LV3.1 Recovery Board Firmware

Updated 4/5/2020 by Ben Kolligs (benkolligs@gmail.com)

Overview

The firmware on the board is running on the STM32F0-42K6 microcontroller. It was written in C/C++ and designed with the following requirements in mind.

Requirements

- When a signal from the Telemetrum is detected, activate the necessary functions
  - Drogue
  - Main Chute
- Use a control loop to determine if the drogue has been released or not using the light sensors
  - Drive the motor based off this sensor reading
- Manage the extension/retraction of the linear actuator
  - Stop before the limit is reached
- Do the above while not pulling too much current
- PWM signals
  - Outputs
    - One timer for the motor at 25KHz
    - One timer for the speaker at 4KHz
  - Inputs
    - For the motor frequency generator (speed reading)
- ADC conversion
  - One channel for the linear actuator position
  - One channel for the battery voltage

Pin Mapping

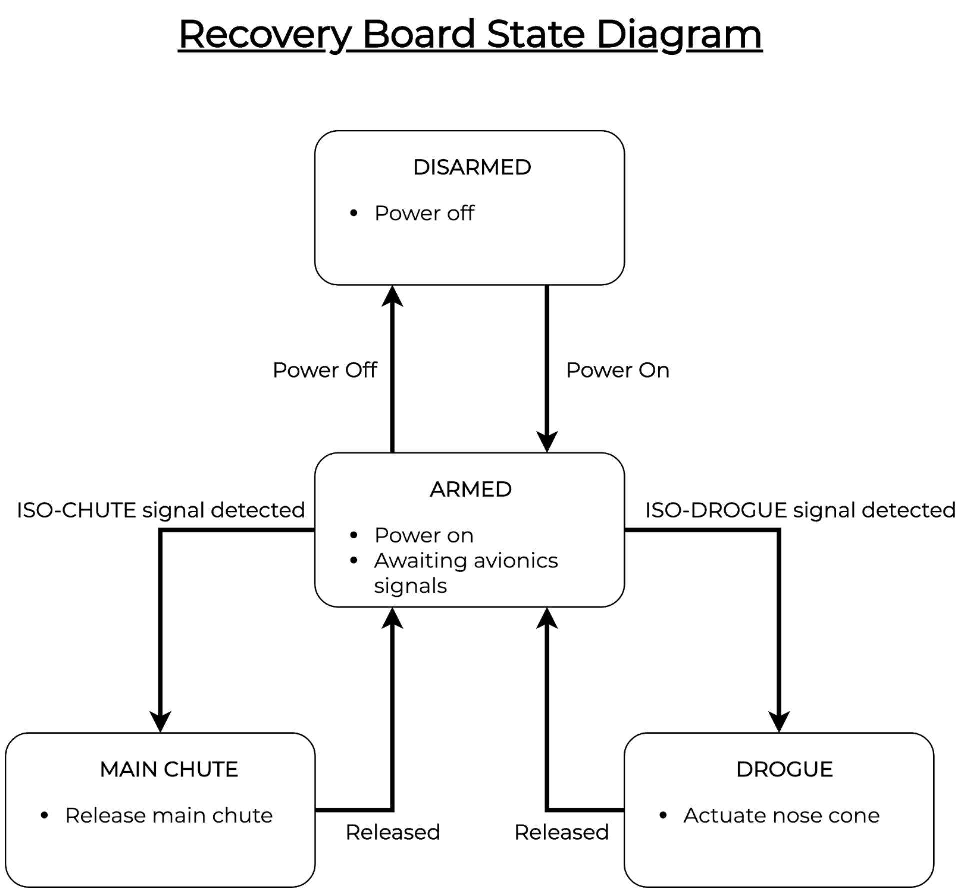
This is the same mapping used to route the PCBA, and is reflected in the recovery board schematic.

Purpose	**Pin Number**	**Schematic Name**	**STM32 Pin Name**	**Pin Mode**
DC motor speed	6	DCM_SPEED	PA0	PWM IN
Analog Input for potentiometer to read LA position	7	LA_POS	PA1	AI
USART protocol Tx	8	USART2_TX	PA2	AF
Chute Signal from OptoIsolator	9	ISO_CHUTE	PA3	DI
Drogue Signal from OptoIso	10	ISO_DROGUE	PA4	DI
Direction of DC Motor	11	DCM_DIR	PA5	DO
DC Motor PWM output	12	DCM_PWM	PA6	PWM OUT
Turn DC Motor on	13	DCM_ON	PA7	DO
Status LED	14	LED	PB0	DO
Read Battery Voltage	15	BATT_READ	PB1	AI
HBridge for LA Input 1	18	LA_IN1	PA8	DO
Hbridge Input 2	19	LA_IN2	PA9	DO
Free	20			
CAN Rx	21	N/A	PA11	CAN
CAN TX	22	N/A	PA12	CAN
SWDIO	23	SWDIO	PA13	SWDIO
SWCLK	24	SWCLK	PA14	SWCLK
USART Protocol	25	USART_RX	PA15	AF

