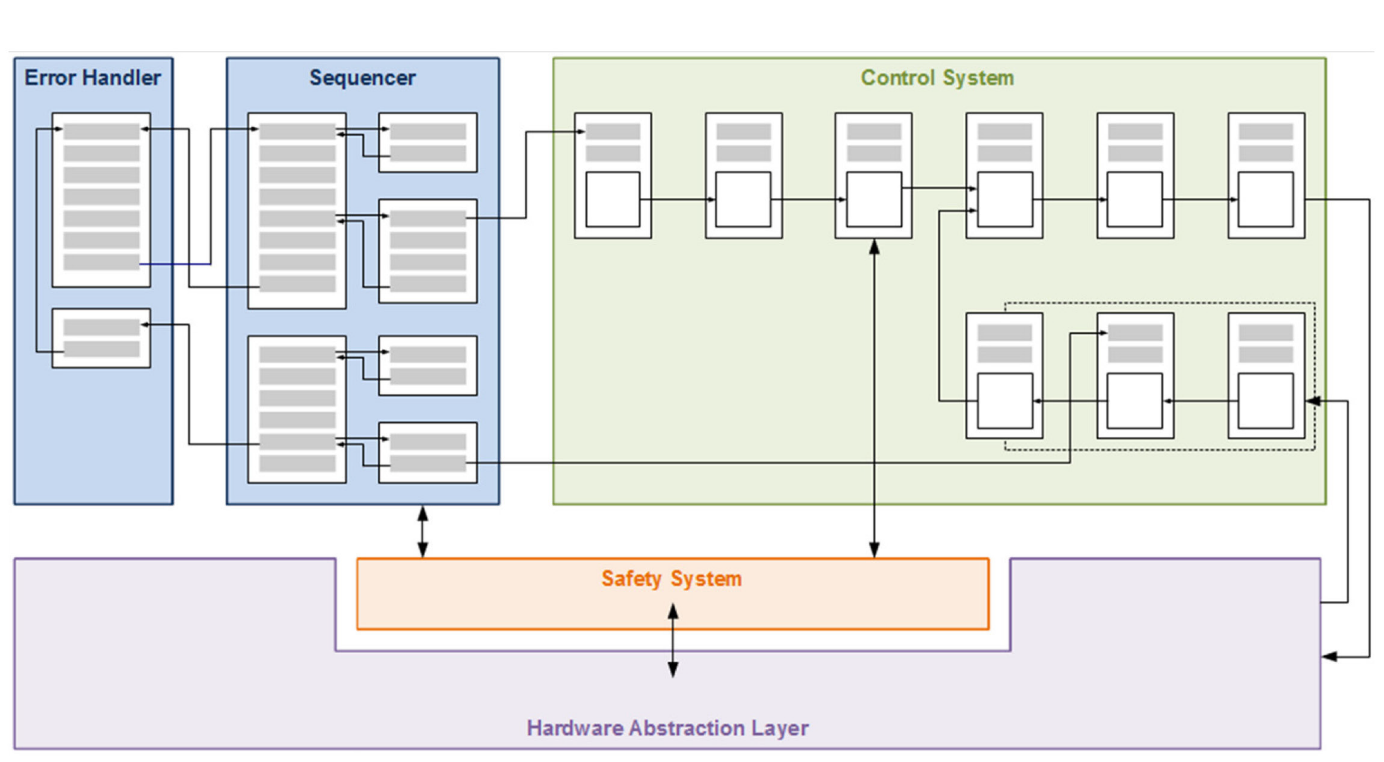


Fig: EEROS Framework Structure

A [control system](http://www.eeros.org/eeros-architecture/control-system/) runs in real time, and consists of blocks which contain algorithms working in parallel and connected by signals. It takes care of the calculation of the reference position for the robot joints, of the path planning and coordinate transformations between different reference systems.

The [safety system](http://www.eeros.org/eeros-architecture/safety-system/) is designed as a series of strictly hierarchical security levels. It monitors critical inputs and outputs and it ensures, for example, that a fall back to the safety level "Emergency" leads immediately and reliably to stopping the robot, no matter who or what invoked the emergency event.

The [sequencer](http://www.eeros.org/eeros-architecture/sequencer/) does not run in parallel but sequentially, since robot movements must be carried out strictly in succession. However, there are some exceptions, for example when an unforeseen event occurs and the current movement must be stopped.

In the current implementation, a main file initializes hardware input and outputs, gets an instance of them and also of the control system, the safety system and initializes the sequencer.

Safety System:

The safety system ensures absolutely safe operation of the controlled robot. Safety levels are the key to this. Each safety level has a unique number and a unique name. A higher level generally means a higher risk in case of error.

