PRÜFUNGSSTUDIENARBEIT (PStA)
SS 2021

Technische Hochschule Deggendorf

Fakultät Elektrotechnik, Medientechnik und Informatik

Subject: C in der automobilen Softwareentwicklung

Topic: Zentraler Fehlerspeicher

Author:

Faizan Mahmood

Supervisor:

Prof. Thomas Limbrunner

Matriculation:

00096480

Submission date: Deggendorf, 30.06.2021

I assure you that I wrote this thesis on my own and that I have only used the specified sources and resources and that I have marked literal and analogous quotations as such. The present study paper was not submitted to any other examination.

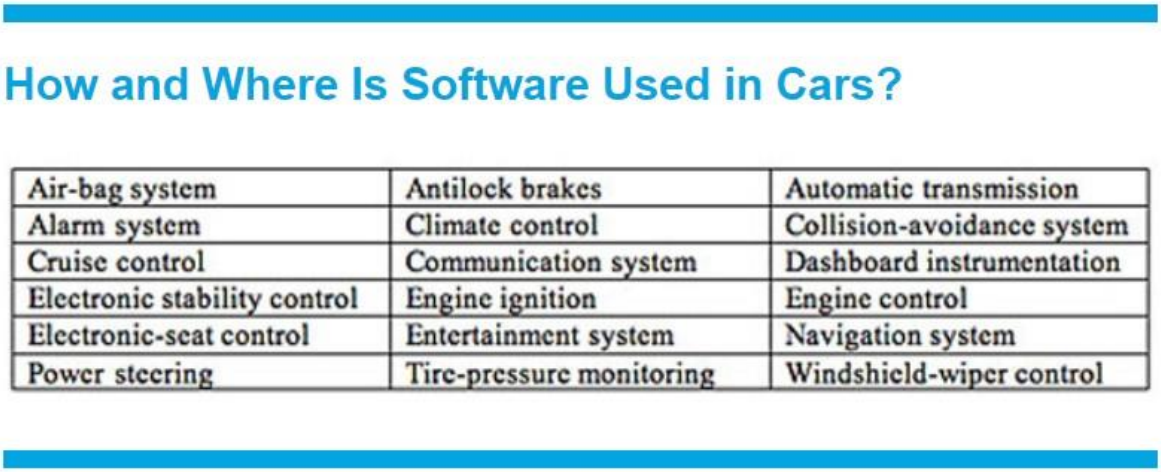
I agree that the results of this examination study paper can be used free of charge for research and teaching purposes.

Deggendorf,21.01.2021 Mahmood, Faizan

Table of Content

1. Introduction

In Automotive Industry modern cars have been around for quite sometime. A modern car is the one which has some sort of control unit that controls its functions or operations. In an average modern car, it has generally 30 to 50 electronic control units (ECUs) [1]. This ECU is an embedded circuit with several microprocessors that run on user defined software to control the electronic hardware. Every ECU deals with a different component in the car. For example, there is one ECU that deals with the braking system or acceleration only. Fig. 1 shows some common ECU units in a car.



How and Where Is Software Used in Cars?

Air-bag system	Antilock brakes	Automatic transmission
Alarm system	Climate control	Collision-avoidance system
Cruise control	Communication system	Dashboard instrumentation
Electronic stability control	Engine ignition	Engine control
Electronic-seat control	Entertainment system	Navigation system
Power steering	Tire-pressure monitoring	Windshield-wiper control

Fig. 1 – Common ECU units in a car [1]

All the ECUs in car can communicate with each other and share the error information with each other. Every ECU then decides how should it respond to its to its functionality e.g., full functionality, reduced functionality etc., depending on the information from other ECUs and its previous information.

In embedded systems one main unit is the error handling unit. It is necessary to check the errors arising in the system memory and cope with them before a serious problem occurs. The error handling unit runs all the time and check for any new errors that appear in memory. A software error handling module is therefore, required to be designed to continuously check for the upcoming errors in the central error memory of an automotive. All the different ECU units have different Module ID and every error in all the ECU units have a unique Error ID. Which means that if an Error ID is known all the other information about that error can be determined. Different errors have different Severity level and according to the Severity level their functionality should be defined.

2. Methodology

The software module is designed in C programming language using windows platform. To keep the module running all the time a super loop is used. The first understanding after gathering information form the problem statement was that errors are required to be handled. And every error has some attributes which tell all the details regarding this specific error. Table I.

