QUICK START GUIDE **CLEARSY**

MADE IN FRANCE

CLEARSY
Safety Solutions Designer

Quick Start Guide

CLEARSY SAFETY PLATFORM

CONTACT@CLEARSY.COM

1 DISCLAIMER

This quick start guide contains information on how to start using the CLEARSY Safety Platform. This document does not supersede the regular user manual and the application of this guide is not sufficient for reaching a SIL3 or SIL4 safe design. Especially, the Safety Related Conditions detailed in the C_D720 User manual must be met to safely design and deploy a safety critical system.
In addition, the following conditions shall also be met for operating the designed system:

- Knowledge of any applicable local regulations.
- Proper technical implementation of the safety instructions detailed in this manual performed by qualified personnel.

CLEARSY will not be held liable for severe personal injuries, damage to property or the environment caused by any of the following:

- Unqualified personnel working on or with the devices.
- De-activation or bypassing of safety functions.
- Failure to comply with the Safety Related Application Conditions detailed in this manual.

Except for the Safety Related Application conditions detailed in the C_D720 User manual CLEARSY does not accept any liability for the other information contained in this document.

2 OVERVIEW

The CLEARSY Safety Platform is a framework that intends to speed up and ease the design of vital embedded applications with a SIL3/SIL4 integrity level. The CLEARSY Safety Platform is made of:

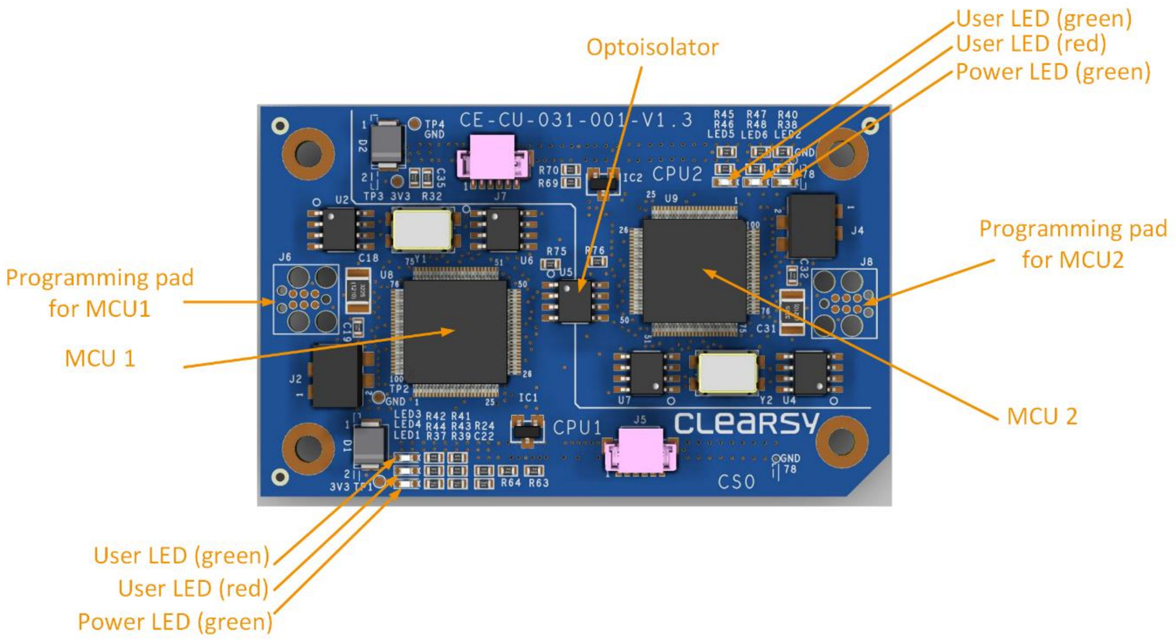
- A software library called CLEARSY Safety Platform library (CSPlib).
- A compatible single board vital computer (72.5x45x12mm) based on a composite safety architecture (CS0).

The hardware consists of two low power 32bits microcontrollers that continuously cross-check. The software library contains a set of routines which help to address the common challenges of a composite vital computing architecture. The board also offers two EEPROMs of 1kBits, temperature sensors, voltage monitoring devices and two times two debug LEDs.

