ACU V2.1 Firmware

2.1

Generated by Doxygen 1.14.0

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

Chapter 2

File Documentation

2.1 src/main.cpp File Reference

Main firmware file for the Actuation Control Unit (ACU) V2.1.

```
#include <Arduino.h>
#include "definitions.h"
#include "FlexCAN_T4_.h"
#include "IntervalTimer.h"
#include "autonomous_temporary.h"
```

Macros

- #define print_state 1
- #define PRESSURE_READINGS 8

Enumerations

- enum ACU_STATE_t {
  STATE_INIT , STATE_MISSION_SELECT , STATE_INITIAL_SEQUENCE , STATE_READY ,
  STATE_DRIVING , STATE_EBS_ERROR , STATE_EMERGENCY , STATE_FINISHED }

  Represents the various operational states of the ACU (Actuation Control Unit).
- enum AS_STATE_t {
  AS_STATE_OFF , AS_STATE_READY , AS_STATE_DRIVING , AS_STATE_EMERGENCY ,
  AS_STATE_FINISHED }

  Represents the different states of the Autonomous System (AS). This enumeration defines the various operational states of the AS, which can be used to manage the system's behavior during autonomous operations.
- enum current_mission_t {
  MANUAL , ACCELERATION , SKIDPAD , AUTOCROSS ,
  TRACKDRIVE , EBS_TEST , INSPECTION }
- enum INITIAL_SEQUENCE_STATE_t {
  WDT_TOOGLE_CHECK , WDT_STP_TOOGLE_CHECK , PNEUMATIC_CHECK , PRESSURE_CHECK1 ,
  IGNITON , PRESSURE_CHECK_FRONT , PRESSURE_CHECK_REAR , PRESSURE_CHECK2 ,
  ERROR }

Functions

- void UpdateState (void)

  Update the state of the ACU based on inputs and conditions.
- void HandleState (void)

  Handle actions specific to the current state.
- void print_state_transition (ACU_STATE_t from, ACU_STATE_t to)

  Print state transition for debugging.
- void canISR (const CAN_message_t &msg)

  CAN receive interrupt callback.
- void led_heartbit ()
- void peripheral_init ()

  Initialize peripherals and pins.
- void Pressure_readings ()

  Read pressure sensors and update system state.
- void send_can_msg ()

  Send CAN messages based on system state.
- void median_pressures ()
- void initial_sequence ()
- void check_ignition ()

  Debounces and checks the ignition input signal.
- void ASSI ()

  Controls the state of the YELLOW_LEDS and BLUE_LEDS based on the current as_state.
- void Mission_Indicator ()

  Updates the mission indicator LEDs based on the current mission state.
- void setup ()
- void loop ()

Variables

- const char * state_names [ ]
- volatile ACU_STATE_t current_state = STATE_INIT
- volatile ACU_STATE_t previous_state = STATE_INIT
- volatile AS_STATE_t as_state = AS_STATE_OFF
- INITIAL_SEQUENCE_STATE_t initial_sequence_state = IGNITON
- current_mission_t current_mission = MANUAL
- current_mission_t jetson_mission = MANUAL
- IntervalTimer PRESSURE_TIMER
- IntervalTimer CAN_TIMER
- FlexCAN_T4< CAN2, RX_SIZE_1024, TX_SIZE_1024 > CAN
- unsigned long HeartBit = 0
- float EBS_TANK_PRESSURE_A_values [PRESSURE_READINGS]

  Array storing recent pressure readings from EBS Tank A.
- float EBS_TANK_PRESSURE_B_values [PRESSURE_READINGS]

  Array storing recent pressure readings from EBS Tank B.
- float TANK_PRESSURE_FRONT = 0

  Pressure value for front tank (in bar).
- float TANK_PRESSURE_REAR = 0

  Pressure value for rear tank (in bar).
- float HYDRAULIC_PRESSURE_FRONT = 0

  Hydraulic pressure value for front brakes (in bar).
- float HYDRAULIC_PRESSURE_REAR = 0

Hydraulic pressure value for rear brakes (in bar).

