Gen 2_2 , 3 Suggestions

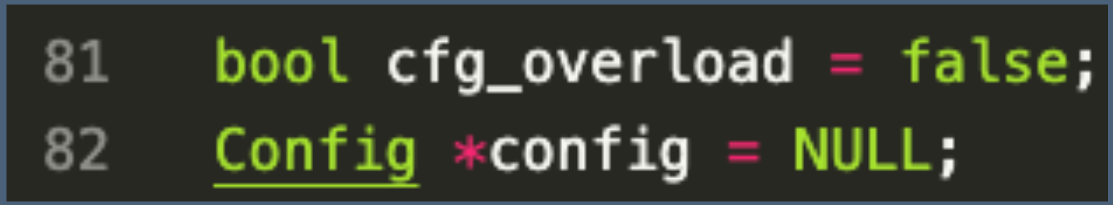
Neekon | 22.08.2025

inquis-gen-2-gen-3-suggestions.md

```
# Inquis Gen 2 - Gen 3 Codebase Suggestions for Zach S.
*The following suggestions are derived from code reviewed in the main branch of inquis_gen_2_2 <877e567d7b8383b5dc6bacf775bbc0a010b0521e> and inquis_gen_3_0 <9dfa6d65ccb128b5cd307f8082603f34d296797b>*

## GEN3

### CMS
#### CLI
**Questions:**
- 81 what is cfg_overload for
- _<funName> -> local and <funName> -> global?
- 85 n t s to check sd_card_mount() and see why/if it is okay to emit non-blocking error if error thrown
- 101 n t s to check check_string_allocates(0,0) if the check is safe / comprehensive for potential string memory leaks
- 96 config_file_contents bbstr not deleted in else branch
- 115 CMD_CHECK_IS is a redundant assert (we need only one system wide realistically) -> CMD_CHECK_IS leads back to a function in CLI; couldn't we call that printf directly with a conditional?
- 117 What is GPIO_Typedef; what is it for
- 120 CMD_CHECK_IS used instead of if (...) {emit_log_comment_record(...)} -> We should have one way to handle errors visible to all modules; should even be a separate module
- 104 function never called
[15h10 22 aug 2025 ~ faster to drop these questions and comments into powerpoint from the getgo]
**Suggestions:**

#### LOG
**Questions:**
- note to self to check where emit_log_error_once is called; may be unsafe given warning about static/dynamic string usage
- note to self to make suggestion regarding base case of empty comment in emit_log_comment_record, which returns instead of emitting associated default comment, allowing for accidentally implementing silent errors (should not be possible)
-
**Suggestions:**

### HANDLE

### COMMON
```

cms_cli

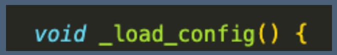
What is cfg_overload for



```
bool cfg_overload = false;
Config *config = NULL;
```

cms_cli

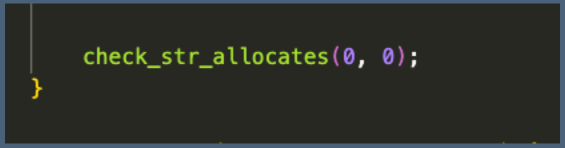
- _<funName> -> local and <funName> -> global?



```
void _load_config() {
```

cms_cli

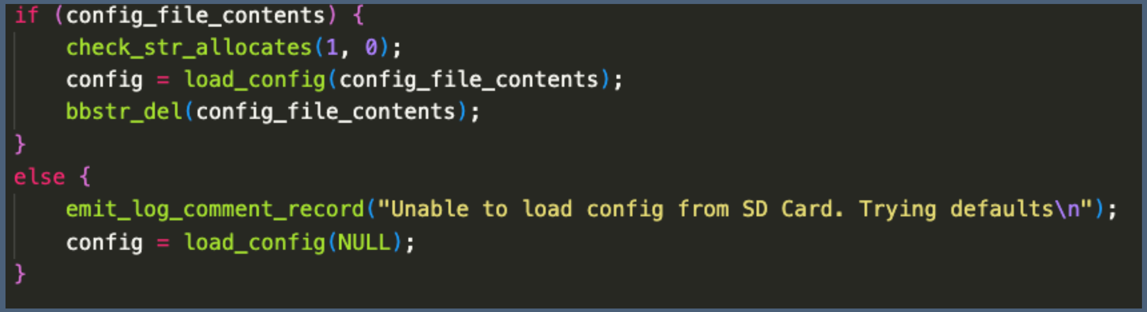
- 101 n t s to check check_string_allocates(0,0) if the check is safe / comprehensive for potential string memory leaks



```
    check_str_allocates(0, 0);
}
```

