ソフトロバスト性設計ポイント

共通

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **作成** | **審査** | | | **承認** |
| Neusoft |  |  |  |  |

**変更履歴**

|  |  |  |  |
| --- | --- | --- | --- |
| バージョン | 変更日 | 変更者 | 変更内容 |
| 0.1 | 2023-12-1 | Neusoft | ALL |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**目次**

[1 はじめに 4](#_Toc150967706)

[1.1 概要 4](#_Toc150967707)

[1.2 適用範囲 4](#_Toc150967708)

[1.3 用語定義 4](#_Toc150967709)

[1.4 関連文書 5](#_Toc150967710)

[1.5 本書の位置づけ 5](#_Toc150967711)

[1.6 ソフトウェアアーキテクチャの定義 5](#_Toc150967712)

[2 設計ポイント 6](#_Toc150967713)

[2.1 アーキテクチャパターン 6](#_Toc150967714)

[2.1.1 COMM-ARCH-001 6](#_Toc150967715)

[2.1.1 COMM-ARCH-002 7](#_Toc150967716)

[2.1.1 COMM-ARCH-003 8](#_Toc150967717)

[2.2 アプリケーションプラットホーム 9](#_Toc150967718)

[2.2.1 COMM-APP-001 9](#_Toc150967719)

[2.3 異常処理 10](#_Toc150967720)

[2.3.1 COMM-FAIL-001 10](#_Toc150967721)

[2.4 Virtual Funtional BUS(AUTOSAR CP) 11](#_Toc150967722)

[2.5 Abstraction Layer 12](#_Toc150967723)

[2.6 Common API,ADL,IDL 13](#_Toc150967724)

[2.7 Complex Drivers 14](#_Toc150967725)

[2.7.1 COMM-DRIVER-001 14](#_Toc150967726)

# はじめに

## 概要

本書は、車載ソフトウェアアーキテクチャ設計に対する通信設計ポイントを定義する、ソフトロバスト性のために、車載部品の脆弱性を低減するための設計ポイントを定める必要がある。

## 適用範囲

本書の適用範囲を図 1‑1 に示す。

本書の対象は、車載ソフトウェアアーキテクチャ設計活動である。

Gateway

ECU

ECU

ECU

DCU

Cockpit Domain

適用範囲

ECU

ECU

ECU

DCU

Body Domain

ECU

ECU

ECU

DCU

Other Domain

図 1‑1 適用範囲

## 用語定義

本書で用いる用語を解説する。

|  |  |
| --- | --- |
| **用語** | **解説** |
| DCU | ドメイン制御ユニット |
| IPC | Inter Process Communication  プロセス間通信 |
| RPC | Remote Process Calling  プロセッサ間通信 |

表 1‑1用語定義

## 関連文書

本書の関連文書を以下に示す。

|  |  |  |
| --- | --- | --- |
| **番号** | **ファイル名称** | **文書名称** |
| １ | XXXX | XXXX |
|  |  |  |
|  |  |  |
|  |  |  |