- uint8_t adc_pointer = 0

  Pointer for pressure readings buffer.
- volatile bool update_median_flag = false

  Flag to indicate if median pressure update is needed.
- uint8_t ignition_flag = 0

  Flag to indicate ignition signal.
- uint8_t ignition_vcu = 0

  Ignition signal state from VCU.
- uint8_t asms_flag = 0

  Current ASMS (Autonomous System Master Switch) signal state.
- uint8_t emergency_flag = 0

  Flag to indicate emergency state.
- volatile uint8_t res_emergency = 0

  Emergency response from AS (Autonomous System).
- volatile bool wdt_togle_enable = true

  Flag to enable WDT (Watchdog Timer) toggle.
- unsigned long wdt_togle_counter = 0

  Counter for WDT toggle timing.
- unsigned long wdt_relay_timout = 0

  Timeout for WDT relay check.
- unsigned long pressure_check_delay = 0

  Delay for pressure check timing.
- uint8_t ignition_enable = 0

  Flag to enable ignition logic.
- volatile bool res_active = false

  Flag to indicate if response from AS is active.
- unsigned long emergency_timestamp = 0

  Timestamp for entering emergency state.
- unsigned long ASSI_YELLOW_time = 0

  Timestamp for ASSI yellow LED blinking.
- unsigned long ASSI_BLUE_time = 0

  Timestamp for ASSI blue LED blinking.
- unsigned long last_button_time_ms = 0

  Last button press time in milliseconds.

2.1.1 Detailed Description

Main firmware file for the Actuation Control Unit (ACU) V2.1.

This file implements the main logic, state machine, and peripheral handling for the ACU, which manages the actuation and safety logic for an autonomous vehicle. The ACU interfaces with pressure sensors, solenoids, CAN bus, and various status indicators.

Key features:

- Implements a robust state machine for ACU operation, including initialization, mission selection, initial safety checks, ready, driving, emergency, and finished states.
- Handles CAN communication for receiving commands and sending status updates.

- Reads and processes pressure sensor data with averaging and conversion to engineering units.
- Manages ignition and emergency logic with debounce and safety checks.
- Controls visual indicators (LEDs) for system and mission status.
- Provides detailed documentation for each function and state.

Note

Key global variables used throughout this file include:

- current_state, previous_state: ACU state machine tracking.
- as_state: Autonomous system state.
- initial_sequence_state: State for initial safety sequence.
- current_mission, jetson_mission: Mission selection tracking.
- EBS_TANK_PRESSURE_A_values, EBS_TANK_PRESSURE_B_values: Pressure sensor readings.
- TANK_PRESSURE_FRONT, TANK_PRESSURE_REAR: Calculated tank pressures.
- HYDRAULIC_PRESSURE_FRONT, HYDRAULIC_PRESSURE_REAR: Hydraulic pressures.
- ignition_flag, ignition_vcu, ignition_enable: Ignition logic.
- asms_flag: Autonomous system master switch state.
- emergency_flag, res_emergency, res_active: Emergency logic.
- wdt_togle_enable, wdt_togle_counter, wdt_relay_timout: Watchdog timer logic.
- pressure_check_delay: Timing for pressure checks.
- emergency_timestamp: Timing for emergency state.
- ASSI_YELLOW_time, ASSI_BLUE_time: Timing for LED indicators.
- last_button_time_ms: Debounce for mission selection button.
- last_ign_state, debounced_ign_state: Debounce for ignition input.

Author

(Bruno Vicente - LART)

Date

(2025)

2.1.2 Macro Definition Documentation

2.1.2.1 PRESSURE_READINGS

```
#define PRESSURE_READINGS 8
```

2.1.2.2 print_state

```
#define print_state 1
```

2.1.3 Enumeration Type Documentation

2.1.3.1 ACU_STATE_t

```
enum ACU_STATE_t
```

Represents the various operational states of the ACU (Actuation Control Unit).

This enumeration defines the possible states in which the ACU can exist during its lifecycle. Each state corresponds to a specific phase or condition of the system.

Enumerator

STATE_INIT	The initial state after power-up or reset, where system initialization occurs.
STATE_MISSION_SELECT	State where the mission or operational mode is selected.
STATE_INITIAL_SEQUENCE	State for executing the initial sequence before becoming ready.
STATE_READY	System is ready and awaiting further commands or actions.
STATE_DRIVING	The ACU is actively controlling the vehicle or system in its driving mode.
STATE_EBS_ERROR	An error has occurred in the Emergency Braking System (EBS).
STATE_EMERGENCY	The system has entered an emergency state, requiring immediate attention.
STATE_FINISHED	The mission or operation has completed, and the system is in a finished state.