Table I - Error attributes and their description

Error Attributes	description
Error ID	ID of the error
Module ID	ID of the module which has error
Error Frequency	Number of times an error occurred
Error Severity	Severity level depending on default condition and error frequency
Time Stamp	Most recent time at which the error occurred
Time Delta	Difference between most recent error time and its last occurrence time

Since all these attributes are related to every single error. So, a new structure datatype named Parameters is defined to store all the information regarding a single error in one place. There are four modules, and every module communicates with the other module to handle the error properly.

3.1. Central Error Memory Module

This is the main module of the error handling software. It starts when the car switch is turned on. It checks continuously if there is an error or not even after the engine switched on. If the switch is on but the car has not started it calls engine start count function to keep an eye on the engine start count. If the engine start count exceeds 3 attempts, it limits the user to start the car for 10 secs. After 10 secs user again gets the 3 attempts to start the car. In case the car has started the engine start count function is no more called. However, the error check still works in the background to check for errors. If due to some reason, there occur an error before the engine starts. The central error memory function check if the error is coming from engine module or the sensor module. In case of either error case error handling module is called, and the engine id and module id are passed as function arguments. Flow chart of this module is in Fig. 2

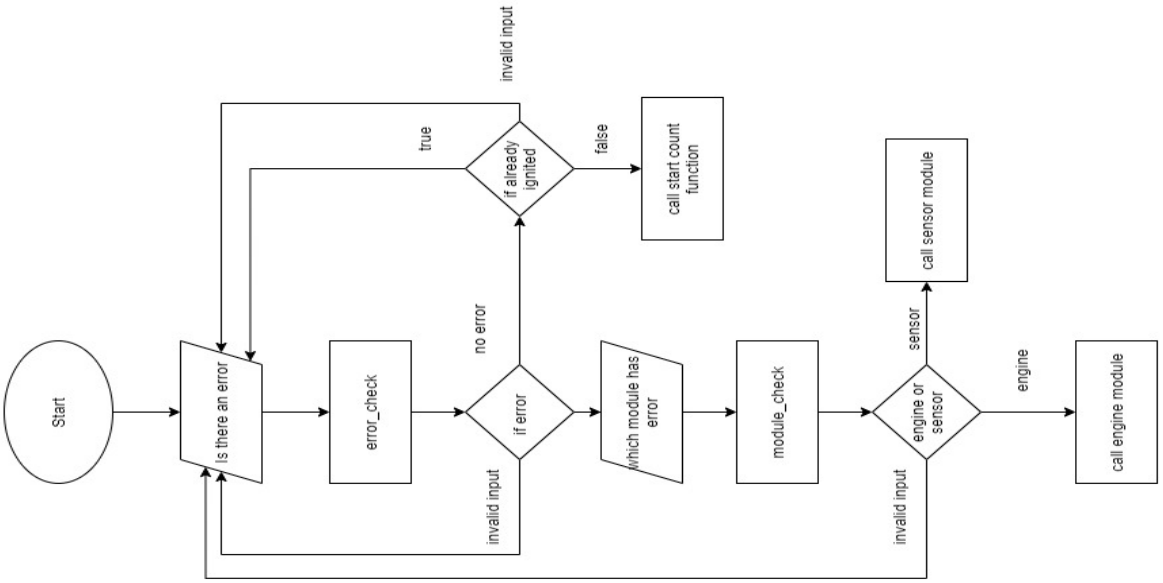


Fig. 2 Flow Chart of Central Error Memory Module

3.2. Engine and Sensor ECU Module

These module deals with the errors related to engine and sensors accordingly. If an error is encountered, they tell the central error memory module that an error has occurred and subsequently call the error handling module. All the error processing is then done by the error handling module. Fig. 3 and Fig. 4 show the flowchart of engine and sensor error module respectively.

