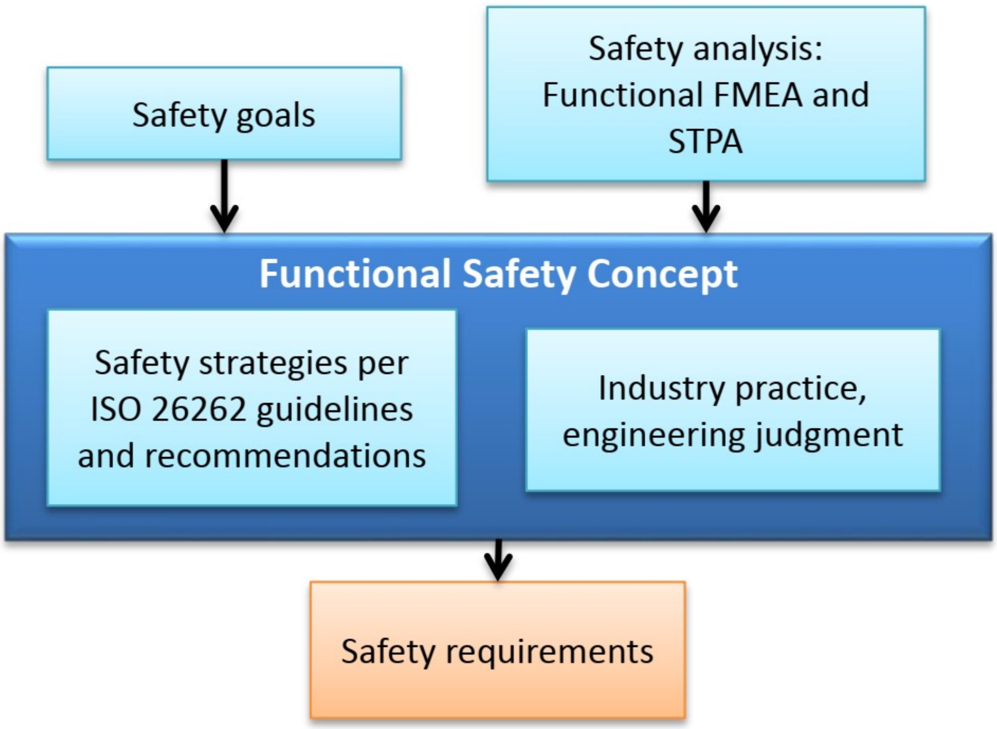
Introduction

Purpose of this exercise is to identify safety requirements and safety measures according to safety functional safety methodology and ISO26262 standard. For purpose of this exercise we will use hypothetical example of electronic accelerator pedal position controller as part of the vehicles electronic acceleration control and throttle control system. The primary purpose of this exercise is to study and analyze the potential hazards that could result with failures impacting the functions of acceleration control systems where the throttle position is controlled electronically. Exercise focus will be on the safety software mechanisms and system architecture that can be used to achieve target ASIL (Automotive Safety Integrity Level) and minimize potential hazard risk. We will not cover various implementation strategies to achieve target ASIL, nor should this be taken as guideline to achieve target ASIL. The ISO 26262 standard provides a flexible framework and explicit guidance to different methods and approaches and various implementations may employ a variety of techniques to do so. Identification of the safety integrity requirements and implementation techniques of these functions are covered at the concept level and are not based on the actual system implementation.

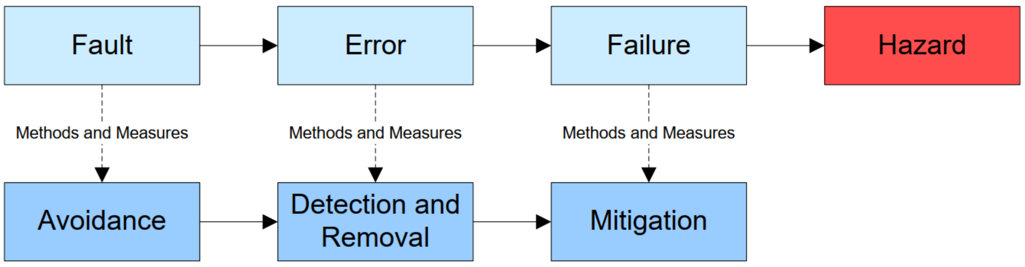
Functional safety concept

The objective of the functional safety concept is to derive a set of functional safety requirements from the safety goals, and to allocate them to the preliminary architectural elements of the system, or to external measures (Part 3 Clause 8.1 in ISO 26262). *Figure 1.* illustrates how the functional safety concept takes into consideration the results from the safety analysis; applies safety strategies, industry practices, and engineering experiences; and derives a set of safety requirements following the established process in ISO 26262.


Safety goals
Safety analysis: Functional FMEA and STPA
Functional Safety Concept
Safety strategies per ISO 26262 guidelines and recommendations
Industry practice, engineering judgment
Safety requirements

Hazard classification

For our example we will identify potential hazardous situations as follows:
- potential uncontrolled vehicle acceleration
- potential uncontrolled vehicle deceleration
- erroneous acceleration/deceleration response
- potential vehicle stalling