2.1.3.2 AS_STATE_t

`enum AS_STATE_t`

Represents the different states of the Autonomous System (AS). This enumeration defines the various operational states of the AS, which can be used to manage the system's behavior during autonomous operations.

Enumerator

AS_STATE_OFF	ASSI OFF
AS_STATE_READY	ASSI yellow
AS_STATE_DRIVING	ASSI Blinking yellow
AS_STATE_EMERGENCY	ASSI blinking Blue
AS_STATE_FINISHED	ASSI Blue

2.1.3.3 current_mission_t

`enum current_mission_t`

Enumerator

MANUAL	
ACCELERATION	
SKIDPAD	
AUTOCROSS	
TRACKDRIVE	
EBS_TEST	
INSPECTION	

2.1.3.4 INITIAL_SEQUENCE_STATE_t

`enum INITIAL_SEQUENCE_STATE_t`

Enumerator

WDT_TOOGLE_CHECK	
WDT_STP_TOOGLE_CHECK	
PNEUMATIC_CHECK	
PRESSURE_CHECK1	
IGNITON	
PRESSURE_CHECK_FRONT	
PRESSURE_CHECK_REAR	
PRESSURE_CHECK2	
ERROR	

2.1.4 Function Documentation

2.1.4.1 ASSI()

```
void ASSI ()
```

Controls the state of the YELLOW_LEDS and BLUE_LEDS based on the current as_state.

This function manages the visual indication of the system's state by toggling or setting the YELLOW_LEDS and BLUE_LEDS according to the value of the as_state variable. The behavior for each state is as follows:

- AS_STATE_OFF: Turns off both YELLOW_LEDS and BLUE_LEDS.
- AS_STATE_READY: Turns on YELLOW_LEDS and turns off BLUE_LEDS.
- AS_STATE_DRIVING: Toggles YELLOW_LEDS every 500 ms, keeps BLUE_LEDS off.
- AS_STATE_EMERGENCY: Toggles BLUE_LEDS every 500 ms, keeps YELLOW_LEDS off.
- AS_STATE_FINISHED: Turns off YELLOW_LEDS and turns on BLUE_LEDS.
- Default: Turns off both YELLOW_LEDS and BLUE_LEDS.

Timing for toggling is managed using ASSI_YELLOW_time and ASSI_BLUE_time variables.

Note

This function assumes that as_state, ASSI_YELLOW_time, and ASSI_BLUE_time are defined and accessible in the current scope, and that digitalWrite, digitalRead, and millis functions are available (e.g., in an Arduino environment).

2.1.4.2 canISR()

```
void canISR (
            const CAN_message_t & msg)
```

CAN receive interrupt callback.

When a can message is received, this function is called to process the message. It decodes the message based on its ID and updates the system state accordingly.

Note

This function is called from the FlexCAN_T4 library's interrupt handler.

Parameters

msg	The received CAN message

2.1.4.3 check_ignition()

```
void check_ignition ()
```

Debounces and checks the ignition input signal.

This function reads the current state of the ignition pin (IGN_PIN) and applies a debounce algorithm to filter out spurious changes due to mechanical switch noise. It updates the ignition_flag based on the debounced state and the ignition_enable flag.

Note

Variables used:

- last_debounce_time (static): Stores the last time the ignition input changed, used for debouncing.
- debounce_delay (const): The debounce interval in milliseconds.
- current_state: The current raw reading from the ignition pin.
- last_ign_state (external): The last raw state read from the ignition pin.
- debounced_ign_state (external): The last debounced state of the ignition pin.
- ignition_flag (external): Set to 1 if ignition is ON and enabled, otherwise 0.
- ignition_enable (external): Enables or disables the ignition logic.

The function ensures that ignition_flag is set only if the ignition input is HIGH and ignition_enable is true, providing reliable ignition state detection.

2.1.4.4 HandleState()

```
void HandleState (
            void )
```

Handle actions specific to the current state.

2.1.4.5 initial_sequence()

```
void initial_sequence ()
```

2.1.4.6 led_heartbit()

```
void led_heartbit ()
```

2.1.4.7 loop()

```
void loop ()
```

2.1.4.8 median_pressures()

```
void median_pressures ()
```

2.1.4.9 Mission_Indicator()

```
void Mission_Indicator ()
```

Updates the mission indicator LEDs based on the current mission state.