表 1‑2関連文書

## 本書の位置づけ

本書の位置づけを下図に示す。

## ソフトウェアアーキテクチャの定義

ソフトウェアアーキテクチャの定義を下図に示す。

Heterogeneous Multicore SoC

s

Hypervisor

OS1

Kernel

Middleware

Container

Service

Container

Service

OS2

Kernel

Middleware

Container

Service

Container

Service

# 設計ポイント

## アーキテクチャパターン

アーキテクチャパターンの設計ポイントについて、以下の通り示す。

### 2.1.1 COMM-ARCH-001

|  |  |  |
| --- | --- | --- |
| **ID** | **COMM-ARCH-001** | |
| 設計ポイント | 编码时要考虑到设定编译器编译优化后可能发生的逻辑改变 | |
| 説明 | 设定编译器优化后，某些代码的既定逻辑、执行顺序可能发生改变 | |
| 目的/理由 | 软件编码过程中要注意保持良好的编码习惯，防止正常的逻辑被编译器给优化掉 | |
| 不具合例 | I2C是半双工通信，每次想要读取数据前，都需要先写入读取的地址。然后才能读出需要的数据。 SDK封装的读写接口一般都是相同的接口名： int i2c\_xfer(uint8 ch,uint8 slave\_addr, uint8 in\_len uint8 \*in\_buf, uint8 out\_len, uint8 \*out\_buf)  使用者传入不同的参数以便驱动执行相应的读写。 不具合发生时，使用者代码如下：  ret = i2c\_xfer(I2C\_CH, ADDR,SndLen,SndBuf,0,NULL); //写入  ret = i2c\_xfer(I2C\_CH, ADDR,0,NULL,Rcvlen,Rcvbuff); //读出  上记代码在编译器优化的时候，第一次的写入动作就会被优化掉，导致读出的数据永远不是期待的值。 | |
| 对策例 | 读数据前，可以加入对上一次通信结果的判断 ret = i2c\_xfer(I2C\_CH, ADDR,SndLen,SndBuf,0,NULL);  if(ret = ok){  ret = i2c\_xfer(I2C\_CH, ADDR,0,NULL,Rcvlen,Rcvbuff);  } | |
| 備考 |  | |
| 適用ドメイン | Cockpit Body | |
| 品質特徴 | 機能適合性 | 機能完全性　機能正確性　機能適切性 |
| 性能効率性 | 時間効率性　資源効率性　容量満足性 |
| 互換性 | 共存性　相互運用性 |
| 使用性 | 適切度認識性　習得性　運用操作性　ユーザエラー防止性　ユーザインタフェース快美性　アクセシビリティ性 |
| 信頼性 | 成熟性　可用性　障害許容性　回復性 |
| セキュリティ | 機秘性　インテグリティ性　否認防止性　責任追跡性　真正性 |
| 保守性 | モジュール性　再利用性　解析性　修正性　試験性 |
| 移植性 | 適応性　設置性　置換性 |

### 2.1.1 COMM-ARCH-002

|  |  |  |
| --- | --- | --- |
| **ID** | **COMM-ARCH-002** | |
| 設計ポイント | 虚拟化平台，容器环境中需要考虑设备文件权限的影响 | |
| 説明 | 在Linux文件系统中，[/sys]是内核的数据映射出来的,是用来管理系统设备的一个内存文件系统。系统中大部分设备的信息，都可以从这个文件系统中获取到，用户空间可以通过这个文件系统来控制和更改设备的状态。  虚拟化平台下，需要合理设计容器环境和宿主机下的设备文件访问权限。既要满足容器环境的需求，又要防止因两个环境同时操作/sys下设备文件导致的资源冲突问题。  Host OS  Container1 OS  Container2 OS  SOC | |
| 目的/理由 | 防止因设备文件权限问题导致的机能异常。 | |
| 不具合例 | LXC虚拟化项目中，在Container OS中执行GPIO导出处理时发生错误，提示错误信息[direction: Read-only file system]。  原因是在Host OS中对Container OS的/sys权限设置为了Mixed模式。Mixed模式下，/sys目录下只有net关联部分是可读可写的，其他部分都是只读的，所以导致了问题发生。  "mount": [  {  "type": "auto",  "param": "cgroup:mixed proc:mixed **sys:mixed**"  },  ] | |
| 对策例 | 在host中对Container OS的配置中，将[sys:mixed]变更为[sys:rw] | |
| 備考 |  | |
| 適用ドメイン | Cockpit Body | |
| 品質特徴 | 機能適合性 | 機能完全性　機能正確性　機能適切性 |
| 性能効率性 | 時間効率性　資源効率性　容量満足性 |
| 互換性 | 共存性　相互運用性 |
| 使用性 | 適切度認識性　習得性　運用操作性　ユーザエラー防止性　ユーザインタフェース快美性　アクセシビリティ性 |
| 信頼性 | 成熟性　可用性　障害許容性　回復性 |
| セキュリティ | 機秘性　インテグリティ性　否認防止性　責任追跡性　真正性 |
| 保守性 | モジュール性　再利用性　解析性　修正性　試験性 |
| 移植性 | 適応性　設置性　置換性 |