Acceleration system and engine control system overview

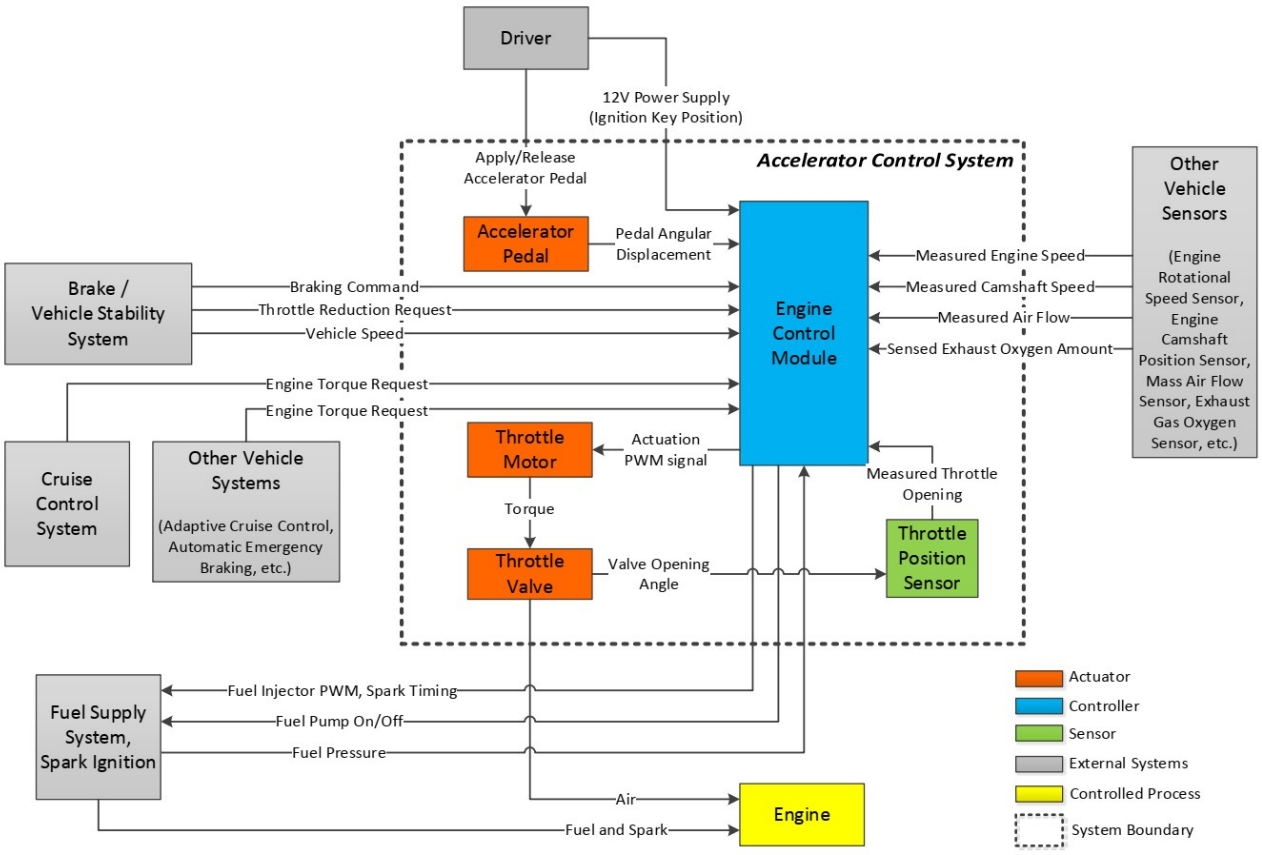
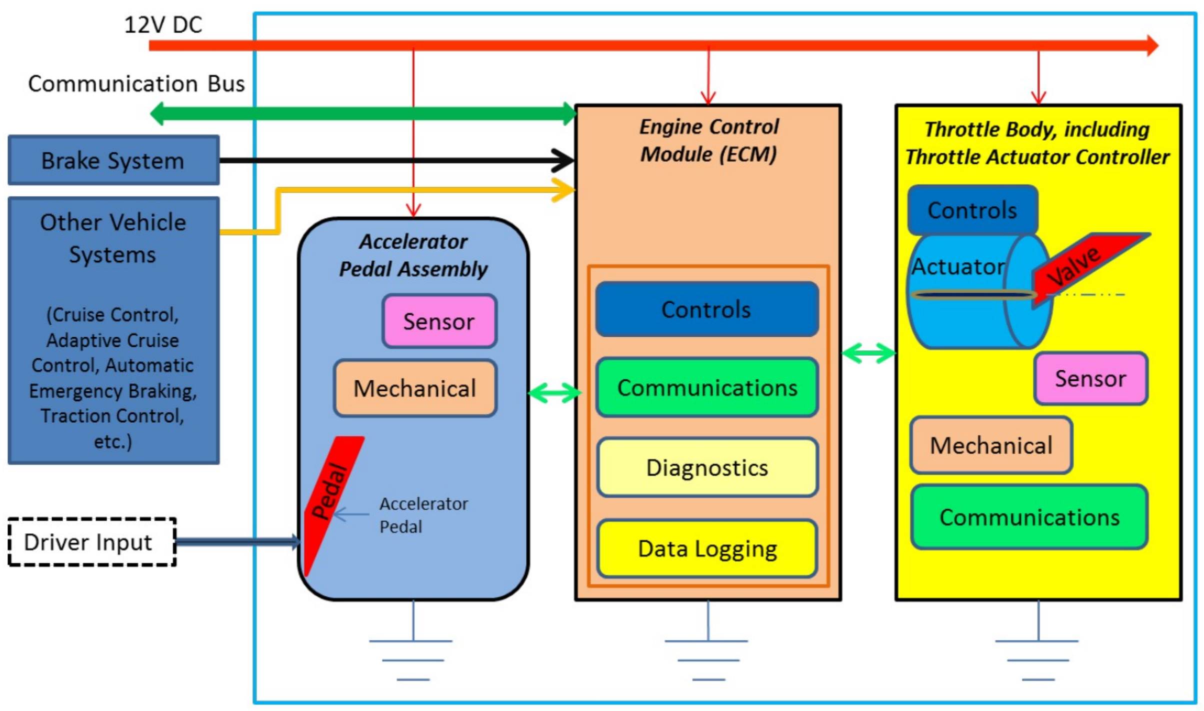