For each level the developer can define several events. An event can be triggered by the sequencer, a block in the control system, another event or by the safety system itself. Further, such an event can be tied to an external input such as an emergency button.

An event can lead to a change in the current safety level. Most events are recognized only when the current safety level has a certain number. If not recognized, they are simply ignored. However, there are also events which act independently from the current level, e.g. emergency stop.

In addition to safety-critical inputs, the user can also define safety-critical outputs. Such outputs are used for brakes or enable signals. Every safety level determines the state of those outputs. The point is that these critical outputs can only be driven by the safety system and never by the sequencer or a block of the control system.

Several levels and possible events have been defined for this application. Some levels monitor the robot during parking and homing actions, some allow to start or stop the robot controller, and some others, like moving and teaching, allow to monitor the robot during movements.

The safety system runs at a rate of 1 ms.

Critical outputs, which are set every 1 ms, are the watchdog signal, capable of detecting any software failure, the enable signals for the drives and the commands to engage or release the brakes.

Critical inputs are also read every 1 ms and are in this case the emergency button signal, the end of line sensors, and the range of the joint angles, in order to avoid that the robot goes to a wrong position.

In the table below all safety levels and events are listed. For every level, possible events and next levels are defined. Additionally, for every level, safety inputs and outputs are set. (In attachment the related excel file).

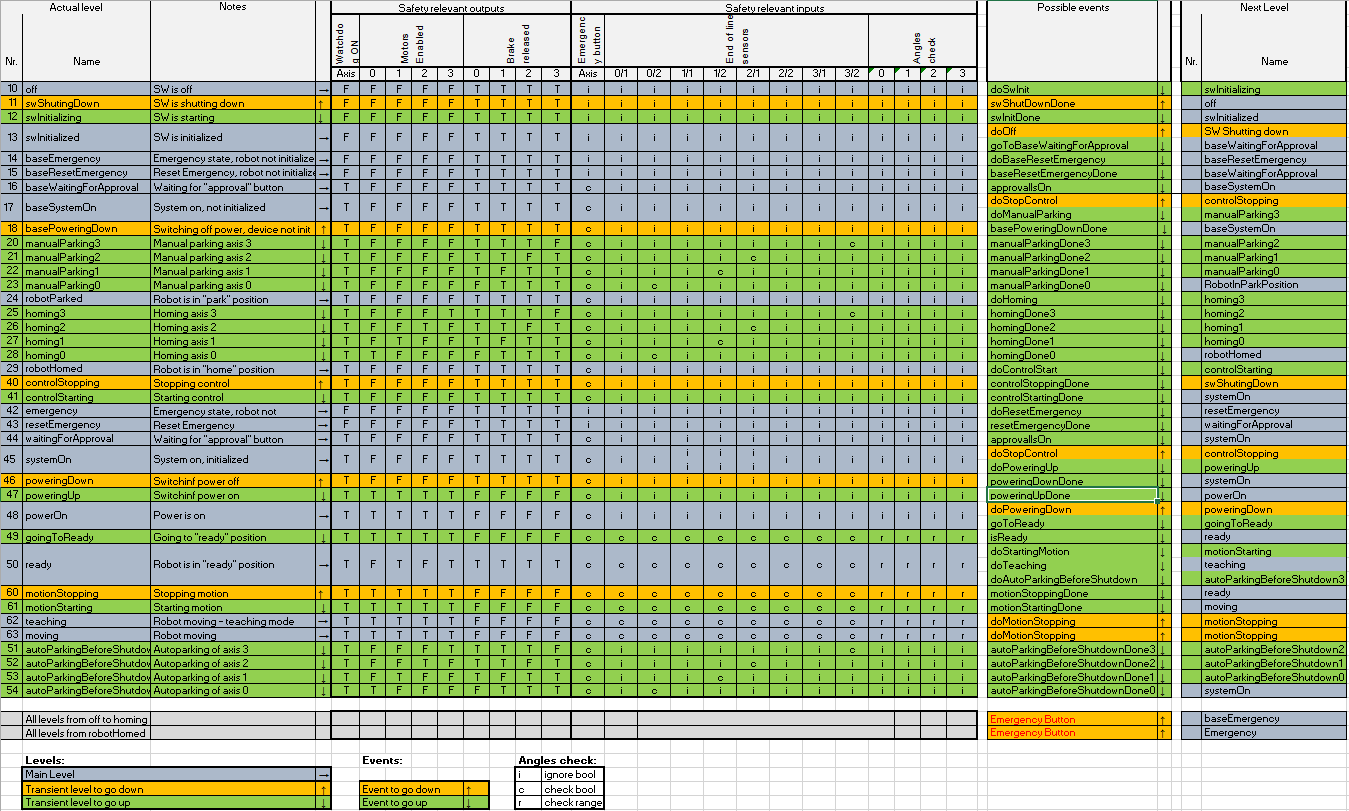


Fig: 2 Safety System - levels and events

Control System:

The control system allows building control structures. The signal flow can be described in a similar way to well-known tools such as Simulink or Xcos. This description has to be made in textual form (programming), not with graphical aid. The developer uses blocks and connects them.