### 2.1.1 COMM-ARCH-003

|  |  |  |
| --- | --- | --- |
| **ID** | **COMM-ARCH-003** | |
| 設計ポイント | 防止软件程序与动态链接库中符号表相同 | |
| 説明 | 通常内核装载可执行程序时会使用动态链接器解析所有依赖的动态库，并构建全局动态符号表，重定向函数及变量引用地址。  全局动态符号表包括了，可执行程序的符号表 + 动态链接器符号表 +所有依赖库的符号表。也就是说，如果动态库内部定义一个全局变量或者全局函数，那么默认情况下使用该动态库的主程序中该符号同样有效。  由于动态库是在装载时才进行符号链接，所以编译和链接阶段是不检测动态库和主程序里的符号是否有多重定义问题的。  如果存在主程序和动态库中有重复定义函数或者变量的情况，就会导致程序在运行时出现异常。 | |
| 目的/理由 | 避免动态库与主程序之间符号的二义性 | |
| 不具合例 | DAB数字广播功能中将核心处理封装成了中间件，以动态库的形式存在。  DAB中间件在获取动态标签数据时，需要将从Air信号中解析完成的数据放到一个全局变量中，然后通知到DAB应用程序。但是由于DAB主应用程序中也定义了一个相同名称的全局变量。  DAB应用程序装载时，首先装载的是可执行对象，所以DAB中间件动态库中的全局变量的内存地址变成了主程序中该名称全局变量的地址，当DAB中间件使用该变量时，由于应用程序中之前对该变量进行了清空处理，导致DAB中间件动态库中读取数据异常。 | |
| 对策例 | ①严格按照事业部编码规约中的函数和变量命名规范进行Code  ②动态库编译时，使用-Wl,-Bsymbolic选项将库中符号表私有化  ※关于②，如果主程序通过extern 库中的全局变量的方式，访问库中变量的场合，需要特殊注意；开启Bsymbolic选项后，主程序extern直接访问库中变量将会变成无效。  ※如果只想私有化函数符号，可以使用-Wl,-Bsymbolic-functions  ③降低全局引用的Code，合理使用static | |
| 備考 |  | |
| 適用ドメイン | Cockpit Body | |
| 品質特徴 | 機能適合性 | 機能完全性　機能正確性　機能適切性 |
| 性能効率性 | 時間効率性　資源効率性　容量満足性 |
| 互換性 | 共存性　相互運用性 |
| 使用性 | 適切度認識性　習得性　運用操作性　ユーザエラー防止性　ユーザインタフェース快美性　アクセシビリティ性 |
| 信頼性 | 成熟性　可用性　障害許容性　回復性 |
| セキュリティ | 機秘性　インテグリティ性　否認防止性　責任追跡性　真正性 |
| 保守性 | モジュール性　再利用性　解析性　修正性　試験性 |
| 移植性 | 適応性　設置性　置換性 |

## アプリケーションプラットホーム

アプリケーションプラットホームの設計ポイントについて、以下の通り示す。

### 2.2.1 COMM-APP-001

|  |  |  |
| --- | --- | --- |
| **ID** | **COMM-APP-001** | |
| 設計ポイント | 传感器联动应用功能，应避免受到高频信号影响 | |
| 説明 | 有些车辆应用功能需要接收传感器信号进行相应处理，如果车辆传感器信号的变化周期小、频率高，可能导致和传感器联动的应用无法在一个变化周期内进行完整动作，或者一直处于繁忙状态无法处理其他请求。这种场合，需要在设计阶段进行实车状态下状况分析。可以通过进行降频等处理手段来防止问题发生。 | |
| 目的/理由 | 传感器相关功能应考虑传感器信号频率的影响 | |
| 不具合例 | 音量随速调节功能，是随着车辆速度变化调节系统音量的功能。Sound模块接收车速信息并进行音量调节。  现象：启用音量随车速调节时，Sound模块动作异常，无法进行基本的动作功能。  原因：实车中车速信号变化是100ms周期，导致Sound模块一直在进行随速调节功能，无法响应其他动作 | |
| 对策例 | Sound模块接收到车速变化信号后仅保存当前车速信息，定周期进行随速调节功能动作. | |
| 備考 |  | |
| 適用ドメイン | Cockpit Body | |
| 品質特徴 | 機能適合性 | 機能完全性　機能正確性　機能適切性 |
| 性能効率性 | 時間効率性　資源効率性　容量満足性 |
| 互換性 | 共存性　相互運用性 |
| 使用性 | 適切度認識性　習得性　運用操作性　ユーザエラー防止性　ユーザインタフェース快美性　アクセシビリティ性 |
| 信頼性 | 成熟性　可用性　障害許容性　回復性 |
| セキュリティ | 機秘性　インテグリティ性　否認防止性　責任追跡性　真正性 |
| 保守性 | モジュール性　再利用性　解析性　修正性　試験性 |
| 移植性 | 適応性　設置性　置換性 |