Driver
12V Power Supply (Ignition Key Position)
Apply/Release Accelerator Pedal
Accelerator Control System
Accelerator Pedal
Pedal Angular Displacement
Engine Control Module
Other Vehicle Sensors
(Engine Rotational Speed Sensor, Engine Camshaft Position Sensor, Mass Air Flow Sensor, Exhaust Gas Oxygen Sensor, etc.)
Measured Engine Speed
Measured Camshaft Speed
Measured Air Flow
Sensed Exhaust Oxygen Amount
Brake / Vehicle Stability System
Braking Command
Throttle Reduction Request
Vehicle Speed
Engine Torque Request
Engine Torque Request
Cruise Control System
Other Vehicle Systems
(Adaptive Cruise Control, Automatic Emergency Braking, etc.)
Throttle Motor
Actuation PWM signal
Torque
Throttle Valve
Valve Opening Angle
Measured Throttle Opening
Throttle Position Sensor
Fuel Supply System, Spark Ignition
Fuel Injector PWM, Spark Timing
Fuel Pump On/Off
Fuel Pressure
Air
Fuel and Spark
Engine
Actuator
Controller
Sensor
External Systems
Controlled Process
System Boundary

Figure 2. shows system overview of a typical throttle and engine control system. Focus of our example case study will be just acceleration pedal position encoder. The Acceleration Pedal (AP) assembly allows the driver to command a desired torque. When the driver presses the AP, the AP Position Sensor (APPS) measures the angular position of the AP. The APPS converts the angular position of the AP to a voltage signal, which is then transmitted to the Engine Control Module (ECM). The signal may be transmitted via a direct analog input or a vehicle communication bus such as the CAN bus. Using calibration maps, the ECM converts the voltage signal from the APPS to a desired engine torque. The ECM then reconciles the torque requested by the driver with requests from other vehicle systems and measurements from other vehicle sensors.

Acceleration control system (ACS) functional overview



1. Provide the APP to the ECM
2. Return the AP to the off (unpressed) position within the specified time
3. AP request rate limiting
4. Provide diagnostics
5. Provide fault detection and failure mitigation
6. Store relevant data

Faults

APPS communication fault – corrupted/missing data
APPS refresh rate fault, signal discontinued – watchdog fault
APPS Idle position fault
APP value interpretation fault
ACS calibration LUT corrupted
APP value change fault – change over time out of bounds (step change)
APP range fault – guard region (to detect open, short hardware faults)
ACS processing fault – control algorithm, data inconsistency

Example Safe States

A safe state may be the intended operating mode, a degraded operating mode, or a switched off mode (Part 1 Clause 1.102 of ISO 26262). The developer of the functional safety concept attempts to maximize the availability of the item while ensuring the safety of the vehicle operation. Therefore, a careful consideration must be given to selecting the safe states in relation to the potential failure modes.
The safe states for our ACS are either full operation (full torque availability), restricted operation (0 < Torque < Full), or switched off mode (zero torque). The restricted operation may include different levels depending on the potential failure mode. The safety analysis at the system level, the hardware level, and the software level may identify potential failure modes with the APPS, the ACS, the ECM, the throttle actuator controller, throttle body, and other interfacing systems. In cases where a good but not confirmed APPS signal is available, the safe state may allow full torque but at a ramp rate slower than normal to give the driver more reaction time in case of unintended vehicle behavior. In case the APPS signal is completely non- reliable, or if the ECM faults, but the vehicle can still be controlled by the brakes and the throttle actuator controller, the vehicle may be allowed a torque level higher than creep torque. In case of APPS, no more than creep torque may be allowed. If the failure mode may result in uncontrolled torque production, then the system torque should be disabled.

Following states commonly used in the automotive industry:

- Safe State 1: Disable input from other vehicle systems, such as ACC and AEB.
- Safe State 2: Limit the maximum allowable propulsion torque to the propulsion torque level that was computed at the instant immediately prior to when the fault occurred.
- Safe State 3: Slow torque ramp rate in response to AP input (e.g., single APPS fault)
- Safe State 4: Torque produced without AP input; speed limited to TBD (> creep) mph (e.g., two APPS faults; an ECM fault with throttle actuator controller still able to control throttle)
- Safe State 5: Torque produced at zero AP input value of the torque map (e.g., two APPS faults plus BPPS fault)
- Safe State 6: Zero torque output (e.g., vehicle disabled; system is unable to mitigate the hazards or ensure Safe States 1-5).

The safe states listed above describe propulsion reduction (Safe States 2, and 4-6) or deviations from the specified speed decrease or increase profiles (Safe State 3). While these vehicle responses may be similar to the identified hazards H2 through H4, there are key differences. • The propulsion reduction or modified speed decrease/increase profiles are controlled when entering a safe state, while the hazards describe uncontrolled changes in propulsion (e.g., changes are not smooth or consistent). When entering a safe state, the driver is informed that the vehicle is in a degraded operating state and can take appropriate action.

Example Driver Warning Strategies

The following is an example of driver warning strategies commonly seen in the automotive industry:

- Yellow Light: Potential violation of a safety goal is detected, but probability is moderate (e.g., single APPS fault)
- Red Light: Potential violation of a safety goal is detected; probability is high (e.g., AP Torque Map corruption, AP communication/data transfer fault),

- Chime: Audible notification of the driver is implemented whenever the conditions for the red-light driver warning are identified. The chime may continue until the fault is removed.
- Messages: Messages are displayed to the driver containing fault description and driver instructions
- Fatal fault: Fatal fault warning may be an additional driver warning strategy. Dashboard lights and audible chimes are commonly used in conjunction with fatal fault warning. It may be beneficial to assess the drivers' reactions to fatal fault warning when the system is at the same time attempting to reach safe state and degraded operation.