This function sets the state of each mission status LED (MS_LED1 to MS_LED7) to indicate the currently active mission. Each mission mode corresponds to a unique LED pattern, where one LED is turned on to represent the active mission, and the others are turned off (logic HIGH). If the mission state is not recognized, all LEDs are turned off by default.

2.1.4.10 peripheral_init()

```
void peripheral_init ()
```

Initialize peripherals and pins.

2.1.4.11 Pressure_readings()

```
void Pressure_readings ()
```

Read pressure sensors and update system state.

2.1.4.12 print_state_transition()

```
void print_state_transition (
            ACU_STATE_t from,
            ACU_STATE_t to)
```

Print state transition for debugging.

Parameters

from	Previous state
to	New state

2.1.4.13 send_can_msg()

```
void send_can_msg ()
```

Send CAN messages based on system state.

2.1.4.14 setup()

```
void setup ()
```

2.1.4.15 UpdateState()

```
void UpdateState (
            void )
```

Update the state of the ACU based on inputs and conditions.

2.1.5 Variable Documentation

2.1.5.1 adc_pointer

```
uint8_t adc_pointer = 0
```

Pointer for pressure readings buffer.

Used in Pressure_readings() to cycle through EBS_TANK_PRESSURE_A_values and EBS_TANK_PRESSURE↩ _B_values.

2.1.5.2 as_state

```
volatile AS_STATE_t as_state = AS_STATE_OFF
```

2.1.5.3 asms_flag

```
uint8_t asms_flag = 0
```

Current ASMS (Autonomous System Master Switch) signal state.

Updated in loop() from digitalRead(ASMS), used in send_can_msg().

2.1.5.4 ASSI_BLUE_time

```
unsigned long ASSI_BLUE_time = 0
```

Timestamp for ASSI blue LED blinking.

Used in ASSI() for timing blue LED blinking in AS_STATE_EMERGENCY.

2.1.5.5 ASSI_YELLOW_time

```
unsigned long ASSI_YELLOW_time = 0
```

Timestamp for ASSI yellow LED blinking.

Used in ASSI() for timing yellow LED blinking in AS_STATE_DRIVING.

2.1.5.6 CAN

```
FlexCAN_T4<CAN2, RX_SIZE_1024, TX_SIZE_1024> CAN
```

2.1.5.7 CAN_TIMER

```
IntervalTimer CAN_TIMER
```

2.1.5.8 current_mission

```
current_mission_t current_mission = MANUAL
```

2.1.5.9 current_state

```
volatile ACU_STATE_t current_state = STATE_INIT
```

2.1.5.10 EBS_TANK_PRESSURE_A_values

```
float EBS_TANK_PRESSURE_A_values[PRESSURE_READINGS]
```

Array storing recent pressure readings from EBS Tank A.

This array holds the last PRESSURE_READINGS number of float values, representing the sampled pressure values from the EBS (Emergency Braking System) Tank A. Updated in Pressure_readings(), used in median_pressures() for filtering/averaging.

See also

EBS_TANK_PRESSURE_B_values

PRESSURE_READINGS

2.1.5.11 EBS_TANK_PRESSURE_B_values

```
float EBS_TANK_PRESSURE_B_values[PRESSURE_READINGS]
```

Array storing recent pressure readings from EBS Tank B.

This array holds the last PRESSURE_READINGS number of float values, representing the sampled pressure values from the EBS (Emergency Braking System) Tank B. Updated in Pressure_readings(), used in median_pressures() for filtering/averaging.

See also

EBS_TANK_PRESSURE_A_values

PRESSURE_READINGS

2.1.5.12 emergency_flag

```
uint8_t emergency_flag = 0
```

Flag to indicate emergency state.

Set in HandleState() and UpdateState(), used in send_can_msg().

2.1.5.13 emergency_timestamp

```
unsigned long emergency_timestamp = 0
```

Timestamp for entering emergency state.

Set in UpdateState() when entering STATE_EMERGENCY, used in HandleState() for timeout.

2.1.5.14 HeartBit

```
unsigned long HeartBit = 0
```

2.1.5.15 HYDRAULIC_PRESSURE_FRONT

```
float HYDRAULIC_PRESSURE_FRONT = 0
```

Hydraulic pressure value for front brakes (in bar).