## 異常処理

異常処理の設計ポイントについて、以下の通り示す。

### 2.3.1 COMM-FAIL-001

|  |  |  |
| --- | --- | --- |
| **ID** | **COMM-FAIL-001** | |
| 設計ポイント | 对于请求无应答的ECU，对其设定时应考虑ECU设定异常保护 | |
| 説明 | 有一些ECU的处理机制是对其请求不能得到应答。即ECU1对ECU2发送设定请求，但没有结果返回。一般情况下，ECU2都会设置成功，但如果ECU2设定动作失败，可能会导致ECU1和ECU2的状态、信息不一致，引发问题。  可以在给ECU2发送设定值后，追加一个获取设定值的处理，判断当前的设定值是否设定成功，来确保设定动作执行成功。  ECU1  ECU2  (无应答机制)  Request  No reponse  可能处理失败 | |
| 目的/理由 | 发送给外部ECU的处理，并不能保证外部ECU都可以处理成功，应该追击Failsafe处理 | |
| 不具合例 | ECU1上显示车门的控制信息，ECU2作为车门控制单元。两者通过CAN进行通信。ECU1通过发送车门设置请求到ECU2来同步两者的状态，但  由于ECU2并未执行成功设定，导致ECU1显示的设定项目内容和ECU2的状态不一致。 | |
| 对策例 | 在设定外部ECU后获取外部ECU的最新状态判断是否跟本次设置的内容一致。 | |
| 備考 |  | |
| 適用ドメイン | Cockpit Body | |
| 品質特徴 | 機能適合性 | 機能完全性　機能正確性　機能適切性 |
| 性能効率性 | 時間効率性　資源効率性　容量満足性 |
| 互換性 | 共存性　相互運用性 |
| 使用性 | 適切度認識性　習得性　運用操作性　ユーザエラー防止性　ユーザインタフェース快美性　アクセシビリティ性 |
| 信頼性 | 成熟性　可用性　障害許容性　回復性 |
| セキュリティ | 機秘性　インテグリティ性　否認防止性　責任追跡性　真正性 |
| 保守性 | モジュール性　再利用性　解析性　修正性　試験性 |
| 移植性 | 適応性　設置性　置換性 |

## Virtual Funtional BUS(AUTOSAR CP)

Virtual Funtional BUS(AUTOSAR CP)の設計ポイントについて、以下の通り示す。

### COMM-VFB-001

|  |  |  |
| --- | --- | --- |
| **ID** |  | |
| 設計ポイント | 避免system tick不准导致数据混乱 | |
| 説明 | 由于系统负荷变高,system tick 不准确，导致依赖system tick 定周期发送数据的时序出问题 | |
| 目的/理由 |  | |
| 不具合例 | Task 依赖systikc（2ms)做了个 10ms定周期取传感器值，必须在9ms内取值并完成发送两组数据（Dn.a，Dn.b）。因为高负荷，某次定周期变成了8.8ms，前次两组数据中的D1.b没发送完成，后次两组数据中的D2.a开始发送，导致接收方丢失一次D1.b，error 发生 | |
| 对策例 |  | |
| 備考 |  | |
| 適用ドメイン | Cockpit Body | |
| 品質特徴 | 機能適合性 | 機能完全性　機能正確性　機能適切性 |
| 性能効率性 | 時間効率性　資源効率性　容量満足性 |
| 互換性 | 共存性　相互運用性 |
| 使用性 | 適切度認識性　習得性　運用操作性　ユーザエラー防止性　ユーザインタフェース快美性　アクセシビリティ性 |
| 信頼性 | 成熟性　可用性　障害許容性　回復性 |
| セキュリティ | 機秘性　インテグリティ性　否認防止性　責任追跡性　真正性 |
| 保守性 | モジュール性　再利用性　解析性　修正性　試験性 |
| 移植性 | 適応性　設置性　置換性 |