In case of a controller hardware fault, the first mitigation strategy is for the system to be able to detect the abnormality and transition the system to a safe state. This requirement corresponds to the safety strategy that involves detection, fault tolerance, and safe state. Additionally, if the vehicle is to transition to a safe state with reduced or very limited propulsion power (e.g., limp-home mode), the driver would need to be notified so that he/she can maneuver the vehicle to a safe location and get the needed repair service to the vehicle.

General ACS System-Level Functional Safety Requirements

General system-level functional safety requirements for the generic acceleration control system:

1. The ACS is to perform power-on tests, periodic tests, or continuous monitoring tests to ensure the correctness of safety-critical parameters and the integrity of critical system elements (ASIL C/D).
    - Critical parameters include calculation and comparison results that confirm the proper operation of the system.
    - The acceleration pedal position maps (LUT) are to be checked.
    - The communications channels between the APPS and the ECM are to be checked.
    - A confirmation of the sanity and health of the ACS is to be confirmed
    - Arbitration logic strategy and algorithm are to be checked for health and sanity periodically
    - Sanity checks may include quizzer, or seed-and-key strategies
    - State-of-health checks may include:
        - RAM/ ROM/ EEPROM tests,
        - Analog-to-digital converter test
        - Shutdown test.
    - The frequency of the periodic tests is to be selected based on the FTTI, and the fault reaction time interval.
    - In the event a failure is detected during the periodic self-tests, the ACS is to transition to the appropriate safe state.
2. Redundant elements are to be verified against common cause failures (ASIL C/D). Failures in one element are not to affect the other element. A failure in the communication path of one element is not to affect the communication path of the other element.
3. In case of a failure of one redundant element, the ACS is to transition into Safe State, driver warning is to be communicated to the driver (ASIL C/D).
4. If redundant elements are used and both elements fail, or if only one element is used and it fails, then the ACS is to transition into Safe State and a red-light driver warning is to be communicated to the driver (ASIL B/C/D).

5. The APP command is to be controlled and updated with the correct magnitude and within the correct time duration, also includes detecting erroneous APP commands issued by malicious intruders or other components (aftermarket)
6. The critical communications include the APP and the diagnostics of the APPS
7. Fault tolerant strategy is to be applied. The fault tolerant techniques may include redundancy, voting logic, or other techniques. Control flow monitoring strategy is to be applied also.
8. Diagnostics of all safety-critical component functions are to be conducted. In case of detected faults, the system is to take mitigation action to prevent failures that lead to a potential violation of a safety goal, and appropriate Diagnostic Trouble Codes (DTCs) are to be set (ASIL QM/A/B).
9. The ACS/ETC is to include diagnostics covering all operational and failure modes (ASIL QM/A/B).
10. DTCs are to be set every time a safety goal may be violated (ASIL QM).
11. Diagnostics covering the safety related functionality of the ACS/ETC system components and connections are to be instituted with a level of coverage corresponding to the ASIL of the safety goal that is affected. Adhere to ISO 26262 diagnostics coverage guidelines for low, medium, and high to comply with the hardware architectural metrics targets (ASIL A/B/QM).
12. Diagnostics covering the failures for the following parts of the system are to be implemented (ASIL QM/A/B):
    - Execution logic (wrong coding, wrong or no execution, execution out of order, execution too fast or too slow, and stack overflow or underflow)
    - On-chip communication and bus arbitration
    - The main controller's o central processing unit
        - processor memory
        - arithmetic logic unit
        - registers
        - A/D converter
        - software program execution
        - connections I/O faults (short/open/drift/oscillation)
        - power supply
        - temperature
    - Diagnostics information must contain:
        - the fault including the time at which the fault was detected and the nature of the fault
        - The time interval from the detection of the fault to reaching the safe state
        - The time the system degradation strategy started, including the start and end of each phase if applicable and the values of the system metrics for each phase
        - The time the driver warning strategy started, including the start and end of each phase if applicable and the values of the system metrics for each phase
    - The data is to be retained until accessed by authorized personnel
13. The ACS is to have diagnostics for potential safety relevant failures caused by EMI/ESD, contamination, organic growths, single event effects, and other environmental conditions (ASIL B/C/D).
14. The ECM is to have a mechanism to prevent unauthorized access to the ECM software.