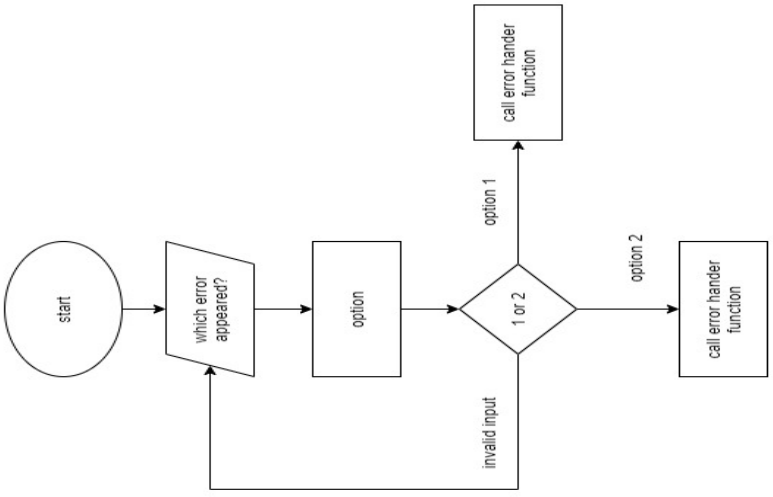


Fig. 3 Flow Chart of Engine Error Module

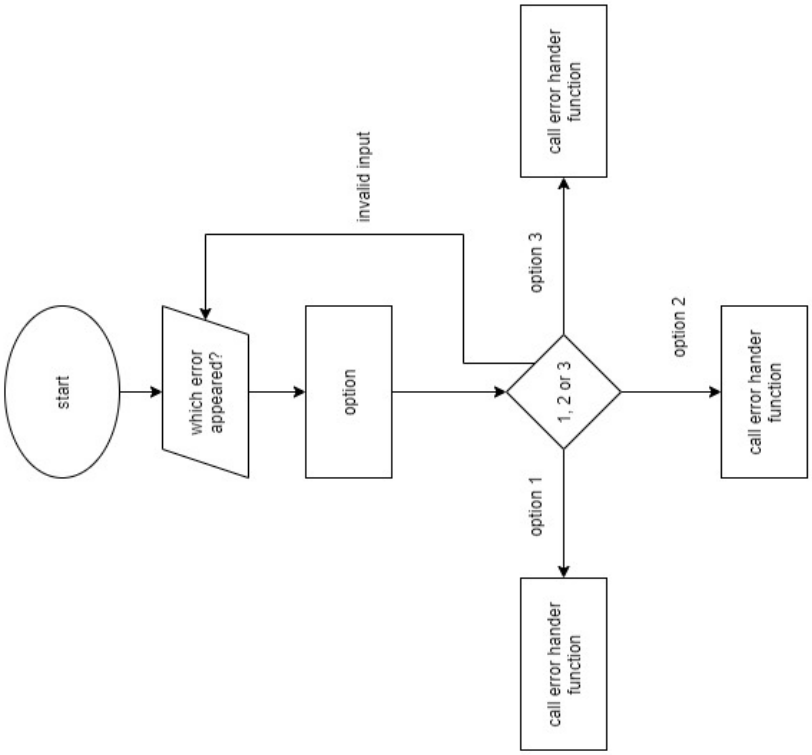


Fig. 4 Flow Chart of Sensor Error Module

3.3. Error Handling Module

This is the module which is responsible for handling all the errors that arise in any ECU module. All other modules call error handling module whenever they encounter an error. In this module the error handler function receives the error id and module id as input parameters and then it checks if there is already an error in the memory or not. If is no error in the memory, then it calls the error initializer function which allocates same in memory for the error. The memory is allocated using calloc() function which allocates memory dynamically at run time and return a void pointer to the base address. In this case the pointer is of structure type with all the parameters related to an error. Memory is allocated for all the errors at once. Which means all the errors are stored in consecutive memory blocks of size of the structure.

Once the error is initialized the error initializer function is not called unless the error module is reset by the user. After this the error handler function check which error has occurred. According to the error id the function displays on the screen the description of the error. Then it checks if that same error has exited before or not. If the error has not existed, it skips the time delta calculation and continue to the next code line.

