

# 1 Header Files

## 1.1 include/main.h

```
1  /**
2   * \file main.h
3   *
4   * Contains common definitions and header files used throughout your PROS
5   * project.
6   *
7   * Copyright (c) 2017-2020, Purdue University ACM SIGBots.
8   * All rights reserved.
9   *
10  * This Source Code Form is subject to the terms of the Mozilla Public
11  * License, v. 2.0. If a copy of the MPL was not distributed with this
12  * file, You can obtain one at http://mozilla.org/MPL/2.0/.
13  */
14
15 #ifndef _PROS_MAIN_H_
16 #define _PROS_MAIN_H_
17
18 /**
19  * If defined, some commonly used enums will have preprocessor macros which give
20  * a shorter, more convenient naming pattern. If this isn't desired, simply
21  * comment the following line out.
22  *
23  * For instance, E_CONTROLLER_MASTER has a shorter name: CONTROLLER_MASTER.
24  * E_CONTROLLER_MASTER is pedantically correct within the PROS styleguide, but
25  * not convenient for most student programmers.
26  */
27 #define PROS_USE_SIMPLE_NAMES
28
29 /**
30  * If defined, C++ literals will be available for use. All literals are in the
31  * pros::literals namespace.
32  *
33  * For instance, you can do `4_mtr = 50` to set motor 4's target velocity to 50
34  */
35 #define PROS_USE_LITERALS
36
37 #include "api.h"
38
39 /**
40  * You should add more #includes here
41  */
42 //#include "okapi/api.hpp"
43 //#include "pros/api_legacy.h"
44
45 /**
46  * If you find doing pros::Motor() to be tedious and you'd prefer just to do
47  * Motor, you can use the namespace with the following commented out line.
48  *
49  * IMPORTANT: Only the okapi or pros namespace may be used, not both
50  * concurrently! The okapi namespace will export all symbols inside the pros
```

```

51  * namespace.
52  */
53  // using namespace pros;
54  // using namespace pros::literals;
55  // using namespace okapi;
56
57  /**
58   * Prototypes for the competition control tasks are redefined here to ensure
59   * that they can be called from user code (i.e. calling autonomous from a
60   * button press in opcontrol() for testing purposes).
61   */
62  #ifdef __cplusplus
63  extern "C" {
64  #endif
65  void autonomous(void);
66  void initialize(void);
67  void disabled(void);
68  void competition_initialize(void);
69  void opcontrol(void);
70  #ifdef __cplusplus
71  }
72  #endif
73
74  #ifdef __cplusplus
75  /**
76   * You can add C++-only headers here
77   */
78  // #include <iostream>
79  #endif
80
81  #endif

```

## 2 Source Files

### 2.1 src/main.cpp

```
1  #include "main.h"
2
3  /**
4   * A callback function for LLEMU's center button.
5   *
6   * When this callback is fired, it will toggle line 2 of the LCD text between
7   * "I was pressed!" and nothing.
8   */
9  void on_center_button() {
10     static bool pressed = false;
11     pressed = !pressed;
12     if (pressed) {
13         pros::lcd::set_text(2, "I was pressed!");
14     } else {
15         pros::lcd::clear_line(2);
16     }
17 }
18
19 /**
20  * Runs initialization code. This occurs as soon as the program is started.
21  *
22  * All other competition modes are blocked by initialize; it is recommended
23  * to keep execution time for this mode under a few seconds.
24  */
25 void initialize() {
26     pros::lcd::initialize();
27     pros::lcd::set_text(1, "Hello PROS User!");
28
29     pros::lcd::register_btn1_cb(on_center_button);
30 }
31
32 /**
33  * Runs while the robot is in the disabled state of Field Management System or
34  * the VEX Competition Switch, following either autonomous or opcontrol. When
35  * the robot is enabled, this task will exit.
36  */
37 void disabled() {}
38
39 /**
40  * Runs after initialize(), and before autonomous when connected to the Field
41  * Management System or the VEX Competition Switch. This is intended for
42  * competition-specific initialization routines, such as an autonomous selector
43  * on the LCD.
44  *
45  * This task will exit when the robot is enabled and autonomous or opcontrol
46  * starts.
47  */
48 void competition_initialize() {}
49
50 /**
```

```

51  * Runs the user autonomous code. This function will be started in its own task
52  * with the default priority and stack size whenever the robot is enabled via
53  * the Field Management System or the VEX Competition Switch in the autonomous
54  * mode. Alternatively, this function may be called in initialize or opcontrol
55  * for non-competition testing purposes.
56  *
57  * If the robot is disabled or communications is lost, the autonomous task
58  * will be stopped. Re-enabling the robot will restart the task, not re-start it
59  * from where it left off.
60  */
61  void autonomous() {}
62
63  /**
64  * Runs the operator control code. This function will be started in its own task
65  * with the default priority and stack size whenever the robot is enabled via
66  * the Field Management System or the VEX Competition Switch in the operator
67  * control mode.
68  *
69  * If no competition control is connected, this function will run immediately
70  * following initialize().
71  *
72  * If the robot is disabled or communications is lost, the
73  * operator control task will be stopped. Re-enabling the robot will restart the
74  * task, not resume it from where it left off.
75  */
76  void opcontrol() {
77      pros::Controller master(pros::E_CONTROLLER_MASTER);
78      pros::Motor left_mtr(1);
79      pros::Motor right_mtr(2);
80
81      // LONG LINE // LONG LINE// LONG LINE// LONG LINE// LONG LINE// LONG LINE// LONG
82      ↪ LINE// LONG LINE// LONG LINE// LONG LINE// LONG LINE// LONG LINE// LONG LINE//
83      ↪ LONG LINE// LONG LINE// LONG LINE// LONG LINE// LONG LINE// LONG LINE// LONG LINE
84      while (true) {
85          pros::lcd::print(0, "%d %d %d", (pros::lcd::read_buttons() & LCD_BTN_LEFT) >> 2,
86              (pros::lcd::read_buttons() & LCD_BTN_CENTER) >> 1,
87              (pros::lcd::read_buttons() & LCD_BTN_RIGHT) >> 0);
88
89          int left = master.get_analog(ANALOG_LEFT_Y);
90          int right = master.get_analog(ANALOG_RIGHT_Y);
91
92          left_mtr = left;
93          right_mtr = right;
94          pros::delay(20);
95      }
96  }

```