The purpose of the CLEARSY Safety Platform is to save the effort for securing a computing node and to ease the development of certified systems. Especially, by following the guidelines and the exported constraint of the C_D720 user manual and complying to the Safety Related Application Conditions (SRACs), you will save yourself the time and effort required for addressing the following hazards/failures:

- Corruption of the FLASH memory containing the program.
- Corruption of the configuration registers (Special function registers or co-processor registers) of the microcontroller.
- Transient corruption of the RAM memory resulting in a wrong runtime execution.
- Compilation toolchain error in the vital software.
- Time related constraint failure due to clock drift or error.
- Flashing error on your final hardware target.
- Damage on the Arithmetic and Logic Unit of the microcontroller

In one word, using the CLEARSY Safety Platform will allow you to focus only on proving the safety of your application and software and free your mind of any hazard related to the underlying hardware on which is running your application.



3 REFERENCES / KEY DOCUMENTS

This document can be read in parallel to other user documentations that highlight a specific topic.
The documents related to the CLEARSY Safety Platform are listed below:

- **C_D710 End user API documentation.** This document defines the software interfaces the end user has to implement in order to use the CLEARSY safety platform
- **C_D712 Generation verification procedure.** This document explains the procedure to follow in order to validate a binary generated by the toolchain. This procedure shall be followed each time the end user wants to build a new SIL4 software.
- **C_D720 User manual.** This document provides all the detailed explanation on how to use the CLEARSY Safety Platform and also provides all the Safety Related Application Condition you need to meet.
- **Product Datasheet.** This document is a summary of the physical properties of the CLEARSY Safety Platform.
- **C_D250 Application example** which is an application note that shows on a minimalist example how to implement a vital system based on the CLEARSY Safety Platform. This document can be used as a guide on the process to follow to reach a compliant SIL4 certificate of a product using the CLEARSY Safety Platform.

4 PRE-REQUISITES

To start using the CLEARSY Safety Platform, you will need at least the following items:

- For the hardware:
  - A Safety Computer Hardware (CS0).
  - A programming tool like the ICD3 or ICD4 sold by Microchip.
  - A motherboard on which you can plug the Safety Computer board. In order to ease the learning curve CLEARSY sells small boards (A5 paper sized) called demonstrator boards. These demonstrator boards berth the CS0 and provide a few sets of non-vital interfaces (2x 16 inputs + 2x 16 outputs + debug interface). If you use this board you will also need an USB-A to miniUSB cable.

- For the software:
  - **No compiler nor complex setup is required**
  - Your preferred text editor for writing B vital software and C low level layer.
  - MPLAB IPE for flashing the firmware on the target (available on Microchip's website)
  - a prompt with standard commands, especially *scp* and *ssh*. This can be obtained with any of the following tools/software: *Putty, git bash, Windows Subsystem Linux*, *Linux distribution*, …

5 STRUCTURE OF THE VITAL LIBRARY

The CLEARSY Safety Platform software can be retrieved through the compilation script with the *-dl* option. In this section, the term "script" refers to a short bash script provided by CLEARSY that retrieves the library and compiles your project.

When you execute the following command from a prompt (username corresponds to the login of your account):

```
user@computer:~/ ./script.sh -dl username
```

a new folder named CSPlib_vx.y.z (where x,y,z are matching the version associated to your script) will be created on your hard drive at the same level as the script. The structure of the folder is the following:

- **CSPlib_vx.y.z**
  - **csplib**
    - **doc:** This directory contains all the relevant documentation you will need to get comfortable with the use of the CLEARSY Safety Platform.
    - **examples**
      - **GreenLed:** Simple hello world project that does nothing except properly calling all the built-in mechanism and toggling a LED in a non-vital context
      - **SafetyFlasher:** Simple example of a flashing vital (SIL4) output (covered by the C_D250 application example document)
    - **framework_library:** This directory *CS0.lib.tar* which is the archive containing the official release of the CLEARSY Safety Platform. You should use this tar package for any development you want to carry out.

Note: The examples directory contains a folder per example. Each example is ready to compile, in the sense that the tar file of the library has already been copied into the working tree of each example. So if you want to try out one of the examples you simply need to copy and paste the script into the corresponding example (GreenLed folder or SafetyFlasher folder) and trigger the compilation.

The tar file embeds several folders. Details the purpose and the contents of these folders can be found in the *C_D720 User manual* section 7.2.

6 RUNNING YOUR FIRST EXAMPLE

The GreenLed example, available in the official package, provides a minimalist example of how to schedule all the safety tasks of the CLEARSY Safety Platform to avoid shutdown. This implementation is an example you can adapt to start developing your own application. In order to run this example you need to copy the compilation script at the root of the GreenLED folder. Then invoke the script with the *-c* option your *username* and *all* as target.

```
user@computer:~/ ./script.sh -c username target
```

The compilation process will be triggered on the remote server and once done, you will find the files mcu1.hex and mcu2.hex in the newly created *dcc_build* directory. These files are the binary file for each processor of the CS0 board.