## Abstraction Layer

Abstraction Layerの設計ポイントについて、以下の通り示す。

|  |  |  |
| --- | --- | --- |
| **ID** |  | |
| 設計ポイント | AUTOSAR系统中，应该避免抽象层级过多，数据链过长，导致数据接收效率低甚至丢失数据 | |
| 説明 | SWC定周期（4ms）从COM取数据，CANIf定周期(polling)将数据放入PDUR，当SWC获取数据的实际耗时超过设定周期4ms，数据未取走时导致数据被覆盖，丢包发生。  CAN BUS  SWC  RTE  COM  PDU Router  CAN TP  CAN Interface  CAN Driver  MCAL  ECU 抽象层  服务层  左图中红色虚框内红色双向箭头是数据从CAN Driver 到SWC 需要经过的链路，无法在10ms内完成传输，新数据将覆盖旧数据  上图中绿色代表已经成功获取数据，黄色代表新获取数据，灰色代表被覆盖的数据（获取失败）  Driver  COM  SWC | |
| 目的/理由 |  | |
| 不具合例 |  | |
| 对策例 | CAN BUS  SWC  RTE  COM  PDU Router  CAN TP  CAN Interface  CAN Driver  物理层  数据链路层  网络层  交互层  如左图，取消PDUR、COM、RTE、SWC之间的数据链路，直接由PDUR通过EVENT的方式将数据传送至SWC，可以大大缩减抽象层的时间开销，保证数据不丢失。 | |
| 備考 |  | |
| 適用ドメイン | Cockpit Body | |
| 品質特徴 | 機能適合性 | 機能完全性　機能正確性　機能適切性 |
| 性能効率性 | 時間効率性　資源効率性　容量満足性 |
| 互換性 | 共存性　相互運用性 |
| 使用性 | 適切度認識性　習得性　運用操作性　ユーザエラー防止性　ユーザインタフェース快美性　アクセシビリティ性 |
| 信頼性 | 成熟性　可用性　障害許容性　回復性 |
| セキュリティ | 機秘性　インテグリティ性　否認防止性　責任追跡性　真正性 |
| 保守性 | モジュール性　再利用性　解析性　修正性　試験性 |
| 移植性 | 適応性　設置性　置換性 |

## Common API,ADL,IDL

Common API,ADL,IDLの設計ポイントについて、以下の通り示す。

|  |  |  |
| --- | --- | --- |
| **ID** |  | |
| 設計ポイント | 防止AD芯片在电源变动的时候采集到非法值，导致程序错误动作 | |
| 説明 | AD芯片在供电电源不稳定时采集到的数据将不可信赖，为了让芯片稳定工作，硬件上要对芯片电源做电路滤波，软件上应对采集到的数据通过算法滤波（算数平均数，最小二乘法），保证数据采集的稳定性。同时要具有对采集到错误数据的对策。  电源变动时，AD采集值不准  AD采集芯片  传感器  MCU  传感器  电源  如果不优化采集的数据，而是直接利用，可能使程序出现错误。 | |
| 目的/理由 |  | |
| 不具合例 | 车机屏幕背光根据AD采集光照传感器的数据进行调节。IGOFF 时AD芯片在掉电瞬间采集到一个很大光照的值并传给了MCU，此时屏幕处于较暗状态，导致屏幕快速高亮，然后熄灭。 | |
| 对策例 | 电源变动时，AD采集通过硬件及软件滤波保障  AD采集芯片  传感器  MCU  传感器  电源  滤波电路芯片  软件滤波器 | |
| 備考 |  | |
| 適用ドメイン | Cockpit Body | |
| 品質特徴 | 機能適合性 | 機能完全性　機能正確性　機能適切性 |
| 性能効率性 | 時間効率性　資源効率性　容量満足性 |
| 互換性 | 共存性　相互運用性 |
| 使用性 | 適切度認識性　習得性　運用操作性　ユーザエラー防止性　ユーザインタフェース快美性　アクセシビリティ性 |
| 信頼性 | 成熟性　可用性　障害許容性　回復性 |
| セキュリティ | 機秘性　インテグリティ性　否認防止性　責任追跡性　真正性 |
| 保守性 | モジュール性　再利用性　解析性　修正性　試験性 |
| 移植性 | 適応性　設置性　置換性 |