In this case the cascaded PID control strategy was chosen to control the robot. The structure of the control system is represented graphically below.

The path of the robot can be planned either in joint space (joint angles values as references) or in Cartesian space (xyz coordinates in space as references). Additionally, the robot can be controlled with a joystick by the user. This function is useful when coordinate systems have to be saved with a taster tool or with the camera tool.

The encoder provide a position information, which is used as feedback in the control loop, to check if the robot is tracking the reference position set correctly.

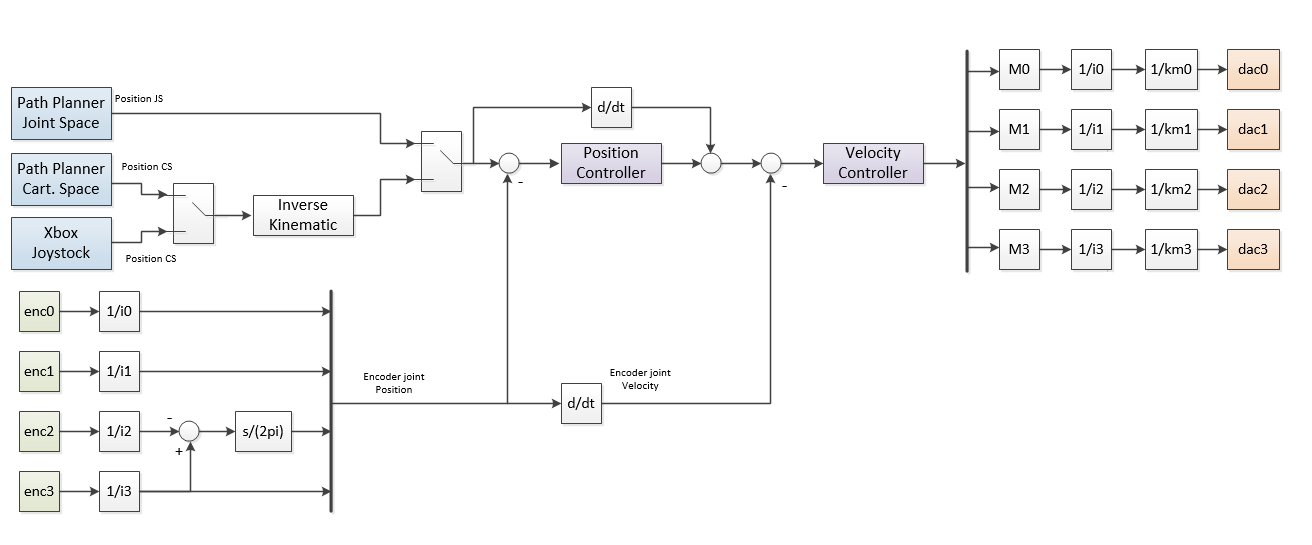
The output of the controller is multiplied by the mass matrix (M), the motor constant (1/km) and the gear ratio (1/i), in order to get a current signal, which will be sent to the drives.

Fig: Control System Structure

Sequencer:

The sequencer is used to create a simple description of a course of action. A sequence is composed by a linear series of steps.

In this specific application the main sequence begins by parking the robot, homing it and loading all necessary reference system descriptions and calibration tables.

After this preliminary steps, the user can choose if he wants to change some reference systems, calibrate the working area, or run the TEXASS sequences. In this case there are several possibilities: the user can run the whole sequence to complete the process, or he can run every single sequence separately.

A joystick mode is also possible, in order to be able to move the robot manually with the provided joystick. Finally, it is possible to park the robot and shut it down.

A graphical description of the main sequence and the TEXASS subsequences is reported below.