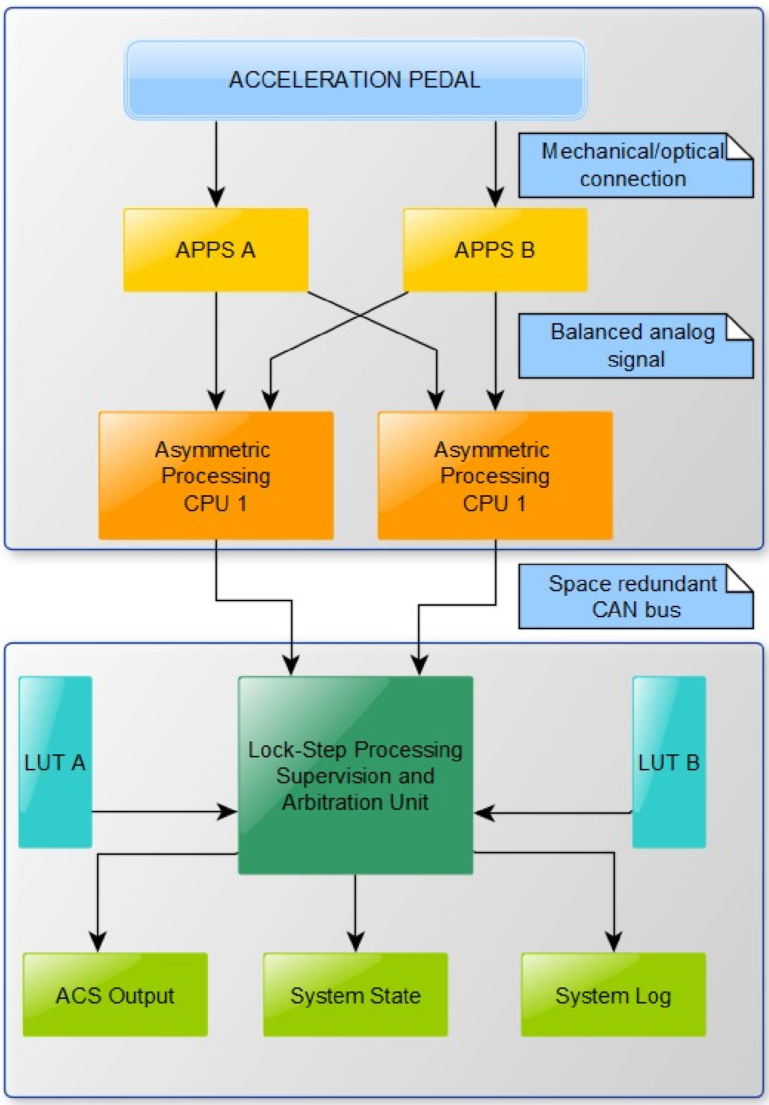
Accelerator Pedal Functional Safety Requirements

There are AP functional safety requirements derived for the generic example ACS/ETC:

1. The APP corresponding to the propulsion torque requested by the driver is to be mapped correctly and consistently, and the results are to be qualified for validity and correctness under all vehicle operating conditions, over the usable life of the vehicle (ASIL C/D).
2. The health and sanity of the APPS is to be monitored and confirmed under all operating vehicle conditions (ASIL C/D).
3. The APP value is to be measured, and the value is to be valid and correct (ASIL B/C/D).
4. The APP to electrical conversion method is to be validated (ASIL B/C/D).
5. The value of the APP is to be communicated to the ECM (ASIL B/C/D):
    - The communication message or data transfer is to be qualified for validity (sent and received signals are the same), correctness (within range), and rationality (does not contradict with previous or other related signals/messages).
    - The updated value of the APP is to be received within determined time interval. This time shall be specified to support the timely update of the throttle position to prevent the potential violation of any safety goal.
6. In case of a fault that violates a safety goal, the APPS is to communicate the fault to the ECM (ASIL B/C/D).
   Causes of faults may include:
    - Internal hardware failure
    - Degradation over time
    - Overheating due to increased resistance in a sub component or internal short
    - Reporting frequency too low
7. The APPS is to have diagnostics for safety-relevant failures that could be caused by electromagnetic interference/electrostatic discharge, contamination, and other environmental condition (ASIL A/B).
8. All single point APPS hardware faults that could lead to potential violation of a safety goal are to be detected and mitigated. In case of the detection of a failure, the system is to transition to the corresponding safe state.

Practical model of example APP controller

Goal of this hands on example will be to implement model and simulation of redundant dual processing system for safe acceleration pedal position encoder. Although lock step processing (same code runs on two synchronized CPUs – hardware faults mitigation) is most popular system architecture, in this example we will take more conservative approach with dual asymmetric processing and arbitration unit (supervisor) in lock step operation.


ACCELERATION PEDAL
Mechanical/optical connection
APPS A
APPS B
Balanced analog signal
Asymmetric Processing CPU 1
Asymmetric Processing CPU 1
Space redundant CAN bus
LUT A
Lock-Step Processing Supervision and Arbitration Unit
LUT B
ACS Output
System State
System Log

Our modeled system architecture is based on two virtual processors (2 parallel threads) running different code while performing same function (acceleration pedal position decoder) and arbitration unit, implemented also as virtual processor (third parallel thread).

Two communication channels (between each processor and arbitration unit) implemented as circular FIFO buffers are modeled.

Acceleration pedal position sensor (APPS) data is generated with matlab (octave) script and read as .csv input file.

File contains two data vectors containing 10bit ADC data for each complementary logarithmic position sensor.

For each sample one run cycle is executed in parallel for each processing unit. Task is to implement functional safety software for each processing unit. Single data point for every run cycle is provided to user program passed trough function interface (struct data pointer) called every run cycle.

Each processing unit process data for both sensor channels using different implementation and packs data to separate communication structures sent to arbitration unit.

Arbitration unit checks data integrity, compares results from both processing units, run system checks, decoding LUT tables integrity and provides acceleration position output data, switches to appropriate system state (or safe state) and writes system log.

Safe state definitions

- SafeState1 - warningOperational
- SafeState2 - warningRestrictive
- SafeState3 - errorSpeedLimit
- SafeState4 - fatalSafeStop

The acceleration pedal position maps (LUT) are to be checked:

- one map corrupted -> **SS1**
- all redundant maps corrupted -> **SS4**

The communications channels between the APPS and the ACS are to be checked

- packet lost/data corruption -> **SS1** --comWarningIntegrator (2,1,2)
- communication timeout -> **SS4** --timeout 2 communication intervals (2,1,4)

In case of a failure of one redundant element, the ACS is to transition into Safe State

