Heraclitus Rocket Controller 1.0 Documentation

Spring 2024

Responsible Engineers: Analicia Sosa, Bonnie White
Developers: Mark Carson, Patrick Clarke, Brayden Kwe, TJ Malaska, Hunter Ruebsamen, Carla Zuccarini
https://github.com/Beach-Launch-Team-CSULB/HeraclitusRocketController

Table of Contents

Peripherals

UML Diagram

Bonnie white

Classes

Rocket

sensorRead

Inputs: sensorId, int - The sensor ID number
Outputs: float
Behavior: Takes in a sensor ID and returns the corresponding sensors current value as a float

ignitionRead

Inputs: igniterID, int - ID of igniter
Outputs: bool
Behavior: Returns if the igniter is on or off

valveRead

Inputs: valveID, int - ID of valve
Outputs: bool
Behavior: Returns if the entered valve is open or close

setIgnitionOn

Inputs: igniterID, int - ID of igniter
Inputs: ignitionOn, bool - true for set on and false for set off
Outputs: bool
Behavior: sets the igniter on or off

setValveOn

Inputs: valveID, int - ID number of the valve
Inputs: valveOpen, bool - true for set on and false for set off
Outputs: bool
Behavior: open or closes valve

changeState

Inputs: state, int - it is the state of the rocket
Outputs: bool
Behavior: finds and changes the state of the rocket valves and igniters

getExecuting (WIP)

Inputs:
Outputs:
Behavior:

getState

Inputs: None
Outputs: int
Behavior: returns the current state of the rocket

initializeIgniters

Inputs: None
Outputs: bool
Behavior: initializes both igniters to the igniter map then returns true

initializeUpperValves

Inputs: None
Outputs: bool
Behavior: Initializes upper valves (high press, high vent, fuel main, lox Main) to the valve map. Returns True

initializeLowerValves

Inputs: None
Outputs: bool
Behavior: Initializes lower valves (Lox vent, Lox Dome Vent, Lox Dome Reg, Fuel Vent, Fuel Dome Vent, Fuel Dome Reg) to the valve map.Returns True

initializeUpperSensors

Inputs: None
Outputs: bool
Behavior: Initializes upper sensors(LOX High, Fuel High, LOX Dome, Fuel Dome, Lox Tank, Fuel Tank 1, Fuel Tank 2).Returns True

initializeLowerSensors

Inputs: None
Outputs: bool
Behavior: Initializes lower sensors (Pneumatics sensor, Lox inlet sensor, Fuel inlet sensor, Fuel injector sensor, Chamber 1, Chamber 2) to the sensor map.Returns True

calibrateSensors

Inputs: node, int - represents the wanted sensor calibration
Outputs: void
Behavior: depending on the value of the node it sets the sensors to different calibration parameters

Valve

getID

Inputs: None
Outputs: int
Behavior: Returns the ID of the valve

getPinPWM

Inputs: None
Outputs: int
Behavior: Returns pulse width modulation of the valve

getPinDigital

Inputs: None
Outputs: int
Behavior: Returns the digital pin of the valve

getValveOpen

Inputs: None
Outputs: bool

Behavior: Returns true if the valve is open and false if it is closed

setPinDigital

Inputs: newPinDigital, int - new digital pin number of the valve
Outputs: bool
Behavior: sets valve pin to the newPinDigital. Returns true if successful

setValveOpen

Inputs: ValveOpenInput, bool - True if valve is open false if not
Outputs: bool
Behavior: sets the value open or off depending on the input. Returns true

Igniter

getID

Inputs: None
Outputs: int
Behavior: Returns the IDof the igniter

getPinPWM

Inputs: None
Outputs: int
Behavior: Returns pulse width modulation of the igniter

getIgniterOn

Inputs: None
Outputs: bool
Behavior: Returns true if the igniter is on and false if it is not

setPinDigital

Inputs: newPinDigital, int - new digital pin number of the igniter
Outputs: bool
Behavior: sets valve pin to the newPinDigital. Returns true if successful

setIgniterOn

Inputs: isignitionOn, bool - True if ignition is on and false if not
Outputs: bool
Behavior: Sets the state of Igniter Objects to True if Open and Returns true if successful

Sensor

getLastValue

Inputs: None
Outputs: float
Behavior: Returns the last value taken by the sensor

readDataRaw

Inputs: None
Outputs: float
Behavior: Gets integer between corresponding to the sensor's current analog voltage

updateValue

Inputs: None
Outputs: void
Behavior: Updates the sensor value with the new data

getCurrentValue

Inputs: None

Outputs: float
Behavior: Updated the sensor's value and returns the new value

resetCalibration

Inputs: None
Outputs: void
Behavior: Sets the linear coefficients to their default values for calibrating

setCalibrationParameters

Inputs: linCoM, float -
Inputs: linCoB, float -
Outputs: void
Behavior: Sets the linear coefficients to new calibration values

getCalibrationSlope

Inputs: None
Outputs: float
Behavior: Gets the number for the linear slope

getCalibrationIntercept

Inputs: None
Outputs: float
Behavior: Gets the linear intercept

hasID

Inputs: id, int - id number of the sensor
Outputs: float
Behavior: Optional function demonstrating how bitmasks can be used