Then error handler function calls the error remove function. The error remove function checks if the error should be healed or not. If the time delta of error is more than 30 secs it removes the error otherwise the error remains in the memory. After this the control returns to the error handler function. Now error set function is called which fills up all the parameter fields of error

structure. Error frequency is incremented also. In order to fill the element “error severity” error get function is called which takes the parameters “error id” and “error frequency” and then return the error severity depending on the default error severity and the error frequency. After the error set function fills all the attributes of the error structure, the control goes back to error handler function and then it calls the error display function to display all the error attributes to the user. The flow chart of it is shown in Fig. 5

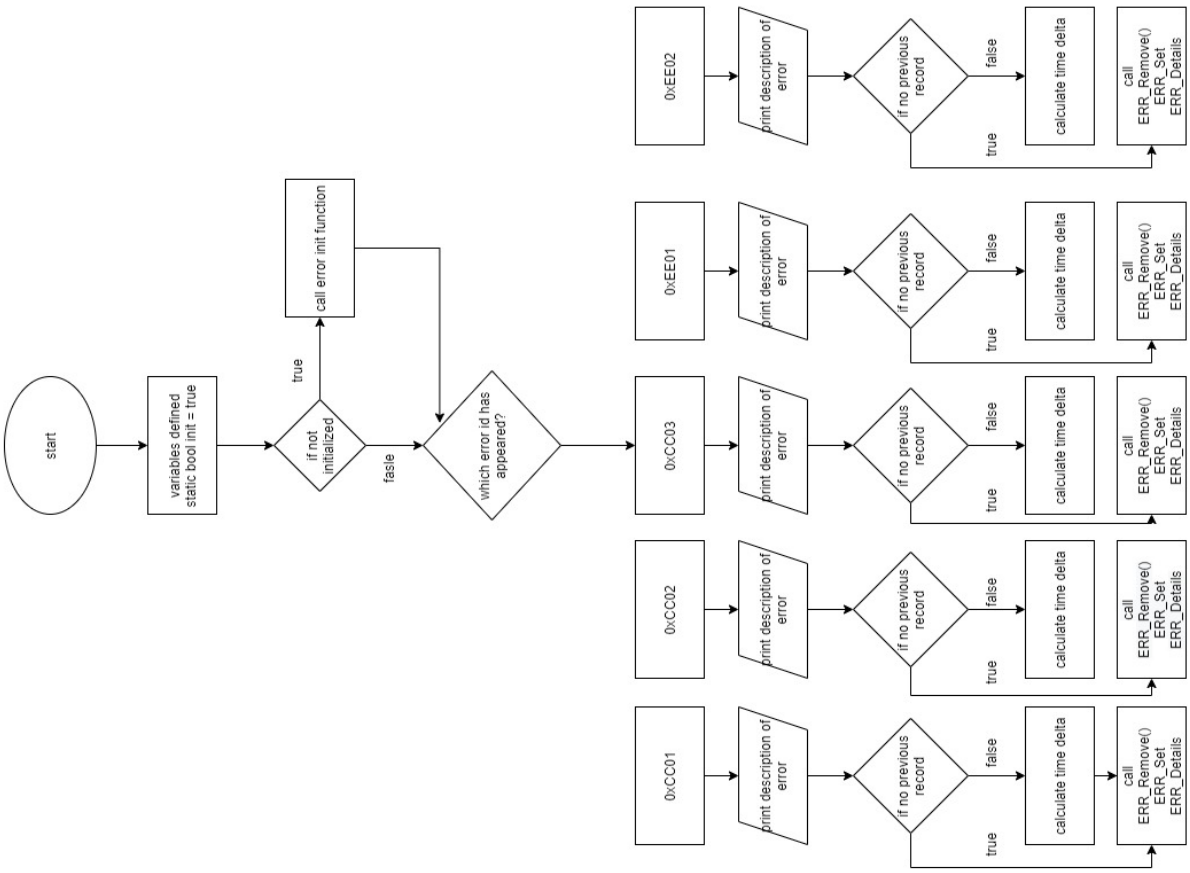


Fig. 5 Flow Chart of Error Handling Module

Also, in error handler function it is checked if the error frequency or error severity has reached it limit or not. If the limit is reached it interrupts the user to either reset the system or exit the system. Fig. 4 shows its flow chart. The severity is defined as “enum” and the values are in Table III.