At this step you simply need to power the board by plugging the CS0 board into the demonstrator board, connect the USB (two green LEDs should light up indicating that the CS0 is powered). You can then use an ICD4, dedicated cable and the MPLAB IPE software (from Microchip) for loading the previously .hex files on the microcontroller.

The electrically reset the board and you will see two additional green LEDs blinking indicating that the example is properly running.

7 AVAILABLE SERVICES

The CLEARSY Safety Platform offers, through its API, some services that you can use right out of the box. This section intends to quickly present them and provides a link to the corresponding section with detailed explanations.

- **Vital services:**
    - Vital voter of memory payload. This service should be used when you want to secure an array of memory, for instance if you want to emit a vital message on a network.
    - Vital voter for register. This service should be used when you want to apply a vital value into a register of the microcontroller.
    - Minimum timing guarantee. This service should be used if you want to ensure, before triggering an action, that a minimum duration elapsed since a given event.
      Deadline guarantee. This service should be used if you want to ensure that a given permissive state is not maintained longer than a timeout or if you want to guarantee that a given action is executed before a timeout.
- **Non-vital services:**
    - Utility functions for type conversion and endianness conversion.
    - Communication layer between the two microcontrollers of the Safety Computer board. This should be used if you want to exchange any kind of data (vital or non-vital) between the two processors.
    - Utility functions for controlling the debug LEDs of the Safety Computer Board.

   - Utility functions to retrieve the globally unique serial number. This should be used if you want to have a unique number in your design or if you need to send serial number to CLEARSY for troubleshooting.
   - Utility functions to retrieve temperature from the built-in sensor.
   - Utility functions for reading or writing into the 1kB EEPROM provided by the Safety Computer Board.

8 REQUIRED SOFTWARE INTERFACES

You have to implement a minimal number of non-vital and vital functions otherwise the compilation will fail. The CLEARSY Safety Platform library relies upon the user defining a number of functions.
These functions are all listed in the file *api.h*. The details of the purpose of each of these interface functions are provided in the dedicated document *C_D710 End user API documentation.*

A synthesis is provided here for the sake of completeness:

- **api_init()**
- **api_postInit()**
- **api_main()**
- **api_messageDispatcher(const uint8_t * p_payload, uint16_t p_size, uint8_t p_type, uint8_t p_emitter, uint8_t p_recipient,uint32_t p_emittedMessageId, uint32_t p_acknowledgedMessageId)**
- **void api_getVersion(uint8_t * p_versionX, uint8_t * p_versionY, uint8_t * p_versionZ)**
- **void api_bootloader(uint32_t argc, uint32_t * argv[])**
- **void nr_f_printDebugMessage(uint32_t code, uint32_t p1, uint32_t p2, uint32_t p3, uint32_t p4, uint32_t p5, uint32_t p6)**
- **void nr_f_printLastDebugMessage(uint32_t code, uint32_t p1, uint32_t p2, uint32_t p3, uint32_t p4, uint32_t p5, uint32_t p6**)
- **void printDebugText(uint8_t * p_message, uint8_t p_size)**
- **user_watchdogTimer()**

Details about each of these functions can be found in the *C_D720 User Manual*

9 DESIGN CONSTRAINTS

If you want to benefit from the safety mechanisms integrated inside the CLEARSY Safety Platform library, you have to call periodically some tasks. In the context of the CLEARSY Safety Platform a task has to be understood as a C non-vital entry point which manages a given aspect of the system.

The following table summarizes the name of the task and the minimum expected call frequency (you can obviously call the function more often).

TASK (NAME OF FUNCTION TO CALL)	MINIMUM REQUIRED CALLS PER PERIOD
csp_vitalVariable_task()	1 call every 450ms
csp_ALUIntegrity_task()	1 call every 10sec
csp_flashIntegrity_task()	1 call every 60ms
csp_RAMIntegrity_task()	1 call every 585ms
csp_clockDrift_task()	1 call every 500ms
csp_diffMCU_task()	1 call every 10sec
csp_commTask()	NA

The following table presents the execution time of the different tasks.

TASK	EXECUTION TIME
csp_vitalVariable_task()	23µs
csp_ALUIntegrity_task()	29.38µs
csp_flashIntegrity_task()	14.60µs
csp_RAMIntegrity_task()	21µs
csp_clockDrift_task()	140.60µs
csp_diffMCU_task()	< 100µs

Safety Solutions Designer

contact@clearsy.com
www.clearsy.com

320 Avenue Archimède
Les Pléiades III Bat A
13100 Aix-en-Provence - France

Tél. +33 (0)4 42 37 12 70

Fax. +33 (0)4 42 37 12 61