LoadCell

Moved to trauma response

Thermocouple

Moved to trauma response

PressureTransducer

getUnitEnum

Inputs: None
Outputs: Work In Progress
Behavior: Gets the output unit enum

changeOutputUnit

Inputs: newUnitSystem, Work in Progress - Work In Progress
Outputs: void
Behavior: Work In Progress

ExtendedIO

extendedIOsetup

Inputs: None
Outputs: Void
Behavior:

digitalPinToBit_int

Inputs: pin, int -
Outputs: int

Behavior:

digitalPinToPort_int

Inputs: pin, int -
Outputs: int
Behavior:

fetchRegister

Inputs: pin, int -
Inputs: RegisteName, int -
Inputs: reg, int -
Outputs: int
Behavior:

pinModeExtended

Inputs: pin, int -
Inputs: isGPIO, int -
Inputs: dataDirection, int -
Outputs: Void
Behavior:

digitalWriteExtended

Inputs: pin, int -
Inputs: value, int -
Outputs: Void
Behavior:

ALARA Setup

Engine Node (Lower, 0)

Igniters

Igniter 1 — software designation: IGN1 — hardware designation: ENG-IGNA
Igniter 2 — software designation: IGN2 — hardware designation: ENG_IGNB

Valves

High Press Valve — software designation: HP — hardware designation: HV HI PRES
High Vent Valve — software designation: HV — hardware designation: SV HI PRES V
Fuel Main Valve — software designation: FMV — hardware designation: SM MV FUEL
Lox Main Valve — software designation: LMV — hardware designation: SV MV LOX

Sensors

PT_PNEUMATICS — Pressure on the pneumatics system to control the ball valves
PT_LOX_INLET — Lox inlet pressure
PT_FUEL_INLET — Fuel inlet pressure
PT_FUEL_INJECTOR — Injector plate pressure
PT_CHAMBER_1 — Engine chamber pressure
PT_CHAMBER_2 — Engine chamber pressure

Prolulsion Node (Upper, 1)

Igniters

None

Valves

Lox Vent Valve — software designation: LV — hardware designation: SV LOX V
Lox Dome Vent Valve — software designation: LDV — hardware designation: SV DREG L
Lox Dome Reg Valve — software designation: LDR — hardware designation: SV DREG LV
Fuel Vent Valve — software designation: FV — hardware designation: SV FUEL V
Fuel Dome Vent Valve — software designation: FDV — hardware designation: SV DREG F V
Fuel Dome Reg Valve — software designation: FDR — hardware designation: SV DREG F

Sensors

PT_LOX_HIGH — High pressure section, no difference between lox and fuel high pressure sections
PT_FUEL_HIGH — High pressure section, no difference between lox and fuel high pressure sections
PT_LOX_DOME — Lox dome reg pressure
PT_FUEL_DOME — Fuel dome reg pressure
PT_LOX_TANK_1 — Lox tank pressure, theoretically the same as lox tank 2
PT_LOX_TANK_2 — Lox tank pressure, theoretically the same as lox tank 1
PT_FUEL_TANK_1 — Fuel tank pressure, theoretically the same as fuel tank 2
PT_FUEL_TANK_2 — Fuel tank pressure, theoretically the same as fuel tank 1