Updated in canISR() from CAN message AUTONOMOUS_TEMPORARY_VCU_HV_FRAME_ID. Used in initial_sequence() for pressure checks.

2.1.5.16 HYDRAULIC_PRESSURE_REAR

```
float HYDRAULIC_PRESSURE_REAR = 0
```

Hydraulic pressure value for rear brakes (in bar).

Updated in canISR() from CAN message AUTONOMOUS_TEMPORARY_VCU_HV_FRAME_ID. Used in initial_sequence() for pressure checks.

2.1.5.17 ignition_enable

```
uint8_t ignition_enable = 0
```

Flag to enable ignition logic.

Set in initial_sequence(), used in check_ignition().

2.1.5.18 ignition_flag

```
uint8_t ignition_flag = 0
```

Flag to indicate ignition signal.

Updated in check_ignition(), used in send_can_msg(), HandleState(), and initial_sequence().

2.1.5.19 ignition_vcu

```
uint8_t ignition_vcu = 0
```

Ignition signal state from VCU.

Updated in canISR() from CAN message AUTONOMOUS_TEMPORARY_VCU_HV_FRAME_ID. Used in initial_sequence().

2.1.5.20 initial_sequence_state

```
INITIAL_SEQUENCE_STATE_t initial_sequence_state = IGNITON
```

2.1.5.21 jetson_mission

```
current_mission_t jetson_mission = MANUAL
```

2.1.5.22 last_button_time_ms

```
unsigned long last_button_time_ms = 0
```

Last button press time in milliseconds.

Used in HandleState() for mission selection button debounce.

2.1.5.23 pressure_check_delay

```
unsigned long pressure_check_delay = 0
```

Delay for pressure check timing.

Used in initial_sequence() to measure elapsed time for pressure checks.

2.1.5.24 PRESSURE_TIMER

```
IntervalTimer PRESSURE_TIMER
```

2.1.5.25 previous_state

```
volatile ACU_STATE_t previous_state = STATE_INIT
```

2.1.5.26 res_active

```
volatile bool res_active = false
```

Flag to indicate if response from AS is active.

Updated in canISR(), used in HandleState() and mission selection logic.

2.1.5.27 res_emergency

```
volatile uint8_t res_emergency = 0
```

Emergency response from AS (Autonomous System).

Updated in canISR() from CAN message AUTONOMOUS_TEMPORARY_RES_FRAME_ID. Used in HandleState() and UpdateState().

2.1.5.28 state_names

```
const char* state_names[]
```

Initial value:

```
= {
    "STATE_INIT",
    "STATE_Mission_Select",
    "STATE_INITIAL_SEQUENCE",
    "STATE_READY",
    "STATE_DRIVING",
    "STATE_EBS_ERROR",
    "STATE_EMERGENCY",
    "STATE_FINISHED"}
```

2.1.5.29 TANK_PRESSURE_FRONT

```
float TANK_PRESSURE_FRONT = 0
```

Pressure value for front tank (in bar).

Calculated in median_pressures() from EBS_TANK_PRESSURE_B_values. Used in initial_sequence(), HandleState(), send_can_msg(), and for state transitions.

2.1.5.30 TANK_PRESSURE_REAR

```
float TANK_PRESSURE_REAR = 0
```

Pressure value for rear tank (in bar).

Calculated in median_pressures() from EBS_TANK_PRESSURE_A_values. Used in initial_sequence(), HandleState(), send_can_msg(), and for state transitions.

2.1.5.31 update_median_flag

```
volatile bool update_median_flag = false
```

Flag to indicate if median pressure update is needed.

Set in Pressure_readings(), checked in loop() to call median_pressures().

2.1.5.32 wdt_relay_timout

```
unsigned long wdt_relay_timout = 0
```

Timeout for WDT relay check.

Used in initial_sequence() for timing WDT relay state.

2.1.5.33 wdt_togle_counter

```
unsigned long wdt_togle_counter = 0
```

Counter for WDT toggle timing.

Used in loop() and initial_sequence() to measure elapsed time for WDT toggling.

2.1.5.34 wdt_togle_enable

```
volatile bool wdt_togle_enable = true
```

Flag to enable WDT (Watchdog Timer) toggle.

Used in loop() and initial_sequence() to control WDT toggling.

Index