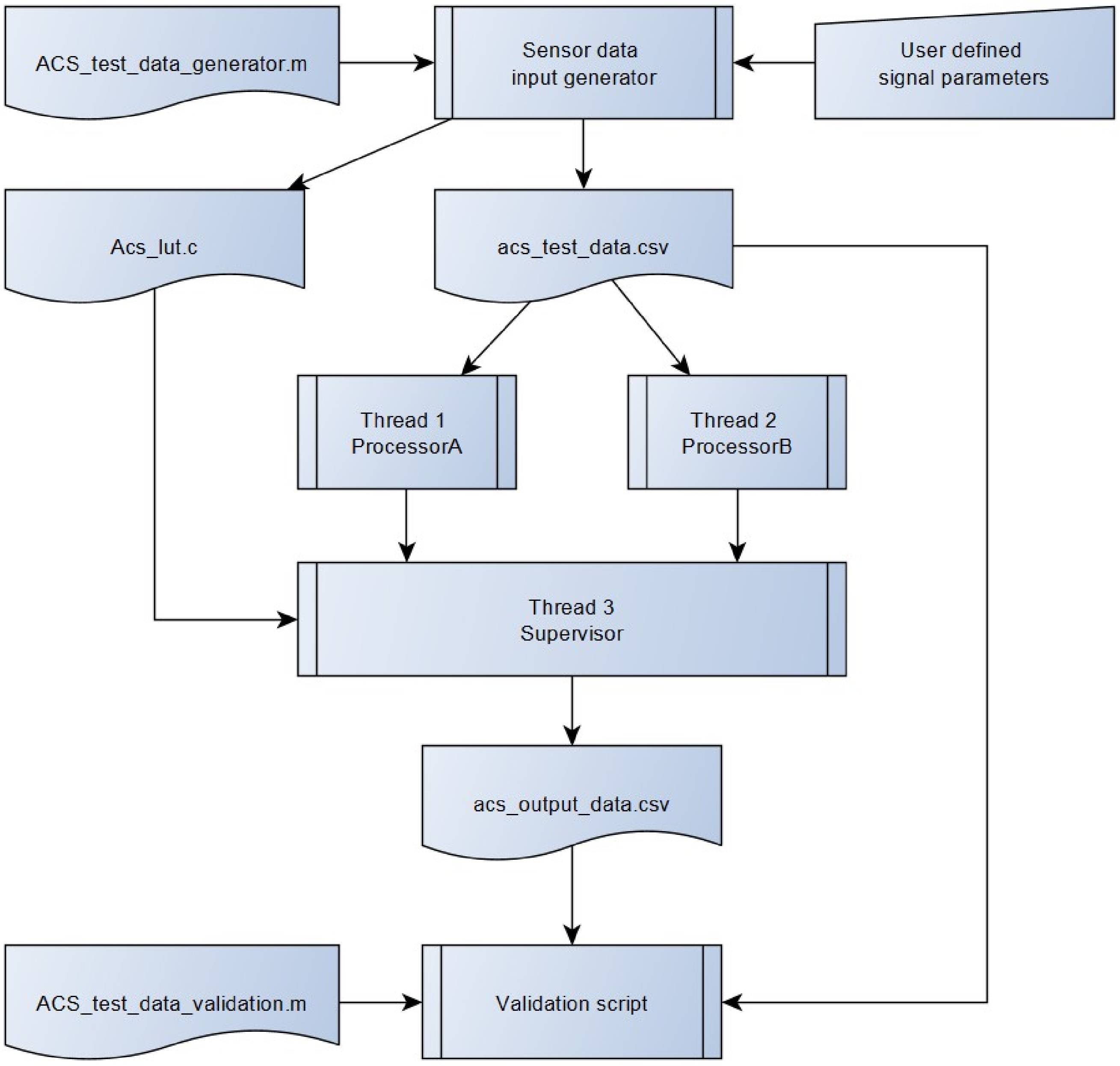
- determent fault -> **SS2** -> one position sensor out of range, invalid value change, single processing fault
- uncertainty fault -> **SS3** -> different parallel processing algorithm results

In case of a failure of one redundant element -> **SS4**

In case of a failure in this arbitration strategy, the ACS system is to
transition into Safe State -> **SS1**

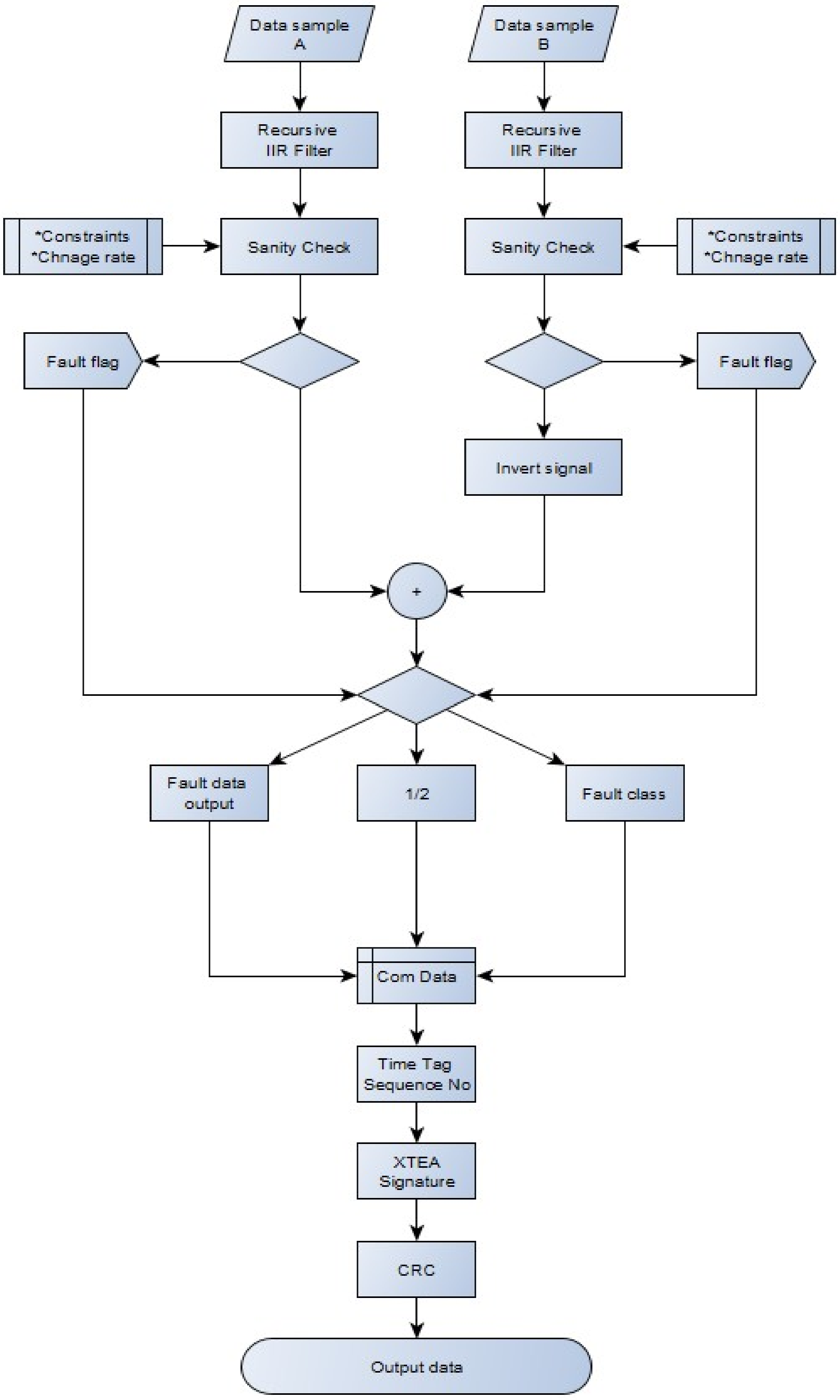
All faults that result in a failure to determine the pedal position are to be detected. In case of a failure in detecting the throttle position, the system is to transition into Safe -> **SS4**