Operation

Sequence Diagram

Command Validation

Test State

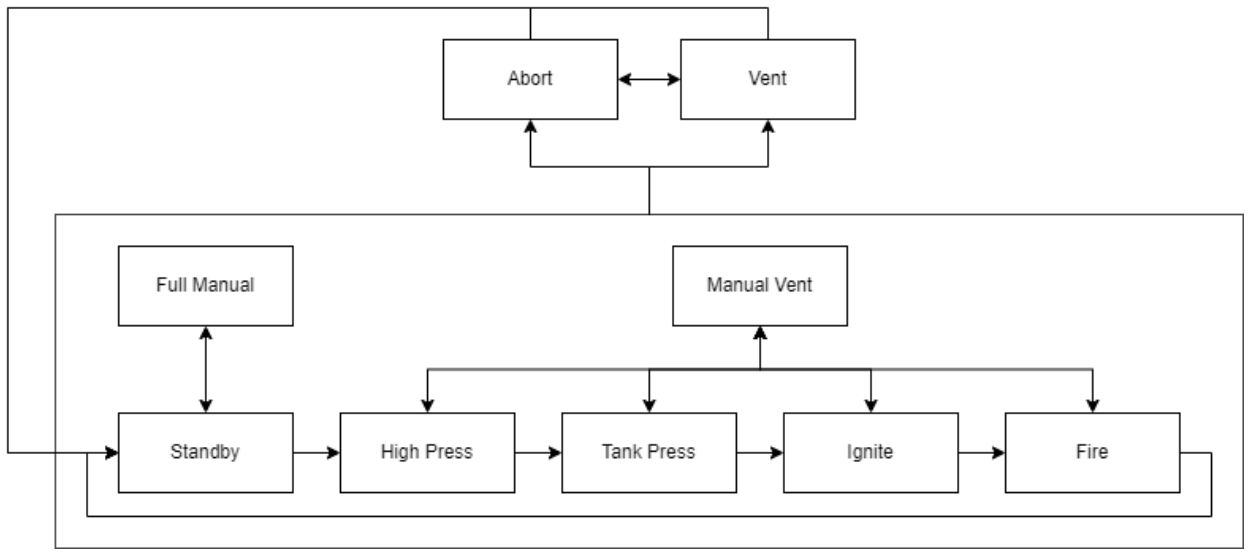
Within the test state, commands to individually actuate the valves and igniters are valid. Valve timings can be changed from any state.

Other States

Commands to individually actuate valves and igniters are invalid. Valve timings can be changed from any state. State transitions are only allowed following the transition matrix in the following section. State transitions are verified using the Rocket class `Analicia Sosa` method.

Rocket States

State:	HP	HV	LV	LMV	LDR	LDV	FV	FMV	FDR	FDV	IGN1	IGN2	LED0	LED1
Abort	X/U	X/U	X/U	X/U	X/U	X/U	X/U	X/U	X/U	X/U	X/U	X/U	GREEN	PURPLE
Vent	X/U	O/P	O/P	X/U	X/U	O/P	O/P	X/U	X/U	O/P	X/U	X/U	PURPLE	PURPLE
Fire	O/P	X/U	X/U	T/P	O/P	X/U	X/U	T/P	O/P	X/U	O/P	O/P	RED	RED
Tank Press	O/P	X/U	X/U	X/U	O/P	X/U	X/U	X/U	O/P	X/U	X/U	X/U	ORANGE	GREEN
High Press	O/P	X/U	X/U	X/U	X/U	X/U	X/U	X/U	X/U	X/U	X/U	X/U	ORANGE	BLUE
Standby	X/U	X/U	X/U	X/U	X/U	X/U	X/U	X/U	X/U	X/U	X/U	X/U	WHITE	WHITE
Ignite	O/P	X/U	X/U	X/U	O/P	X/U	X/U	X/U	O/P	X/U	T/P	T/P	ORANGE	ORANGE
Test	C/C	C/C	C/C	C/C	C/C	C/C	C/C	C/C	C/C	C/C	C/C	C/C	GREEN	GREEN



Acronyms:
HP: high-press valve
HV: high vent valve
LV: lox vent valve
LMV: lox main valve
LDR: lox dome reg valve
LDV: lox dome vent valve
FV: fuel vent valve
FMV: fuel main valve
FDR: fuel dome reg valve
FDV: fuel dome vent valve
IGN1: igniter 1
IGN2: igniter 2
KEY: (valve state/power state)
O - open
X - closed
T - timed open
C - controllable
P - powered
U - unpowered

Transition Matrix

		To							
		vent	abort	standby	test	hi press	tank press	fire	ignite
From	vent	0	1	1	1	0	0	0	0
	abort	1	0	1	0	0	0	0	0
	standby	1	1	0	1	1	0	0	0
	test	1	1	1	0	0	0	0	0
	hi press	1	1	0	0	0	1	0	0
	tank press	1	1	0	0	0	0	1	0
	fire	1	1	0	0	0	0	0	1
	ignite	1	1	1	0	0	0	0	0

Encodings

Rocket states are encoded as an enum with valid states listed as follows:

- Test
- Passive
- Standby
- High Press
- Tank Press
- Fire
- Vent
- Abort

Transitions are allowed in code following the transition matrix. Any state can transition to vent or abort, save vent to vent and abort to abort. States can otherwise only transition linearly towards fire.