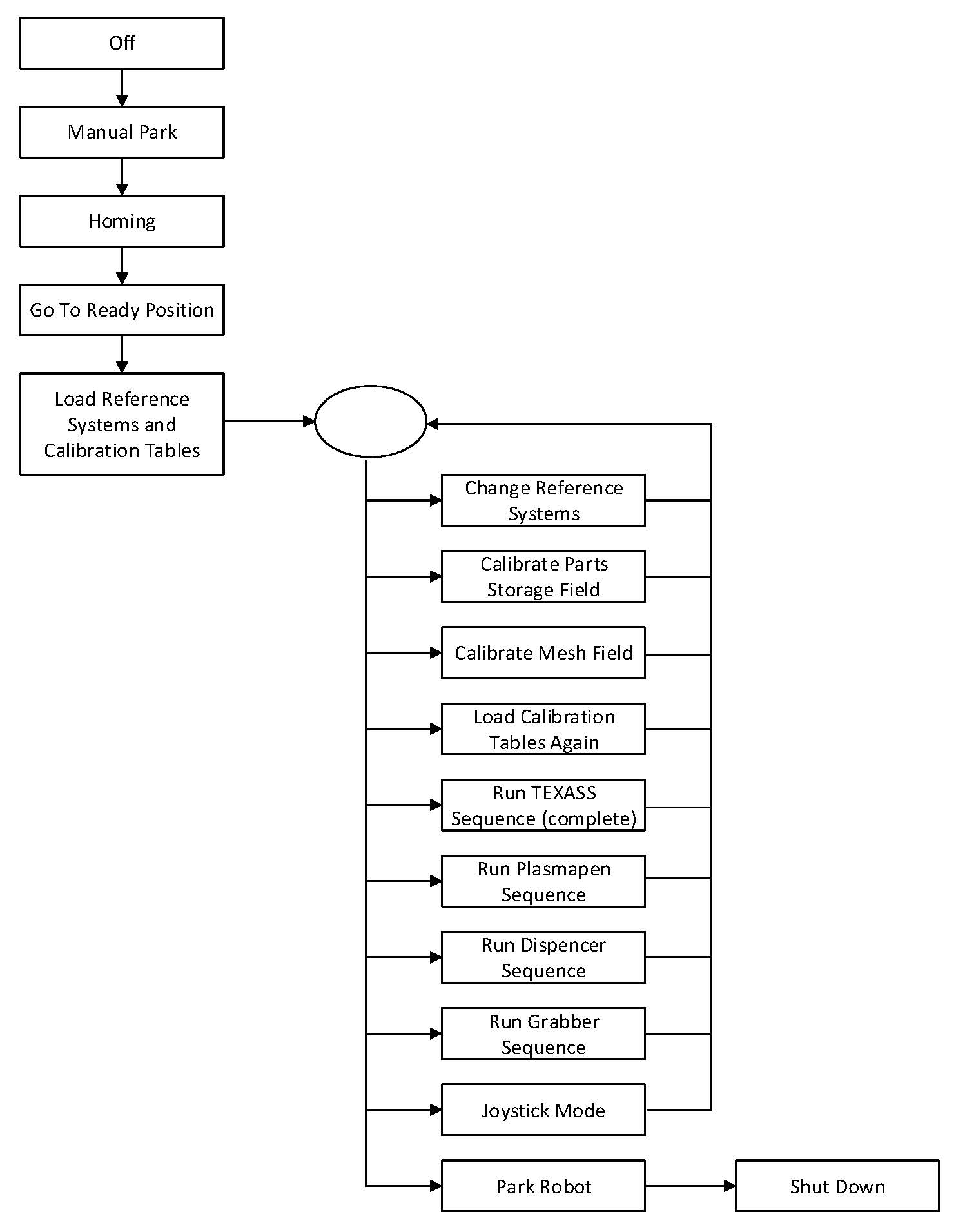


Fig: Main Sequencer

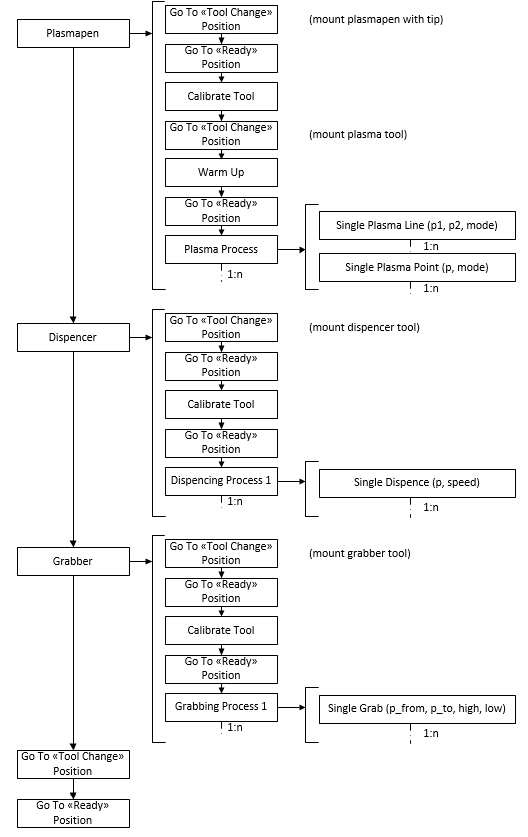


Fig: TEXASS Sequence