Virtual ACS model architecture



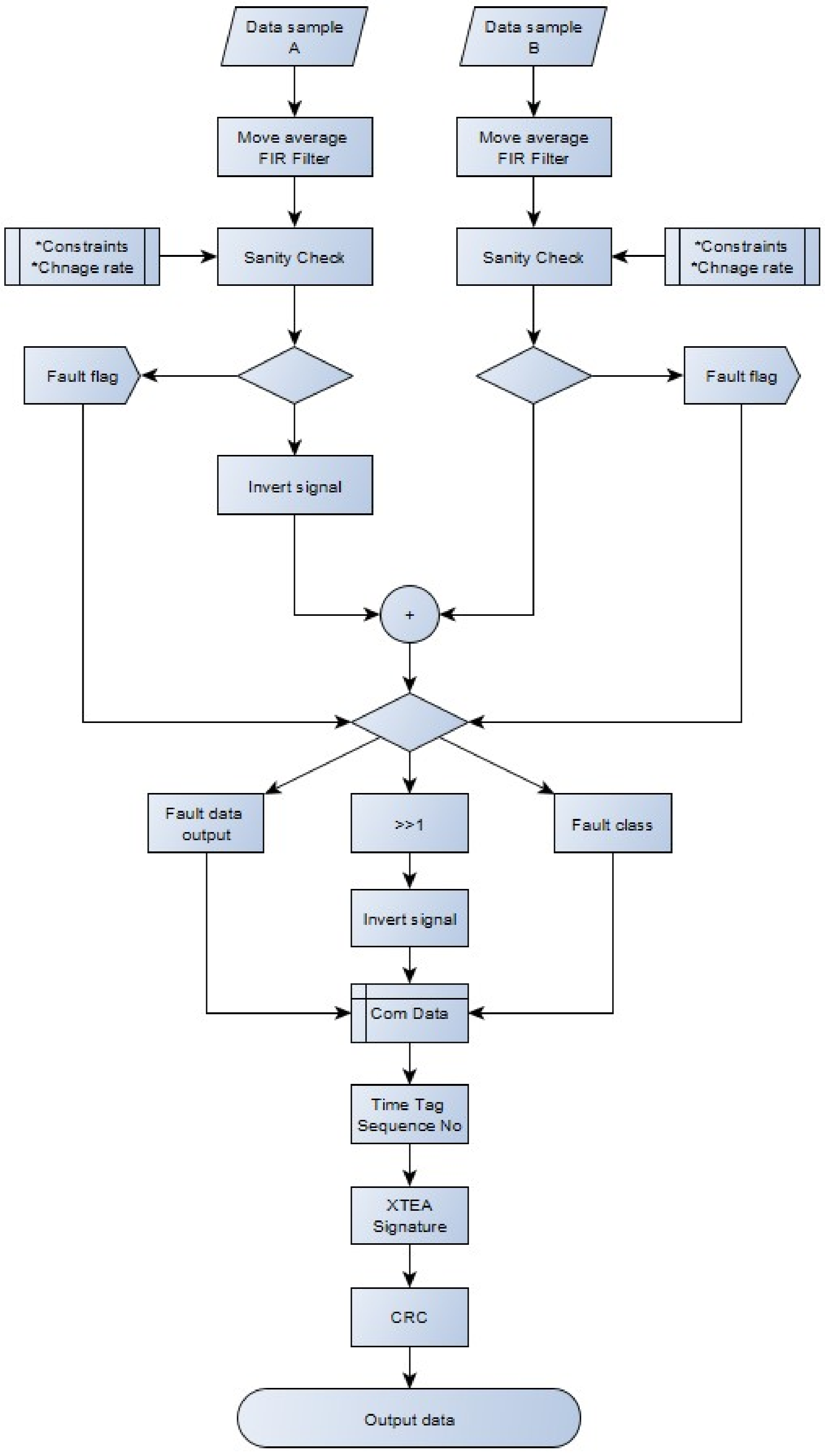
Processor A specification

```
void processorCodeA(input_data_pt inputData, com_data_pt outputData)
```