Rx				
Turn Sensors on	26	SENSOR_ON	PB3	DO
Sensor 2	27	SENSOR2	PB4	DI
Sensor 1	28	SENSOR1	PB5	DI
Free	29			
Plugged into Umbilical?	30	ACOK	PB7	DI
Bootloader/CH1 PWM output for speaker	31	SPKR	PB8	PWM OUT

Code Overview

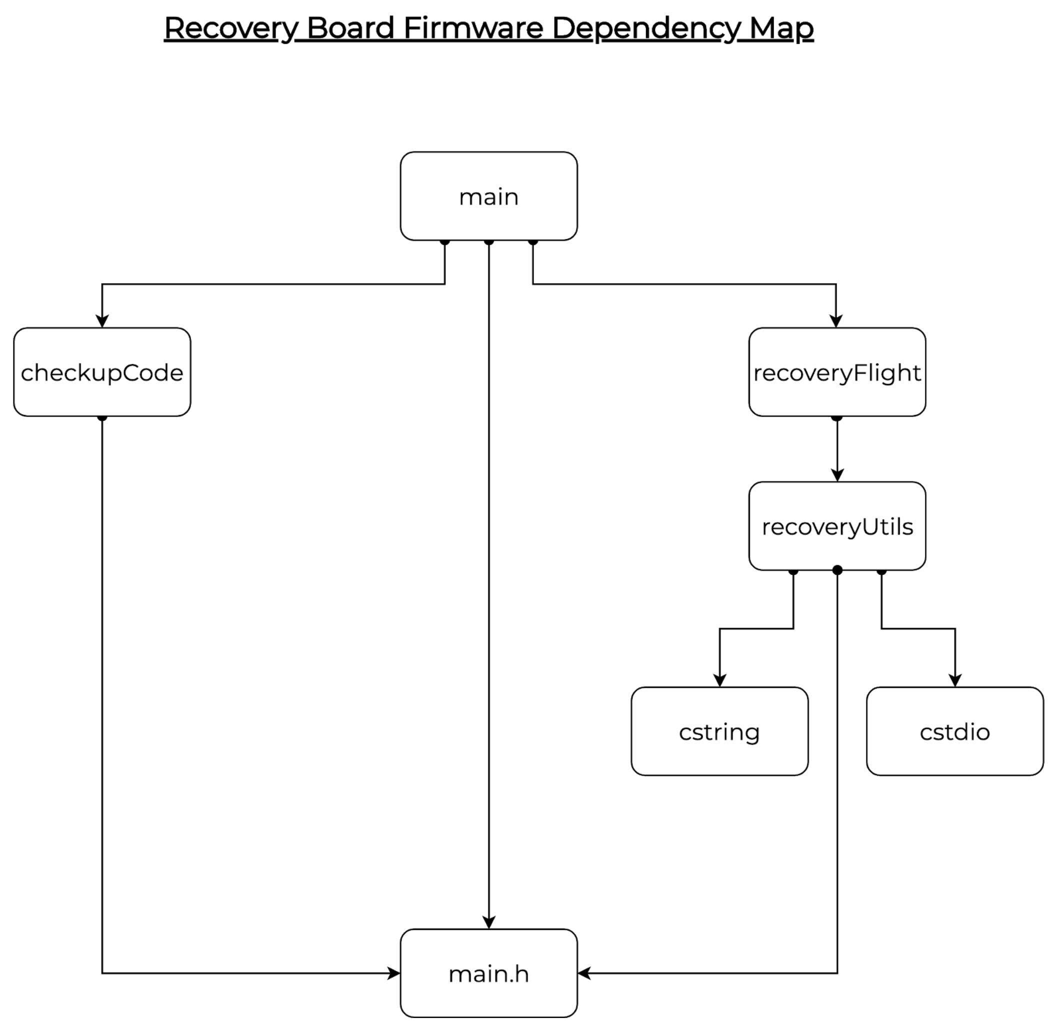
I wrote the code using an IDE called STM32CubeIDE which was designed by ST Microelectronics. The program uses a hardware abstraction layer library provided by ST, and is written in both C and C++. The IDE makes starting a project easier than if I was to write this firmware in another editor like say, visual studio code. This IDE comes with the cross-compiler and linker for the particular microcontroller I am using, and lets you select the particular board/chip you're using from ST. This state diagram details the main functions of the microcontroller.



The "FirmWare" folder of the "lv3.1-recovery/RecoveryBoard/" repository on github contains the full project I used to load the code onto the microcontroller. There are a lot of files that were generated by STM32CubeIDE in addition to the ones written for the board. I will talk about the relevant files below.

File Descriptions

To find these files go to the *core* directory in the project tree. These files interact with each other as follows:



recoveryUtils

Header file: recoveryUtils.h
Source file: recoveryUtils.cpp

This is a library containing utility functions that are used by functions in the flight software library.

recoveryFlight

Header file: recoveryFlight.h
Source file: recoveryFlight.cpp

This is a library containing the main flight software, such as the drogue release, and main chute release functions.

checkupCode

Header file: checkupCode.h
Source file: checkupCode.cpp

This is a library that contains functions that test all the aspects of the board to make sure they are connected to the MCU and function properly.

Main.cpp

The main file where the magic happens. This initializes all the peripherals through the built in HAL library, and is where the main control loop exists.

Function Map

This function map details what functions are called by what, and displays the structure of the whole program.