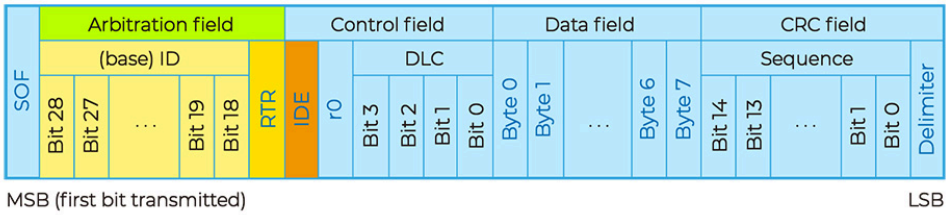
Classes

Communication

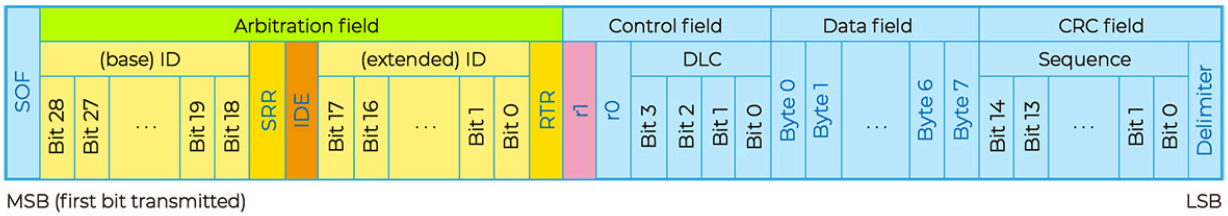
CAN Overview

clarke.patrick.r@gmail.com

Base CAN data frame format



Extemded CAN data frame format



CAN Protocol

<built on the FlexCAN library by Pawlesky> Insert link to code for it

CAN ID Table

CAN ID	Command
0	Abort
1	Vent
2	Fire
3	Tank press
4	High press
5	Standby
6	Passive
7	Test
8	IGN1 off
9	IGN1 on
10	IGN2 off
11	IGN1 on
12	HV open

13	HV close
14	HP open
15	HP close
16	LDV open
17	LDV close
18	FDV open
19	FDV close
20	LDR open
21	LDR close
22	FDR open
23	FDR close
24	LV open
25	LV close
26	FV open
27	FV close
28	LMV open
29	LMV close
30	FMV open
31	FMV close

Classes

CANDriver

Manages all interactions between the GUI and the rocket code. The CANDriver class is built off the FlexCAN library authored by Pawlesky.

readMessage

Inputs: None
Outputs: int
Behavior: Retrieves any available messages in the mailbox and returns the integer command ID associated with the highest priority command.

sendRocketState

Takes in a vehicle State. Send it to the GUI.

sendStateReport

Inputs: time, int - run time in milliseconds
Inputs: rocketState, 8 bit int - what state is the rocket in
Inputs: valves[], Valve - array of valve objects
Inputs: igniters[], Igniter - array of igniters
Inputs: Prop **,** bool - whether propulsion is active or not
Outputs: void
Behavior: Returns the valve and igniter states of one of the ALARAs

sendSensorData

Inputs: sensorID, int - The ID number of the sensor
Inputs: sensorData1, float - First voltage reading

Inputs: sensorData2, float - Second voltage reading
Inputs: sensorData3, float - Third voltage reading
Inputs: sensorData4, float - Fourth voltage reading
Outputs: void
Behavior: Calculates raw voltage

Usage

Dependencies

PlatformIO

Setup for PlatformIO:

1. Install PlatformIO extension in VS code
2. On left hand vertical menu bar click the alien icon
3. Click new project
4. In the menu click new project button, take note of which directory your project will be in
5. Name the Project, Select teensy3.6, Select Arduino Framework
6. Open a separate window of VS code
7. Clone the github repo to inside the platformIO project you just made
8. you should see the files inside of your platform IO project
9. Now you changes will be regonized in VS codes version control and the compiler will recognize the <ardruino.h> import as well as new types like interruptTimer without giving an error

Compilation and Deployment

When building and deploying, select the teensy36 framework rather than the default framework in PlatformIO.

Test Results

Peripherals Test

Click Test

Test

The peripherals test iterated through the peripherals attached to each ALARA to activate and deactivate them. The valves were actuated and the operators listened for the click each one makes as it opens and closes. HP and HV are not connected to the stand. Those were tested by watching the LED on the ALARA that turns on when a high or low signal are sent. An e-match was attached to test the igniter code. Operators could see the igniter activate and deactivate. Raw voltage readings were taken from the sensors and written to the on-board SD card.

Result

All peripherals functioned as expected. We confirmed function of all 10 valves, 2 igniters, and 14 sensors.

CAN Test

clarke.patrick.r@gmail.com

Logic Test

Test

Appendix

Abbreviations

E-match: electronic match
Lox: Liquid oxygen
PT: Pressure transducer
LC: Load cell
TC: Thermocouple
Dome reg: dome loaded regulator

Additional Resources

Hardware Resources

MCU Manual:

Previous Code Bases

RocketDriver 2.0:

RocketDriver 3.0:

Tutorials and References

Event handler

Interrupt service routines