Table II - Error attributes and their description

Severity	Information	Low	Medium	High	Critical	Finish
value	0	1	2	3	4	5

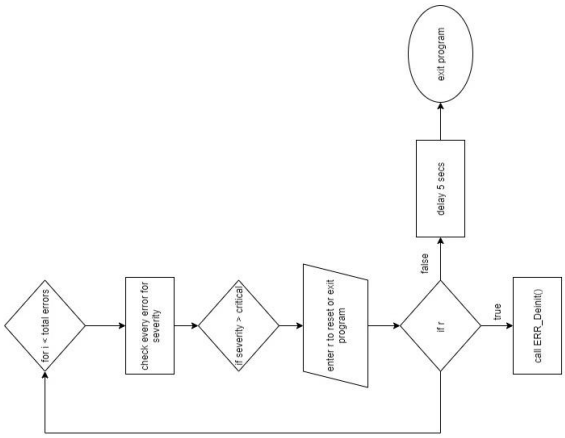


Fig. 6 Flow Chart of Error limit check

4. Results

There are several test cases but only a few a shown here. The results of the test cases are perfectly fine according to the code written.

```
################################################################################
#####                          Central Error Memory                        #####
################################################################################



Is there an error? Press y/n   y

Is it an engine module error or sensor module error? Press e/s e

Which Error code has appeared:
1) EE01
2) EE02        1

Error initialized.

Engine Oil needs to be changed. Please chnage it at first priority

################################################################################

Error ID          ee01
Module ID         ee
Error frequency:  1
Error severity:   3
Time Stamp:       285878828
Time delta:       0

################################################################################



Is there an error? Press y/n
```

Fig. 7 Output screen for first error

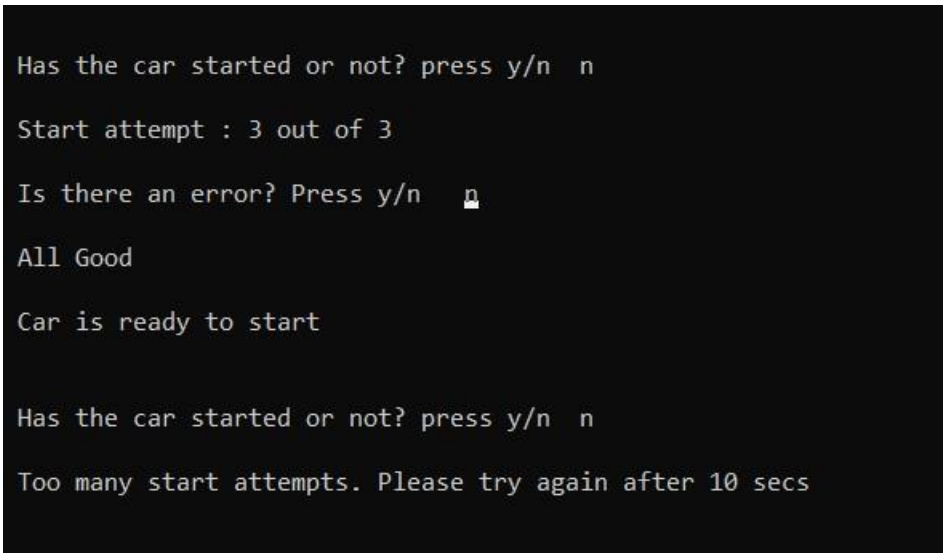


Fig. 8 Output screen engine start count

5. Discussion of Results

According to the code implementation the results show the same results as expected. However, if a different approach would be used the results could have been better in performance and execution time. Memory is allocated for all the errors at once. Which means all the errors are stored in consecutive memory blocks of size of the structure. If memory would be allocated dynamically for each error separately this would have caused less memory usage and hence the software performance would be better. Also, the healing process could be improved by slowly recovering the error in step by step cool down. In current software the errors get removed completely after they do not occur for a specific time difference.

6. Summary and Outlook

The gist of the topic is that in today's era semiconductor chips are very cheap and they make very good, embedded circuits. These embedded circuits are used an almost every electrical device from a digital to the biggest ship in the ocean. To operate these embedded devices a programming language is needed to instruct them. Most commonly used language is C and it is still maintaining its legacy. In embedded devices the most common unit is of error handling. In this project a similar handler has been designed that handles the automotive ECU errors and warns the user before any severe or critical situation arises.

The error handling in embedded systems will always be a demand. Because as the complexity increases the rate of error also increases. In future this software module can be modified to make it more reusable and general purpose. Also, the lags discussed in result discussion

section can be overcome. Like better error severity management and a better error memory management.

7. Bibliography

[1] Robert N. Charette, "This Car Runs on Code," 01 Feb 2009. [Online]. Available: https://spectrum.ieee.org/transportation/systems/this-car-runs-on-code/. [Accessed 27 06 2021].