cms_cli

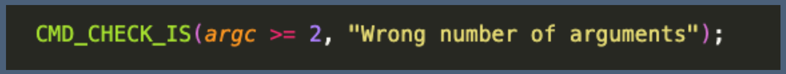
- 96 config_file_contents bbstr not deleted in else branch



```
if (config_file_contents) {
    check_str_allocates(1, 0);
    config = load_config(config_file_contents);
    bbstr_del(config_file_contents);
}
else {
    emit_log_comment_record("Unable to load config from SD Card. Trying defaults\n");
    config = load_config(NULL);
}
```

cms_cli

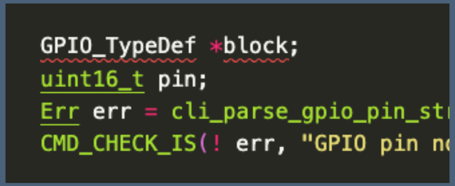
- 115 CMD_CHECK_IS is a redundant assert (we need only one system wide realistically) -> CMD_CHECK_IS leads back to a function in CLI; couldn't we call that printf directly with a conditional?



```
CMD_CHECK_IS(argc >= 2, "Wrong number of arguments");
```

cms_cli

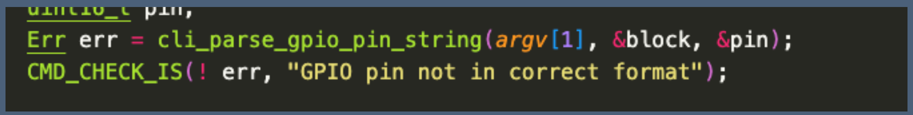
- 117 What is GPIO_Typedef; what is it for



```
GPIO_TypeDef *block;
uint16_t pin;
Err err = cli_parse_gpio_pin_st
CMD_CHECK_IS(! err, "GPIO pin n
```

cms_cli

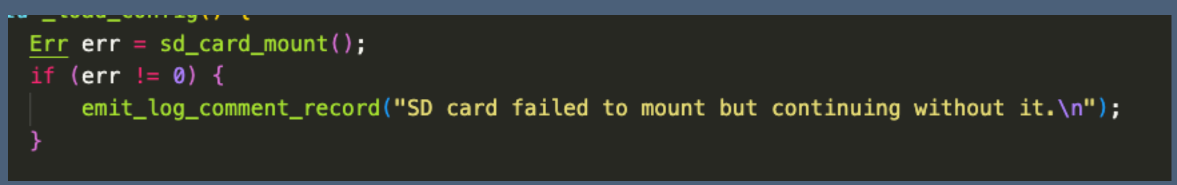
- 120 CMD_CHECK_IS used instead of if (...) {emit_log_comment_record(...)} -> We should have one way to handle errors visible to all modules; should even be a separate module



```
uint16_t pin;
Err err = cli_parse_gpio_pin_string(argv[1], &block, &pin);
CMD_CHECK_IS(! err, "GPIO pin not in correct format");
```

cms_cli

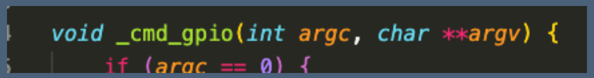
- 85 n t s to check sd_card_mount() and see why/if it is okay to emit non-blocking error if error thrown



```
id _load_config() {
  Err err = sd_card_mount();
  if (err != 0) {
      emit_log_comment_record("SD card failed to mount but continuing without it.\n");
  }
```

cms_cli

- 104 function never called



```
void _cmd_gpio(int argc, char **argv) {
    if (argc == 0) {
```

cms_cli

cms_cli

cms_cli

cms_cli

cms_cli

cms_cli

cms_cli

LOG

- note to self to check where emit_log_error_once is called; may be unsafe given warning about static/dynamic string usage
- note to self to make suggestion regarding base case of empty comment in emit_log_comment_record, which returns instead of emitting associated default comment, allowing for accidentally implementing silent errors (should not be possible)