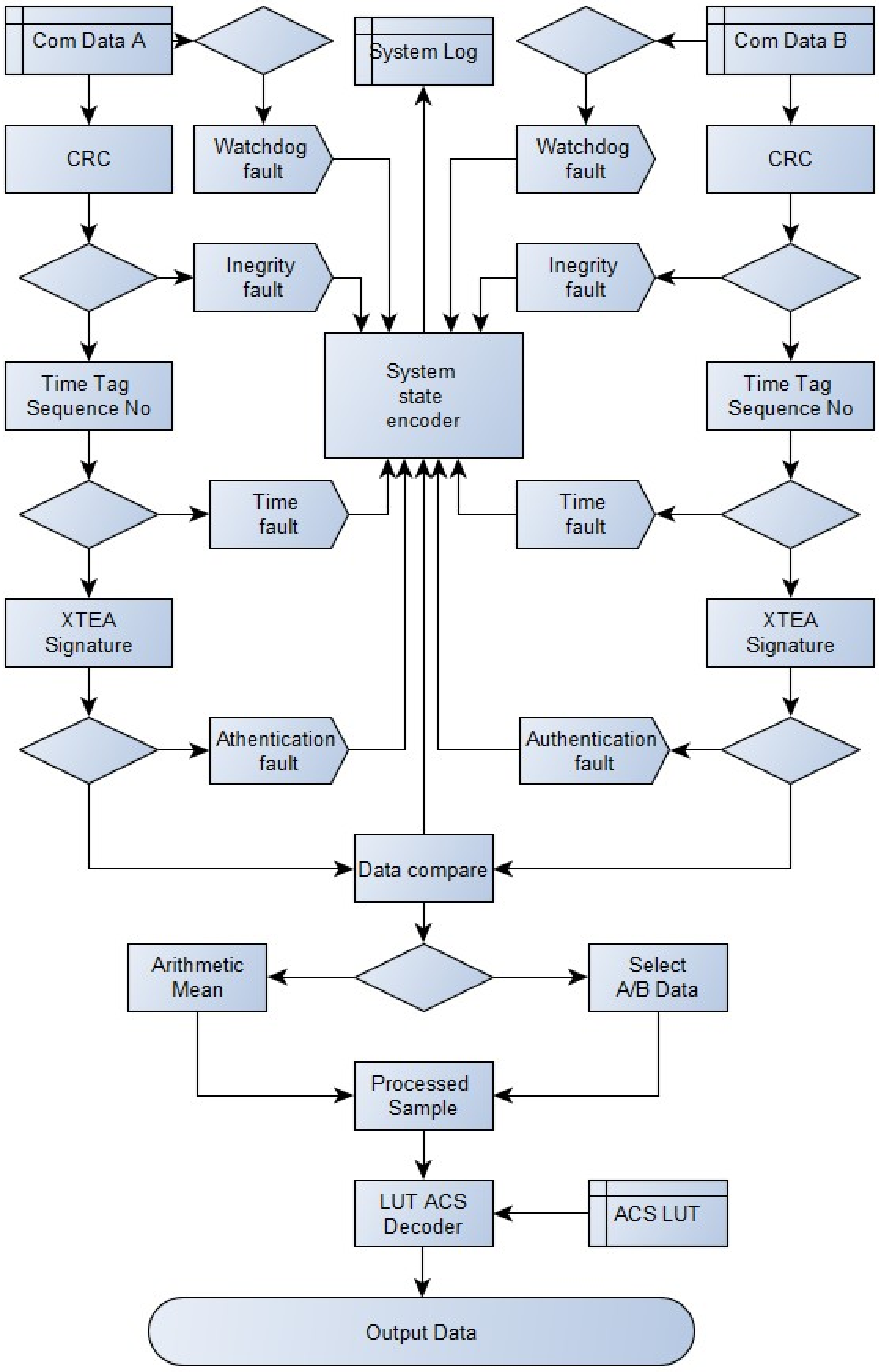
Processor B specification

```
void processorCodeB(input_data_pt inputData, com_data_pt outputData)
```



Arbitration algorithm specification

```
void supervisorCode(com_data_pt comDataProcA, com_data_pt comDataProcB, output_data_pt
outputData)
```



```
/******************************************************************************
**                              Data Structures
******************************************************************************/

typedef uint32_t sample_data_t;
typedef uint32_t samples_no_t;
typedef uint16_t acs_flags_t;
typedef uint32_t time_stamp_t;
typedef uint8_t sequence_no_t;
typedef uint16_t crc_t;

typedef enum
{
    ACS_IDLE,
    ACS_ACTIVE,
    ACS_WARNING_OPERATIONAL,
    ACS_WARNING_RESTRICTIVE,
    ACS_ERROR_SPEED_LIMIT,
    ACS_FATAL_SAFE_STOP
} acs_state_e;

typedef struct
{
    sample_data_t sensorSampleA;
    sample_data_t sensorSampleB;
    samples_no_t sampleNo;
} input_data_t;

typedef input_data_t* const input_data_pt;  /*const pointer to proces_data_t*/

typedef struct
{
    sample_data_t dataSample;
    time_stamp_t time;
    sequence_no_t seqNo;
    authentication_t signature[SIGNATURE_WORDS];
    crc_t crc;
} com_data_t;

typedef com_data_t* const com_data_pt;  /*const pointer to com_data_t*/

typedef struct
{
    sample_data_t output;
    acs_flags_t flags;
    acs_state_e state;
} output_data_t;

typedef output_data_t* const output_data_pt; /*const pointer to output_data_t*/
```