## Complex Drivers

Complex Driversの設計ポイントについて、以下の通り示す。

### 2.7.1 COMM-DRIVER-001

|  |  |  |
| --- | --- | --- |
| **ID** | **COMM-DRIVER-001** | |
| 設計ポイント | 驱动管理的共通内存，应考虑使用场景交叉时的内存管理 | |
| 説明 | 有一些驱动程序有自己的内存，如Video驱动等，如果内存管理机制有缺陷或者没有内存管理机制，如内存数据不清除等，在多个驱动使用者交替使用时，可能导致共通内存中残留上次使用的数据，造成数据相互影像。Video驱动的场合，可能由于残留了上一帧的数据，导致画面闪烁的问题。这类问题往往很难调查及定位，所以应在设计阶段考虑这部分内存的管理。  如按照使用者进行内存的分离、或对共通内存的清除机制进行设计。  Driver  Buffer  User1  User2  H/W  1. User1和User2的Buffer分离  2. 清除机制设计 | |
| 目的/理由 | 驱动管理的内存可能没有考虑多使用交替的使用场景，应在设计阶段进行充分考虑 | |
| 不具合例 | Video驱动管理video显示缓存，但没有考虑多用户交替使用场景，多用户进行交替数据刷新时，由于没有对之前的用户Video数据进行清除，内存中数据残留，导致在video映像切换时发生画面闪烁现象。 | |
| 对策例 | 追加内存清除处理 | |
| 備考 |  | |
| 適用ドメイン | Cockpit Body | |
| 品質特徴 | 機能適合性 | 機能完全性　機能正確性　機能適切性 |
| 性能効率性 | 時間効率性　資源効率性　容量満足性 |
| 互換性 | 共存性　相互運用性 |
| 使用性 | 適切度認識性　習得性　運用操作性　ユーザエラー防止性　ユーザインタフェース快美性　アクセシビリティ性 |
| 信頼性 | 成熟性　可用性　障害許容性　回復性 |
| セキュリティ | 機秘性　インテグリティ性　否認防止性　責任追跡性　真正性 |
| 保守性 | モジュール性　再利用性　解析性　修正性　試験性 |
| 移植性 | 適応性　設置性　置換性 |

|  |  |  |
| --- | --- | --- |
| **ID** | **COMM-DRIVER-001** | |
| 設計ポイント | BUS OFF发生，无法复归，总线始终处于BUSOFF状态 | |
| 説明 | 收发器芯片不能自动复归  CAN收发器发生BUSOFF，收发器芯片发送功能锁定，需要手动复归。Bsw层只做了控制器的复归手段，收发器的复归需要人为检测标志位并清除  BSW  MCU  CAN收发器  RTE  SWC  CAN  CAN  AUTOSAR  无自动复归处理  有自动复归处理 | |
| 目的/理由 |  | |
| 不具合例 |  | |
| 对策例 |  | |
| 備考 |  | |
| 適用ドメイン | Cockpit Body | |
| 品質特徴 | 機能適合性 | 機能完全性　機能正確性　機能適切性 |
| 性能効率性 | 時間効率性　資源効率性　容量満足性 |
| 互換性 | 共存性　相互運用性 |
| 使用性 | 適切度認識性　習得性　運用操作性　ユーザエラー防止性　ユーザインタフェース快美性　アクセシビリティ性 |
| 信頼性 | 成熟性　可用性　障害許容性　回復性 |
| セキュリティ | 機秘性　インテグリティ性　否認防止性　責任追跡性　真正性 |
| 保守性 | モジュール性　再利用性　解析性　修正性　試験性 |
| 移植性 | 適応性　設置性　置換性 |